

Texturas BSDF a Partir de Fotografias com Flash

Tiago Petená Veraldi *Hélio Pedrini*

Relatório Técnico - IC-PFG-22-34
Projeto Final de Graduação
2022 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

The contents of this report are the sole responsibility of the authors.
O conteúdo deste relatório é de única responsabilidade dos autores.

Texturas BSDF a Partir de Fotografias com Flash

Tiago Petená Veraldi* Hélio Pedrini†

Resumo

Este é um estudo sobre a criação de materiais realistas, fisicamente baseados, para aplicações gráficas 3D, tais como jogos e filmes animados. Os métodos tradicionais para recriar texturas requerem ferramentas caras, perícia e consomem tempo. Um método recente, desenvolvido por Henzler et al. [5], propõe criar parâmetros da função de distribuição da reflectância bidirecional (do inglês, *Bidirectional Reflectance Distribution Function* - BRDF) realistas - difusos, normais, rugosos e mapas especulares - a partir de uma única fotografia obtida por telefone celular com a lanterna do dispositivo ligada. Ele aprende a codificar a imagem em um espaço latente e depois como decodificá-la nos mapas de textura com uma abordagem sem supervisão. Os resultados são realistas, sem costura e podem ser interpolados com outros materiais, resultando em um gerador de textura virtualmente infinito. Revisamos este método e descrevemos possíveis modificações que poderiam ser aplicadas à abordagem. O uso de imagens cruas, em vez de fotografias finais processadas para se beneficiar de sua característica linear, pode melhorar o realismo dos materiais. Além disso, incluímos o *Blender* como o motor de renderização para que materiais mais complexos possam ser criados.

*Instituto de Computação, Universidade Estadual de Campinas, 13083-852, Campinas, SP.
†Instituto de Computação, Universidade Estadual de Campinas, 13083-852, Campinas, SP.

1 Introdução

O uso de gráficos tridimensionais (3D) tem crescido cada vez mais em diferentes aplicações como em jogos, filmes, visualizações arquiteturais e em realidades aumentada e virtual. Para a criação destes ambientes, além da reconstrução de malha e geometria — que define a sua forma no espaço 3D — também é necessário definir como estes objetos irão reagir à luz (Figura 1).

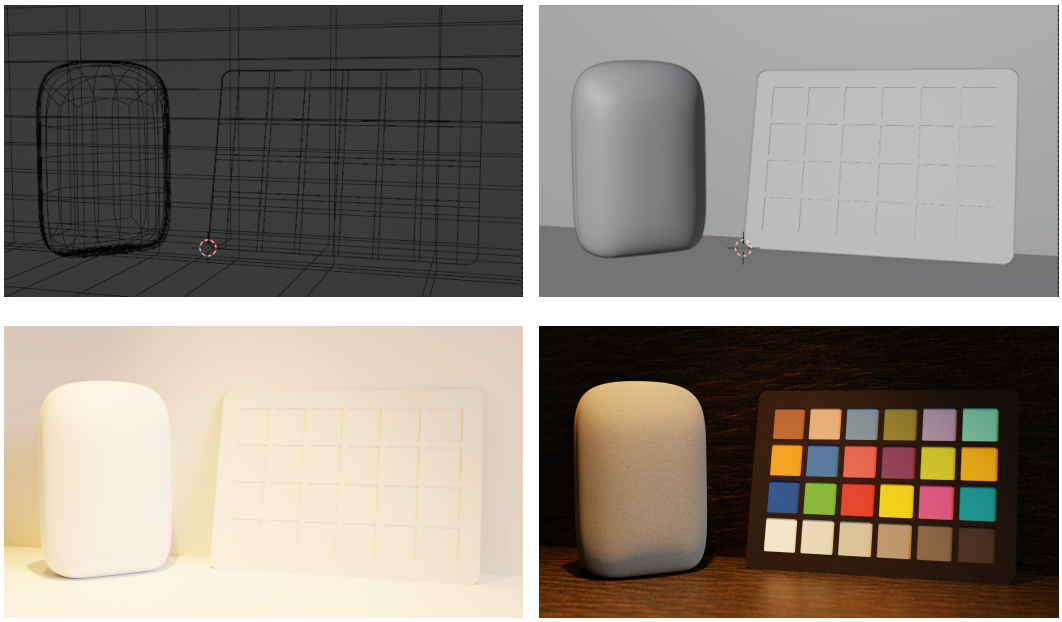


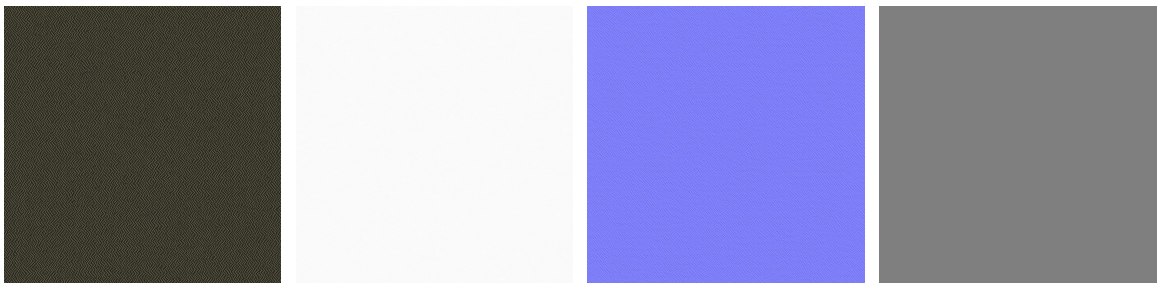
Figura 1: As figuras mostram as etapas necessárias para a criação de uma imagem renderizada 3D, desde a definição da estrutura dos objetos à imagem final com textura e iluminação.

