

INSTITUTO TECNOLÓGICO DE AERONÁUTICA



Raíssa Brasil Andrade

**DESENVOLVIMENTO DE UM SIMULADOR
RADAR UTILIZANDO SOFTWARE ABERTO DE
MODELAGEM E RENDERIZAÇÃO 3D**

Trabalho de Graduação
2020

Curso de Engenharia Eletrônica

Raíssa Brasil Andrade

**DESENVOLVIMENTO DE UM SIMULADOR
RADAR UTILIZANDO SOFTWARE ABERTO DE
MODELAGEM E RENDERIZAÇÃO 3D**

Orientador

Prof. Dr. Renato Machado (ITA)

ENGENHARIA ELETRÔNICA

**SÃO JOSÉ DOS CAMPOS
INSTITUTO TECNOLÓGICO DE AERONÁUTICA**

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Andrade, Raíssa Brasil

Desenvolvimento de um simulador radar utilizando software aberto de modelagem e renderização 3D / Raíssa Brasil Andrade.

São José dos Campos, 2020.

75f.

Trabalho de Graduação – Curso de Engenharia Eletrônica– Instituto Tecnológico de Aeronáutica, 2020. Orientador: Prof. Dr. Renato Machado.

1. Radar. 2. Simuladores. 3. Rastreamento (posição). I. Instituto Tecnológico de Aeronáutica. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

ANDRADE, Raíssa Brasil. **Desenvolvimento de um simulador radar utilizando software aberto de modelagem e renderização 3D**. 2020. 75f. Trabalho de Conclusão de Curso (Graduação) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DA AUTORA: Raíssa Brasil Andrade

TÍTULO DO TRABALHO: Desenvolvimento de um simulador radar utilizando software aberto de modelagem e renderização 3D.

TIPO DO TRABALHO/ANO: Trabalho de Conclusão de Curso (Graduação) / 2020

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias deste trabalho de graduação e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. A autora reserva outros direitos de publicação e nenhuma parte deste trabalho de graduação pode ser reproduzida sem a autorização da autora.

Raíssa Brasil Andrade

Raíssa Brasil Andrade

Av. Lisboa, 50

12.216-630 – São José dos Campos–SP

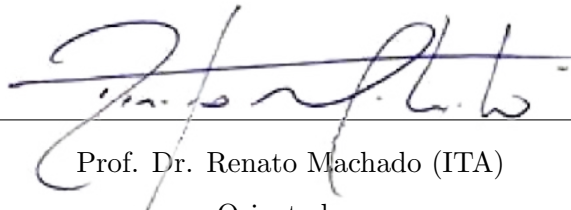
DESENVOLVIMENTO DE UM SIMULADOR RADAR UTILIZANDO SOFTWARE ABERTO DE MODELAGEM E RENDERIZAÇÃO 3D

Essa publicação foi aceita como Relatório Final de Trabalho de Graduação



Raíssa Brasil Andrade

Autora



Prof. Dr. Renato Machado (ITA)

Orientador



Prof. Dr. Marcelo da Silva Pinho
Coordenador do Curso de Engenharia Eletrônica

São José dos Campos, 20 de novembro de 2020.

Dedico este trabalho ao meu pai (*in memoriam*), à minha mãe, à minha avó, à minha irmã e ao meu noivo.

Agradecimentos

Agradeço à minha família, pelo apoio e exemplo dados ao longo da minha vida. À minha mãe, por sua compreensão, força e carinho, à minha avó por sua paciência e cuidado, e ao meu pai, que mesmo não estando mais aqui, deixou-me como legado sua bondade, dedicação e autenticidade. Agradeço também ao meu noivo, que durante nossa trajetória me fazer sorrir todos os dias, foi companheiro e acreditou em mim, além de ter colaborado com ideias que fizeram a diferença neste trabalho. Também sou grata a Deus pela benção de ter pessoas tão boas em minha vida.

Ao meu orientador, Professor Renato Machado, que mesmo em meio às suas atribuições, sempre foi prestativo e paciente, e cuja dedicação foi fator essencial para a execução deste trabalho. Sou muito grata por sua orientação e por sempre se mostrar uma pessoa tranquila, justa e correta.

Agradeço também ao aluno de doutorado Rômulo Fernandes da Costa, que foi muito prestativo, didático e humilde ao compartilhar seu conhecimento comigo. Assim como o Professor Renato, ele também foi fundamental para a conclusão deste trabalho. Pelo mesmo motivo, agradeço ao também doutorando Ten Cel Gustavo Farhat, que pacientemente dedicou parte de seu tempo para colaborar com este trabalho.

À Força Aérea Brasileira por acreditar nos alunos do ITA e oferecer o suporte necessário para que pudéssemos nos formar neste centro de excelência.

E a todos os outros que fizeram parte da minha trajetória até aqui, também registro meu agradecimento.

*“One’s life has value so long as one attributes value to the life of others,
by means of love, friendship, and compassion.”*

— SIMONE DE BEAUVOIR

Resumo

Radars são fundamentais para atividades de defesa e segurança que envolvem reconhecimento e monitoramento de alvos. Por envolver a detecção de sinais refletidos pelo ambiente, é necessário estabelecer métodos capazes de diferenciar os alvos de interesse daqueles que não trazem informação relevante para o sistema, e para isso o conhecimento prévio do comportamento desses alvos é de grande importância. Nesse contexto, simulações computacionais são uma forma eficiente e segura de coletar dados úteis para este fim. O presente trabalho propõe o desenvolvimento e a validação de um simulador radar através da construção de cenários e obtenção de dados no Blender, um *software open-source* de modelagem 3D, e posterior tratamento desses dados em MATLAB. O ambiente em Blender conta com um emissor de luz e uma câmera, de forma que a renderização carrega informações sobre a interação da luz com o cenário através do Cycles, que é o *path-tracer* interno do *software*. No simulador apresentado, os elementos da cena têm seu comportamento manipulado para que as informações relevantes para aplicação radar sejam incluídas nas componentes RGB dos pixels da imagem renderizada pela câmera, que atua como um receptor. O emissor de luz, por sua vez, é tratado como um transmissor. Finalmente, o tratamento adequado dos *frames* resultantes da renderização através da aplicação de conceitos de processamento de sinais e teoria de radar permite analisar o sinal reconstruído para diferentes cenários 3D e confrontá-lo com o que é previsto pela teoria e literatura da área.

Abstract

Radars are essential in activities related to defense and security through the recognition and monitoring of targets. Since it involves the detection of signals reflected in the environment, it is necessary to establish methods that allow the distinction between targets of interest from those that do not carry relevant information for the system. In such a context, computational simulations are an efficient and secure way of collecting data for these purposes. The present work proposes the design and validation of a radar simulator by means of building a scene and collecting data from Blender, a 3D modeling open-source software, and later treatment of this data in MATLAB. A generic Blender scene has a light emitter and a camera, and its rendering carries information about the light interactions with its elements through Cycles, the software's internal path-tracer. For the simulator, the scene's elements have their behavior altered so that relevant information for radar applications is included in the RGB channels of the image rendered by the camera, which acts as a receptor. The light emitter, on the other hand, acts as a transmitter. Finally, the adequate treatment of the rendered frames, made possible through the application of data processing and radar systems concepts, allows for the reconstructed signal to be analyzed for different 3D scenes and compared to what is expected from theory and the area's literature.

Lista de Figuras

FIGURA 1.1 – Radar SABER M60. Os parâmetros de projeto do SABER M60 foram considerados na realização de grande parte simulações deste trabalho. Fonte: (SILVA et al., 2014).	18
FIGURA 2.1 – Ilustração do <i>path tracing</i> para um número de <i>bounces</i> igual 0 (a), 1 (b) e 2 (c). Adaptado de (Blender Manual, 2018c).	22
FIGURA 2.2 – Exemplificação da lógica adotada para análise do caminho do luz após sofrer 2 <i>bounces</i> nas superfícies do cenário.	23
FIGURA 2.3 – Iluminação gradual da cena a cada <i>frame</i> para permitir a identificação de alvos obstruídos.	24
FIGURA 2.4 – Interface para a entrada dos parâmetros das superfícies dos objetos Blender. A partir desses dados o <i>shader</i> implementado regula a interação da luz com os elementos da cena.	25
FIGURA 2.5 – Primeira parte da implementação do <i>shader</i> usado para a simulação de materiais.	26
FIGURA 2.6 – Segunda parte da implementação do <i>shader</i> usado para a simulação de materiais.	27
FIGURA 2.7 – Terceira parte da implementação do <i>shader</i> usado para a simulação de materiais.	27
FIGURA 2.8 – Matrizes com a distância percorrida (a) e amplitude (b) associada aos pixels de um determinado <i>frame</i> . A distancia percorrida é obtida a partir da componente vermelha, u_R , através de 2.4, e amplitude é a própria componente azul, u_B	28
FIGURA 3.1 – Cena montada no Blender para verificação do decaimento da amplitude do sinal.	31

FIGURA 3.2 – Resultados obtidos com o simulador para o decaimento da amplitude com a distância e curva ajustada aos pontos.	31
FIGURA 4.1 – Pulso não modulado no domínio do tempo.	34
FIGURA 4.2 – Espectro do pulso não modulado.	34
FIGURA 4.3 – Partes real (a) e imaginária (b) do sinal descrito por (4.6).	35
FIGURA 4.4 – Comportamento das integrais de Fresnel definidas em (4.13) e (4.14).	36
FIGURA 4.5 – Comportamento de x_1 e x_2 quando $B_M T_M \rightarrow \infty$	36
FIGURA 4.6 – Comparação entre o espectro de $s_M(t)$ dado por (4.16) e (4.18).	37
FIGURA 4.7 – Cena usada para verificação da resolução em <i>range</i>	40
FIGURA 4.8 – Amplitude do sinal recebido pelo radar para os diferentes valores de d_2	40
FIGURA 4.9 – Sinal associado a cada alvo para os diferentes valores de d_2	41
FIGURA 4.10 – Sinal associado a cada alvo para os diferentes valores de d_2 após o uso da janela de Kaiser.	41
FIGURA 4.11 – Amplitude do sinal recebido pelo radar para os diferentes valores de d_2 após o uso da janela de Kaiser.	42
FIGURA 5.1 – Esquema de mapeamento das frequências Doppler em N_d intervalos uniformes.	46
FIGURA 5.2 – Diagrama da implementação proposta para identificação da frequência Doppler do sinal recebido pelo radar.	46
FIGURA 5.3 – Helicóptero bipá usado na simulação (acima) e os três últimos <i>frames</i> renderizados (abaixo). Modelo retirado de (Free 3D, 2019b).	47
FIGURA 5.4 – Helicóptero quadripá usado na simulação (acima) e os três últimos <i>frames</i> renderizados (abaixo). Modelo adaptado de (Free 3D, 2019b).	48
FIGURA 5.5 – Frequência Doppler percebida pelo radar.	48
FIGURA 5.6 – Amplitude do sinal recebido pelo radar.	49
FIGURA 5.7 – Cena montada em Blender para verificar a detecção de múltiplos alvos em movimento. Fonte dos modelos 3D dos alvos: (Free 3D, 2019a), (Free 3D, 2019b) e (TurboSquid, 2017).	50
FIGURA 5.8 – Resultado obtido para a simulação da cena da Figura 5.7 após análise em MATLAB.	51

FIGURA 5.9 – Efeito da supressão de <i>clutter</i> no sinal recebido.	52
FIGURA 6.1 – Ilustração da definição intuitiva de RCS. Adaptado de (KNOTT et al., 2004).	54
FIGURA 6.2 – RCS de uma esfera de raio r_e em função da razão entre sua circunferência e o comprimento de onda. Adaptado de (SKOLNIK, 2001).	56
FIGURA 6.3 – Referência para os valores de θ e das dimensões da equação (6.3).	57
FIGURA 6.4 – RCS do elipsoide em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,4.	58
FIGURA 6.5 – Referência para os valores de θ e das dimensões da equação (6.4).	59
FIGURA 6.6 – Referência para os valores de θ e das dimensões da equação (6.5).	59
FIGURA 6.7 – RCS do cilindro em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,3.	59
FIGURA 6.8 – RCS do tronco de cone em função do ângulo polar. O resultado apresentado na curva vermelha (Blender 1) corresponde a uma rugosidade especular igual a 0,5, enquanto que a curva azul (Blender 2) foi obtida para uma rugosidade especular de 0,2.	60
FIGURA 6.9 – Referência para as dimensões e os valores de θ da equação (6.6) e da Figura 6.12.	61
FIGURA 6.10 – Referência para as dimensões e os valores de θ da equação (6.7) e da Figura 6.13.	61
FIGURA 6.11 – Referência para as dimensões e os valores de θ das equações (6.10) e (6.11), e das Figuras 6.14 e 6.15.	62
FIGURA 6.12 – RCS da placa retangular em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,6.	63
FIGURA 6.13 – RCS da placa circular em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,2.	63
FIGURA 6.14 – RCS da placa triangular em função do ângulo polar para um ângulo azimutal de 0° . O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,3.	64

FIGURA 6.15 –RCS da placa triangular em função do ângulo polar para um ângulo azimutal de 90° . O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,3.	64
FIGURA 6.16 –Referência para as dimensões e os valores de φ da equação (6.12) e da Figura 6.17.	65
FIGURA 6.17 –RCS do diedro em função do ângulo azimutal para um ângulo polar de 90°	66
FIGURA 6.18 –Caça <i>stealth</i> F-22 Raptor real (a) e seu modelo no Blender (b). Fontes: (ROSSO, 2013), (TurboSquid, 2015).	67
FIGURA 6.19 –Bombardeiro <i>stealth</i> B-2 Spirit real (a) e seu modelo no Blender (b). Fontes: (Davis III, 2006), (Free 3D, 2020).	68
FIGURA 6.20 –Avião comercial A310 real (a) e seu modelo no Blender (b). Fontes: (WEDELSTAEDT, 2009), (Free 3D, 2019a).	68
FIGURA 6.21 –Resultado obtido para a RCS do F-22 Raptor, do B-2 Spirit e do A310.	69

Lista de Tabelas

TABELA 4.1 – Parâmetros do radar SABER M60. Fonte: (CARVALHO et al., 2008). 39

TABELA 5.1 – Dados adotados para a simulação da cena mostrada na Figura 5.7.
Os valores apresentados têm como referência o sistema de eixos co-ordenados (x, y, z) mostrado na mesma imagem. 50

Lista de Abreviaturas e Siglas

BSDF	<i>Bidirectional Scattering Distribution Function</i>
CTEx	Centro Tecnológico do Exército
DFT	<i>Discrete Fourier Transform</i>
GO	<i>Geometrical Optics</i>
GTD	<i>Geometrical Theory of Diffraction</i>
HDR	<i>High Dynamic Range</i>
LIDAR	<i>Light Detection and Ranging</i>
LFM	<i>Linear Frequency Modulation</i>
MEC	<i>Method of Equivalent Currents</i>
PO	<i>Physical Optics</i>
PTD	<i>Physical Theory of Diffraction</i>
RCS	<i>Radar Cross Section</i>
RGB	<i>Red Green Blue</i>
SABER	Sistema de Acompanhamento de alvos aéreos Baseados em Emissão de Radiofrequência
UHF	<i>Ultra High Frequency</i>

Sumário

1	INTRODUÇÃO	17
1.1	Breve histórico	17
1.2	Contextualização e motivação	18
1.3	Objetivos	19
1.4	Organização do trabalho	19
2	INFORMAÇÕES SOBRE O BLENDER	21
2.1	O algoritmo de <i>path-tracing</i>	21
2.2	Codificação de distância nas componentes RGB	23
2.3	Iluminação gradual da cena	24
2.4	Implementação do <i>shader</i> para configuração de materiais	25
2.5	<i>Frames</i> de saída	28
3	EQUAÇÃO DO RADAR	30
4	COMPRESSÃO DE PULSOS	33
4.1	Implementação em MATLAB	37
4.2	Verificação da resolução em <i>range</i> com dois alvos simples	39
5	FREQUÊNCIA DOPPLER	44
5.1	Detecção do efeito Doppler no simulador	44
5.2	Identificação de aeronave de asas rotativas	47
5.3	Detecção de múltiplos alvos em movimento	49
6	RADAR CROSS SECTION	53
6.1	Geometrias Elementares	55

6.1.1	Esfera	56
6.1.2	Elipsoide	57
6.1.3	Cilindro e tronco de cone	58
6.1.4	Planos retangular, circular e triangular	60
6.1.5	Diedro	65
6.2	Comparação da RCS de duas aeronaves	67
7	CONCLUSÃO	71
7.1	Trabalhos futuros	72
	REFERÊNCIAS	73

1 Introdução

Conforme sugere a palavra RADAR, que vem do termo em inglês *RAdio Detection And Ranging*, radares são equipamentos utilizados para detectar e adquirir informações sobre alvos através do processamento adequado do sinal eco, que é aquele recebido pela antena receptora do radar, sendo o resultado da interação do sinal transmitido pela antena transmissora com o ambiente dentro da área de alcance do equipamento. Ao longo dos anos, houve um aumento da quantidade e variedade de informações que podem ser obtidas por um sistema radar, o que também traz um aumento de complexidade que, muitas vezes, exige o emprego de técnicas computacionais mais sofisticadas.

1.1 Breve histórico

Em 1895, o físico escocês James Clerk Maxwell publicou sua Teoria do Campo Eletromagnético, que foi verificada experimentalmente cerca de 20 anos depois pelo alemão Heinrich Rudolf Hertz e serviu de base para a análise de fenômenos eletromagnéticos que regem o comportamento das ondas empregadas em radares e outras tecnologias. Em 1900, o também alemão Christian Hülsmeier desenvolveu um equipamento capaz de detectar embarcações através do eco refletido por elas, aplicando assim o princípio que fundamenta o que se conhece hoje como radar.

Apesar do sucesso momentâneo na época, a invenção recebeu pouca atenção nos anos seguintes, e a ideia de se usar o sinal refletido por um alvo para identificá-lo ganhou o interesse de várias nações apenas nos anos de 1920, após a Primeira Guerra Mundial. Nos anos subsequentes, Estados Unidos, Reino Unido, Alemanha, União Soviética, França, Itália, Japão e Países Baixos desenvolveram seus sistemas radar independentemente e concomitantemente (SKOLNIK, 2001).

Esses avanços foram ainda mais fomentados com a deflagração da Segunda Guerra Mundial, e envolviam tecnologias que permitiam a detecção de aeronaves e embarcações. Mesmo com o fim da guerra, o interesse por essa área não cessou, e desde então as aplicações de radares vêm sendo exploradas para fins que vão além da mera detecção de alvos.

Atualmente, além das aplicações tipicamente militares, radares são amplamente utilizados em sensoriamento remoto, controle de tráfego aéreo, controle de velocidade em rodovias, acionamento de sistemas de vigilância e segurança, guiamento de embarcações e aeronaves, detecção de satélites e exploração de óleo e gás, dentre outras aplicações (SKOLNIK, 2001).

1.2 Contextualização e motivação

Os radares são de interesse estratégico para o setor de defesa de vários países, inclusive do Brasil. De fato, a Estratégia Nacional de Defesa cita a necessidade de “desenvolver as capacidades de monitorar e controlar o espaço aéreo, o território e as águas jurisdicionais brasileiras”, e garantir a “alocação de recursos [...] para o desenvolvimento e a fabricação de radares”, dentre outras tecnologias (Ministério da Defesa, 2008).

Um dos sistemas radares mais recentemente desenvolvido no Brasil é o sistema SABER M60, mostrado na Figura 1.1. Esse sistema é empregado em busca e vigilância aérea, e foi o primeiro radar feito com tecnologia totalmente nacional pelo CTEEx, no ano de 2006 (SILVA et al., 2014). Tendo em vista a importância operacional dessa tecnologia, grande parte das simulações realizadas neste trabalho consideraram parâmetros semelhantes aos utilizados pelo sistema SABER M60.



FIGURA 1.1 – Radar SABER M60. Os parâmetros de projeto do SABER M60 foram considerados na realização de grande parte simulações deste trabalho. Fonte: (SILVA et al., 2014).

1.3 Objetivos

O desenvolvimento de um simulador gratuito para radares é de grande interesse do meio acadêmico e militar, por permitir reproduzir de maneira controlada uma variedade de alvos e cenários. A partir dos dados obtidos em simulação, é possível testar algoritmos sem a necessidade do equipamento físico (OUZA et al., 2017), e assim desenvolver métodos que possibilitem a distinção entre alvos de interesse e sinais eco que não são relevantes para a aplicação em questão. Destaca-se também a possibilidade de uso do simulador para finalidades didáticas, como a demonstração de propriedades elementares de certos fenômenos eletromagnéticos.

Como o ambiente é efetivamente simulado em um *software open-source*, o acesso irrestrito e gratuito ao funcionamento do programa e, portanto, à lógica usada na implementação, é garantido e não gera nenhum tipo de custo.

Com isso em mente, o objetivo deste trabalho de graduação é desenvolver e validar um simulador radar baseado em Blender, um *software open-source* de modelagem 3D. A análise dos dados obtidos no Blender é realizada em MATLAB e leva em conta conceitos da teoria de radares e de processamento de sinais, permitindo a validação do simulador proposto através da comparação entre o comportamento obtido e aquele previsto pela teoria.

Para que isso seja feito, torna-se necessário entender o funcionamento do algoritmo de renderização do Blender e manipular os dados de saída que ele fornece a fim de obter as informações pertinentes. Também é preciso compreender como grandezas relacionadas à composição e à geometria dos objetos influenciam nos dados obtidos, no comportamento esperado e nos parâmetros adotados para o pós-processamento em MATLAB.

Espera-se que o trabalho desenvolvido seja capaz de simular adequadamente cenários realistas e, com isso, fornecer uma alternativa gratuita para testes de sistemas que envolvam radares e suas tecnologias. Em especial, deseja-se oferecer uma plataforma de baixo custo que possibilite o desenvolvimento de tecnologias que contribuam com a pesquisa, o ensino e a defesa nacional.

1.4 Organização do trabalho

O trabalho foi dividido em capítulos que abordam aspectos distintos do simulador proposto. Em cada um deles, são expostos os fundamentos teóricos e os resultados pertinentes a cada tema. No Capítulo 2 são apresentadas informações sobre o Blender e características da implementação e dos dados de saída do programa. O Capítulo 3 exibe e valida a Equação do Radar. No Capítulo 4, a técnica de compressão de pulsos é ex-

plicada e verificada. A frequência Doppler é tema da explicação e validação do Capítulo 5, que emprega cenários mais operacionais e complexos que os anteriores para a análise. O Capítulo 6 avalia as assinaturas radar obtidas com o simulador para alvos simples e compara os resultados com aqueles fornecidos por outro *software*, além de também analisar a assinatura radar de uma aeronave. Finalmente, o Capítulo 7 conclui o trabalho e apresenta algumas possibilidades de trabalhos futuros.

2 Informações sobre o Blender

2.1 O algoritmo de *path-tracing*

O Blender é um *software open source* de modelagem 3D muito utilizado na criação de jogos, animações e cenários de realidade aumentada. A fim de tornar os cenários construídos mais realistas, o Blender e outros *softwares* similares empregam uma técnica de renderização conhecida como *path-tracing*, que utiliza modelos estatísticos para simular o comportamento dos raios de luz que atingem a câmera após sofrerem sucessivas reflexões e transmissões partindo de uma fonte luminosa. Nesse procedimento, o caminho da luz é percorrido no sentido contrário da propagação, ou seja, inicia na câmera e termina no emissor.

De forma simplificada, a imagem renderizada é consequência das iluminações direta e indireta no cenário montado. A iluminação direta corresponde à contribuição dos raios de luz que refletem apenas uma vez antes de atingir a câmera, enquanto os que refletem múltiplas vezes influenciam na iluminação indireta. Para cada pixel renderizado, um número muito grande de raios (chamados *camera rays*) é gerado a partir da câmera em várias direções. Quando o raio intersecciona uma superfície e interage com ela, diz-se que ele sofreu um *bounce*. A partir do ponto em que isso ocorre, é traçado um raio denominado *shadow ray* até a fonte de luz para o cálculo da iluminação direta no ponto de intersecção. Também é gerado outro raio, cuja direção é determinada estocasticamente a partir das propriedades da superfície, para o cálculo da iluminação indireta. Este raio, por sua vez, passa pelo mesmo processo: geração de raios que permitam computar as iluminações direta e indireta nos pontos de intersecção. O processo recursivo acaba quando o raio encontra o emissor, ou é rebatido para fora do cenário, ou ainda quando o número máximo de *bounces* é atingido (CHRISTENSEN; JAROSZ, 2016).

O *path tracing* surgiu da técnica de *ray tracing*, na qual cada *bounce* da origem a um número muito grande de raios para cálculo da iluminação indireta. O primeiro, apesar de gerar vários raios para cada pixel e usar a média deles para renderização, adota modelos estatísticos para determinar a direção do raio usado para cálculo da iluminação indireta, de forma que cada *bounce* gera um número reduzido de raios. O Blender im-

plementa o *path tracing* internamente através do *Cycles*, que utiliza funções BSDF para mapear a probabilidade de um raio de luz ser refletido e transmitido com um certo ângulo (CHARGIN, 2013) e, assim, determinar a direção do raio usado para cálculo da iluminação indireta após cada *bounce*. Dessa forma, um raio que não é *camera ray* ou *shadow ray* é necessariamente um *reflection ray* ou *transmission ray* (Blender Manual, 2018c). A Figura 2.1 ilustra o algoritmo de *path-tracing* descrito para um número máximo de *bounces* igual a 2, e evidencia a classificação dos raios traçados.

