

# Visualização em Tempo Real de um Modelo Esparsos de Mistura Paramétrica para síntese da BTF

Nuno Silva \*

Centro de Computação Gráfica  
Campus Azurém, Guimarães  
nuno.silva@ccg.pt

Luís Paulo Santos

Dep. Informática, Universidade do Minho  
Campus Gualtar, Braga  
psantos@di.uminho.pt

## Resumo

As funções de textura bidirecionais (*Bidirectional Texture Functions* - BTF) permitem visualizações de alta qualidade de materiais reais, que exibem detalhes complexos na sua aparência, e que não podem ser fielmente representados por funções paramétricas mais simples. Representações fíeis deste tipo de materiais requerem grandes quantidades de dados, dificultando a sua visualização em tempo real. A compressão de BTFs constitui um compromisso entre qualidade visual e tempo de síntese. Este artigo apresenta um visualizador a correr integralmente no GPU, usando um motor de *ray tracing*, de uma representação recente para BTFs, o Modelo Esparsos de Mistura Paramétrica (*Sparse Parametric Mixture Model* - SPMM). A escalabilidade com o número de BTFs e o número de luzes é também estudado.

## Palavras-Chave

Materiais complexos, BTF, Visualização em tempo real, GPU, Ray tracing

## 1 INTRODUÇÃO

Representações digitais de materiais complexos têm um grande impacto na indústria têxtil e do calçado, auxiliando *designers* e artistas na prototipagem de novos produtos, e fornecendo visualizações realistas destes produtos ao consumidor final. Para o *designer*, a utilidade destas ferramentas depende tanto da qualidade da representação do material, como da sua visualização interativa.

A aparência do material depende da forma como o fluxo radiante interage com a superfície, e varia de acordo com as direções de incidência e de reflexão, entre outros fatores [Weyrich08].

É comum usar a *Bidirectional Reflectance Distribution Function* (BRDF), para modelar a aparência de um material, mas estas funções não permitem simular vários fenómenos complexos, tais como auto-oclusão, transluscência ou inter-reflexões. Para esse efeito, a *Bidirectional Texture Function* (BTF) [Dana99] está a tornar-se popular devido ao elevado realismo permitido, melhorias nos sistemas de captura e aumento do poder computacional dos GPUs [Weyrich08, Müller05, Filip09].

A BTF é uma função 6D que modela a aparência de um material capturando várias imagens deste para várias direções de iluminação e de observação. As imagens são armazenadas em grandes tabelas, e o processo de síntese

envolve simples acessos às mesmas. Uma única BTF pode consumir vários *gigabytes* de armazenamento, inabilitando a visualização em tempo real. A BTF tem que ser transformada numa representação mais compacta, que permita a respetiva visualização eficiente, sem grande perda de fidelidade para com a aparência real do material [Müller03, Sattler03, Ma04].

Wu et al. [Wu11] apresentaram uma nova representação para a BTF, o modelo esparsos de mistura paramétrica (SPMM). Esta abordagem foi demonstrada pelos autores usando um *ray tracer* paralelo implementado em CPU que, embora tenham obtido visualizações de alta qualidade, não permite taxas de visualização interativas.

Este artigo apresenta uma implementação da SPMM, usando o GPU com o *ray tracer* Optix [Parker10], que obtém taxas de visualização interativas, sem comprometer a qualidade visual. O tempo de síntese depende do número de pontos intersetados que mapeiam numa SPMM, o número de fontes de luz e o número de SPMMs; a escalabilidade é também estudada conforme estes parâmetros.

O visualizador permite mapear SPMMs nos materiais definidos em modelos geométricos, e visualizá-los sob iluminação direta e especular, com uma ou mais luzes pontuais. Adicionalmente a contribuição da iluminação difusa indireta é aproximada por oclusão de ambiente. Este programa está atualmente a ser usado por parceiros na indústria do calçado Português, envolvidos num projeto de financiamento nacional.

\*Trabalho parcialmente financiado pelo projeto QREN nº 13114 TO-PIC Shoe e por Fundos Nacionais através da FCT - Fundação para a Ciência e a Tecnologia no projeto PEst-OE/EEI/UI0752/2011

O artigo está organizado da seguinte maneira: primeiro é apresentado o estado da arte da representação de materiais complexos, juntamente com o formato SPMM; seguidamente é descrita a nossa abordagem para obter taxas de visualização interativas no GPU usando Optix, juntamente com os resultados obtidos e respetiva análise. O artigo termina com conclusões e trabalho futuro.