A definição do material influencia drasticamente no resultado final, representando uma parcela significativa do tempo gasto ao se replicar determinado objeto em 3D. Obter estes parâmetros de forma que o resultado seja uma representação realística é um processo custoso e demorado. Os métodos tradicionais requerem artistas experientes, equipamentos caros, processos técnicos, além de envolver uma série de ferramentas específicas para pós-processamento de dados.

Neste contexto e aliados à expansão do uso de técnicas de aprendizado de máquina, vários estudos vêm tentando encontrar novas formas de abordar o problema da criação de objetos virtuais. Alguns trabalhos [2, 8] encontram formas de reconstruir a geometria, além do material, com apenas algumas imagens. Aittala et al. [1] apresentaram um método que, a partir de uma fotografia obtida com a iluminação do flash de um celular, possibilita a sintetização da textura da fotografia e inspirou diversos outros trabalhos.

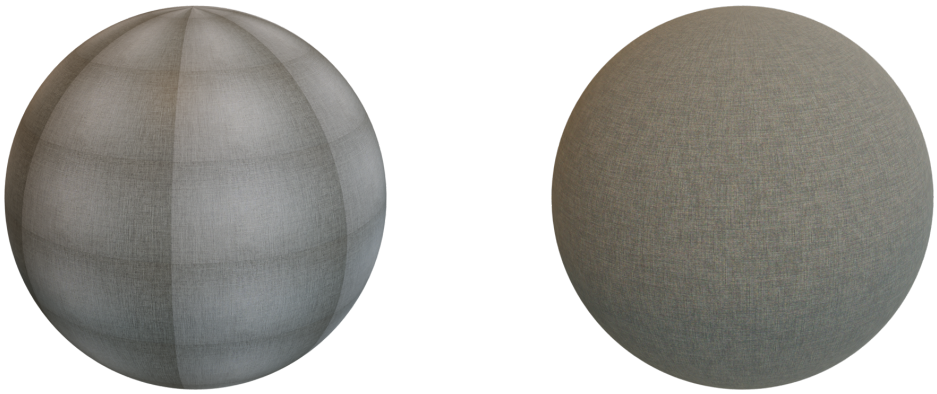
Este estudo baseou-se no trabalho de Henzler et al. [5], os quais disponibilizaram seu método e código, para estudar o funcionamento e realizar modificações de modo a aprimorar seu funcionamento. Nele, a partir de uma fotografia capturada por um celular, em que a iluminação predominante é o flash, é gerado um conjunto de texturas BRDF (Figura 2) que

possibilita a renderização do material. O resultado é livre de efeitos de *tiling* (Figura 3), possibilita a interpolação de dois materiais e pode gerar um número virtualmente infinito de resultados.



(a) *Diffuse* (b) *Roughness* (c) *Normal* (d) *Specular*

Figura 2: Mapas BRDF.



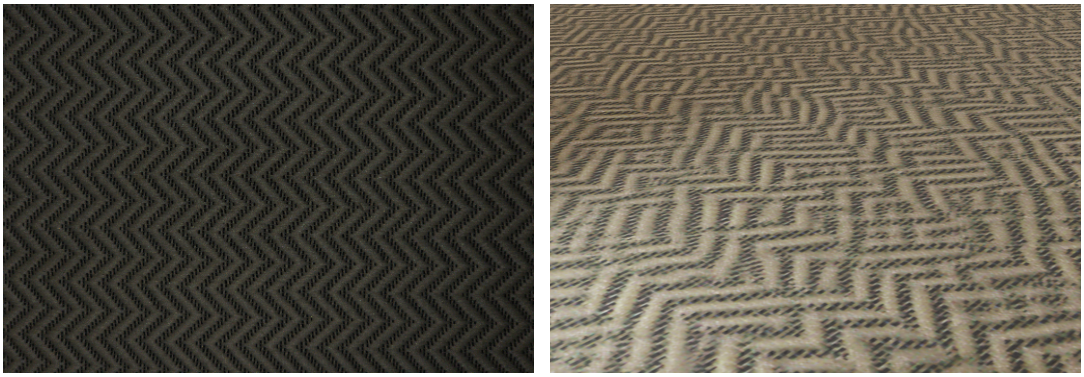
(a) Imagem simples aplicada como mapa *diffuse*. (b) Material gerado a partir da imagem com flash.

Figura 3: Quando uma textura é replicada diversas vezes em uma mesma superfície e sua borda fica nítida, quebrando a ilusão de realismo.

