

# ReSTIR-C: Renderizador ReSTIR Experimental com Controle de Amostragem Temporal e Espacial em C++

GRINALDO OLIVEIRA, Universidade Federal da Bahia, Brasil

O algoritmo ReSTIR (*Spatiotemporal Reservoir Resampling*) fundamenta-se na técnica de *Reservoir Importance Sampling* (RIS) para abordar o problema de amostragem eficiente em cenários de iluminação com múltiplas fontes luminosas. Este trabalho implementa experimentalmente o algoritmo ReSTIR em C++ para validação empírica de seus fundamentos teóricos mediante um renderizador baseado em RIS com reutilização espaço-temporal de reservatórios, permitindo análise comparativa entre as variantes *biased* e *unbiased*. O framework experimental permite controle paramétrico do número de candidatos de amostragem e ativação condicional da reutilização espaço-temporal, possibilitando avaliação isolada dos componentes algorítmicos. A validação é conduzida mediante a renderização de uma superfície com reflectância Lambertiana com albedo variável em configuração de tabuleiro xadrez com esferas, sob iluminação por múltiplas fontes pontuais com distribuição controlada. A implementação reproduz os elementos fundamentais do ReSTIR, tais como a atualização probabilística de reservatórios via *weighted reservoir sampling*, combinação temporal e espacial de amostras através de operações de *merge* e estimativa de contribuição baseada em função-peso derivada da luminância espectral. Os experimentos demonstram a reprodutibilidade dos benefícios reportados na literatura, evidenciando redução significativa na variância do estimador Monte Carlo mantendo convergência para a solução de referência da equação de renderização.

CCS Concepts: • Computing methodologies → Ray tracing.

Additional Key Words and Phrases: Ray Tracing, ReSTIR

## ACM Reference Format:

Grinaldo Oliveira. 2025. ReSTIR-C: Renderizador ReSTIR Experimental com Controle de Amostragem Temporal e Espacial em C++. 1, 1 (July 2025), 9 pages.  
<https://doi.org/XXXXXX.XXXXXXX>

## 1 Introdução

A renderização fotorrealista em tempo real de cenas iluminadas por milhares de fontes dinâmicas constitui um desafio computacional crítico. Técnicas convencionais de *importance sampling* apresentam limitações de desempenho em cenários de alta cardinalidade luminosa, resultando em ruído significativo e custo computacional proibitivo para aplicações interativas.

Para abordar esses desafios, a técnica **ReSTIR** (*Spatiotemporal Reservoir Resampling*) foi apresentada em 2020 por [Bitterli et al. 2020], permitindo a renderização eficiente de iluminação direta dinâmica proveniente de milhões de fontes luminosas com um número mínimo de *rays* de sombra por *pixel*. Utilizando métodos de Monte Carlo, o algoritmo emprega reutilização espaço-temporal de amostras luminosas – tanto em quadros precedentes quanto entre *pixels* adjacentes – para ampliar a contagem efetiva de amostras e reduzir o ruído. Essa abordagem viabiliza a geração de imagens de alta qualidade combinadas com *denoisers* em altas taxas de *frame*, mesmo em cenas complexas com iluminação extensa.

Com o intuito de comprovar o arcabouço teórico do ReSTIR, o presente trabalho apresenta um *pipeline* de renderização em C++ que replica operadores fundamentais desta técnica, permitindo controle paramétrico do número de *rays* candidatos ( $M$ ) e iterações temporal ( $t$ ) e espacial ( $k$ ) em variantes *biased* e *unbiased*. Os resultados obtidos possibilitam a

Author's Contact Information: Grinaldo Oliveira, Universidade Federal da Bahia, Salvador, Bahia, Brasil, grinaldooliveira@ufba.br.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM XXXX-XXXX/2025/7-ART  
<https://doi.org/XXXXXX.XXXXXXX>

avaliação do desempenho do algoritmo de renderização aplicado a uma superfície com reflectância Lambertiana e albedo espacialmente variável, configurado em padrão xadrez com a inclusão de geometria adicional (esferas), iluminadas por  $N$  fontes pontuais distribuídas espacialmente.

Este artigo explora os fundamentos teóricos do ReSTIR (Seção 2), que estende o *Resampled Importance Sampling* (RIS) para amostras correlacionadas e domínios variados. São analisados trabalhos relacionados que apresentam aprimoramentos subsequentes (Seção 3), bem como potencialidades e limitações da técnica (Seção 4). Adicionalmente, é apresentado o **ReSTIR-C**, um renderizador com controle de amostragem temporal e espacial, cuja metodologia investiga a relação  $f(M, t, k)$  envolvendo o número de *rays* candidatos e a avaliação temporal e espacial em cenas configuradas (Seção 5). Por fim, apresentam-se conclusões e direcionamentos para trabalhos futuros.

## 2 Fundamentos da Técnica ReSTIR [Bitterli et al. 2020]

O ReSTIR utiliza a **reamostragem iterativa de amostras candidatas** armazenadas em estruturas de dados chamadas *reservatórios* para aproximar a amostragem por importância perfeita de forma computacionalmente eficiente. Para melhor compreendê-la, esta seção abordará um referencial teórico inicial sobre os métodos Monte Carlo, melhoramentos propostos por [Talbot et al. 2005] e algoritmos que implementam a abordagem espaço-temporal de amostragem.

### 2.1 Preliminares

Os métodos Monte Carlo constituem a base fundamental para simulação de transporte de luz em renderização baseada em física. Estes métodos estatísticos estimam integrais complexas através de amostragem aleatória, sendo especialmente adequados para resolver a equação de renderização:

$$L(y, \omega) = \int_A \rho(y, \bar{y} \leftrightarrow \bar{\omega}) L_e(x \rightarrow y) G(x \leftrightarrow y) V(x \leftrightarrow y) dA_x, \quad (1)$$

onde  $L(y, \omega)$  é a radiância refletida no ponto  $y$  na direção  $\omega$ ,  $\rho$  é a função de reflectância bidirecional (BRDF),  $L_e(x \rightarrow y)$  é a radiância emitida do ponto  $x$  para  $y$ ,  $G(x \leftrightarrow y)$  é o termo geométrico incluindo fatores de distância e cosseno,  $V(x \leftrightarrow y)$  é a função de visibilidade, e a integral é calculada sobre todas as superfícies emissivas  $A$ .

Na equação (1), descartando a direção de visualização ( $\omega$ ), o ponto de sombreamento ( $y$ ) e a área diferencial ( $dx$ ), a mesma pode ser simplificada para:

$$L = \int_A f(x) dx, \quad \text{onde } f(x) \equiv \rho(x) L_e(x) G(x) V(x). \quad (2)$$

Esta forma simplificada permite uma abordagem mais direta para os métodos Monte Carlo, agrupando todos os termos da função integranda em uma única função  $f(x)$ , pois permite o tratamento unificado de todos os componentes da equação de renderização durante o processo de amostragem.

### 2.2 Amostragem por Importância Monte Carlo

A amostragem por importância reduz a variância ao concentrar amostras em regiões que contribuem mais significativamente para o resultado final:

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)}, \quad (3)$$

onde  $f(x_i)$  é a função integranda e  $p(x_i)$  é a função densidade de probabilidade (PDF) usada para gerar as amostras. Na prática, a técnica de Monte Carlo utiliza-se de amostras aleatórias de fontes de luz, ajustando seus pesos apropriadamente para manter a correção estatística e aproximar-se ao valor correto da função integranda.

Todavia, embora eficazes, os métodos Monte Carlo tradicionais enfrentam desafios significativos:

- **Alta variância:** Especialmente em cenários com iluminação complexa.
- **Convergência lenta:** Requer muitas amostras para alcançar qualidade aceitável.
- **Distribuições sub-ótimas:** Dificuldade em amostrar próximo à distribuição ideal.

