Marco Diego Aurélio Mesquita

Coherent bidirectional path tracing: uma abordagem livre de ruído para estimativa de iluminação indireta

Fortaleza

2010

Marco Diego Aurélio Mesquita

Coherent bidirectional path tracing: uma abordagem livre de ruído para estimativa de iluminação indireta

Dissertação apresentada ao Departamento de Computação da Universidade Federal do Ceará como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador:
Creto Augusto Vidal

Coorientador:
Joaquim Bento Cavalcante Neto

UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO
CRAB - COMPUTAÇÃO GRÁFICA, REALIDADE VIRTUAL E ANIMAÇÃO

Fortaleza

2010

Agradecimentos

Este trabalho não se teria realizado sem a colaboração e o apoio de algumas pessoas. Agradece-se a Creto Augusto Vidal e Joaquim Bento Cavalcante Neto por, respectivamente, orientação e coorientação fornecidas; a CAPES e FUNCAP pelo finaciamento; aos desenvolvedores de cada um dos *softwares* necessários à realização do trabalho (uma lista que é demasiadamente extensa para se manter no escopo do texto); aos pesquisadores envolvidos de forma direta ou indireta em cada um dos tabalhos relacionados (uma lista que também é demasiadamente longa para se manter no escopo do texto); a Eduardo Gurgel e Charles Welton por terem sido fundamentais com implementações iniciais e colaborações posteriores; a Jean-Philippe Grimaldi por toda instrução, verificação de resultados, ajuda com implementação, dicas, críticas, explicações, correções, esclarecimentos, informações e uma longa lista de contribuições e colaborações (que também é demasiadamente longa para se manter no escopo do texto); a Yuri Lennon e Markos Freitas por correções, revisões e ajudas; a Francisca Lígia Aurélio Mesquita e Paulo Abner Aurélio Mesquita; a Karolina Costa de Sousa por todo o carinho, atenção, ânimo e inspiração; a Antônia Inês Mesquita da Costa e a Aurélio da Costa Neto a cuja memória este trabalho é dedicado.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Motivação

Renderização é o processo de se gerar uma imagem a partir da descrição de uma cena. Obviamente, esse processo pode ser feito usando-se métodos diversos, com resultados e propriedades igualmente diversos. Distintas características da cena ou da imagem a ser gerada podem ser realçadas ou obtidas, dependendo do método de renderização utilizado. O presente trabalho, interessa-se em investigar propriedades de métodos de renderização capazes de gerar imagens que representem bem a realidade de um dado cenário.

Um modelo de iluminação é dito global se a forma como a luz interage com um ponto é calculada levando-se em consideração as interações da luz com outros pontos da cena. Alguns modelos de iluminação global são relevantes para o trabalho proposto. O presente trabalho é fortemente inspirado em alguns métodos (modelos de iluminação global) já existentes: *Path Tracing*, *Bidirectional Path Tracing* e *Coherent Path Tracing*.

Outros métodos de renderização também existem, mas podem apresentar algumas desvantagens: podem ser tendenciosos [1], podem assumir restrições na cena [2], podem exigir características muito específicas sobre as superfícies [3], ou não são fisicamente realistas [4]. Os métodos preferidos para o presente trabalho, no entanto, apresentam também uma desvantagem relevante: a presença de ruído.

1.2 Objetivos

No presente trabalho, apresenta-se um método de renderização que é livre de ruído e, ainda assim, tão geral quanto *Bidirectional Path Tracing*. No entanto, o método proposto tem a característica de gerar padrões durante o processo de renderização. O principal objetivo deste trabalho é investigar e comparar o desempenho do método desenvolvido com alguns métodos atualmente disponíveis. São exibidas algumas cenas comparando métodos de renderização pro-

pensos a padrões e métodos de renderização propensos a ruídos.

1.3 Estrutura da dissertação

No Capítulo 2, são descritos métodos que inspiraram o desenvolvimento do método proposto. São apresentados alguns exemplos e é fornecida informação suficiente para uma implementação independente de cada um dos métodos. Também são levantados alguns pontos fracos de cada um dos métodos que são atacados pelo método proposto.

No Capítulo 3, é descrito como funciona o método proposto. É dada informação suficiente para um implementação independente do método e as estratégias utilizadas são justificadas e discutidas.

No Capítulo 4, são apresentados resultados obtidos com cada um dos métodos. As cenas de teste são descritas e justificadas. São dadas tabelas comparativas de desempenho dos métodos com o objetivo de clarificar o melhoramento feito.

No Capítulo 5, os objetivos alcançados são mostrados e é feita uma discussão de problemas ainda existentes. As principais contribuições são descritas e são discutidos trabalhos futuros para o melhoramento do método atual.

2 *Trabalhos Relacionados*

2.1 Introdução

Neste capítulo, descrevem-se algumas técnicas de renderização mais modernas que serviram de inspiração no desenvolvimento do método proposto. Mais detalhes sobre o funcionamento do método proposto são descritos no Capítulo 3. Portanto, o texto limita-se a apresentar e a explicar como funcionam os métodos de renderização mais fortemente relacionados com o método proposto.

Os principais métodos descritos neste capítulo são: *Path Tracing* (PT) e *Bidirectional Path Tracing* (BDPT). À medida que esses métodos são descritos, alguns métodos derivados também são apresentados. Ao se descrever PT, apresenta-se *Coherent Path Tracing* (CPT) e, ao se descrever BDPT, apresentam-se dois métodos derivados de BDPT: *Metropolis Light Transport* (MLT) e *Energy Redistribution Path Tracing*.

2.2 Path Tracing

Path Tracing (PT) foi desenvolvido por James Kajya em 1986 [5]. O trabalho feito por Kajya tem também relevância histórica, já que no mesmo artigo em que Kajya descreve a equação de renderização (vide Equação 2.1) é descrita também a técnica de *Path Tracing*.

A técnica de *Path Tracing* é fielmente inspirada na equação de renderização e tem sido, desde então, usada como critério de comparação entre métodos de renderização. Uma vez que o método é fielmente inspirado na equação de renderização o tipo de imagem obtido é bastante fiel à realidade. Assim, os resultados obtidos por essa técnica são usados como padrões com os quais as imagens geradas com outras técnicas são comparadas para avaliar o critério de realismo. Todavia, o método de *Path Tracing* não teve inicialmente muito uso fora de círculos de pesquisa, onde servia como base de comparação de métodos de renderização, dado o longo tempo necessário para se obter uma imagem. Hoje em dia, com computadores substancialmente

mais rápidos do que os que eram disponíveis na época de sua apresentação, o método pode ser testado com computadores comuns já amplamente disponíveis.

Uma compreensão mais cuidadosa da equação de renderização, (Equação 2.1), é necessária para a compreensão do método de *Path Tracing*.

A equação de renderização é uma descrição, com base nos princípios físicos, de como a luz incidente em um ponto específico da cena contribuirá para a imagem gerada. É claro que a equação de renderização é muito mais geral do que a aplicação para geração de imagem, mas, por uma simples questão de escopo, o texto se limita a ver a equação de renderização apenas como uma ferramenta para apoiar técnicas de renderização de imagem. Desde sua apresentação, a equação de renderização tem sido usada nos principais métodos de renderização desenvolvidos com base em princípios físicos.

De forma simplificada, a equação de renderização pode ser escrita como

$$L_o(x,\mathbf{w}) = L_e(x,\mathbf{w}) + \int_\Omega f_r(x,\mathbf{w}',\mathbf{w}) L_i(x,\mathbf{w}') \mathbf{N}(x)\cdot\mathbf{w}' d\mathbf{w}' \tag{2.1}$$

onde:

$d\mathbf{w}'$ é a área diferencial usada na integração,

Ω é o espaço de integração (hemisfério em torno do ponto x),

$L_o(x,\mathbf{w})$ é a radiância sendo emitida pelo ponto x na direção $\mathbf{w}$,

$\int_\Omega f_r(x,\mathbf{w}',\mathbf{w}) L_i(x,\mathbf{w}') d\mathbf{w}'$ indica como a luz que incide no ponto x é retransmitida através da direção $\mathbf{w}$,

$L_e(x,\mathbf{w})$ é a luz naturalmente emitida pela superfície no ponto x na direção $\mathbf{w}$,

$f_r(x,\mathbf{w}',\mathbf{w})$ é a BRDF (*Bidirectional Reflectance Distribution Function*) da superfície no ponto x,

$L_i(x,\mathbf{w}')$ é a luz incidente pela direção $\mathbf{w}'$ no ponto x,

$\mathbf{N}(x)\cdot\mathbf{w}'$ é devido a Lei de Lambert [6].

A radiância emitida pelo ponto x na direção $\mathbf{w}$ irá influenciar diretamente a cor que se quer computar em um dado pixel. Em um pensamento matematicamente mais rigoroso, renderizar imagens é simplesmente uma questão de se resolver a equação de renderização para cada pixel.

A forma como a luz se espelha (reflete) em um dado ponto de uma superfície é determinada pela BRDF no dito ponto. A BRDF é definida na forma de uma função que retorna quanto da luz chegando da direção $\mathbf{w}'$ no ponto x é retransmitida na direção $\mathbf{w}$ [7].

Pode-se também usar formas diferentes da equação de renderização ao longo do texto. Já que alguns algoritmos explicitamente diferenciam alguns termos como, por exemplo, usar o ângulo de incidência "fora" da BRDF. Não será usada uma notação fixa no texto (embora haja esforço para manter a notação consistente e evitar transições bruscas) já que alguns algoritmos e casos são melhores descritos e compreendidos usando uma notação específica mais adequada.

2.2.1 Visão geral do algoritmo

De forma simplificada, para estimar o fluxo em um determinado pixel, o método de *Path Tracing* traça um caminho, com origem no observador, que passa por aquele pixel e examina como as fontes luminosas da cena interagem com os objetos interceptados pelo caminho, contribuindo para o fluxo luminoso ao longo desse caminho. Um visão em forma de algoritmo do método é:

Algoritmo 2.1: Algoritmo de path tracing.

```
1  para cada pixel da imagem faça
2      cor = 0;
3      para cada amostra faça
4          traçar raio a partir do observador passando pelo pixel;
5          cor = cor + tracar(raio);
6      cor = cor/namostras;
7      atualiza pixel;
8  tracar(raio)
9      ponto = achar interseção mais próxima;
10     cor = iluminacao(ponto, normal);
11     retorna cor;
12 iluminacao(ponto, normal)
13     cor = estimativa de iluminação com fontes de luz;
14     cor = cor + tracar(raio refletido aleatoriamente);
15     retorna cor;
```

2.2.2 Traçado de caminho a partir do observador

A Figura 2.1 ilustra um caminho traçado a partir do observador.

Inicialmente, o primeiro vértice do caminho é obtido buscando-se a primeira interseção entre o cenário e o raio que parte do observador e passa pelo pixel cujo fluxo se quer estimar. Os vértices subsequentes são definidos como a primeira interseção entre o cenário e o raio lançado em uma direção aleatória a partir do vértice corrente. O número de vértices do caminho e estratégias de amostragem são parâmetros do algoritmo.

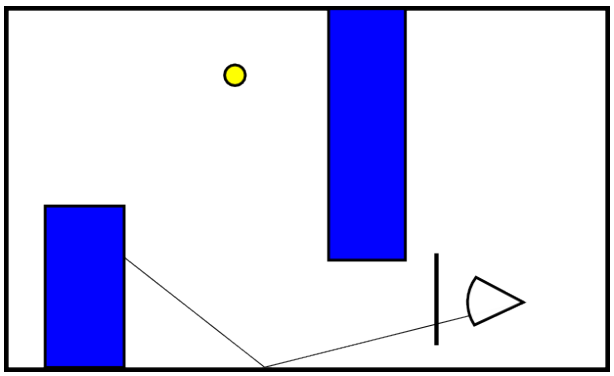


Figura 2.1: Caminho sendo traçado a partir do observador.

2.2.3 Estimativa de fluxo em um pixel

Escrevendo-se a equação de renderização como $L_o = L_e + L_r$ obtem-se

$$L_r(x, \mathbf{w}) = \int_{\Omega} f_r(x, \mathbf{w}, \mathbf{w}') L_i(x, \mathbf{w}') \mathbf{w}' \cdot \overrightarrow{n} \, dw', \tag{2.2}$$

que pode ser calculado de forma ingênua usando Monte Carlo como

$$L_r(x, \mathbf{w}) \approx \frac{2\pi}{N} \sum_{j=1}^{N} f_r(x, \mathbf{w}, \mathbf{w}_j) L_i(x, \mathbf{w}_j) \mathbf{w}_j \cdot \overrightarrow{n} \tag{2.3}$$

Estimar a equação de renderização dessa forma equivale a fazer um *Ray Tracing* distribuído. Jensen [8] explica que é melhor estimar o caminho sem ramificações. Isso é obtido fazendo-se $N = 1$

$$L_r(x, \mathbf{w}) \approx 2\pi f_r(x, \mathbf{w}, \mathbf{w}_1) L_i(x, \mathbf{w}_1) \mathbf{w}_1 \cdot \overrightarrow{n} \tag{2.4}$$

O método de *Path Tracing* é recursivo, calcular L_i é feito da mesma forma que L_o tomando o cuidado de se fazer $L_i = L_e$ no nível máximo de profundidade.

Ao se fazer $N = 1$, cada pixel é estimado através de um único caminho. Para se ter uma estimativa melhor, o mesmo pixel é estimado várias vezes (usando-se caminhos diferentes, mas sempre mantendo $N = 1$) e, ao final, a estimativa é obtida como uma média das estimativas obtidas até então.

2.2.4 Geração da imagem renderizada

A direção $\mathbf{w}'$ é escolhida aleatoriamente. Uma imagem renderizada inteiramente dessa forma pode ter aproximações muito ruins, dependendo das direções escolhidas. Para sanar esse problema, um processo de amostragem é realizado. Assim, para cada pixel, são computadas N estimativas $L_{o_1}, L_{o_2}, ..., L_{o_i}, ..., L_{o_N}$; e a média das estimativas é calculada como

$$L_o = \frac{1}{N} \sum_{i=1}^{N} L_{o_i}. \tag{2.5}$$

A imagem mostrada na Figura 2.2 foi renderizada com 5 amostras por pixel:

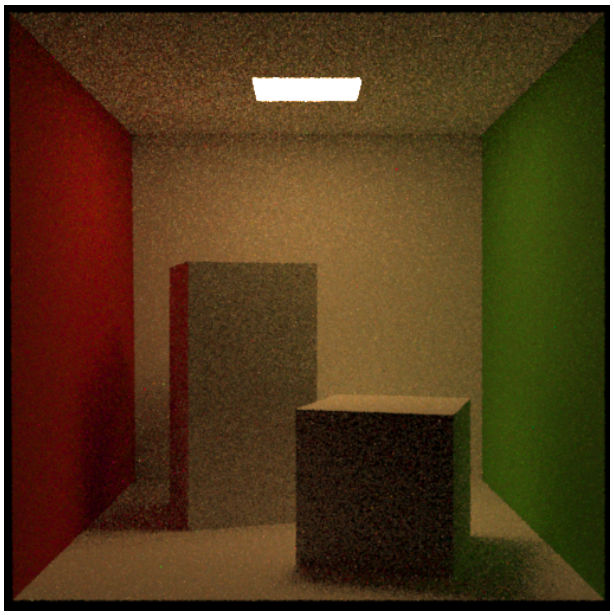


Figura 2.2: Renderização usando Path Tracing.

Devido à aleatoriedade do algoritmo, as imagens geradas podem apresentar ruídos. A tentativa de minimizar os ruídos através do aumento do número de amostras em cada pixel torna a estratégia computacionalmente cara. A relação custo/ruído é um aspecto comum em técnicas de renderização baseadas em algoritmos estocásticos. Uma das abordagens recentes para redução de ruído em *Path Tracing* é a técnica chamada *Coherent Path Tracing*, descrita na Seção 2.4.

2.3 Bidirectional Path Tracing

Bidirectional Path Tracing (BDPT) foi desenvolvido independentemente por Eric Veach [9] e Eric Lafortune em 1993 [10]. A ideia de BDPT consiste em não apenas construir caminhos a partir do obserador, como é feito em PT, mas também construir caminhos a partir da fonte de luz. Usando-se algumas estratégias, o caminho traçado a partir do observador e o caminho traçado a partir da fonte de luz são combinados dando origem a uma outra quantidade de caminhos e a um maior número de estimativas.

2.3.1 Visão geral do algoritmo

O Algoritmo 2.2 ilustra o método de BDPT.

Algoritmo 2.2: Algoritmo de bidirectional path tracing.

```
1  para cada amostra faça
2      para cada pixel faça
3          cor = tracar caminhos(posição do pixel);
4  tracar caminhos(posição)
5      gerar caminho a partir do observador;
6      gerar caminho a partir da fonte de luz;
7      cor = Combinar(caminho do observador, caminho da fonte de luz);
8      Retorna cor;
9  combinar(caminho do observador, caminho da fonte de luz)
10     para cada par de vértices (x[i], y[i]) visíveis entre si faça
11         calcular o peso para o caminho x[i] − y[i];
```

Esse algoritmo pode ser dividido em quatro partes principais: Traçado de caminho a partir do observador, Traçado de caminho a partir da fonte luminosa, Estimativa de fluxo em um pixel, Geração da imagem renderizada.

2.3.2 Traçado de caminho a partir do observador

O traçado de caminho a partir do observador se faz da mesma forma que em PT. Note-se que o número de vértices é um parâmetro do algoritmo. Em [9] e [10], critérios de parada para a amostragem de um caminho feito a partir do observador são discutidos. Também o número de vértices desse caminho é crítico com relação ao tipo de efeito que se pretende obter com o método. Um caminho mais longo traçado a partir do observador será mais propenso a captar

efeitos de cáustica, os quais são mais facilmente modelados (traçados) a partir do observador.

Note que, como o método é baseado em princípios físicos, usando-se caminhos curtos pode-se perder efeitos que exijam mais quicadas. Porém, o número total de vértices num caminho usado para fazer uma estimativa depende também do caminho traçado a partir da fonte de luz. Dessa forma, embora alguns efeitos possam ser perdidos com caminhos mais curtos, a principal influência dos comprimentos dos caminhos se dá em quão rápido um tipo de efeito convergirá, dependendo de quão mais fácil o efeito pode ser amostrado (a partir do observador ou a partir da fonte de luz).

Um caminho traçado a partir do observador é idêntico ao que é feito na Figura 2.1.

2.3.3 Traçado de caminho a partir da fonte luminosa

Uma vez que se tem um caminho traçado a partir do observador, segue-se para o passo de se traçar um caminho a partir da fonte de luz. As questões relativas ao comprimento do caminho traçado a partir da fonte de luz são as mesmas a serem levadas em consideração quando se faz o traçado de um caminho a partir do observador.

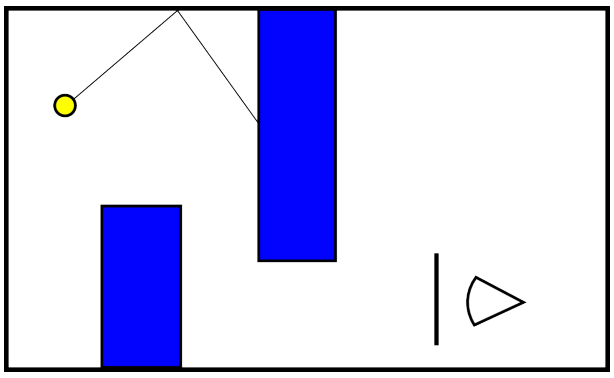


Figura 2.3: Caminhos traçado a partir da fonte de luz.

2.3.4 Estimativa de fluxo em um pixel

A última fase na estimativa de fluxo passando por um pixel em BDPT se dá na combinação dos dois caminhos: o caminho traçado a partir do observador e o caminho traçado a partir da fonte de luz. Nessa fase, tenta-se ligar cada vértice de um caminho a cada um dos vértices do outro caminho a fim de se obter um caminho que siga do observador à fonte de luz. As Figuras 2.1 e 2.4 ilustram, respectivamente, um caminho a partir do observador e a combinação dos caminhos oriundos da fonte luminosa e do observador.

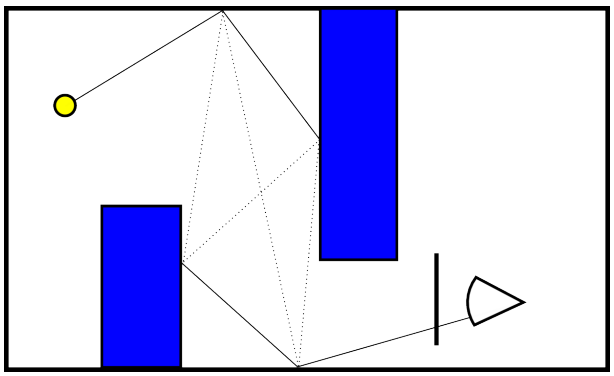


Figura 2.4: Combinando caminhos.

Essa combinação de ambos os caminhos constrói um número maior de caminhos ligando o observador à fonte de luz. Não se pode simplesmente fazer uma média aritmética desses caminhos ao se fazer a estimativa de fluxo. Usando-se a notação de que $\langle\Phi\rangle$ é o valor médio de Φ, em [10] é descrito que se deve fazer uma média ponderada na forma

$$\langle\Phi\rangle = \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_e-1} \omega_{ij}\langle C_{ij}\rangle, \tag{2.6}$$

onde:

$\langle\Phi\rangle$ é a estimativa de fluxo,

N_l é o número de vértices do caminho traçado a partir da fonte de luz,

N_e é o número de vértices do caminho traçado a partir do observador,

$\langle C_{ij}\rangle$ é a estimativa de fluxo do caminho formado ligando-se o i-ésimo vértice do caminho traçado a partir da fonte de luz ao j-ésimo vértice do caminho traçado a partir do observador (A Figura 2.5 ilustra caminhos obteníveis usando-se esse tipo de ligação), e

ω_{ij} é o peso atribuído a estimativa $\langle C_{ij}\rangle$.

O valor de $\langle C_{ij}\rangle$ pode ser calculado (simplificadamente) da mesma forma que a estimativa é feita em PT. Um tratamento mais cuidadoso de situações especiais é descrito em [9].

O cálculo de ω_{ij} exige que pesos para caminhos com o mesmo número de vértices somem 1. Assim , para $N = (0,1,...)$, é necessário

$$\sum_{i=0}^{N} \omega_{i,N-i} = 1. \tag{2.7}$$

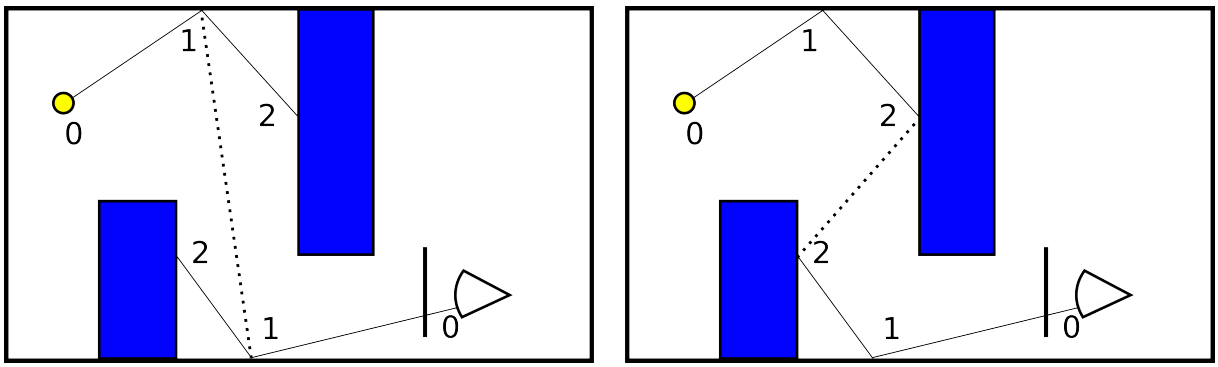


Figura 2.5: Caminhos plausíveis entre a fonte de luz e o observador.