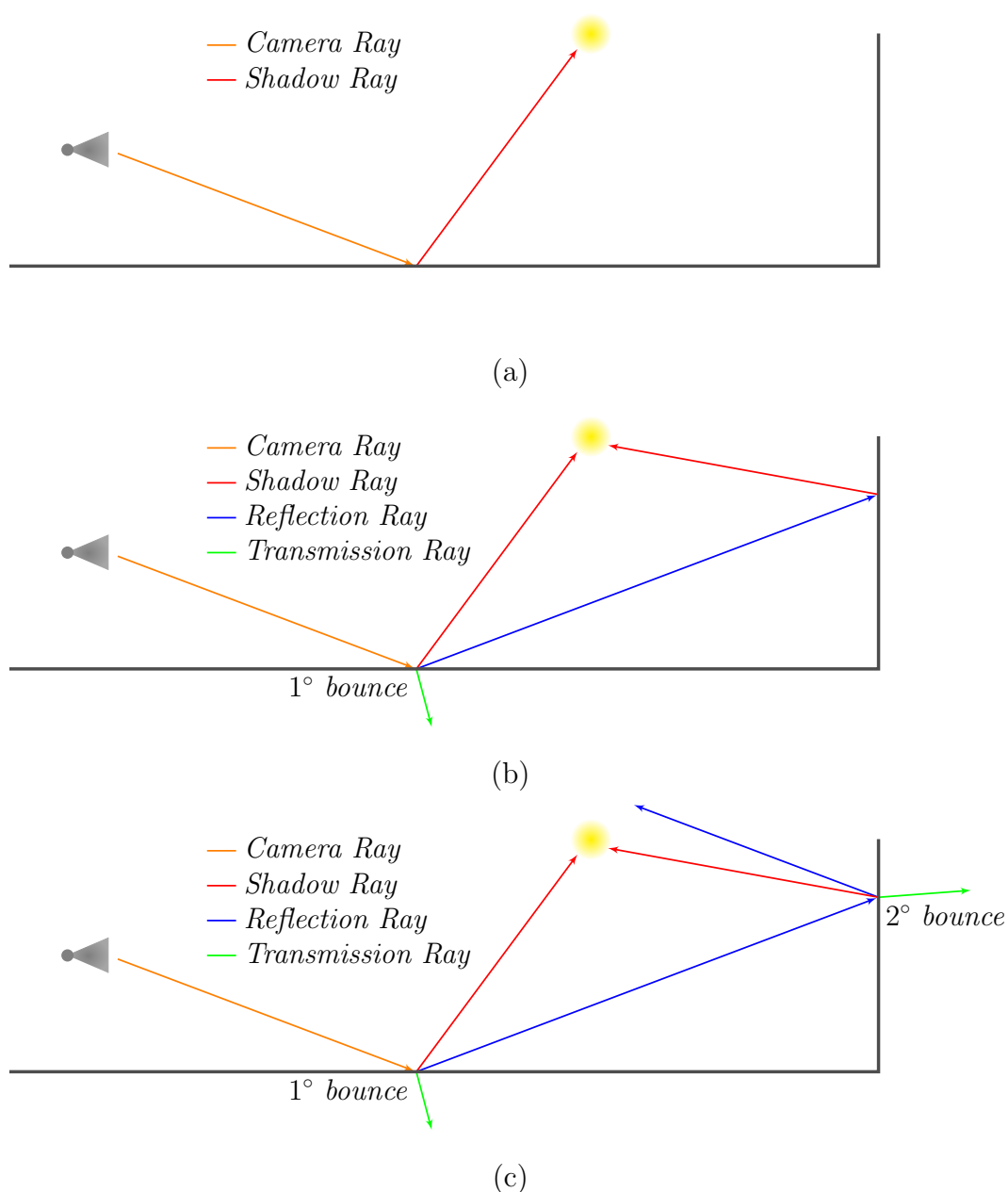


FIGURA 2.1 – Ilustração do *path tracing* para um número de *bounces* igual 0 (a), 1 (b) e 2 (c). Adaptado de (Blender Manual, 2018c).

O *path tracing* pode ser visto como uma particularização do *ray tracing*, e ambos se

baseiam em traçar raios de trajetória retilínea partir da câmera e, de maneira recursiva, fazer com que a interação desses raios com os objetos do cenário gere novos raios até que eles atinjam o emissor.

2.2 Codificação de distância nas componentes RGB

No simulador proposto, a fonte de luz do programa funciona como uma antena transmissora, e a câmera como uma antena receptora. A reconstrução do sinal é feita através da reprogramação das componentes RGB dos pixels renderizados para carregarem informações sobre a intensidade de seu respectivo raio e a distância percorrida por ele. Os *frames* renderizados são salvos em formato HDR e lidos através da função nativa do MATLAB `hdrread`, que permite extrair as informações dos canais RGB. Para um pixel cujas componentes são dadas por:

$$\underline{u} = \begin{bmatrix} u_R & u_G & u_B \end{bmatrix} \quad (2.1)$$

e que foi renderizado a partir de um raio que sofreu N_b *bounces*, sabe-se que a superfície de cada *bounce* está associada a um valor b_k , $k = 1, 2, \dots, N_b$, relacionado à sua geometria e composição. Entre o $(k - 1)$ -ésimo e o k -ésimo *bounce*, o raio percorre uma distância r_k , de forma que r_0 é a distância percorrida entre a fonte luminosa e a primeira superfície interseccionada, e r_{N_b} é a distância percorrida entre a última superfície interseccionada e a câmera. Note que agora o caminho da luz é percorrido em seu sentido convencional, partindo do emissor e terminando na câmera, conforme o esquema da Figura 2.2, pois analisa-se apenas os raios que efetivamente chegaram à fonte luminosa durante o *path-tracing*.

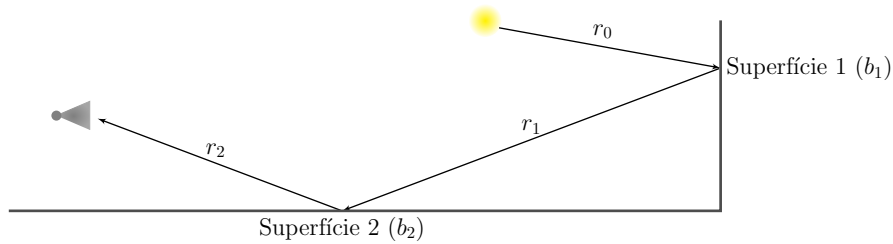


FIGURA 2.2 – Exemplificação da lógica adotada para análise do caminho do luz após sofrer 2 *bounces* nas superfícies do cenário.

É necessário escolher uma das componentes para servir de referência para o cálculo da distância percorrida pelo raio (i.e., $\sum r_k$). Arbitrando u_B , tem-se que:

$$u_B = u_0 \frac{1}{r_0} \frac{b_1}{r_1} \frac{b_2}{r_2} \dots \frac{b_{N_b}}{r_{N_b}} = u_0 \frac{\prod_{k=1}^{N_b} b_k}{\prod_{k=0}^{N_b} r_k} \quad (2.2)$$

em que u_0 é a iluminação da fonte. O Cycles não armazena diretamente informação sobre a distância total percorrida pelo raio, apenas os valores de cada r_k e o produto deles. Uma forma de se obter o valor desejado é arbitrar uma componente que será multiplicada por a^{r_k} em cada *bounce* (SEDMAN, 2017). Tomando u_R para este fim, conclui-se que:

$$u_R = u_0 \frac{a^{r_0}}{r_0} \frac{a^{r_1} b_1}{r_1} \frac{a^{r_2} b_2}{r_2} \dots \frac{a^{r_{N_b}} b_{N_b}}{r_{N_b}} = u_B a^{r_0+r_1+r_2+\dots+r_{N_b}}. \quad (2.3)$$

É possível recuperar a distância total $r = \sum r_k$ percorrida pelo raio associado ao pixel sendo analisado através de:

$$r = \sum_{k=0}^{N_b} r_k = \log_a \left(\frac{u_R}{u_B} \right). \quad (2.4)$$

2.3 Iluminação gradual da cena

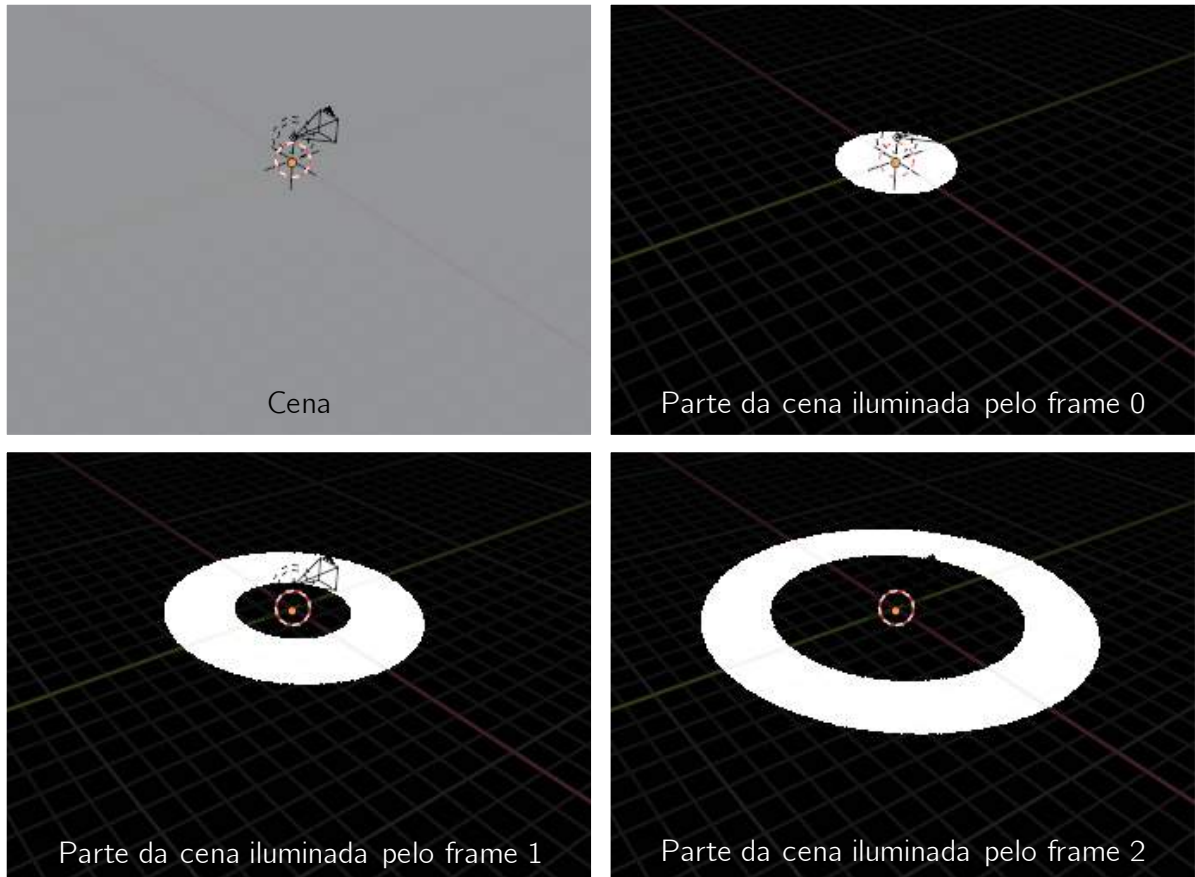


FIGURA 2.3 – Iluminação gradual da cena a cada *frame* para permitir a identificação de alvos obstruídos.

O Cycles também permite usar o valor de r_k para que a fonte luminosa ilumine a cena gradualmente a cada *frame*, ou seja, é possível estabelecer um incremento Δr de forma

que no *frame* k , apenas os pontos que estiverem a uma distância entre $k\Delta r$ e $(k+1)\Delta r$ do emissor sejam iluminados. Nesse caso, $k \in \{0, \dots, N_F - 1\}$, em que N_F é o número total de *frames* renderizados e o incremento Δr está expresso em unidades arbitrárias do Blender, que podem ser convertidas para metros através da multiplicação por uma constante de correção. O procedimento é mostrado na Figura 2.3 para $N_F = 2$ e uma cena composta apenas por um plano horizontal, o que resulta em “anéis” de iluminação a cada *frame*. Essa estratégia permite iluminar objetos obstruídos por outros na mesma cena.

2.4 Implementação do *shader* para configuração de materiais

A codificação dos componentes e a iluminação gradual da cena é implementada no Blender através dos *shaders*, que são atributos internos do programa que permitem manipular a interação da luz com a superfície dos elementos do ambiente simulado. Para a execução dos *shaders*, é necessário fornecer os parâmetros de entrada na interface mostrada na Figura 2.4 para cada objeto adicionado. A partir dos valores informados, o *shader* implementado para simulação do material escolhido é executado.

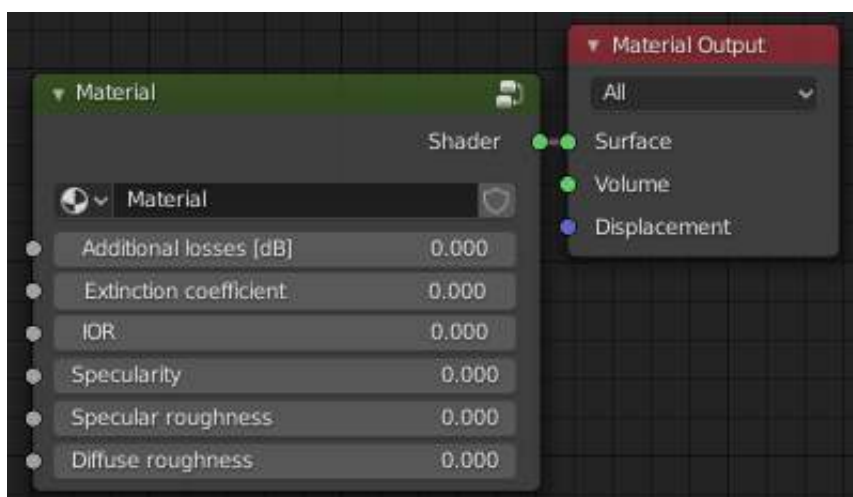


FIGURA 2.4 – Interface para a entrada dos parâmetros das superfícies dos objetos Blender. A partir desses dados o *shader* implementado regula a interação da luz com os elementos da cena.

A fim de tornar a explicação mais clara, a implementação adotada para a simulação de materiais foi dividida em três diagramas de blocos. Neles, a cor de cada bloco indica o tipo da variável descrita, no caso de blocos de valores de entrada, e para os demais a cor define o tipo da variável da saída do bloco. Os *shaders* são um tipo interno do *software* que se conectam diretamente à superfície do objeto e definem seu comportamento quando

iluminado, portanto a saída final deve ser desse tipo.

O primeiro deles, mostrado na Figura 2.5, utiliza as rugosidades especular e difusa, e as componentes RGB já codificadas conforme (2.2) para gerar uma saída A ponderada pelo fator de especularidade.

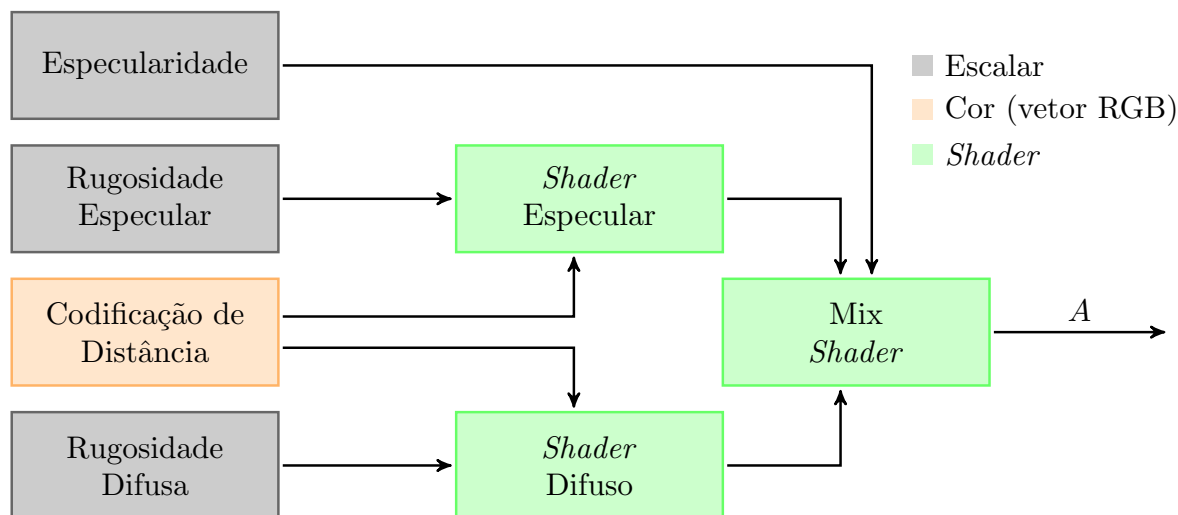


FIGURA 2.5 – Primeira parte da implementação do *shader* usado para a simulação de materiais.

A rugosidade especular define a reflexão especular da superfície e pode variar entre 0 e 1. Quanto mais próxima ela for de 0, mais a reflexão se assemelha à de um espelho ideal, ou seja, com ângulo de incidência igual ao de reflexão, e valores mais próximos de 1 provocam uma dispersão maior nos raios de luz refletidos em torno desse ângulo. A rugosidade difusa também pode variar no mesmo intervalo e determina qual deve ser o modelo de reflexão difusa, que é aquela na qual os raios de luz irradiam em todas as direções, deve ser empregado. Um valor igual a 1 para esse parâmetro indica que deve ser adotado o modelo de Oren-Nayar, que leva em conta aspectos físicos ao contabilizar rugosidades microscópicas na superfície do material. Uma particularização desse caso é a reflexão difusa Lambertiana, que é adotada para um valor de rugosidade difusa igual a 0 e possui desempenho satisfatório para materiais com baixa reflexão especular (Blender Manual, 2018a) ou de superfície uniforme. A especularidade também varia entre 0 e 1 e controla o nível de reflexão especular e difusa no material. Valores próximos de 1 aumentam a influência da reflexão especular, e valores próximos de 0 levam mais em conta a reflexão difusa. O primeiro diagrama de blocos, portanto, diz respeito aos efeitos de reflexão na cena.

O diagrama de blocos da segunda parte do *shader* é mostrado na Figura 2.6. O coeficiente de extinção descreve a atenuação exponencial do sinal dentro do material e é aplicado às três componentes RGB. O sinal atenuado passa pelo *shader* transparente, de

forma a não alterar o percurso da luz, e também passa por um *shader* de refração, que a partir do valor fornecido de índice de refração faz com o que o raio de luz se afaste ou se aproxime da normal à superfície. Também é a partir do índice de refração que se calcula o fator de Fresnel, que determina o quanto de luz é refletida ou refratada/transmitida pelo material (Blender Manual, 2018b). Esse valor pondera a influência do sinal refletido, que é a saída A do diagrama da Figura 2.5, nas saídas B_1 e B_2 . Note que B_1 corresponde ao raio transmitido sem desvio angular, e B_2 ao raio refratado, de forma que ambos sofreram a atenuação devido ao coeficiente de extinção. Dessa forma, o segundo diagrama de blocos leva em conta os efeitos de transmissão, absorção e refração e os combinam com os de reflexão do primeiro diagrama.

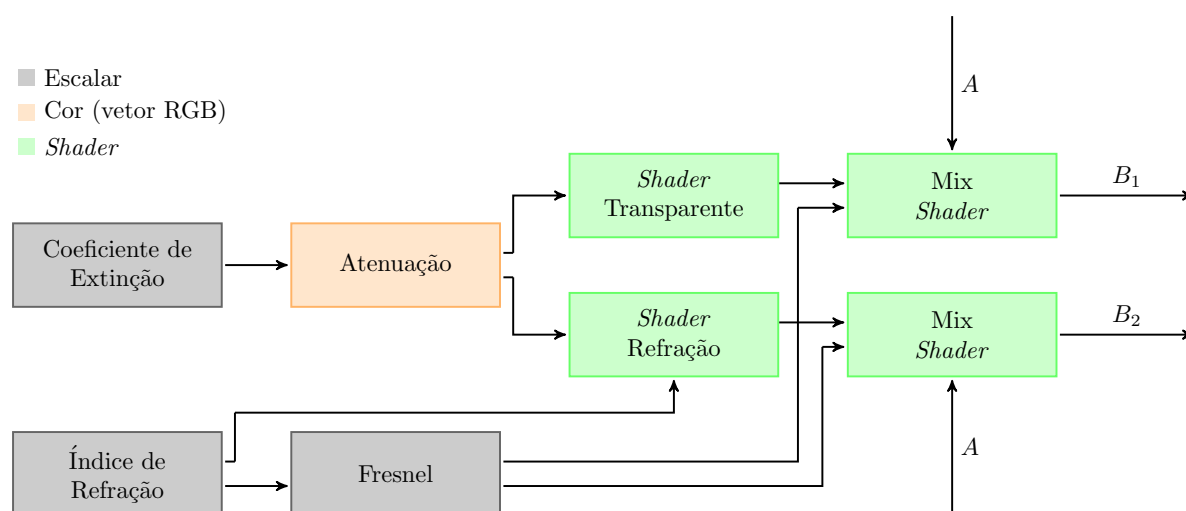


FIGURA 2.6 – Segunda parte da implementação do *shader* usado para a simulação de materiais.

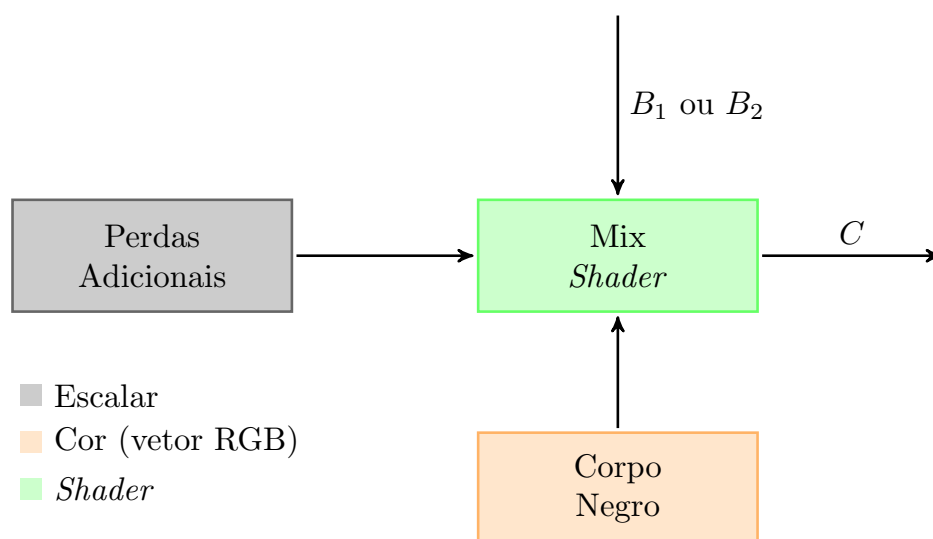


FIGURA 2.7 – Terceira parte da implementação do *shader* usado para a simulação de materiais.

A Figura 2.7 mostra o diagrama de blocos da terceira parte, no qual as perdas adicionais ponderam a influência de um corpo negro, de componentes RGB iguais a 0, na saída do *shader*. Esse corpo negro se combina com uma das saídas do diagrama de blocos da Figura 2.6, a depender do tipo de raio: para os *shadow rays* e *reflection rays*, é usado o raio não refratado (saída B_1), e para os *camera rays* e *transmission rays* é usado o raio refratado (saída B_2).

2.5 *Frames* de saída

A partir da implementação descrita nas Seções 2.2, 2.3 e 2.4, pode-se renderizar uma sequência de *frames* a partir da qual é possível construir um sinal no tempo em MATLAB. Suponha que a renderização de uma determinada cena resultou em vários *frames* constituídos por N_L linhas e N_C colunas de pixels. A partir da componente azul u_B de cada um desses pixels, é possível associar ao raio que deu origem eles uma amplitude A_0 . Além do mais, a distância percorrida por esse raio pode ser recuperada usando (2.4), de forma a extrair duas matrizes para cada *frame*: uma para a distância percorrida pelo raio e uma para sua amplitude. A Figura 2.8 ilustra as matrizes mencionadas para um *frame* qualquer.