### 2.3 Contribuição Fundamental de Talbot

A técnica ReSTIR fundamenta-se no trabalho pioneiro de [Talbot et al. 2005], que introduziu o conceito de *Resampled Importance Sampling* (RIS). Talbot identificou uma limitação crucial nos métodos Monte Carlo tradicionais: Embora teoricamente se possa amostrar qualquer distribuição, na prática é inviável amostrar diretamente de distribuições complexas como o produto de múltiplos termos (BRDF × emissão × geometria × visibilidade).

**2.3.1 Inovação Conceitual de Talbot.** Talbot então propôs uma solução elegante em duas etapas (codificado no algoritmo 1):

- (1) **Geração de candidatos:** Produzir  $M$  amostras de uma distribuição  $p(x)$  fácil de amostrar.
- (2) **Reseleção ponderada:** Escolher uma única amostra final com probabilidades proporcionais aos pesos  $w(x) = \frac{\hat{p}(x)}{p(x)}$ .

O estimador RIS de Talbot é matematicamente expresso como:

$$\langle L \rangle_{ris}^{1,M} = \frac{f(y)}{\hat{p}(y)} \cdot \left( \frac{1}{M} \sum_{j=1}^M w(x_j) \right). \quad (4)$$

Neste caso, a iluminação é calculada dividindo-se a contribuição real da luz selecionada  $f(y)$  pela sua probabilidade ideal  $\hat{p}(y)$ . Com isto, multiplicasse o resultado por um fator de correção, que é a média dos pesos de importância de todos os candidatos gerados. Isto permite que se possa usar uma única amostra reselecionada de forma estatisticamente correta, mesmo quando esta não veio exatamente da distribuição ideal.

---

#### Algorithm 1 Resampled Importance Sampling

```

1: Entrada:  $M, q$ : número de candidatos a gerar ( $M \geq 1$ ) para o pixel  $q$ 
2: Saída: Amostra  $y$  e a soma dos pesos RIS  $\sum_{i=1}^M w(x_i)$ 
3: // Gerar propostas  $x = \{x_1, \dots, x_M\}$ 
4:  $x \leftarrow \emptyset$ 
5:  $w \leftarrow \emptyset$ 
6:  $wsum \leftarrow 0$ 
7: for  $i \leftarrow 1$  até  $M$  do
8:   gerar  $x_i \sim p$ 
9:    $x \leftarrow x \cup \{x_i\}$ 
10:   $w_i \leftarrow \hat{p}_q(x_i)/p(x_i)$ 
11:   $wsum \leftarrow wsum + w_i$ 
12:   $w \leftarrow w \cup \{w_i\}$ 
13: end for
14: // Selecionar dos candidatos  $x$ 
15: Calcular CDF normalizada  $C$  a partir de  $w$ 
16: sortear índice aleatório  $z \in [0, M]$  usando  $C$  para amostrar  $\propto w_z$ 
17:  $y \leftarrow x_z$ 
18: retornar  $y, wsum$ 

```

---

**2.3.2 Significado Teórico.** A contribuição de Talbot transcende a implementação técnica, estabelecendo um novo paradigma conceitual:

- **Separação de preocupações:** Dissocia a geração de amostras da distribuição alvo;
- **Aproximação progressiva:** À medida que  $M \rightarrow \infty$ , a distribuição efetiva converge para  $\hat{p}(x)$ ;
- **Flexibilidade algorítmica:** Permite incorporar múltiplas estratégias de amostragem via MIS (*Multiple Importance Sampling*).

### 2.4 Definição Formal do ReSTIR

O ReSTIR estende o RIS de Talbot através da reutilização espacial e temporal de amostras, sendo matematicamente expresso como:

$$\langle L \rangle_{restir}^{1,M} = \frac{f(y)}{\hat{p}(y)} \cdot W(x, z), \quad (5)$$

onde  $W(x, z)$  é o peso estocástico que incorpora informações de múltiplos reservatórios.

### 2.5 Conceito de Reservatório

Um reservatório é uma estrutura de dados compacta inspirada no *weighted reservoir sampling* de [Efraimidis and Spirakis 2006]:

$$\text{Reservatório} = \{y, wsum, M, W\}, \quad (6)$$

onde:

- $y$ : amostra selecionada atual.
- $wsum$ : soma dos pesos de todas as amostras processadas.
- $M$ : número total de candidatos vistos.
- $W$ : peso de correção estatística final.

### 2.6 Pipeline ReSTIR

Os Algoritmos 2 e 3 implementam o conceito de pipeline de reuso espacotemporal que transforma o processo tradicional de amostragem por importância em um sistema altamente eficiente capaz de renderizar milhões de luzes dinâmicas em tempo real. O pipeline é estruturado em cinco fases distintas que trabalham de forma sequencial para maximizar a qualidade das amostras, enquanto mantém a complexidade computacional em uma dimensão tratável.

A primeira fase constitui o núcleo da amostragem Monte Carlo tradicional, onde cada pixel  $q$  da imagem executa independentemente o algoritmo RIS. Esta etapa:

- **Gera  $M$  candidatos** por pixel usando amostragem por importância das fontes de luz;
- **Aplica weighted reservoir sampling** para selecionar uma amostra representativa;
- **Calcula o peso inicial  $W$**  usando a Equação (3).

Esta é a fase mais custosa do pipeline, com complexidade  $O(M)$  por pixel, onde  $M$  é tipicamente 32 candidatos. A paralelização em GPU permite que todos os pixels sejam processados simultaneamente.

A segunda fase (avaliação de visibilidade) implementa o conceito de *visibility reuse*, uma otimização crucial que:

- **Testa oclusão** da amostra selecionada  $y$  em cada reservatório;
- **Zera o peso  $W$**  para amostras ocluídas, efetivamente descartando-as;
- **Previne propagação** de amostras inválidas para pixels vizinhos.

O descarte de amostras ocluídas é matematicamente correto porque estas contribuem com valor zero para a integral de iluminação. Ao eliminar essas amostras antes do reuso espacial, o algoritmo evita contaminar reservatórios vizinhos com informação inútil.

A terceira fase (reuso temporal) explora a coerência entre frames consecutivos para aumentar dramaticamente o número efetivo de candidatos:

A função `pickTemporalNeighbor(q)` utiliza **motion vectors** para:

- Calcular a posição correspondente do pixel  $q$  no frame anterior;

**Algorithm 2** Pipeline Completo do ReSTIR

---

```

1: Entrada: Buffer de imagem contendo os reservatórios do frame anterior (prevFrameReservoirs)
2: Saída: Reservatórios do frame atual, cada um contendo uma amostra de luz selecionada e seu peso correspondente
3:
4: Fase 1: Geração Inicial Monte Carlo
5: for each pixel  $q$  na imagem do
6:    $reservoirs[q] \leftarrow RIS(q)$            ▷  $M$  candidatos via Monte Carlo
7: end for
8: Fase 2: Avaliação de Visibilidade
9: for each pixel  $q$  na imagem do
10:  if shadowed(reservoirs[ $q$ ]. $y$ ) then
11:     $reservoirs[q].W \leftarrow 0$              ▷ Rejeita amostras ocluídas
12:  end if
13: end for
14: Fase 3: Reuso Temporal
15: for each pixel  $q$  na imagem do
16:    $q' \leftarrow pickTemporalNeighbor(q)$           ▷ Motion vectors
17:    $reservoirs[q] \leftarrow combineReservoirs(q, reservoirs[q],$ 
18:     prevFrame[ $q'$ ])
19: end for
20: Fase 4: Reuso Espacial Iterativo
21: for  $i \leftarrow 1$  to  $n$  do
22:   for each pixel  $q$  na imagem do
23:      $Q \leftarrow pickSpatialNeighbors(q)$ 
24:      $R \leftarrow \{reservoirs[q'] | q' \in Q\}$ 
25:      $reservoirs[q] \leftarrow combineReservoirs(q, reservoirs[q], R)$ 
26:   end for
27: end for
28: Fase 5: Renderização Final
29: for each pixel  $q$  na imagem do
30:    $Image[q] \leftarrow shadePixel(reservoirs[q], q)$ 
31: end for
32: return reservoirs

```

---

**Algorithm 3** Weighted Reservoir Sampling Estendido

---

```

1: function UPDATERESERVOIR( $r, x_i, w_i$ )
2:    $r.wsum \leftarrow r.wsum + w_i$ 
3:    $r.M \leftarrow r.M + 1$ 
4:   if random() <  $\frac{w_i}{r.wsum}$  then
5:      $r.y \leftarrow x_i$                          ▷ Aceita nova amostra
6:   end if
7: end function
8: function COMBINERESERVOIRS( $r_1, r_2, \dots, r_k$ )
9:    $s \leftarrow emptyReservoir()$ 
10:  for cada  $r_i$  do
11:     $s.update(r_i.y, \hat{p}_q(r_i.y) \cdot r_i.W \cdot r_i.M)$ 
12:  end for
13:   $s.W \leftarrow \frac{1}{\hat{p}_q(s.y)} \left( \frac{1}{s.M} s.wsum \right)$ 
14:  return s
15: end function

```

---

- Compensar movimentos de câmera e objetos dinâmicos;
- Manter correspondência espacial precisa entre frames.

