Accurate Volume Rendering based on Adaptive Numerical Integration

Leonardo Quatrin Campagnolo
Waldemar Celes Filho
Luiz Henrique de Figueiredo, IMPA

TecGraf/PUC-Rio

August 27, 2015

Volume Rendering Integral (VRI)

- The VRI, based on the emission plus absortion model (Max [1995]), is defined by:

$$I = \int_0^D C(s(t))\tau(s(t))e^{-\int_0^t \tau(s(t'))\mathrm{d}t'}\,\mathrm{d}t$$

- Two integrals: Outer and Inner.

- Challenge: Evaluate volume rendering integral accurately, maintaning good performance.

Our proposal

- Iterative and adaptive integration algorithm to evaluate the Volume Rendering Integral.
- Based on Adaptive Simpson's Rule.
- Integrate both integrals in consonance.

- We aim to obtain:
  - Higher performance in low variation regions.
  - Higher precision in high variation regions.

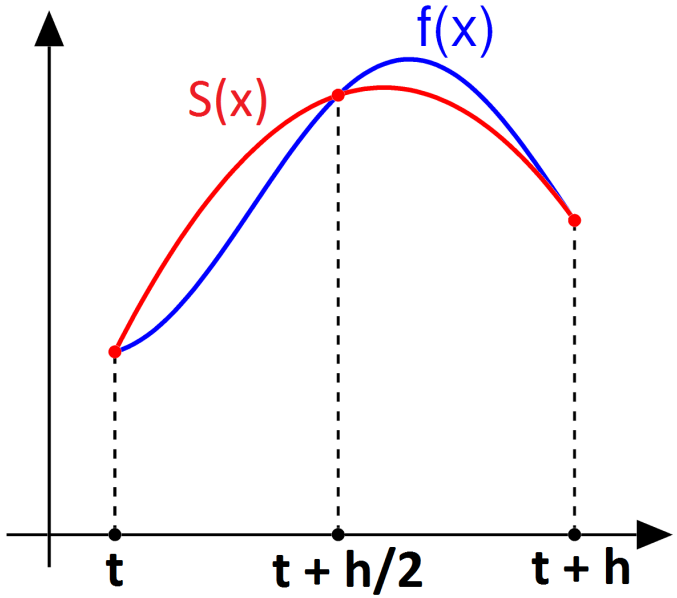
- Two strategies to control the integration step.

Related work

- Novins & Arvo [1992]
  - Adaptive integration
  - Pre-classification of the scalar field
- Marchesin & de Verdiere [2009]
  - Accurate evaluation
  - Use of spatial data structure
- Lindholm et al. [2013]
  - Adaptive integration near boundaries
  - Require isosurface extraction

Review: Simpson's rule

$$\int_t^{t+h} f(x)dx \approx S(t,h) = \frac{h}{6}\left[f(t) + 4f\left(t + \frac{h}{2}\right) + f(t+h)\right]$$



Review: Adaptive Simpson's Rule

$$\int_t^{t+h} f(x)dx \approx D(t,h) + E(t,h)$$

$$D(t,h) = S\left(t, \tfrac{h}{2}\right) + S\left(t + \tfrac{h}{2}, \tfrac{h}{2}\right)$$

$$E(t,h) = \tfrac{1}{15}\left|D(t,h) - S(t,h)\right|$$

Review: Adaptive Simpson's Rule

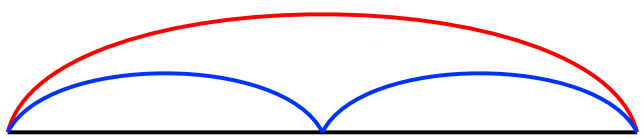
$$\int_t^{t+h} f(x)dx \approx D(t,h) + E(t,h)$$

$$D(t,h) = S\left(t, \tfrac{h}{2}\right) + S\left(t + \tfrac{h}{2}, \tfrac{h}{2}\right)$$