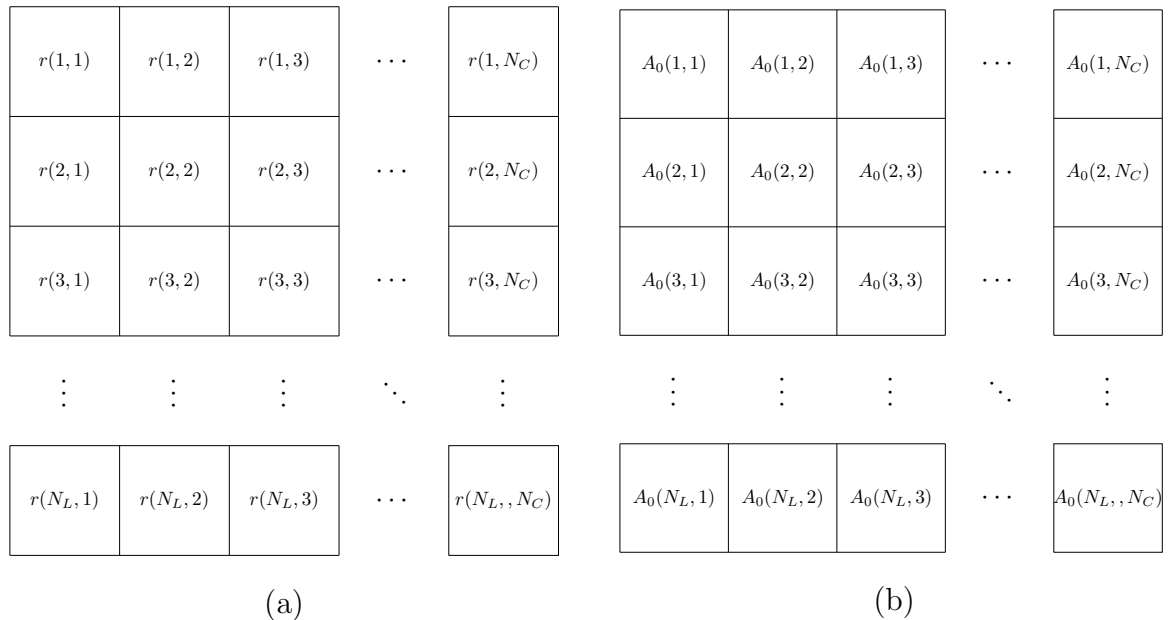


FIGURA 2.8 – Matrizes com a distância percorrida (a) e amplitude (b) associada aos pixels de um determinado *frame*. A distancia percorrida é obtida a partir da componente vermelha, u_R , através de 2.4, e amplitude é a própria componente azul, u_B .

Supondo ainda uma sequência de vários *frames* e usando a notação em negrito para representar matrizes, temos que o n -ésimo *frame* gera as matrizes $\mathbf{r}^{(n)}$ e $\mathbf{A}_0^{(n)}$, em que

$r^{(n)}(i, j)$ e $A_0^{(n)}(i, j)$ são, respectivamente, a distância e a amplitude do raio associado ao pixel da i -ésima linha e j -ésima coluna do frame n .

3 Equação do Radar

Radares são equipamentos utilizados para detectar e adquirir informações sobre alvos através do processamento adequado do sinal eco. O sinal eco é recebido pela antena receptora do radar, sendo o resultado da interação do sinal transmitido com o ambiente e com algum obstáculo da ordem do comprimento de onda do sinal. A Equação do Radar é parte fundamental deste estudo por relacionar grandezas associadas ao alvo e ao sistema transceptor com o alcance (*range*) do radar (SKOLNIK, 2001).

Supondo uma antena transmissora de potência P_t e ganho G , sabe-se que a densidade de potência a uma distância R da antena é dada por:

$$D_t = \frac{P_t G}{4\pi R^2}. \quad (3.1)$$

Um determinado alvo com seção reta radar, assinatura radar, ou RCS, σ irradia parte dessa densidade de potência de volta para o radar, cuja antena receptora possui área efetiva A_e , de forma que a potência P_r recebida pelo radar é:

$$P_r = D_t \times \frac{\sigma}{4\pi R^2} \times A_e = \frac{P_t G \sigma A_e}{(4\pi)^2 R^4}. \quad (3.2)$$

Considerando P_t , G , σ e A_e constantes e sabendo que a potência do sinal é proporcional ao quadrado de sua amplitude, conclui-se que o sinal recebido por um radar monoestático (i.e., com transmissor e receptor na mesma posição) devido ao eco gerado por um alvo a uma distância R deve ter sua amplitude proporcional a $1/R^2$. A equação (3.2), apesar de não considerar fontes de ruído, como o ruído térmico no receptor, e certas perdas, como por exemplo aquelas devido aos efeitos da superfície e da atmosfera terrestres na propagação do sinal, é fundamental para a derivação da forma simplificada da Equação do Radar, que relaciona o *range* máximo (R_{\max}) ao mínimo sinal detectável (S_{\min}) através de:

$$R_{\max} = \left[\frac{P_t G \sigma A_e}{(4\pi)^2 S_{\min}} \right]^{\frac{1}{4}}. \quad (3.3)$$

A fim de verificar o comportamento da amplitude do sinal em função da distância entre o alvo e o radar, a cena da Figura 3.1 foi simulada no Blender. Usando o console

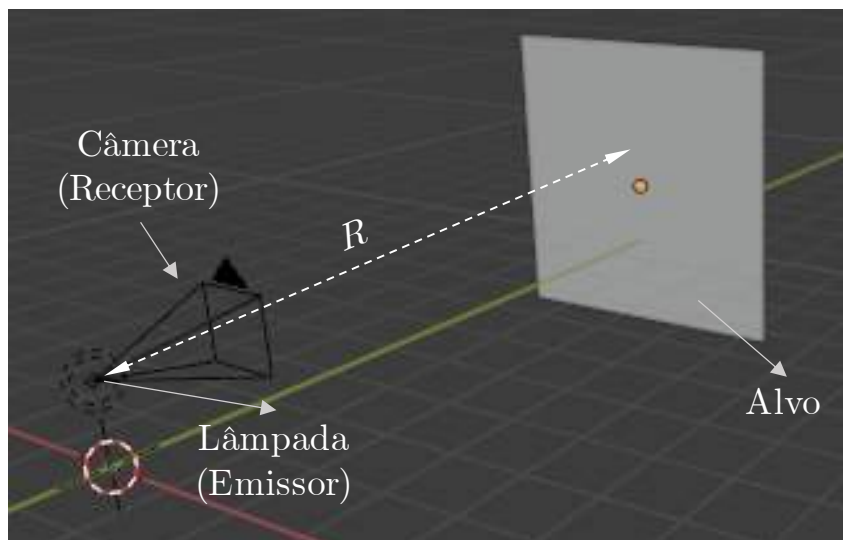


FIGURA 3.1 – Cena montada no Blender para verificação do decaimento da amplitude do sinal.

Python do próprio *software* foi possível empregar vários valores de R sem a necessidade da execução manual de diversos cenários. Visto que cada cena (uma por cada valor de distância analisado) só contava com um único alvo, ela foi renderizada apenas com um *frame*, e a amplitude do sinal foi calculada pela soma dos elementos da matriz \mathbf{A}_0 desse *frame*.

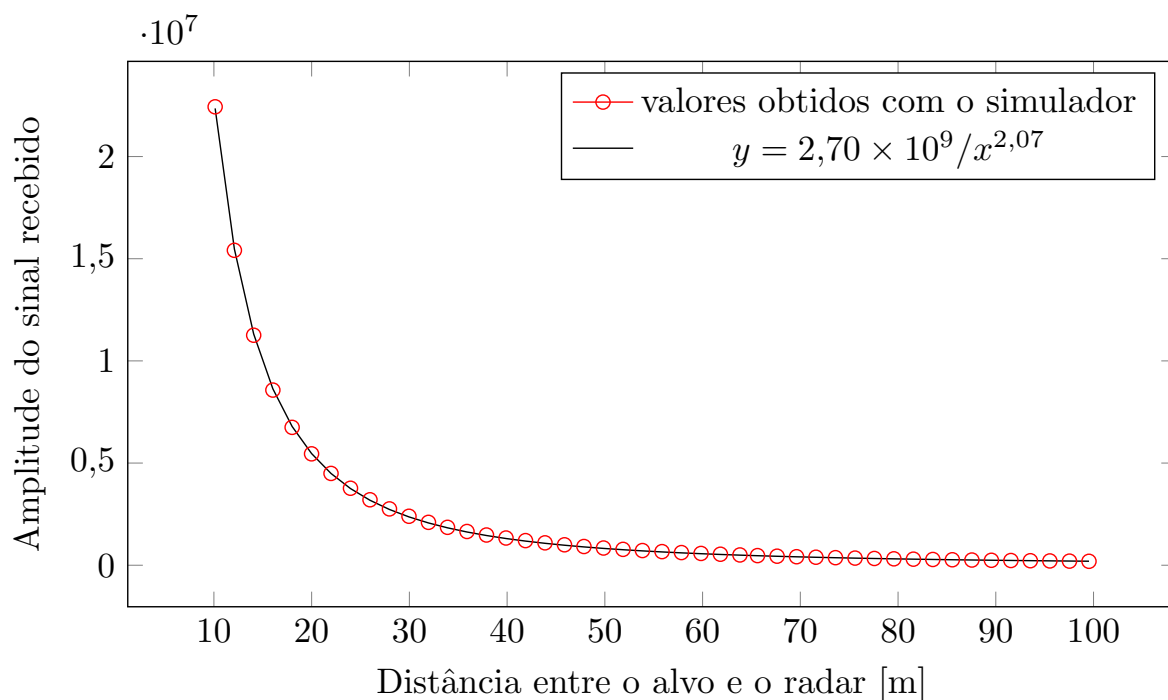


FIGURA 3.2 – Resultados obtidos com o simulador para o decaimento da amplitude com a distância e curva ajustada aos pontos.

Apesar de já se saber a distância R entre o alvo e o sistema transceptor, ela foi recuperada da seguinte forma: os elementos da matriz \mathbf{r} foram adicionados e a soma foi dividida pelo número de elementos não nulos da matriz, e o valor resultante foi dividido por 2 (caso contrário a distância obtida corresponderia ao caminho de ida e volta do raio). Esse procedimento foi adotado pois, à medida que o alvo se afastava da câmera, ele ocupava um espaço cada vez menor da imagem renderizada, de forma que, se a soma dos elementos de \mathbf{r} fosse dividida pelo tamanho total do *frame*, o valor médio obtido não seria correspondente à distância percorrida pelos raios válidos (que são aqueles associados a uma distância percorrida não nula, ou seja, $r(i, j) \neq 0$). A recuperação da distância dessa forma só é possível pois a análise feita levou em conta um único alvo simples. No caso de múltiplos alvos ou superfícies com quinas e cavidades, por exemplo, as múltiplas reflexões aumentariam o valor da distância percorrida, inviabilizando a recuperação do afastamento entre alvo e radar desta forma.

A Figura 3.2 mostra os resultados obtidos para R variando entre 10 m e 100 m com passo de 2 m. Note que os valores foram recuperados corretamente, comprovando que o método adotado para codificar a distância na componente vermelha foi bem-sucedido. Usando a ferramenta `cftool` do MATLAB foi possível ajustar a curva mostrada em preto na Figura 3.2, evidenciando a correspondência entre os comportamentos obtido e esperado.

4 Compressão de Pulsos

A resolução em *range*, denotada por δR , corresponde à mínima separação entre dois alvos para que eles sejam identificados separadamente pelo radar. Esse parâmetro depende do tipo de sinal empregado pelo sistema, uma vez que é calculado a partir da largura de banda B do sinal através da equação:

$$\delta R = \frac{c}{2B}, \quad (4.1)$$

em que c é a velocidade da luz no vácuo. A fim de demonstrar a vantagem da técnica de compressão de pulsos para a resolução em *range*, é conveniente analisar antes o comportamento de um pulso não modulado $s_{\text{NM}}(t)$, de duração T_{NM} e amplitude A_{NM} , dado por:

$$s_{\text{NM}}(t) = A_{\text{NM}} \text{rect} \left(\frac{t}{T_{\text{NM}}} \right), \quad (4.2)$$

em que $\text{rect}(\cdot)$ denota a função retangular definida da seguinte forma:

$$\text{rect}(t) = \begin{cases} 1 & \text{se } |t| < \frac{1}{2}, \\ 0 & \text{caso contrário.} \end{cases} \quad (4.3)$$

O sinal $s_{\text{NM}}(t)$ é mostrado na Figura 4.1, e sua transformada de Fourier, obtida através da equação (4.4), é mostrada na Figura 4.2.

$$\begin{aligned} S_{\text{NM}}(f) &= \mathcal{F} \{s_{\text{NM}}(t)\} = \int_{-\infty}^{\infty} s_{\text{NM}}(t) \exp(-j2\pi ft) dt \\ &= A_{\text{NM}} \int_{-\frac{T_{\text{NM}}}{2}}^{\frac{T_{\text{NM}}}{2}} \exp(-j2\pi ft) dt \\ &= -A_{\text{NM}} \frac{\exp(-j2\pi ft)}{j2\pi f} \Bigg|_{-\frac{T_{\text{NM}}}{2}}^{\frac{T_{\text{NM}}}{2}} \\ &= A_{\text{NM}} T_{\text{NM}} \text{sinc}(fT_{\text{NM}}) \end{aligned} \quad (4.4)$$

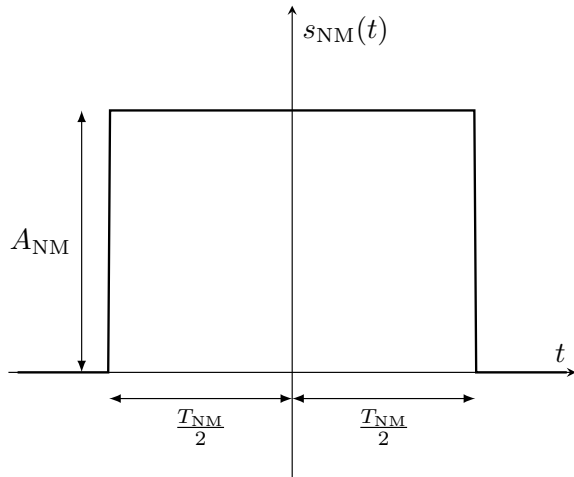


FIGURA 4.1 – Pulso não modulado no domínio do tempo.

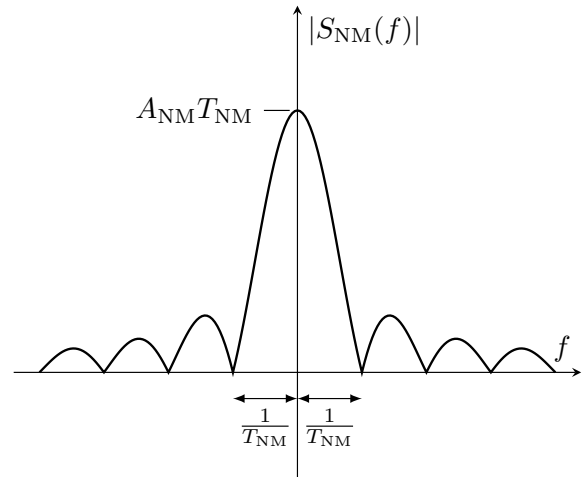


FIGURA 4.2 – Espectro do pulso não modulado.

Da Figura 4.2, percebe-se que a largura de banda do sinal não modulado é determinada pela duração do pulso, de forma que a resolução em *range* é de:

$$\delta R_{\text{NM}} = \frac{c}{2\frac{1}{T_{\text{NM}}}} = \frac{cT_{\text{NM}}}{2}. \quad (4.5)$$

Ao se adotar um pulso não modulado, é necessário reduzir a duração do pulso a fim de melhorar a resolução (i.e., torná-la menor). Para manter um nível razoável de energia em um pulso estreito, é preciso usar potências de pico muito altas, o que pode sobrecarregar as linhas de transmissão usadas no sistema (SKOLNIK, 2001). A modulação linear em frequência, ou LFM, permite o uso de pulsos longos para a obtenção de resoluções baixas empregando um sinal igual àquele descrito pela equação (4.6) e mostrado na Figura 4.3. Nesse caso, a amplitude e a duração do pulso são denotadas por A_M e T_M , respectivamente, e μ é o índice de modulação.

$$s_M(t) = A_M \text{rect}\left(\frac{t}{T_M}\right) \exp(j\pi\mu t^2) \quad (4.6)$$

Note que o envelope do sinal LFM é um pulso retangular semelhante ao sinal não modulado, mas sua fase é igual a $\phi_M(t) = \pi\mu t^2$, de forma que a frequência instantânea do sinal varia linearmente no tempo através da equação:

$$\frac{1}{2\pi} \frac{d\phi_M(t)}{dt} = \mu t. \quad (4.7)$$

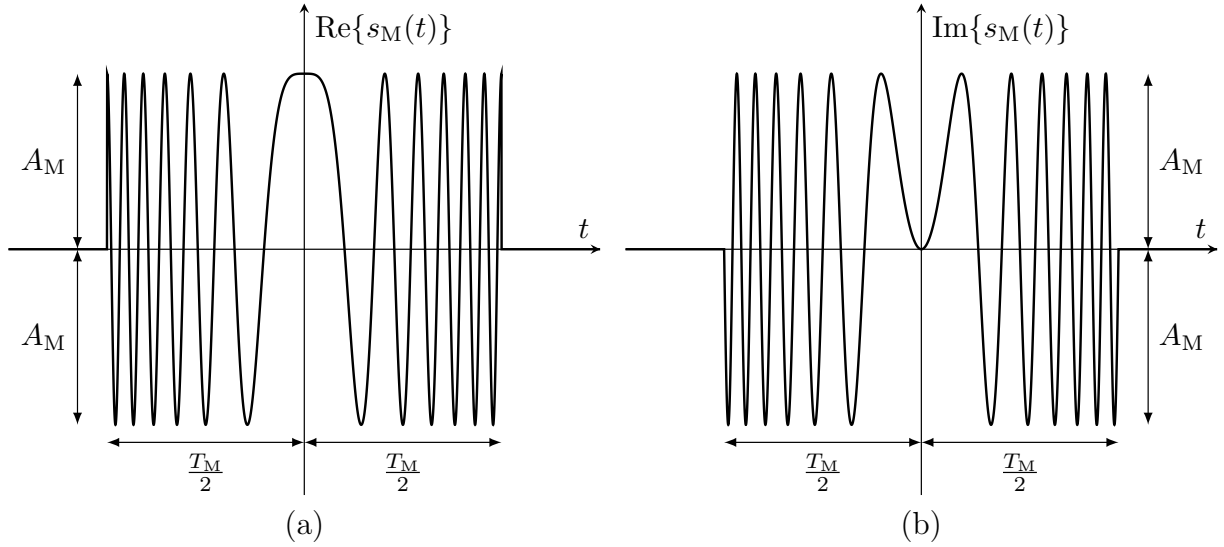


FIGURA 4.3 – Partes real (a) e imaginária (b) do sinal descrito por (4.6).

A transformada de Fourier de $s_M(t)$ é dada por:

$$\begin{aligned}
 S_M(f) &= \mathcal{F}\{s_M(t)\} = \int_{-\infty}^{\infty} s_M(t) \exp(-j2\pi ft) dt \\
 &= A_M \int_{-\frac{T_M}{2}}^{\frac{T_M}{2}} \exp(j\pi\mu t^2 - j2\pi ft) dt \\
 &= A_M \exp\left(-\frac{j\pi f^2}{\mu^2}\right) \int_{-\frac{T_M}{2}}^{\frac{T_M}{2}} \exp\left[j\pi\mu\left(t - \frac{f}{\mu}\right)^2\right] dt. \quad (4.8)
 \end{aligned}$$

Usando a mudança de variável $x = \sqrt{2\mu}\left(t - \frac{f}{\mu}\right)$ obtém-se:

$$S_M(f) = \frac{A_M}{\sqrt{2\mu}} \exp\left(-\frac{j\pi f^2}{\mu^2}\right) \int_{-x_1}^{x_2} \exp\left(\frac{j\pi x^2}{2}\right) dx, \quad (4.9)$$

em que

$$x_1 = \sqrt{2B_M T_M} \left(\frac{1}{2} + \frac{f}{B_M}\right), \quad (4.10)$$

$$x_2 = \sqrt{2B_M T_M} \left(\frac{2}{2} - \frac{f}{B_M}\right), \quad (4.11)$$

$$B_M = \mu T_M. \quad (4.12)$$

Definindo as seguintes integrais de Fresnel por:

$$C(x) = \int_0^x \cos\left(\frac{j\pi\nu^2}{2}\right) d\nu, \quad (4.13)$$

$$S(x) = \int_0^x \sin\left(\frac{j\pi\nu^2}{2}\right) d\nu, \quad (4.14)$$

e sabendo que $C(-x) = -C(x)$ e $S(-x) = -S(x)$, é possível reescrever (4.9) como:

$$S_M(f) = \frac{A_M}{\sqrt{2\mu}} \exp\left(-\frac{j\pi f^2}{\mu^2}\right) \{C(x_2) + C(x_1) + j[S(x_2) + S(x_1)]\}. \quad (4.15)$$

Consequentemente, a magnitude do espectro de $S_M(f)$ é dada por:

$$|S_M(f)| = \frac{A_M}{\sqrt{2\mu}} \sqrt{[C(x_1) + C(x_2)]^2 + [S(x_1) + S(x_2)]^2}. \quad (4.16)$$

A Figura 4.4 mostra o comportamento das integrais de Fresnel definidas por (4.13) e (4.14). De (4.10) e (4.11), tem-se que para $|f| < B_M/2$, tanto x_1 quanto x_2 são positivos, de forma que, quando o produto $B_M T_M \rightarrow \infty$, $x_1 \rightarrow \infty$ e $x_2 \rightarrow \infty$. Já para $|f| > B_M/2$, x_1 permanece positivo mas x_2 fica negativo. Assim, quando $B_M T_M \rightarrow \infty$, $x_1 \rightarrow \infty$ e $x_2 \rightarrow -\infty$. A Figura 4.5 ilustra a situação descrita.

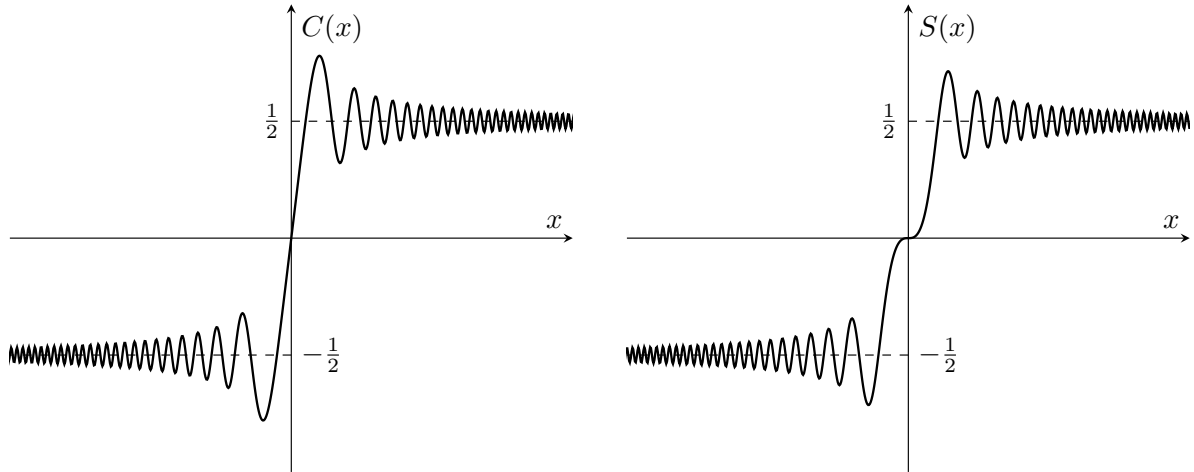


FIGURA 4.4 – Comportamento das integrais de Fresnel definidas em (4.13) e (4.14).

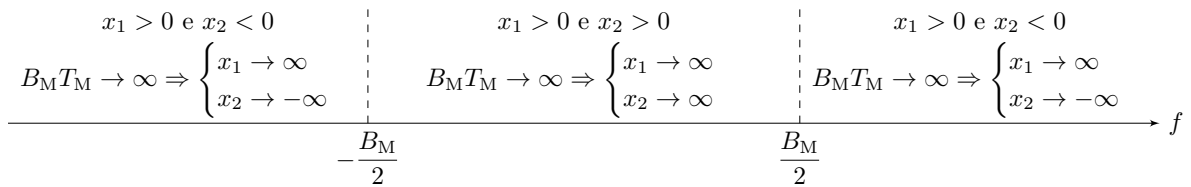


FIGURA 4.5 – Comportamento de x_1 e x_2 quando $B_M T_M \rightarrow \infty$.

Logo, no limite em que $B_M T_M \rightarrow \infty$, o que se tem é:

$$\begin{cases} |S_M(f)| \rightarrow \frac{A_M}{\sqrt{2\mu}} \sqrt{\left[\frac{1}{2} + \frac{1}{2}\right]^2 + \left[\frac{1}{2} + \frac{1}{2}\right]^2} = \frac{A_M}{\sqrt{\mu}} & \text{se } |f| < \frac{B_M}{2}, \\ |S_M(f)| \rightarrow \frac{A_M}{\sqrt{2\mu}} \sqrt{\left[\frac{1}{2} - \frac{1}{2}\right]^2 + \left[\frac{1}{2} - \frac{1}{2}\right]^2} = 0 & \text{se } |f| > \frac{B_M}{2}. \end{cases} \quad (4.17)$$

Isso significa que, para valores altos do produto $B_M T_M$, o espectro de S_M pode ser aproximado por:

$$|S_{M,\text{aprox}}(f)| = \frac{A_M}{\sqrt{\mu}} \text{rect}\left(\frac{f}{B_M}\right). \quad (4.18)$$

De fato, para $B_M T_M \geq 10$, 95% da energia do sinal se encontra no intervalo $[-\frac{B_M}{2}, \frac{B_M}{2}]$, e quando $B_M T_M \geq 100$, essa porcentagem chega a 99% (YANG et al., 2018). A Figura 4.3 compara o espectro de $|S_M(f)|$ e $|S_{M,\text{aprox}}(f)|$. Conclui-se que a resolução fornecida por um sinal LFM é de:

$$\delta R_M = \frac{c}{2B_M} = \frac{c}{2\mu T_M}. \quad (4.19)$$

Comparando (4.5) com (4.19), percebe-se que um pulso comprimido longo pode atingir valores baixos de resolução, que exigiriam um pulso não modulado muito estreito. Além do mais, através da introdução do índice de modulação μ , é possível escolher os valores de B_M e T_M independentemente, tornando a LFM ainda mais vantajosa.