A função *combineReservoirs* implementa o algoritmo de combinação estatisticamente correto:

$$\text{Peso combinado} = \hat{p}_q(r.y) \cdot r.W \cdot r.M, \quad (7)$$

$$\text{Novo peso} = \frac{1}{\hat{p}_q(s.y)} \left( \frac{1}{s.M} \cdot s.wsum \right). \quad (8)$$

Após o reuso temporal, cada pixel efetivamente possui  $\approx 2M$  candidatos:  $M$  do frame atual mais  $M_{prev}$  do frame anterior.

A quarta fase (reuso espacial iterativo) é o componente mais inovador do ReSTIR, implementando múltiplas rodadas de combinação de reservatórios vizinhos. A função *pickSpatialNeighbors*( $q$ ) seleciona  $k$  pixels vizinhos (tipicamente  $k = 5$ ) usando:

- **Amostragem aleatória** em um raio de 30 pixels.
- **Sequências de baixa discrepância** para distribuição uniforme.
- **Filtragem geométrica** para rejeitar vizinhos com geometria muito diferente.

A fase final converte os reservatórios otimizados em valores de cor:

$$\text{Cor do pixel} = f_q(r.y) \cdot r.W, \quad (9)$$

onde:

- $f_q(r.y)$ : Função de contribuição de iluminação (BRDF × emissão × geometria × visibilidade).
- $r.W$ : Peso de correção estatística final.
- $r.y$ : Amostra de luz selecionada.

## 2.7 Complexidade Computacional

O ReSTIR mantém complexidade de memória  $O(1)$  por pixel e complexidade temporal:

$$\text{Geração Inicial : } O(M), \quad (10)$$

$$\text{Reuso Temporal : } O(1), \quad (11)$$

$$\text{Reuso Espacial : } O(nk), \quad (12)$$

$$\text{Total : } O(M + nk), \quad (13)$$

para um número efetivo de candidatos da ordem de  $k^n M$ , representando uma melhoria exponencial na relação qualidade/custo.

## 2.8 Tratamento de Bias e Correção MIS

Na seção 2.6, foi apresentado um algoritmo prático para o reuso de computações, tanto espacial quanto temporalmente, que melhora significativamente a qualidade do RIS com baixo custo computacional.

No entanto, ignorou-se um detalhe importante: cada pixel utiliza um domínio de integração e uma distribuição alvo distintos, e o reuso de amostras provenientes de pixels adjacentes pode, potencialmente, introduzir viés. Isso ocorre porque a função densidade de probabilidade (PDF) das amostras, após o reamostramento, varia de pixel para pixel devido às diferentes distribuições alvo. O RIS convencional não foi projetado para acomodar a mistura de amostras candidatas provenientes de diferentes PDFs, como ocorre durante o reuso, e ignorar esse fato pode resultar em ruído e viés.

Para manter o não-enviesamento quando combinando amostras de diferentes distribuições, o ReSTIR utiliza correção baseada em *Multiple Importance Sampling* (MIS):

$$W(x, z) = \frac{1}{\hat{p}(x_z)} \left( m(x_z) \sum_{i=1}^M w_i(x_i) \right), \quad (14)$$

onde  $m(x_z) = \frac{p_z(x_z)}{\sum_{j=1}^M p_j(x_z)}$  é o peso de *balance heuristic*, garantindo que  $\sum_{i \in Z(y)} m(x_i) = 1$ . Neste caso, o procedimento de combinação de múltiplos reservatórios é atualizado para refletir esta equação, conforme mostrado no algoritmo 4.

## 3 Trabalhos Relacionados

A Tabela 1 sintetiza os principais aprimoramentos da técnica ReSTIR, destacando uma breve descrição das inovações introduzidas em cada variante. Esses dados evidenciam não apenas a versatilidade do método, mas também sua escalabilidade para diferentes tipos de cenas e requisitos de qualidade, aspectos fundamentais para aplicações interativas.

Tabela 1. Sumário de trabalhos relacionados com aprimoramentos da técnica ReSTIR.

Aprimoramento	Descrição do Aprimoramento
ReSTIR GI ( <i>Global Illumination</i> ) [Ouyang et al. 2021]	Extensão para iluminação global difusa. Combina reservatórios com fótons para capturar iluminação indireta, sem suporte eficiente para cáusticas.
Volumetric ReSTIR [Lin et al. 2021]	Adaptação para meios participativos (fumaça, neblina). Combina reamostragem espaço-temporal com estimativas rápidas de transmissão, permitindo espalhamento múltiplo.
ReGIR (Grid-Based Reservoirs) [Boksansky et al. 2021]	Armazena reservatórios em grades espaciais. Reutiliza amostras para raios secundários e iluminação volumétrica, com baixo custo de memória (16 MB).
ReSTIR PT ( <i>Path Tracing</i> ) [Lin et al. 2022]	Generalização para path tracing. Reutiliza amostras de caminhos completos (até 3 saltos), incluindo superfícies e volumes.
ReSTIR para Renderização Diferenciável [Wang et al. 2023]	Aplicação em otimização inversa. Reutiliza amostras temporalmente entre iterações de gradiente, reduzindo ruído em estimativas de parâmetros de cena.
ReSTIR FG ( <i>Final Gathering</i> ) [Kern et al. 2024]	Foco em cáusticas. Combina <i>photon mapping</i> com reservatórios para "final gathering" eficiente, capturando efeitos complexos como reflexão/refração em vidro.
Area ReSTIR [Zhang et al. 2024]	Estende a reamostragem para coordenadas de subpixel e lente (espaço 4D). Elimina <i>aliasing</i> e permite profundidade de campo em tempo real, mesmo com geometria complexa.

**Algorithm 4** Combinação não enviesada de múltiplos reservatórios

```

1: Entrada: Reservatórios  $r_i$  e os pixels  $q_i$  de origem
2: Saída: Um reservatório combinado não enviesado
   Neste caso, o algoritmo 3 utilizado para combinação de múltiplos reservatórios é alterada para o algoritmo 4.
3: function COMBINERESERVOIRSUNBIASED( $q, r_1, r_2, \dots, r_k, q_1, \dots, q_k$ )
4:    $s \leftarrow$  reservatório vazio
5:   for cada  $r \in \{r_1, \dots, r_k\}$  do
6:      $s.\text{update}(r.y, \hat{p}_q(r.y) \cdot r.W \cdot r.M)$ 
7:   end for
8:    $s.M \leftarrow r_1.M + r_2.M + \dots + r_k.M$ 
9:    $Z \leftarrow 0$ 
10:  for cada  $q_i \in \{q_1, \dots, q_k\}$  do
11:    if  $\hat{p}_{q_i}(s.y) > 0$  then
12:       $Z \leftarrow Z + r_i.M$ 
13:    end if
14:  end for
15:   $m \leftarrow \frac{1}{Z}$ 
16:   $s.W \leftarrow \frac{1}{\hat{p}_q(s.y)} \cdot (m \cdot s.wsum)$            ▷ Equação (14)
17:  retornar  $s$ 
18: end function

```

## 4 Potencialidades e Limitações do ReSTIR

O ReSTIR apresenta diversas potencialidades e algumas limitações, conforme relatado na literatura técnica.