## 2 MODELOS DE APARÊNCIA E VISUALIZAÇÃO

A maioria dos materiais do mundo real têm uma aparência complexa que pode ser descrita em três níveis [Suykens03, Müller05]: ao nível macroscópico está a geometria do objeto, tradicionalmente modelada com representações explícitas, tais como malhas poligonais. O nível microscópico engloba as interações luminosas num determinado ponto da superfície do objeto, que podem ser representadas com BRDFs. Finalmente, o nível mesoscópico apresenta-se entre estes dois níveis, e é composto pelos vários fenómenos de iluminação mais subtils, como a auto-oclusão, inter-reflexões ou translucência. Para estes fenómenos a BRDF é insuficiente, sendo geralmente substituída por técnicas baseadas em imagens, como o mapeamento de texturas.

*Spatially-Varying BRDFs* (SVBRDF) [McAllister02], podem ser vistas como uma combinação da BRDF com o mapeamento de texturas, permitindo representar materiais com diferentes BRDFs ao longo da sua superfície. Embora representem alguns dos fenómenos ao nível mesoscópico, estas não conseguem capturar a auto-oclusão. A BTF apresenta-se assim como uma alternativa viável para capturar todos estes efeitos. A *Bidirectional Subsurface Scattering Reflectance Distribution Function* (BSSRDF) consegue modelar todos estes fenómenos de transporte de luz, mas esta função é demasiado complexa para ser usada no contexto de visualização interativa, e os sistemas de captura atuais apenas conseguem medir sub-conjuntos desta função [Weyrich08, Müller05].

Note-se que a aparência de um material também pode ser reproduzida seguindo uma abordagem procedural, isto é, definindo algoritmos e ajustando os respetivos parâmetros até que o efeito desejado seja obtido [Ebert02]. No entanto, esta é uma tarefa morosa que necessita de mão de obra qualificada, e não consegue, por vezes, representar fielmente todos os materiais do mundo real. Abordagens baseadas em imagens são cada vez mais populares devido ao uso de câmeras para capturar a aparência, sem exigir mão de obra qualificada, constituindo portanto, uma alternativa adequada para projetos na indústria do calçado Português.

### 2.1 Bidirectional Texture Function

A BTF emergiu como uma alternativa que representa simultaneamente a escala mesoscópica e a microscópica. Esta função modela, para cada comprimento de onda, a aparência do material num ponto da superfície ( $x$ ), para um par de direções de incidência e observação ( $\omega_i, \omega_o$ ). Esta informação é obtida capturando várias imagens do material sob diversas direções de iluminação e de observação.

A BTF pode ser vista como um caso particular da SV-BRDF, dado que se assume que a superfície capturada é planar [Lawrence06]. Conceitualmente, não existem grandes diferenças entre a BTF e a textura 2D, requerendo ambas uma fase de aquisição, representação e síntese.

Uma BTF de boa qualidade, tais como as existentes na base de dados de Bonn [Sattler03], contém  $81 \times 81$  imagens para as direções de iluminação e de observação, cada uma com  $256^2$  *texels* para cada canal de cor RGB. Isto corresponde a cerca de 1.2GB de dados brutos para uma única BTF, sem incluir efeitos com alta gama dinâmica (HDR). De modo a obter taxas de visualização interativas em objetos mapeados com várias BTFs, é necessário comprimir estes dados, preservando as características mais relevantes da aparência do material. Os métodos de compressão devem também aproveitar a redundância existente nos dados de uma forma eficiente, e permitir uma rápida descompressão dos dados. Mais detalhes sobre o estado da arte na captura, representação e síntese de BTFs podem ser consultados em [Müller05] e [Filip09].

### 2.2 O Modelo Esparso de Mistura Paramétrica

Na representação SPMM proposta por Wu et al. [Wu11], os dados capturados são analisados e ajustados a uma combinação linear de funções paramétricas, cada uma definida como uma BRDF rodada e pesada pelo coseno do ângulo definido pela normal e a direção de incidência. A equação 1 descreve tais funções, onde  $f_j(k_j, \cdot)$  é um modelo analítico da BRDF, com parâmetros  $k_j$ .  $R$  é uma rotação que transforma um vetor para o sistema de coordenadas locais definido pela normal  $n_j$ .