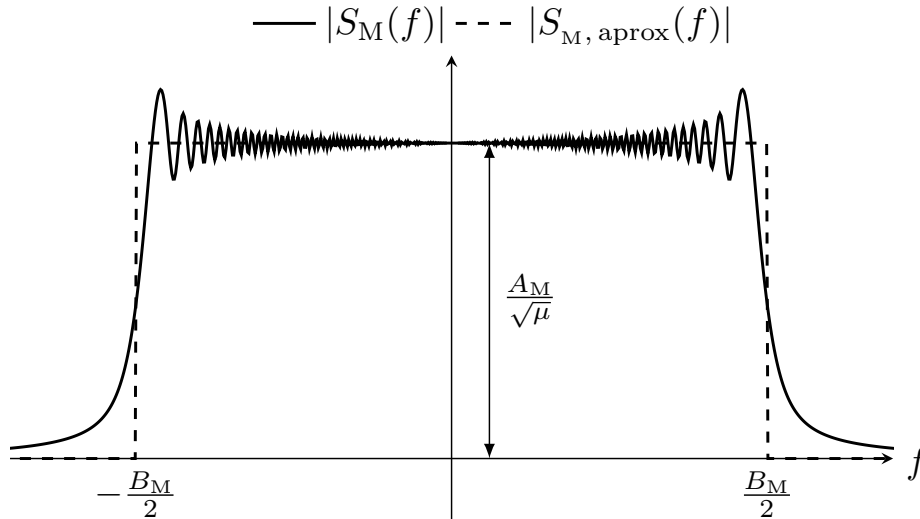


FIGURA 4.6 – Comparação entre o espectro de $s_M(t)$ dado por (4.16) e (4.18).

4.1 Implementação em MATLAB

Em sistemas radar é comum o emprego de sinais LFM em filtros casados, que efetuam a correlação entre o sinal do filtro e o sinal recebido de forma a maximizar a relação sinal-ruído. Para que a filtragem pudesse ser implementada em MATLAB e usada para o tratamento dos *frames* renderizados pelo Blender, seu sinal foi adotado como sendo:

$$s_{\text{filtro}}[k] = \text{rect}\left(\frac{\tau[k]}{T_p}\right) \exp(j\pi\mu(\tau[k])^2), \text{ com } k = 1, 2, \dots, N_s, \quad (4.20)$$

em que T_p é a duração do pulso, N_s é o número de amostras analisadas e μ é calculado a partir da largura de banda B através de $\mu = B/T_p$. Denotando por T_s o tempo de amostragem, $\tau[k]$ é dado por:

$$\tau[k] = (k - 1)T_s - \frac{T_p}{2}. \quad (4.21)$$

Um vez que a indexação em MATLAB se inicia em 1, a expressão (4.21) garante que o início da amostragem coincida com o início do pulso. Para a análise dos *frames* resultantes da simulação em Blender, assumiu-se que cada pixel se comportasse como um alvo pontual de sinal:

$$s_{\text{pixel}}[k] = A_0 \text{rect} \left(\frac{\tau[k] - \tau_0}{T_p} \right) \exp(j\pi\mu(\tau[k] - \tau_0)^2). \quad (4.22)$$

Para cada pixel os valores de atraso τ_0 e amplitude A_0 são diferentes e obtidos através das componentes RGB do frame renderizado. A amplitude A_0 é aquela mencionada na Seção 2.5, e o atraso τ_0 se refere ao tempo que o sinal leva para percorrer a distância r , que pode ser obtida através de (2.4). Com isso, o atraso é dado por:

$$\tau_0 = \frac{r}{c}. \quad (4.23)$$

O sinal associado a um frame com N_L linhas e N_C colunas de pixels é dado pela soma dos sinais associados a cada pixel dividido pelo número de raios válidos no *frame*. Essa divisão é necessária pois, à medida que os alvos se afastam da câmera, eles ocupam cada vez menos pixels na imagem final, o que aumenta o número de raios não válidos. Caso essa divisão não fosse feita, é como se os alvos mais afastados tivessem intensidade ainda menor que aquela armazenada na componente azul do pixel associado, o que estaria incorreto. Dessa forma, denotando por N_V o número de raios válidos por *frame* (i.e., o número de elementos não nulos da matriz \mathbf{r} associada), tem-se que:

$$\begin{aligned} s_{\text{frame}}[k] &= \frac{1}{N_V} \sum_{i=1}^{N_C} \sum_{j=1}^{N_L} s_{\text{pixel}}^{(i,j)}[k] \\ &= \frac{1}{N_V} \sum_{i=1}^{N_C} \sum_{j=1}^{N_L} A_0(i, j) \text{rect} \left(\frac{\tau[k] - \tau_0(i, j)}{T_p} \right) \exp(j\pi\mu(\tau[k] - \tau_0(i, j))^2), \end{aligned} \quad (4.24)$$

em que $A_0(i, j)$ e $\tau_0(i, j) = r(i, j)/c$ são, respectivamente, a amplitude e o atraso associados ao sinal do pixel da i -ésima linha e j -ésima coluna do *frame*, denotado por $s_{\text{pixel}}^{(i,j)}$. A equação (4.24) só é válida para $N_V \neq 0$, pois quando $N_V = 0$ o sinal do *frame* também é nulo. Note que, na análise da cena da Figura 3.1, no Capítulo 3, a divisão pelo número de raios válidos foi feita para os valores de distância, pois naquele caso desejava-se obter um valor médio desse parâmetro associado à cada *frame*. Na implementação proposta neste Capítulo, no entanto, o valor médio computado para cada *frame* se refere à amplitude, e

por isso ela é normalizada pelo número de raios válidos. De maneira resumida, a divisão pelo número de raios válidos é necessária quando um valor médio é associado a cada *frame*, para corrigir o fato de alvos distantes ocuparem menos pixels na imagem final, e a depender da análise feita pós-renderização, essa etapa pode ou não ser necessária. Na prática, essa constatação muitas vezes é feita ao se comparar os resultados obtidos com o que era esperado.

Em uma cena renderizada em N_F *frames*, o sinal resultante é a soma dos sinais de cada *frame*. Sendo $s_{\text{frame}}^{(n)}$ o sinal do n -ésimo *frame*, tem-se que:

$$s_{\text{cena}}[k] = \sum_{n=1}^{N_F} s_{\text{frame}}^{(n)}[k]. \quad (4.25)$$

Denotando por S_{cena} a Transformada de Fourier Discreta, DFT, de s_{cena} e, S_{filtro} a DFT de s_{filtro} , a saída do filtro casado, s_{filtrado} , é dada por:

$$s_{\text{filtrado}} = \text{DFT}^{-1} \{ S_{\text{cena}} S_{\text{filtro}}^* \}. \quad (4.26)$$

4.2 Verificação da resolução em *range* com dois alvos simples

Para a verificação da resolução em *range* a partir da implementação descrita, foi montada a cena da Figura 4.7 no Blender, e adotou-se os parâmetros da Tabela 4.1, que são aqueles empregados no radar SABER M60. Note que, nesse caso, a renderização deve ser feita em, pelo menos, dois *frames*, com os alvos em *frames* distintos, visto que há obstrução do Alvo 2 pelo Alvo 1. A partir dos dados fornecidos, a resolução é de:

$$\delta R = \frac{c}{2B} = 75 \text{ m}. \quad (4.27)$$

TABELA 4.1 – Parâmetros do radar SABER M60. Fonte: (CARVALHO et al., 2008).

Parâmetro	Valor
Largura de pulso	22 μs
Banda	2 MHz
Frequência de operação	1,2 GHz

A Figura 4.8 mostra os resultados obtidos para diferentes valores de d_2 . Esperava-se

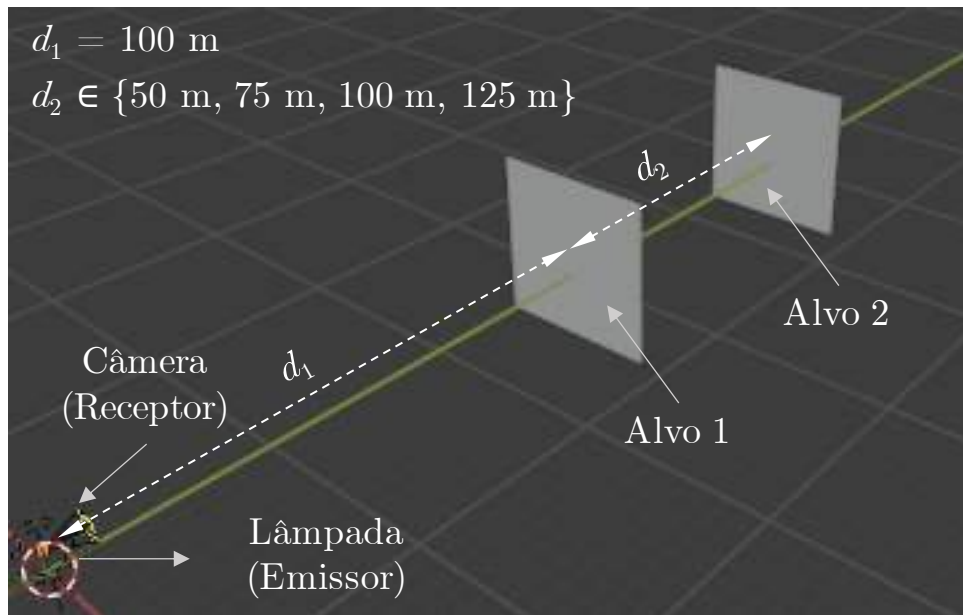


FIGURA 4.7 – Cena usada para verificação da resolução em *range*.

que, para $d_2 > 75 \text{ m}$, os dois alvos fossem identificados separadamente pelo radar, mas não foi isso que aconteceu. Nem mesmo em $d_2 = 125 \text{ m}$ a separação entre os alvos estava nítida. Para verificar a causa dessa inconsistência, os sinais associados a cada alvo foram analisados separadamente. Como cada um dos alvos foi renderizado em um único *frame*, foi necessário apenas recuperar o sinal associado ao *frame* desejado.

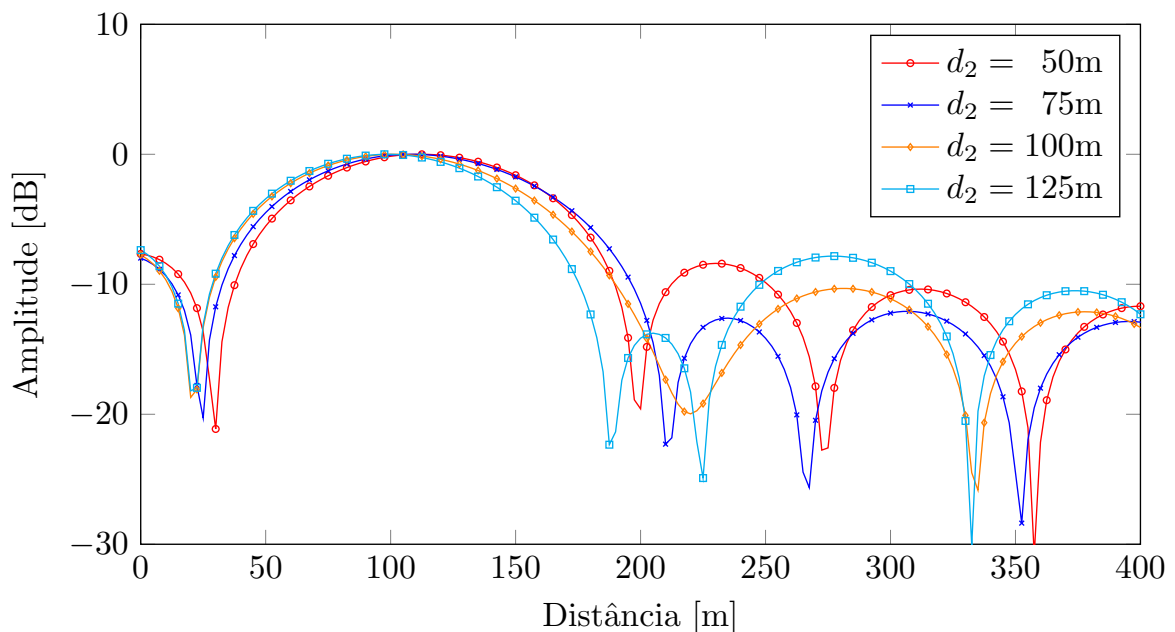


FIGURA 4.8 – Amplitude do sinal recebido pelo radar para os diferentes valores de d_2 .

A Figura 4.9 mostra como o sinal associado a cada alvo se comporta para diferentes

valores de distância. A partir da análise desse gráfico, constatou-se que, nos casos em que a distância entre os alvos era suficiente para a identificação deles, o lóbulo secundário associado ao primeiro alvo tinha amplitude próxima àquela do lóbulo principal do segundo. Com isso, quando se efetuava a superposição dos sinais, não era possível identificar o segundo alvo, visto que ele se confundia com os lóbulos do objeto mais próximo da câmera.

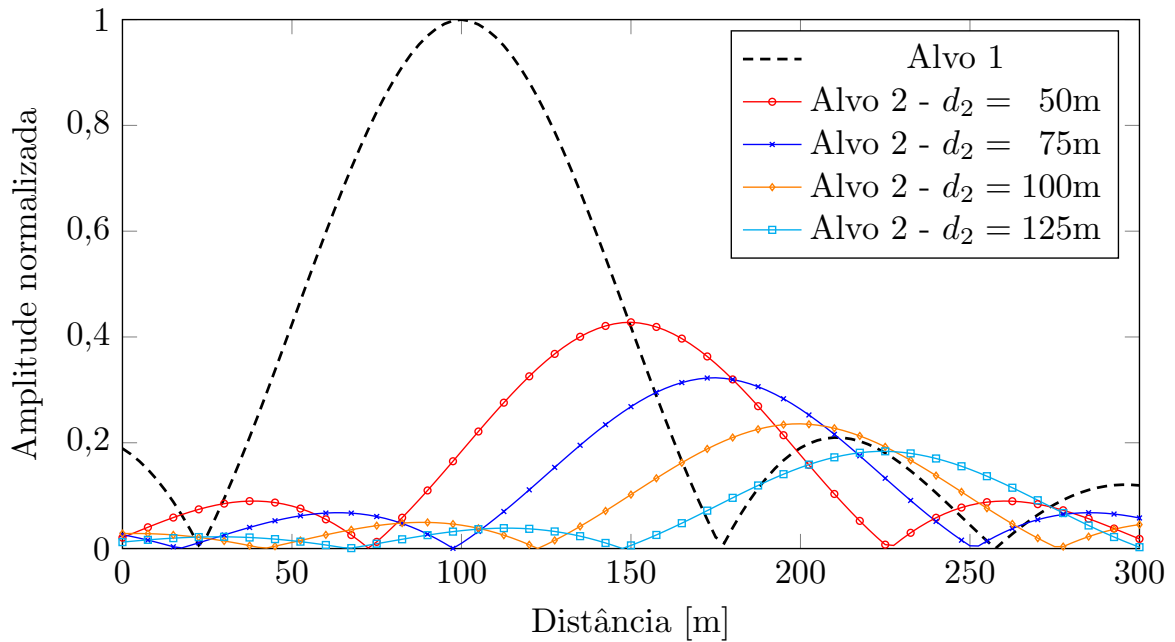


FIGURA 4.9 – Sinal associado a cada alvo para os diferentes valores de d_2 .

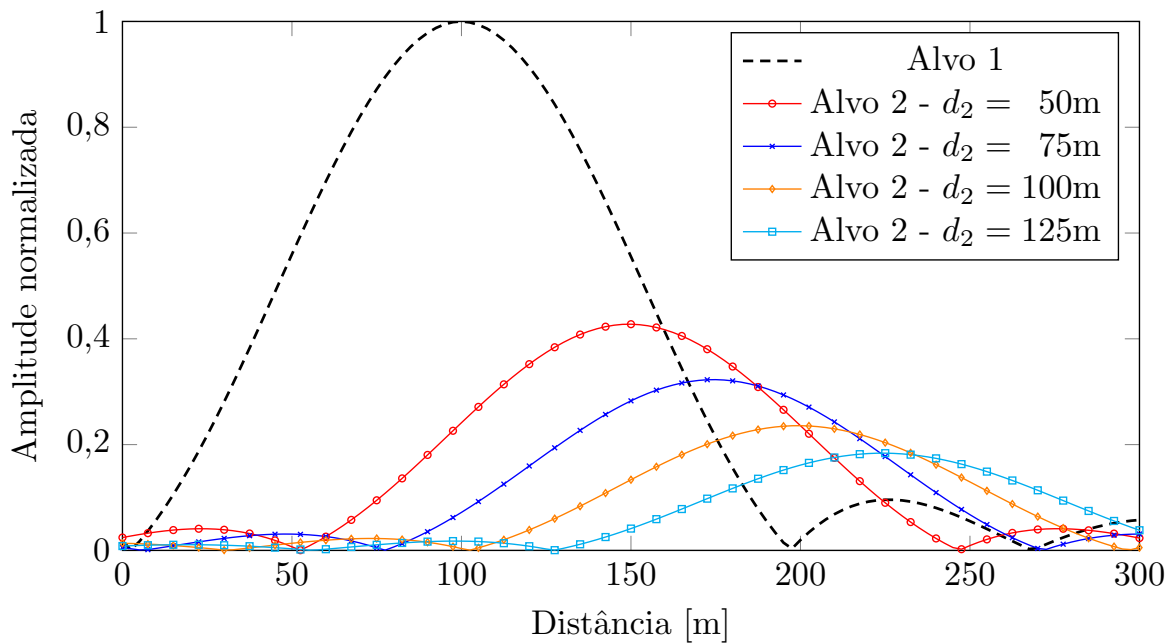


FIGURA 4.10 – Sinal associado a cada alvo para os diferentes valores de d_2 após o uso da janela de Kaiser.

De fato, esse é um problema comum que pode afetar a resolução de um radar. Uma forma de contornar isso é através de janelas, que são sinais com formato específico e que, ao serem combinados com o sinal do filtro, levam à redução dos lóbulos laterais. Em contrapartida, há um aumento da largura do lóbulo principal (LEVANON; MOZESON, 2004).

A fim de verificar se, de fato, a discordância da resolução poderia ser amenizada com o uso de janelas, a função nativa do MATLAB, `kaiser`, foi empregada para a obtenção de uma janela de Kaiser adequada. Para isso, é preciso informar um parâmetro `beta` que define atenuação dos lóbulos laterais. Após alguns testes, verificou-se que, para o caso analisado, a janela mais efetiva era aquela em que `beta = 2,5`.

O efeito da janela no sinal de cada alvo é mostrado na Figura 4.9. Note que os lóbulos laterais do primeiro alvo foram atenuados, mas houve um aumento significativo de sua largura. Antes da janela de Kaiser, o lóbulo principal do alvo em uma posição $d_2 = 125$ m era menor que o lóbulo secundário do primeiro alvo, o que deixou de acontecer com o emprego da janela, às custas de um alargamento significativo do lóbulo principal.

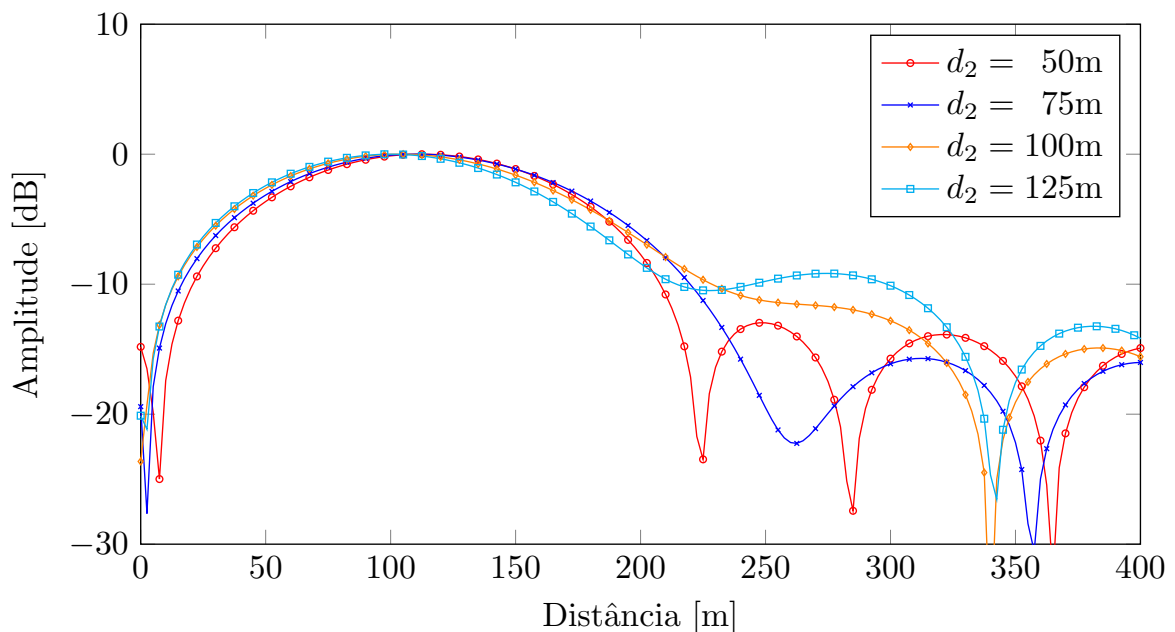


FIGURA 4.11 – Amplitude do sinal recebido pelo radar para os diferentes valores de d_2 após o uso da janela de Kaiser.

A Figura 4.11 mostra o efeito da janela de Kaiser no sinal resultante recebido pelo radar. Para $d_2 = 50$ m e $d_2 = 75$ m, assim como na Figura 4.8, os dois alvos são vistos como um só localizado em uma posição intermediária entre eles. Já para $d_2 = 100$ m e $d_2 = 125$ m percebe-se uma mudança significativa na curva, e apesar de o ponto de máximo associado ao segundo alvo não estar totalmente separado daquele associado ao primeiro, é possível identificar um sinal de formato diferente daquele recebido para $d_2 = 50$ m e

$d_2 = 75$ m, indicando a detecção gradual do segundo alvo devido ao emprego da janela.

Uma possível explicação para a incoerência observada entre o valor de resolução esperado e o obtido com a simulação, mesmo com o emprego de uma janela, é o fato de o Blender, por ser um *software* voltado para a produção de jogos e animações, ter seu funcionamento otimizado para uma faixa de operação próxima daquela da luz visível. Sendo assim, ao realizar análises em frequências mais altas a partir dos dados fornecidos pelo Cycles, é comum que essas incompatibilidades surjam.

5 Frequência Doppler

O desvio em frequência devido ao efeito Doppler surge quando existe movimento relativo entre o radar e o alvo, e é útil para a verificação da velocidade em cenários com alvos móveis. Sendo R a distância entre o alvo e o radar, e λ seu comprimento de onda, a mudança de fase ϑ no percurso de ida e volta da onda é de:

$$\vartheta = 2\pi \frac{2R}{\lambda}. \quad (5.1)$$

A frequência Doppler, f_d , pode ser escrita em termos da taxa de variação da fase, ω_d , através da equação:

$$f_d = \frac{\omega_d}{2\pi} = \frac{1}{2\pi} \frac{d\vartheta}{dt} = \frac{2}{\lambda} \frac{dR}{dt} = \frac{2v_r}{\lambda}, \quad (5.2)$$

em que v_r é a velocidade radial, ou seja, a componente da velocidade relativa entre radar e alvo que se encontra sobre a linha imaginária que os liga.

5.1 Detecção do efeito Doppler no simulador

Quando existe movimento relativo entre o alvo e conjunto emissor-câmera, a cena deve ser renderizada em duas ou mais etapas. Para isso, utiliza-se o console Python interno do Blender, que permite acessar os parâmetros de posição dos componentes da cena, e assim definir a velocidade do(s) objeto(s) em uma determinada direção e o intervalo de tempo Δt entre as duas etapas sucessivas de renderização. Suponha, por exemplo, que se deseja simular um cenário simples como aquele mostrado na Figura 3.1, de forma que o alvo se aproxima do radar a 1 m/s e se encontra, inicialmente, a 10 m dele. Arbitrando um Δt de 2 s tem-se uma primeira renderização (que pode conter vários *frames*, caso fosse necessário) com o alvo nessa posição inicial, seguida de outra na qual o alvo se encontra 2 m mais próximo do radar, ou seja, a 8 m dele. A segunda renderização também pode conter vários *frames*, e esse processo pode ser feito mais de uma vez, ou seja, em múltiplas etapas de renderização. O uso do console Python interno do *software* permite automatizar esse processo para cenários com múltiplos alvos e/ou analisados em várias etapas.