Note-se que, se o caminho traçado a partir da fonte de luz tiver um único vértice, na própria fonte de luz, o método de BDPT se confunde com o método PT. No entanto, o objetivo aqui é obter um método de renderização diferente de PT. Em [10], é descrita a seguinte estratégia para o cálculo de ω_{ij}:

$$\omega_{ij} = \begin{cases} \prod_{k=0}^{j-2} W_{k+1} & \text{para } i = 0, \\ 0 & \text{para } j = 0, \text{ e} \\ (\prod_{k=0}^{j-2} W_{k+1})(1 - W_{j+1}) & \text{caso contrário.} \end{cases} \tag{2.8}$$

Os valores de W_j podem ser escolhidos com alguma liberdade, no entanto, em [10], é justificado que W_j deve ser porporcional à especularidade da superfície no j-ésimo vértice do caminho traçado a partir do observador. Assim, para superfícies muito especulares, W_j aproxima-se de 1 e, em superfícies difusas, W_j aproxima-se de 0.

2.3.5 Geração da imagem renderizada

Assim como em PT, em BDPT, são necessárias várias estimativas para se calcular uma média e se obter uma imagem renderizada. A vantagem de BDPT sobre PT está associada principalmente ao maior número de caminhos que podem ser gerados ligando pontos dos caminhos traçados a partir do observador a pontos dos caminhos traçados a partir da fonte luminosa. Note-se que, por exemplo, admitindo-se que o caminho traçado a partir do observador tenha M vértices e o caminho traçado a partir da fonte de luz tenha N vértices, a combinação dos dois caminhos tem potencial para gerar até $M \cdot N$ estimativas. Cada uma dessas novas estimativas é bem mais barata do que traçar um novo caminho inteiro como é feito em PT, uma vez que menos cálculos de interseção são necessários por vértice.

Usando-se BDPT ao invés de PT, pode-se obter um número bem maior de estimativas para cada pixel a partir de apenas dois caminhos (um traçado a partir do observador e um outro traçado a partir da fonte de luz). Com um número maior de estimativas por pixel, obtidas com um número menor de caminhos, o método pode convergir mais rapidamente do que PT.

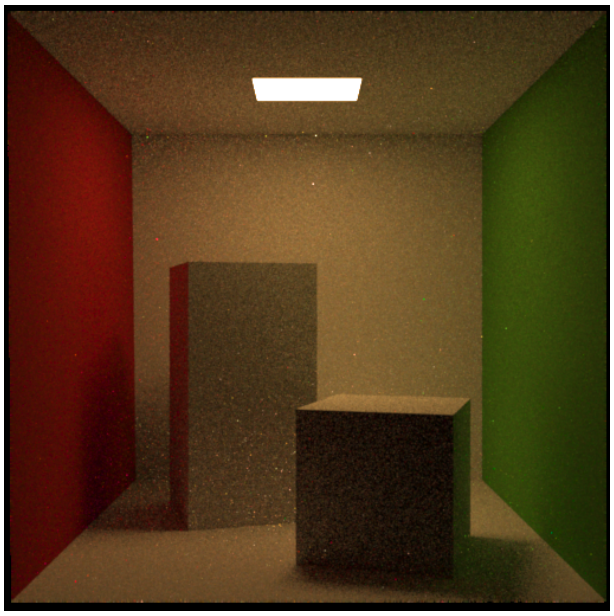


Figura 2.6: Renderização usando BDPT.

Note-se, que há situações em que se deve exercer cautela. Por exemplo, não é prático traçar um caminho a partir do Sol para a Terra (partindo-se de direções aleatórias a partir do Sol) a fim de estimar-se a iluminação de uma sala em penumbra, mas, em situações de cenas fechadas, com caminhos que podem facilmente ser amostrados a partir da fonte de luz para o observador, BDPT é consideravelmente melhor que PT [8].

2.4 Coherent Path Tracing

Coherent Path Tracing (CPT) é uma técninca que foi desenvolvida em 2009 por Iman Sadeghi [11]. *Coherent Path Tracing* trata do mais perceptível problema de *Path Tracing*, o ruído, de forma bem simples. Para estimar-se o fluxo através de um pixel, uma quantidade infinita (no caso contínuo) de caminhos pode ser utilizada. A origem do ruído em *Path Tracing* advém do fato de que pixels vizinhos podem ser estimados por caminhos muito diferentes, dessa forma a imagem renderizada terá ruído. CPT usa uma estratégia para escapar dessa limitação.

2.4.1 Visão geral do algoritmo

A Figura 2.7 ilustra uma estimativa de renderização usando *Path Tracing* com caminhos bem diferentes para pixels próximos. Isso, propicia o aparecimento de ruídos na imagem renderizada. Assim, para tentar eliminar os ruídos, a forma mais ingênua seria usar um número muito grande de estimativas.

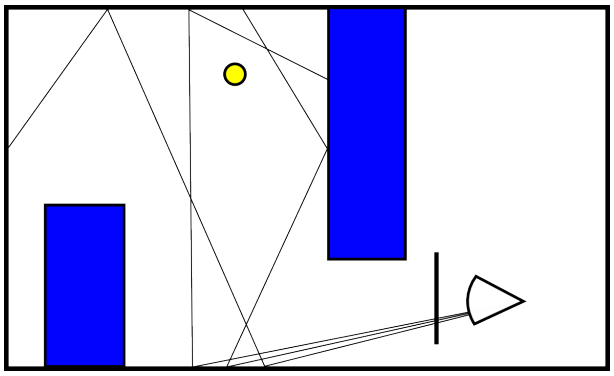


Figura 2.7: Caminhos aleatórios para Path Tracing.

Note que a direção do próximo raio (quicada) em cada vértice do caminho traçado é aleatória. De certa forma, uma estimativa feita usando-se *Path Tracing* pode, então, ser vista como $q(\overrightarrow{u})$ onde $\overrightarrow{u}$ é um vetor em que cada uma de suas u_i componentes é um valor aleatório (não necessariamente um único número) que será usado para definir a direção de uma quicada. Usando-se essa ideia, obter uma renderização Q usando-se várias estimativas em *Path Tracing* pode ser vista como:

$$Q = \frac{1}{n}\sum_{k=1}^{n} q(\overrightarrow{u}(k)) \tag{2.9}$$

onde:

Q é a renderização (estimativa) obtida como uma média de renderizações intermediárias;

q é uma renderização intermediária; e

$\overrightarrow{u}$ é um vetor de valores aleatórios usados para definir os caminhos em uma renderização por *Path Tracing*.

Note que $\overrightarrow{u}(k)$, indica que $\overrightarrow{u}$ é atualizado a cada nova estimativa. Ainda assim, por conta da aleatoriedade do método, não há garantias de que $u_i = u_j$, ou seja, não há garantias de que quaisquer componentes de $\overrightarrow{u}$ sejam iguais. Dessa forma, os caminhos são aleatórios,

e necessárias várias estimativas para obter convergência na estimativa. Já que esse processo representa simplesmente uma integração de Monte Carlo de maneira ingênua, é de se esperar que essa estimativa convirja.

A ideia usada em CPT para resolver o problema do ruído é fazer $u_i = u_j$ para todas as compoentes de $\overrightarrow{u}$.

O Algoritmo 2.3 descreve o método CPT cujos passos principais são detalhados nas subseções 2.4.2 a 2.4.4.

Algoritmo 2.3: Algoritmo de coherent path tracing.

```
1  para cada amostra faça
2      defina vetor de valores aleatórios;
3      para cada pixel da imagem faça
4          trace raio a partir do observador passando pelo pixel;
5          cor = tracar(raio);
6          atualize pixel;
7  tracar(raio)
8      ache interseção mais próxima;
9      cor = iluminacao(ponto, normal);
10     retorna cor;
11 iluminacao(ponto, normal)
12     cor = estimativa de iluminação com fontes de luz;
13     cor = cor + tracar(raio refletido aleatoriamente usando vetor atual);
14     retorna cor;
```

2.4.2 Traçado de caminho a partir do observador

Intuitivamente, o que o algoritmo faz é garantir que, ao longo de uma renderização da imagem inteira (estimativa da imagem), os caminhos usados para estimar-se os pixels quiquem da mesma forma (vide Figura 2.8).

Por conta de caminhos coerentes, a imagem torna-se livre de ruído, porém o método é propenso a padrões.

2.4.3 Estimativa de fluxo em um pixel

A estimativa de fluxo em CPT é feita da mesma forma que em PT. Tal forma é descrita na seção 2.2.3.

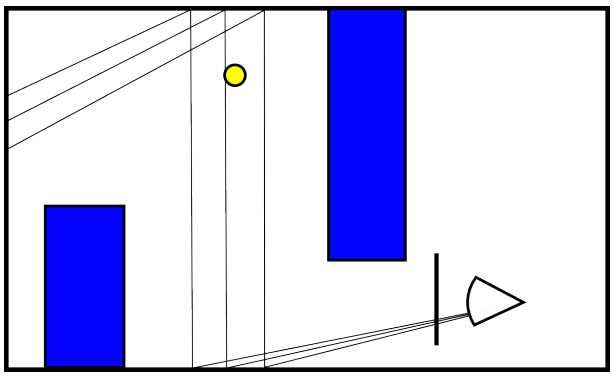


Figura 2.8: Caminhos coerentes.

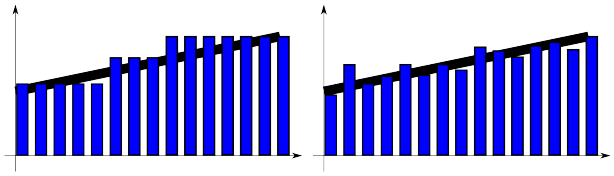


Figura 2.9: Padrões e ruído.

2.4.4 Geração da imagem renderizada

Assim como PT e BDPT, CPT precisa de várias estimativas para se renderizar uma imagem. CPT é uma boa solução em situações onde os padrões são aceitáveis ao invés de ruídos. A Figura 2.10 foi renderizada usando CPT:

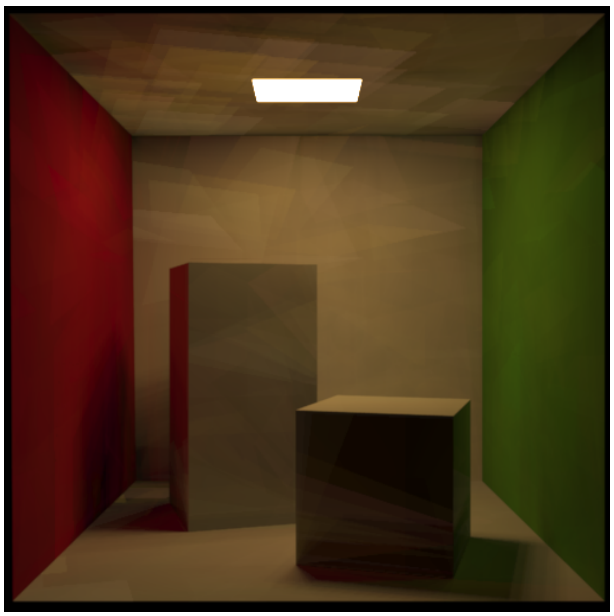


Figura 2.10: Renderização usando coherent path tracing.

Em [11], Sadeghi também aprensentou estratégias para diminuir o quão perceptíveis os padrões podem se tornar. A ideia é usar $\overrightarrow{u}$ diferente para grupos de pixels. A Figura 2.11 ilustra uma tela de pixels, onde os pixels de mesma cor são estimados usando o mesmo $\overrightarrow{u}$.

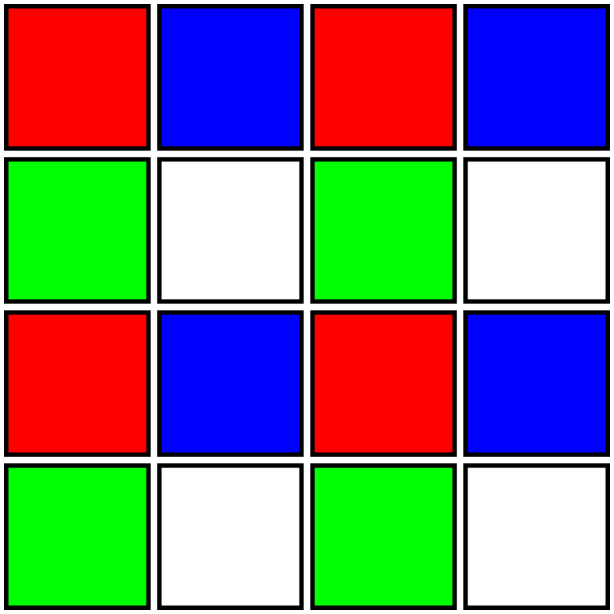


Figura 2.11: Matriz de pixels.

Embora o método funcione, isso simplesmente gera alguma forma de "ruído padronizado".

Um outro aspecto interessante de CPT levantado pelo autor é o fato de que, por conta de os caminhos serem coerentes, o método é paralelizável.

2.5 Métodos derivados de BDPT

Embora haja vários métodos derivados do BDPT, nesta seção são apresentados dois dos principais: *Metropolis Light Transport* e *Energy Redistribution Path Tracing*.

2.5.1 Metropolis Light Transport

Metropolis Light Transport (MLT) é uma técnica apresentada por Veach e Guibas em 1997 [12]. A técnica trata de, primeiramente, encontrar um caminho entre o observador e uma fonte de luz e, em seguida, aplicar mutações no caminho de forma a conseguir outras amostras para a estimativa de radiância em um determinado ponto da cena.

Note-se que determinar a radiância em algum ponto da cena significa encontrar como a energia está chegando ao ponto. Uma forma de calcular isso é encontrar todos os caminhos

possíveis de troca de energia que passam por aquele ponto. Obviamente, o número de caminhos de trocas de energia que passam por um determinado ponto é infinito. Com métodos de Monte Carlo, no entanto, pode-se estimar como os infinitos caminhos contribuem para a radiância naquele ponto a partir de uma média de um número finito de caminhos, aleatoriamente amostrados, que passam por aquele ponto. O método é descrito de forma algorítmica no Algoritmo 2.4.

Algoritmo 2.4: Algoritmo de Metropolis light transport.

```
1 x = CaminhoInicial();
2 imagem = matriz de zeros;
3 para i = 0 até N faça
4 |   y = Mutar(x);
5 |   a = ProbabilidadeDeAceitação(x|y);
6 |   se Random() < a então
7 |   |   MarcarAmostra(imagem, x)
8 Retorna imagem
```

Metropolis Light Transport é bem mais eficiente do que *Bidirectional Path Tracing*, uma vez que, para se amostrar um novo caminho, basta fazer uma mutação em um caminho já existente, dessa forma o custo por amostragem é muito menor do que em um *Bidirectional Path Tracing* clássico. Ademais, *Metropolis Light Transport* usa estratégias de aceitação que concentram a distribuição das amostras em trechos mais brilhosos da cena. Isso faz com que caminhos com maior contribuição sejam mais facilmente amostrados, e com isso, consegue-se uma convergência mais rápida da imagem.

Há várias estratégias de mutação descritas por Veach e Guibas. A idéia por trás de algumas das estratégias é simples, já outras estratégias envolvem um arcabouço matemático e estatístico mais complicado e contribuem para aumentar o grau de sofisticação do algoritmo e melhorar seu desempenho em condições de iluminação consideradas difíceis, como iluminação indireta forte. Uma cena em que a iluminação de uma sala é fortemente influenciada por um pequeno buraco na parede é mais facilmente renderizada com MLT do que com alguns dos métodos descritos neste capítulo. Para cenas simples, todavia, MLT é um desperdício. As figuras 2.12 e 2.13 ilustram estratégias de mutação de caminhos.

Cenas que não apresentam coerência são uma armadilha para MLT, uma parede com várias pequenas passagens, por exemplo, pode fazer com que o algoritmo desperdice tempo demais em uma única passagem ao invés de fazer uma visita mais larga. Um outro inconveniente do algoritmo é a grande quantidade de parâmetros que influenciam pesadamente a variância da amostragem.

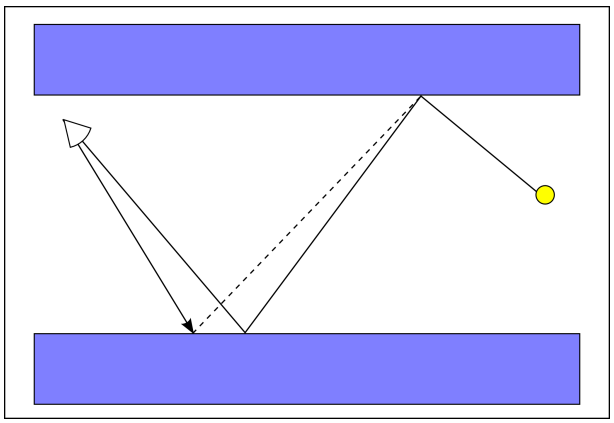


Figura 2.12: Mutação de lente.

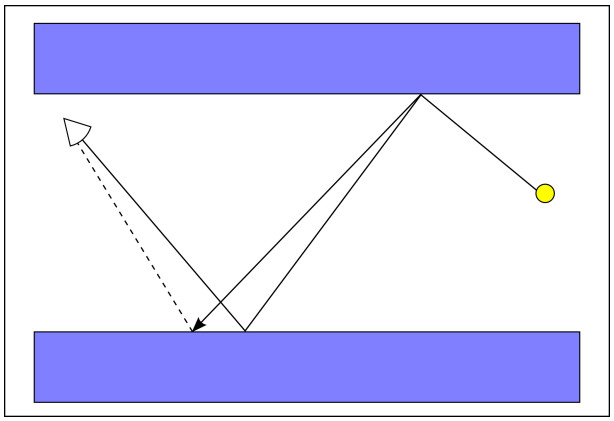


Figura 2.13: Mutação de cáustica.

2.5.2 Energy Redistribution Path Tracing

Energy Redistribution Path Tracing (ERPT) foi proposto por David Cline em 2005 [13]. O método, um melhoramento de MLT, consiste em usar estratégias de mutação mais cautelosas em MLT e combinar a fase final de renderização com a acoplagem de um filtro de ruído. De forma simplificada, ERPT pode ser visto como MLT combinado com uma fase de processamento de imagem a fim de se obter uma filtragem do ruído.

2.6 Considerações finais

Neste capítulo, foram apresentados e discutidos alguns métodos de renderização considerados importantes para o desenvolvimento do método proposto. Alguns aspectos mais relevantes a respeito de cada um dos métodos foram abordados.

No Capítulo 3, é descrito o método proposto neste trabalho, onde a inspiração advinda dos métodos descritos neste capítulo se tornará mais clara.

3 *Coherent Bidirectional Path Tracing*

3.1 Introdução

No Capítulo 2, foram descritos e comentados alguns métodos de renderização que estão mais intimamente relacionados ao método denominado *Coherent Bidirectional Path Tracing* (CBDPT), proposto neste trabalho e descrito em detalhes neste capítulo. O CBDPT foi concebido com o objetivo de tratar alguns dos problemas encontrados nos métodos apresentados no Capítulo 2.

CBDPT pertence à classe de métodos de renderização baseados em pixel oriundos de Ray Tracing. Portanto, é necessário descrever o lançamento e rastreamento de raios, a ordem em que os caminhos são traçados e como esses caminhos são combinados para estimar a iluminação de um determinado pixel da imagem a ser renderizada.

Os passos necessários para a renderização de uma imagem através do CBDPT são: traçado de caminho a partir da fonte de luz, traçamento de caminho a partir do observador (para cada pixel), combinação dos caminhos. Note que esses passos são bastante semelhantes ao que é feito em BDPT, a principal diferença está no fato de se manter o mesmo caminho traçado a partir da fonte de luz para a imagem inteira (ao invés de traçar um novo caminho a partir da fonte de luz para cada pixel). Essa estratégia permite que a imagem renderizada seja avaliada a partir de caminhos semelhantes para pixels próximos. Dessa forma, evita-se a principal fonte de ruído do BDPT. Algumas outras diferenças, em especial o traçado coerente de caminhos a partir do observador, também desempenha papel importante na eliminação do ruído. À medida que maiores detalhes sobre os passos do algoritmo são dados, essas diferenças, assim como seus efeitos, se tornam mais óbvios.

3.2 Visão geral do algoritmo

O algoritmo do CBDPT (Algoritmo 3.1), por ser fortemente inspirado no BDPT e no CPT, traz características de ambos os métodos. Basicamente, herdam-se as características de CPT que o

fazem ser livre de ruído em BDPT. Além disso, o caminho traçado a partir da fonte de luz é mantido durante a renderização de uma imagem inteira.

Algoritmo 3.1: Algoritmo de Coherent Bidirectional Path Tracing.

```
1  inicialize imagem com zeros;
2  para K Estimativas faça
3      inicialize sementes para traçados de caminhos;
4      trace caminho a partir da fonte de luz;
5      inicilize imagem temporária com zeros;
6      para cada pixel faça
7          trace caminho a partir do observador usando sementes dadas;
8          estime fluxo para o pixel combinando caminhos;
9          atualize imagem temporária;
10     atualize imagem usando imagem temporária;
```

3.3 Traçando caminho a partir do observador

O CBDPT é um método baseado em pixels e, consequentemente, é necessário traçar um caminho a partir do observador, passando por cada pixel. Isso é bastante semelhante ao que se faz em métodos mais conhecidos, como BDPT e PT. Uma diferença fundamental é que os caminhos a partir do observador são feitos de maneira coerente (assim como *Coherent Path Tracing* [11]). O traçado coerente de caminhos evita ruído na imagem renderizada, uma vez que cada um dos pixels será avaliado a partir de caminhos semelhantes.

No caso mais simples, um caminho com apenas dois vértices (um sendo o olho do observador e o outro sendo o ponto da cena mais próximo do observador na direção do pixel a ser estimado) é trivial. Por uma questão de facilitar a compreensão do método esse caso será descrito. A Figura 3.1 ilustra o problema.

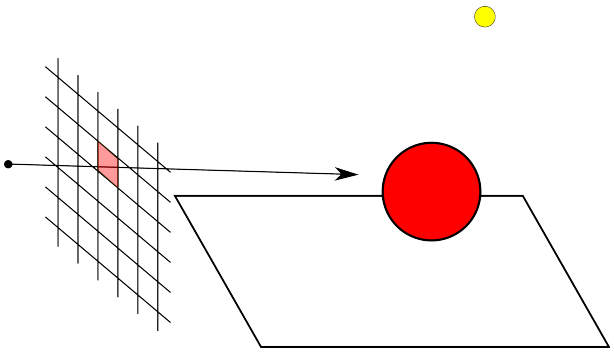


Figura 3.1: Traçando raio a partir do observador passando por um pixel.

Obviamente, o método proposto não é tão simples quanto um *Ray Casting* ingênuo. Pode-se aplicar uma série de fatores com valores aleatórios no traçado desse caminho a fim de obter um método estocástico e, por conseguinte, obter os tipos de efeitos tão desejados em métodos de *Ray Tracing* baseados em Monte-Carlo (MC).

De qualquer forma, a estratégia para se encontrar o primeiro vértice de um caminho com um *Ray Casting* ingênuo pode ser usada para o método proposto. Dado que se tem o primeiro vértice do caminho, encontrar-se os outros vértices se dá da mesma forma quando se traça o caminho a partir da fonte de luz.