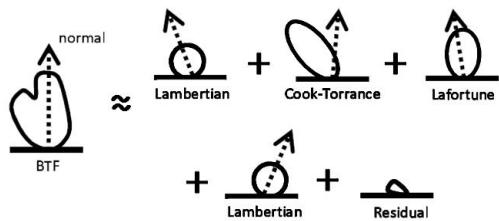
$$\rho_j(\omega_i, \omega_o) = f_j(k_j, R(\omega_i), R(\omega_o))(n_j \cdot \omega_i). \quad (1)$$

Os dados originais no ponto  $x$  podem ser aproximados por uma soma pesada de  $m$  funções, cada uma com peso  $\alpha_j$ , como ilustrado na equação 2 e na figura 1. Os detalhes de aparência mais subtils que não possam ser capturados pelas funções paramétricas, são armazenados numa função residual,  $\epsilon_x$ , que é obtida subtraindo a combinação linear de funções paramétricas da BTF original.

$$\text{BTF}_x(\omega_i, \omega_o)(n_x \cdot \omega_i) = \sum_{j=1}^m \alpha_j \rho_j(\omega_i, \omega_o) + \epsilon_x(\omega_i, \omega_o). \quad (2)$$

Dado que ajustar cada *texel* da BTF é computacionalmente muito exigente, a coerência espacial dos dados é aproveitada com o agrupamento multi-nível dos *texels* (k-means clustering). O algoritmo completo de ajuste é aplicado apenas a alguns *texels* de cada grupo, sendo que as funções paramétricas resultantes são usadas como dicionário para acelerar o ajuste dos restantes *texels*.

A função residual  $\epsilon_x$  é também calculada por grupo. É sugerido que  $\epsilon_x$  possa ser melhorada por análise local dos componentes principais (LPCA), armazenando as funções base resultantes e respetivos coeficientes [Müller03]. O au-



**Figura 1. Ilustração da representação SPMM.**  
A BTF é aproximada por uma soma pesada de modelos analíticos e uma parte residual. Cada modelo analítico tem o seu próprio sistema de coordenadas local. Adaptado de [Wu11].

mento do número de funções base permite melhorias marginais em troca de um maior consumo de memória. O visualizador apresentado permite a adaptação da qualidade de imagem limitando a parte residual a um sub-conjunto das suas funções base; assim, o tempo de síntese é adaptado às capacidades computacionais da placa gráfica.

A aproximação da BTF por uma soma de funções paramétricas conhecidas, fornece uma representação geral da aparência dos materiais, que reduz significativamente o volume de dados, permitindo uma síntese eficiente e, adicionalmente, a edição intuitiva dos seus parâmetros. Wu et al. testaram a SPMM com BTFs da base de dados de Bonn [Sattler03]; a nossa implementação é baseada nestas representações.

### 3 VISUALIZAÇÃO INTERATIVA DE SPMMS

O visualizador interativo usa o motor de *ray tracing* Optix da NVIDIA [Parker10], versão 2.5.1, e foi implementado em duas partes: a primeira corresponde à implementação base, simulando apenas a iluminação direta e raios especulares; a segunda aproxima fenômenos de iluminação global por oclusão de ambiente.

As experiências foram realizadas numa máquina equipada com um processador Intel 2.4GHz de quatro núcleos, 4GB de RAM e uma NVIDIA GeForce GTX 580, apresentando o resultado numa janela de  $512 \times 512$  pixels, com um raio primário por pixel. Todos os testes usam as representações SPMM das BTFs da base de dados de Bonn [Sattler03], que apresentam uma resolução espacial de  $256 \times 256$  *texels* e uma resolução angular de  $81 \times 81$  direções. Todos os testes apresentados usam o SPMM *Wool*, exceto quando indicado o contrário, e todos os coeficientes da função residual são avaliados, para maximizar a qualidade de imagem. Os valores indicados são obtidos da média de uma execução do programa durante sessenta segundos.