### 4.1 Potencialidades do ReSTIR

- Desempenho excepcional em cenas complexas:** O ReSTIR foi projetado para renderizar iluminação direta em tempo real mesmo em cenas com milhares ou milhões de luzes dinâmicas. Ele alcança velocidades de  $6\times$  a  $60\times$  superiores às técnicas anteriores para o mesmo erro, podendo chegar a  $35\times$  a  $65\times$  mais rápido usando um modo enviesado, o que é especialmente relevante para aplicações interativas [Bitterli et al. 2020].
- Redução significativa de ruído:** Ao reutilizar amostras espacial e temporalmente (entre pixels vizinhos e quadros anteriores), o ReSTIR reduz drasticamente a variância das imagens, permitindo

resultados visuais de alta qualidade mesmo com poucos raios por pixel - vide Figura 1. Isso potencializa o desempenho de denoisers, que conseguem entregar imagens mais limpas quando recebem amostras de maior qualidade [Bitterli et al. 2020].

- Versatilidade e extensibilidade:** Desde sua introdução para iluminação direta, o ReSTIR foi estendido para iluminação global (ReSTIR GI), *path tracing* completo (ReSTIR PT), renderização volumétrica (Volumetric ReSTIR) e integração em domínios de área (Area ReSTIR), além de aplicações em renderização diferenciável e inversa. Essa capacidade de adaptação a diferentes domínios e problemas o torna uma ferramenta central para renderização moderna [Lin et al. 2021; Ouyang et al. 2021; Wang et al. 2023; Zhang et al. 2024].
- Baixa complexidade estrutural:** O método utiliza apenas estruturas simples (reservatórios de amostras por pixel), dispensando a necessidade de construir e manter estruturas de dados complexas, o que simplifica a implementação e manutenção do algoritmo em pipelines gráficos modernos [Bitterli et al. 2020].
- Interatividade e escalabilidade:** O ReSTIR permite renderização interativa de cenas com milhões de triângulos emissiones em menos de 50 ms por quadro em GPUs modernas, mesmo com orçamentos de poucos raios por pixel, sendo altamente escalável para diferentes tamanhos e complexidades de cena [Bitterli et al. 2020].

### 4.2 Limitações

- Introdução de viés em alguns modos:** O modo enviesado do ReSTIR, embora mais rápido e menos ruidoso, pode resultar em perda de energia (imagens mais escuras) e artefatos próximos a descontinuidades geométricas ou em regiões de alta frequência, especialmente quando há grande diferença entre as distribuições de probabilidade das amostras reutilizadas [Bitterli et al. 2020; Zhang et al. 2024].
- Dificuldade de reutilização em domínios complexos:** Em cenas com detalhes geométricos muito finos, normais de alta frequência ou mudanças abruptas (como cabelo, folhagem ou superfícies altamente detalhadas), a reutilização espacial e temporal pode ser menos eficaz, levando à degradação da qualidade ou à

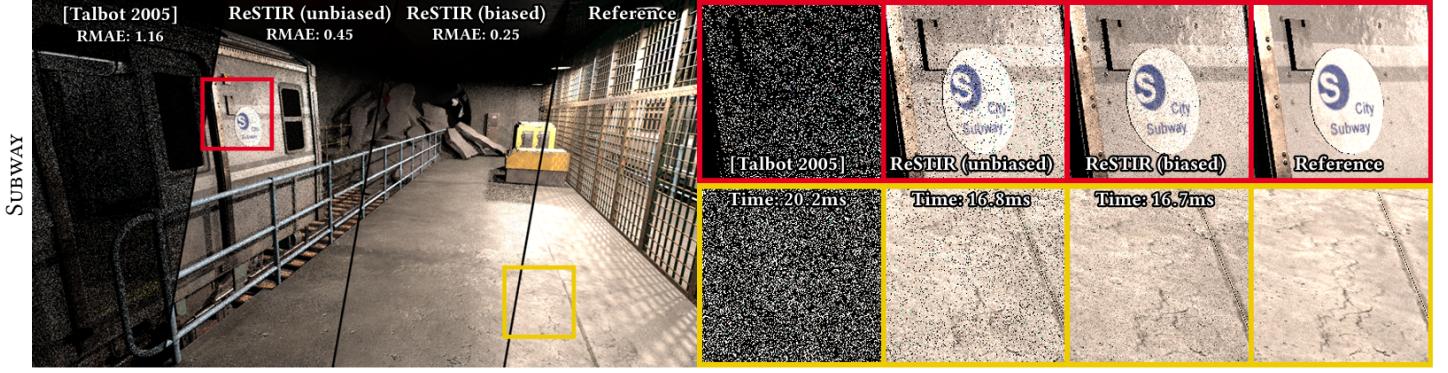


Fig. 1. Comparação de renderizações com tempo aproximadamente igual de uma implementação streaming de [Talbot et al. 2005] com Restir enviesado e não-enviesado. Uma referência convergida também é mostrada para comparação. Subway possui 23.452 triângulos emissores texturizados.

necessidade de reinicializar o processo de agregação de amostras [Zhang et al. 2024].

- **Dependência de heurísticas para rejeição de amostras:** O reuso de amostras de pixels vizinhos ou quadros anteriores pode, em certos casos, prejudicar a qualidade se as distribuições de probabilidade diferirem muito. Isso exige heurísticas para rejeitar amostras inadequadas, o que pode aumentar a complexidade do pipeline e impactar a robustez do método [Bitterli et al. 2020; Lin et al. 2021].
- **Limitações em domínios volumétricos e de área:** Embora as variantes volumétricas e de área do ReSTIR tenham ampliado seu alcance, ainda existem desafios relacionados ao custo computacional e à manutenção do viés sob movimento rápido ou em volumes de alta resolução, além da necessidade de balancear a granularidade das grades espaciais para evitar perda de detalhes ou aumento de ruído [Lin et al. 2021; Zhang et al. 2024].

## 5 Análise Experimental: Implementação e Validação do ReSTIR-C

A implementação do código-fonte do ReSTIR-C foi conduzida com o apoio de sistemas de inteligência artificial, especificamente Claude.AI e Perplexity.AI, empregados como ferramentas computacionais para auxiliar na interpretação, transposição e depuração dos algoritmos descritos por [Bitterli et al. 2020]. Tais ambientes foram integrados ao fluxo de desenvolvimento com o propósito de assegurar maior aderência à formulação teórica original, promovendo a precisão semântica na estruturação dos métodos de RIS, bem como no tratamento das variáveis críticas envolvidas na reutilização espacial e temporal. O uso dessas plataformas contribuiu significativamente para o refinamento iterativo da lógica algorítmica, assegurando fidelidade conceitual e rigor na implementação computacional.

Os experimentos foram conduzidos em uma máquina equipada com processador Intel(R) Core(TM) i7-4770 a 3.40 GHz, 12 GB de memória RAM e sistema operacional Windows 10.

### 5.1 Configuração do Cenário Experimental (*baseline*)

O cenário experimental (Figura 2) consiste, respectivamente, em uma superfície planar com reflectância Lambertiana, albedo espacialmente variável, configurada como padrão xadrez e inclusão de esferas, para maximizar a diversidade espectral das interações luz-superfície. A configuração de iluminação emprega um conjunto de  $N = 7$  fontes pontuais com distribuição espacial estratificada, garantindo cobertura angular completa do hemisfério de iluminação.

Para a construção do cenário experimental de referência, foi empregada a renderização baseada exclusivamente no método de

Monte Carlo com amostragem direta, utilizando uma configuração de 1000 raios candidatos por pixel para a composição da imagem. O tempo necessário para a geração do arquivo no formato PPM foi de 50,9 segundos, valor que será adotado como referência de desempenho (*baseline*) para as comparações subsequentes. Este *baseline* será utilizado na avaliação relativa do algoritmo ReSTIR, tanto em sua versão com viés (*biased*) quanto na configuração não enviesada (*unbiased*), com base em métricas quantitativas de performance, como tempo de renderização e variância de luminância.