Entretanto, o método possui algumas restrições quanto ao seu funcionamento. O material escolhido precisa ser *estacionário*, isto é, possuir as mesmas propriedades estatísticas seja qual for a região da imagem. Também é necessário o posicionamento da câmera fronto-paralelamente à superfície plana de interesse, o que limita bastante os possíveis materiais que podem ser obtidos com o método.

O software *Open-Source Blender* tem como renderizador *path-tracer* padrão o "Cycles". Nele, os materiais são principalmente definidos por uma função *Principled Bidirectional Scattering Distribution Function* (BSDF), baseada em física ([*Physically-Based Rendering* - PBR). A variedade de materiais que é possível de representar é muito maior devido à

possibilidade de definição de materiais isotrópicos e anisotrópicos, metálicos e não-metálicos, níveis de transmissão entre outros parâmetros. O objetivo foi gerar texturas diretamente para este, levando em conta suas possíveis aplicações (Figura 4).



(a) Fotografia com flash (b) Textura gerada

Figura 4: A partir da fotografia com flash, pode-se gerar um objeto 3D com um material com propriedades parecidas com as do objeto original, até mesmo sob outros ângulos e iluminações.

1.1 Modelos de Reflexão

A função de distribuição da reflectância bidirecional (BRDF) [7] relaciona como um raio de uma fonte de luz, na direção w_i e com radiância L, reage quando refletido por um ponto p de um determinado material refletindo com irradiância E na direção w_o (Equação 1).

$$f_r(p, \omega_0, \omega_i) = \frac{dL_0(p, \omega_0)}{dE_0(p, \omega_i)} \tag{1}$$

Os mapas BRDF *Diffuse* são responsáveis pela cor do raio refletido quando iluminado por uma fonte de luz difusa, os mapas *Normal* descrevem a direção do vetor normal à superfície, os mapas *Specular* definem a intensidade, enquanto os mapas *Roughness* indicam o quão lisa ou irregular a superfície é.

1.2 Estacionaridade

Um material estacionário (Figura 5) possui as mesmas propriedades estatísticas em todas as texturas, independentemente da região ou orientação da imagem. A restrição a materiais estacionários possibilita o mapeamento da imagem com flash às texturas a um espaço menor de possibilidades, já que, sem ela, poderiam existir infinitos mapeamentos que produziriam a mesma saída.

Materiais não estacionários são aqueles que aparecem com mais frequência, ou seja, que não possuem as mesmas propriedades na imagem como um todo, tais como os exemplos na Figura 6.

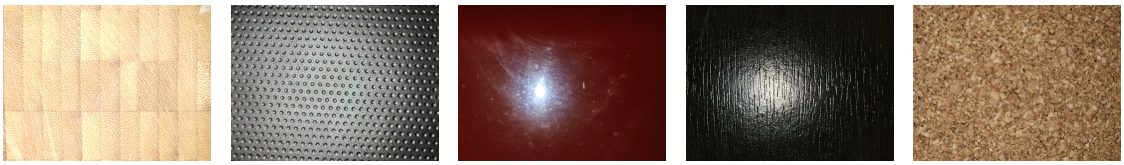


Figura 5: Exemplos de materiais estacionários.



Figura 6: Exemplos de materiais não estacionários.

1.3 Imagem com Flash

A imagem com flash de um material estacionário contém, em uma mesma imagem, o equivalente a um material sob diversas iluminações diferentes.

1.4 Formato RAW

Fotografias capturadas com telefones celulares (Figura 7) atuais passam por diversos processamentos antes de chegar à imagem final que é apresentada ao usuário. Devido às propriedades não lineares dessas funções, a linearidade da imagem em relação à quantidade de luz recebida pelo sensor acaba sendo perdida.



(a) Foto *raw* RGB (b) Foto linear sRGB (c) Foto sRGB e *tonemapping*

Figura 7: Fotografias reais capturadas com um telefone celular.

Diversos estudos vêm mostrando que o uso de imagens sem processamento pode trazer benefícios a tarefas de visão computacional e processamento de imagens [4,9,11]. Estudamos se, ao fazer modificações na rede proposta por Henzler et al. [5], ela traria algum benefício se fosse treinada com imagens RAW.

Todos os mapas da textura devem ser lineares em relação à luz, ao fazer o *encoding* das imagens muito processadas a rede deve aprender estas funções e, após a aplicação de *tonemappings* não se consegue voltar à imagem original, existindo trabalhos dedicados

apenas a essa tarefa. O fator mais importante é a linearidade do espaço de cor que pode ajudar no *encoding-decoding* das imagens.



Figura 8: Pré-processamento feito a partir das imagens '.dng'.

1.5 Mapa Especular

Para representar os materiais corretamente, geralmente é recomendado que se mantenha no valor padrão de 0.5, já que o valor está, na maior parte, entre 0.4 e 0.6 para materiais dielétricos [3, 10]. Com o intuito de chegar em mapas fisicamente mais realistas para uma maior variedade de materiais e diminuir o número de parâmetros da rede, fixou-se o mapa especular no valor de 0.5 já que este mapa, com os pesos pré-treinados fornecidos, tende a ser muito volátil e em um mesmo material varia de 0 a 1 muito rapidamente.