#### 3.1 Implementação Base

O modelo de iluminação utilizado inclui apenas as componentes de iluminação direta e reflexões especulares. São simuladas fontes de luz pontuais, sendo a visibilidade de cada uma aferida com um raio sombra. As reflexões es-

peculares são calculadas disparando um raio ao longo da direção de reflexão ideal.

O uso eficiente do GPU requer que os dados do SPMM estejam armazenados na memória de textura da placa, como um array 1D, 2D ou 3D [Fernando04, Pharr05], permitindo assim acessos aleatórios rápidos. Desta forma, a memória de textura armazena, para cada texel, o identificador do grupo, os índices das funções paramétricas, os seus pesos  $\alpha_j$  e os coeficientes da função residual. Para cada grupo armazenam-se as funções paramétricas  $\rho_j$ , os parâmetros correspondentes  $k_j$  e a função residual  $\epsilon_x$ .

O formato SPMM não é apropriado para uma implementação no GPU pois consiste numa combinação linear de diferentes modelos analíticos, e traduzindo diretamente o código CPU para GPU resultaria em bastante divergência e num elevado número de ciclos. Consequentemente, a transformação das estruturas de dados, por forma a adaptar o código de síntese ao modelo de programação do GPU é uma tarefa fundamental.

As estruturas de dados utilizadas são um aspeto crucial pois determinam o consumo de memória e a largura de banda necessária; é importante minimizar estes dois fatores. Dado que as SPMMs geradas da base de dados de Bonn agrupam os *textels* em trinta e dois grupos, o identificador do grupo pode ser representado com um único byte. Outras estruturas que usam valores inteiros representam índices ou identificadores, e podem ser representados com dois bytes. As estruturas para os pesos das funções paramétricas, os seus parâmetros e a função residual são armazenadas no formato vírgula flutuante de precisão simples, com quatro bytes - o menor permitido em Optix.

O segundo aspeto a considerar é que o número de funções paramétricas  $m$  não é o mesmo para cada *texel*, e enviar esta informação dinamicamente para o GPU poderia comprometer o tempo de síntese. Uma abordagem possível seria calcular o número máximo de funções, uniformizando o número de funções avaliadas para cada texel, mas isto resultou num desempenho fraco. A nossa implementação pré-calculta  $m$  para cada *texel* e armazena o resultado numa estrutura adicional. Isto resulta em taxas de síntese mais elevadas, especialmente quando  $m$  varia muito de um *texel* para outro, isto é, quando a BTF apresenta variações locais mais complexas.

O terceiro problema advém da discretização esparsa das direções de captura da BTF. De modo a sintetizar a BTF para outras direções é necessário interpolar os dados das direções capturadas mais próximas. As funções paramétricas não são afetadas por este problema, dado que estas estão definidas para toda a semiesfera superior. No entanto, a função residual necessita de interpolação para evitar o aparecimento de artefactos quando mudam as direções capturadas mais próximas. Estas direções podem ser interpretadas como pontos 3D na superfície da semiesfera e projetadas no plano XY, ignorando a componente Z, resultando, assim, num conjunto de pontos 2D. Aplicando uma triangulação de Delaunay a esse conjunto de pontos, e armazenando os triângulos resultantes noutra es-

trutura de dados, é possível interpolar intersetando um raio com um triângulo e usando as respectivas coordenadas báricêtricas [Pharr04]. Os pesos da interpolação, no caso do raio atravessar esse triângulo, são estas coordenadas. Isto é realizado separadamente para a direção de iluminação e de observação, para um total de nove pesos de interpolação.

Finalmente, as matrizes 2D cujas dimensões excedam o tamanho suportado pela placa são transformadas em fatias de uma matriz 3D, com as dimensões máximas suportadas pela placa gráfica. A figura 2 ilustra as estruturas de dados usadas (algumas das quais são apresentadas na seção seguinte), e o fluxo de dados necessário para sintetizar a SPMM para um ponto da superfície ( $x$ ) e um par de direções ( $\omega_i, \omega_o$ ).