3.4 Traçando caminho a partir da fonte luminosa

O traçado de caminhos a partir de uma fonte de luz é relativamente simples e, em grande parte, similar ao traçado de caminhos a partir do observador (vide Seção 2.2.2). Assim, nesta seção, apenas os aspectos peculiares deste traçado são discutidos.

Para traçar-se caminhos a partir da fonte de luz, primeiramente é necessário escolher-se uma fonte de luz. Quando a fonte luminosa é pontual, a posição do ponto de luz é a origem do primeiro vértice do caminho a ser traçado. Amostrar um caminho partindo uma fonte de luz pontual é uma simples questão de se amostrar uma direção, o que é equivalente a amostrar um ponto numa superfície de uma esfera (Vide Figura 3.2).

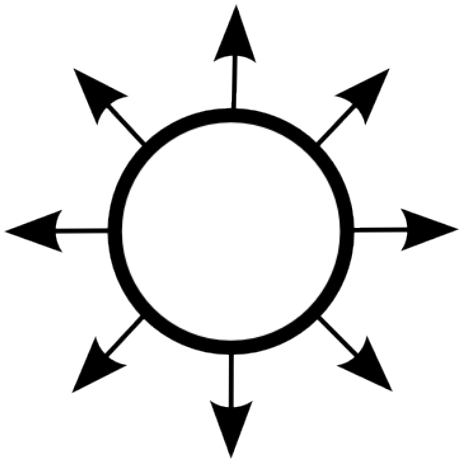


Figura 3.2: Amostrando uma fonte de luz pontual.

Quando se quer traçar um caminho a partir de uma fonte de luz contínua, é necessário primeiramente escolher um ponto dessa fonte luz. Note que, por conta de ser utilizado um método estocástico, esse ponto deve ser escolhido aleatoriamente. É importante que a distribuição de

pontos escolhidos na superfície de uma fonte de luz contínua seja homogênea. Isto significa que não deve haver, na fonte de luz, trechos com maior probabilidade de serem amostrados do que outros. A Figura 3.3 mostra uma superfície com amostragem homogênea e uma outra superfíce com pontos amostrados de forma não-homogênea.

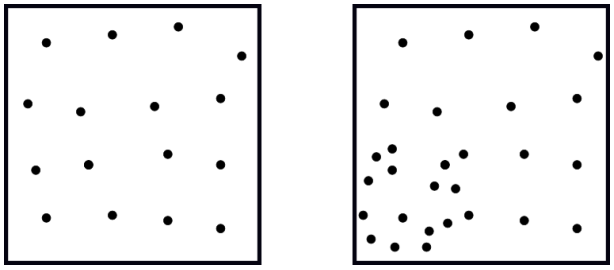


Figura 3.3: Amostragem homogêna e não-homogênea.

Após ter sido amostrado um ponto na superfície de uma fonte de luz contínua, deve-se, novamente de maneira aleatória e homogênea, amostrar uma direção que seja "compatível" com a fonte de luz no ponto amostrado. Diz-se que a direção amostrada é compatível com uma fonte de luz contínua em um determinado ponto se a normal à superfície naquele ponto estiver no mesmo sentido da direção amostrada. Analiticamente, seja $\overrightarrow{N}_s(\overrightarrow{P})$ a normal à superfície da fonte de luz contínua no ponto $\overrightarrow{P}$, diz-se que a direção (um vetor normalizado) $\overrightarrow{D}$ é compatível com a fonte de luz em questão no ponto $\overrightarrow{P}$ se $\overrightarrow{N}_s(\overrightarrow{P}) \cdot \overrightarrow{D} > 0$ (Figura 3.4).

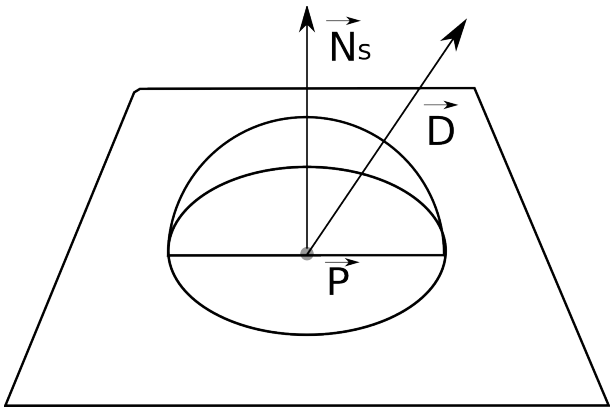


Figura 3.4: Direção compatível com normal no ponto dado.

Após a amostragem de um ponto da superfície de uma fonte de luz contínua e de uma direção compatível com a superfície da fonte de luz contínua no ponto amostrado, traça-se o raio que parte de $\overrightarrow{P}$ na direção $\overrightarrow{D}$ e encontram-se os objetos da cena que ele intercepta (Figura 3.5).

A partir de então, o processo de se escolher direções compatíveis é refeito recursivamente

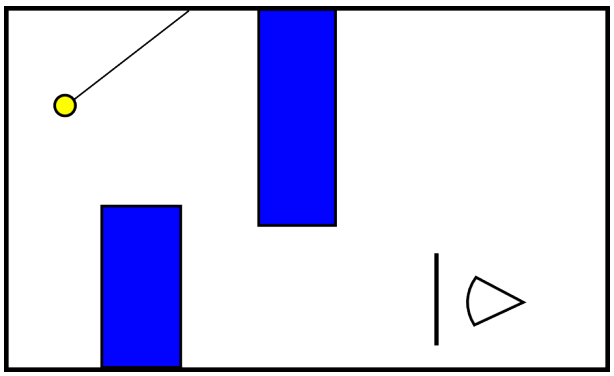


Figura 3.5: Caminho amostrado a partir da fonte de luz com 2 vértices.

para se traçar um caminho de M vértices. A Figura 3.6 ilustra um caminho com 3 vértices amostrado a partir de uma fonte de luz.

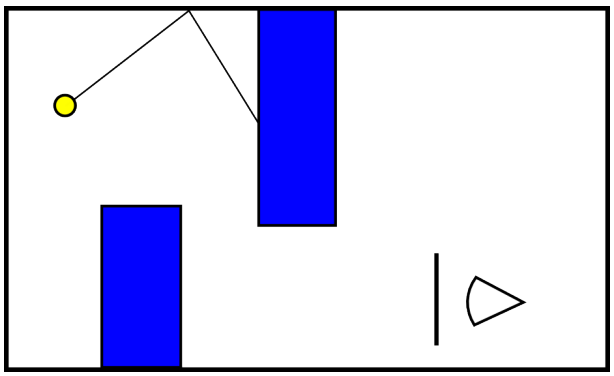


Figura 3.6: Caminho amostrado a partir da fonte de luz com 3 vértices.

Note que M é um parâmetro do algoritmo. O uso de critérios de parada, assim como a influência no resultado final do parâmetro M será discutido posteriormente.

3.5 Estimativa de fluxo em um pixel

A forma de se combinar caminhos é a mesma usada em BDPT e foi descrita no capítulo anterior.

O fluxo de um caminho é calculado de maneira recursiva como feito em PT. Ou seja, dado um caminho, o fluxo através desse caminho é computado como

$$\langle C \rangle = L_e(x, \mathbf{w}) + \frac{f_r(x, \mathbf{w}', \mathbf{w}) L_i(x, \mathbf{w}') \cos(\theta)}{2\pi r^2} \tag{3.1}$$

onde:

$\langle C \rangle$ é o fluxo através do caminho que está sendo estimado;

$\int_\Omega f_r(x, \mathbf{w}', \mathbf{w}) L_i(x, \mathbf{w}') d\mathbf{w}'$ é como a luz incidente no ponto x é retransmitida através da direção $\mathbf{w}$;

$L_e(x, \mathbf{w})$ é a luz naturalmente emitida pela superfície no ponto x na direção $\mathbf{w}$;

$f_r(x, \mathbf{w}', \mathbf{w})$ é a BRDF da superfície no ponto x;

$L_i(x, \mathbf{w}')$ é a luz incidente pela direção $\mathbf{w}'$ no ponto x;

$\cos(\theta)$ é o cosseno do ângulo entre a normal no ponto x e a direção $\mathbf{w}'$; e

r é a distância entre x e o ponto da cena mais próximo de x na direção $\mathbf{w}'$ (vide Figura 3.7).

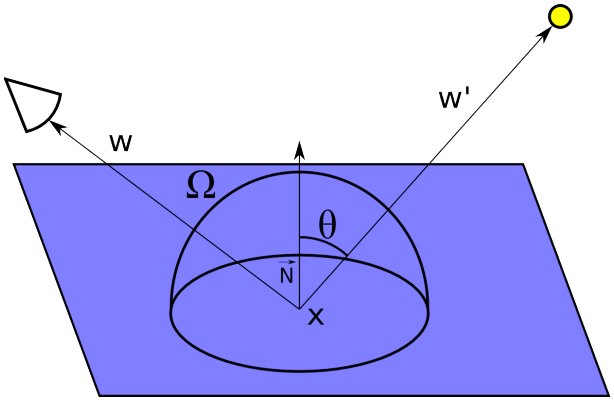


Figura 3.7: Estimando a equação de renderização.

O fator $L_i(x, \mathbf{w}')$ é calculado de maneira recursiva ao longo do caminho, partindo do observador até a fonte de luz e assumindo que $L_i = L_e$ quando o vértice em questão está no ponto final do caminho (fonte de luz).

A partir da combinação de dois caminhos – um caminho de N vértices com origem na fonte de luz e um caminho de M vértices com origem no observador – podem ser gerados $M \cdot N$ caminhos. A estimativa de fluxo para a combinação dos dois caminhos deve ser de acordo com a Equação 3.2.

$$\langle \Phi \rangle = \sum_{i=0}^{N_l - 1} \sum_{j=0}^{N_e - 1} \omega_{ij} \langle C_{ij} \rangle \tag{3.2}$$

Onde:

$\langle \Phi \rangle$ é a estimativa de fluxo,

N_l é o número de vértices do caminho traçado a partir da fonte de luz,

N_e é o número de vértices do caminho traçado a partir do observador, e

$\langle C_{ij} \rangle$ é a estimativa de fluxo do caminho formado pela combinação dos caminhos traçados a partir do observador e a partir da fonte de luz ligando o i-ésimo vértice do caminho traçado a partir da fonte de luz ao j-ésimo vértice do caminho traçado a partir do observador.

O fator $\langle C_{ij} \rangle$ é estimado da mesma forma que se faz a estimativa de fluxo em PT (L_r na Equação 2.4). O fator ω_{ij} é um peso atribuído ao caminho em questão e deve obedecer à Equação 3.3

$$\sum_{i=0}^{N} \omega_{i,N-i} = 1 \tag{3.3}$$

Assim como em [10], ω_{ij} é calculado como:

$$\omega_{ij} = \begin{cases} \prod_{k=0}^{j-2} W_{k+1} & \text{para } i = 1, \\ 0 & \text{para } j = 1, \text{ e} \\ (\prod_{k=0}^{j-2} W_{k+1})(1 - W_{j+1}) & \text{caso contrário.} \end{cases} \tag{3.4}$$

Onde W_k é porporcional à especularidade da superfície no k-ésimo vértice do caminho amostrado a partir do observador. Para superfícies muito especulares, aproxima-se de 1 e, em superfícies difusas, aproxima-se de 0.

3.6 Geração da imagem renderizada

Na Seção 3.5, foi descrito como estimar o fluxo através de um pixel da imagem a ser renderizada. O método proposto mantém, durante a renderização de uma imagem, o caminho traçado a partir da fonte de luz. Além de manter o caminho traçado a partir da fonte de luz, o método proposto também traça caminhos a partir do observador da mesma forma que CPT. Aplicando-se o Algoritmo 3.1 obtém-se uma renderização.

Há alguns incovenientes com a aplicação direta do Algoritmo 3.1 como a propensão a padrões ou a possibilidade de se obter uma estimativa muito longe da ideal. A Figura 3.8 mostra uma renderização feita usando-se apenas uma estimativa por pixel. O resultado é equivalente

ao que se obteria aplicando-se diretamente o Algoritmo 3.1.

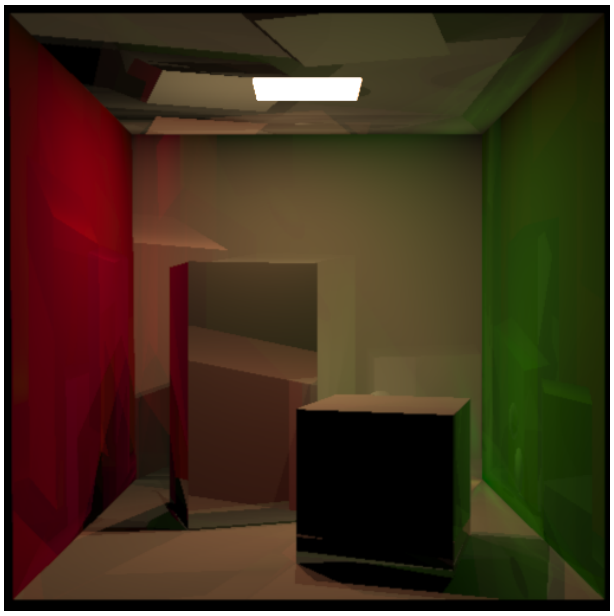


Figura 3.8: Renderização com 4 vértices a partir do observador e da fonte de luz. 1 estimativa por pixel.

Com o objetivo de sanar tais problemas, usa-se a mesma estratégia de geração da imagem renderizada que em PT, BDPT e CPT: várias imagens são renderizadas e uma imagem final é obtida a partir da média das imagens renderizadas até então. Com isso, estimativas distantes do ideal se tornam menos relevantes com a média das estimativas obtidas e acabam tendo um impacto menor na imagem final. A redução dos padrões também ocorre por conta de que, pela natureza estocástica do método, é improvável que padrões se manifestem exatamente da mesma forma em várias renderizações. O resultado dessa estratégia pode ser visto na Figura 3.9 que mostra uma renderização feita usando-se 25 estimativas por pixel.

3.7 Considerações finais

Neste capítulo, foi descrito o funcionamento do método proposto. Detalhes de como são traçados caminhos a partir da fonte de luz e a partir do observador foram dados. Foram feitas explicações de como se traçar caminhos a partir da fonte de luz e a partir do obsrevador e como combiná-los de forma a obter uma estimativa de fluxo para um pixel. A estratégia usada para gerar uma imagem renderizada também foi descrita.

Porém, não se fez uma avaliação objetiva do método e nem comparações com outros métodos já descritos. O Capítulo 4 trata de mostrar resultados obtidos com o método proposto e

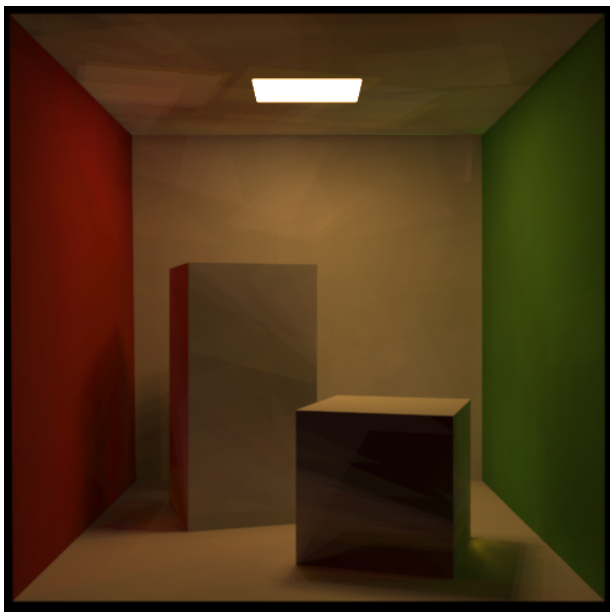


Figura 3.9: Renderização com 4 vértices a partir do observador e da fonte de luz. 25 estimativas por pixel.

métodos discutidos anteriormente. Testes com a *Cornell Box*, modelos clássicos e cenas customizadas são feitos com o objetivo de comparar as imagens obtidas e, medindo o tempo de renderização, o desempenho dos métodos.

4 Resultados e discussão

4.1 Introdução

Neste ponto, já se descreveu como funcionam os métodos de PT, BDPT, CPT e o método proposto: CBDPT. Embora alguns exemplos de resultados dos métodos em questão tenham sido dados, as imagens renderizadas até então serviram apenas para ilustrar os métodos.

Neste capítulo, são apresentados resultados obtidos com os métodos anteriormente discutidos. Vários testes são realizados e as imagens renderizadas são apresentadas. Comparações de desempenho dos métodos também são feitas.

4.2 Testes

Os testes estão agrupados em três categorias: renderizações da *Cornell Box*, testes com alta contagem de polígonos e testes em condições específicas de iluminação.

Na Seção 4.2.1 são apresentados os testes realizados com a Cornell Box. Com a finalidade de se ter um exemplo realmente clássico para comparação de métodos adotou-se, como exemplo a ser renderizado, uma Cornell Box. Foram testados os métodos de PT, BDPT, CPT e CBDPT com 5, 25 e 100 estimativas por pixel e caminhos de 8 vértices.

Nas Seções 4.2.2, 4.2.3 e 4.2.4 são apresentados os testes realizados com cenas em que há um grande número de polígonos. Com a finalidade de se ter mais exemplos clássicos, foram adotados alguns modelos comumente utilizados para comparação de métodos de renderização: Dragão, Buddha e Bunny. Foram feitos testes com os métodos PT, BDPT, CPT e CBDPT com 100 estimativas por pixel e caminhos de 4 vértices. Algumas imagens foram processadas com o objetivo de melhor ilustrar as diferenças entre os métodos. Foi tomado o cuidado de usar-se o mesmo processamento para as mesmas cenas com métodos diferentes.

Nas Seções 4.2.5, 4.2.6 e 4.2.7 são apresentados os testes realizados com condições específicas de iluminação. Com a finalidade de que se possa avaliar os métodos em condições

de iluminação não providas pelas outras renderizações, foram construídas algumas cenas para testá-los. As cenas customizadas utilizadas foram a de uma grade, uma gaiola e uma luminária. A cena da grade foi testada com caminhos de até 4 vértices para os métodos de BDPT, CBDPT e MLT e caminhos de até 8 vértices para BDPT e CBDPT. As cenas da gaiola e da luminária foram testada com caminhos de até 4 vértices para os métodos de BDPT e CBDPT. Nas cenas customizadas as renderizações foram feitas com 100 estimativas por pixel.

4.2.1 Cornell Box

A *Cornell Box* utilizada nos próximos exemplos usa propriedades muito próximas às da *Cornell Box* original. Mais detalhes a respeito do modelo da *Cornell Box* são dados no Apêndice A.

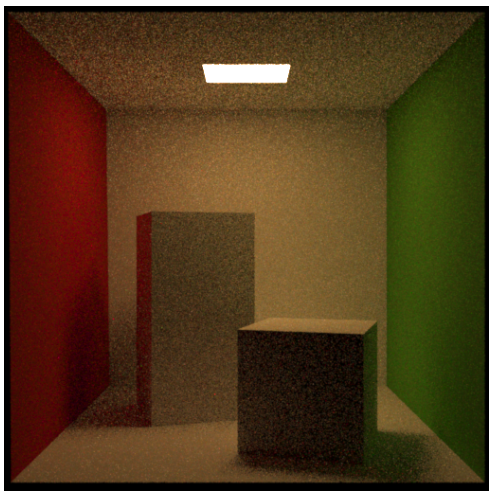
Cornell Box com 5 estimativas por pixel

Os resultados deste teste são apresentados nas figuras 4.1 e 4.2 e na Tabela 4.1. A Figura 4.1 ilustra os resultados da renderização da cena obtidos com 5 estimativas por pixel, pelos métodos de: PT (Figura 4.1a), BDPT (Figura 4.1b), CBDPT (Figura 4.1c) e CPT (Figura 4.1d). Para manter o número de vértices total idêntico em todos os métodos, foram utilizados 8 vértices no método de PT. A Figura 4.2 e a Tabela 4.1 registram os desempenhos dos quatro métodos.

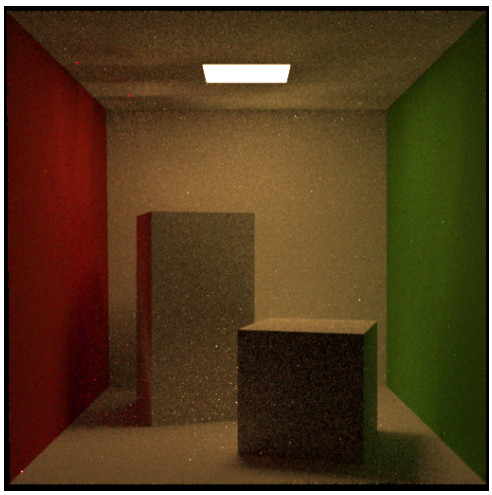
Tabela 4.1: Cornell Box com 5 estimativas por pixel.

Método	Tempo em segundos	N_e	N_l
PT	55	8	-
BDPT	96	4	4
CBDPT	72	4	4
CPT	48	8	-

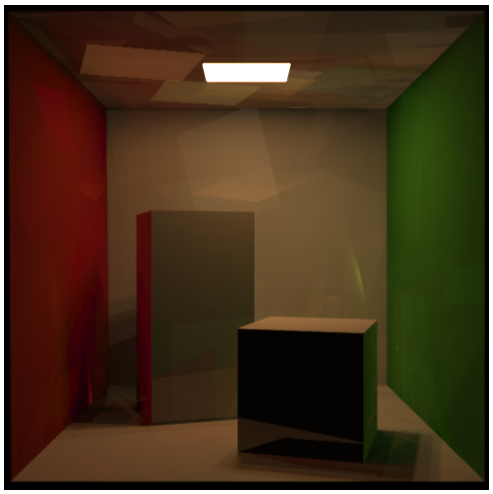
CPT é o método com o melhor desempenho neste teste, seguido dos métodos PT, CBDPT e BDPT. Note-se que a cena é simples e favorece métodos como PT e CPT. Ademais, é importante notar que, diferentemente de PT e BDPT, CBDPT gerou imagens sem ruído e os padrões produzidos são menos perceptíveis do que os padrões produzidos por CPT. A diferença em tempo de execução é justificável. Como se esperava, o método proposto apresenta padrões e é livre de ruído. Os testes subseqüentes permitem avaliar como os padrões se tornam menos perceptíveis aumentando-se o número de estimativas por pixel. Neste teste, o método proposto teve desempenho 33,33% pior que CPT (método com melhor desempenho) e 25% melhor que BDPT (método com pior desempenho).



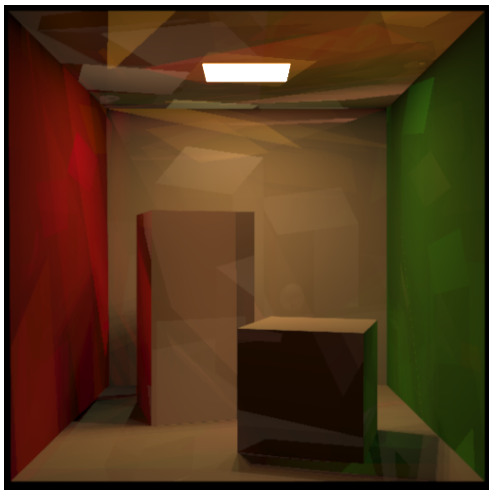
(a) PT, 8 vértices.



(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(c) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(d) CPT, 8 vértices.

