

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

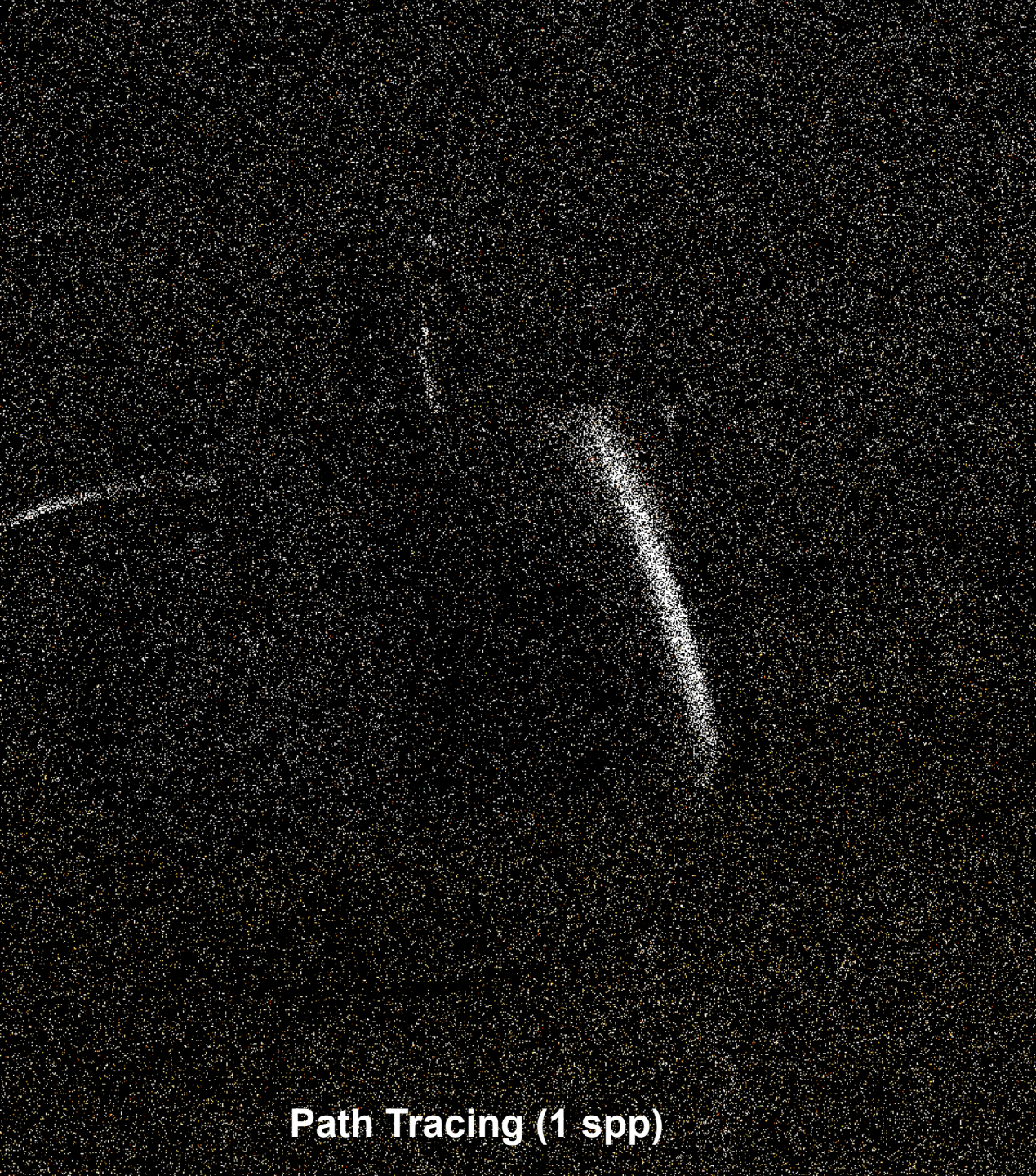
Autoria: Grinaldo Lopes de Oliveira

Baseado em 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

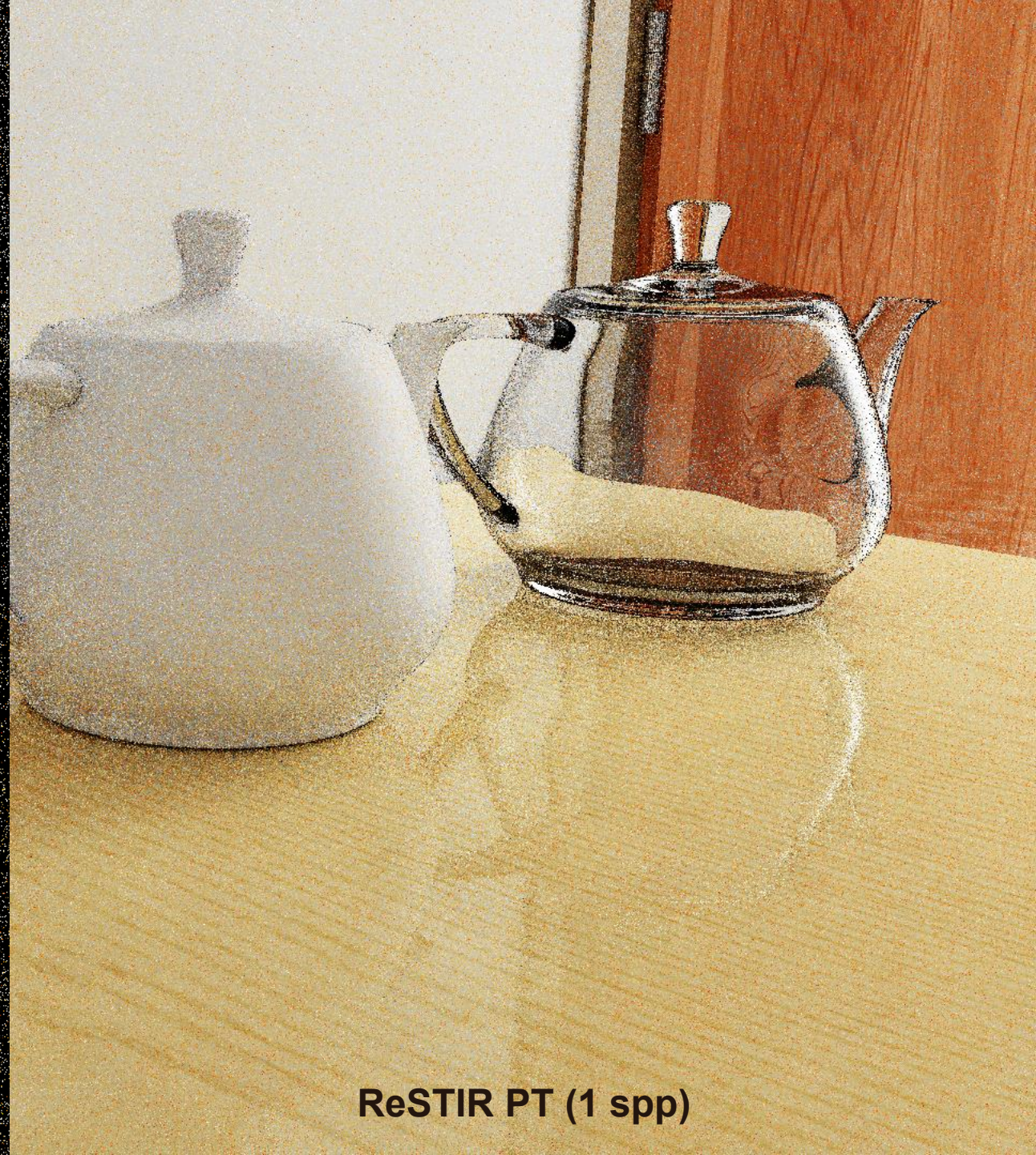
Sonho dos Ray Tracing Maníacos

“E se você pudesse renderizar uma cena complexa, composta por múltiplas luzes diretas, a partir de pouquíssimos raios de luz por pixel? Quem sabe, apenas um único raio... Isto te deixaria feliz? 😊”





Path Tracing (1 spp)



ReSTIR PT (1 spp)

Renderizo
imagens com
poucos raios
candidatos!