### 3.2 Otimizações

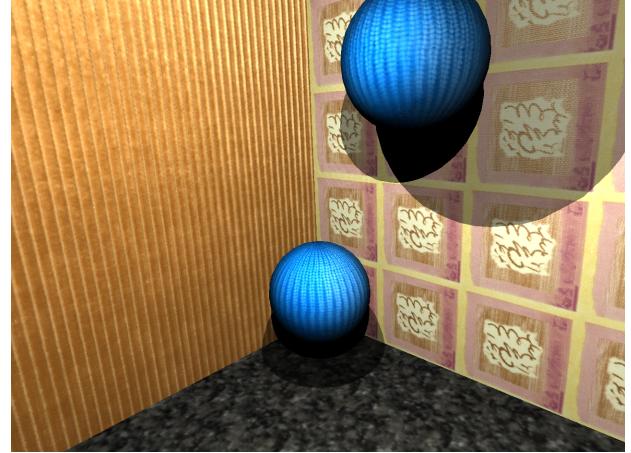
A análise dos perfis de execução indicou que o maior limitador do desempenho era a interpolação da função residual. Dado que esta é baseada na interseção de um raio com todos os triângulos, e estes estão bem distribuídos ao longo de um círculo, usamos uma grelha regular para rapidamente descartar triângulos, limitando assim os testes de interseção a um subconjunto bastante reduzido dos mesmos. A grelha implementada é compacta, com requisitos de memória mínimos [Lagae08], construída no CPU e enviada para o GPU usando dois arrays: as estruturas *Grid Cells* e *Grid Data* no diagrama da figura 2.

A equação 3 é usada para o teste de interseção do raio com um triângulo, onde  $\lambda$  é o vetor de coordenadas báricêtricas,  $r$  é a direção do raio, e  $T$  é a matriz formada pelas coordenadas cartesianas dos vértices do triângulo. Este formato permite a precomputação da matriz inversa  $T^{-1}$  para cada triângulo, reduzindo o teste de interseção no GPU a uma subtração e uma multiplicação vetorial.

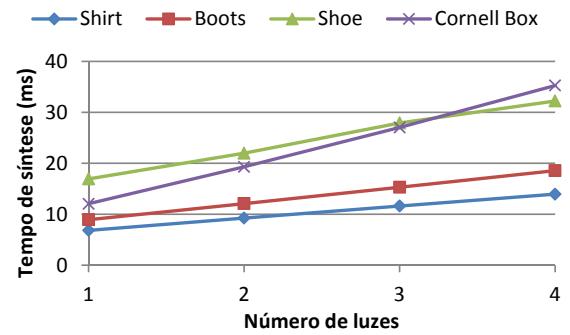
$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = T^{-1}(r - v_3); \quad (3)$$

Adicionalmente, de modo a maximizar a localidade dos dados, a matriz inversa, as coordenadas cartesianas de cada vértice e os correspondentes identificadores são armazenados numa mesma estrutura de dados, para um total de dezasseis valores de vírgula flutuante, por triângulo. Com estas duas otimizações o custo computacional da avaliação da função residual é aproximadamente o mesmo que da avaliação da combinação linear de funções paramétricas, enquanto que anteriormente, o primeiro era cerca de três vezes mais caro que o segundo.

De modo a tirar o máximo partido da largura de banda do GPU, são utilizadas as instruções SIMD para cálculo vetorial, e o acesso aos dados nas texturas é feito nos quatro canais disponíveis, RGBA, sempre que possível. Finalmente, durante o desenho da cena, o número de componentes resultantes da LPCA podem ser limitados pelo utilizador, de modo a adaptar o tempo de síntese às capacidades computacionais da máquina. Nas nossas experiências, esta opção



**Figura 3.** O interior de uma caixa (cena Cornell Box) mapeado com várias SPMMs. O material das esferas é Wool, e das paredes é Corduroy, Wallpaper e Floor tile. Com duas luzes pontuais obtém-se taxas de visualização de  $\approx 50$  FPS.