Figura 4.1: Cornell Box com 5 estimativas por pixel.

Cornell Box com 25 estimativas por pixel

Os resultados deste teste são apresentados nas figuras 4.3 e 4.4 e na Tabela 4.2. A Figura 4.3 ilustra os resultados da renderização da cena obtidos com 25 estimativas por pixel, pelos métodos de: PT (Figura 4.3a), BDPT (Figura 4.3b), CBDPT (Figura 4.3c) e CPT (Figura 4.3d). Para manter o número de vértices total idêntico em todos os métodos, foram utilizados 8 vértices no método de PT. A Figura 4.4 e a Tabela 4.2 registram os desempenhos dos quatro métodos.

Novamente, CPT é o método com o melhor desempenho neste teste, seguido por PT. CBDPT tem desempenho mais próximo ao dos métodos PT e CPT do que BDPT, o que indica que, em termos de tempo de execução, CBDPT se coloca numa posição entre métodos

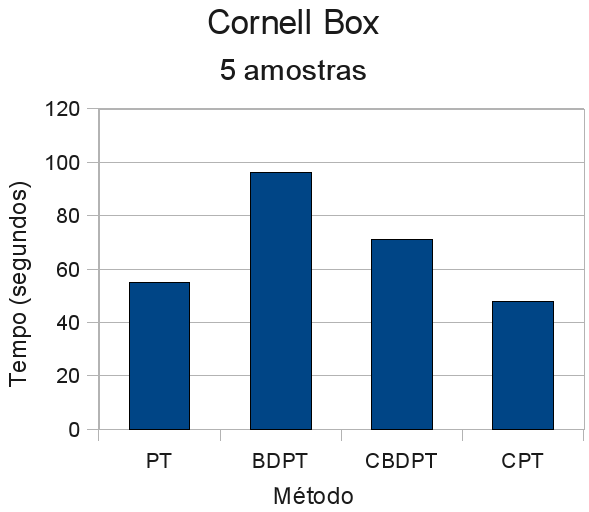


Figura 4.2: Comparação de tempo de execução para Cornell Box com 5 amostras.

Tabela 4.2: Cornell Box com 25 estimativas por pixel.

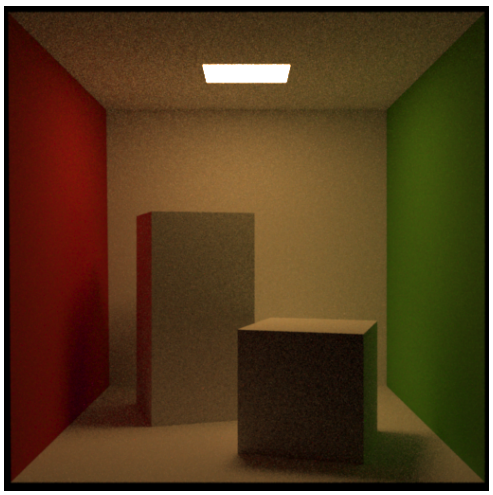
Método	Tempo em segundos	N_e	N_l
PT	245	8	-
BDPT	420	4	4
CBDPT	290	4	4
CPT	202	8	-

mais simples, porém, ainda assim, obtendo efeitos apenas obteníveis com métodos mais avançados como BDPT. As observações sobre tempo de execução e simplicidade da cena são as mesmas do teste anterior e são válidas para este teste também. Neste teste, o método proposto teve desempenho 30,34% pior que CPT (método com melhor desempenho) e 30,95% melhor que BDPT (método com pior desempenho).

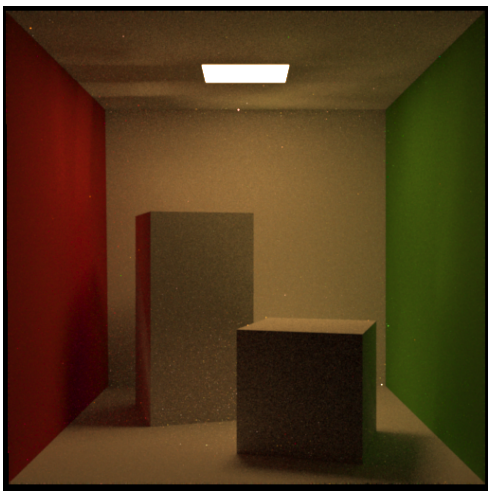
Cornell Box com 100 estimativas por pixel

Os resultados deste teste são apresentados nas figuras 4.5 e 4.6 e na Tabela 4.3. A Figura 4.5 ilustra os resultados da renderização da cena obtidos com 100 estimativas por pixel, pelos métodos de: PT (Figura 4.5a), BDPT (Figura 4.5b), CBDPT (Figura 4.5c) e CPT (Figura 4.5e). A Figura 4.5d é renderizada com a mesma configuração da 4.5c. Para manter o número de vértices total idêntico em todos os métodos, foram utilizados 8 vértices no método de PT. A Figura 4.6 e a Tabela 4.3 registram os desempenhos dos quatro métodos.

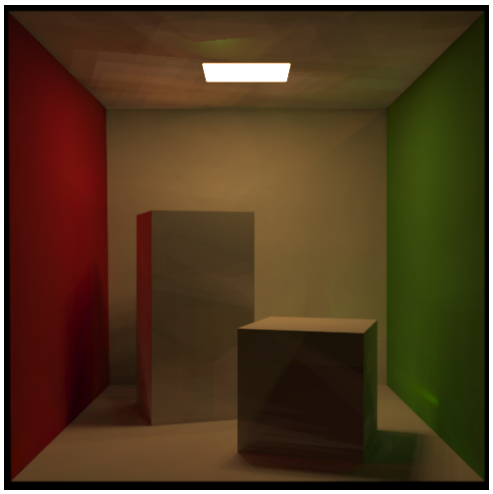
O que se nota neste teste é, basicamente, uma confirmação do que se esperava a partir dos testes anteriores. Uma novidade interessante deste último teste é o fato de CBDPT ser testado



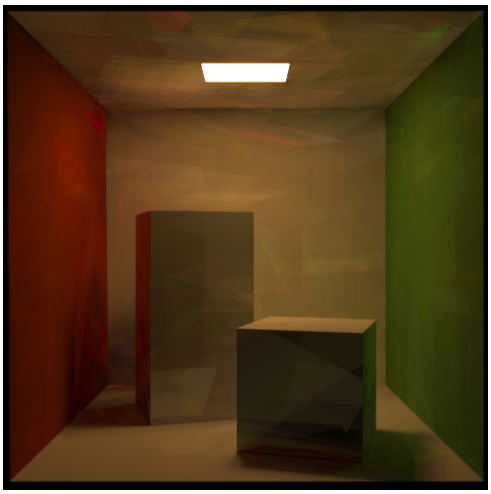
(a) PT, 8 vértices.



(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(c) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(d) CPT, 8 vértices.

Figura 4.3: Cornell box com 25 estimativas por pixel.

Tabela 4.3: Cornell Box com 100 estimativas por pixel.

Método	Tempo em segundos	N_e	N_l
PT	945	8	-
BDPT	1.622	4	4
CBDPT	1.100	4	4
CBDPT	1.098	4	4
CPT	812	8	-

duas vezes. A observação dos resultados mostram que a variação em tempo de execução é pequena, o que aumenta a confiabilidade dos resultados. Outra observação a ser feita é que os padrões gerados por duas execuções diferentes do método CBDPT são suficientemente diferentes para permitir (em trabalhos futuros) uma filtragem de erros a partir de várias execuções do

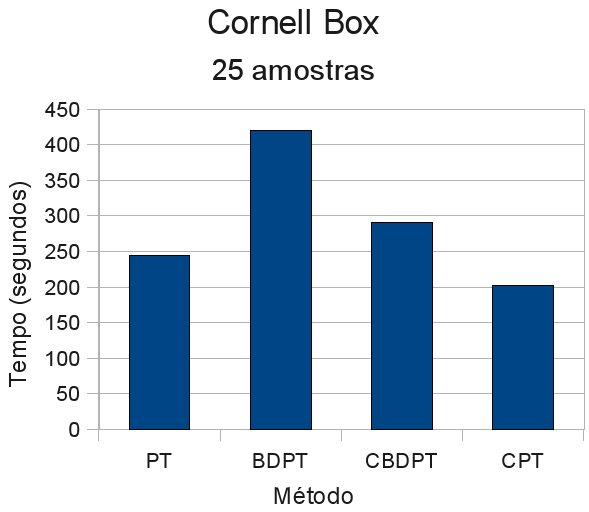
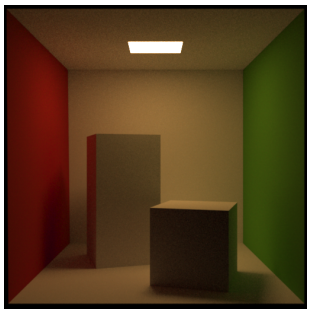
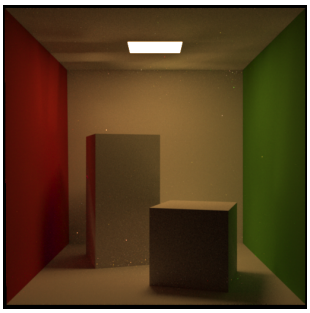


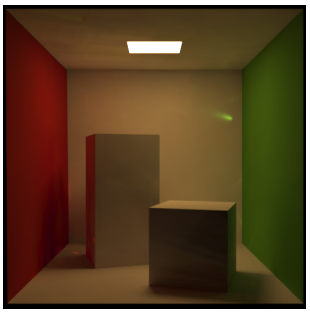
Figura 4.4: Comparação de tempo de execução para Cornell Box com 25 amostras.



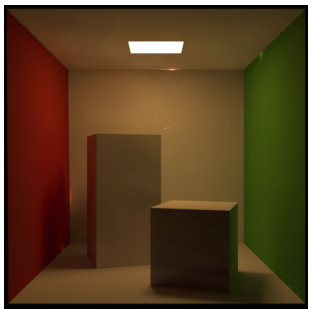
(a) PT, 8 vértices.



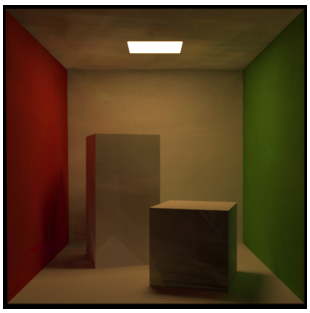
(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(c) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(d) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(e) CPT, 8 vértices.

Figura 4.5: Cornell box com 100 estimativas por pixel.

método. Por se tratar de um método estocástico, efeitos com menor probabilidade precisam de um tempo maior de execução para se manifestarem, um filtro de *outliers* pode ser suficiente para obter uma imagem com menos descontinuidades. Um compromisso entre tendenciosidade

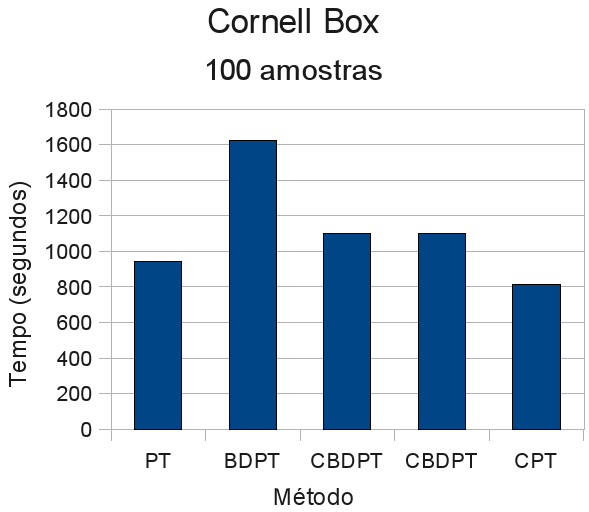


Figura 4.6: Comparação de tempo de execução para Cornellbox com 100 amostras.

do método e padrões menos perceptíveis pode (em trabalhos futuros) ser adotado.

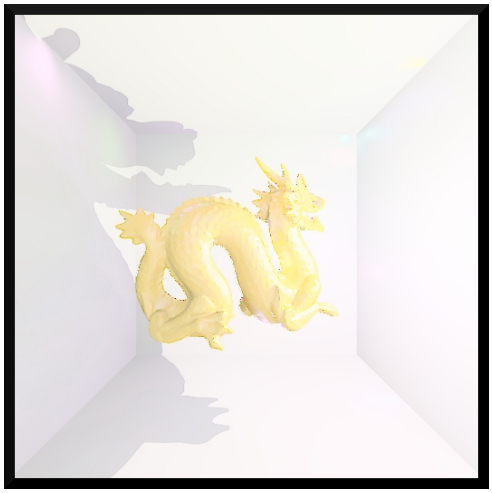
Como se esperava, a diferença absoluta entre BDPT e CBDPT aumenta com o número de amostras. Novamente, as observações a respeito do tempo de execução e simplicidade da cena são válidas para este teste. Neste teste, o método proposto teve desempenho 26,04% pior que CPT (método com melhor desempenho) e 32,30% melhor que BDPT (método com pior desempenho).

4.2.2 Dragão

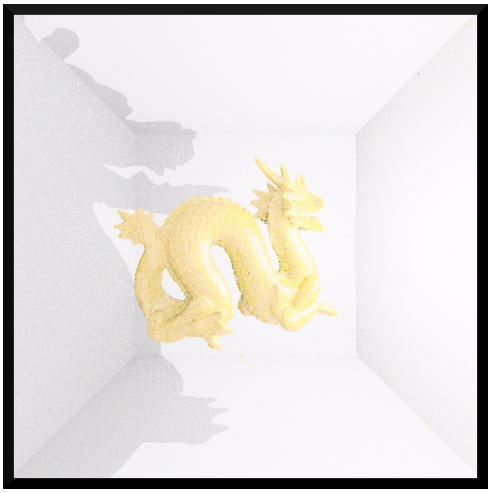
O modelo do dragão utilizado tem um total de 30.000 faces e 15.000 vértices. As propriedades da superfície são uma mistura de duas superfícies. Para a primeira superfície, o coeficiente de reflexão difusa é (em componentes vermelho, verde e azul) (0,68; 0,54; 0,18) e o coeficiente de reflexão especular da superfície é (0; 0; 0). Para a segunda superfície, o coeficiente de reflexão é (0,68; 0,54; 0,18). Em 70% das estimativas, usa-se as propriedades da primeira superfície; nos outros casos, usa-se as propriedades da segunda superfície. A luz é pontual, tem cor branca e coordenadas (0,190; 0; 0,239). A caixa envoltória do modelo tem centro em (0; 0; 0,239), rotações (em graus) no eixos (x, y, e z) são (-90; 0; 17,266), dimensões (ao longo dos eixos x, y, e z) são (0,292; 0,206; 0,130), um fator de escala de 0,146 é aplicado a cada um dos eixos. Detalhes sobre o processamento utilizado na imagem renderizada são dados no Apêndice A.

Os resultados deste teste são apresentados nas figuras 4.7 e 4.8 e na Tabela 4.4. A Figura 4.7 ilustra os resultados da renderização da cena obtidos com 100 estimativas por pixel, pelos métodos de: PT (Figura 4.7d), BDPT (Figura 4.7b), CBDPT (Figura 4.7a) e CPT (Figura 4.7c).

Para manter o número de vértices total idêntico em todos os métodos, foram utilizados 4 vértices no método de PT. A Figura 4.8 e a Tabela 4.4 registram os desempenhos dos quatro métodos.



(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.



(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

(c) CPT, 4 vértices.

(d) PT, 4 vértices.

Figura 4.7: Dragão com 100 estimativas por pixel.

Tabela 4.4: Dragão com 100 estimativas por pixel.

Método	Tempo em segundos	N_e	N_l
CBDPT	462	2	2
BDPT	638	2	2
CPT	563	4	-
PT	744	4	-

Cenas com uma maior contagem de polígonos e condições de iluminação mais específicas são o ponto forte de CBDPT. Neste tipo de teste CBDPT tem um desempenho consideravelmente melhor do que BDPT e PT, além de ficar melhor posicionado do que CPT. Este é o tipo

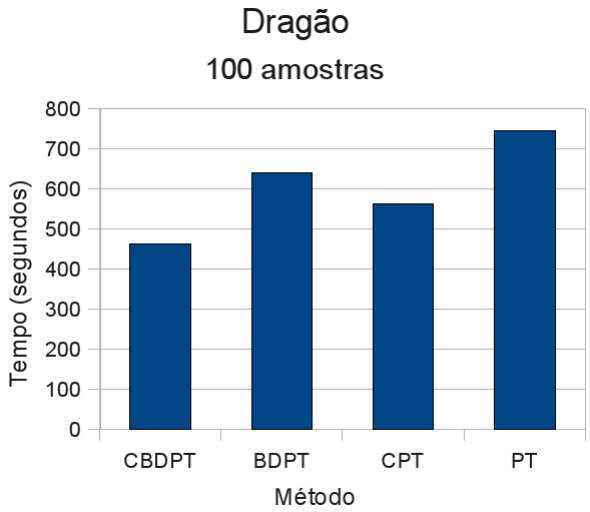


Figura 4.8: Comparação de tempo de execução para Dragão com 100 amostras.

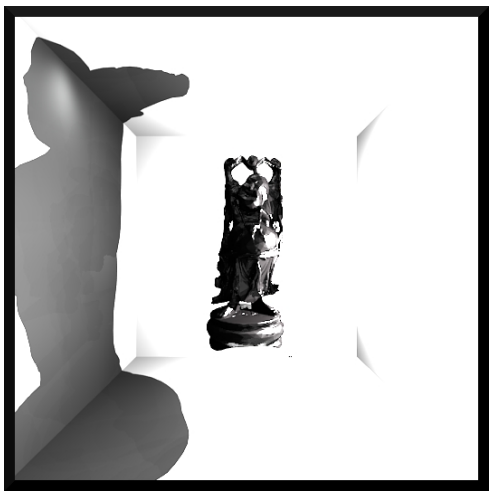
de teste em que CBDPT é superior aos métodos comparados. Diferentemente de PT e BDPT, CBDPT é livre de ruído e gera uma imagem bem mais agradável. Assim como CPT, CBDPT também gera padrões. Porém, o rastreamento bidirecional utilizado é suficiente para tornar os padrões gerados por CBDPT mais visualmente agradáveis. Neste teste, o método proposto teve desempenho 17,93% melhor que CPT (método com melhor desempenho).

4.2.3 Buddha

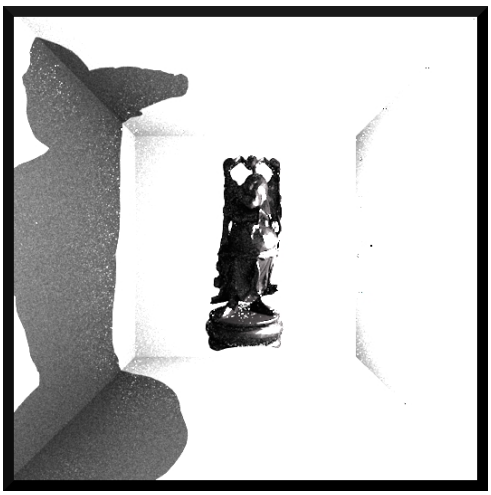
O modelo do *Buddha* utilizado tem um total de 29.999 faces e 14.990 vértices. O coeficiente de reflexão difusa da superfície é (em componentes vermelho, verde e azul) (0,17; 0,06; 0,09). O coeficiente de reflexão especular da superfície é (0,67; 0,67; 0,67). A luz é pontual, tem cor branca e coordenadas (0,150; 0; 0,253). A caixa envoltória do modelo tem centro em (-0,004; 0,016; 0,254), rotações (em graus) no eixos (x, y, e z) são (270; 0; 0), dimensões (ao longo dos eixos x, y, e z) são (0,125; 0,303; 0,254), um fator de escala de 0,151 é aplicado a cada um dos eixos. Detalhes sobre o processamento utilizado na imagem renderizada são dados no Apêndice A.

Os resultados deste teste são apresentados nas figuras 4.9 e 4.10 e na Tabela 4.5. A Figura 4.9 ilustra os resultados da renderização da cena obtidos com 100 estimativas por pixel, pelos métodos de: PT (Figura 4.9d), BDPT (Figura 4.9b), CBDPT (Figura 4.9a) e CPT (Figura 4.9c). Para manter o número de vértices total idêntico em todos os métodos, foram utilizados 4 vértices no método de PT. A Figura 4.10 e a Tabela 4.5 registram os desempenhos dos quatro métodos.

Este teste tem o resultado exatamente dentro do esperado e confirma os resultados obtidos

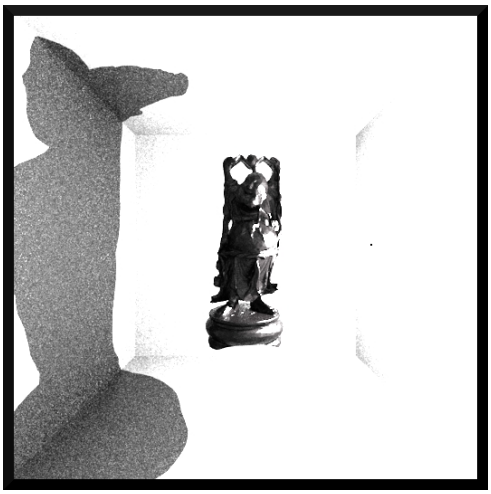


(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.



(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

(c) CPT, 4 vértices.



(d) PT, 4 vértices.

Figura 4.9: Buddha com 100 estimativas por pixel.

Tabela 4.5: Buddha com 100 estimativas por pixel.

Método	Tempo em segundos	N_e	N_l
CBDPT	450	2	2
BDPT	624	2	2
CPT	545	4	-
PT	708	4	-

com o teste anterior. Neste teste, o método proposto teve desempenho 17,43% melhor que CPT (método com melhor desempenho).

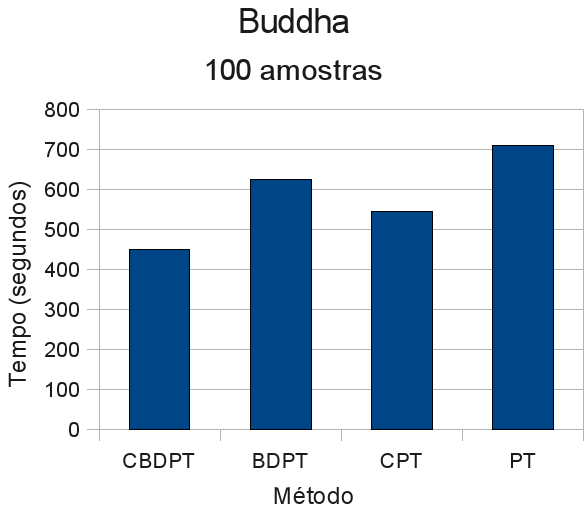


Figura 4.10: Comparação de tempo de execução para Buddha com 100 amostras.