Denotando por $\mathbf{r}^{(n)}(t)$ e $\mathbf{r}^{(n)}(t + \Delta t)$ as matrizes com as distâncias percorridas pelos

raios associados aos pixels do n -ésimo *frame* de duas etapas consecutivas de renderização, pode-se obter a matriz de velocidade radial associada ao *frame* n através de:

$$\mathbf{v}_r^{(n)} = - \left(\frac{\mathbf{r}^{(n)}(t + \Delta t) - \mathbf{r}^{(n)}(t)}{2\Delta t} \right). \quad (5.3)$$

Consequentemente, a frequência Doppler associada ao *frame* n é dada pela matriz:

$$\mathbf{f}_d^{(n)} = \frac{2\mathbf{v}_r^{(n)}}{\lambda} = - \left(\frac{\mathbf{r}^{(n)}(t + \Delta t) - \mathbf{r}^{(n)}(t)}{\lambda\Delta t} \right). \quad (5.4)$$

É possível mapear os valores de cada um dos elementos $f_d(i, j)$ da matriz $\mathbf{f}_d^{(n)}$ em um intervalo $[-f_m, f_m]$. Para isso, o intervalo é dividido em N_d partes iguais, cada uma delas associada a um índice $m^{(i, j)}$ de forma que, se $-f_m \leq f_d(i, j) \leq f_m$, esse valor é dado pela condição:

$$m^{(i, j)} = m \text{ se } -1 + (m-1)\frac{2}{N_d} \leq \frac{f_d(i, j)}{f_m} < -1 + m\frac{2}{N_d}. \quad (5.5)$$

Para estender (5.5) para $|f_d(i, j)| > f_m$, basta tomar a parte fracionária de $f_d(i, j)/f_m$. Suponha, por exemplo, $f_m = 1000$ Hz e $N_d = 100$. Nesse caso, a aplicação direta de (5.5) diz que um desvio em frequência de 10 Hz tem $m = 51$, pois se encontra no intervalo que vai de 0 a 0,02. No entanto, um desvio em frequência de 1010 Hz também tem $m = 51$, pois a parte fracionária de 1010/1000 é igual a 0,01, que pertence ao mesmo intervalo. Para um desvio em frequência negativo de módulo maior que f_m , como por exemplo -1010 Hz, tem-se que $m = 50$, visto que, nesse caso, a parte fracionária de $-1010/1000$ é $-0,01$, que está entre $-0,02$ e 0. Note que, para números negativos, diferentemente de algumas definições tradicionais adotadas, a parte fracionária pode ser negativa pois deseja-se contabilizar um desvio em relação a um dos limites do intervalo $[-f_m, f_m]$. A Figura 5.1 elucida, através de uma ilustração, a solução proposta. Note que, com isso, surge um problema de ambiguidade, pois é preciso limitar o intervalo em que as frequências são mapeadas.

Assim, é possível construir uma matriz $\mathbf{s}_{\text{frame}}$ de N_d linhas e N_s colunas associada ao sinal de um *frame*, com cada linha sendo constituída da soma dos sinais dos pixels cuja frequência Doppler esteja dentro do intervalo correspondente ao índice da linha, ou seja,

$$s_{\text{frame}}[m, k] = \sum_{i=1}^{N_L} \sum_{j=1}^{N_C} s_{\text{pixel}}^{(i, j)}[k] \quad \text{desde que } m^{(i, j)} = m. \quad (5.6)$$

A Figura 5.2 ilustra o processo descrito. Percebe-se que $\mathbf{s}_{\text{frame}}$, conforme definido em (4.24), é a soma das linhas da matriz $\mathbf{s}_{\text{frame}}$ definida por (5.6). O sinal resultante também passa a ser uma matriz \mathbf{s}_{cena} de dimensões $N_d \times N_s$ correspondente à soma dos sinais de

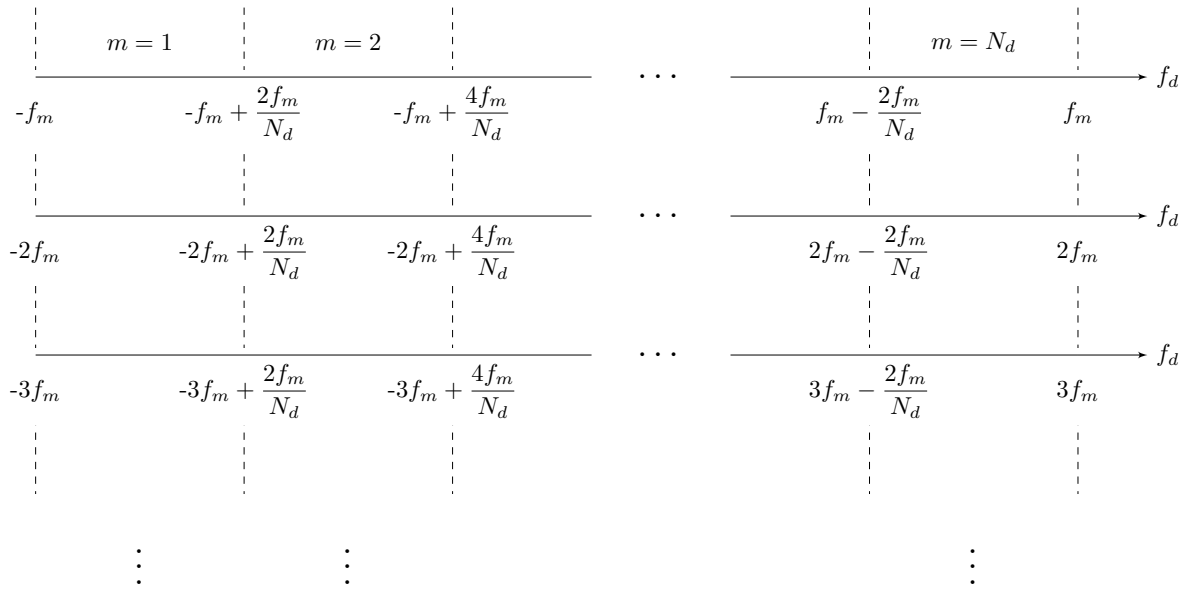


FIGURA 5.1 – Esquema de mapeamento das frequências Doppler em N_d intervalos uniformes.

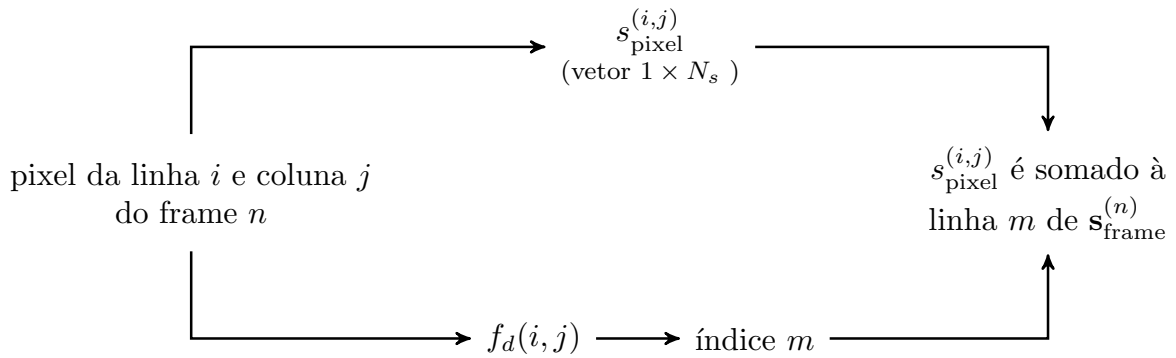


FIGURA 5.2 – Diagrama da implementação proposta para identificação da frequência Doppler do sinal recebido pelo radar.

cada *frame*.

$$s_{\text{cena}}[m, k] = \sum_{n=1}^{N_F} s_{\text{frame}}^{(n)}[m, k] \quad (5.7)$$

Para a reconstrução do sinal filtrado, a operação definida por (4.26) deve ser feita em cada linha da matriz s_{cena} , com o sinal do filtro sendo aquele expresso por (4.20). Consequentemente, o sinal filtrado também é uma matriz de dimensões $N_d \times N_s$.

A seguir serão apresentados dois cenários usados para a validação da solução proposta. Diferentemente das análises dos Capítulos 3 e 4, os cenários propostos envolvem alvos complexos, buscando simular um contexto mais operacional.

5.2 Identificação de aeronave de asas rotativas

De acordo com (SKOLNIK, 2001) e (CARVALHO et al., 2008), helicópteros podem ser identificados por radares devido ao seu eco característico gerado pelo movimento das pás do rotor principal. Além de máximos de amplitude periódicos, os desvios de frequência Doppler também oscilam em torno daquele causado pela fuselagem, visto que a pá que avança gera um desvio de frequência que se soma ao da fuselagem, enquanto que o desvio da pá que recua é contrário a ele. Para a simulação, foram montados dois cenários, cada um com um helicóptero. A única diferença entre as aeronaves era o número de pás do rotor principal: um possuía duas pás (bipá) e o outro, quatro pás (quadripá). As Figuras 5.3 e 5.4 mostram os helicópteros usados e os três últimos *frames* renderizados. Foram usadas 48 etapas de renderização, e cada uma delas foi constituída de um único *frame*, com Δt arbitrado como 1/12 do período de revolução do rotor principal, que girava a 40 rad/s.

A velocidade da fuselagem direção ao radar era de 60 m/s. Novamente, adotou-se a frequência de operação de 1,2 GHz, o que significa que o desvio em frequência associado ao movimento da fuselagem é de 480 Hz. Esse valor é representado pela linha tracejada da Figura 5.5, e a partir dela fica claro que o simulador identifica o desvio em frequência não só devido ao movimento da fuselagem, mas também às pás dos rotor.



FIGURA 5.3 – Helicóptero bipá usado na simulação (acima) e os três últimos *frames* renderizados (abaixo). Modelo retirado de (Free 3D, 2019b).



FIGURA 5.4 – Helicóptero quadripá usado na simulação (acima) e os três últimos *frames* renderizados (abaixo). Modelo adaptado de (Free 3D, 2019b).

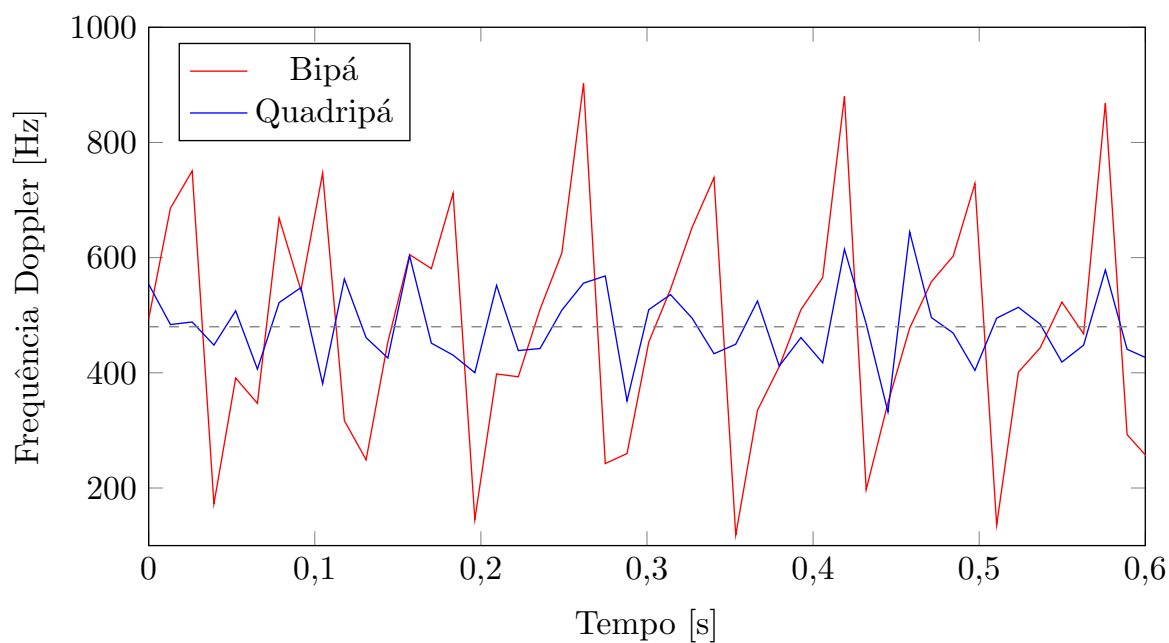


FIGURA 5.5 – Frequência Doppler percebida pelo radar.

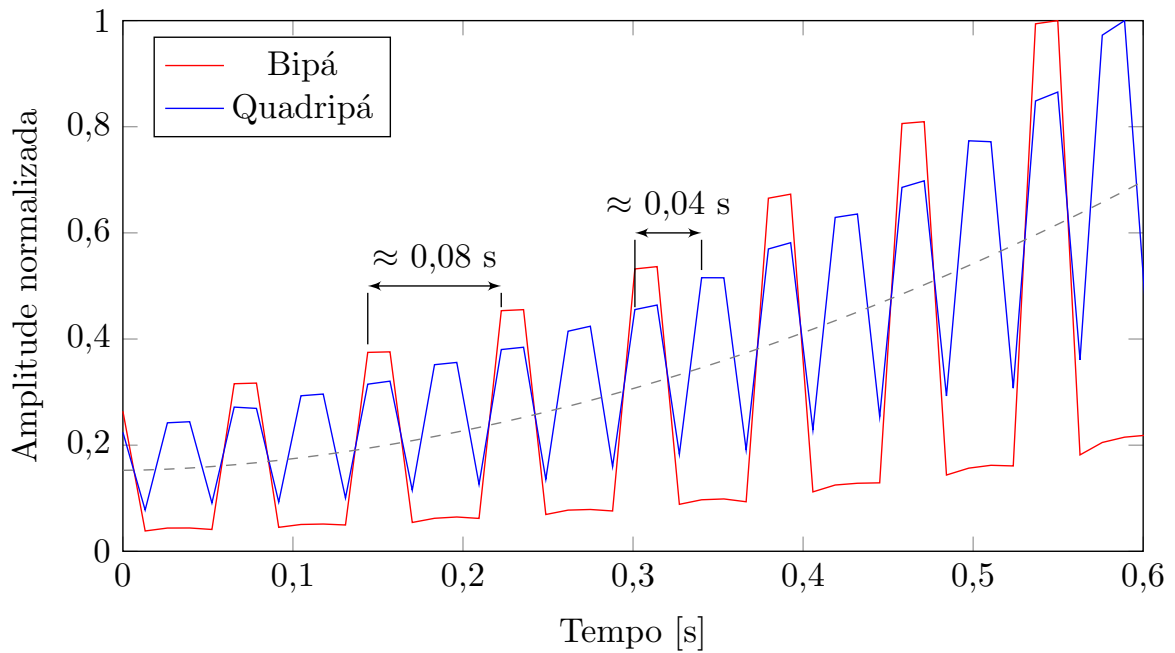


FIGURA 5.6 – Amplitude do sinal recebido pelo radar.

Os picos acima de 480 Hz estão relacionados à pá que avança, e aqueles abaixo, à pá que recua. Note também que a oscilação do helicóptero quadripá tem frequência maior. Isso fica mais claro quando se analisa a amplitude do sinal recebido, que é mostrada na Figura 5.6, em que a linha tracejada representa a amplitude do sinal associado à fuselagem. De acordo com (SKOLNIK, 2001), as oscilações devem ter período dado por:

$$T_{\text{heli}} = \frac{\omega}{2\pi P}, \quad (5.8)$$

em que ω é a rotação do rotor principal e P seu número de pás. Dessa forma, espera-se que as aeronaves bipá e quadripá possuam períodos de 0,0785 s e 0,0393 s, respectivamente. Da Figura 5.6 percebe-se a coerência entre esses valores e o resultado da simulação, com o período do bipá igual a duas vezes o do quadripá.

5.3 Detecção de múltiplos alvos em movimento

Para a verificação da capacidade do simulador de detectar múltiplos alvos em movimento, foi montada a cena mostrada na Figura 5.7, que também mostra o sistema de eixos coordenados em relação ao qual as posições iniciais e velocidades dos alvos, mostrados na Tabela 5.1, foram configuradas. Os valores de altitude e velocidade foram escolhidos com ordem de grandeza condizente com os parâmetros reais dos alvos adotados.

TABELA 5.1 – Dados adotados para a simulação da cena mostrada na Figura 5.7. Os valores apresentados têm como referência o sistema de eixos coordenados (x, y, z) mostrado na mesma imagem.

Alvo	Dados	
Avião	Posição inicial	(0, 45, 3) km
	Velocidade	(0, -150, 0) m/s
Helicóptero	Posição inicial	(-1, 20, 0,5) km
	Velocidade	(0, 60, 0) m/s
Blindado	Posição inicial	(2, 15, 0) km
	Velocidade	(10, 20, 0) m/s

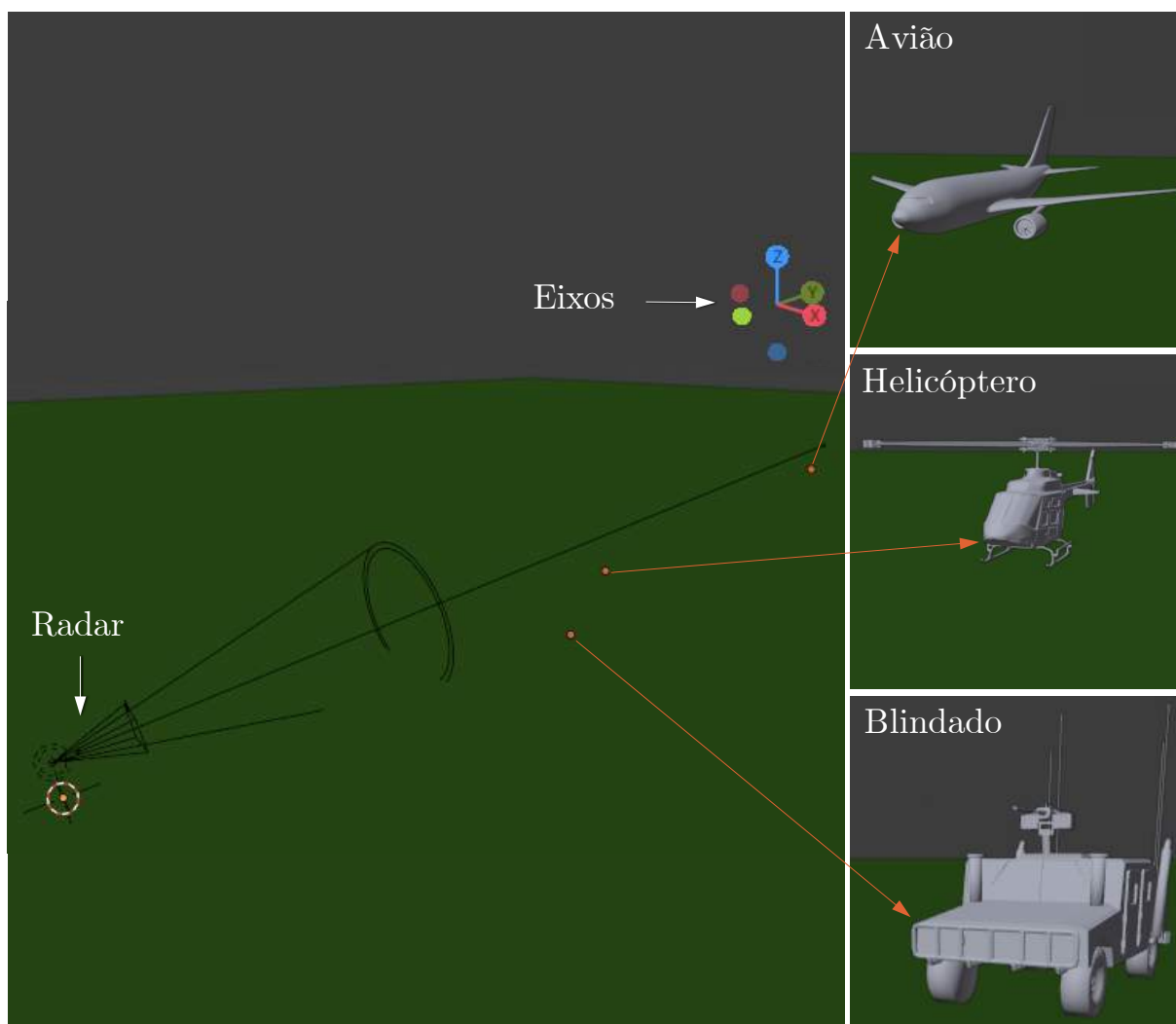


FIGURA 5.7 – Cena montada em Blender para verificar a detecção de múltiplos alvos em movimento. Fonte dos modelos 3D dos alvos: (Free 3D, 2019a), (Free 3D, 2019b) e (TurboSquid, 2017).

Note que o avião se aproxima do radar, enquanto o helicóptero e blindado se afastam. Além do mais, foi incluído um plano no cenário para simular o solo. A Figura 5.8 mostra o resultado obtido, e nele é possível identificar os três alvos, bem como suas respectivas frequências Doppler e distâncias em relação ao radar. Também é possível verificar um problema comum em sistemas radar, que é a presença de *clutter*, que são ecos indesejados que podem atrapalhar a identificação de alvos. Na simulação, o *clutter* se deve ao solo e ele pode ser identificado no sinal associado à frequência Doppler nula. A fim de eliminar esse efeito, pode-se suprimir o *clutter* através do descarte dos sinais de frequência Doppler igual a zero (i.e., os sinais associados a pixels de velocidade radial nula não são somados ao sinal final do *frame*). A Figura 5.9 compara a amplitude do sinal em função da distância para as situações com e sem a supressão de *clutter*, evidenciando o quanto ele pode prejudicar a identificação de alvos móveis. É interessante ressaltar que o desvio em frequência devido à rotação das pás do helicóptero não foi identificado nesse cenário pois sua renderização ocorreu em duas etapas, e para que a observação desse efeito fosse possível seria necessário um tempo maior de monitoramento do alvo (SKOLNIK, 2001).

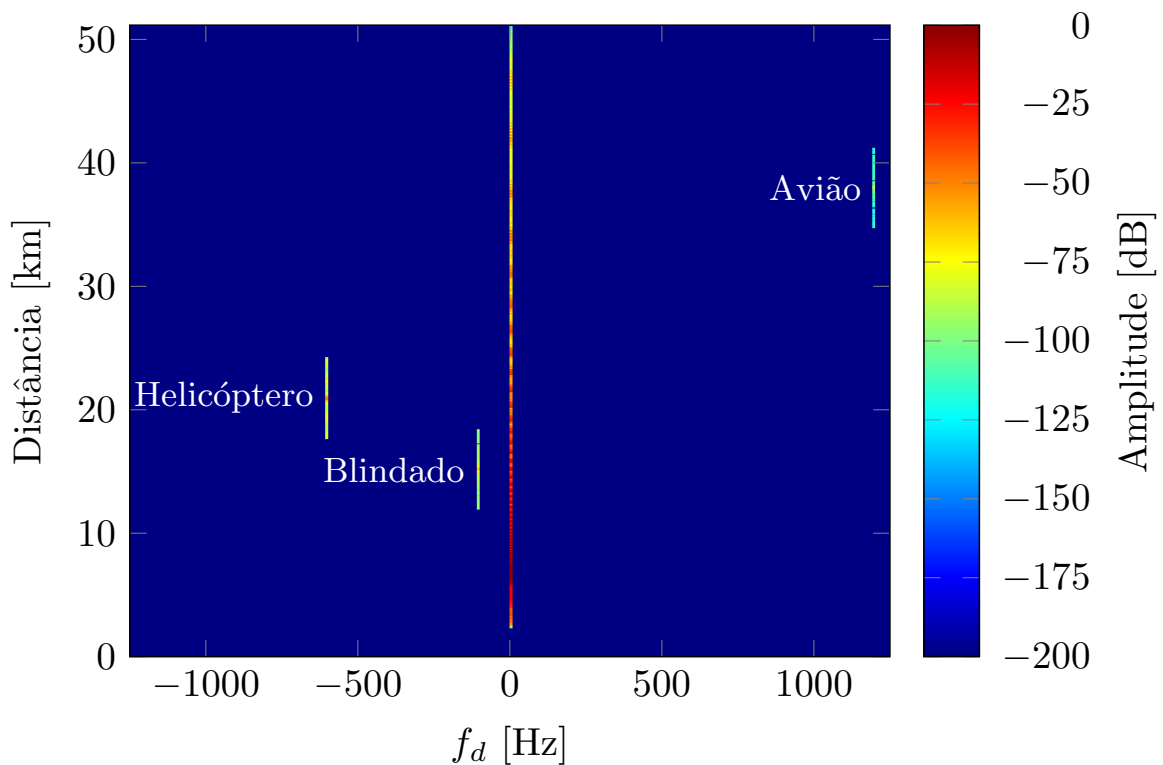
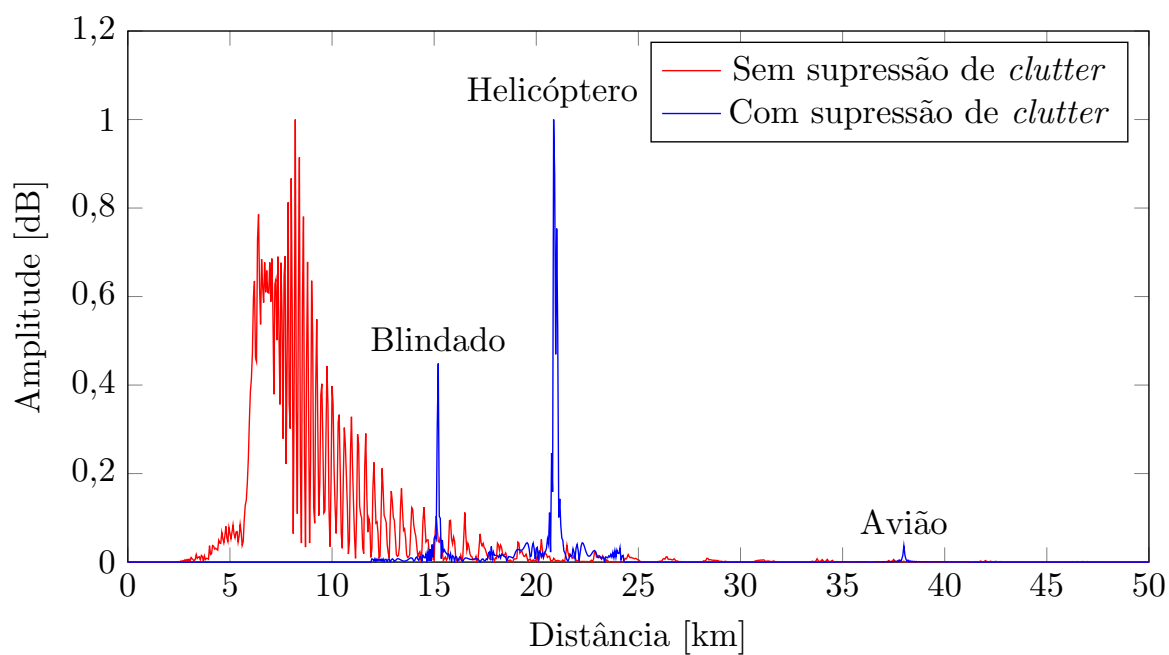


FIGURA 5.8 – Resultado obtido para a simulação da cena da Figura 5.7 após análise em MATLAB.