$$E(t,h) = \tfrac{1}{15}\left|D(t,h) - S(t,h)\right|$$

Success

if $E(t,h) \leq \epsilon$ **then**
  return $D(t,h) + E(t,h)$
end if



Review: Adaptive Simpson's Rule

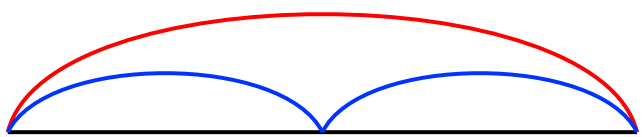
$$\int_t^{t+h} f(x)dx \approx D(t,h) + E(t,h)$$

$$D(t,h) = S\left(t, \tfrac{h}{2}\right) + S\left(t + \tfrac{h}{2}, \tfrac{h}{2}\right)$$

$$E(t,h) = \tfrac{1}{15}\left|D(t,h) - S(t,h)\right|$$

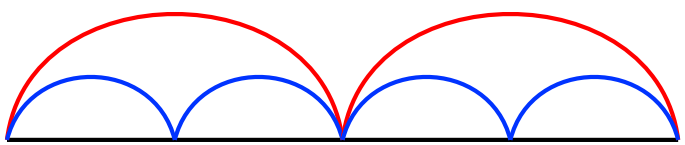
Success

if $E(t,h) \leq \epsilon$ **then**
  return $D(t,h) + E(t,h)$
end if



Failure

if $E(t,h) > \epsilon$ **then**
  return $\int(t, \frac{h}{2}, \frac{\epsilon}{2}) + \int(t + \frac{h}{2}, \frac{h}{2}, \frac{\epsilon}{2})$
end if



Computing VRI – Naïve adaptive approach

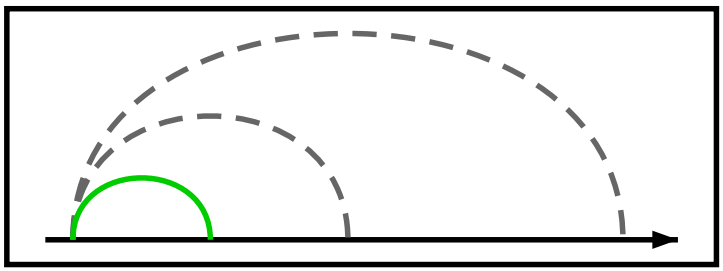
- Simple recursive implementation.

$$I = \int_0^D C(s(t))\tau(s(t))e^{-\int_0^t \tau(s(t'))\mathrm{d}t'}\,\mathrm{d}t$$

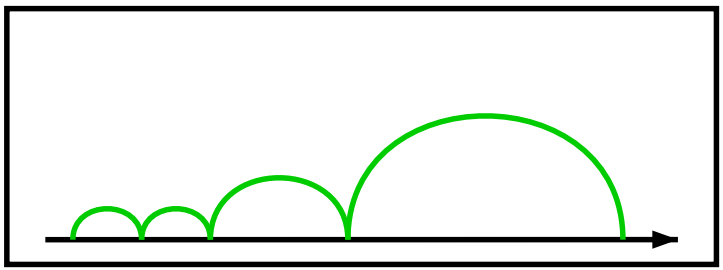
- Not easily mapped to GPUs (recursive).
- Costs several re-evaluations of the inner integral.

Our proposal – Iterative process

- Based on numerical methods for ODE solvers (e.g. Adaptive Runge–Kutta).
- Evaluate intervals using adaptive simpson's rule.
- Two main cases of our iterative process:
  - Halve the step if the error exceeds the tolerance and re-evaluate this new step.



  - Double the step size if we get two consecutive successes.