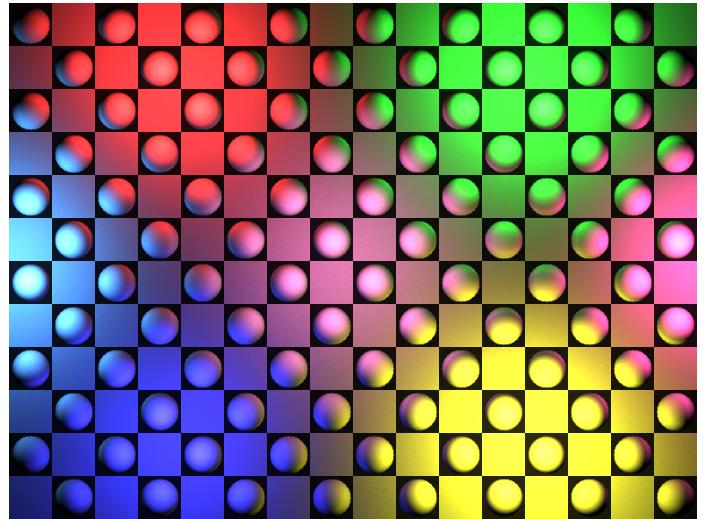


Fig. 2. Configuração do cenário experimental (*baseline*) configurado com uma superfície planar de reflectância Lambertiana com albedo espacialmente variável, com a inclusão de esferas, configurada como padrão xadrez para maximizar a diversidade espectral das interações de 9 fontes de luz-superfície.

### 5.2 Pipeline de Renderização ReSTIR-C

Conforme ilustrado na Figura 3, o pipeline de renderização ReSTIR implementado no código segue uma arquitetura sequencial estruturada em quatro etapas principais, cada uma executada de forma independente para cada pixel da imagem.

A primeira etapa consiste na Amostragem Inicial RIS, onde são gerados  $M$  candidatos por pixel através do parâmetro MAX\_CANDIDATES. Durante esta fase, cada pixel  $q$  executa o algoritmo de streaming RIS implementado na função RIS( $q$ ), gerando candidatos através de amostragem uniforme sobre as superfícies emissivas da cena. Os candidatos são ponderados pela função de importância  $\hat{p}_q(x) = \rho(x) \cdot L_e(x) \cdot G(x)$  e processados através do *weighted*

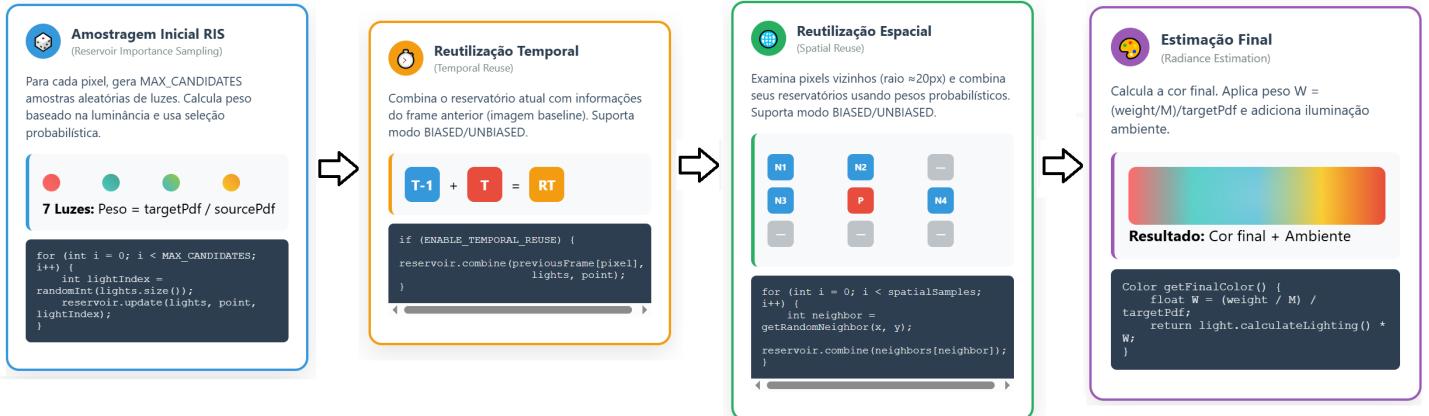


Fig. 3. Pipeline de Renderização do Restir-C. (I) Amostragem inicial com a geração de  $k$  candidatos por pixel. (II) Execução do procedimento de reutilização temporal com a combinação do reservatório atual com informações do frame anterior com variância *biased* ou *unbiased*. (III) Execução do procedimento de reutilização espacial de *pixels* vizinhos com variância *biased* ou *unbiased*. (IV) Estimação final da cor do *pixel*.

*reservoir sampling* (Algoritmo 3), resultando em um reservatório contendo uma única amostra selecionada estocasticamente, juntamente com o peso acumulado e a contagem de candidatos processados.

A segunda etapa implementa a Reutilização Temporal através da função `combineReservoirs`, onde o reservatório atual de cada pixel é combinado com o reservatório correspondente do frame anterior (neste caso, o arquivo PPM do *baseline* será utilizado para compor estes reservatórios anteriores). O código utiliza a estratégia de *backprojection* para determinar a correspondência temporal entre pixels, aplicando uma limitação no crescimento do parâmetro  $M$  do reservatório temporal para no máximo  $20 \times M$  do reservatório atual, conforme implementado na condição `if (tempReservoir.M > 20 * reservoir.M)`. Esta limitação previne o crescimento descontrolado da influência temporal e mantém a estabilidade do algoritmo ao longo da sequência de frames.

A terceira etapa executa a Reutilização Espacial através da função `spatialReuse`, que combina o reservatório de cada pixel com reservatórios de pixels vizinhos selecionados aleatoriamente. O código implementa uma estratégia de seleção de vizinhos baseada em amostragem estocástica dentro de um raio de 20 pixels, utilizando sequências de baixa discrepância para garantir distribuição uniforme. Para o modo *biased*, são selecionados  $k = 4$  vizinhos com  $n = 1$  iterações de reutilização espacial, enquanto o modo *unbiased* utiliza  $k = 3$  vizinhos com  $n = 1$  iteração para reduzir o custo computacional adicional das avaliações de PDF necessárias para manter a correção estatística.

A quarta e última etapa realiza o Cálculo da Cor Final através da função `getFinalColor`, que aplica o peso RIS corrigido à contribuição luminosa da amostra selecionada. Para o modo *biased*, o peso é calculado diretamente como  $W = \frac{(\text{weight}/M)}{\text{targetPpdf}}$ , enquanto o modo *unbiased* aplica correções baseadas em Multiple Importance Sampling (MIS), que responderá as contribuições considerando as diferenças entre as distribuições de pixels distintos. O resultado final incorpora uma componente de iluminação ambiente mínima ( $\text{albedo} \times 0.005$ ) antes da normalização e clamp dos valores de cor. Este pipeline mantém complexidade computacional linear em relação ao número de candidatos iniciais  $M$ , com custo adicional  $O(nk)$  para a reutilização espacial, onde  $n$  representa o número de iterações e  $k$  o número de vizinhos selecionados. A implementação preserva as propriedades estatísticas do estimador Monte Carlo através do uso consistente de *weighted reservoir sampling* em todas as etapas de combinação, garantindo que a distribuição

de probabilidade efetiva permaneça proporcional à função de importância agregada dos reservatórios combinados.