1.6 Arquitetura

O método consiste em aprender uma transformação que codifica determinada imagem de entrada em um espaço latente. Dados a imagem codificada e um mapa de ruído aleatório, decodifica-se nas texturas BRDF.

A arquitetura das redes foram mantidas em suas configurações originais. Retirou-se uma camada do *decoder* que representava o mapa especular.

1.6.1 Encoder

A imagem de entrada com 512×384 pixels, para manter o aspecto original de entrada, é codificada a um espaço latente aprendido de 64 dimensões por uma arquitetura ResNet.

1.6.2 Decoder

O *decoder*, a partir da imagem com flash codificada e de uma matriz de ruído, passa por uma rede com arquitetura da U-Net de modo a gerar os mapas *normal*, *diffuse* e *roughness*.

1.6.3 Loss

A rede é não-supervisionada e, portanto, não precisa de um *ground truth* para realizar o treinamento. As comparações entre as imagens de entrada e saída são feitas verificando a similaridade visual de *patches* de localizações aleatórias com as ativações em uma rede *VGG*, comparando as diferenças L_1 e L_2 das *Gram matrix* e seu espectro de frequência em diversas escalas. Assim, a *loss* não possui um termo explícito sobre a estacionaridade do material, porém, força essa condição ao condicionar o trainamento da rede à comparação das estatísticas de ativação na VGG.

Apesar do *encoding* ser feito na imagem linear e a imagem gerada também, estas são melhores visualizadas em sRGB, então, uma função *gamma* é aplicada à entrada e saída da rede para que a *loss* seja calculada.

2 Treinamento

Esta seção descreve brevemente os principais aspectos no processo de treinamento da rede.

2.1 Conjunto de Dados

Por não existirem conjuntos de dados no formato estudado, fez-se necessário a obtenção de uma nova base de dados. Para isso, foram obtidas 1295 fotografias no formato '.dng' a partir de uma câmera *Telephoto* de um celular Samsung Galaxy S22. Também foram salvas as mesmas imagens processadas pelo celular em formato '.jpg'.

Foram contemplados um total de 258 materiais, sobre os quais, de uma a quatro fotografias foram tiradas com pequenas movimentações do aparelho. O ISO da câmera foi mantido em 100 para evitar maiores níveis de ruído, com o tempo de exposição automático. Devido ao flash, a iluminação é suficiente para que não haja *motion blur* devido a um tempo de exposição maior.

Para equilibrar o conjunto de dados, variou-se a quantidade de cores e as texturas dos objetos. Também equilibrou-se a quantidade de objetos lisos para ásperos. Dentre estes, 15% foram dedicados à validação e 85% ao treinamento. Uma quantidade pequena de materiais visivelmente não estacionários também foi incluída para adicionar entropia ao conjunto de dados.

2.1.1 Pré-Processamento

Os arquivos '.dng' possuem as imagens com um processamento mínimo ainda no padrão Bayer GBRG e no espaço de cor da câmera. O mapa *diffuse*, principal responsável pela cor base da textura, precisa ser sRGB para que funcione nos renderizadores mais conhecidos sem modificações posteriores. Para que as redes não sejam responsáveis por aprender essa configurações, são aplicados alguns processamentos às imagens para que a entrada já esteja no padrão sRGB, entretanto, linear.

Para isso, foi considerado que a correção de cor consegue perfeitamente traduzir o espaço de cor da câmera ao espaço sRGB, o que não é garantido. Porém, foi usada uma combinação dos metadados dados pelo ISP da câmera e usado um *color checker* para minimizar os possíveis erros deste processo.

2.1.2 White Balance

Considerando que todas as imagens são iluminadas pelo mesmo iluminante – o flash do celular – podemos considerar o ganho ao ser aplicado em cada canal como constante a

todas as imagens de uma mesma câmera.

$$\begin{bmatrix} R_{Wb} \\ G_{Wb} \\ B_{Wb} \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \cdot \begin{bmatrix} R_{gain} \\ G_{gain} \\ B_{gain} \end{bmatrix} \tag{2}$$

2.1.3 Demosaicing

As imagens RAW são formadas no padrão Bayer (Figura 9) nos sensores da maioria das câmeras digitais. Para transferi-lo ao padrão RGB, o algoritmo AHD [6] foi usado.

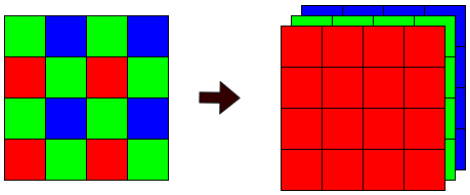


Figura 9: Padrão CFA Bayer (esquerda) encontrado nas imagens *raw* e o padrão RGB (direita) mais usado com tamanho 4×4 pixels.