Our proposal – Iterative process

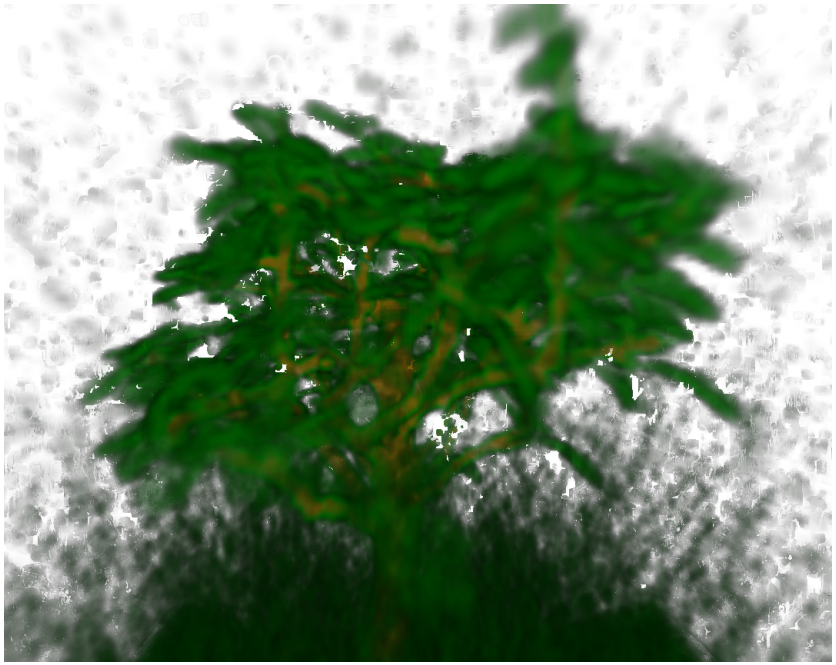
Input: $t, \Delta t, h_0, \epsilon$
Output: $\int(t, \Delta t, h_0)$

$I = 0$
$t_{final} = t + \Delta t$
$h = \min(h_0, t_{final} - t)$
$s_{count} = 1$
while $t < t_{final}$ **do**
  if $E(t,h) \leq \epsilon$ **then**
    $I = I + D(t,h) + E(t,h)$
    $t = t + h$
    if $s_{count} = 1$ **then**
      $h = 2h$
      $\epsilon = 2\epsilon$
    else
      $s_{count} = 1$
    end if
    $h = \min(h, t_{final} - t)$
  else
    $h = h/2$
    $\epsilon = \epsilon/2$
    $s_{count} = 0$
  end if
end while
return I

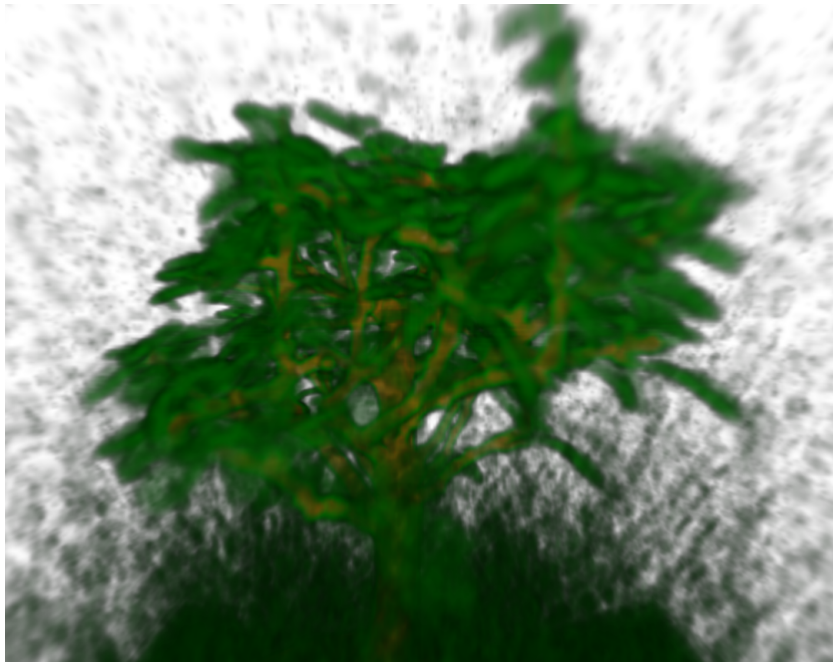
Additional parameters – h_{min} and h_{max}

- Set a defined step-size range.
- h_{min}
  - Prevents the unecessary reduction of the step size where, in practive, the error cannot be evaluated precisely.
  - If $h < h_{min}$, we have a success.
- h_{max}
  - Prevents missing features.