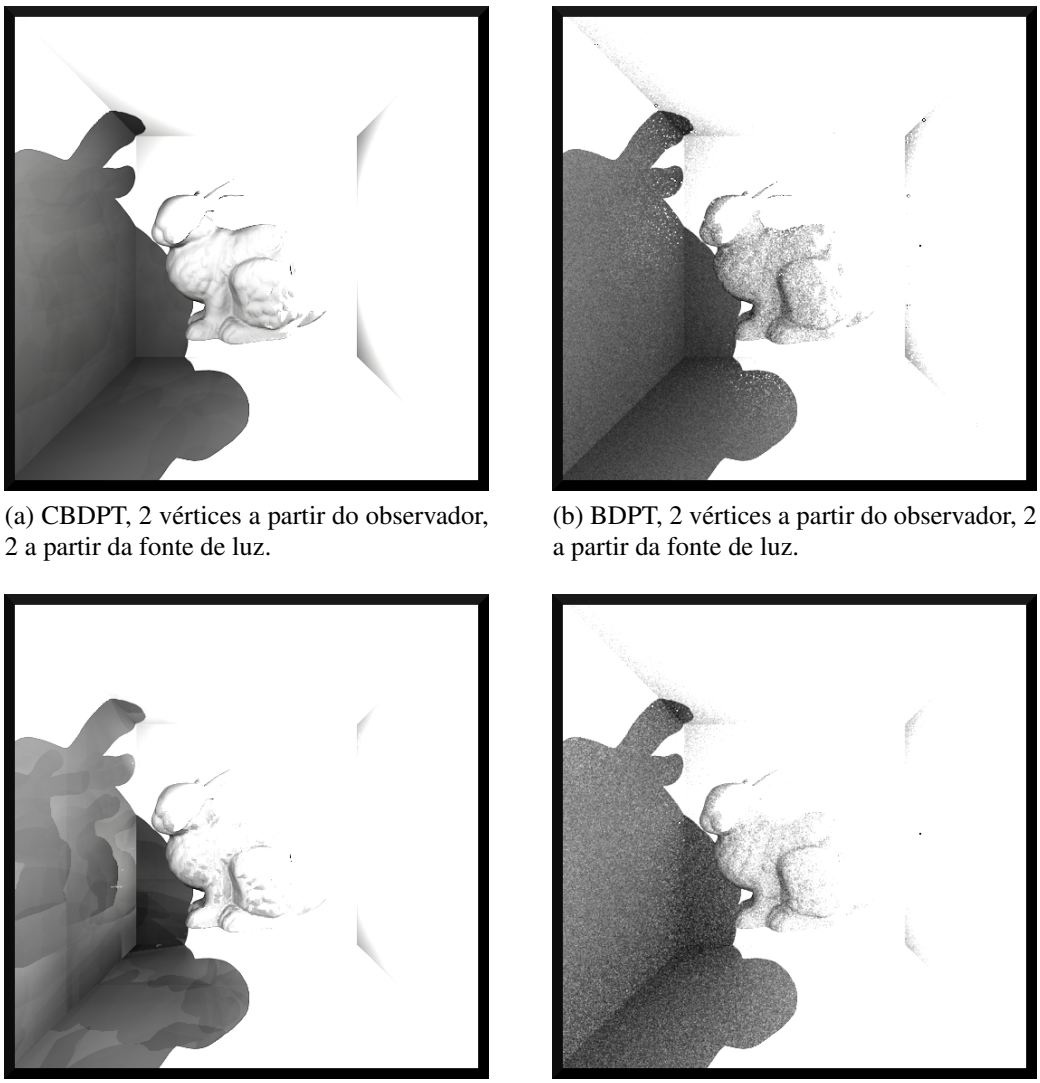
4.2.4 Bunny

O modelo do coelho utilizado tem um total de 34.832 faces e 17.418 vértices. O coeficiente de reflexão difusa da superfície é (em componentes vermelho, verde e azul) (0,89; 0,89; 0,89). O coeficiente de reflexão especular da superfície é (0,12; 0,12; 0,12). A luz é pontual, tem cor branca e coordenadas (0,164; 0; 0,239). A caixa envoltória do modelo tem centro em (0; 0,036; 0,254), dimensões (ao longo dos eixos x, y, e z) são (0,239; 0,186; 0,237), um fator de escala de 0,120 é aplicado a cada um dos eixos. Detalhes sobre o processamento utilizado na imagem renderizada são dados no Apêndice A.

Os resultados deste teste são apresentados nas figuras 4.11 e 4.12 e na Tabela 4.6. A Figura 4.11 ilustra os resultados da renderização da cena obtidos com 100 estimativas por pixel, pelos métodos de: PT (Figura 4.11d), BDPT (Figura 4.11b), CBDPT (Figura 4.11a) e CPT (Figura 4.11c). Para manter o número de vértices total idêntico em todos os métodos, foram utilizados 4 vértices no método de PT. A Figura 4.12 e a Tabela 4.6 registram os desempenhos dos quatro métodos.

Tabela 4.6: Bunny com 100 estimativas por pixel.

Método	Tempo em segundos	N_e	N_l
CBDPT	449	2	2
BDPT	625	2	2
CPT	550	4	-
PT	721	4	-



(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

(c) CPT, 4 vértices.

(d) PT, 4 vértices.

Figura 4.11: Bunny com 100 estimativas por pixel.

Novamente, devido às semelhanças entre as cenas e condições de iluminação, os resultados são semelhantes, confirmando a superioridade de CBDPT sobre os outros métodos testados para este tipo de cena. Neste teste, o método proposto teve desempenho 18,36% melhor que CPT (método com melhor desempenho).

4.2.5 Grade

O modelo da grade utilizado tem um total de 228 faces e 100 vértices. O coeficiente de reflexão difusa da superfície é (em componentes vermelho, verde e azul) (0,5; 0,5; 0,5). O coeficiente de reflexão especular da superfície é (0; 0; 0). A luz é a fonte de luz presente na *Cornell Box* utilizada. Mais detalhes do modelo são dados no Apêndice A. A caixa envoltória do modelo tem

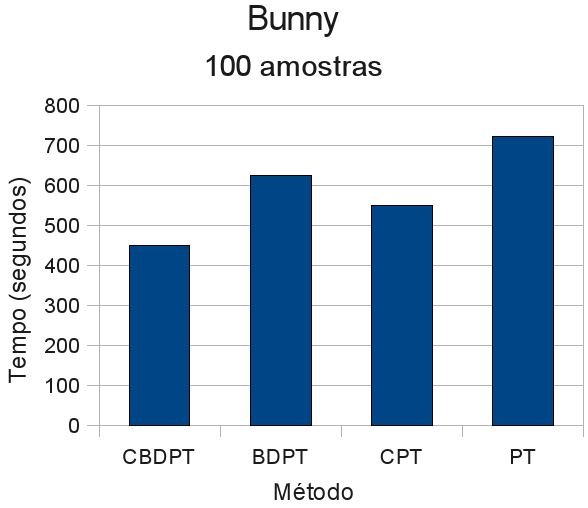


Figura 4.12: Comparação de tempo de execução para Bunny com 100 amostras.

centro em (0.090; 0; 0,226), rotações (em graus) no eixos (x, y, e z) são (90; -90; 0), dimensões (ao longo dos eixos x, y, e z) são (0,114; 0,474; 0,114), um fator de escala de 0,057 é aplicado a cada um dos eixos.

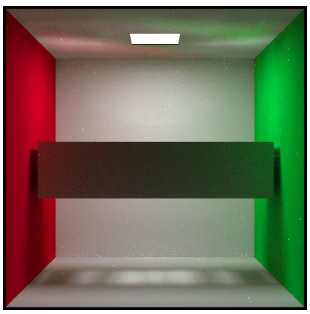
Os resultados deste teste são apresentados nas figuras 4.13 e 4.14 e na Tabela 4.7. A Figura 4.13 ilustra os resultados da renderização da cena obtidos com 100 estimativas por pixel, pelos métodos de: CBDPT (figuras 4.13a e 4.13c), BDPT (figuras 4.13b e 4.13d), e MLT (Figura 4.13e). A Figura 4.14 e a Tabela 4.7 registram os desempenhos dos três métodos.

Tabela 4.7: Grade com 100 estimativas por pixel.

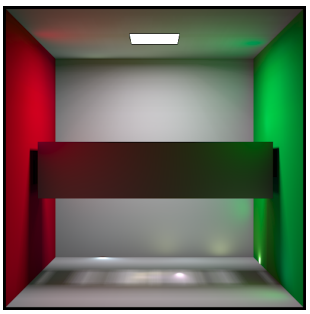
Método	Tempo em segundos	N_e	N_l
CBDPT	1.006	4	4
BDPT	1.342	4	4
CBDPT	507	2	2
BDPT	716	2	2
MLT	674	2	2

Para o tipo de cena em que a sombra produz padrões, o nosso método é superior em desempenho a BDPT tanto em caminhos traçados com 4 vértices a partir do observador e 4 vértices a partir da fonte de luz, quanto para caminhos traçados com 2 vértices a partir do observador e 2 vértices a partir da fonte de luz. Diferentemente de BDPT, CBDPT não gera ruído e uma imagem com continuidade bastante suave é produzida. Um experimento com MLT foi feito para este teste. O desempenho de MLT foi muito semelhante ao de CBDPT e a imagem gerada não é livre de ruído. Neste teste, o método proposto teve desempenho 25,03% melhor que BDPT

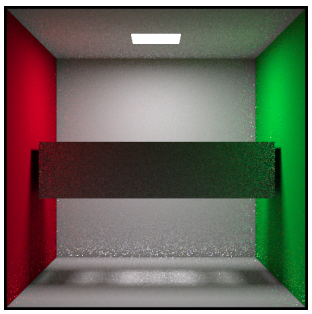
(a) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



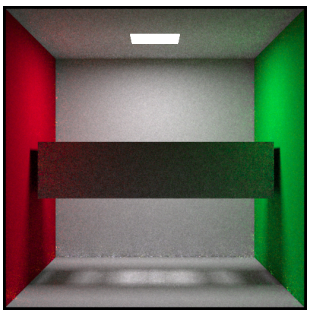
(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz.



(c) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.



(d) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.



(e) MLT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

Figura 4.13: Grade com 100 estimativas por pixel.

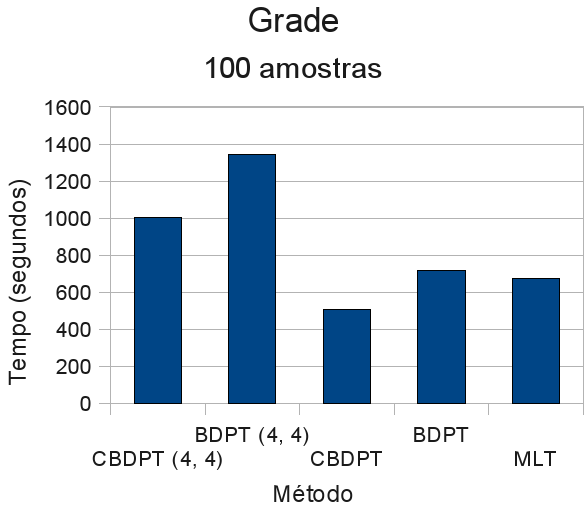


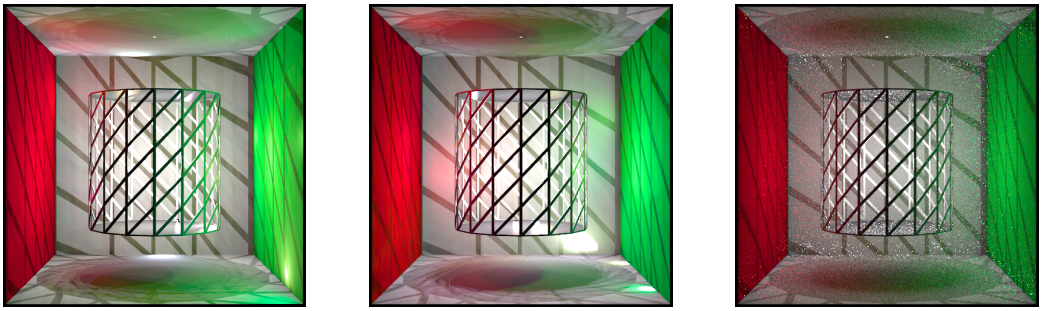
Figura 4.14: Comparação de tempo de execução para Grade com 100 amostras.

usando caminhos com 2 vértices a partir do observador e 2 vértices a partir da fonte de luz, e 29,18% melhor que BDPT usando caminhos com 2 vértices a partir do observador.

4.2.6 Gaiola

O modelo da gaiola utilizado tem um total de 2.168 faces e 896 vértices. O coeficiente de reflexão difusa da superfície é (em componentes vermelho, verde e azul) (0,5; 0,5, 0,5). O coeficiente de reflexão especular da superfície é (0; 0; 0). A luz é branca e tem coordenadas (0; 0; 0,237). Mais detalhes do modelo são dados no Apêndice A. A caixa envoltória do modelo tem centro em (0; 0; 0,237), dimensões (ao longo dos eixos x, y, e z) são (0,279; 0,279; 0,279), um fator de escala de 0,139 é aplicado a cada um dos eixos.

Os resultados deste teste são apresentados nas figuras 4.15 e 4.16 e na Tabela 4.8. A Figura 4.15 ilustra os resultados da renderização da cena obtidos com 100 estimativas por pixel, pelos métodos de: CBDPT (figuras 4.15a e 4.15b) e BDPT (Figura 4.15c). A Figura 4.16 e a Tabela 4.8 registram os desempenhos dos dois métodos.



(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

(b) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

(c) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

Figura 4.15: Gaiola com 100 estimativas por pixel.

Tabela 4.8: Gaiola com 100 estimativas por pixel.

Método	Tempo em segundos	N_e	N_l
CBDPT	755	2	2
CBDPT	750	2	2
BDPT	1.012	2	2

Os dois testes feitos confirmam a pouca variabilidade do tempo de execução de CBDPT e o melhor desempenho comparado a BDPT. Novamente os padrões gerados em duas execuções diferentes de CBDPT são diferentes o suficiente para levantar a ideia (em trabalhos futuros) de um filtro de *outliers*. Neste teste, o método proposto teve desempenho 25,88% melhor que BDPT.

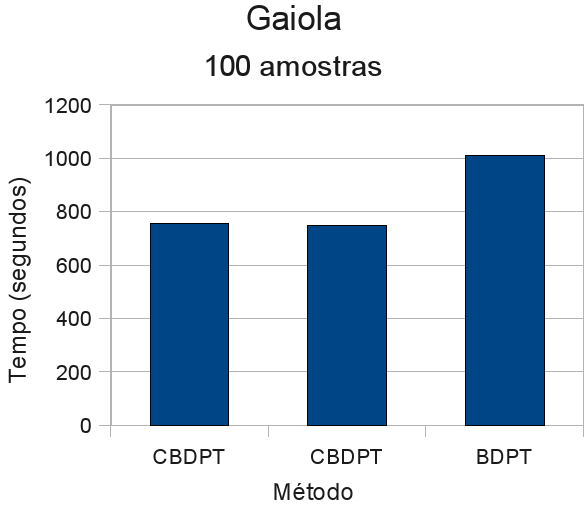


Figura 4.16: Comparação de tempo de execução para Gaiola com 100 amostras.

4.2.7 Luminária

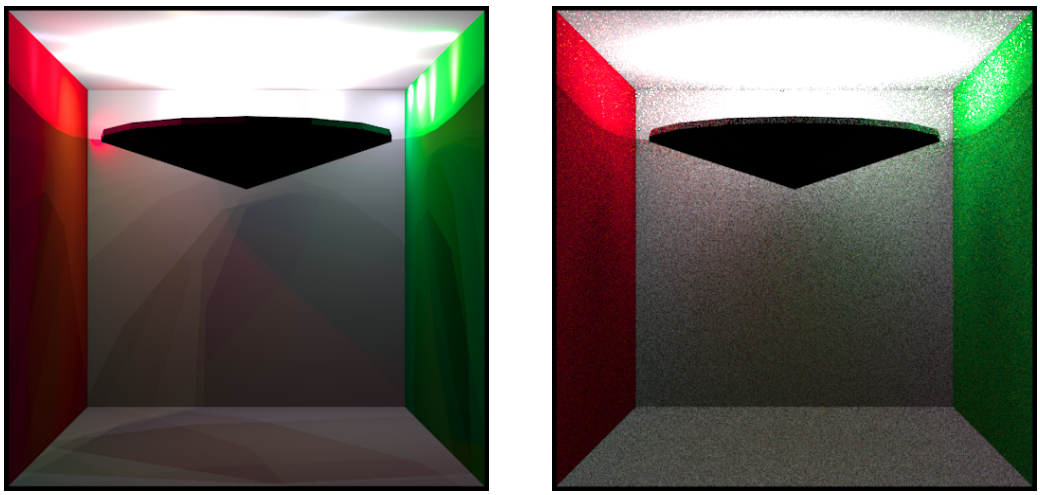
O modelo da luminária utilizado tem um total de 188 faces e 98 vértices. O coeficiente de reflexão difusa da superfície é (em componentes vermelho, verde e azul) (0,5; 0,5; 0,5). O coeficiente de reflexão especular da superfície é (0; 0; 0). A luz pontual é branca e tem coordenadas (0; 0; 0.248). Mais detalhes do modelo são dados no Apêndice A. A caixa envoltória do modelo tem centro em (0; 0; 0,248), dimensões (ao longo dos eixos x, y, e z) são (0,383; 0,383; 0,076), um fator de escala de 0,192 é aplicado a cada um dos eixos.

Os resultados deste teste são apresentados nas figuras 4.17 e 4.18 e na Tabela 4.9. A Figura 4.17 ilustra os resultados da renderização da cena obtidos com 100 estimativas por pixel, pelos métodos de: CBDPT (Figuras 4.17a) e BDPT (Figura 4.17b). A Figura 4.17 e a Tabela 4.9 registram os desempenhos dos dois métodos.

Tabela 4.9: Luminária com 100 estimativas por pixel.

Método	Tempo em segundos	N_e	N_l
CBDPT	664	2	2
BDPT	775	2	2

Novamente, CBDPT produz uma imagem sem ruído em um tempo de execução inferior ao de BDPT. Neste teste, o método proposto teve desempenho 14,32% melhor que BDPT.



(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz.

Figura 4.17: Luminária com 100 estimativas por pixel

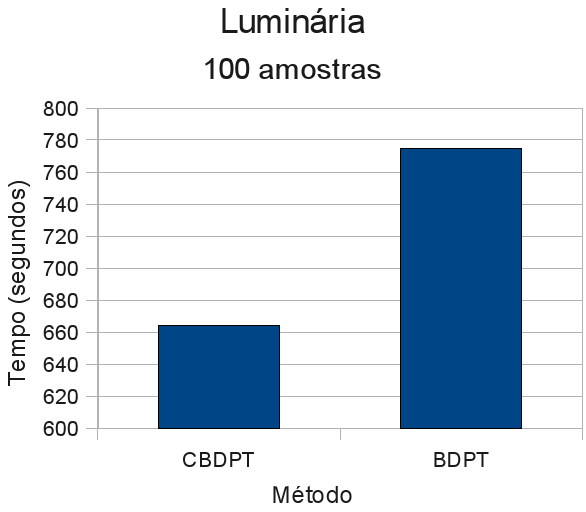


Figura 4.18: Comparação de tempo de execução para Luminária com 100 amostras

4.3 Discussões

Os resultados mostraram que CBDPT é livre de ruído e que os padrões que ele gera são menos perceptíveis do que os produzidos em renderizações por CPT. O desempenho de CBDPT em cenas com muitos polígonos é superior ao de PT e, em todos os testes, seu desempenho foi superior ao de BDPT. Os padrões gerados em duas execuções diferentes de CBDPT pode permitir (em trabalhos futuros) uma redução perceptível dos padrões.

4.4 Considerações finais

Agora que se tem resultados tanto do método proposto quanto dos métodos usados como parâmetros de comparação, segue-se, no Capítulo 5, as conclusões tomadas a partir dos resultados obtidos.

5 *Conclusão*

5.1 Principais contribuições

Um novo método de renderização foi desenvolvido. Esse método é bastante geral, não exige tesselagem da cena, é fisicamente baseado, livre de ruído e tem melhor desempenho do que os métodos que serviram de inspiração para o seu desenvolvimento.

Diferentemente de métodos menos gerais, não é necessário ter qualquer assertiva a respeito da cena a ser renderizada. O método desenvolvido serve para cenas abertas, fechadas e com geometria arbitrariamente complexa, diferentemente de métodos que exigem processamento ou tesselagem prévia da cena a ser renderizada. O método desenvolvido não apresenta ruído nas imagens renderizadas. O desempenho do método desenvolvido é superior ao desempenho dos métodos comparados nos testes realizados.

Como era um dos objetivos, a vantagem do método proposto sobre outros métodos se dá principalmente na ausência de ruído. Embora o método de CPT também seja livre de ruído, o método proposto obteve melhor desempenho em tempo, foram obtidas estimativas de iluminação superior usando o método proposto (uma vez que traçamos o caminho bidirecionalmente) e os padrões obtidos com o CBDPT são menos perceptíveis do que o que se foi obtido com CPT.

5.2 Trabalhos futuros

Um método de processamento de imagem pode ser desenvolvido para tratar os padrões gerados. Já que os padrões e artefatos do método proposto possuem características muito variáveis em renderizações diferentes (uma vez que o método é estocástico) pode-se investigar a possibilidade de se combinar algumas imagens com um método de processamento de imagem a fim de livrar a renderização de padrões e artefatos. Uma possibilidade é usar uma imagem com ruído como guia para a combinação de algumas outras imagens com padrões: a imagem guia serviria para determinar os limites dos padrões de forma que se possa tratá-los a partir das outras

imagens a serem combinadas.

Os testes realizados para comparar o método proposto com MLT mostraram que há vantagem sobre MLT em cenas onde a iluminação advém de caminhos simples (poucas quicadas e caminhos com alta probabilidade). Porém, os métodos precisam ser comparados em situações de iluminação mais complexas onde, espera-se, MLT tenha melhor desempenho. Uma boa possibilidade pode ser também estender (ou combinar) o CBDPT com a técnica de MLT.

Também é necessário testar o método proposto em cenas com tempo de renderização maior, assim pode-se ter uma comparação mais detalhada com relação a convergência dos métodos testados. Cenas em que há refração com separação de cores também são uma boa oportunidade para o teste, dada a natureza de padrões do resultado esperado.

O uso do fator de esforço deve ser melhor investigado.

Métodos com caminhos coerentes têm a propriedade de permitir a reutilização cálculos feitos anteriormente [11]. Essa característica pode ser melhor investigada e o uso de GPU ou instruções de sistemas massivamente paralelos pode ser testada.

Em algumas cenas, ou em alguns trechos de cenas, métodos com ruído podem ser considerados superior ao método proposto. Um possibilidade de investigação é desenvolver uma maneira que combine imagens geradas por métodos propensos a ruídos e métodos propensos a padrões a fim de se detectar automaticamente regiões das imagens em que um método é superior a outro. Uma vez que essas regiões sejam determinadas, pode-se fazer uma combinação inteligente das imagens a fim de manter as características desejáveis de diferentes métodos.

APÊNDICE A – Sobre os testes

A.1 Descrição dos testes

Para tornar mais viável a reprodução dos resultados obtidos, dá-se uma descrição mais técnica dos modelos e processamento de imagens utilizados.

A.1.1 Cornell Box

Descreve-se aqui a posição das fontes de luz, vértices dos objetos e são detalhados materiais da cena *Cornell Box*.

Fontes de luz

Há uma fonte luz esférica na posição $(0,200; 0,100; 0,400)$ com raio $0,025$. O material emissivo dessa fonte de luz para alguns diferentes comprimentos de onda é dado na tabela A.1.

Tabela A.1: Material "luz" na cena Cornell Box

Comprimento de onda (*nm*)	Emissividade
400	0,0
500	8,0
600	15,6
700	18,4

A fonte de luz no teto da cena tem as coordenadas de seus vértices como dado na tabela A.2 e usa o mesmo material da outra fonte de luz. A potência é omitida na descrição da fonte de luz, já que o renderizador utilizado gera uma imagem HDR, no formato OpenEXR, que permite a variação desses parâmetros mesmo após a renderização.