2.1.4 Color Correction

Por último, a cor da imagem foi transferida para sRGB calculando a matriz de correção de cor CCM resolvendo a Equação 3 utilizando o método dos mínimos quadrados com os valores dos *patches* P na fotografia e seus valores conhecidos C pelo *color checker*.

$$\text{CCM} = \begin{bmatrix} P_{R0} & P_{R1} & ... & P_{R23} \\ P_{G0} & P_{G1} & ... & P_{G23} \\ P_{B0} & P_{B1} & ... & P_{B23} \end{bmatrix} * \begin{bmatrix} C_{R0} & C_{R1} & ... & C_{R23} \\ C_{G0} & C_{G1} & ... & C_{G23} \\ C_{B0} & C_{B1} & ... & C_{B23} \end{bmatrix} \tag{3}$$

$$Imagem_{sRGB} = CCM * Imagem_{RGB} \tag{4}$$

2.2 Blender

Blender é uma ferramenta de software 3D gratuita e de código aberto utilizada para a criação de filmes, efeitos visuais, modelagem, animação e diversas outras funções. *Cycles* é seu motor gráfico baseado em *path-tracing*, utilizando a equação de Monte Carlo, sendo capaz de simular cenas com alto grau de realismo.

O *shader* "Principled BSDF" é responsável por possibilitar um fluxo de criação de materiais baseados em física (PBR). Dentre suas entradas, estão os mesmos mapas *normal*, *diffuse*, *roughness* e *specular*. Porém, este módulo é muito mais complexo. Além da BRDF, as funções BTDF e BSSRDF podem simular transmissão de luz, dispersão subsuperficial e objetos metálicos.

Também é possível fazer o deslocamento da superfície com base em um mapa de altura que, diferentemente do mapa normal, desloca os vértices da geometria. Atrelado à função

de subdivisão adaptativa da geometria, que divide a geometria com base na necessidade para representação do mapa de deslocamento, ela possibilita uma simulação mais realista da superfície.

O programa inclui um protocolo de comunicação com a linguagem *Python* com o qual, virtualmente, tudo o que se pode fazer na interface gráfica também é possível fazer com código. Exploramos isso para gerar cenas automaticamente, configurando os materiais automaticamente após a inferência da rede e geração das texturas.

Também foi realizada uma configuração de um segundo treinamento para usar o *Blender* e *Cycles* no treinamento da rede, gerando a cena com a câmera, ponto de luz e superfície para o treinamento da rede. O *decoder* já entrega como saída o mapa de altura sobre o qual é calculado o mapa normal. O *Blender* também foi usado em um segundo treinamento, onde o renderizador final desloca a geometria ao invés de usar o mapa normal.

O *encoder* de rotação foi alterado para aprender a rotação do plano, aplicando uma rotação em até 10 graus nos eixos x e y. A iluminação usada foi apenas uma luz pontual de 5mm com o valor aproximado de um flash do celular (45 lumens) atrelada à câmera que são posicionados a 20cm da superfície. Os parâmetros de câmera foram gerados e parametrizados com base nas informações do fabricante do aparelho (largura do sensor: 6.45mm; distância focal: 5.4mm). Os renderizadores são salvos como *OpenEXR* 32 bits, preservando a *range* dinâmica e linearidade das cores, sem nenhuma transformação. São coletadas 512 amostras por pixels pelo *path tracer*.

O mapeamento UV é ajustado para que, seja qual for a resolução dos mapas gerados, quando sem rotação, o plano preencha toda a visão da câmera. Um escalonamento aleatório do plano durante o treinamento foi implementado mas não usado durante o novo treinamento. O plano inicialmente é subdividido 4 vezes, resultando em um plano de inicialmente 256 faces que é então subdividido dinamicamente segundo necessidade. O mapa de altura é configurado para representar uma distância de 0 a 2cm, tornando a auto-oclusão (*self occlusion*) um fenômeno não mais desprezível.

A exposição não é ajustada e poderia se basear nos metadados do arquivo de entrada ou adicionar ao *encoder* este parâmetro. Assim, o renderizador sempre corresponderá a uma fotografia tirada na exposição de 100 de ISO, 1 segundo de tempo de exposição e 1 de abertura - o que não é válido como no renderizador implementado por Henzler et al. [5]. Durante o treinamento, uma compensação de 0 a +2 *stops* de luz são aplicados ao renderizador. O treinamento foi feito a partir dos pesos pré-treinados sem o *Blender* durante 5 épocas com *batch* de 1 imagem.

3 Resultados

Esta seção apresenta os principais resultados obtidos neste projeto.

3.1 Parâmetros

A rede foi treinada com parâmetros em uma NVIDIA RTX3080 12GB durante aproximadamente 6 dias. Para uma análise fiel dos resultados, a rede deveria ser treinada nas configurações originais, entretanto, com o mesmo conjunto de dados, junto às imagens '.jpg'

processadas totalmente pelo celular, e comparadas com a rede treinada com as mesmas imagens, porém, salvas como 'raw' e processadas para sRGB linear. Entretanto, devido ao tempo necessário para o treinamento da rede, não foi possível fazer esta devida comparação.

Além disso, não há uma métrica valida para esta comparação já que o resultado não é uma imagem inteiramente alinhada com a fotografia de entrada. Assim, somente uma análise qualitativa pode ser feita. Porém, existem limitações claras como a reprodução de padrões geométricos (Figura 16) e a dificuldade de representação de materiais lisos com cores sólidas (Figura 16).