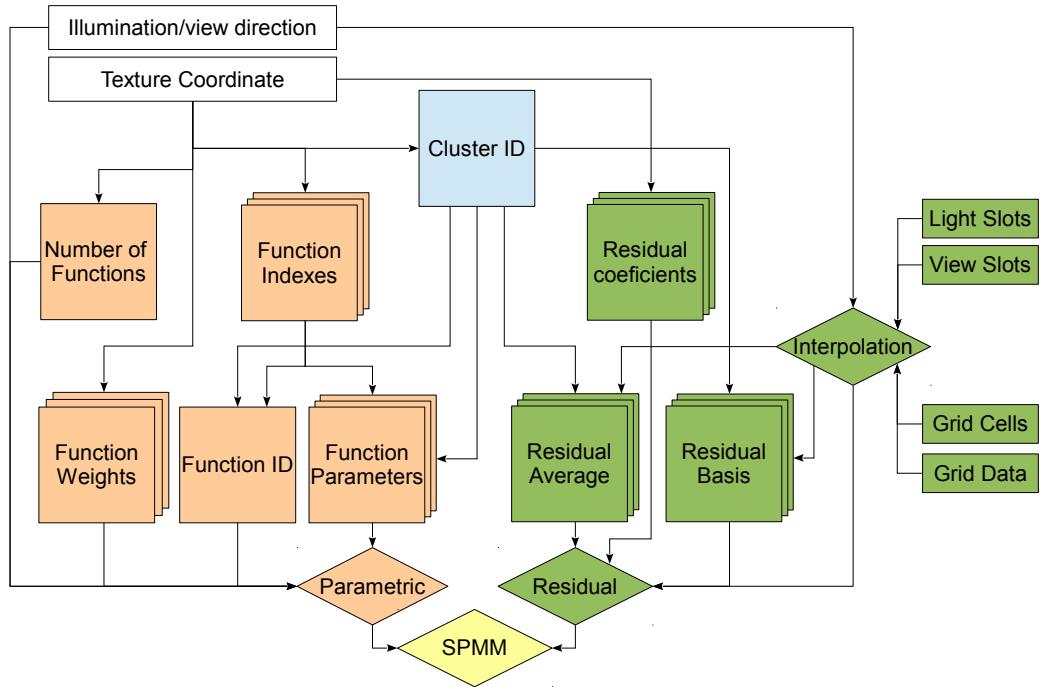


**Figura 4.** Escalabilidade com o número de luzes pontuais, em quatro modelos geométricos diferentes.

reduz bastante o tempo de síntese, tendo um impacto marginal na qualidade de imagem. A imagem 3 demonstra o resultado visual com iluminação direta, usando duas fontes de luz.

A escalabilidade do visualizador com a complexidade da cena é fulcral para o sucesso em ambiente de produção industrial. As medições de desempenho, tendo em conta a escalabilidade com o número de luzes e com o número de SPMMs são apresentadas nas figuras 4 e 5, respetivamente. Dado que o Optix fornece estruturas de aceleração eficientes, as variações de desempenho conforme a complexidade geométrica do modelo não se revelaram significativas.

O tempo de síntese aumenta linearmente com o número de luzes. Isto acontece devido ao cálculo da SPMM conforme o fluxo de dados no diagrama 2. No entanto, tal como indicado nesse diagrama, nem todas as estruturas de dados



**Figura 2. O fluxo de dados no programa de síntese. Os pequenos retângulos, os quadrados e os quadrados empilhados representam, respetivamente, texturas 1D, 2D, e 3D. Os diamantes representam a combinação dos dados de entrada.**

dependem da direção de iluminação; implementações futuras poderão explorar este facto de modo a atingir melhor escalabilidade com o número de luzes pontuais.

Tal como indicado na figura 5, o aumento do número de SPMMs não afeta significativamente o número de FPS obtidos. De facto, o maior limitador do desempenho é o número de pontos intersetados que mapeiam numa SPMM, dado que a estes está associada uma carga computacional muito maior que a pontos com um modelo de iluminação mais simples. No entanto, mesmo que todos os pontos visíveis mapeiem em SPMMs, ou seja,  $512 \times 512$  pontos (modelo *Cornell Box* no gráfico 5(a)), obtém-se uma taxa de visualização de cerca de 62 FPS.

### 3.3 Interreflexões Difusas

A qualidade das imagens sintetizadas depende dos fenómenos de transporte de luz simulados. As interreflexões difusas são especialmente relevantes, pelo que decidimos acrescentar a simulação deste fenómeno no nosso visualizador. Uma vez que este é um processo computacional muito exigente optamos por utilizar a técnica de oclusão ambiente; esta resulta numa aproximação pouco precisa às interreflexões difusas, mas com qualidade suficiente para esta aplicação.