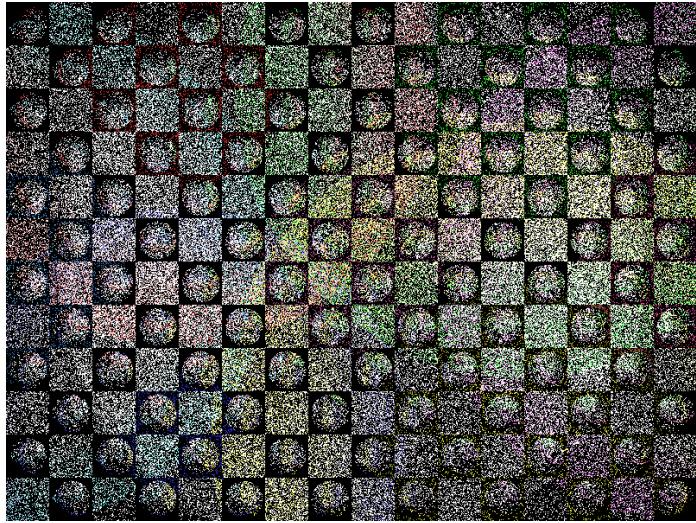
### 5.3 Análise de Resultados

Com base nos resultados apresentados na Figura 4 do documento anexo, a implementação ReSTIR-C demonstra comportamentos distintos entre as três variantes algorítmicas testadas: RIS puro, ReSTIR *biased* e ReSTIR *unbiased*, cada uma avaliada com diferentes números de candidatos ( $C=1, 10, 50, 100$ ) em comparação com uma cena *baseline* de referência.

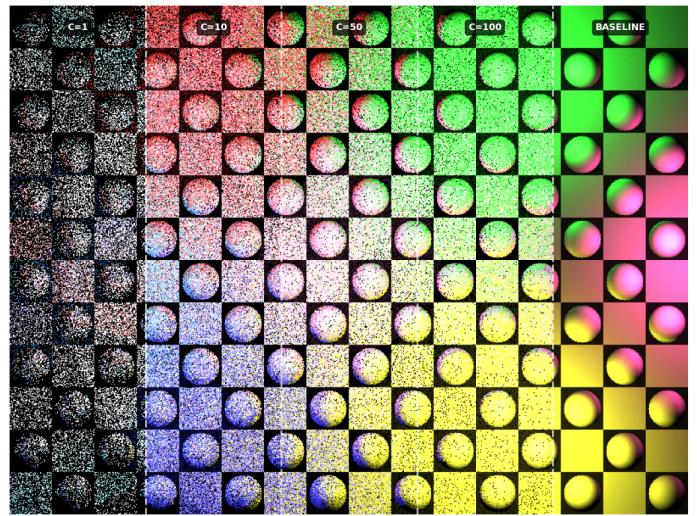
**RIS Puro (Figuras 4a e 4b):** A implementação do algoritmo RIS sem reutilização espacial ou temporal demonstra comportamento de convergência gradual conforme o aumento do número de candidatos. Os resultados evidenciam melhoria progressiva da qualidade visual, confirmando o comportamento esperado do streaming RIS implementado no código. Mesmo com  $C=100$  candidatos, a imagem mantém ruído perceptível significativo, validando a análise teórica de que o RIS puro apresenta limitações de eficiência em cenários de baixa amostragem. A implementação demonstra custo computacional linear  $O(M)$  esperado, onde  $M$  representa o número de candidatos, conforme implementado na função `RIS(q)`.

**ReSTIR Biased com Reutilização Espacial e Temporal (Figuras 4c e 4d):** A variante *biased* apresenta melhorias substanciais em relação ao RIS puro, demonstrando redução dramática de ruído mesmo com  $C=1$  candidato. A qualidade visual supera significativamente o RIS puro com múltiplos candidatos, evidenciando a eficácia da reutilização implementada na função `spatialReuse()`. Observa-se tendência de suavização excessiva que pode mascarar detalhes finos, característica do viés introduzido pela simplificação do cálculo de pesos durante a combinação de reservatórios. Esta suavização resulta em imagens visualmente mais agradáveis em baixas amostragens, mas com potencial perda de fidelidade física em regiões de transição geométrica.

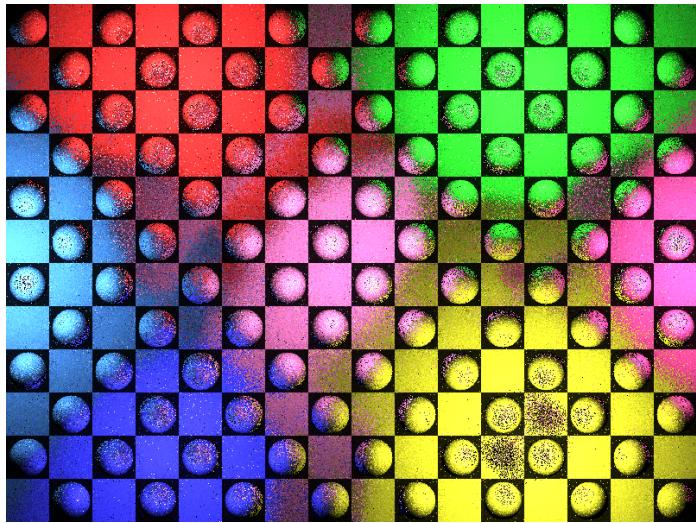
**ReSTIR Unbiased com Reutilização Espacial e Temporal (Figuras 4e e 4f):** A implementação *unbiased* corrigida apresenta características equilibradas, mantendo maior fidelidade aos detalhes da cena *baseline*, especialmente em regiões de transição geométrica. Este comportamento resulta da correção MIS implementada. A variante apresenta ruído ligeiramente superior ao modo *biased*, mas com distribuição mais uniforme e menor incidência de artefatos estruturais. A progressão de  $C=1$  para  $C=100$  demonstra convergência consistente sem perda de energia luminosa, confirmando a eficácia das correções implementadas no



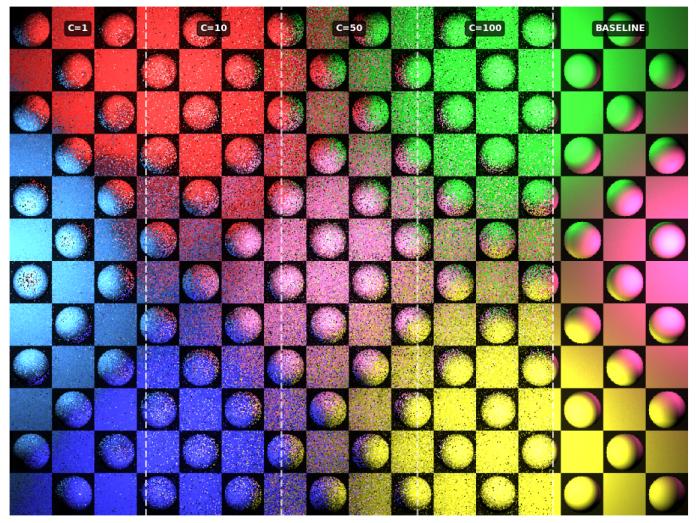
(a) Resultados com utilização de apenas 1 raio candidato utilizando a função *RIS*.



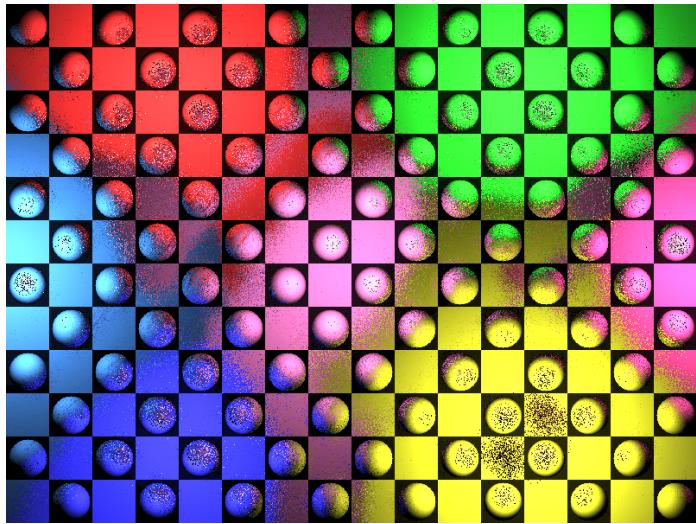
(b) Resultados com aplicação de função *RIS* em cenários graduais de 1, 10, 50 e 100 raios candidatos comparados à cena *baseline*.