3.2 Resultados com Blender

Como esperado, o treinamento com a ferramenta adiciona um alto tempo de processamento, praticamente triplicando o tempo de processamento de uma época. Foi usado o acelerador *NVIDIA Optix* e uma cena simples. Apesar das simplificações, o processo de calcular a geometria e renderizar usando *path tracer* ainda adiciona um tempo muito grande de processamento. Isto é somado à limitação de obter o renderizador diretamente da memória, fazendo-se necessário salvar o arquivo e carregá-lo do disco a cada imagem, processo que não deveria ser necessário, e que já está em processo de implementação no *Blender*.

Sendo assim, não foi possível obter um tempo de treinamento significativo para avaliar os resultados de treinamento na rede.

4 Conclusões

A necessidade de estacionaridade dos materiais e fronto-paralelidade da câmera diminui a variedade de materiais que podem ser sintetizados. Apesar disso, o método prova-se como uma alternativa válida na criação de materiais por um fluxo generativo. Em minutos, pode-se criar texturas realistas que levariam horas em um processo tradicional.

O uso de imagens lineares e fixação do mapa especular mostraram-se beneficiais à reprodução dos materiais sob diferentes iluminações e um teste mais profundo deveria ser realizado para confirmar este ganho.

No estado atual, a utilização do *Blender* para a aplicação mostra-se pouco viável. Contudo, mostra o potencial da plataforma em criar mais facilmente cenários mais complexos. Podendo tirar a necessidade de restrição do material ter a superfície plana (Figura14). Com a ampliação no número de parâmetros da rede, poderia se adicionar outros mapas a serem aprendidos, como o de materiais metálicos e de dispersão subsuperficial. Também poderiam ser feitas algumas otimizações como a renderização de *batches* como uma animação para diminuir o tempo de processamento.

O uso de redes neurais para a geração de texturas mostra-se viável para uma grande quantidade de materiais com uma alta qualidade que deve ter seu uso adotado em breve por um número maior de artistas.

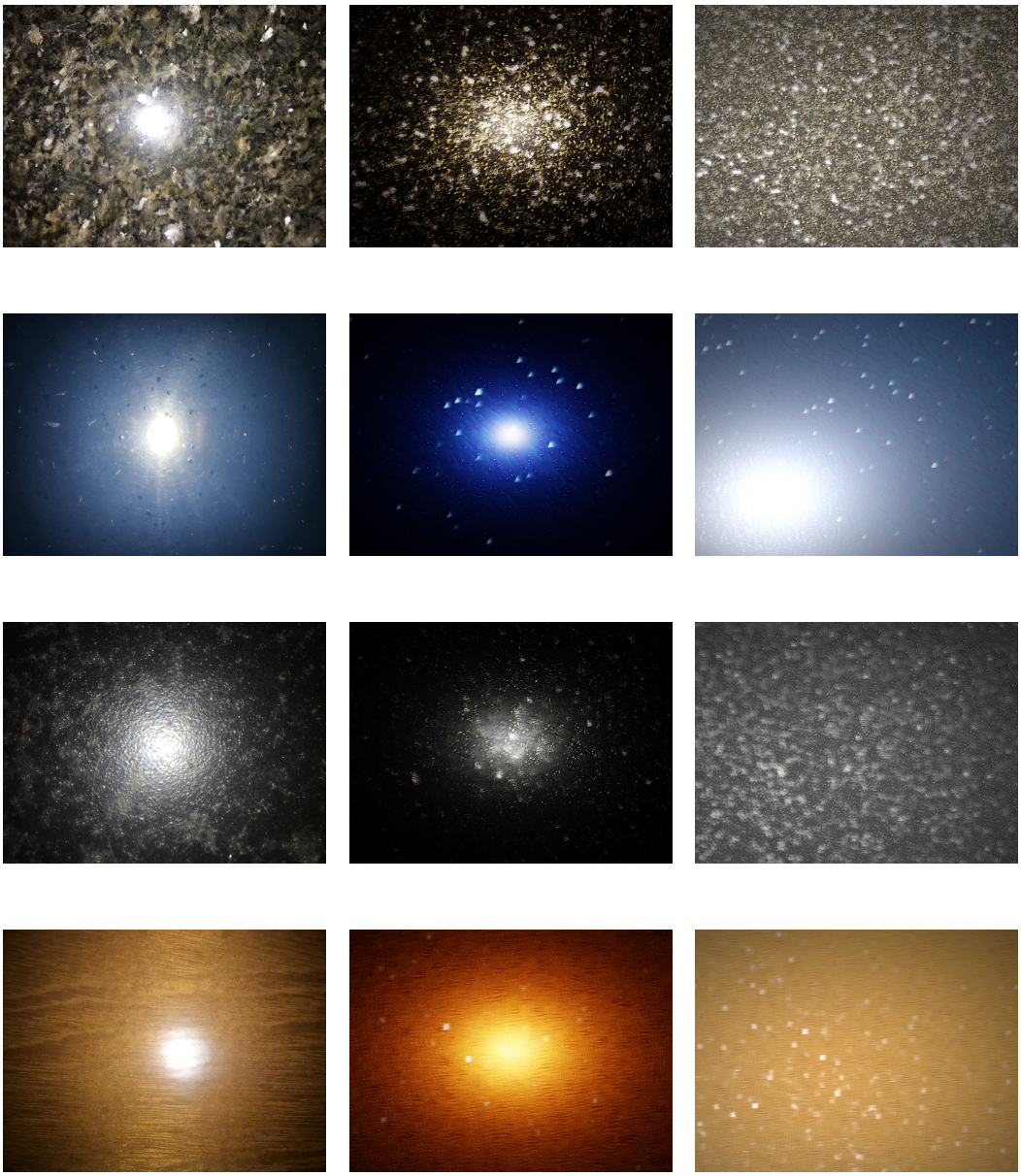
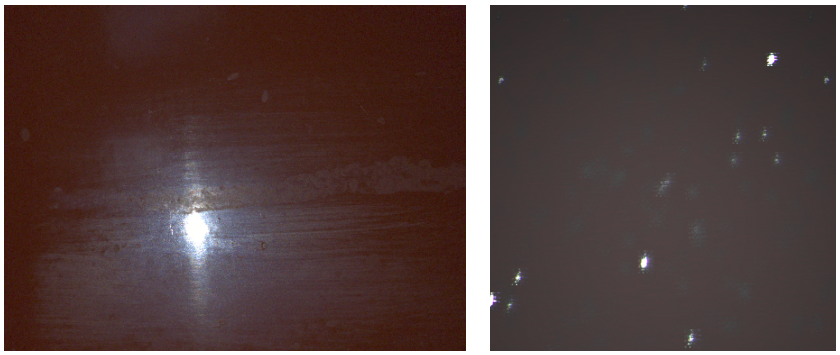