Tabela A.2: Vértices da fonte de luz no teto na cena Cornell Box

X	*Y*	*Z*
0,343	0,548	0,227
0,343	0,548	0,332
0,213	0,548	0,332
0,213	0,548	0,227

Objetos

O teto da *Cornell Box* tem vértices como descrito na tabela A.3 e material como descrito na tabelas A.18 e A.19.

Tabela A.3: Vértices do teto na cena Cornell Box

X	*Y*	*Z*
0,556	0,5488	0,0
0,556	0,5488	0,5592
0,0	0,5488	0,5592
0,0	0,5488	0,0

O piso da *Cornell Box* tem vértices como descrito na tabela A.4 e material como descrito na tabelas A.18 e A.19.

Tabela A.4: Vértices do piso na cena Cornell Box

X	*Y*	*Z*
0,5528	0,0	0,0
0,0	0,0	0,0
0,0	0,0	0,5592
0,5496	0,0	0,5592

A parede do fundo da *Cornell Box* tem vértices como descrito na tabela A.5 e material como descrito na tabelas A.18 e A.19.

Tabela A.5: Vértices da parede do fundo na cena Cornell Box

X	*Y*	*Z*
0,5496	0,0	0,5592
0,0	0,0	0,5592
0,0	0,5488	0,5592
0,5560	0,5488	0,5592

A parede da direita da *Cornell Box* tem vértices como descrito na tabela A.6 e material como descrito na tabelas A.20 e A.21.

Tabela A.6: Vértices da parede da direita na cena Cornell Box

X	*Y*	*Z*
0,0	0,0	0,5592
0,0	0,0	0,0
0,0	0,5488	0,0
0,0	0,5488	0,5592

A parede da esquerda da *Cornell Box* tem vértices como descrito na tabela A.7 e material como descrito na tabelas A.22 e A.23.

Tabela A.7: Vértices da parede da esquerda na cena Cornell Box

X	*Y*	*Z*
0,5528	0,0	0,0
0,5496	0,0	0,5592
0,5560	0,5488	0,5592
0,5560	0,5488	0,0

O bloco curto da cena *Cornell Box* tem faces descritas pelos vértices dados nas tabelas A.8, A.9, A.10, A.11 e A.12 e usa o material descrito na tabelas A.18 e A.19.

Tabela A.8: Vértices da face 1 do bloco curto na cena Cornell Box

X	*Y*	*Z*
0,130	0,165	0,065
0,082	0,165	0,225
0,240	0,165	0,272
0,290	0,165	0,114

Tabela A.9: Vértices da face 2 do bloco curto na cena Cornell Box

X	*Y*	*Z*
0,290	0,0	0,114
0,290	0,165	0,114
0,240	0,165	0,272
0,240	0,0	0,272

O bloco mais alto da cena *Cornell Box* tem faces descritas pelos vértices dados nas tabelas A.13, A.14, A.15, A.16 e A.17 e usa o material descrito na tabelas A.18 e A.19.

Tabela A.10: Vértices da face 3 do bloco curto na cena Cornell Box

X	*Y*	*Z*
0,130	0,0	0,065
0,130	0,165	0,065
0,290	0,165	0,114
0,290	0,0	0,114

Tabela A.11: Vértices da face 4 do bloco curto na cena Cornell Box

X	*Y*	*Z*
0,082	0,0	0,225
0,082	0,165	0,225
0,130	0,165	0,065
0,130	0,0	0,065

Tabela A.12: Vértices da face 5 do bloco curto na cena Cornell Box

X	*Y*	*Z*
0,240	0,0	0,272
0,240	0,165	0,272
0,082	0,165	0,225
0,082	0,0	0,225

Tabela A.13: Vértices da face 1 do bloco alto na cena Cornell Box

X	*Y*	*Z*
0,423	0,330	0,247
0,265	0,330	0,296
0,314	0,330	0,456
0,472	0,330	0,406

Tabela A.14: Vértices da face 2 do bloco alto na cena Cornell Box

X	*Y*	*Z*
0,423	0,0	0,247
0,423	0,330	0,247
0,472	0,330	0,406
0,472	0,0	0,406

[Este espaço foi propositalmente deixado em branco]

Tabela A.15: Vértices da face 3 do bloco alto na cena Cornell Box

X	*Y*	*Z*
0,472	0,0	0,406
0,472	0,330	0,406
0,314	0,330	0,456
0,314	0,0	0,456

Tabela A.16: Vértices da face 4 do bloco alto na cena Cornell Box

X	*Y*	*Z*
0,314	0,0	0,456
0,314	0,330	0,456
0,265	0,330	0,296
0,265	0,0	0,296

Tabela A.17: Vértices da face 5 do bloco alto na cena Cornell Box

X	*Y*	*Z*
0,265	0,0	0,296
0,265	0,330	0,296
0,423	0,330	0,247
0,423	0,0	0,247

Materiais

A refletividade do material utilizado para a cor branca na cena *Cornell Box* em diferentes comprimentos de onda são dados nas tabelas A.18 e A.19.

[Este espaço foi propositalmente deixado em branco]

Tabela A.18: Material “branco” na cena Cornell Box

Comprimento de onda (*nm*)	Refletividade
400	0,343
404	0,445
408	0,551
412	0,624
416	0,665
420	0,687
424	0,708
428	0,723
432	0,715
436	0,710
440	0,745
444	0,758
448	0,739
452	0,767
456	0,777
460	0,765
464	0,751
468	0,745
472	0,748
476	0,729
480	0,745
484	0,757
488	0,753
492	0,750
496	0,746
500	0,747
504	0,735
508	0,732
512	0,739
516	0,734
520	0,725
524	0,721
528	0,733
532	0,725
536	0,732
540	0,743

[Este espaço foi propositalmente deixado em branco]

Tabela A.19: Material “branco” na cena Cornell Box (continuação)

Comprimento de onda (*nm*)	Refletividade
544	0,744
548	0,748
552	0,728
556	0,716
560	0,733
564	0,726
568	0,713
572	0,740
576	0,754
580	0,764
584	0,752
588	0,736
592	0,734
596	0,741
600	0,740
604	0,732
608	0,745
612	0,755
616	0,751
620	0,744
624	0,731
628	0,733
632	0,744
636	0,731
640	0,712
644	0,708
648	0,729
652	0,730
656	0,727
660	0,707
664	0,703
668	0,729
672	0,750
676	0,760
680	0,751
684	0,739
688	0,724
692	0,730
696	0,740
700	0,737

A refletividade do material utilizado para a cor verde na cena *Cornell Box* em diferentes comprimentos de onda são dados nas tabelas A.20 e A.21.

Tabela A.20: Material “verde” na cena Cornell Box

Comprimento de onda (*nm*)	Refletividade
400	0,092
404	0,096
408	0,098
412	0,097
416	0,098
420	0,095
424	0,095
428	0,097
432	0,095
436	0,094
440	0,097
444	0,098
448	0,096
452	0,101
456	0,103
460	0,104
464	0,107
468	0,109
472	0,112
476	0,115
480	0,125
484	0,140
488	0,160
492	0,187
496	0,229
500	0,285
504	0,343
508	0,390
512	0,435
516	0,464
520	0,472
524	0,476
528	0,481
532	0,462
536	0,447
540	0,441

[Este espaço foi propositalmente deixado em branco]

Tabela A.21: Material “verde” na cena Cornell Box (continuação)

Comprimento de onda (*nm*)	Refletividade
544	0,426
548	0,406
552	0,373
556	0,347
560	0,337
564	0,314
568	0,285
572	0,277
576	0,266
580	0,250
584	0,230
588	0,207
592	0,186
596	0,171
600	0,160
604	0,148
608	0,141
612	0,136
616	0,130
620	0,126
624	0,123
628	0,121
632	0,122
636	0,119
640	0,114
644	0,115
648	0,117
652	0,117
656	0,118
660	0,120
664	0,122
668	0,128
672	0,132
676	0,139
680	0,144
684	0,146
688	0,150
692	0,152
696	0,157
700	0,159

[Este espaço foi propositalmente deixado em branco]

A refletividade do material utilizado para a cor vermelha na cena *Cornell Box* em diferentes comprimentos de onda são dados nas tabelas A.22 e A.23.

Tabela A.22: Material "vermelho" na cena Cornell Box

Comprimento de onda (*nm*)	Refletividade
400	0,040
404	0,046
408	0,048
412	0,053
416	0,049
420	0,050
424	0,053
428	0,055
432	0,057
436	0,056
440	0,059
444	0,057
448	0,061
452	0,061
456	0,060
460	0,062
464	0,062
468	0,062
472	0,061
476	0,062
480	0,060
484	0,059
488	0,057
492	0,058
496	0,058
500	0,058
504	0,056
508	0,055
512	0,056
516	0,059
520	0,057
524	0,055
528	0,059
532	0,059
536	0,058
540	0,059

[Este espaço foi propositalmente deixado em branco]

Tabela A.23: Material “vermelho” na cena Cornell Box (continuação)

Comprimento de onda (*nm*)	Refletividade
544	0,061
548	0,061
552	0,063
556	0,063
560	0,067
564	0,068
568	0,072
572	0,080
576	0,090
580	0,099
584	0,124
588	0,154
592	0,192
596	0,255
600	0,287
604	0,349
608	0,402
612	0,443
616	0,487
620	0,513
624	0,558
628	0,584
632	0,620
636	0,606
640	0,609
644	0,651
648	0,612
652	0,610
656	0,650
660	0,638
664	0,627
668	0,620
672	0,630
676	0,628
680	0,642
684	0,639
688	0,657
692	0,639
696	0,635
700	0,642

[Este espaço foi propositalmente deixado em branco]

A.1.2 Dragão

O modelo do Dragão está disponível no "The Stanford 3D Scanning Repository" e pode ser obitdo em [http://graphics.stanford.edu/data/3Dscanrep/]. O recurso "*decimate*" do *software Blender* é aplicado ao modelo com o objetivo de se ter um modelo por volta de 30000 polígonos.

A imagem renderizada foi processada utilizando-se o software "*qtpfsgui*" (atualmente chamado de *Luminance*). A configuração utilizada no processamento é dada por:

```
TMO=Reinhard02
KEY=0.18
PHI=1
SCALES=NO
RANGE=8
LOWER=1
UPPER=43
PREGAMMA=1
```

A.1.3 Buddha

O modelo do *Buddha* está disponível no "The Stanford 3D Scanning Repository" e pode ser obitdo em [http://graphics.stanford.edu/data/3Dscanrep/]. O recurso "*decimate*" é aplicado ao modelo com o objetivo de se ter um modelo por volta de 30000 polígonos.

A imagem renderizada foi processada utilizando-se o software "*qtpfsgui*" (atualmente chamado de *Luminance*). A configuração utilizada no processamento é dada por:

```
TMO=Pattanaik00
MULTIPLIER=1
LOCAL=NO
AUTOLUMINANCE=NO
CONE=0.5
ROD=0.5
PREGAMMA=1
```

A.1.4 Bunny

O modelo do *Bunny* está disponível no "The Stanford 3D Scanning Repository" e pode ser obitdo em [http://graphics.stanford.edu/data/3Dscanrep/]. O recurso "*decimate*" é aplicado ao modelo com o objetivo de se ter um modelo por volta de 30000 polígonos.

A imagem renderizada foi processada utilizando-se o software "*qtpfsgui*" (atualmente chamado de Luminance). A configuração utilizada no processamento é dada por:

```
TMO=Pattanaik00
MULTIPLIER=1
LOCAL=NO
AUTOLUMINANCE=NO
CONE=0.5
ROD=0.5
PREGAMMA=1
```

A.1.5 Grade

Por uma questão de simplicidade e simetria, descreve-se aqui apenas um dos 8 "gomos" da grade utilizada. A lista de vértices é descrita na tabela A.24 e a lista de faces é descrita na tabela A.25. As faces são triangulares a tabela lista o "Id" de cada um dos vértices de cada face.

[Este espaço foi propositalmente deixado em branco]

Tabela A.24: Vértices de um gomo da grade

Id	*X*	*Y*	*Z*
1	-1,00000000	-1,00000000	0,52000000
2	-1,00000000	1,00000000	0,52000000
3	1,00000000	1,00000000	0,52000000
4	1,00000000	-1,00000000	0,52000000
5	-1,00000000	-1,00000000	-0,52000000
6	-1,00000000	1,00000000	-0,52000000
7	1,00000000	1,00000000	-0,52000000
8	1,00000000	-1,00000000	-0,52000000
9	1,00000000	0,84400000	0,43888000
10	1,00000000	-0,84400000	0,43888000
11	1,00000000	-0,84400000	-0,43888000
12	1,00000000	0,84400000	-0,43888000
13	-1,00000000	-0,84400000	0,43888000
14	-1,00000000	0,84400000	0,43888000
15	-1,00000000	0,84400000	-0,43888000
16	-1,00000000	-0,84400000	-0,43888000

[Este espaço foi propositalmente deixado em branco]

Tabela A.25: Faces de um gomo da grade

V_1	V_2	V_3
1	4	2
1	5	4
1	14	13
1	16	5
2	4	3
2	7	6
2	14	1
3	7	2
3	10	9
3	12	7
4	10	3
5	7	8
5	8	4
5	16	6
6	7	5
6	14	2
6	15	14
6	16	15
7	12	8
8	10	4
8	11	10
8	12	11
9	12	3
9	14	12
10	14	9
10	16	13
11	16	10
12	16	11
13	14	10
13	16	1
14	15	12
15	16	12

A.1.6 Gaiola

Por uma questão de simplicidade e simetria, descreve-se aqui apenas metade de uma das "fatias" da gaiola utilizada. A lista de vértices é descrita na tabela A.26 e a lista de faces é descrita na tabela A.27 e A.28. As faces são triangulares a tabela lista o "Id" de cada um dos vértices de cada face.

[Este espaço foi propositalmente deixado em branco]

Tabela A.26: Vértices de metade de uma fatia da gaiola

Id	*X*	*Y*	*Z*
1	-0,49606497	1,00000000	0,86828541
2	-0,79058271	1,00000000	0,61235527
3	-0,66642838	0,60000000	0,74556906
4	-0,52017514	0,20000000	0,85405962
5	-0,80741363	0,20000000	0,58998579
6	-0,78584511	0,94921118	0,61221740
7	-0,80596861	0,25078882	0,58547184
8	-0,68164554	0,60841118	0,72644480
9	-0,66595772	0,65078882	0,74085288
10	-0,52517565	0,97933133	0,84110794
11	-0,76904957	0,97933133	0,63318673
12	-0,53860805	0,20841118	0,83801196
13	-0,66179433	0,54921118	0,74457434
14	-0,78987626	0,19158882	0,60700755
15	-0,52056753	0,25078882	0,84933628
16	-0,49174076	0,94921118	0,86634503
17	-0,64572343	0,59158882	0,75855384
18	-0,48488188	0,98753584	0,86001998
19	-0,77708784	0,98753584	0,60899859
20	-0,65795917	0,60000000	0,73609411
21	-0,51356457	0,20000000	0,84320593
22	-0,79715273	0,20000000	0,58248805
23	-0,77482327	0,94921118	0,60639724
24	-0,79732271	0,25078882	0,57649388
25	-0,67165706	0,60841118	0,71898922
26	-0,65937698	0,65078882	0,73026755
27	-0,51708908	0,97933133	0,83162307
28	-0,76096299	0,97933133	0,62370187
29	-0,53013719	0,20841118	0,82886866
30	-0,65201067	0,54921118	0,73685194
31	-0,78147578	0,19158882	0,59779956
32	-0,51601548	0,25078882	0,83773309
33	-0,48378518	0,94921118	0,85675002
34	-0,63943073	0,59158882	0,74779479
35	-5,5511151e-17	1,00000000	-5,5511151e-17
36	0,0000000e+0	0,98753584	0,0000000e+0

[Este espaço foi propositalmente deixado em branco]

Tabela A.27: Faces de metade de uma fatia da gaiola

V_1	V_2	V_3
1	10	3
1	11	10
1	17	16
1	18	35
1	35	2
2	6	3
2	11	1
2	19	6
3	6	8
3	7	5
3	10	9
3	11	2
3	12	4
3	14	13
3	15	17
3	17	1
4	15	3
4	21	15
5	14	3
5	22	14
6	19	23
6	25	8
7	22	5
7	25	24
8	7	3
8	25	7
9	11	3
9	26	11
10	26	9
10	28	27
11	26	28
11	28	10
12	21	4
12	30	29
13	12	3
13	30	12
14	22	31
14	30	13
15	21	32
15	34	17
16	18	1
16	34	33

Tabela A.28: Faces de metade de uma fatia da gaiola(continuação)

V_1	V_2	V_3
17	34	16
18	26	27
18	27	19
18	34	20
19	25	23
19	26	20
19	27	28
19	35	36
19	36	18
20	25	19
20	26	18
20	30	22
20	34	21
21	30	20
21	34	32
22	25	20
22	30	31
23	25	6
24	22	7
24	25	22
27	26	10
28	26	19
29	21	12
29	30	21
31	30	14
32	34	15
33	18	16
33	34	18
35	19	2
36	35	18

[Este espaço foi propositalmente deixado em branco]

A.1.7 Luminária

Por uma questão de simplicidade e simetria, descreve-se aqui apenas uma das fatias da luminária utilizada. A lista de vértices é descrito na tabela A.29 e a lista de faces é descrita na tabela A.30. As faces são triangulares a tabela lista o "Id" de cada um dos vértices de cada face.

Tabela A.29: Vértices de uma fatia

Id	*X*	*Y*	*Z*
1	0,38268343	0,78143077	0,92387953
2	6,1232340e-17	0,78143077	1,00000000
3	1,3877788e-16	0,45253612	5,5511151e-17
4	0,38268343	0,74295673	0,92387953
5	6,1232340e-17	0,74295673	1,00000000
6	1,3877788e-16	0,41406208	1,9094958e-17

Tabela A.30: Faces de uma fatia

V_1	V_2	V_3
1	3	2
1	4	3
1	5	4
2	5	1
3	5	2
5	3	6
5	6	4
6	3	4

[Este espaço foi propositalmente deixado em branco]

A.2 Justificativa dos testes

A quantidade variável de estimativas foi utilizada apenas no teste com a *Cornell Box* a fim de se verificar a linearidade do tempo de renderização com o número de estimativas utilizado. Uma vez que tal linearidade é verificada, torna-se desnecessário repetir todos os outros testes com números diferentes de estimativas.

Os processamentos de imagem são utilizados apenas com a finalidade de facilitar a visualização e evidenciar as diferenças entre os resultados dos diferentes métodos.

O método de *Path Tracing* e *Coherent Path Tracing* não foram testados contra as cenas da grade, gaiola e luminária já que seu desempenho pode ser estimado a partir dos resultados obitdos e por serem cenas que foram feitas com o objetivo de ilustrar as diferenças entre BDPT e CBDPT.

A comparação com MLT foi feita apenas para ilustrar a diferença entre resultados de dois melhoramentos distintos de BDPT.

A.3 Máquina usada nos testes

Saída do comando “gcc –version”:

```
gcc (Ubuntu 4,4,1-4ubuntu9) 4,4,1
Copyright (C) 2009 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Saída do comando “uname -a”:

```
Linux marco-desktop 2,6,31-22-generic-pae #60-Ubuntu SMP
Thu May 27 01:40:15 UTC 2010 i686 GNU/Linux
```

Saída do comando “cat /proc/cpuinfo”:

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
```

```
model : 15
model name : Intel(R) Core(TM)2 Duo CPU     E4600  @ 2,40GHz
stepping : 13
cpu MHz : 1200,000
cache size : 2048 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 2
apicid : 0
initial apicid : 0
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 10
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx lm constant_tsc arch_perfmon
pebs bts pni dtes64 monitor ds_cpl est tm2 ssse3 cx16 xtpr pdcm lahf_lm
bogomips : 4798,91
clflush size : 64
power management:
```

APÊNDICE B – Sobre a implementação

A implementação foi feita usando-se o renderizador *Luxrender* que é um *"fork"* do pbrt que é o renderizador descrito em [6].

APÊNDICE C – Sobre integração de Monte Carlo

C.1 Introdução

Foi anteriormente dito que renderizar uma imagem é equivalente resolver-se uma integral para cada pixel. Discute-se, agora, um método de integração que tem uma larga aplicação na área de computação gráfica. Tal método de integração, baseada em processo estocásticos, é chamado de integração de Monte Carlo.

Chamamos de método de Monte Carlo, uma classe de métodos numéricos baseados em processos estocásticos. Integração de Monte Carlo é uma aplicação de um método de Monte Carlo a fim de se resolver (mais corretamente, estimar ou aproximar) uma integral. Integração de Monte Carlo tem suas raízes em técnicas de simulação de espalhamento de nêutrons em física computacional. Em computação gráfica, estes métodos têm uma grande aceitação dada sua simplicidade e recursos de ajustabilidade.

C.2 Visão geral de integração de Monte Carlo

De maneira simplificada, suponha que queira-se resolver a seguinte integral:

$$\int_a^b f(x)\,dx \tag{C.1}$$

Isto é equivalente a calcular-se a área hachurada na Figura C.1.

Se tivermos o valor médio para f no intervalo $[a,b]$, denotado aqui por $\langle f \rangle$, calcular a integral seria simplesmente uma questão de fazer-se:

$$\int_a^b f(x)\,dx = (b-a) * \langle f \rangle \tag{C.2}$$

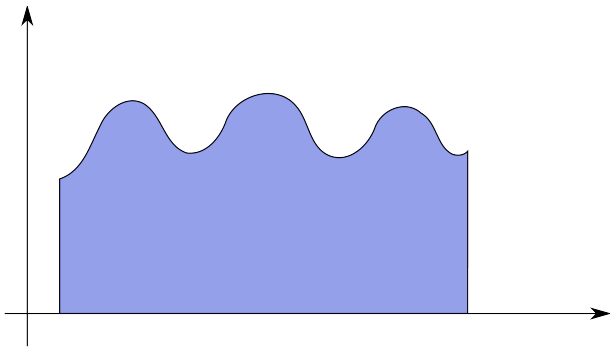


Figura C.1: Área a ser calculada

Na sua forma mais simples, uma integração de Monte Carlo é baseada na ideia de que se tivermos uma boa aproximação para $\langle f \rangle$, podemos obter uma boa aproximação para $\int_a^b f(x)\,dx$. Assim, a questão agora é reduzida a desenvolver-se um método para estimar $\langle f \rangle$. Por questão de simplicidade, estimaremos $\langle f \rangle$ como uma média aritmética de f amostrada aleatoriamente no intervalo $[a,b]$. Isto é:

$$\langle f \rangle \approx \frac{1}{N} * \sum_{i=1}^{N} f(x_i) \tag{C.3}$$

Onde N é o número de amostras usadas na nossa estimativa. Desta forma, temos um aproximação para a nossa integral:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{N} * \sum_{i=1}^{N} f(x_i) \tag{C.4}$$

Na Figura C.2 tem-se a função amostrada aleatoriamente.

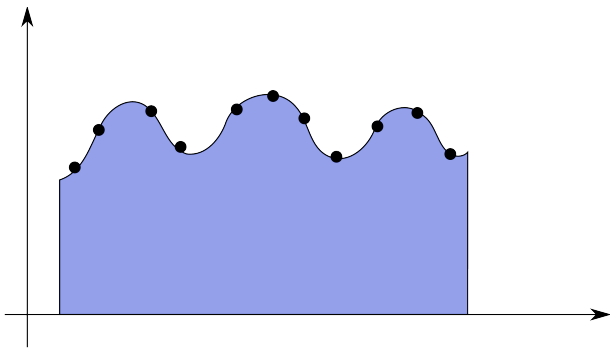


Figura C.2: Função amostrada aleatoriamente

Na Figura C.3 computa-se o valor médio para a função no intervalo amostrado.