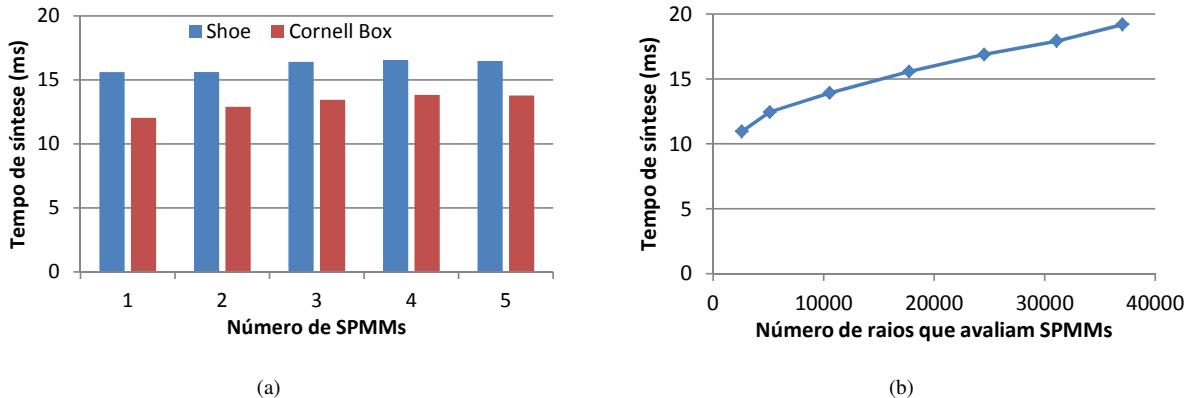
A oclusão ambiente determina qual a percentagem de semiesfera superior que não está ocluída por geometria a uma dada distância do ponto a iluminar; a iluminação ambiente é depois multiplicada por esta percentagem antes de ser adicionada à iluminação incidente neste ponto. Para determinar a oclusão ambiente são disparados  $N$  raios ao

longo da semiesfera, distribuídos estocasticamente com estratificação. A percentagem destes raios que encontram uma interseção até à distância pré-determinada contribui para a oclusão ambiente.

O método descrito atrás tem um custo significativo, proporcional a  $N$ , que inviabiliza taxas de visualização interativas. Desenhamos então uma versão progressiva que dispara apenas um raio de oclusão por ciclo de desenho. O processo é reiniciado sempre que há movimentos da câmera. As imagens iniciais apresentam um ruído elevado, mas como as taxas de visualização se mantêm acima dos trinta FPS, a solução rapidamente converge para um nível de ruído perceptualmente irrelevante. A figura 6 apresentada em apêndice contém os resultados obtidos e compara com as imagens sem interreflexões difusas; o vídeo submetido como material suplementar inclui uma demonstração desta técnica.

## 4 CONCLUSÃO E TRABALHO FUTURO

Apresentamos um visualizador implementado em GPU usando o motor de *ray tracing* Optix, que combina os benefícios do formato compacto da SPMM, uma representação estado-da-arte para BTFs, com o poder computacional permitido pelos GPUs. A escalabilidade com o número de luzes e o número de pontos intersetados que mapeiam numa SPMM é linear; com o número de SPMMs presentes na cena é aproximadamente constante. O visualizador permite mapear várias SPMMs em modelos geométricos, e, combinando técnicas para aproximar fenómenos de iluminação direta e indireta no GPU, obtém



**Figura 5. (a) Escalabilidade com o número de SPMMSs em dois modelos geométricos, o número de raios disparados para SPMMSs é constante. (b) Escalabilidade com o número de raios que avaliam uma SPMM, para o modelo do sapato com três SPMMSs.**

visualizações de alta qualidade e de alta fidelidade a taxas interativas.

Como trabalho futuro gostaríamos de melhorar a escalabilidade e expandir o visualizador para permitir edição interativa dos parâmetros da SPMM. Acreditamos que isto poderá ser útil para os *designers* e artistas da indústria, permitindo a prototipagem virtual rápida de novos produtos. Adicionalmente, técnicas de *displacement mapping* ([Wang03]) podem melhorar a aparência do material e permitir uma melhor percepção da profundidade e até da silhueta do material.