Pergunte-me
como....

Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting

BENEDIKT BITTERLI, Dartmouth College
CHRIS WYMAN, NVIDIA
MATT PHARR, NVIDIA
PETER SHIRLEY, NVIDIA
AARON LEFOHN, NVIDIA
WOJCIECH JAROSZ, Dartmouth College



Fig. 1. Two complex scenes ray traced with direct lighting from many dynamic lights. (Left) A still from the Zero Day video [Winkelmann 2015] with 11,000 dynamic emissive triangles. (Right) A view of one ride in an AMUSEMENT PARK scene containing 3.4 million dynamic emissive triangles. Both images show three methods running in equal time on a modern GPU, from left to right: Moreau et al. [2019]’s efficient light-sampling BVH, our new unbiased estimator, and our new biased estimator. The ZERO DAY image is rendered in 15 ms and AMUSEMENT PARK in 50 ms, both at 1920 × 1080 resolution. ZERO DAY ©beepie, Pirate Ship ©sema edis

Efficiently rendering direct lighting from millions of dynamic light sources using Monte Carlo integration remains a challenging problem, even for off-line rendering systems. We introduce a new algorithm—ReSTIR—that renders such lighting interactively, at high quality, and without needing to maintain complex data structures. We repeatedly resample a set of candidate

Authors’ addresses: Benedikt Bitterli, Dartmouth College, benedikt.m.bitterli.gr@dartmouth.edu; Chris Wyman, NVIDIA, chris.wyman@nvidia.com; Matt Pharr, NVIDIA, matt.pharr@gmail.com; Peter Shirley, NVIDIA, ptrshir@gmail.com; Aaron Lefohn, NVIDIA, 2788 San Tomas Expressway, Santa Clara, CA, 95051, alefohn@nvidia.com; Wojciech Jarosz, Dartmouth College, Department of Computer Science, 9 Maynard St. Hanover, NH, 03755, wojciech.k.jarosz@dartmouth.edu.

Permission to make digital or hard copies of all or part of this work for personal or

light samples and apply further spatial and temporal resampling to leverage information from relevant nearby samples. We derive an unbiased Monte Carlo estimator for this approach, and show that it achieves equal-error 60×-60× faster than state-of-the-art methods. A biased estimator reduces noise further and is 35×-65× faster, at the cost of some energy loss. We implemented our approach on the GPU, rendering complex scenes containing up to 3.4 million dynamic, emissive triangles in under 50 ms per frame while tracing at most 8 rays per pixel.

CCS Concepts: • Computing methodologies → Ray tracing.

Additional Key Words and Phrases: Photorealistic rendering, resampled

Preliminares

Integração

- A equação de Renderização

$$L_o(\omega_o) = \int_{\Omega} L_i(\omega_i) f_s(\omega_i, \omega_o) (\mathbf{n} \cdot \omega_i) d\omega_i$$

$$I = \int_{\Omega} f(x) dx$$

Integração

- Integral

$$I = \int_{\Omega} f(x) \, dx$$

- Integração Monte Carlo:

$$I \approx \langle I \rangle = |\Omega| \frac{1}{M} \sum_i^M f(X_i)$$

Distribuição *Aleatória Uniforme* de X_i

$$p_i(X_i) = \frac{1}{|\Omega|}$$

Amostragem por Importância:

$$I \approx \langle I \rangle = \frac{1}{M} \sum_i^M \frac{f(X_i)}{p_i(X_i)}$$

Usando uma função de densidade de probabilidade (PDF) $p_i(X_i)$ para X_i

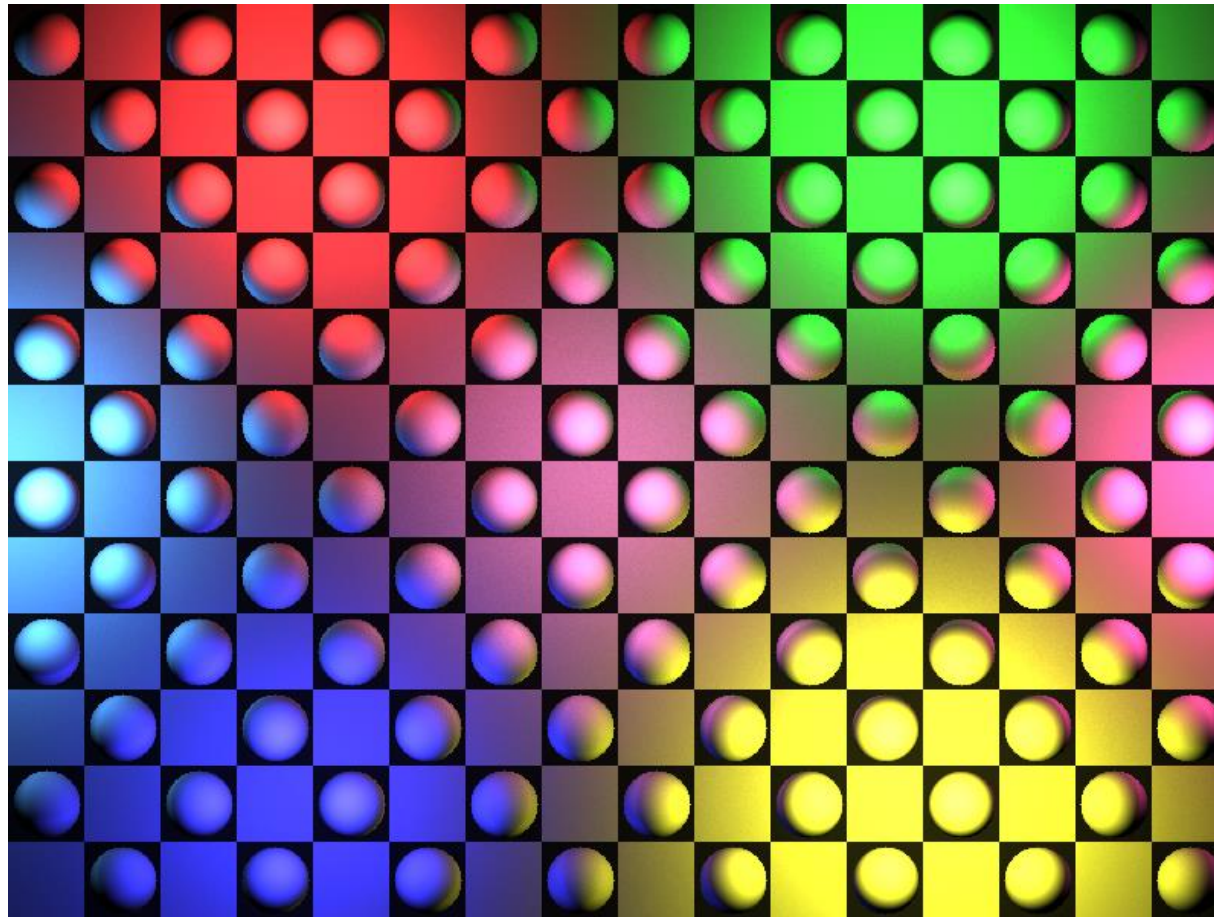
Integração Monte Carlo

$$I = \int_{\Omega} f(x) \, dx \qquad I \approx \langle I \rangle = \frac{1}{M} \sum_i^M \frac{f(X_i)}{p_i(X_i)}$$

- $\langle I \rangle$ é um **estimator** de I .
- $\langle I \rangle$ é **unbiased (não enviesado)**, se $\langle I \rangle$ retorna o valor correto sobre a média.
- $\langle I \rangle$ é **consistente**, se ele converge para I quando M tende a infinito.
- $M = 1$

$$\langle I \rangle = \frac{f(X_i)}{p_i(X_i)}$$

E Monte Carlo funciona?