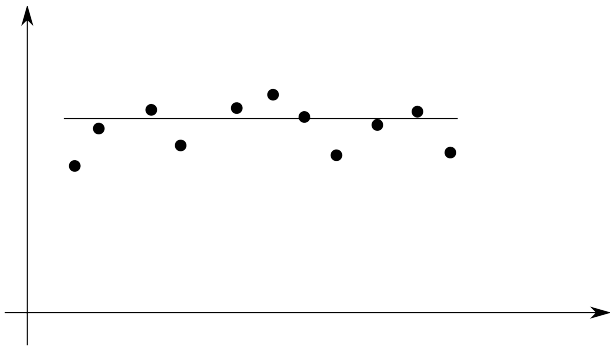


Figura C.3: Computando valor médio

Na Figura C.4 estima-se o valor da integral a partir do valor médio computado.

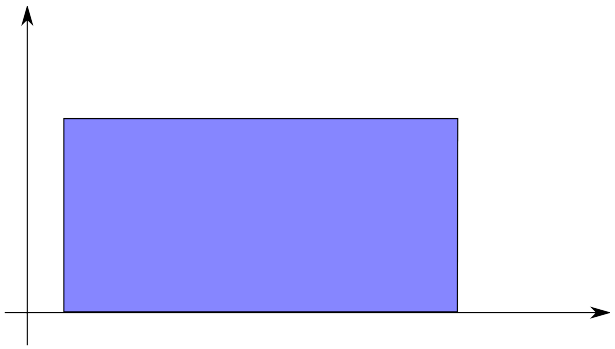


Figura C.4: Estimando a integral

É claro que existem métodos melhores para computarmos uma aproximação para $\langle f \rangle$ e alguns outros métodos para esta classe de problemas são discutidos em [6], [8] e [14]. Também, há algoritmos muito mais bem comportados para calcular-se uma integral unidimensional simples como $\int_a^b f(x)\,dx$ e usar-se integração de Monte Carlo para este caso é, de fato, um abuso do método. De qualquer forma, integração de Monte Carlo é uma solução simples, elegante, e ajustável para uma larga classe de problemas. Sua vantagem sobre métodos tradicionais são mais relevantes em situações onde sua generalidade e simplicidade tornam-se um ponto forte, daí então é de amplo uso em computação gráfica. Um exemplo frequentemente utilizado é a estimativa de integrais multidimensionais. Por exemplo:

$$\int_{x_l}^{x_u} \int_{y_l}^{y_u} f(x,y,\ldots)\,dx\,dy\ldots = \int_V f(x,y,\ldots)\,dx\,dy\ldots \tag{C.5}$$

Sobre o hipercubo com "volume" V definido por $\{(x,y,\ldots)|x_l < x < x_u, y_l < y < y_u \ldots\}$.

O símbolo V usado aqui denota volume de integração. Não é necessariamente a forma comum de senso geral da definição de volume tridimensional e pode até mesmo ter menos de três dimensões. Será intuitivamente descrito como ”volume do espaço de integração”, e para espaços simples e lineares a serem amostrados pode ser facilmente calculado como o produto do comprimento dos intervalos de cada variável de integração.

Neste caso, temos agora uma estimativa para o valor médio de nossa função multidimensional f no espaço de integração:

$$\langle f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{C.6}$$

E nossa integral pode facilmente ser estimada por:

$$\int_V f(x,y,\ldots)\,dx\,dy\ldots \approx V * \langle f \rangle \tag{C.7}$$

$$\approx \frac{V}{N} \sum_{i=1}^{N} f(x_i) \tag{C.8}$$

Estimativa de erro em integração de Monte Carlo

Uma descrição detalhada de estimativa de erro em uma integração de Monte Carlo está além do escopo deste texto. De qualquer forma, discutiremos alguns pontos relevantes para reduzir-se o erro.

É sabido que o erro em uma integração de Monte Carlo simples como a que é descrita é da ordem do inverso da raiz quadrada do número de amostras. Isto é:

$$O(\frac{1}{\sqrt{N}}) \tag{C.9}$$

Isto significa que para diminuir-se o erro da integração pela sua metade, tem-se de fazer uma acréscimo de 4 vezes o número de amostras na integração.

Outro ponto relevante é que também é sabido que o erro está relacionado à variância das amostras. Integração de Monte Carlo, assim como ocorre com outros métodos estocásticos, pode ser sensível a distribuições de amostragens ruins. Por esta razão, discute-se algumas técnicas simples para melhorar a homogeneidade da distribuição das amostras, reduzir variância e determinar quais partes do domínio têm maior influência no erro da estimativa.

C.2.1 Jitter Sampling unidimensional

Jitter Sampling é uma forma de amostragem estratificada. A ideia é forçar algumas propriedades da distribuição tais que se possa ter uma distribuição de amostragem mais homogênea do domínio. O conceito por trás de *Jitter Sampling* é bastante simples. Deseja amostrar-se N amostras em um domínio de volume V; então, divide-se o domínio de integração em N partições "igualmente espaçadas", "igualmente distribuídas" e com "igual volume" e usa-se uma amostra de cada qual dessas partições. A Figura C.5 ilustra o método para o caso unidimensional.

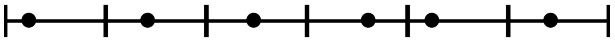


Figura C.5: Jitter Sampling unidimensional

C.2.2 Jitter Sampling bidimensional

A Figura C.6 ilustra o mesmo conceito aplicado a domínios multidimensionais. Na área de computação gráfica é comum o caso de integração de domínios bidimensionais.

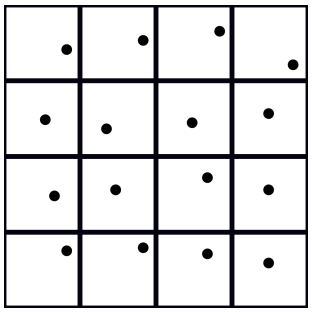


Figura C.6: Jitter sampling bidimensional

C.2.3 Subdivisão de domínio

Subdivisão de domínio compõe uma classe de algoritmos que alocam mais amostras em regiões do domínio que têm uma contribuição maior para o erro da integração. Explora-se a ideia de que "regiões do domínio onde a variância das amostras é maior tem uma maior contribuição para o erro da integração". Pode-se então fazer uma subdivisão inteligente do domínio baseada na estimativa de variância que pode ser calculada a partir das amostras. A Figura C.7 ilustra o processo para um espaço unidimensional.

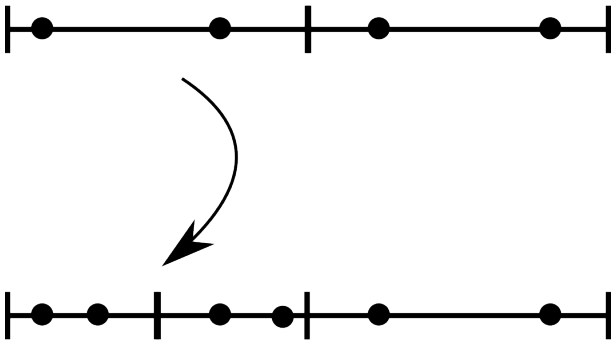


Figura C.7: Subdivisão de domínio unidimensional

O conceito é também facilmente explicável: se uma região do domínio tem amostras com variância relativa acima de um ponto mínimo pré-determinado, então subidivide-se esta região e, recursivamente, faz-se uma nova integração da região com mais amostras. É claro que um critério de parada deve ser definido, caso contrário a integração pode levar um tempo infinito dependendo da variância da distribuição do domínio que é recursivamente amostrada.

Da mesma forma como acontece em *Jitter Sampling*, o mesmo conceito pode ser aplicado a domínios multidimensionais como ilustrado na figura C.8.

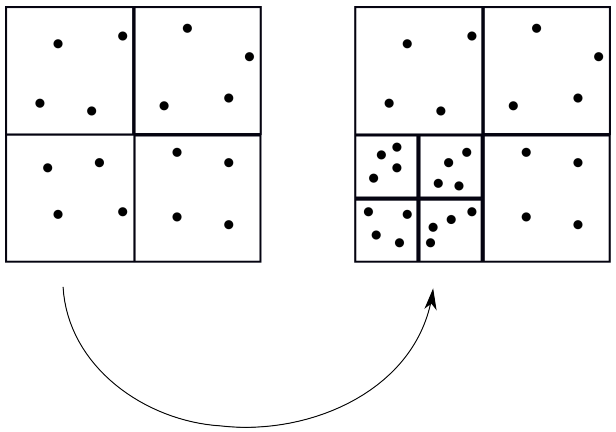


Figura C.8: Subdivisão de domínio bidimensional

APÊNDICE D – Sobre Ray Tracing

D.1 História

Ray Tracing foi anteriormente (mas em um conceito diferente) utilizado em física e desenho de lentes séculos antes da sua introdução em computação gráfica. *Ray Tracing* foi introduzido em computação gráfica em 1979 por Turner Whitted [15] como uma técnica de renderização e desde então tem sido bem sucedida em aceitação [8].

D.2 Resumo

A idéia por trás de *Ray Tracing* é a disparar raios do observador para uma grade de projeção e, em seguida, verificar o que acontece com os raios dada a cena. Se o raio atinge um objeto, pode-se calcular a iluminação nesse ponto e colorir o pixel.

A iluminação local em um determinado ponto pode ser computado usando *Phong Shading*. Um raio refletido (e em meios transparentes, também um refratado) é calculado e sua cor é adicionada à estimativa de cor do pixel levando em conta a reflexividade da superfície naquele ponto (ou a equação de Snell no caso de um raio refratado).

D.3 Descrição

O algorimtmo de *Ray Tracing* começa por traçar um raio através de cada pixel da imagem. Cada pixel da imagem corresponde (biunivocamente) a uma célula (ou ponto) na grade de projeção, como é ilustrado na Figura D.1.

Deve-se, então, verificar o objeto mais próximo da cena que é intersectado pelo raio. Uma vez que se tem o ponto de intersecção entre raio e o objeto mais próximo da cena que é intersectado pelo raio deve-se verificar se esse ponto está em sombra. Isto é feito lançando-se um “raio

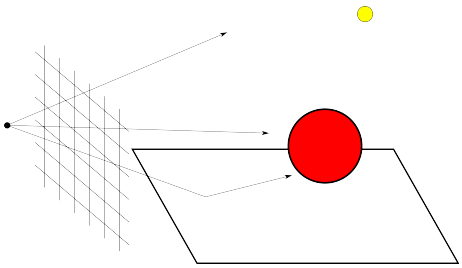


Figura D.1: Ilustração de Ray Tracing

de sombra", que é um raio que vai do ponto de interseção a uma fonte luminosa. Se o "raio de sombra" não tem qualquer interseção entre o ponto sendo testado e a fonte luz, considera-se que este ponto recebe luz da fonte luminosa. Caso contrário, considera-se que o ponto está em sombra.

A iluminação local em *Ray Tracing* é feita (em geral) usando *Phong Shading*. Depois que o a iluminação local é computada, traça-se recursivamente um raio refletido.

Computar o raio refletido é apenas uma questão de geometria como ilustrado na Figura D.2.

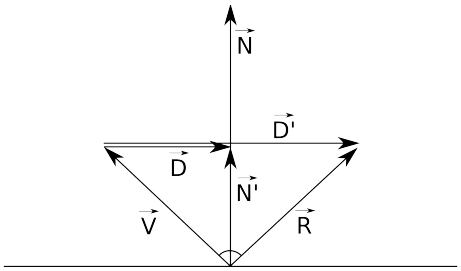


Figura D.2: Reflexão especular

A partir da Figura D.2 pode-se ver que $\overrightarrow{N'}$ é a projeção de $\overrightarrow{V}$ sobre $\overrightarrow{N}$. $\overrightarrow{N'}$ é dado por:

$$\overrightarrow{N'} = \frac{\overrightarrow{V} \cdot \overrightarrow{N}}{|\overrightarrow{N}|} \tag{D.1}$$

O vetor $\overrightarrow{D}$ é a diferença entre os vetores $\overrightarrow{N'}$ e $\overrightarrow{V}$:

$$\overrightarrow{D} = \overrightarrow{N'} - \overrightarrow{V} \tag{D.2}$$

Também é fácil ver que $\overrightarrow{D'}$ é "duas vezes" o vetor $\overrightarrow{D}$:

$$\overrightarrow{D'} = 2 \cdot \overrightarrow{D} \tag{D.3}$$

A partir da equação anterior, obtem-se:

$$\overrightarrow{D'} = 2 \cdot (\overrightarrow{N'} - \overrightarrow{V}) \tag{D.4}$$

Indo um pouco mais longe com substituições, obtemos:

$$\overrightarrow{D'} = 2 \cdot \left(\frac{\overrightarrow{V} \cdot \overrightarrow{N}}{|\overrightarrow{N}|} - \overrightarrow{V} \right) \tag{D.5}$$

Então, calcular $\overrightarrow{V}$ refletido especularmente sobre $\overrightarrow{N}$ agora é simples:

$$\overrightarrow{R} = \overrightarrow{V} + \overrightarrow{D'} \tag{D.6}$$

$$= \overrightarrow{V} + 2 \cdot \left(\frac{\overrightarrow{V} \cdot \overrightarrow{N}}{|\overrightarrow{N}|} - \overrightarrow{V} \right) \tag{D.7}$$

$$= 2 \cdot \left(\frac{\overrightarrow{V} \cdot \overrightarrow{N}}{|\overrightarrow{N}|} \right) - \overrightarrow{V} \tag{D.8}$$

Uma vez que o raio refletido é calculado recursivamente, pode-se aplicar o algoritmo e obter reflexão especular na imagem renderizada.

D.4 Detalhes

Ray Tracing provocou e inspirou muitas outras técnicas de renderização semelhantes. É uma técnica de simples de compreensão e uma solução elegante para renderização. Além disso, *Ray Tracing* automaticamente trata reflexões especulares, refraçãoes e sombras.

D.5 Resultados

Imagens de *Ray Tracing* têm um sentimento comum de especularidade. Embora o resultado seja visualmente agradável, o resultado não é necessariamente foto-realistíco. Novos métodos nasceram para superar este ponto fraco de ray-tracing.

APÊNDICE E – Sobre outros métodos

E.1 Photon Mapping

Photon Mapping (PM) é uma técnica desenvolvida por Wann Jensen em 1995 [16]. A ideia por trás de PM trata de lançar fótons a partir das fontes de luz, anotar (em um mapa de fótons) aonde esses fótons "quicam" e utilizar essa informação posteriormente para calcular a radiância chegando em cada ponto da cena a partir de fótons "próximos" daquele ponto.

O método foi usado com sucesso para situações complexas de iluminação: usar uma mapa de fótons com uma coordenada temporal a fim de renderizar corretamente cenas em que há movimento [17], situações de iluminação em cabelos [18], espalhamento sub-superficial e meio participativo [8].

A estratégia, contudo, tem seus problemas. Ajustar corretamente os parâmetros do algritmo para se obter uma cena realística não é uma tarefa automática. O método exige a manutenção de uma estrutura para consulta da posição dos fótons na fase de renderização. O desempenho do algoritmo é muito dependente da estrutura utilizada e da distribuição de fótons na cena. Guardar a lista de fótons exige memória. A convergência para a renderização correta depende do número de fótons utilizado (diferentemente dos métodos derivados de PT que convergem com o tempo de execução).

Melhoramentos, como *Progressive Photon Mapping* [19], diminuem alguns dos problemas.

E.2 Radiosidade

Radiosidade é um método desenvolvido por Cindy Goral em 1984 [2]. A ideia por trás de radiosidade trata de considerar as superfícies como retalhos e perfeitamente lambertianas e encontrar um equilíbrio entre trocas de energia ente esses retalhos.

O método tem bons resultados para renderização de cenas que estão dentro de suas limitações. Os principais problemas de aplicação da técnica são as suas limitações. Radiosidade

não trata BRDF's arbitrárias, exige tesselação da cena, precisa resolver um sistema linear que se torna maior dependendo da complexidade e fineza da tesselação da cena.

Algumas estratégias recentes, como tesselação adaptativa ou radiosidade hierárquica, diminuem alguns dos problemas.

E.3 Radiance Caching

Radiance Caching (RC) foi desenvolvido por Gregory Ward em 1988 [3]. A ideia por trás de RC trata de calcular a radiância em um ponto da cena a ser renderizada; se o próximo ponto a ser renderizado está suficientemente próximo ao ponto anteriormente renderizado, então a radiância previamente calculada é reutilizada.

O método é mais indicado para situações com transições de iluminação suave e com poucas reflexões. Em superfícies muito especulares (pouco "*glossy*") há a recomendação de usar-se algum outro método como "*fallback*" para renderização [20] [21]. O método também não é tão geral e simples como os métodos derivados de PT.

Alguns desenvolvimentos recentes, como *Radiance Gradients* [22], reduzem alguns dos problemas.

E.4 Métodos de tempo real

Existem vários métodos de renderização em tempo real. A maioria desses métodos não tem o objetivo de ser tão geral ou correto como acontece com as estratégias derivadas de PT. Em geral, métodos de tempo real dependem de suporte de *hardware* e pré-calculam o máximo de informação possível [4]. Várias restrições e aproximações são impostas com o objetivo de se ter a renderização a tempo.

APÊNDICE F – Sobre imagens geradas

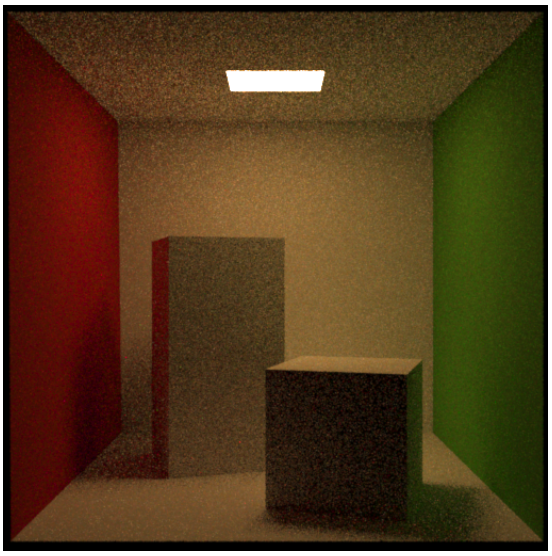
F.1 Introdução

Para simplificar a visualização das imagens renderizadas no Capítulo 4, coloca-se aqui as imagens renderizadas em dimensões maiores.

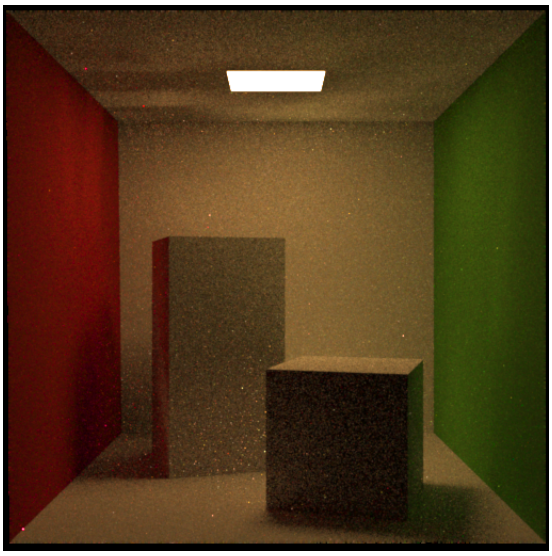
F.2 Cornell Box

[Este espaço foi propositalmente deixado em branco]

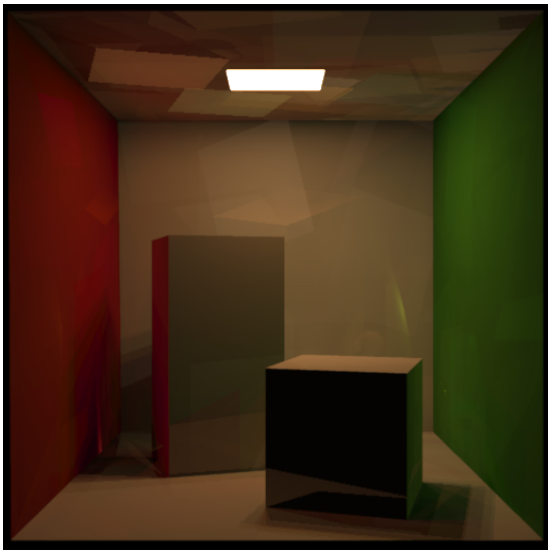
F.2.1 Cornell Box com 5 estimativas



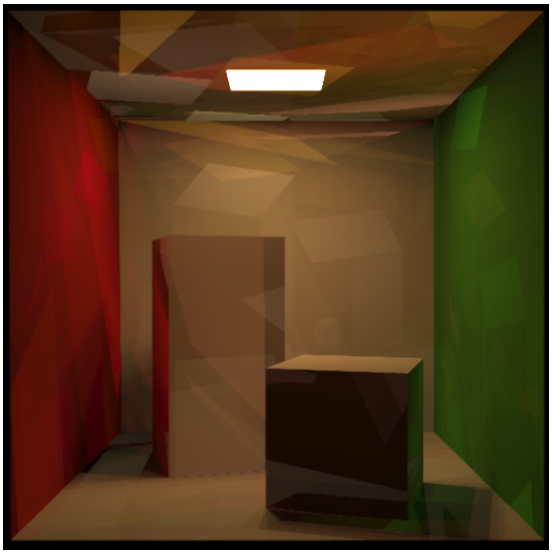
(a) PT, 8 vértices



(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



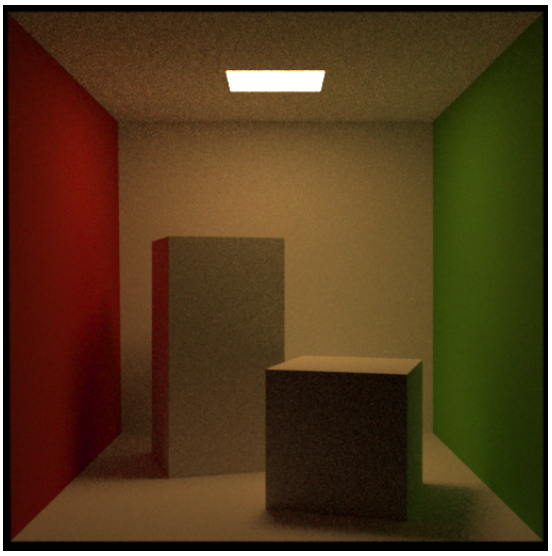
(c) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