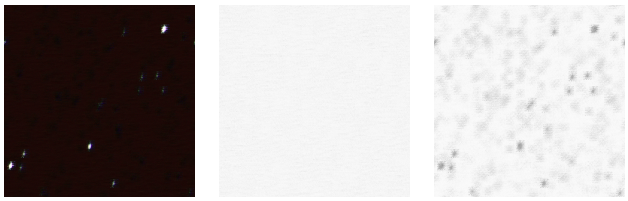
Figura 10: Comparação dos resultados entre os renderizadores do *Blender* (segunda coluna) e o renderizador usado no treinamento (terceira coluna) nas imagens do conjunto de dados disponibilizado por Henzler et al. [5] (primeira coluna).

Referências

[1] M. Aittala, T. Aila, and J. Lehtinen. Reflectance Modeling by Neural Texture Synthesis. *ACM Transactions on Graphics*, 35(4), July 2016.

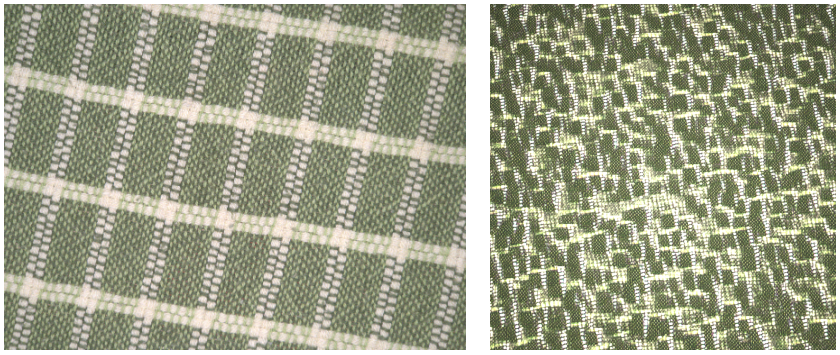


(a) Entrada (b) *Renderizada (Blender)*

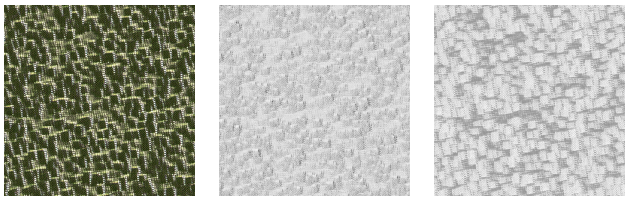


(c) *Diffuse* (d) *Height* (e) *Roughness*

Figura 11: Material liso com cor sólida confunde a rede gerando artefatos devido ao reflexo especular.