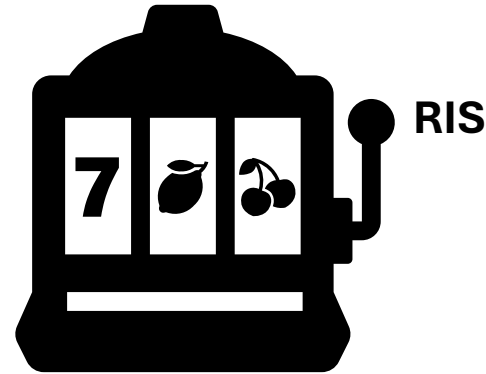
1000 raios em 50,9 s

Ok... Mas foram 1.000
raios... Cadê o raio
matador?

Resampled Importance Sampling (RIS)

RIS: uma máquina que produz amostras proporcionalmente aproximadas a uma distribuição alvo

Amostras X_1, X_2, \dots, X_M



Uma amostra melhor distribuída

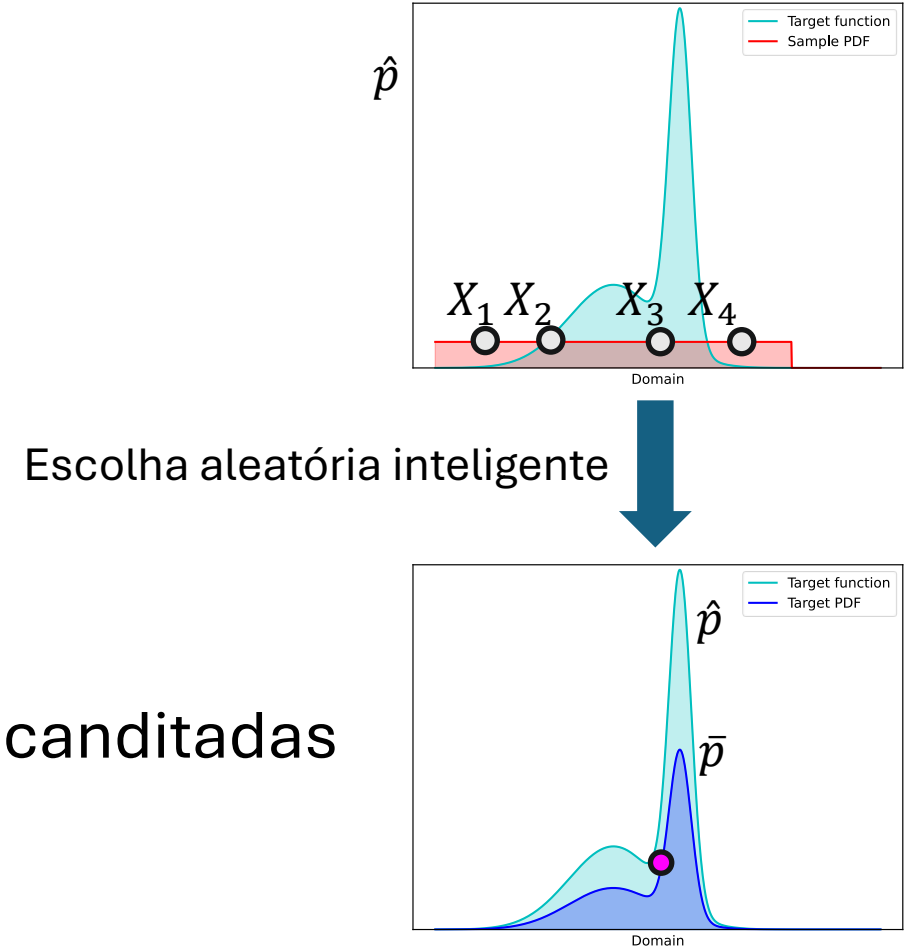
RIS em Alto Nível

Dado:

- função alvo \hat{p}
- amostras candidatas (X_1, X_2, \dots, X_M)

RIS:

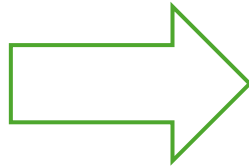
- avalia *resampling weights* $w_i \sim \hat{p}$ para todas candidadas
- Retorna uma aleatória, proporcional à w_i



Encapsulando o RIS

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  $y, wsum$ 
```



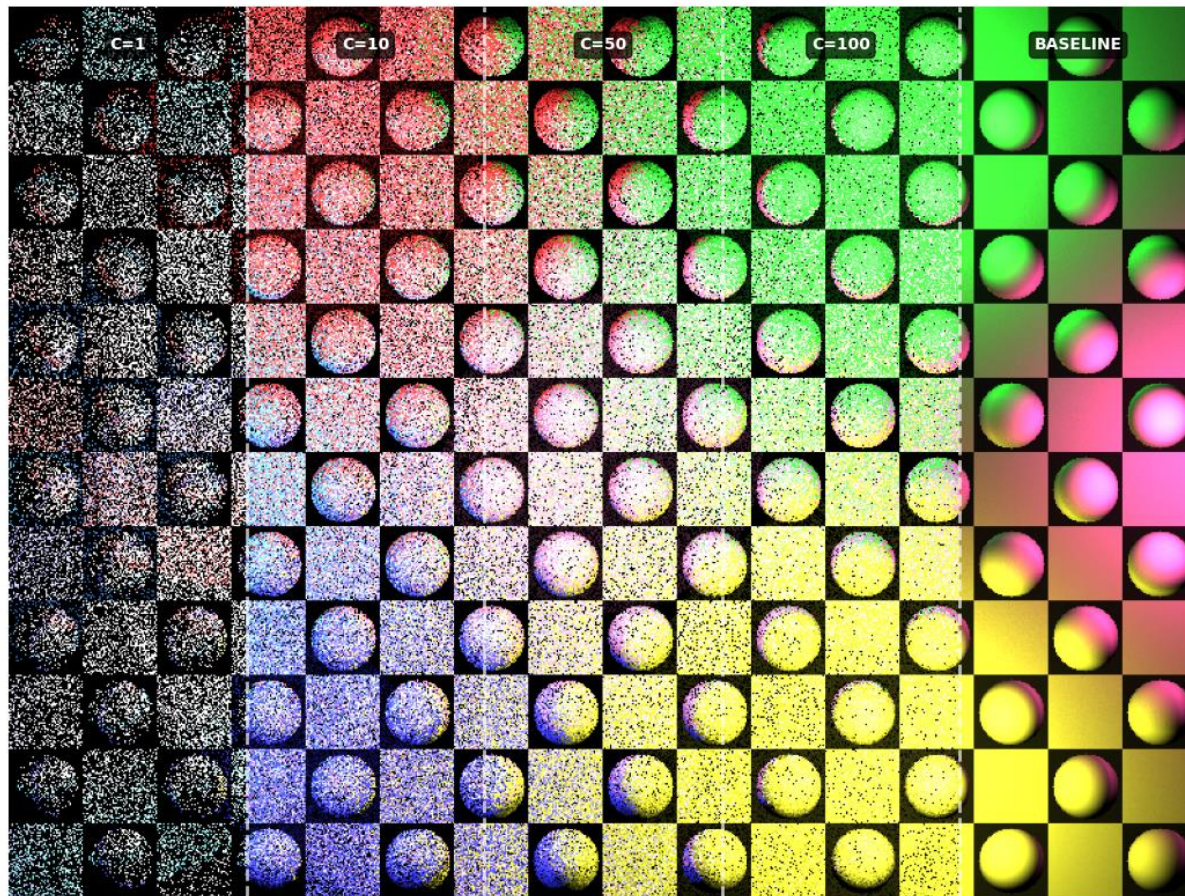
Reservatório = $\{y, w_{sum}, M, W\}$,

onde:

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

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

E funciona (RIS)?



E como o ReSTIR entra
nesta história?

Algoritmo ReSTIR

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: $reservoirs[q] \leftarrow RIS(q)$ \triangleright M candidatos via Monte Carlo
- 7: **end for**

Algoritmo ReSTIR