Additional parameters – h_{max} example



without h_{max}

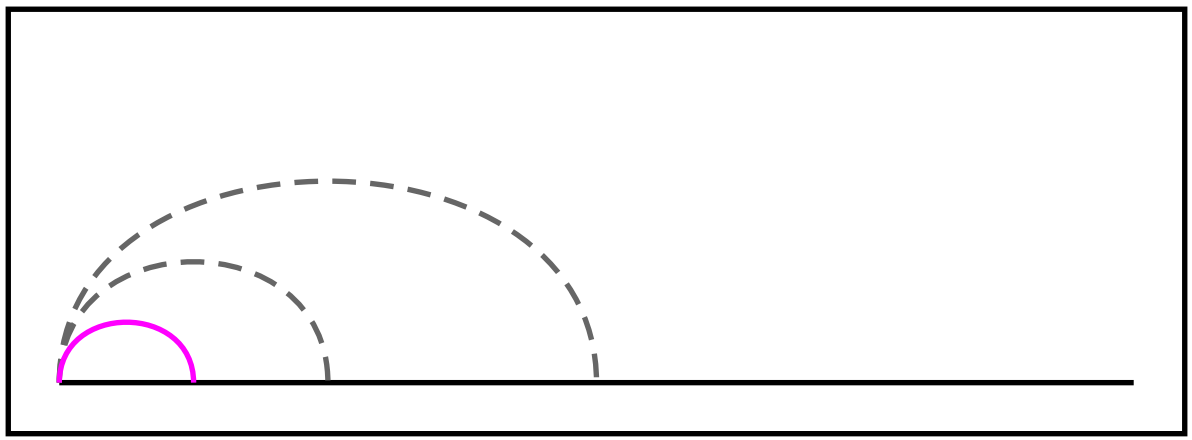


with $h_{max} = 8.0$

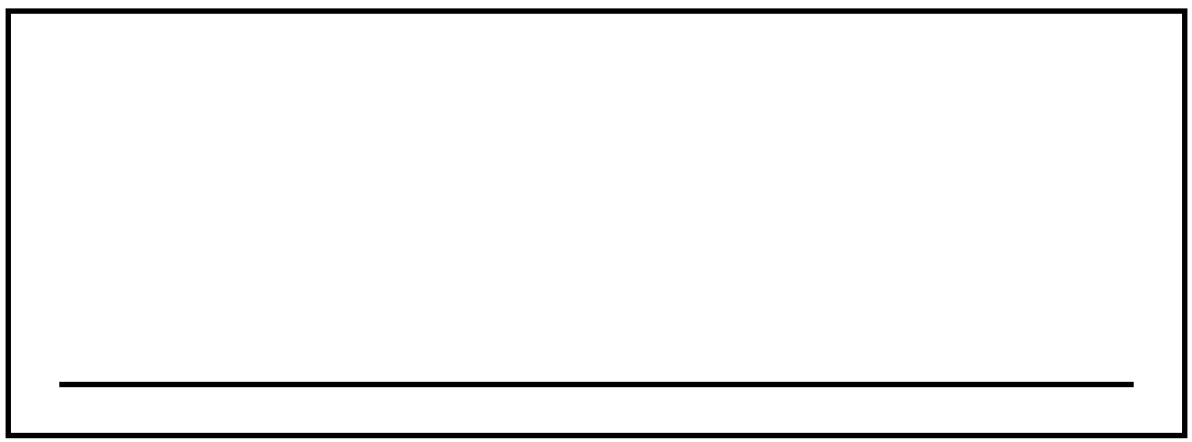
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



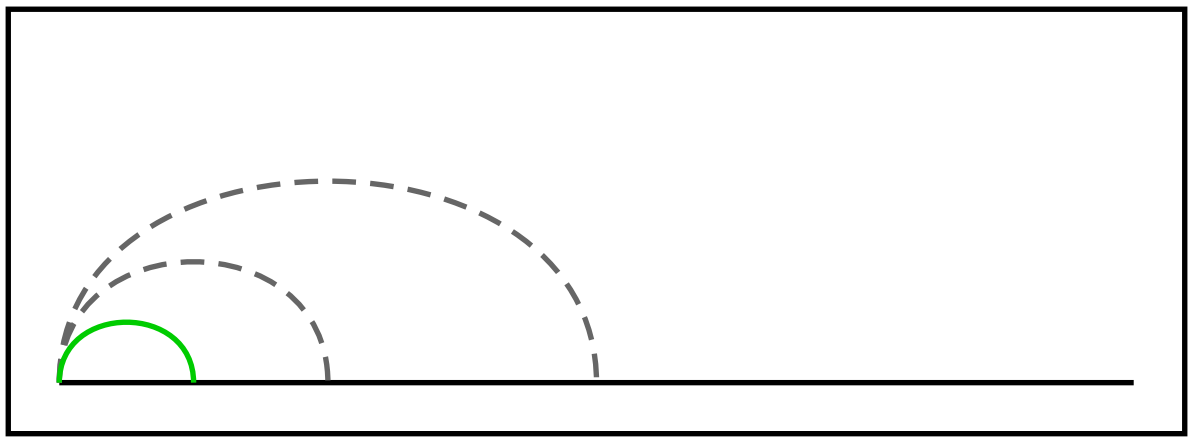
$\int_{[O]}$ – Outer Integral:



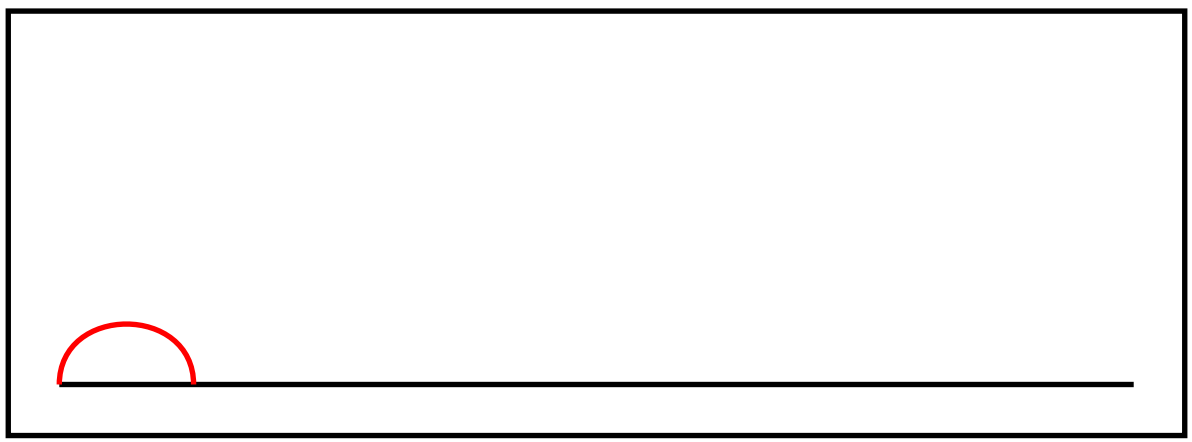
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



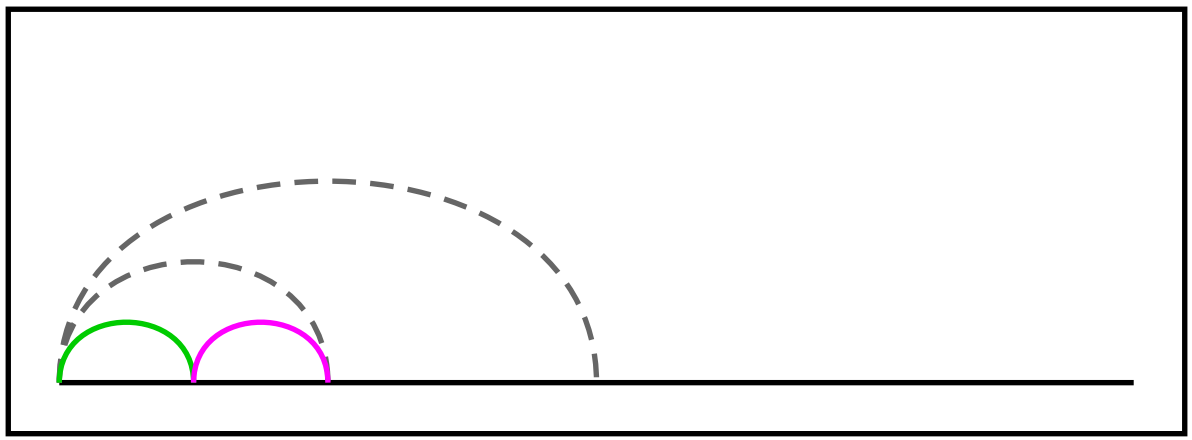
$\int_{[O]}$ – Outer Integral:



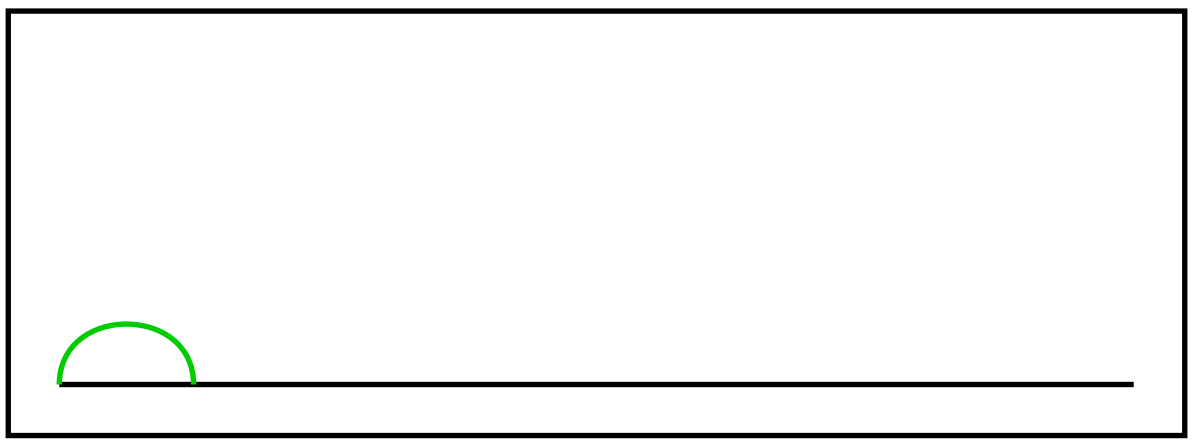
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



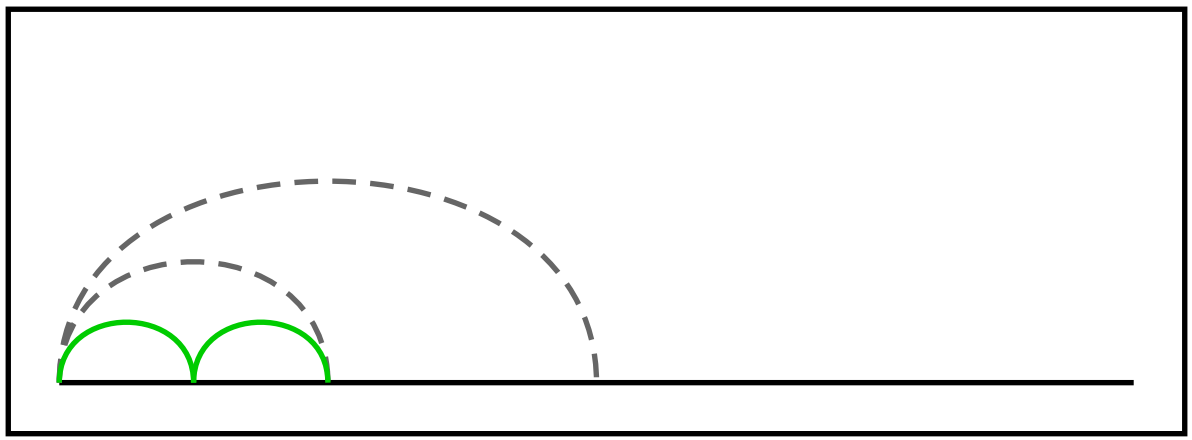
$\int_{[O]}$ – Outer Integral:



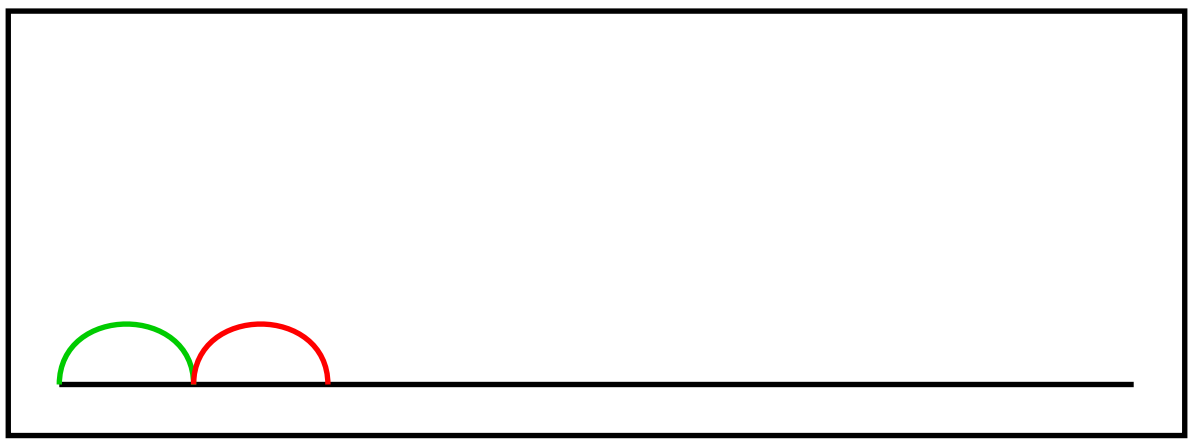
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



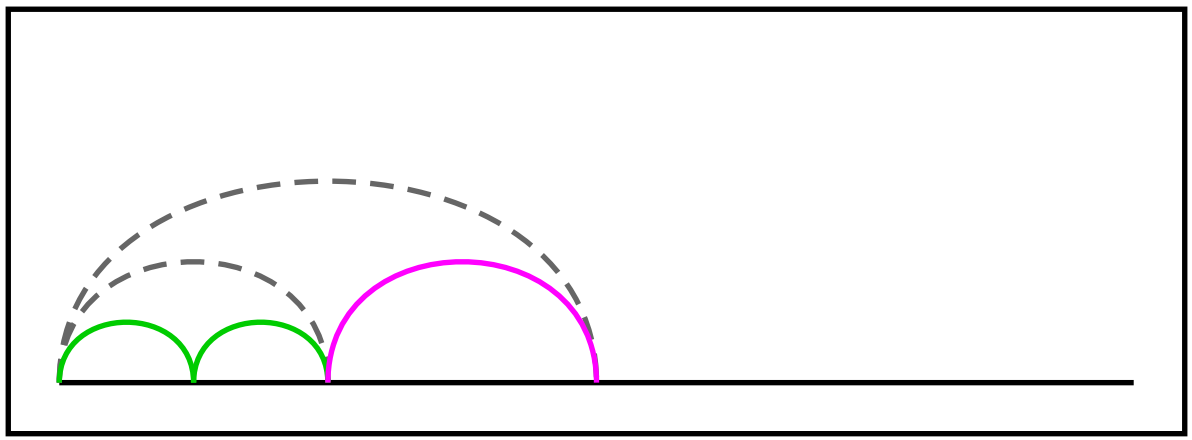
$\int_{[O]}$ – Outer Integral:



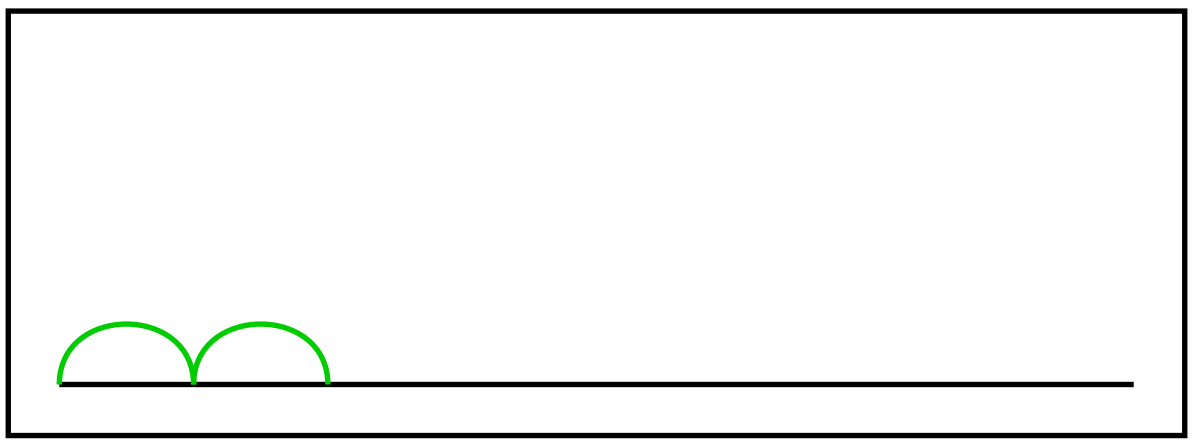
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