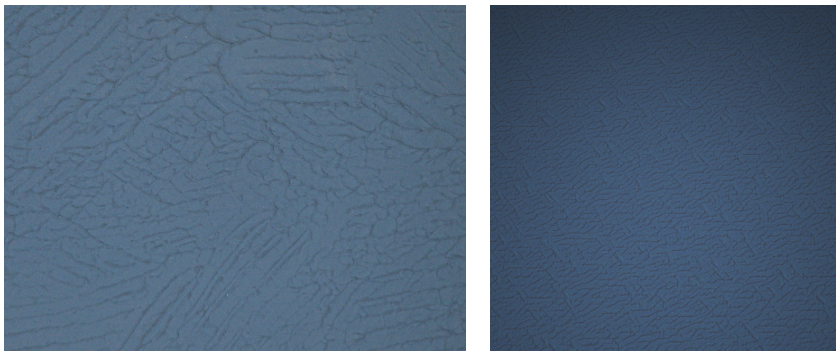


(a) Entrada (b) *Renderizada (Blender)*

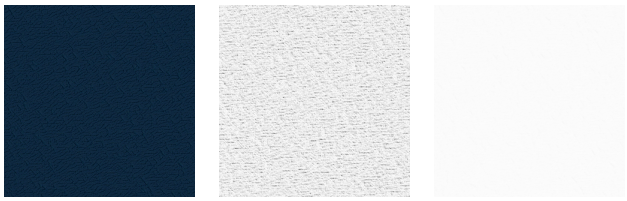


(c) *Diffuse* (d) *Height* (e) *Roughness*

Figura 12: Material com padrões geométricos. Apesar de representar bem o material, as estruturas não são replicadas.



(a) Entrada (b) *Renderizada (Blender)*



(c) *Diffuse* (d) *Height* (e) *Roughness*

Figura 13: Material difuso é bem replicado, adicionando as texturas onduladas ao mapa *Normal* sem adicionar as sombras ao mapa *Diffuse*, já que estas são suficientemente pequenas.

[2] S. Bi, Z. Xu, K. Sunkavalli, D. Kriegman, and R. Ramamoorthi. Deep 3D Capture: Geometry and Reflectance from Sparse Multi-View Images. *10.48550/arxiv.2003.12642*, 2020.

[3] Blender. Specular BSDF, 2022. `https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/specular_bsdf.html`.

[4] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron. Unprocessing Images for Learned Raw Denoising. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

[5] P. Henzler, V. Deschaintre, N. J. Mitra, and T. Ritschel. Generative Modelling of BRDF Textures from Flash Images. *ACM Transactions of Graphics (Proc. SIGGRAPH Asia)*, 40(6), 2021.

[6] K. Hirakawa and T. Parks. Adaptive Homogeneity-Directed Demosaicing Algorithm. *IEEE Transactions on Image Processing*, 14(3):360–369, 2005.

[7] W. J. Matt Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 3 edition, 2016.

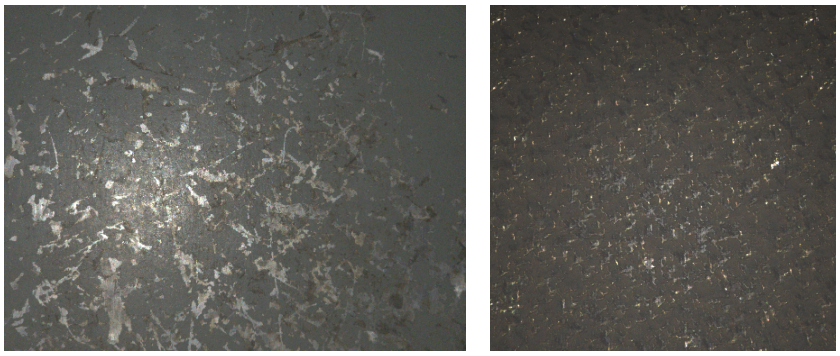


(a) Entrada (b) *Renderizada (Blender)*

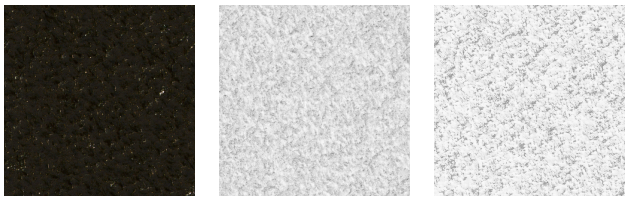


(c) *Diffuse* (d) *Height* (e) *Roughness*

Figura 14: Material com oclusão própria tem a sombra adicionada ao mapa *Diffuse*, além do mapa *Normal* não sendo bem representado.



(a) Entrada (b) *Renderizada (Blender)*



(c) *Diffuse* (d) *Height* (e) *Roughness*

Figura 15: Exemplo de geração de texturas de um material heterogêneo.



(a) Entrada (b) *Renderizada (Blender)*



(c) *Diffuse* (d) *Height* (e) *Roughness*

Figura 16: Exemplo de geração de texturas de uma tábua de madeira. Apesar de variação de altura exagerada, o material é bem representado.

[8] G. Nam, J. H. Lee, D. Gutierrez, and M. H. Kim. Practical SVBRDF Acquisition of 3D Objects with Unstructured Flash Photography. *ACM Transactions of Graphics*, 37(6), Dec. 2018.

[9] C. Stamatopoulos, C. Fraser, and S. Cronk. Accuracy Aspects of Utilizing Raw Imagery in Photogrammetric Measurement. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39:387–392, July 2012.

[10] Unreal Engine. Physically Based Materials, 2022. `https://docs.unrealengine.com/5.0/en-US/physically-based-materials-in-unreal-engine/`.

[11] W. Zhou, L. Zhang, S. Gao, and X. Lou. Gradient-Based Feature Extraction From Raw Bayer Pattern Images. *IEEE Transactions on Image Processing*, 30:5122–5137, 2021.