FIGURA 5.9 – Efeito da supressão de *clutter* no sinal recebido.

6 Radar Cross Section

A equação (3.3) expressa a forma mais fundamental da Equação do Radar e leva em conta parâmetros importantes do sistema transceptor e sua distância em relação ao alvo. O único parâmetro da equação que é totalmente determinado pelo alvo é a RCS, denotada por σ , que expressa a detectabilidade de um objeto através da medida da potência por ele irradiada em uma certa direção ao ser iluminado por uma onda incidente (KNOTT et al., 2004). As fórmulas e resultados mostrados neste relatório se referem ao RCS monoestático, que é aquele medido/calculado para uma situação em que transmissor e receptor se encontram na mesma posição.

Uma forma intuitiva de entender a definição de RCS é apresentada por (KNOTT et al., 2004) e mostrada na Figura 6.1. Seja D_i a densidade de potência incidente no alvo devido a uma antena transmissora distante, de forma que o alvo intercepta uma potência σD_i e, supondo que ele a irradia isotropicamente, a densidade de potência D_s a uma distância R do alvo é de:

$$D_s = \frac{\sigma D_i}{4\pi R^2}. \quad (6.1)$$

Para evitar efeitos relacionados ao campo próximo, toma-se $R \rightarrow \infty$, de forma que a definição de RCS é dada por:

$$\sigma = \lim_{R \rightarrow \infty} 4\pi R^2 \left(\frac{D_s}{D_i} \right). \quad (6.2)$$

A RCS de um determinado alvo depende, portanto, de sua geometria, material e posição em relação ao conjunto transceptor. A frequência e a polarização da onda transmitida pelo radar também influenciam nesse parâmetro.

A obtenção analítica da RCS é muito trabalhosa e quase sempre inviável (MAHAFZA, 2000). Existem, no entanto, técnicas que permitem estimar a RCS a partir de considerações feitas para cada contexto de análise. Em altas frequências, alguns métodos comuns são a Óptica Geométrica (GO), Óptica Física (PO), Teoria Geométrica da Difração (GTD), Teoria Física da Difração (PTD) e Método das Correntes Equivalentes (MEC) (KNOTT et al., 2004). No geral, a acurácia dessas aproximações melhora à medida que o(s)

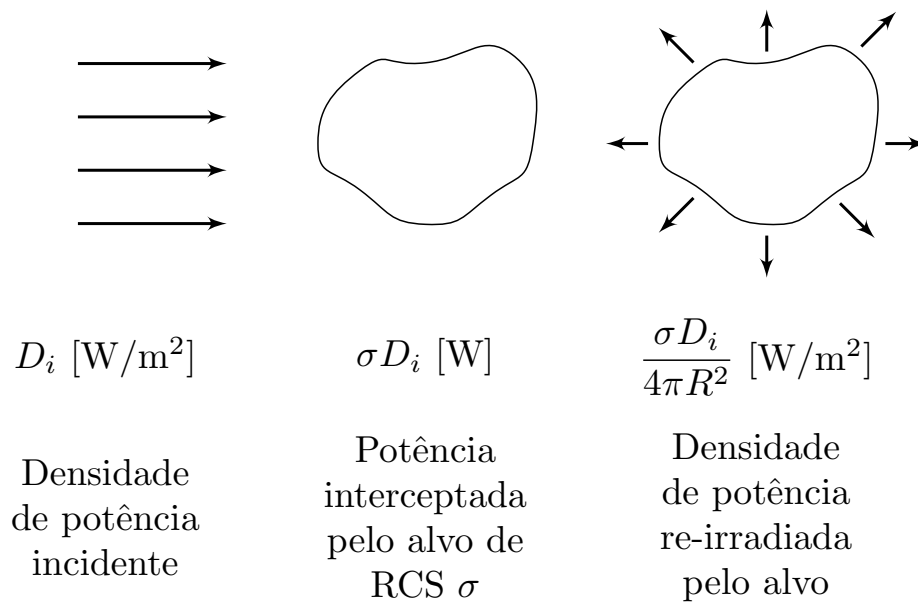


FIGURA 6.1 – Ilustração da definição intuitiva de RCS. Adaptado de (KNOTT et al., 2004).

alvo(s) se tornam eletricamente maiores, e alguns deles podem ser razoavelmente satisfatórios até mesmo para objetos de dimensão igual a um comprimento de onda (SKOLNIK, 2008). A quantidade de métodos evidencia o interesse em se estimar a RCS de alvos para altas frequências de operação, em especial para aplicações de diferenciação e classificação de alvos. Quando se pensa, por exemplo, em radares UHF, cujos comprimentos de onda são menores que 1 m, de forma que a maioria dos alvos aéreos tem dimensões que permitem a aplicação de técnicas de alta frequência, percebe-se a utilidade dessas aproximações. (KNOTT et al., 2004)

Os métodos de alta frequência simplificam a obtenção da RCS ao assumir que cada parte do alvo irradia energia de forma independente, ou seja, os campos induzidos em uma determinada porção do corpo se devem exclusivamente à onda incidente, e não dependem da energia irradiada por outras partes. Essa premissa não se aplica a casos específicos nos quais a reflexão especular de uma parte do corpo ilumina as outras, como é o caso, por exemplo, de quinas e cavidades reentrantes (KNOTT et al., 2004).

Dentre os métodos citados, dois deles merecem atenção especial. Um deles é a GO, na qual a onda incidente é modelada como um conjunto de raios que se propagam em trajetória retilínea. Ao atingir uma superfície, parte da energia desses raios é refletida, e a outra parte é transmitida. Esta parcela da energia, por sua vez, pode ser refratada e/ou absorvida pelo material. Os fenômenos citados são influenciados pelas características da superfície. Dessa descrição, percebe-se que, ao utilizar o Blender como um simulador radar, os resultados obtidos estão alinhados com o método da GO. Note também que essa técnica não prevê efeitos de difração e não contabiliza a influência da polarização da onda,

levando em conta apenas os fenômenos de reflexão, refração e absorção.

Outro método de interesse para a análise feita é a PO, visto que grande parte das fórmulas analíticas disponíveis na literatura são baseadas nesse tipo de aproximação. As expressões obtidas usando essa técnica também não dependem da polarização (SKOLNIK, 2008), no entanto, ela é capaz de modelar efeitos de difração, sendo uma alternativa mais precisa que a GO em planos e nas chamadas superfícies desenvolvíveis, que são aquelas que podem ser obtidas a partir de um plano, sem que para isso seja preciso deformá-lo (Brana, 2017). É o caso, por exemplo, de cilindros, cones e troncos de cone, mas não de esferas e elipsoides.

6.1 Geometrias Elementares

A seguir serão apresentadas as análises das RCS de geometria elementares e a comparação com os resultados fornecidos pelas fórmulas analíticas baseadas em PO, retiradas de (MAHAFZA, 2000) e (SKOLNIK, 2008), e também com aqueles por obtidos com um simulador pago, o FEKO[®], da empresa Altair Engineering.

Nas simulações em Blender, fixou-se o coeficiente de extinção, o índice de refração e a especularidade, e a rugosidade especular foi variada entre 0,1 e 1 com passo 0,1. O coeficiente de extinção foi fixado em um valor elevado, visto que em condutores perfeitos a condutividade é infinita, de modo que a energia incidente não consegue se propagar em seu interior (KNOTT et al., 2004). O índice de refração também foi mantido em um valor alto para que um número máximo de raios incidentes fossem refletidos. A especularidade foi fixada em 1, a fim de que toda a reflexão fosse especular. Dessa forma, o valor de rugosidade difusa não interfere no resultado. Como o Blender não recebe informações diretas de frequência para computar todos os efeitos de reflexão e transmissão, a ideia desse método é mapear o comportamento de certos objetos em um dado valor de frequência nas características da superfície. Para determinar qual é a configuração de material mais adequada, toma-se a que minimiza o erro quadrático médio em relação ao valor fornecido pelas fórmulas analíticas.

Para uma validação inicial com simulação de condutores perfeitos, optou-se por variar apenas um parâmetro, o que não é verdade em um material mais realista, cuja análise seria mais trabalhosa e demandaria tempo maior de processamento pelo *software*.

Outro ponto importante é que, conforme previsto na equação (6.2), a RCS é calculada com o alvo no campo distante das antenas do radar, por isso todas as distâncias foram medidas com o centro do alvo afastado em cerca de 100 m do radar. Como a frequência adotada foi de 1,2 GHz, essa distância corresponde a cerca de 400 comprimentos de onda.

Ressalta-se também que, exceto nas situações em que há menção explícita do intervalo em que a minimização do erro quadrático médio foi feita, a faixa de valores usadas corresponde àquela no eixo das abscissas do respectivo gráfico.

6.1.1 Esfera

A esfera perfeitamente condutora é a geometria de RCS mais simples de se obter analiticamente, visto que sua simetria puramente radial faz com que essa grandeza independa do ângulo de incidência. Ainda assim sua fórmula é bastante complexa, sendo expressa em função de um somatório com funções de Bessel de primeira e segunda ordem. A curva correspondente à essa equação é mostrada na Figura 6.2, e pode ser dividida em três regiões distintas. Uma delas é a região de Rayleigh, na qual o comprimento de onda é bem maior que a circunferência da esfera e a RCS é proporcional ao quadrado da frequência. Na região de Mie, ou região ressonante, a esfera possui dimensões da mesma ordem de grandeza do comprimento de onda, o que leva à interferência entre as ondas refletidas pela parte frontal do alvo e as *creeping waves* que passam pela parte traseira do objeto e retornam para o radar. Essa interferência é o que causa as oscilações características dessa região. Na região óptica, onde o comprimento de onda é bem maior que a circunferência da esfera, é que as aproximações de alta frequência, como a GO e a PO, são válidas. Nesse caso, a RCS é constante e igual à área projetada da esfera.

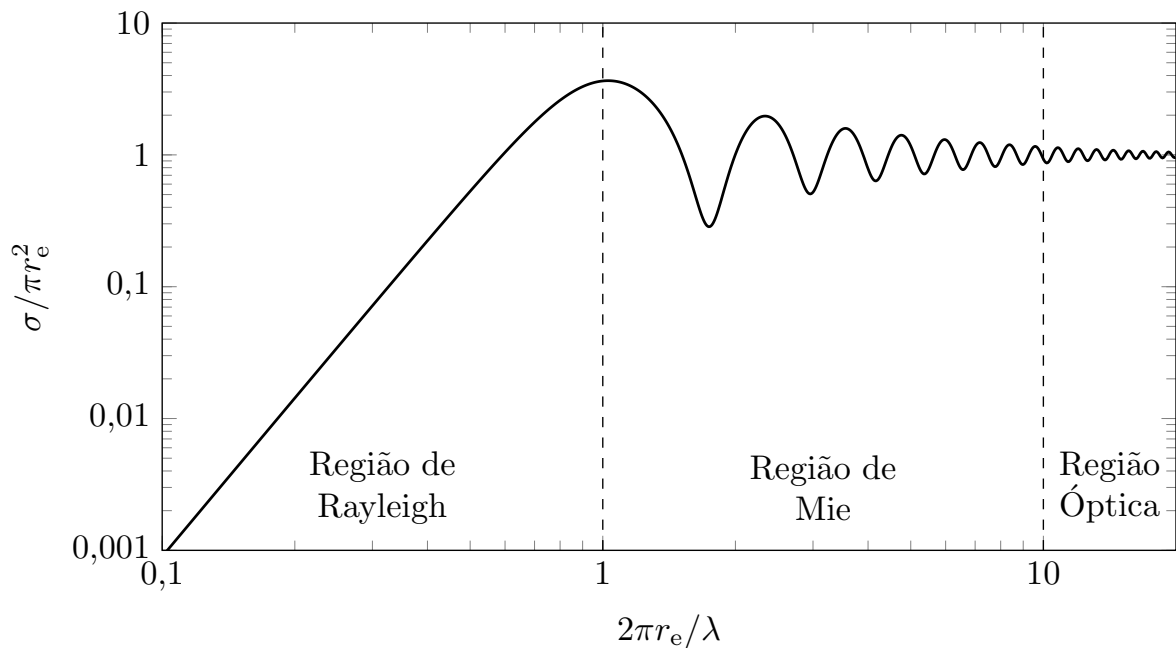


FIGURA 6.2 – RCS de uma esfera de raio r_e em função da razão entre sua circunferência e o comprimento de onda. Adaptado de (SKOLNIK, 2001).

No caso das simulações apresentadas, a esfera foi usada para calibração, algo comum

em experimentos reais (KNOTT et al., 2004). Para cada valor de rugosidade especular, a RCS foi obtida no Blender e usada como fator de conversão para as demais geometrias para que o valor obtido fosse absoluto, e não normalizado.

6.1.2 Elipsoide

Para um elipsoide de seção circular de raio a_e no plano xy e semi-eixo vertical c_e , como aquele mostrado na Figura 6.3, a RCS é dada por:

$$\sigma = \frac{\pi a_e^4 c_e^2}{[a_e^2 \sin^2 \theta + c_e^2 \cos^2 \theta]^2}. \quad (6.3)$$

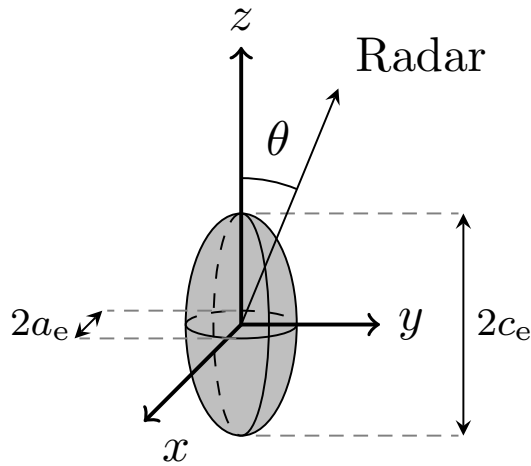


FIGURA 6.3 – Referência para os valores de θ e das dimensões da equação (6.3).

Na Figura 6.4 é mostrado o resultado obtido a partir dessa equação e das simulações para $a_e = 0,5$ m e $c_e = 1$ m. Os resultados se tornam mais distintos à medida que o ângulo de incidência se afasta daquele em que a reflexão máxima ocorre, mas o Blender consegue reproduzir razoavelmente bem o comportamento da RCS, respeitando os pontos de incidência máxima e mínima para essa grandeza.

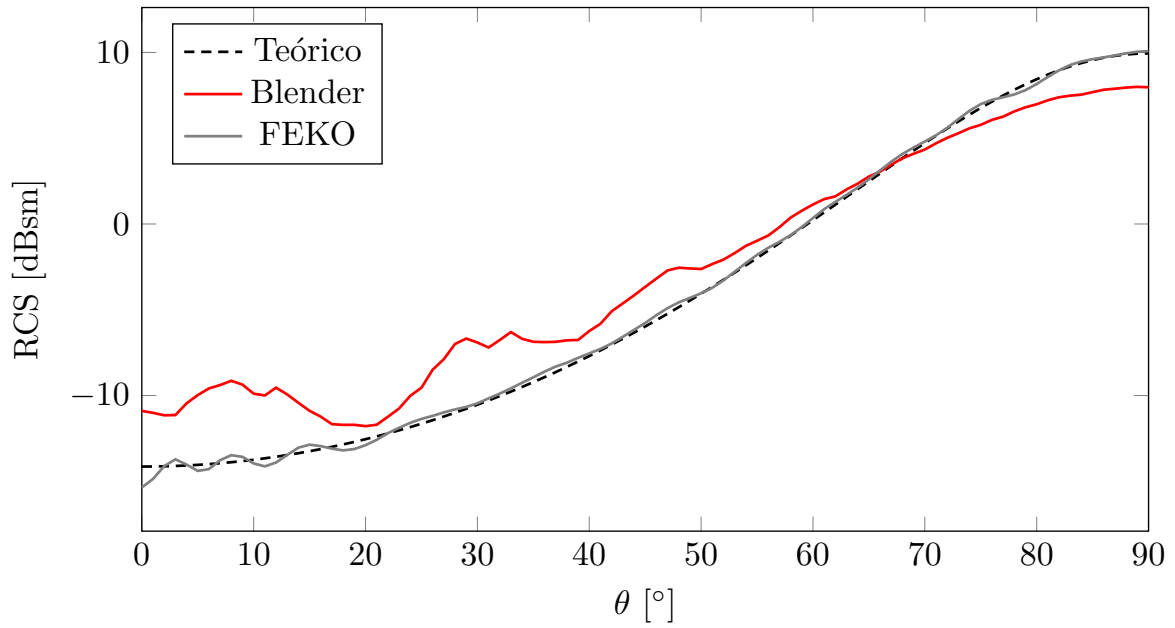


FIGURA 6.4 – RCS do elipsoide em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,4.

6.1.3 Cilindro e tronco de cone

As Figuras 6.5 e 6.6 mostram o cilindro e o cone usados para as simulações. Para o cilindro adotou-se $r_c = 0,5$ m e $H_c = 1$ m, e para o tronco de cone as medidas foram $r_{t_2} = 1$ m, $r_{t_1} = 0,5$ m e $\alpha_t = 15^\circ$, que resultou em $H_t = (r_{t_2} - r_{t_1}) / \tan \alpha_t \approx 1,87$ m. As equações para a RCS dessas geometrias são apresentadas em (6.4) e (6.5), respectivamente. É importante ressaltar que essas expressões não levam em conta as bases desses sólidos, apenas sua superfície lateral.

$$\sigma = \frac{2\pi r_c H_c^2}{\lambda} \text{sinc}^2 \left(\frac{2H_c}{\lambda} \sin \theta \right) \quad (6.4)$$

$$\sigma = \begin{cases} \frac{\lambda r_{t_2}}{8\pi \sin \theta} \tan^2 (\theta - \alpha_t) & \text{se } 0^\circ < \theta < 90^\circ + \alpha_t \\ \frac{8\pi (r_{t_2}^{3/2} - r_{t_1}^{3/2})^2}{9\lambda} \frac{1}{\sin^2 \alpha_t \cos \alpha_t} & \text{se } \theta = 90^\circ + \alpha_t \\ \frac{\lambda r_{t_1}}{8\pi \sin \theta} \tan^2 (\theta - \alpha_t) & \text{se } 90^\circ + \alpha_t < \theta < 180^\circ \end{cases} \quad (6.5)$$

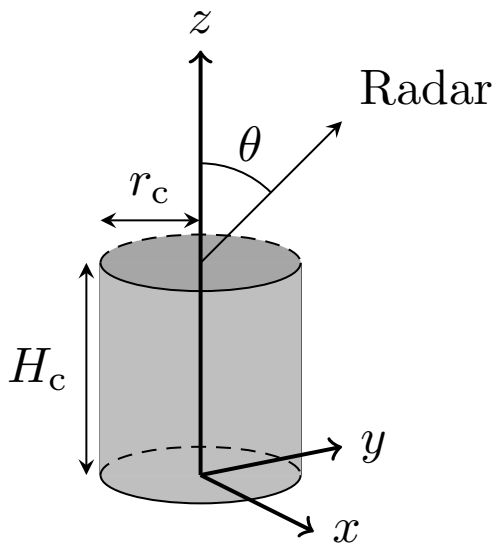


FIGURA 6.5 – Referência para os valores de θ e das dimensões da equação (6.4).

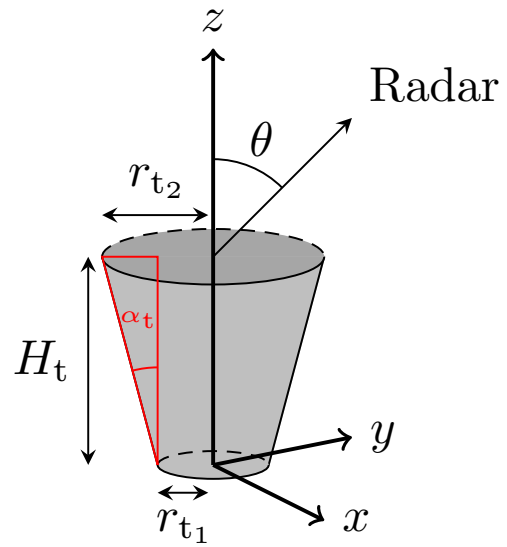


FIGURA 6.6 – Referência para os valores de θ e das dimensões da equação (6.5).

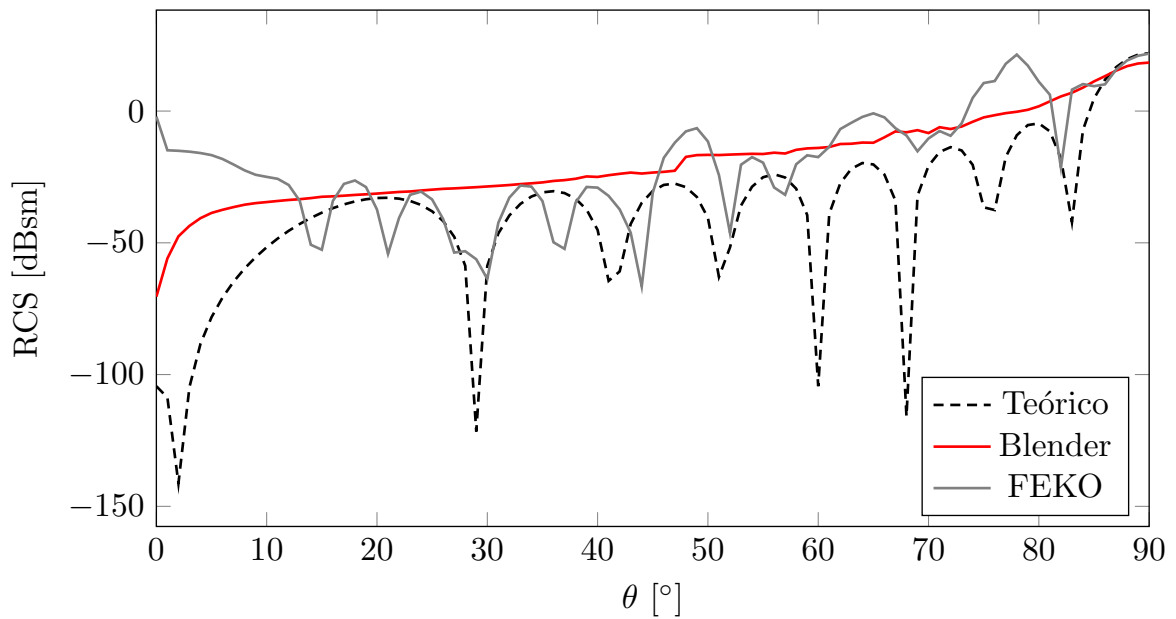


FIGURA 6.7 – RCS do cilindro em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,3.

A Figura 6.7 mostra o resultado obtido para o cilindro. Note que o FEKO consegue acompanhar melhor as ondulações da curva teórica, enquanto que o Blender parece seguir o envelope delas. Já para o tronco de cone, cujo resultado é mostrado na Figura 6.8, nenhum dos simuladores parece se aproximar da curva teórica, em especial para $|\theta| < 80^\circ$. Nem mesmo próximo da reflexão máxima, em $|\theta| = \alpha_t + 90^\circ = 105^\circ$, o resultado do

Blender (curva vermelha, descrita como Blender 1 na Figura 6.8) se aproximou das outras. Percebendo isso, a minimização do erro quadrático médio foi feita em um intervalo mais restrito, entre 75° e 135° , o que levou a um resultado (curva azul, descrita como Blender 2 na Figura 6.8) mais condizente não só com a curva teórica para incidências próximas à especular, mas também com a curva do FEKO em todo o intervalo.