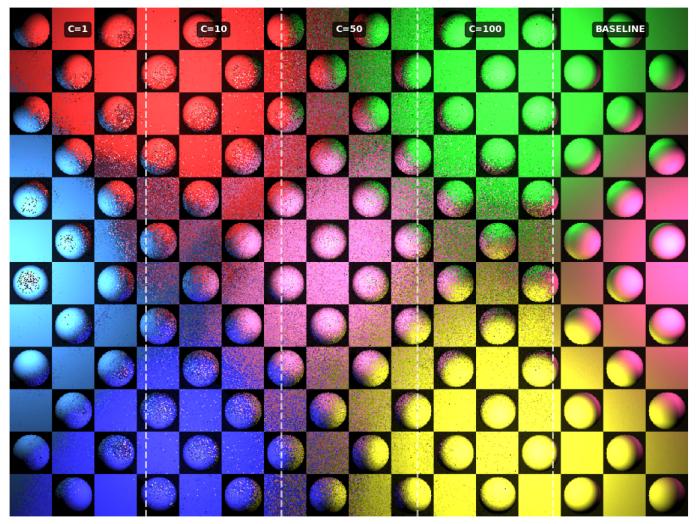
(c) Resultados com reutilização temporal e espacial em apenas 1 raio candidato utilizando a variação *biased*.



(d) Resultados *biased* com reutilização temporal e espacial em cenários graduais de 1, 10, 50 e 100 raios candidatos comparados à cena *baseline*.

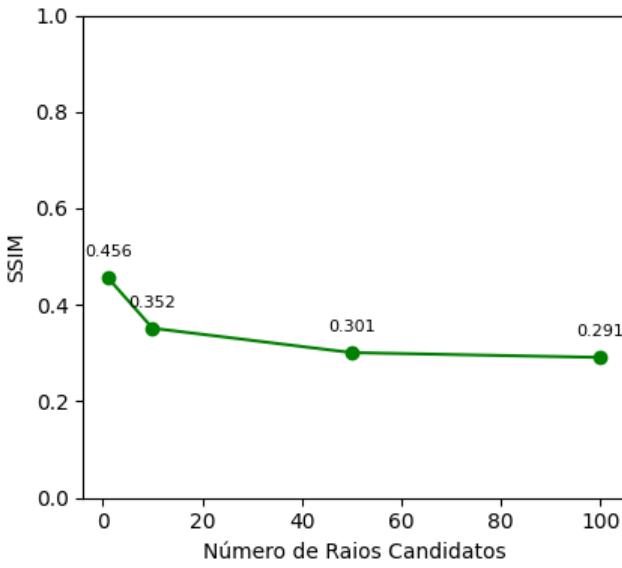


(e) Resultados com reutilização temporal e espacial em apenas 1 raio candidato utilizando a variação *unbiased*.

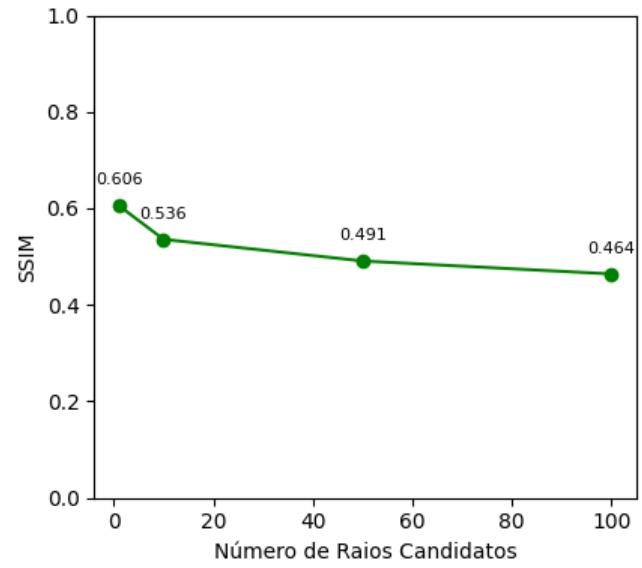


(f) Resultados *unbiased* com reutilização temporal e espacial em cenários graduais de 1, 10, 50 e 100 raios candidatos comparados à cena *baseline*.

Fig. 4. Comparativo de resultados nos cenários *RIS*, *biased* e *unbiased* (com reutilização temporal e espacial).



(a) Evolução do índice SSIM em função do número de raios candidatos com variação *biased*.



(b) Evolução do índice SSIM em função do número de raios candidatos com variação *unbiased*.

Fig. 5. Evolução do índice SSIM em função do número de raios candidatos utilizados na renderização com variação *biased* e *unbiased*.

código. A preservação de detalhes e a ausência de escurecimento artificial representam vantagens significativas para aplicações que exigem fidelidade física, justificando o custo computacional adicional das avaliações de PDF necessárias para manter a correção estatística.

**Análise Comparativa de Desempenho - Avaliação de Similaridade:** O *Structural Similarity Index* (SSIM) é uma métrica perceptual projetada para quantificar a similaridade entre duas imagens com base na maneira como o sistema visual humano percebe alterações estruturais. Matematicamente, o SSIM entre dois blocos de imagem  $x$  e  $y$  é definido como:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (15)$$

onde  $\mu_x$  e  $\mu_y$  são as médias locais de luminância,  $\sigma_x^2$  e  $\sigma_y^2$  as variâncias locais,  $\sigma_{xy}$  a covariância entre  $x$  e  $y$ , e  $C_1$  e  $C_2$  são constantes de estabilização que evitam divisões por zero em regiões de baixa variância. A métrica assume valores no intervalo  $[0, 1]$ , em que 1 representa similaridade estrutural perfeita. A avaliação global do SSIM é realizada pela média dos valores locais obtidos ao longo de toda a imagem, tipicamente em janelas deslizantes.

Nos experimentos conduzidos, foram avaliadas imagens renderizadas com a técnica ReSTIR, tanto na forma enviesada quanto não-enviesada, utilizando diferentes quantidades de raios candidatos (1, 10, 50 e 100). Os valores de SSIM foram calculados em relação a uma imagem de referência gerada com maior acurácia (Figuras 5a e 5b).

A implementação *unbiased* demonstra consistentemente valores SSIM superiores em todas as configurações de raios testadas, com diferencial médio de aproximadamente 0,15-0,17 pontos. Esta superioridade quantitativa reflete a capacidade do estimador *unbiased* de preservar características estruturais da imagem de referência através da correção matemática dos pesos de reamostragem.

O comportamento decrescente observado em ambas as curvas sugere que o aumento no número de raios candidatos não necessariamente resulta em melhoria da similaridade estrutural quando avaliada pela métrica SSIM. Este fenômeno pode ser atribuído à

natureza estocástica do processo de reamostragem espaciotemporal, onde a incorporação de amostras adicionais pode introduzir variabilidade estrutural que impacta negativamente a correlação com a imagem de referência.

A estabilização assintótica observada em ambas as implementações a partir de 50 raios indica que o algoritmo atinge um ponto de equilíbrio entre qualidade de amostragem e preservação estrutural, sugerindo que configurações com número excessivo de raios candidatos podem não justificar o custo computacional adicional para aplicações que priorizam fidelidade estrutural.

**Análise Comparativa de Desempenho - Tempo de Renderização:** O *speedup* é uma métrica fundamental para avaliação de desempenho computacional que quantifica o ganho de velocidade obtido por uma implementação otimizada em relação a uma implementação de referência. Matematicamente, o speedup é definido pela relação:

$$\text{Speedup} = \frac{T_{\text{baseline}}}{T_{\text{otimizado}}} \quad (16)$$

onde  $T_{\text{baseline}}$  representa o tempo de execução do algoritmo de referência e  $T_{\text{otimizado}}$  o tempo de execução da implementação otimizada. Valores de speedup superiores a 1,0 indicam melhoria de desempenho, enquanto valores inferiores denotam degradação. Com base nos resultados apresentados na figura 6, observa-se um comportamento decrescente consistente para ambas as implementações do algoritmo ReSTIR ao longo do incremento no número de raios candidatos. A análise da evolução temporal revela que a implementação **biased** mantém consistentemente valores de speedup superiores à versão **unbiased** em todas as configurações testadas, demonstrando uma vantagem de desempenho que se mantém proporcional ao longo de todo o espectro de candidatos avaliados.