## 5. REFERÊNCIAS

- [Dana99] K. Dana, B. van Ginneken, S. K. Nayar, e J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18:1–34, January 1999.
- [Ebert02] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, e S. Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edição, 2002.
- [Fernando04] R. Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Pearson Higher Education, 2004.
- [Filip09] J. Filip e M. Haindl. Bidirectional texture function modeling: A state of the art survey. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 31(11), nov. 2009.
- [Lagae08] A. Lagae e P. Dutré. Compact, fast and robust grids for ray tracing. *Computer Graphics Forum (Proc. of the 19th Eurographics Symp. on Rendering)*, 27(4), June 2008.
- [Lawrence06] J. Lawrence. *Acquisition and representation of material appearance for editing and rendering*. Tese de Doutoramento, Princeton, NJ, USA, 2006. AAI3214568.
- [Ma04] W. Ma, S. Chao, B. Chen, C. Chang, M. Ouhyoung, e T. Nishita. An efficient representation of complex materials for real-time rendering. Em *Proc. of the ACM symp. on Virtual reality software and technology*, VRST '04, páginas 150–153, New York, NY, USA, 2004. ACM.
- [McAllister02] D. McAllister, A. Lastra, e W. Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. Em *Proc. of the ACM SIGGRAPH/EUROGRAPHICS conf. on Graphics hardware*, HWWS '02, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
- [Müller03] G. Müller, J. Meseth, e R. Klein. Compression and real-time rendering of measured btfs using local pca. Em T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, e R. Westermann, editores, *Vision, Modeling and Visualisation 2003*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, Novembro 2003.
- [Müller05] G. Müller, J. Meseth, M. Sattler, R. Sarlette, e R. Klein. Acquisition, synthesis, and rendering of bidirectional texture functions. *Computer Graphics Forum*, 24(1):83–109, 2005.
- [Parker10] S. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, e M. Stich. Optix: a general purpose ray tracing engine. *ACM Trans. Graph.*, 29(4), Julho 2010.
- [Pharr04] M. Pharr e G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- [Pharr05] M. Pharr e R. Fernando. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005.
- [Sattler03] M. Sattler, R. Sarlette, e R. Klein. Efficient and realistic visualization of cloth. Em *Eurographics Symp. on Rendering 2003*, Junho 2003.
- [Suykens03] F. Suykens, K. V. Berge, A. Lagae, e P. Dutré. Interactive rendering with bidirectional texture functions. *Computer Graphics Forum*, 22:463–472, 2003.
- [Wang03] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, e H. Shum. View-dependent displacement mapping. *ACM Trans. Graph.*, 22(3):334–339, Julho 2003.
- [Weyrich08] T. Weyrich, J. Lawrence, H. Lensch, S. Rusinkiewicz, e T. Zickler. Principles of appearance acquisition and representation. Em *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, New York, NY, USA, 2008. ACM.
- [Wu11] H. Wu, J. Dorsey, e H. Rushmeier. A sparse parametric mixture model for btf compression, editing and rendering. *Computer Graphics Forum*, 30(2):465–473, 2011.



(a) Um sapato mapeado com três SPMMs, sob iluminação direta e espelcular. Taxas de visualização de  $\approx 52$  FPS.



(b) O mesmo sapato mapeado sob iluminação direta, espelcular e difusa indireta (occlusão ambiente). Taxas de visualização de  $\approx 37$  FPS.



(c) Um par de botas, com a SPMM Wool, sob iluminação direta e espelcular. Taxas de visualização de  $\approx 84$  FPS.



(d) O mesmo par de botas, sob iluminação direta, espelcular e difusa indireta (occlusão ambiente). Taxas de visualização de  $\approx 50$  FPS.



(e) Uma camisola, com a SPMM Pulli, sob iluminação direta e espelcular. Taxas de visualização de  $\approx 68$  FPS.



(f) A mesma camisola, sob iluminação direta, espelcular e difusa indireta (occlusão ambiente). Taxas de visualização de  $\approx 32$  FPS.

**Figura 6. Vários modelos geométricos mapeados com SPMMs, sob iluminação direta e espelcular (coluna da esquerda) e sob iluminação difusa indireta por oclusão de ambiente (coluna da direita).**