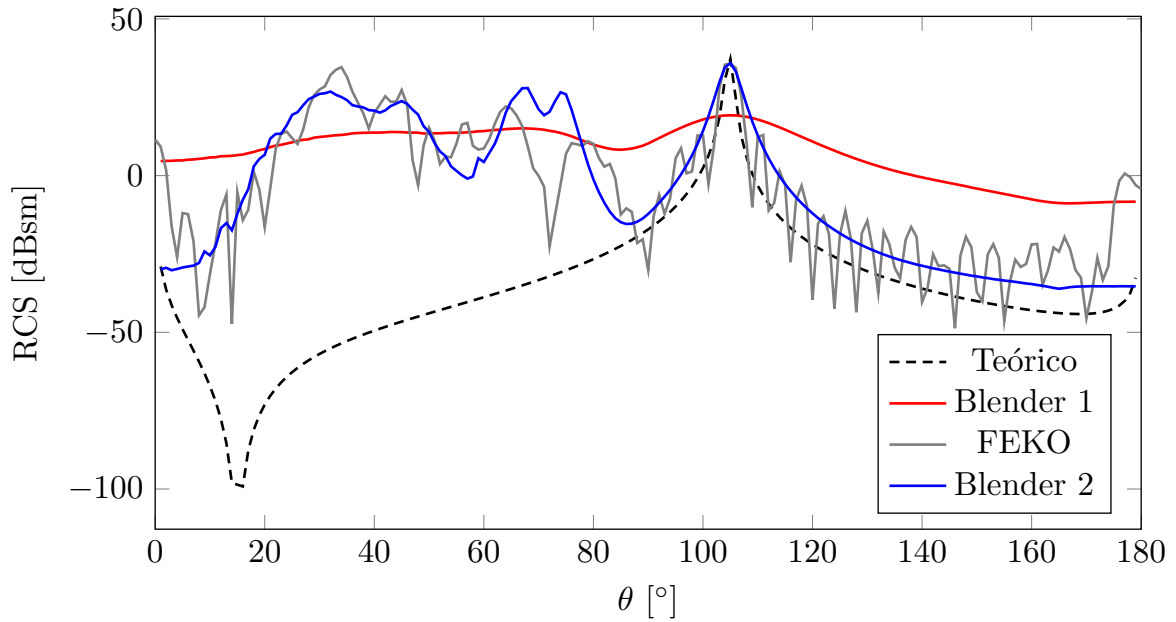


FIGURA 6.8 – RCS do tronco de cone em função do ângulo polar. O resultado apresentado na curva vermelha (Blender 1) corresponde a uma rugosidade especular igual a 0,5, enquanto que a curva azul (Blender 2) foi obtida para uma rugosidade especular de 0,2.

6.1.4 Planos retangular, circular e triangular

Para o plano retangular (Figura 6.9), adotou-se $\ell = w = 0,5$ m. Sua RCS em função dessas dimensões quando a incidência ocorre no plano xz ($\varphi = 0^\circ$) é dada por:

$$\sigma = \frac{4\pi\ell^2w^2}{\lambda^2} \operatorname{sinc}^2\left(\frac{2\ell}{\lambda} \sin\theta\right) \cos^2\theta. \quad (6.6)$$

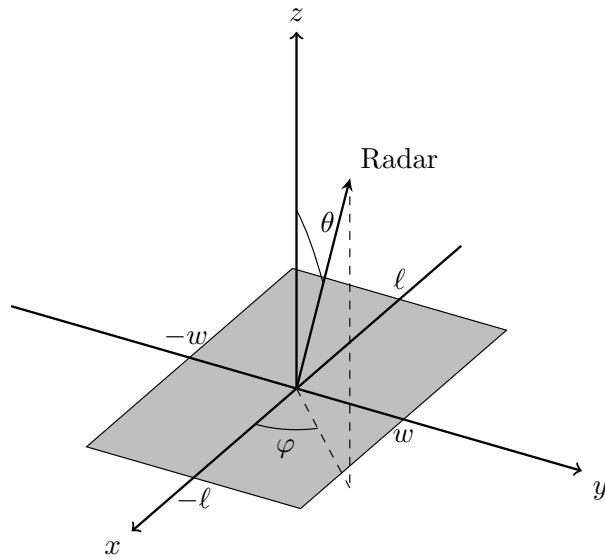


FIGURA 6.9 – Referência para as dimensões e os valores de θ da equação (6.6) e da Figura 6.12.

Já para um plano circular de raio r_p , como aquele da Figura 6.10, a RCS é dada pela equação (6.7), na qual $k = 2\pi/\lambda$ e J_1 é a função de Bessel de primeiro tipo e ordem 1. Na análise feita adotou-se $r_p = 1$ m para esse objeto.

$$\sigma = \begin{cases} \frac{4\pi^3 r_p^4}{\lambda^2} & \text{se } \theta = 0^\circ \\ \frac{16\pi^3 r_p^4}{\lambda^2} \left(\frac{J_1(2kr_p \sin \theta)}{2kr_p \sin(\theta)} \right)^2 \cos^2 \theta & \text{se } 0^\circ < \theta < 90^\circ \end{cases} \quad (6.7)$$

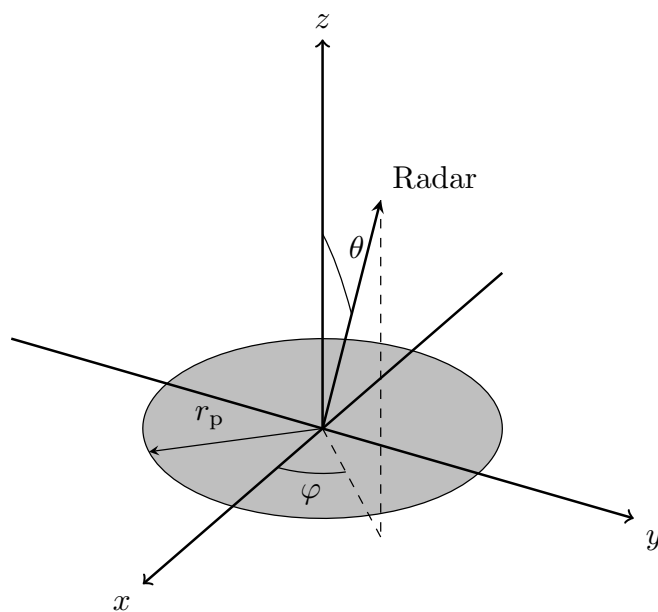


FIGURA 6.10 – Referência para as dimensões e os valores de θ da equação (6.7) e da Figura 6.13.

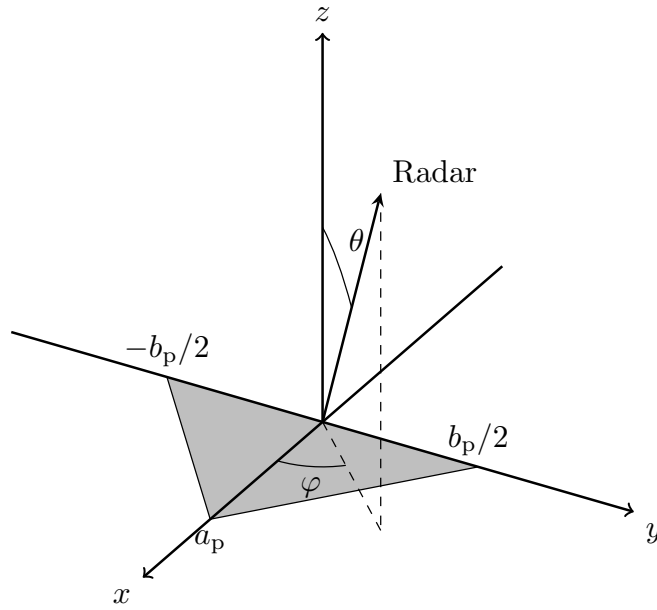


FIGURA 6.11 – Referência para as dimensões e os valores de θ das equações (6.10) e (6.11), e das Figuras 6.14 e 6.15.

E para a placa triangular da Figura 6.11, analisou-se a incidência em $\varphi = 0^\circ$ e $\varphi = 90^\circ$ em um triângulo equilátero de $b_p = 1$ m e $a_p = \sqrt{3}/2$ m. A RCS para esses dois valores de ângulo azimutal é dada pelas equações (6.10) e (6.11), respectivamente. Nelas, α_p e β_p são dados por:

$$\alpha_p = \frac{2\pi a_p}{\lambda} \sin \theta, \quad (6.8)$$

$$\beta_p = \frac{2\pi b_p}{\lambda} \sin \theta. \quad (6.9)$$

$$\sigma = \begin{cases} \frac{\pi a_p^2 b_p^2}{\lambda^2} & \text{se } \theta = 0^\circ \\ \frac{\pi a_p^2 b_p^2}{\lambda^2} \cos^2 \theta \left[\text{sinc}^4 \left(\frac{\alpha_p}{\pi} \right) + \left(\frac{\sin(2\alpha_p) - 2\alpha_p}{2\alpha_p^2} \right)^2 \right] & \text{se } 0 < |\theta| < 90^\circ \end{cases} \quad (6.10)$$

$$\sigma = \begin{cases} \frac{\pi a_p^2 b_p^2}{\lambda^2} & \text{se } \theta = 0^\circ \\ \frac{\pi a_p^2 b_p^2}{\lambda^2} \cos^2 \theta \text{sinc}^4 \left(\frac{\beta_p}{2\pi} \right) \text{ para} & \text{se } 0 < |\theta| < 90^\circ \end{cases} \quad (6.11)$$

As Figuras 6.12 e 6.13 mostram os resultados obtidos para os planos retangular e circular, respectivamente. Para a placa triangular com incidência em $\varphi = 0^\circ$ e $\varphi = 90^\circ$, é possível observar os valores obtidos para a RCS nas Figuras 6.14 e 6.15. Exceto pela

incidência em uma placa triangular com ângulo azimutal nulo, que gerou um resultado razoavelmente próximo dos demais, em todos os outros casos a simulação do Blender não tem as ondulações previstas pela equação teórica e obtidas no FEKO, mas parece seguir o envelope dessas curvas.

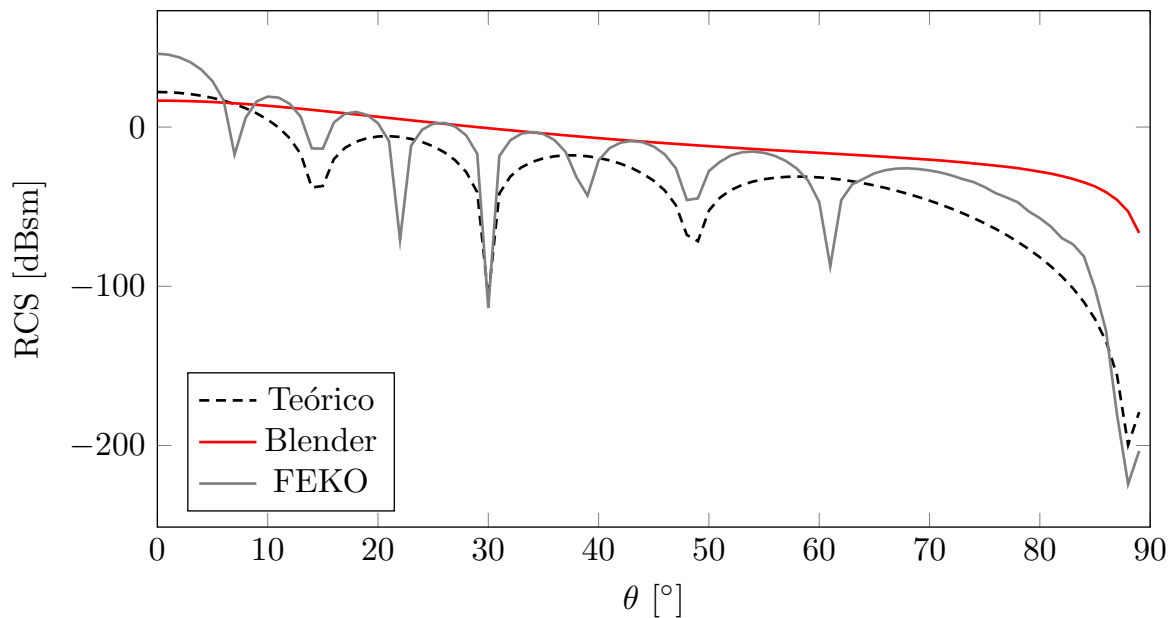


FIGURA 6.12 – RCS da placa retangular em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,6.

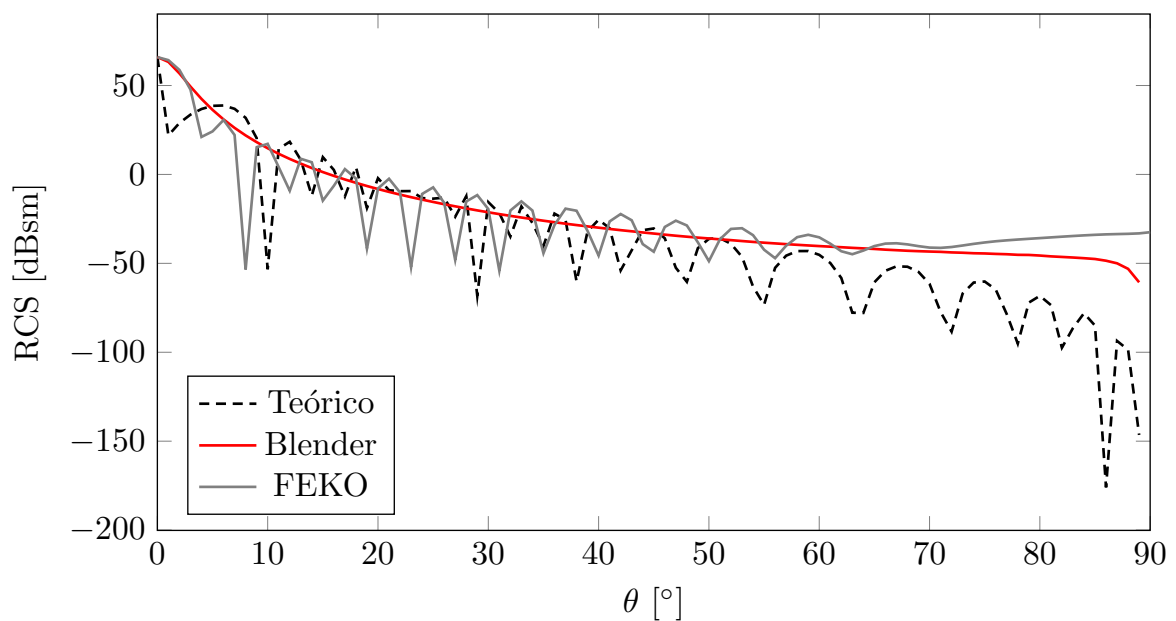


FIGURA 6.13 – RCS da placa circular em função do ângulo polar. O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,2.

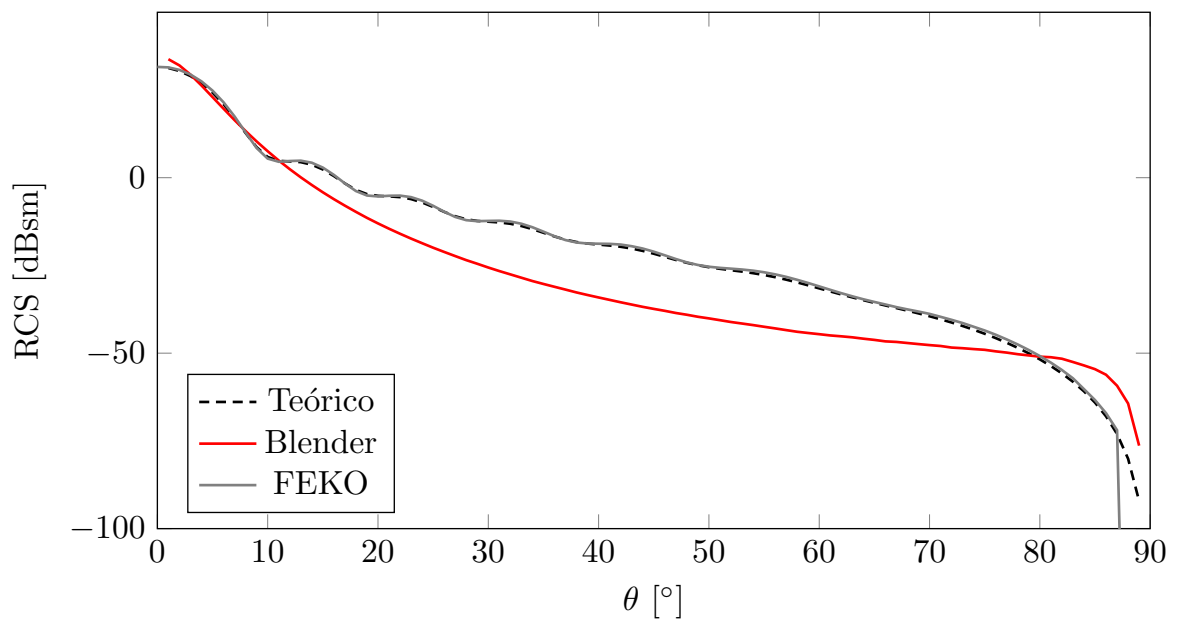


FIGURA 6.14 – RCS da placa triangular em função do ângulo polar para um ângulo azimutal de 0° . O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,3.

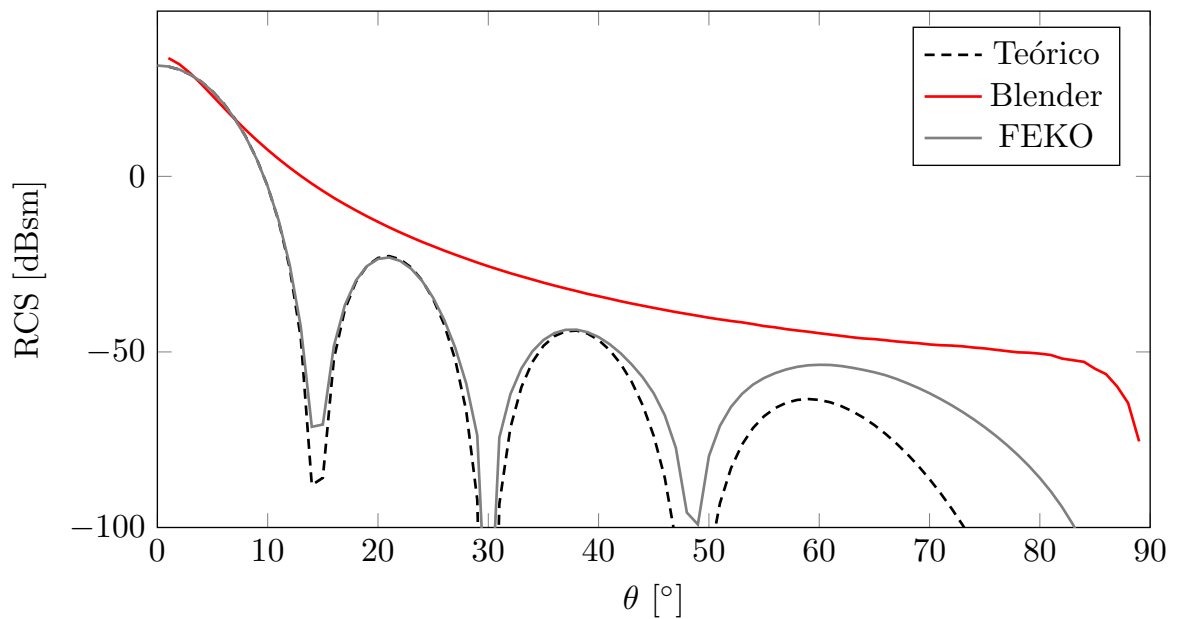


FIGURA 6.15 – RCS da placa triangular em função do ângulo polar para um ângulo azimutal de 90° . O resultado apresentado para o Blender corresponde a uma rugosidade especular de 0,3.

6.1.5 Diedro

Para um diedro de duas faces retangulares, idênticas e perpendiculares entre si, como o da Figura 6.16, a RCS em função do ângulo azimutal pode ser aproximada pela equação (6.12). O resultado obtido para uma simulação com $a_d = b_d = 1$ m é mostrado no gráfico da Figura 6.17.

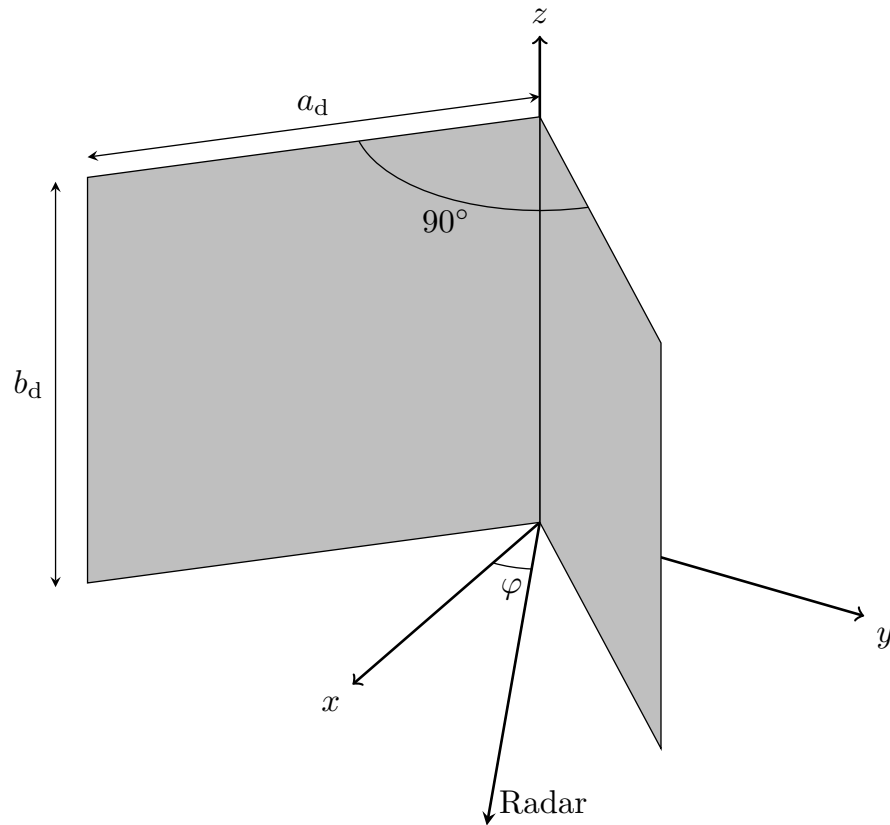


FIGURA 6.16 – Referência para as dimensões e os valores de φ da equação (6.12) e da Figura 6.17.

$$\sigma = \begin{cases} \frac{4\pi a_d^4}{\lambda^2} \left(2 |\sin \varphi| + \left| \text{sinc} \left(\frac{2a_d}{\lambda} \sin \varphi \right) \right| \right)^2 & \text{se } 0^\circ \leq |\varphi| < 45^\circ \\ \frac{8\pi a_d^2 b_d^2}{\lambda^2} & \text{se } |\varphi| = 45^\circ \\ \frac{4\pi a_d^4}{\lambda^2} \left(2 |\cos \varphi| + \left| \text{sinc} \left(\frac{2a_d}{\lambda} \cos \varphi \right) \right| \right)^2 & \text{se } 45^\circ < |\varphi| < 90^\circ \end{cases} \quad (6.12)$$

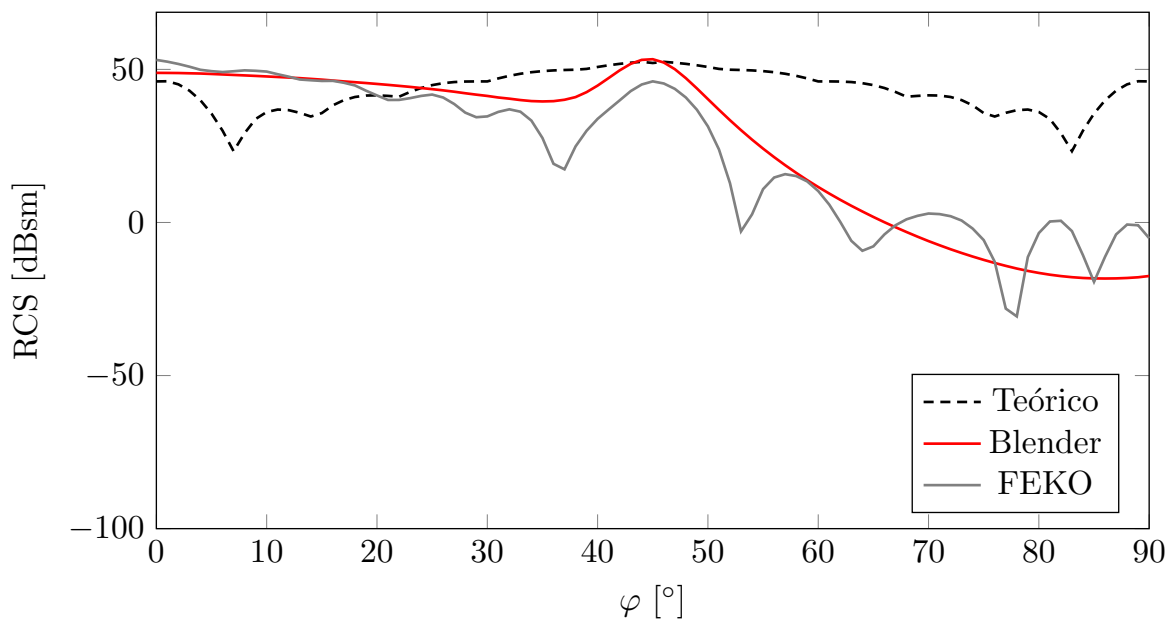


FIGURA 6.17 – RCS do diedro em função do ângulo azimutal para um ângulo polar de 90° .