```
8: Fase 2: Avaliação de Visibilidade  
9: for each pixel  $q$  na imagem do  
10:   if  $\text{shadowed}(\text{reservoirs}[q].y)$  then  
11:      $\text{reservoirs}[q].W \leftarrow 0$             $\triangleright$  Rejeita amostras ocluídas  
12:   end if  
13: end for
```


Algoritmo ReSTIR

```
14: Fase 3: Reuso Temporal  
15: for each pixel  $q$  na imagem do  
16:    $q' \leftarrow \text{pickTemporalNeighbor}(q)$  ▷ Motion vectors  
17:    $\text{reservoirs}[q] \leftarrow \text{combineReservoirs}(q, \text{reservoirs}[q],$   
18:      $\text{prevFrame}[q'])$   
19: end for
```



Risco de Viés e Super estimação do valor de M

Algoritmo ReSTIR

```
20: Fase 4: Reuso Espacial Iterativo  
21: for  $i \leftarrow 1$  to  $n$  do  
22:   for each pixel  $q$  na imagem do  
23:      $Q \leftarrow \text{pickSpatialNeighbors}(q)$   
24:      $R \leftarrow \{\text{reservoirs}[q'] | q' \in Q\}$   
25:      $\text{reservoirs}[q] \leftarrow \text{combineReservoirs}(q, \text{reservoirs}[q], R)$   
26:   end for  
27: end for
```



Risco de Viés

Modo	k (vizinhos)	n (iterações)	Raio típico
Biased	5	2	30 pixels
Unbiased	3	1	30 pixels

Algoritmo ReSTIR

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** $reservoirs$

Experimentos

```
C:\Users\grina\OneDrive\Doutorado UFBA 2025\MATE22\Artigo Final\Code>restir_completo -h
=== Renderizador ReSTIR CORRIGIDO - Sem Escurecimento - Compatível C++98 ===
Uso: restir_completo [opções]
Opções:
  -c, --candidates <numero>      Define MAX_CANDIDATES (padrao: 30)
  -s, --spatial-reuse             Ativa amostragem espacial
      --no-spatial-reuse         Desativa amostragem espacial
  -t, --temporal-reuse            Ativa amostragem temporal
      --no-temporal-reuse        Desativa amostragem temporal
  -b, --baseline <arquivo.ppm>   Usa imagem baseline para reutilizacao temporal
      --biased                   Usa versão biased (padrao)
      --unbiased                 Usa versão unbiased CORRIGIDA
      --monte-carlo              Usa Monte Carlo puro (desabilita RIS)
  -h, --help                     Mostra esta ajuda


Exemplos CORRIGIDOS:
  restir_completo -c 50 -s -t --unbiased      # Configuração unbiased CORRIGIDA
  restir_completo -c 50 -s -t --biased        # Configuração biased completa
  restir_completo -b baseline.ppm -t --unbiased # Usa imagem baseline (unbiased CORRIGIDO)
  restir_completo --monte-carlo -c 100        # Monte Carlo puro com 100 candidatos
```

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

ReSTIR - C

Amostragem Inicial RIS (Reservoir Importance Sampling)

Para cada pixel, gera MAX_CANDIDATES amostras aleatórias de luzes. Calcula peso baseado na luminância e usa seleção probabilística.


7 Luzes: $\text{Peso} = \text{targetPdf} / \text{sourcePdf}$

```
for (int i = 0; i < MAX_CANDIDATES; i++) {  
    int lightIndex =  
        randomInt(lights.size());  
    reservoir.update(lights, point,  
        lightIndex);  
}
```

Reutilização Temporal (Temporal Reuse)

Combina o reservatório atual com informações do frame anterior (imagem baseline). Suporta modo BIASED/UNBIASED.

$$T-1 + T = RT$$

```
if (ENABLE_TEMPORAL_REUSE) {  
    reservoir.combine(previousFrame[pixel],  
        lights, point);  
}
```

Reutilização Espacial (Spatial Reuse)

Examina pixels vizinhos (raio $\approx 20\text{px}$) e combina seus reservatórios usando pesos probabilísticos. Suporta modo BIASED/UNBIASED.



```
for (int i = 0; i < spatialSamples; i++) {  
    int neighbor =  
        getRandomNeighbor(x, y);  
    reservoir.combine(neighbors[neighbor]);  
}
```

Estimação Final (Radiance Estimation)

Calcula a cor final. Aplica peso $W = (\text{weight}/M)/\text{targetPdf}$ e adiciona iluminação ambiente.

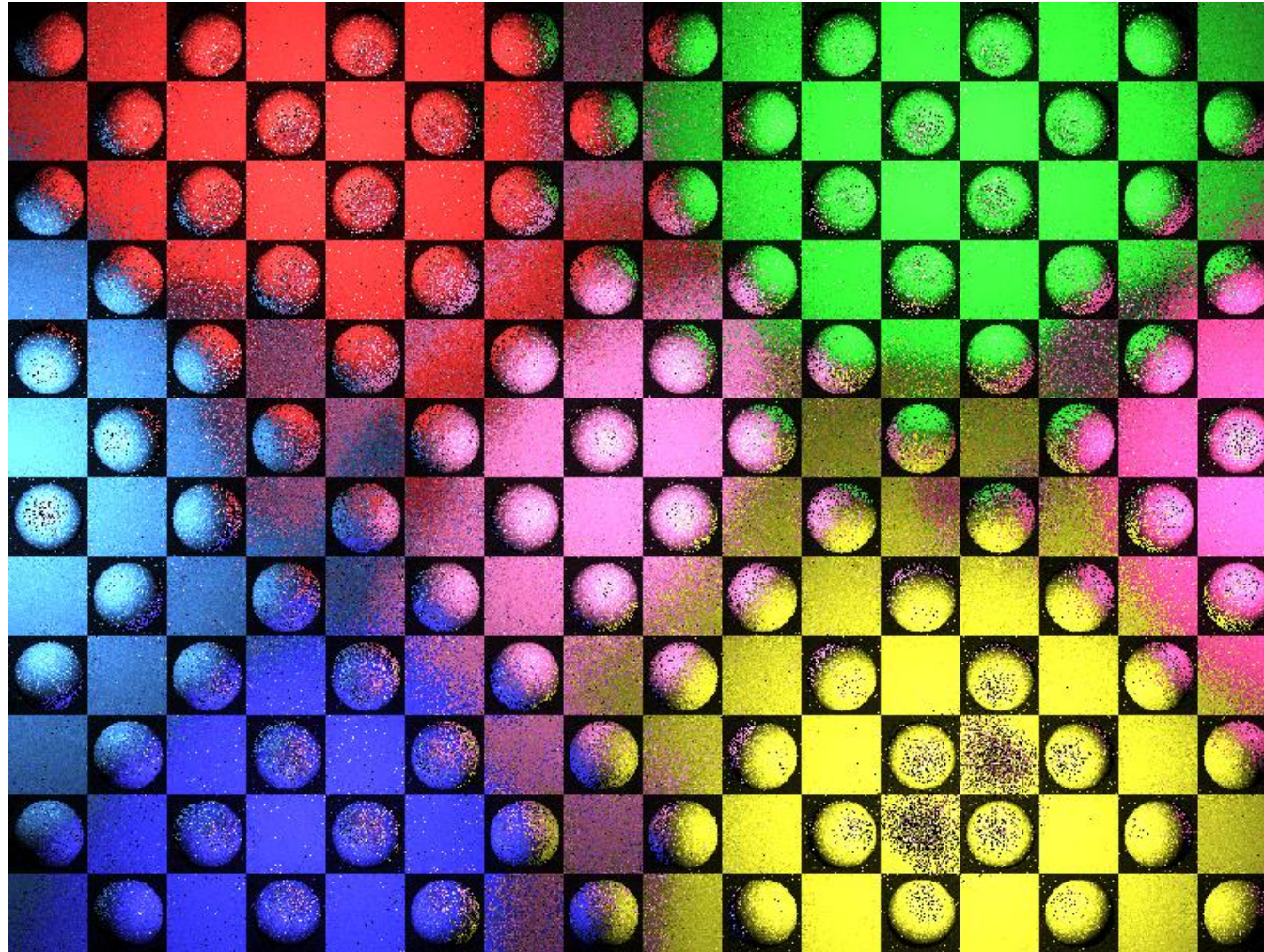


Resultado: Cor final + Ambiente