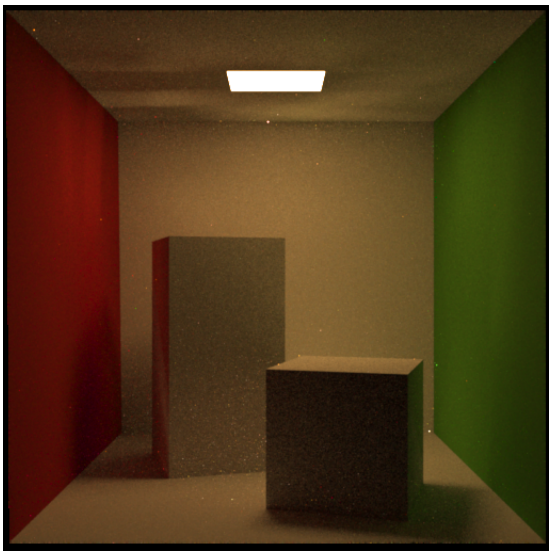
(d) CPT, 8 vértices

Figura F.1: Cornell box com 5 estimativas

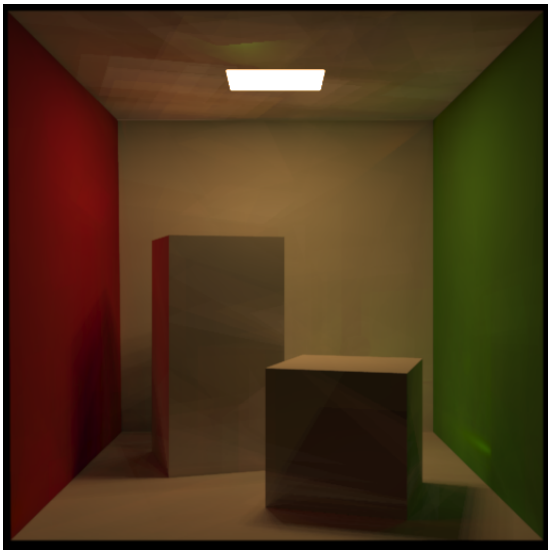
F.2.2 Cornell Box com 25 estimativas



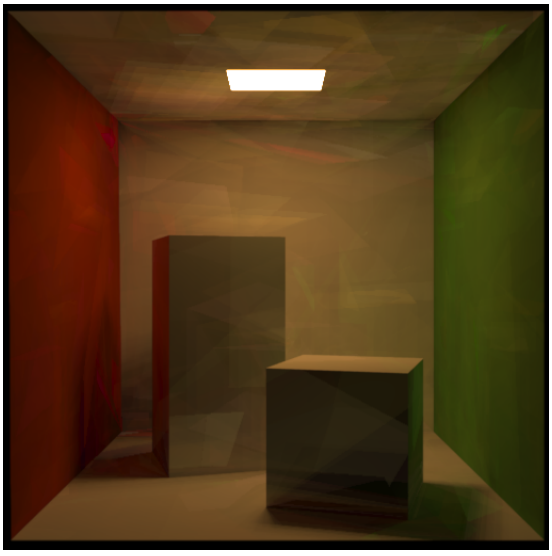
(a) PT, 8 vértices



(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



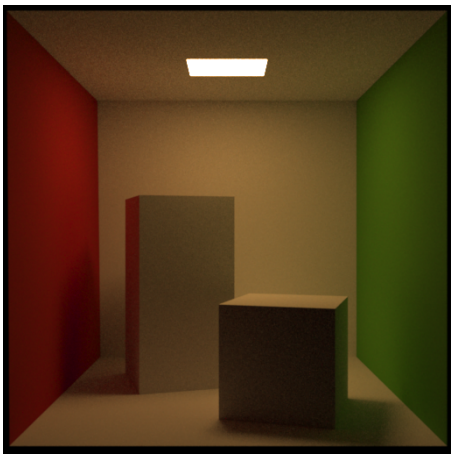
(c) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



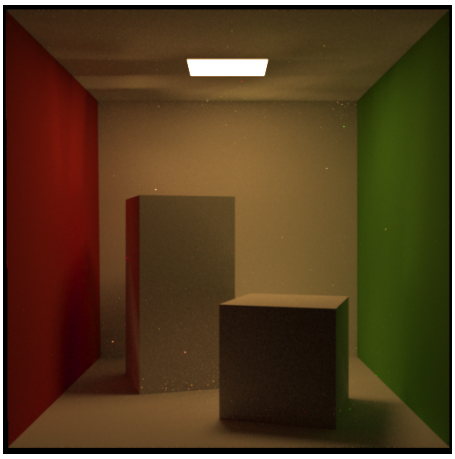
(d) CPT, 8 vértices

Figura F.2: Cornell box com 25 estimativas

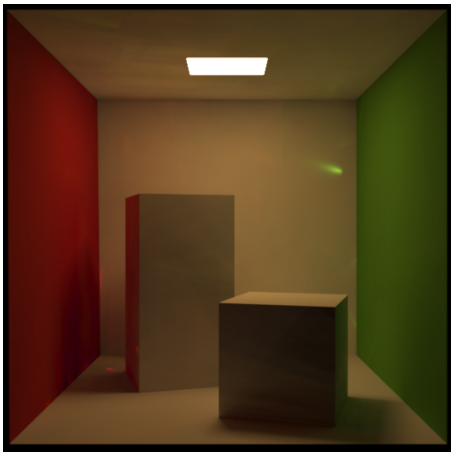
F.2.3 Cornell Box com 100 estimativas



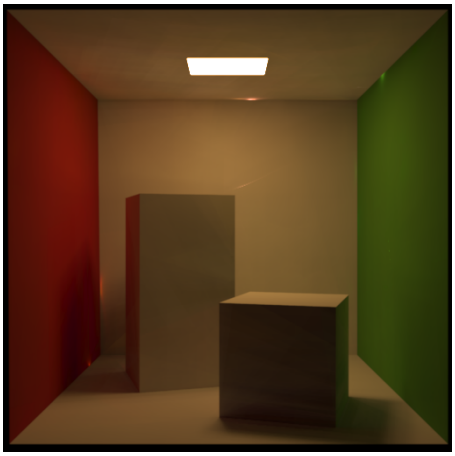
(a) PT, 8 vértices



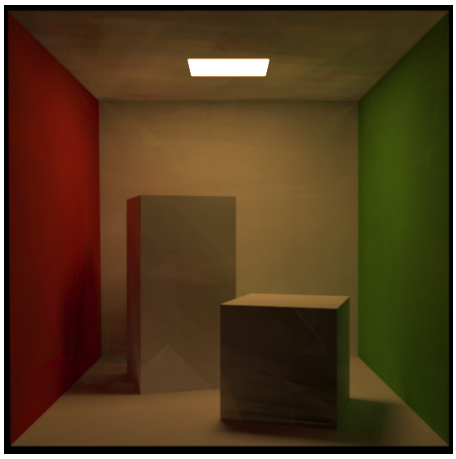
(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



(c) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



(d) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz

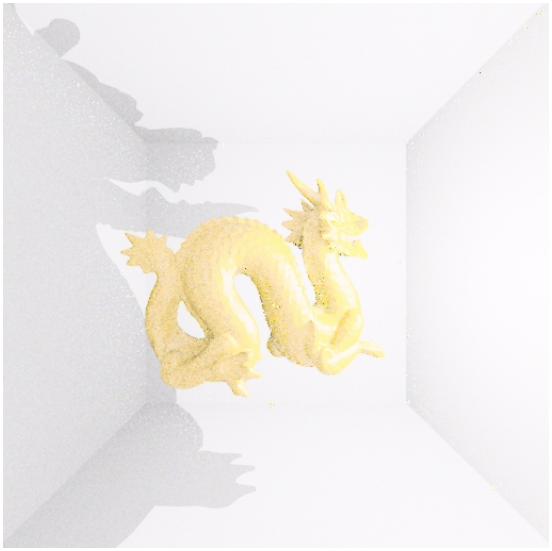


(e) CPT, 8 vértices

Figura F.3: Cornell box com 100 estimativas

F.3 Dragão

(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz

(c) CPT, 4 vértices

(d) PT, 4 vértices

Figura F.4: Dragão com 100 estimativas

F.4 Buddha

(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz

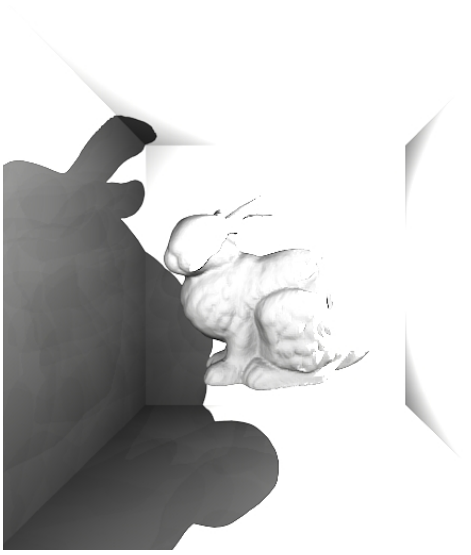
(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz

(c) CPT, 4 vértices

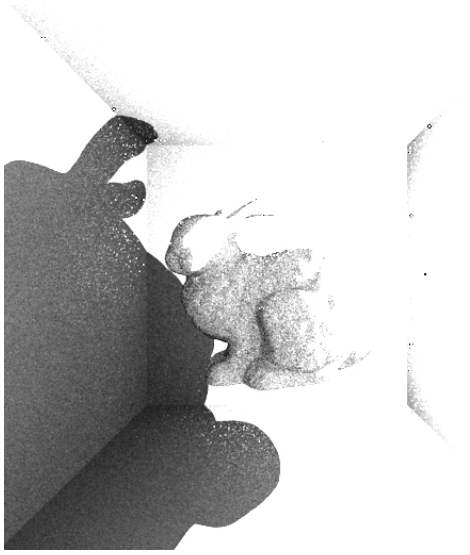
(d) PT, 4 vértices

Figura F.5: Buddha com 100 estimativas

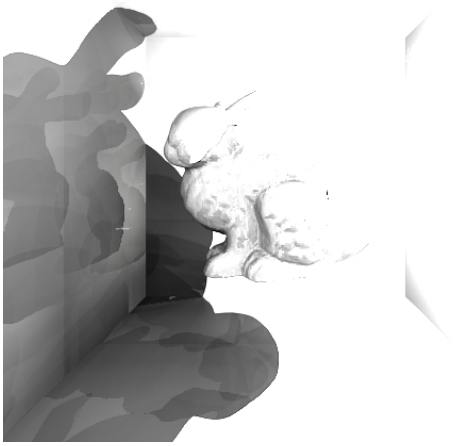
F.5 Bunny



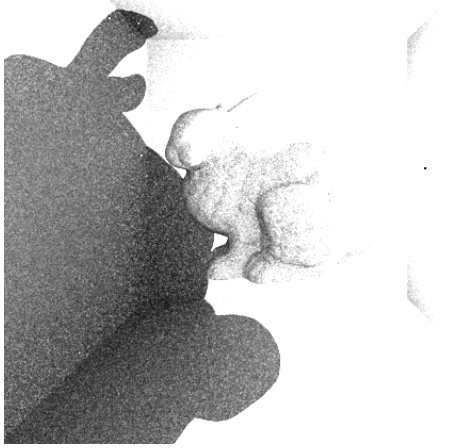
(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



(c) CPT, 4 vértices

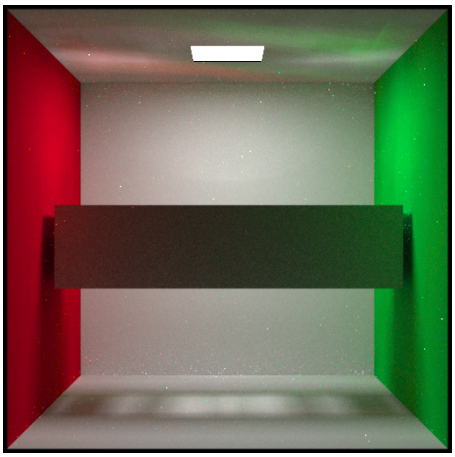


(d) PT, 4 vértices

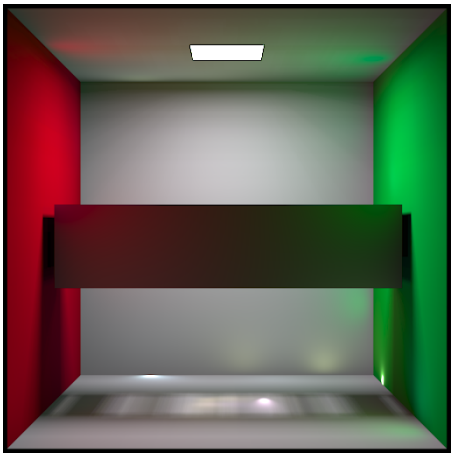
Figura F.6: Bunny com 100 estimativas

F.6 Grade

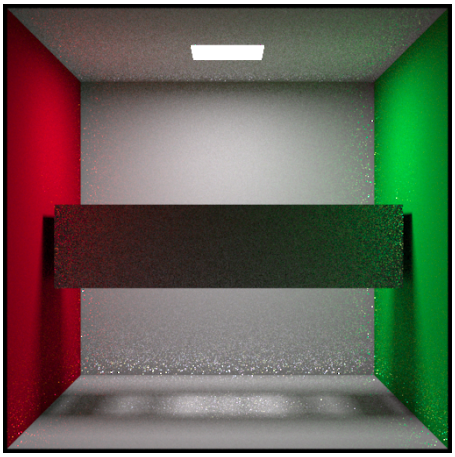
(a) CBDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



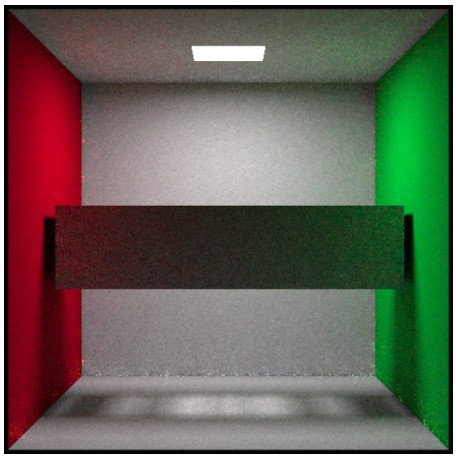
(b) BDPT, 4 vértices a partir do observador, 4 a partir da fonte de luz



(c) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



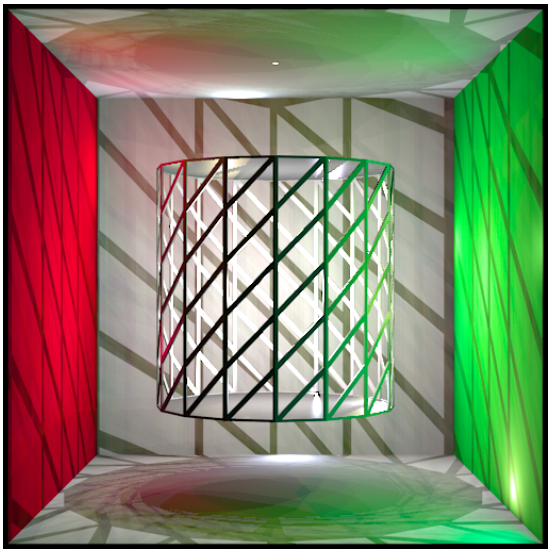
(d) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



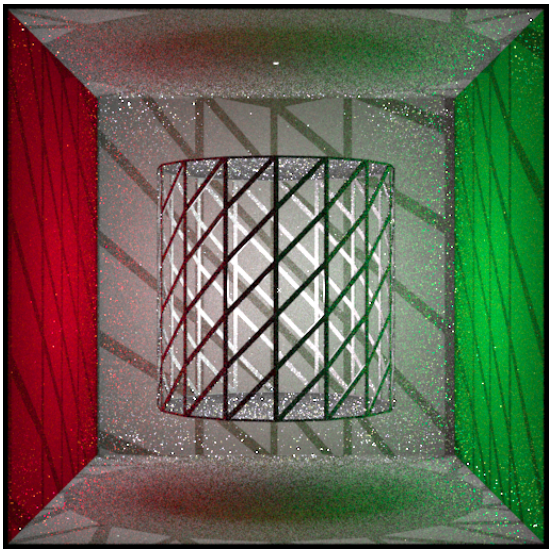
(e) MLT, 2 vértices a partir do observador, 2 a partir da fonte de luz

Figura F.7: Grade com 100 estimativas

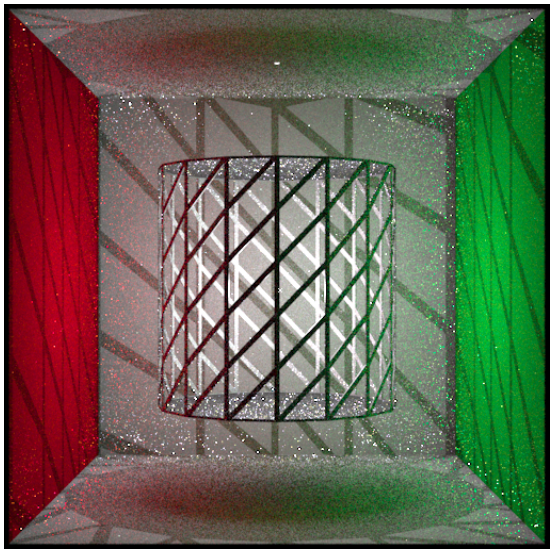
F.7 Gaiola



(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



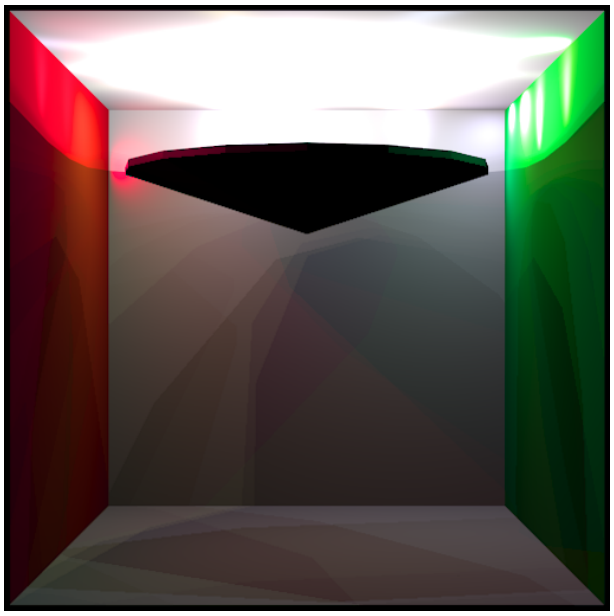
(b) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



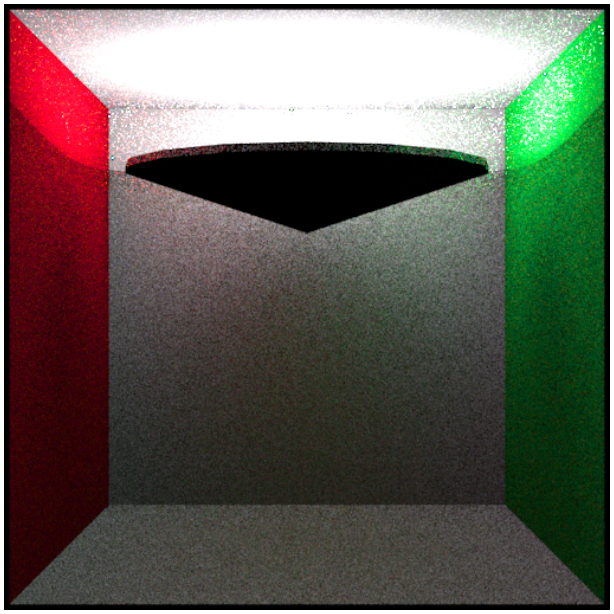
(c) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz

Figura F.8: Gaiola com 100 estimativas

F.8 Luminária



(a) CBDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz



(b) BDPT, 2 vértices a partir do observador, 2 a partir da fonte de luz

Figura F.9: Luminária com 100 estimativas

Referências Bibliográficas

[1] JENSEN, H. W. Global illumination using photon maps. In: *Proceedings of the eurographics workshop on Rendering techniques '96*. London, UK: Springer-Verlag, 1996. p. 21–30. ISBN 3-211-82883-4.

[2] GORAL, C. M.; TORRANCE, K. E.; GREENBERG, D. P.; BATTAILE, B. Modeling the interaction of light between diffuse surfaces. In: *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1984. p. 213–222. ISBN 0-89791-138-5.

[3] WARD, G. J.; RUBINSTEIN, F. M.; CLEAR, R. D. A ray tracing solution for diffuse interreflection. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1988. p. 85–92. ISBN 0-89791-275-6.

[4] KAUTZ, J.; SLOAN, P.-P.; LEHTINEN, J. Precomputed radiance transfer: theory and practice. In: *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*. New York, NY, USA: ACM, 2005. p. 1.

[5] KAJIYA, J. T. The rendering equation. In: *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1986. p. 143–150. ISBN 0-89791-196-2.

[6] PHARR, M.; HUMPHREYS, G. *Physically Based Rendering: From Theory to Implementation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. ISBN 012553180X.

[7] NICODEMUS, F. E. Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, OSA, v. 4, n. 7, p. 767–773, 1965. Disponível em: <http://ao.osa.org/abstract.cfm?URI=ao-4-7-767>.

[8] JENSEN, H. W. *Realistic image synthesis using photon mapping*. Natick, MA, USA: A. K. Peters, Ltd., 2001. ISBN 1-56881-147-0.

[9] VEACH, E.; GUIBAS, L. J. Optimally combining sampling techniques for monte carlo rendering. In: *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1995. p. 419–428. ISBN 0-89791-701-4.

[10] LAFORTUNE, E. P.; WILLEMS, Y. D. Bi-directional path tracing. In: *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93*. [S.l.: s.n.], 1993. p. 145–153.

[11] SADEGHI, I.; CHEN, B.; JENSEN, H. W. Coherent path tracing. *Journal of Graphics, GPU, & Game Tools*, v. 14, n. 2, p. 33–43, Jan 2009. Disponível em: <http://akpeters.metapress.com/content/g587l01346565265>.

[12] VEACH, E.; GUIBAS, L. J. Metropolis light transport. In: *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997. p. 65–76. ISBN 0-89791-896-7.

[13] CLINE, D.; TALBOT, J.; EGBERT, P. Energy redistribution path tracing. In: *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005. p. 1186–1195.

[14] COOK, R. L. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 5, n. 1, p. 51–72, 1986. ISSN 0730-0301.

[15] WHITTED, T. An improved illumination model for shaded display. In: *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1979. p. 14. ISBN 0-89791-004-4.

[16] JENSEN, H. W.; CHRISTENSEN, N. J. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers and Graphics*, v. 19, n. 2, p. 215 – 224, 1995. ISSN 0097-8493. Disponível em: <http://www.sciencedirect.com/science/article/B6TYG-3YB513D-20/2/3dcf97abd36e31d534151908c8b88c43>.

[17] CAMMARANO, M.; JENSEN, H. W. Time dependent photon mapping. In: *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2002. p. 135–144. ISBN 1-58113-534-3.

[18] MOON, J. T.; MARSCHNER, S. R. Simulating multiple scattering in hair using a photon mapping approach. In: *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*. New York, NY, USA: ACM, 2006. p. 1067–1074. ISBN 1-59593-364-6.

[19] HACHISUKA, T.; OGAKI, S.; JENSEN, H. W. Progressive photon mapping. In: *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*. New York, NY, USA: ACM, 2008. p. 1–8.

[20] KRIVANEK, J.; GAUTRON, P. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 11, n. 5, p. 550–561, 2005. ISSN 1077-2626. Member-Pattanaik,, Sumanta and Member-Bouatouch,, Kadi.

[21] KRIVANEK, J.; GAUTRON, P.; BOUATOUCH, K.; PATTANAIK, S. Improved radiance gradient computation. In: *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*. New York, NY, USA: ACM, 2005. p. 155–159. ISBN 1-59593-203-6.

[22] WARD, G. J.; HECKBERT, P. S. Irradiance gradients. In: *Eurographics workshop on Rendering*. [S.l.: s.n.], 1992.

[23] COOK, R. L.; PORTER, T.; CARPENTER, L. Distributed ray tracing. In: *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1984. p. 137–145. ISBN 0-89791-138-5.

[24] VEACH, E.; GUIBAS, L. J. Optimally combining sampling techniques for monte carlo rendering. In: *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1995. p. 419–428. ISBN 0-89791-701-4.