Particularmente notável é a manutenção da diferença relativa entre as duas implementações ao longo de toda a progressão, sugerindo que o overhead da correção de bias na versão **unbiased** representa uma penalidade computacional constante e proporcional, independentemente da complexidade da amostragem. Esta observação é crucial para aplicações que devem balancear precisão matemática contra desempenho temporal, pois indica que a

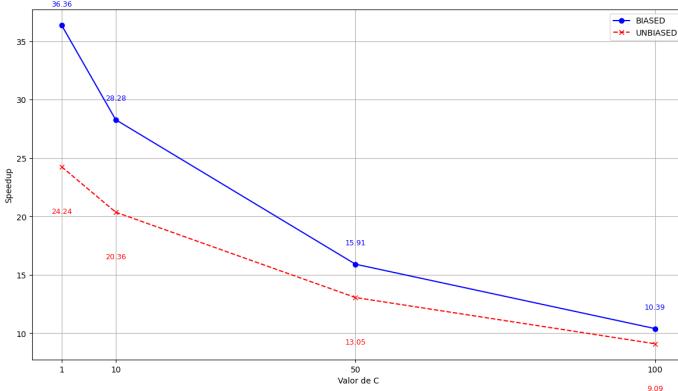


Fig. 6. Speedup das variações BIASED e UNBIASED ao longo da evolução da quantidade de raios.

escolha entre as implementações deve considerar não apenas o custo absoluto, mas também a escalabilidade relativa em cenários de alta complexidade amostral.

A análise da evolução dos números demonstra que, embora ambas as implementações mantenham speedups significativos mesmo nas configurações mais computacionalmente intensivas, a taxa de degradação sugere a existência de um número ótimo de raios candidatos que maximiza a relação custo-benefício entre qualidade de renderização e eficiência computacional.

Por outro lado, a figura 7 complementa a análise quantitativa ao apresentar os tempos absolutos de renderização, revelando comportamento inversamente proporcional ao speedup. A análise da progressão temporal demonstra que ambas as implementações do algoritmo ReSTIR apresentam crescimento linear controlado nos tempos de execução conforme o número de raios candidatos aumenta, confirmando a relação inversa esperada com os valores de speedup observados anteriormente.

A linearidade observada nas curvas temporais é particularmente significativa, pois indica que o algoritmo ReSTIR apresenta escalabilidade computacional previsível e controlada. Esta característica é fundamental para aplicações em tempo real, onde a previsibilidade do desempenho é crucial para manter taxas de quadros estáveis. A ausência de crescimento exponencial ou comportamento irregular nos tempos de execução demonstra que a arquitetura do algoritmo é adequada para implementações em GPU com paralelização eficiente.

## 6 Conclusão

A implementação experimental do ReSTIR-C validou empiricamente os fundamentos teóricos do algoritmo ReSTIR, demonstrando redução significativa da variância do estimador Monte Carlo em cenários de iluminação direta com múltiplas fontes luminosas. Os experimentos comparativos entre configurações com e sem reutilização espacial evidenciaram a eficácia da técnica de reamostragem espaço-temporal proposta por Bitterli et al. [2020]. A análise quantitativa revelou comportamentos distintos entre as variantes algorítmicas. A avaliação através do índice SSIM confirmou a superioridade estrutural da implementação *unbiased*, com diferencial médio de 0,15-0,17 pontos em todas as configurações testadas. Por outro lado, a análise de desempenho demonstrou que a medida de *speedup* para a variante *biased*, juntamente com o tempo de renderização significativamente reduzido, compensa efetivamente o viés introduzido pelo algoritmo. Esta versão manteve consistentemente valores de *speedup* superiores à implementação

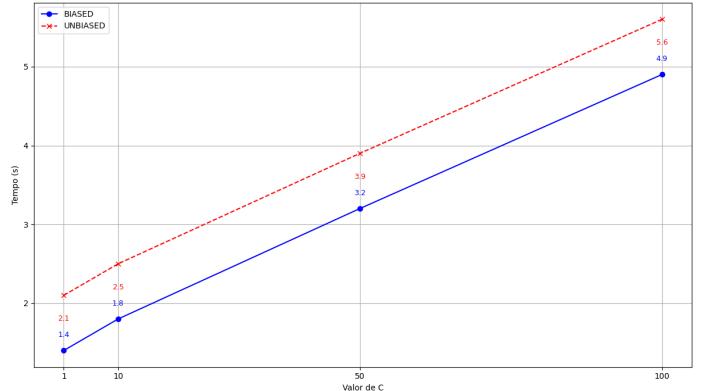


Fig. 7. Tempo de execução das variações BIASED e UNBIASED ao longo da evolução da quantidade de raios.

*unbiased* em todas as configurações testadas, apresentando vantagem de desempenho que se mantém proporcional ao longo de todo o espectro de candidatos avaliados.

É importante frisar que a implementação atual do REST-C limitou-se à renderização de superfícies Lambertianas sob iluminação direta. Extensões futuras podem incluir a incorporação de materiais com BRDF complexa, otimização de estruturas de dados espaciais para aceleração da reutilização espacial e validação quantitativa através de métricas de erro estatístico comparativo com métodos de referência.

## Referências

- Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4, Article 148 (Aug. 2020), 17 pages. doi:10.1145/3386569.3392481
- Jakub Boksansky, Paula Jukarainen, and Chris Wyman. 2021. *Rendering Many Lights with Grid-Based Reservoirs*. Apress, Berkeley, CA, 351–365. doi:10.1007/978-1-4842-7185-8\_23
- Pavlos S. Efraimidis and Paul G. Spirakis. 2006. Weighted random sampling with a reservoir. *Inform. Process. Lett.* 97, 5 (2006), 181–185. doi:10.1016/j.iplet.2005.11.003
- René Kern, Felix Brüll, and Thorsten Grosch. 2024. ReSTIR FG: Real-Time Reservoir Resampled Photon Final Gathering. In *Eurographics Symposium on Rendering*, Eric Haines and Elena Garces (Eds.). The Eurographics Association. doi:10.2312/sr.20241155
- Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized resampled importance sampling: foundations of ReSTIR. *ACM Trans. Graph.* 41, 4, Article 75 (July 2022), 23 pages. doi:10.1145/3528223.3530158
- Daqi Lin, Chris Wyman, and Cem Yuksel. 2021. Fast volume rendering with spatio-temporal reservoir resampling. *ACM Trans. Graph.* 40, 6, Article 279 (Dec. 2021), 18 pages. doi:10.1145/3478513.3480499
- Y. Ouyang, S. Liu, M. Kettunen, M. Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. *Computer Graphics Forum* 40 (11 2021), 17–29. doi:10.1111/cgf.14378
- Justin F. Talbot, David Cline, and Parris Egbert. 2005. Importance resampling for global illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques* (Konstanz, Germany) (EGSR '05). Eurographics Association, Goslar, DEU, 139–146.
- Yu-Chen Wang, Chris Wyman, Lifan Wu, and Shuang Zhao. 2023. Amortizing Samples in Physics-Based Inverse Rendering Using ReSTIR. *ACM Trans. Graph.* 42, 6, Article 214 (Dec. 2023), 17 pages. doi:10.1145/3618331
- Song Zhang, Daqi Lin, Markus Kettunen, Cem Yuksel, and Chris Wyman. 2024. Area ReSTIR: Resampling for Real-Time Defocus and Antialiasing. *ACM Trans. Graph.* 43, 4, Article 98 (July 2024), 13 pages. doi:10.1145/3658210

## A Implementação ReSTIR em C++

Disponível em : <https://github.com/grinaldo-oliveira/MATE22>

Received 30 June 2025; revised 30 June 2025; accepted 30 June 2025