```
Color getFinalColor() {  
    float W = (weight / M) /  
        targetPdf;  
    return light.calculateLighting() *  
        W;  
}
```

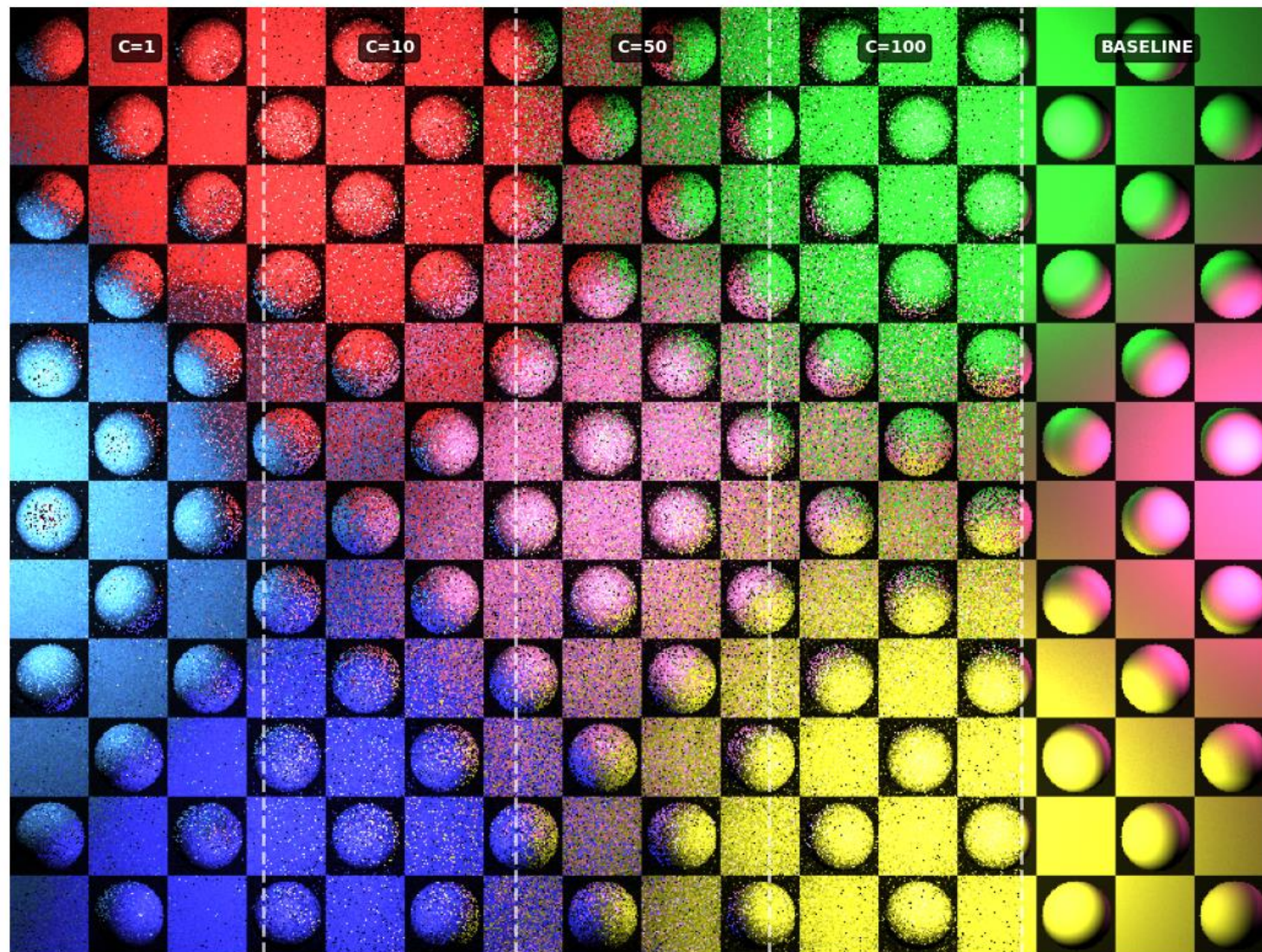
* Programado em C++ e conduzido com o apoio de sistemas de inteligência artificial, especificamente Claude.AI e Perplexity.AI.

E funciona (biased)?

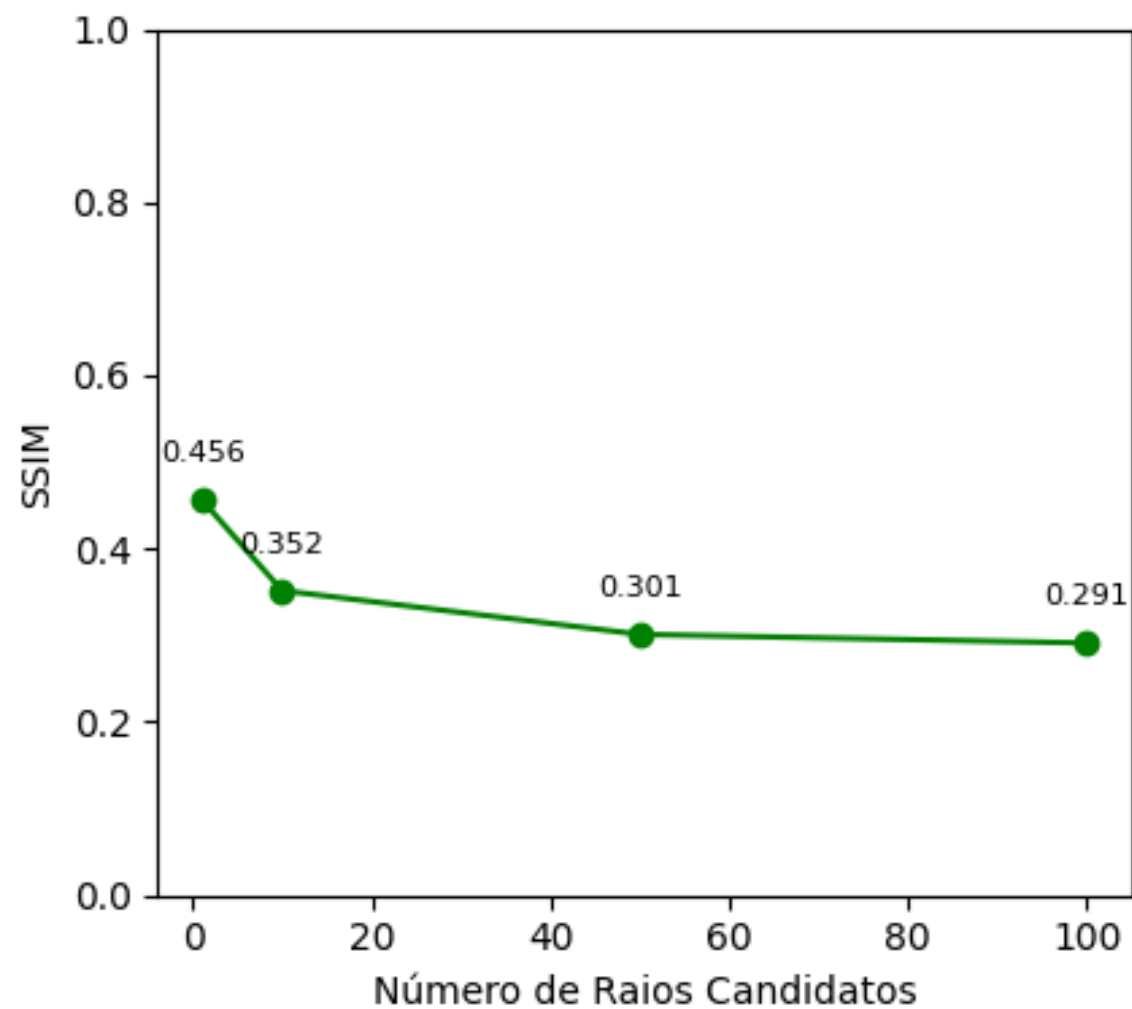


1 único raio em 1,4 s

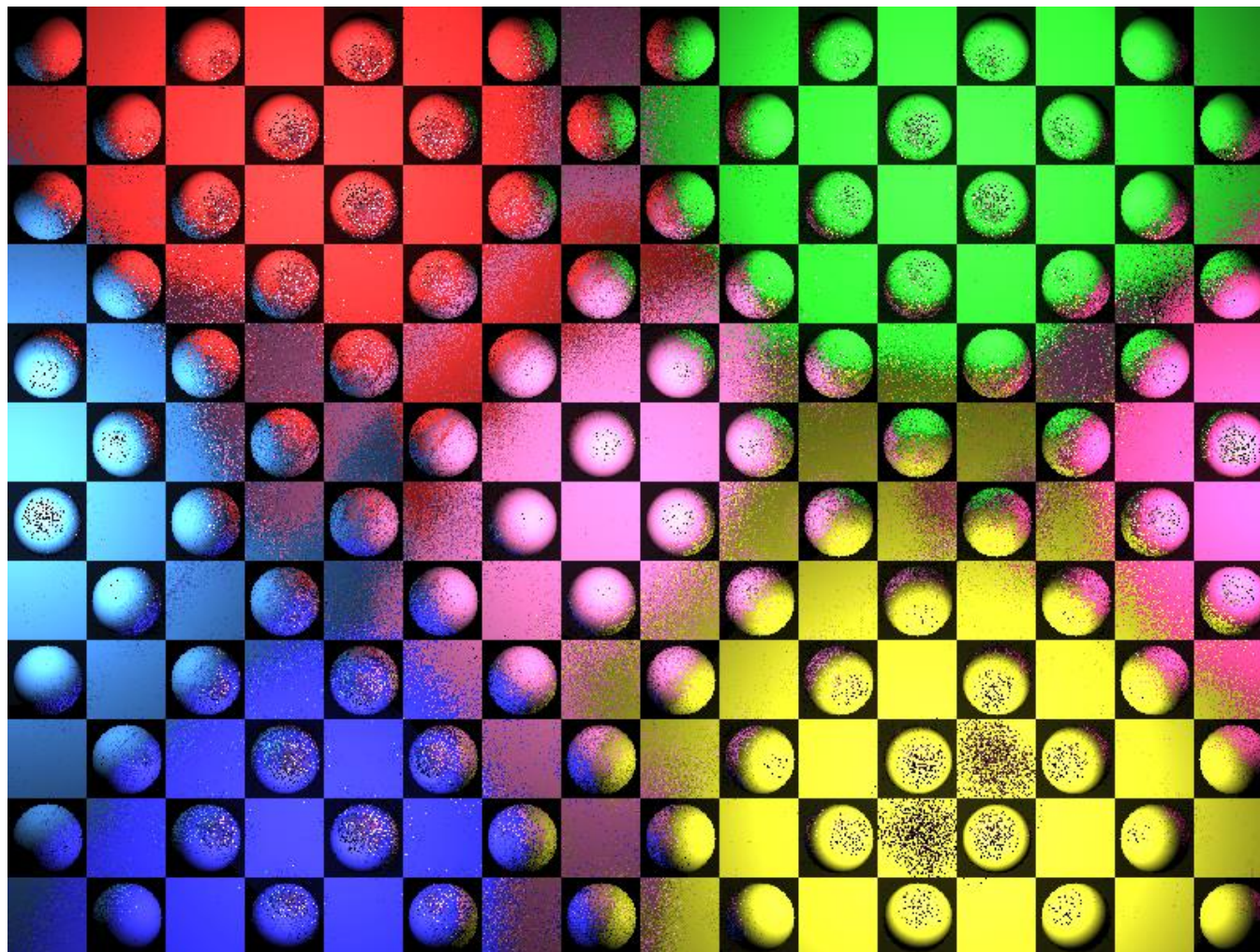
E funciona (biased)?



E funciona (biased)?

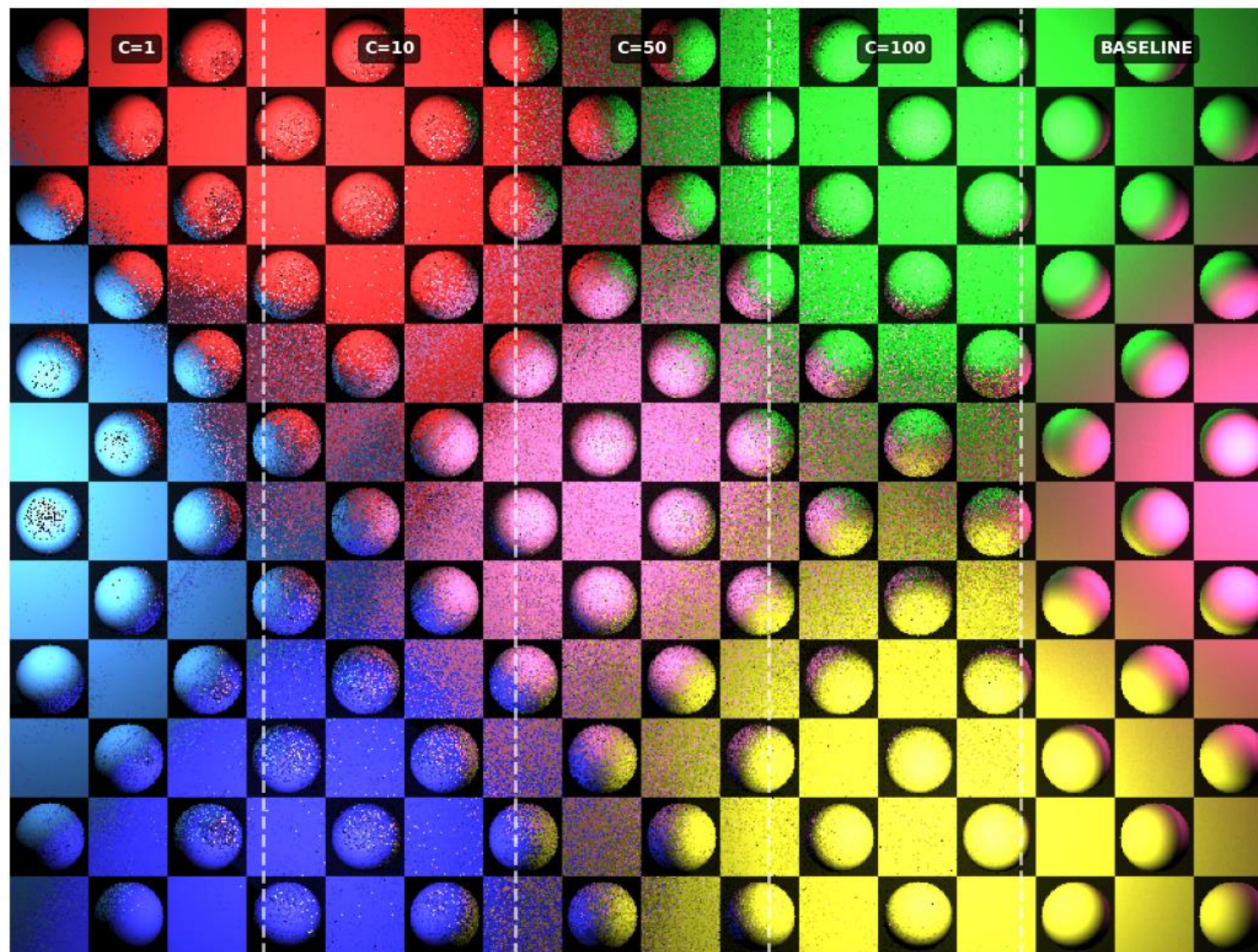


E funciona (unbiased)?

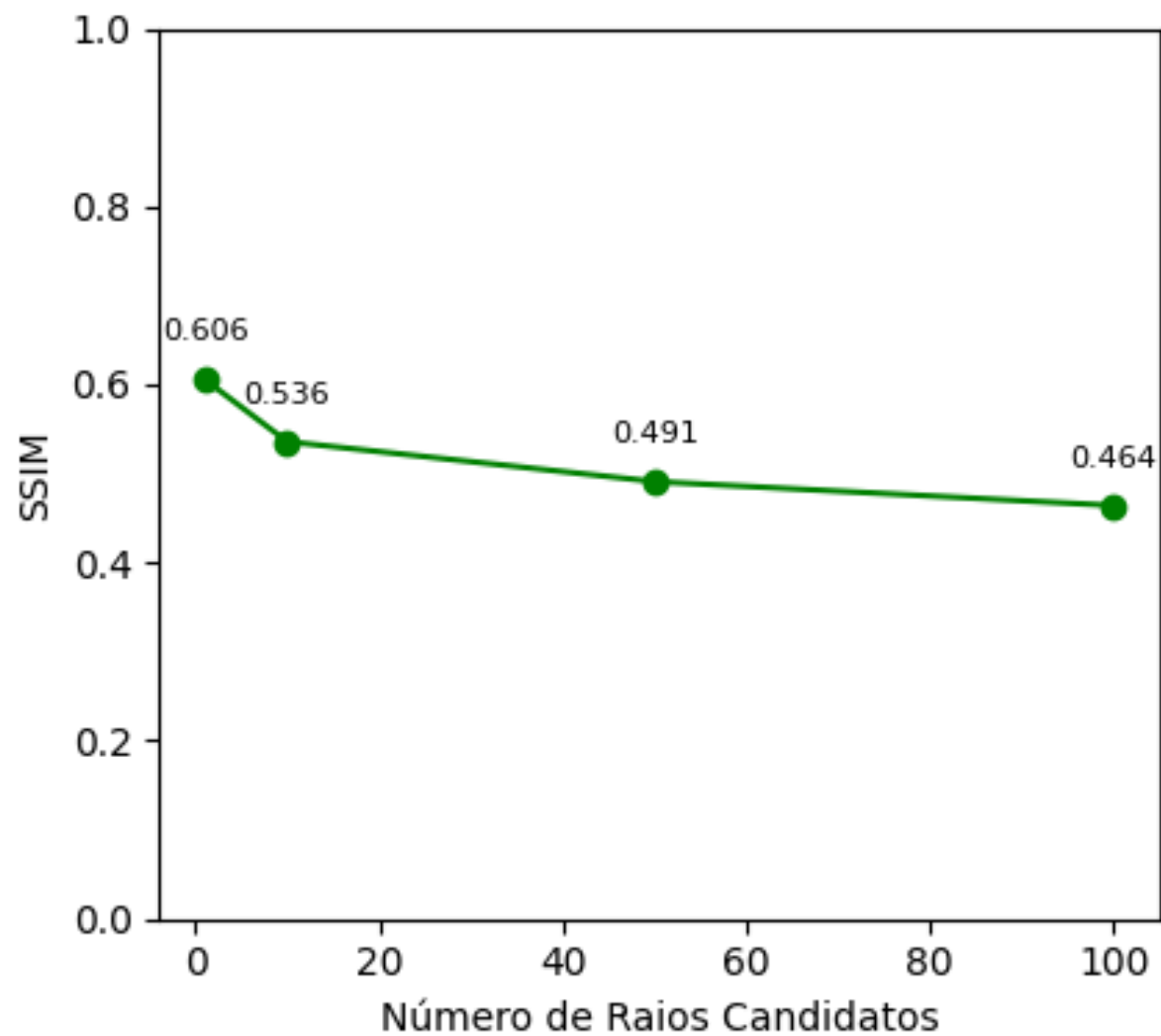