Nessa situação o Blender acompanha bem o envelope da curva do FEKO para ângulos inferiores a 60° , e apesar de não apresentar ondulações, também identifica um padrão de RCS mais alargado para o diedro quando comparado com um plano retangular, que não sofre os efeitos de iluminação devido à reflexão em outra face. Ambas as curvas só acompanham bem a curva teórica para $\varphi < 45^\circ$, onde ocorre o máximo de reflexão para os três casos. Isso não deve ser motivo de grande preocupação, pois resultados experimentais (JAYASRI et al., 2018) mostram que, na prática, o comportamento se assemelha ao fornecido pelo FEKO, que está razoavelmente próximo do resultado obtido no Blender.

A análise das geometrias elementares mostrou que o Blender consegue apresentar razoavelmente bem o comportamento dos alvos analisados, no sentido de acompanhar os pontos de mínimo e máximo e, com a calibração preliminar, atingir ordens de grandeza parecidas com as esperadas. Nos pontos em que a reflexão é máxima, o desempenho do simulador é melhor. Isso ocorre pois dominância desse fenômeno nesses pontos diminui a influência daqueles efeitos que o Blender não simula, como por exemplo a difração, que causa as ondulações nas curvas teóricas e do FEKO.

Assim como foi pontuado na análise da resolução em *range* feita na Seção 4.2, o Blender possui implementação otimizada para o espectro visível, o que causa certas discordâncias quando se analisa frequências mais altas. Outro ponto importante e que pode ter levado a diferenças nos resultados fornecidos pelos simuladores é a maneira como cada um discretiza a superfície dos objetos para a realização dos cálculos: o FEKO divide o alvo em triângulos, enquanto que o Blender traça anéis longitudinais e transversais, e a partir dos pontos de

interseção deles efetua a discretização em planos trapezoidais. Ainda estabelecendo uma comparação entre simuladores, é importante mencionar que o tempo renderização no Blender foi significativamente superior ao do FEKO, o que pode ser um problema em análises que envolvam um intervalo muito grande de ângulos de incidência.

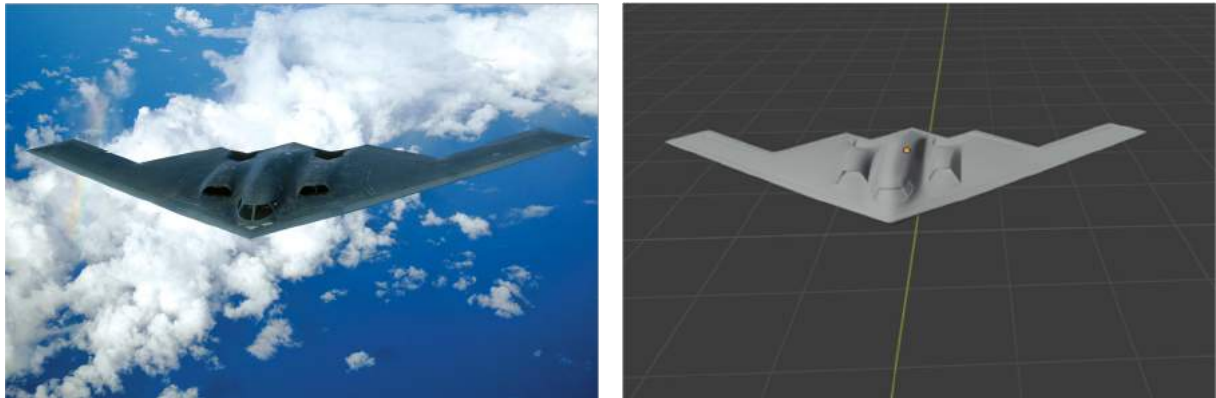
6.2 Comparação da RCS de duas aeronaves

A possibilidade de detecção de veículos de ataque por sistemas radar motivou o desenvolvimento de tecnologias *stealth*, que permitem o controle do eco gerado por um certo alvo e conseqüente redução de sua RCS e detectabilidade. Em aeronaves isso pode ser feito através da divisão da superfície em vários planos, o que faz com que os lóbulos especulares se tornem mais estreitos, ou ainda com o emprego de superfícies curvas, que reduzem a intensidade desses lóbulos (SKOLNIK, 2008). Este é o caso, por exemplo, do caça F-22 Raptor, mostrado na Figura 6.18, cujo modelo usado para a obtenção da RCS no Blender é mostrado ao lado. O bombardeiro B-2 Spirit, da Figura 6.19, também emprega esse tipo de tecnologia e possui dimensões bem maiores que aquelas do F-22, e sua RCS foi obtida usando o modelo mostrado na mesma Figura.

Para que houvesse um parâmetro de comparação do efeito da geometria do alvo, também foi obtida a RCS de uma aeronave comercial, o A310, mostrado na Figura 6.20, e todas as aeronaves foram posicionadas à mesma distância do radar, constituídas do mesmo material e de dimensões semelhantes às das aeronaves reais, conforme dados obtidos em (Lockheed Martin, 2020), (US Air Force, 2015) e (Airbus, 2020).



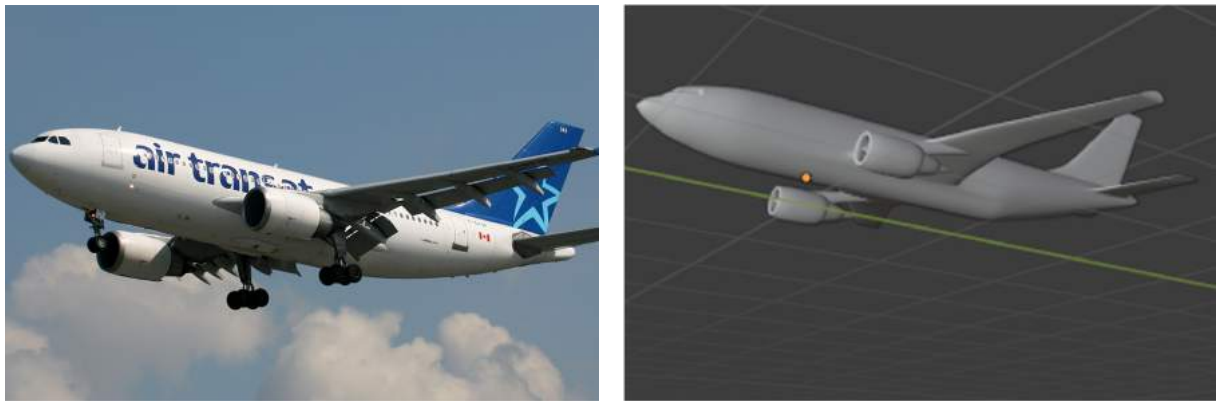
FIGURA 6.18 – Caça *stealth* F-22 Raptor real (a) e seu modelo no Blender (b). Fontes: (ROSSO, 2013), (TurboSquid, 2015).



(a)

(b)

FIGURA 6.19 – Bombardeiro *stealth* B-2 Spirit real (a) e seu modelo no Blender (b). Fontes: (Davis III, 2006), (Free 3D, 2020).



(a)

(b)

FIGURA 6.20 – Avião comercial A310 real (a) e seu modelo no Blender (b). Fontes: (WEDELSTAEDT, 2009), (Free 3D, 2019a).

A Figura 6.21 mostra o resultado obtido. O ângulo de incidência nulo corresponde ao nariz da aeronave, logo os máximos de reflexão para o A310 ocorrem para incidência perpendicular à fuselagem. Para ângulos de incidência próximos de 30° e 330° , identifica-se também o eco das naceles dos motores do A310, algo comum de se observar na RCS de aeronaves com esse tipo de configuração (SKOLNIK, 2001).

De acordo com (SKOLNIK, 2008), a alteração da geometria de uma aeronave com o objetivo de diminuir seu eco só faz sentido quando se é capaz de determinar a direção mais provável de uma ameaça, o que pode explicar os diferentes comportamentos observados na Figura 6.21. Para um avião convencional, grande parte da RCS se concentra na região frontal. Para o F-22, ela se distribui ao longo dos ângulos de incidência e, por ter dimensões menores, apresenta RCS de magnitude significativamente menor. É interessante observar, no entanto, que o B-2 Spirit possui uma envergadura de 52,12 m, maior que a do A310,

que mede 43,90 m, mas ainda assim sua RCS não é tão elevada, estando concentrada na parte traseira da aeronave.

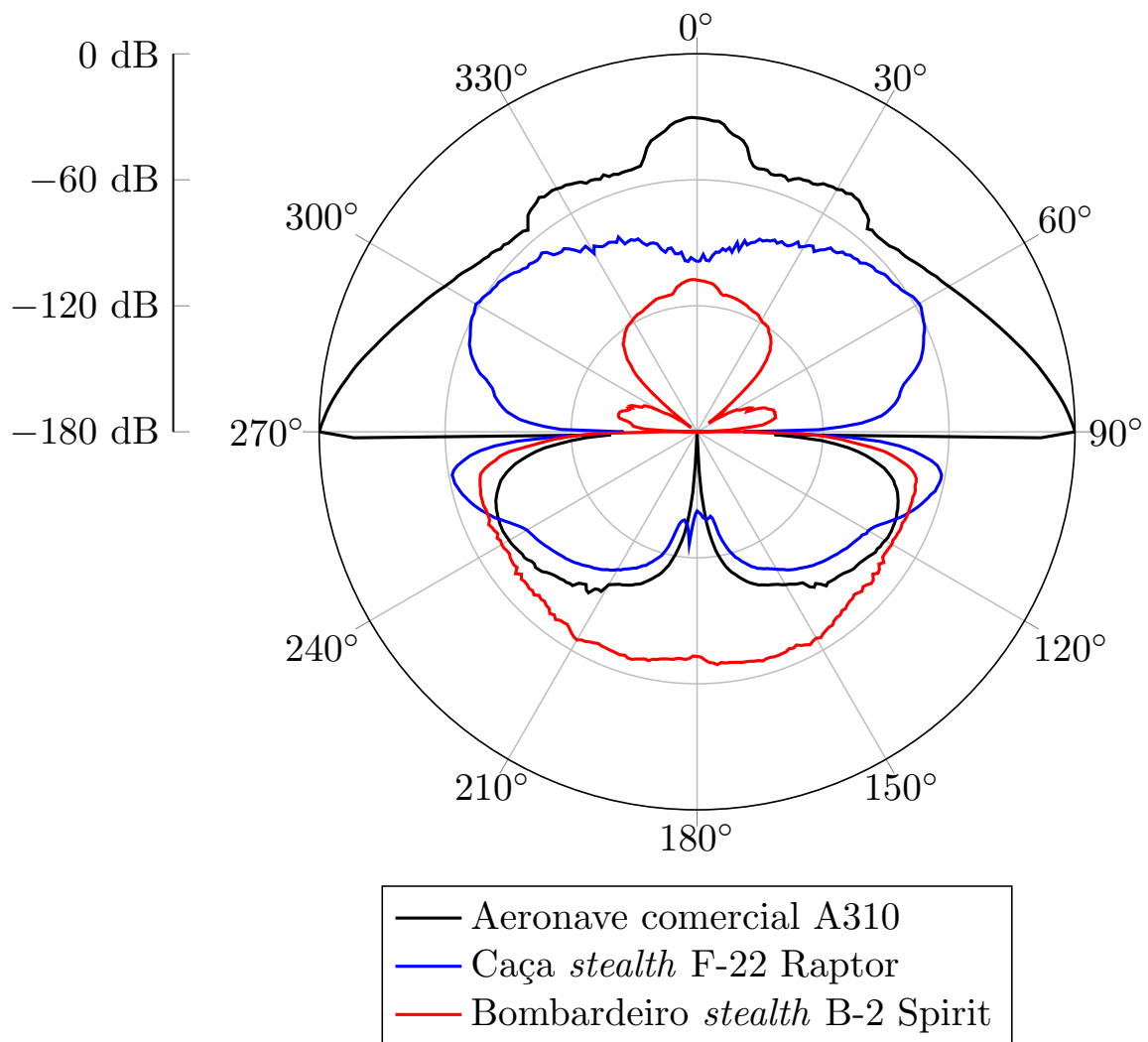


FIGURA 6.21 – Resultado obtido para a RCS do F-22 Raptor, do B-2 Spirit e do A310.

Apesar de os modelos em Blender não corresponderem fidedignamente às aeronaves reais, o que é inviável não só pela complexidade de suas geometrias, mas também pelo grau de sigilo dessa informação, e de não ter sido usado material absorvedor no F-22 e no B-2, suas RCS possuem formato nitidamente diferente da RCS do A310, evidenciando o potencial do simulador na análise da geometria de alvos mais complexos, mesmo que ele tenha suas limitações quanto à modelagem da difração e comportamento sub-ótimo fora do espectro da luz visível.

Outro ponto interessante é que, da mesma forma que os modelos usados no Blender para as medições de RCS foram obtidos gratuitamente na internet, é possível encontrar vários outros tipos de objetos disponíveis. No caso de um alvo mais complexo, ou que necessite de um nível de detalhe maior, existem também as opções pagas. Estas, no

entanto, podem ter valor elevado, e outra alternativa seria a confecção do objeto em Blender, o que, por sua vez, exige tempo e conhecimento técnico avançado do *software*.

7 Conclusão

Neste trabalho foi possível validar a implementação proposta para um simulador radar através da análise de uma série de conceitos teóricos fundamentais. O primeiro deles é a forma simplificada da Equação do Radar, cuja verificação evidenciou concordância com a teoria e adequação do método usado para a codificação das informações de interesse nas componentes RGB.

A análise da compressão de pulsos chamou a atenção para um problema comum em radares, que é a interferência dos lóbulos laterais de um alvo que irradia maior potência com o lóbulo principal dos demais, e levou à aplicação de uma janela de Kaiser como tentativa de mitigar a discordância provocada por este efeito, o que ocorreu parcialmente, dado que houve um alargamento do lóbulo principal do alvo mais próximo do radar. Ainda assim, houve uma melhora da resolução após o uso da janela.

Também foi possível constatar a identificação correta da frequência Doppler através da caracterização do eco de uma aeronave de asas rotativas e da simulação de um cenário operacional que envolvia alvos se aproximando e afastando do radar. Outro problema comum que ficou evidente foi a influência negativa do *clutter* na identificação de alvos móveis, que foi solucionado com a supressão dos sinais de frequência Doppler nula. Nessa etapa ficou clara a capacidade de detecção de distância e velocidade do simulador.

Seguindo a linha de simulação de alvos operacionais, foi constatada a RCS reduzida de uma aeronave *stealth*, levando em conta apenas os efeitos de sua geometria. Também foram avaliadas as RCS de alvos simples, o que evidenciou o efeito da ausência da difração no simulador. Apesar disso, os gráficos obtidos no Blender se aproximaram dos teóricos e daqueles encontrados em um simulador pago, na medida em que seguiam o envelope formado pelas ondulações causadas por difração, o que indica que os efeitos de reflexão e transmissão são modelados adequadamente, ainda que limitados pela otimização do *software* para uma faixa de frequência diferente da utilizada na análise.

7.1 Trabalhos futuros

Este trabalho atingiu sua proposta inicial de desenvolver e validar um simulador em Blender, e os resultados obtidos, apesar de satisfatórios, indicam que é possível buscar melhorias para o projeto.

É interessante a inclusão de parâmetros das antenas, como ganho e área efetiva, que no *software* são representadas pela câmera e pelo emissor. Também é possível empregar outros sinais além do pulso LFM para recuperar o sinal, e outros tipos de janelas, a depender da aplicação desejada. Destaca-se também a importância de incluir efeitos de difração no simulador, o que pode ser um desafio tendo em vista que os *shaders* nativos do Blender não incluem esse fenômeno físico.

Como o Blender possui implementação otimizada para o espectro visível, inconsistências surgem em algumas análises mais exatas envolvendo as frequências de um sistema radar. Dessa forma, a continuação deste trabalho voltada para sistemas LIDAR é uma possibilidade promissora e que exploraria muito bem as funcionalidades do *software*.

Por fim, os resultados obtidos através das simulações reafirmam o potencial de aplicação do simulador para a obtenção de dados a fim de facilitar a avaliação preliminar de sistemas radar em uma plataforma *open-source*.

Referências

Airbus. **A310 - Previous Generation Aircraft**. 2020. Disponível em: <<https://www.airbus.com/aircraft/previous-generation-aircraft/a310.html>>. Acesso em: 19 nov. 2020.

Blender Manual. **Diffuse Shaders**. 2018. Disponível em: <https://docs.blender.org/manual/fr/2.79/render/blender_render/materials/properties/diffuse_shaders.html?highlight=diffuse%20shaders>. Acesso em: 7 nov. 2020.

Blender Manual. **Fresnel Node**. 2018. Disponível em: <https://docs.blender.org/manual/en/latest/render/shader_nodes/input/fresnel.html>. Acesso em: 7 nov. 2020.

Blender Manual. **Introduction to Cycles**. 2018. Disponível em: <<https://docs.blender.org/manual/en/latest/render/cycles/introduction.html>>. Acesso em: 7 nov. 2020.

Brana. **Superfície Desenvolvível, Uma Questão de Custo**. 2017. Disponível em: <<https://www.brana.com.br/serie-projeto-superficie-desenvolvivel-uma-questao-de-custo/>>. Acesso em: 15 nov. 2020.

CARVALHO, B. S. de; PRALON, M. G.; RITA, V. A. F. S. et al. Desdobramentos tecnológicos no desenvolvimento do radar SABER M60. **X Simpósio de Aplicações Operacionais em Áreas de Defesa**, Set. 2008.

CHARGIN, W. **What is a BSDF?** 2013. Blender StackExchange. Disponível em: <<https://blender.stackexchange.com/questions/785/what-is-a-bsdf>>. Acesso em: 10 Jun. 2020.

CHRISTENSEN, P.; JAROSZ, W. **The Path to Path-Traced Movies**. 2016. 103 - 175 p.

Davis III, B. J. **B-2A Spirit**. 2006. Disponível em: <<https://flic.kr/p/ciahf5>>. Acesso em: 19 nov. 2020.

Free 3D. **Companhias Aéreas Turcas Modelo 3D**. 2019. Disponível em: <<https://free3d.com/pt/3d-model/turkish-airlines-25138.html>>. Acesso em: 10 nov. 2020.

- Free 3D. **Heli Bell Modelo 3D**. 2019. Disponível em: <<https://free3d.com/3d-model/heli-bell-206-359500.html>>. Acesso em: 10 nov. 2020.
- Free 3D. **Northrop Grumman B-2 Spirit Modelo 3D**. 2020. Disponível em: <<https://free3d.com/3d-model/northrop-grumman-b-2-spirit-656073.html>>. Acesso em: 19 nov. 2020.
- JAYASRI, P.; NIHARIKA, K.; YEDUKONDALU, K.; KUMARI, E.; PRASAD, A. Radar cross section characterization of corner reflectors in different frequency bands and polarizations. **The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, v. 42, n. 5, p. 637–642, Nov. 2018.
- KNOTT, E.; SCHAEFFER, J.; TULLEY, M. **Radar Cross Section**. 2. ed. Raleigh: Institution of Engineering and Technology, 2004.
- LEVANON, N.; MOZESON, E. **Radar Signals**. New Jersey: John Wiley & Sons, 2004.
- Lockheed Martin. **F-22 Specifications**. 2020. Disponível em: <<https://www.lockheedmartin.com/en-us/products/f-22/f-22-specifications.html>>. Acesso em: 19 nov. 2020.
- MAHAFZA, B. R. **Radar systems analysis and design using MATLAB**. Huntsville: Chapman & Hall/CRC, 2000.
- Ministério da Defesa. **Estratégia Nacional de Defesa**. 2008. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/decreto/d6703.htm>. Acesso em: 12 nov. 2020.
- OUZA, M.; ULRICH, M.; YANG, B. A simple radar simulation tool for 3D objects based on blender. **2017 18th International Radar Symposium (IRS)**, p. 1–10, Jun. 2017.
- ROSSO, D. **Raptor in flight**. 2013. Disponível em: <<https://flic.kr/p/dKE4ZV>>. Acesso em: 19 nov. 2020.
- SEDMAN, R. **How to calculate for every ray the total distance it has traveled from camera to emitter**. 2017. Blender StackExchange. Disponível em: <<https://blender.stackexchange.com/questions/81485/how-to-calculate-for-every-ray-the-total-distance-it-has-traveled-from-camera-to>>. Acesso em: 10 jun. 2020.
- SILVA, J. A. N.; POMPEO, B. S.; RITA, V. A. F. S.; CARVALHO, B. S. Uma visão geral sobre os radares desenvolvidos pelo exército brasileiro. In: **Cadernos Cadernos CPqD Tecnologia**. Campinas: Fundação CPqD, 2014. v. 10, p. 27–40.
- SKOLNIK, M. I. **Introduction to Radar Systems**. 3. ed. New York: McGraw-Hill, 2001.
- SKOLNIK, M. I. **Radar Handbook**. 3. ed. Baltimore: McGraw-Hill, 2008.
- TurboSquid. **F-22 Raptor Modelo 3D**. 2015. Disponível em: <https://www.turbosquid.com/pt_br/3d-models/free-blend-model-f-22-raptor/986738>. Acesso em: 10 nov. 2020.

TurboSquid. **US Humvee (Lowpoly) 3D model**. 2017. Disponível em: <<https://www.turbosquid.com/3d-models/humvee-3d-model-1200461>>. Acesso em: 10 nov. 2020.

US Air Force. **B-2 Spirit**. 2015. Disponível em: <<https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104482/b-2-spirit/>>. Acesso em: 19 nov. 2020.

WEDELSTAEDT, K. von. **Airbus A310-304 - Air Transat**. 2009. Disponível em: <<https://www.airliners.net/photo/Air-Transat/Airbus-A310-304/1571009/L>>. Acesso em: 19 nov. 2020.

YANG, R.; LI, H.; LI, S.; ZHANG, P.; TAN, L.; GAO, X.; KANG, X. **High-Resolution Microwave Imaging**. Pequim: Springer, 2018.

FOLHA DE REGISTRO DO DOCUMENTO

¹ CLASSIFICAÇÃO/TIPO <p style="text-align: center;">TC</p>	² DATA <p style="text-align: center;">23 de novembro de 2020</p>	³ REGISTRO Nº <p style="text-align: center;">DCTA/ITA/TC-050/2020</p>	⁴ Nº DE PÁGINAS <p style="text-align: center;">75</p>
⁵ TÍTULO E SUBTÍTULO: Desenvolvimento de um simulador radar utilizando software aberto de modelagem e renderização 3D.			
⁶ AUTOR(ES): Raíssa Brasil Andrade			
⁷ INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES): Instituto Tecnológico de Aeronáutica – ITA			
⁸ PALAVRAS-CHAVE SUGERIDAS PELO AUTOR: Radar; Path-tracing; Simulador, Modelagem 3D.			
⁹ PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Radar; Simuladores; Rastreamento (posição); Receptores; Computação.			
¹⁰ APRESENTAÇÃO: <input checked="" type="checkbox"/> Nacional () Internacional ITA, São José dos Campos. Curso de Graduação em Engenharia Eletrônica. Orientador: Prof. Renato Machado. Publicado em 2020.			
¹¹ RESUMO: Radares são fundamentais para atividades de defesa e segurança que envolvem reconhecimento e monitoramento de alvos. Por envolver a detecção de sinais refletidos pelo ambiente, é necessário estabelecer métodos capazes de diferenciar os alvos de interesse daqueles que não trazem informação relevante para o sistema, e para isso o conhecimento prévio do comportamento desses alvos é de grande importância. Nesse contexto, simulações computacionais são uma forma eficiente e segura de coletar dados úteis para este fim. O presente trabalho propõe o desenvolvimento e a validação de um simulador radar através da construção de cenários e obtenção de dados no Blender, um <i>software open-source</i> de modelagem 3D, e posterior tratamento desses dados em MATLAB. O ambiente em Blender conta com um emissor de luz e uma câmera, de forma que a renderização carrega informações sobre a interação da luz com o cenário através do Cycles, que é o <i>path-tracer</i> interno do <i>software</i> . No simulador apresentado, os elementos da cena têm seu comportamento manipulado para que as informações relevantes para aplicação radar sejam incluídas nas componentes RGB dos pixels da imagem renderizada pela câmera, que atua como um receptor. O emissor de luz, por sua vez, é tratado como um transmissor. Finalmente, o tratamento adequado dos <i>frames</i> resultantes da renderização através da aplicação de conceitos de processamento de sinais e teoria de radar permite analisar o sinal reconstruído para diferentes cenários 3D e confrontá-lo com o que é previsto pela teoria e literatura da área.			
¹² GRAU DE SIGILO: <p style="text-align: center;"> <input checked="" type="checkbox"/> OSTENSIVO () RESERVADO () SECRETO </p>			