1 raio em 2,1 s

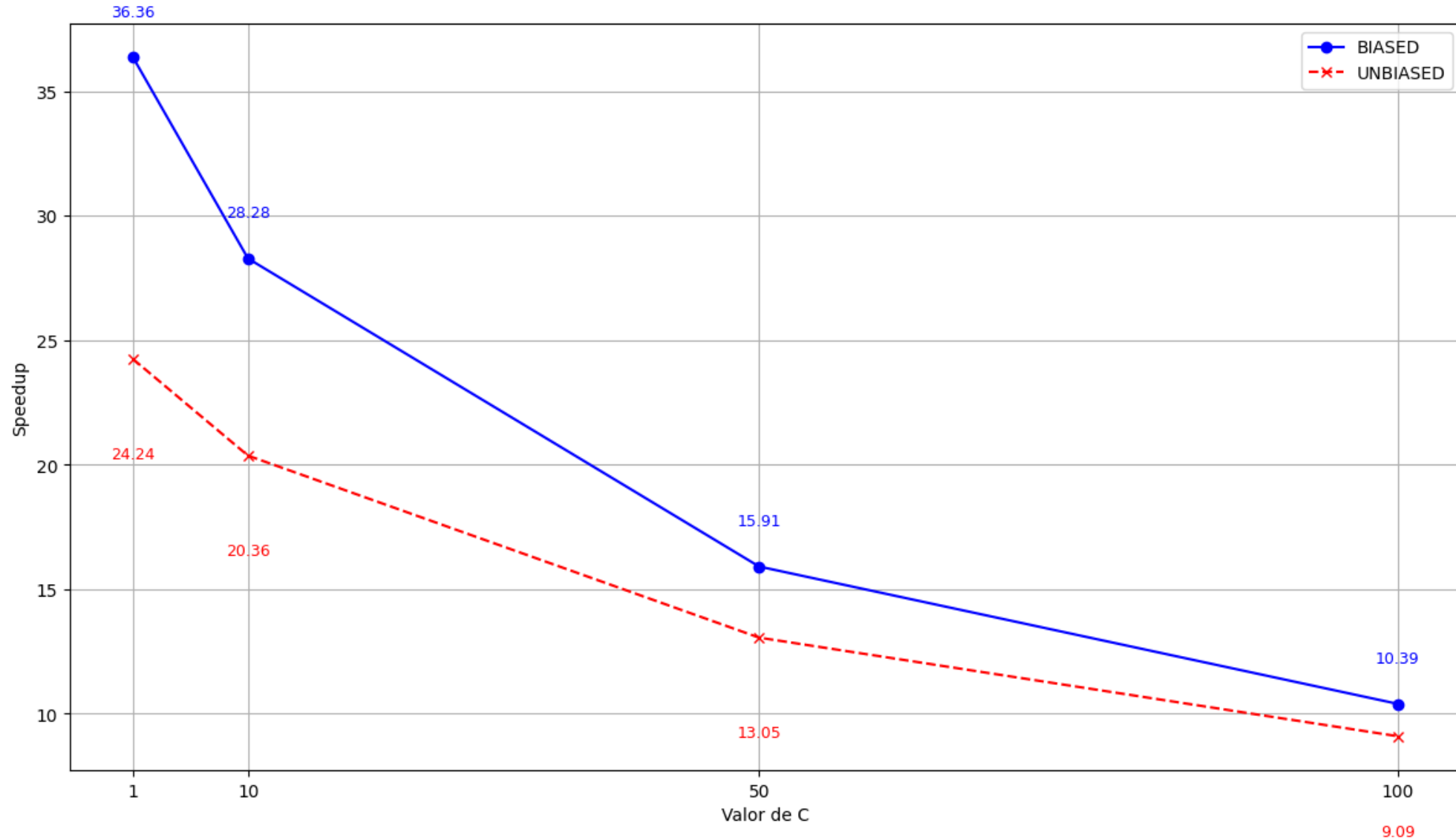
E funziona (unbiased)?



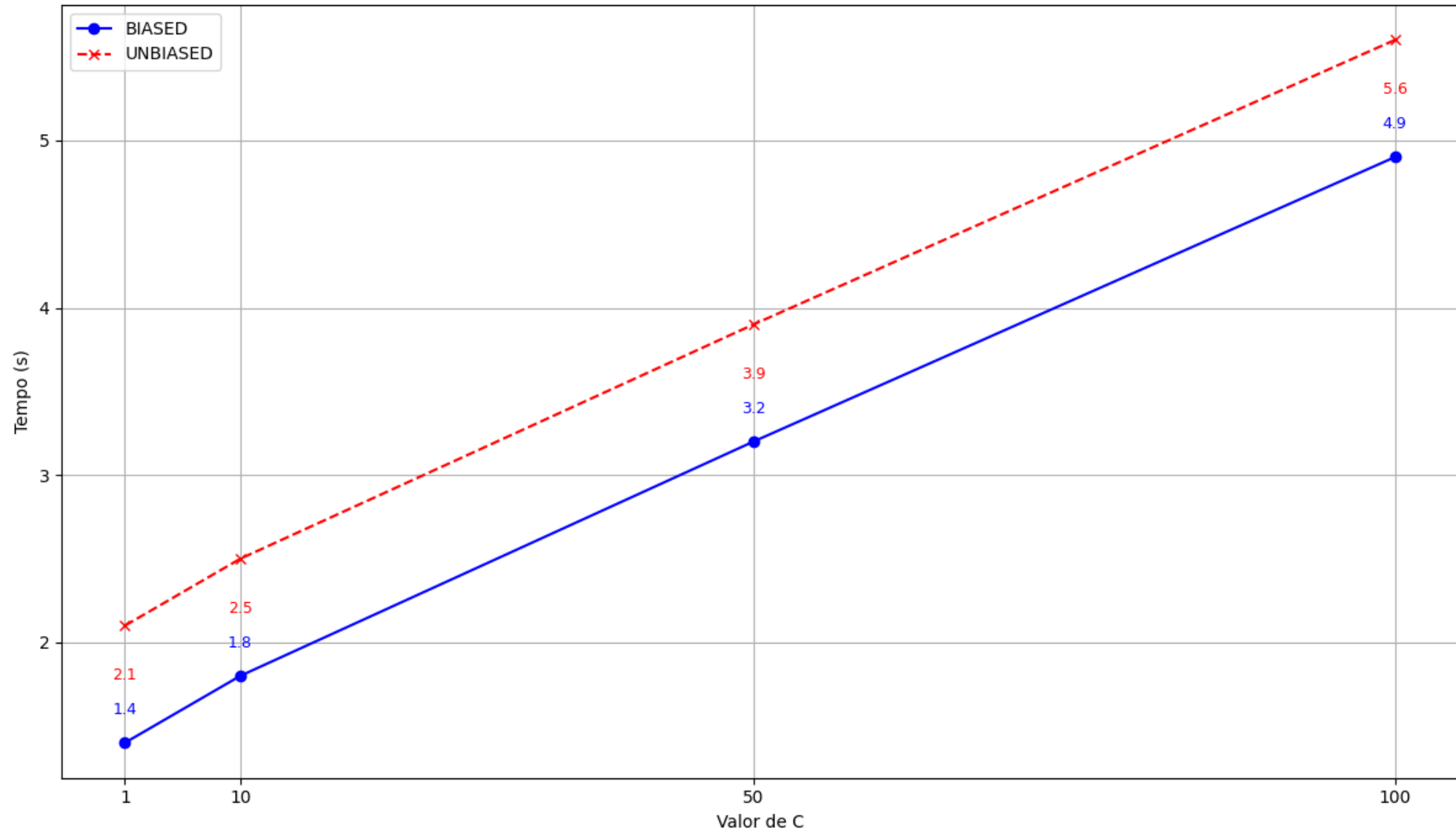
E funciona (biased)?



E a performance?



E a performance $O(M + nk)$?



Epílogo

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 " <i>final gathering</i> " 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.

Quadro Comparativo: Potencialidades vs Limitações

Aspecto	Potencialidades	Limitações
Desempenho	<ul style="list-style-type: none">• 6× a 60× mais rápido que técnicas anteriores• Modo enviesado: 35× a 65× mais rápido• Renderização interativa <50ms por quadro	<ul style="list-style-type: none">• Modo enviesado pode introduzir perda de energia (imagens mais escuras)• Ruído em Artefatos próximos a descontinuidades
Qualidade Visual	<ul style="list-style-type: none">• Alta qualidade com poucos raios por pixel• Melhora performance de denoisers	<ul style="list-style-type: none">• Degradação em superfícies de alta frequência• Problemas com cabelo, folhagem e detalhes finos
Escalabilidade	<ul style="list-style-type: none">• Suporta milhões de luzes dinâmicas• Escalável para diferentes tamanhos de cena• Funciona bem em GPUs modernas	<ul style="list-style-type: none">• Desafios com movimento rápido
Implementação	<ul style="list-style-type: none">• Estruturas simples (reservatórios por pixel)• Dispensa estruturas de dados complexas• Facilita implementação e manutenção	<ul style="list-style-type: none">• Requer heurísticas para rejeição de amostras, o que aumenta complexidade do pipeline e pode impactar robustez do método
Versatilidade	<ul style="list-style-type: none">• Extensível para múltiplos domínios Variantes: GI, PT, Volumétrico, Area.	<ul style="list-style-type: none">• Necessidade de adaptações específicas

Obrigado!