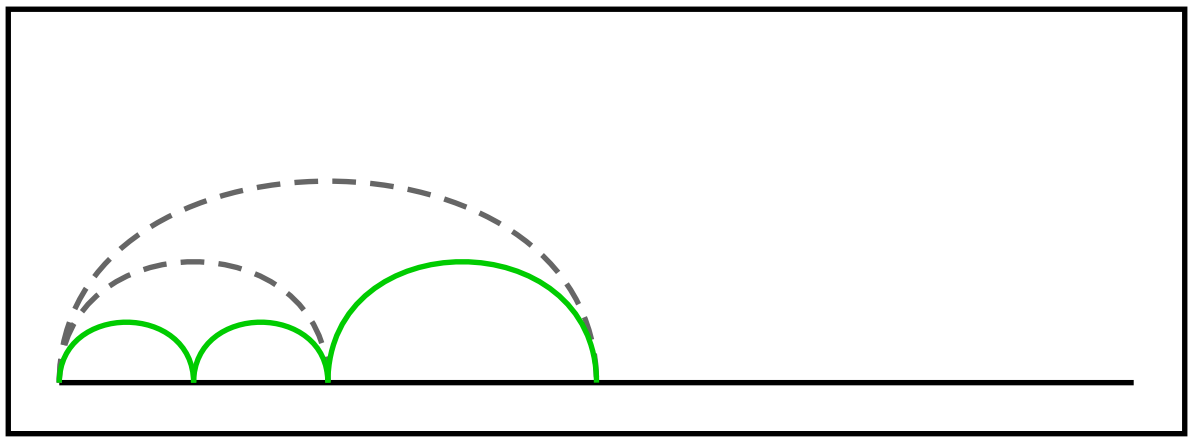
$\int_{[O]}$ – Outer Integral:



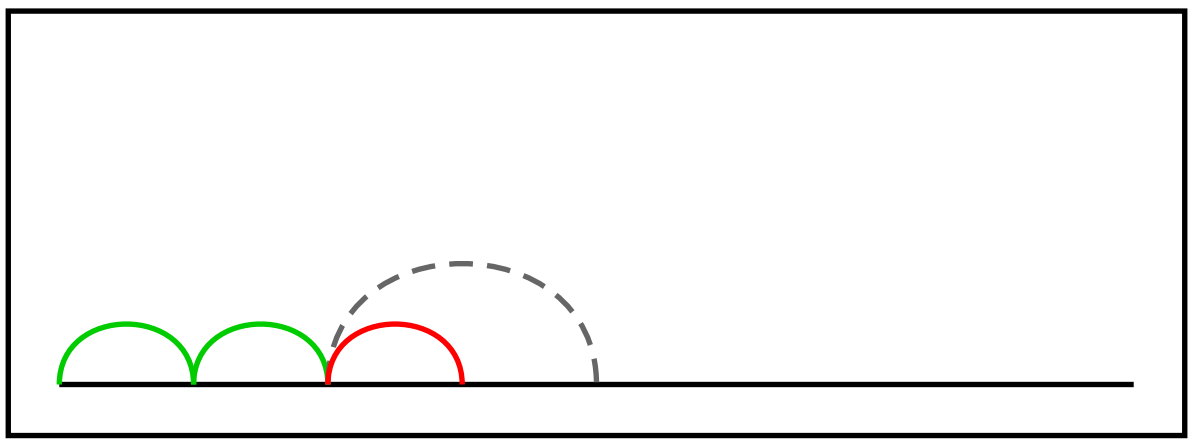
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



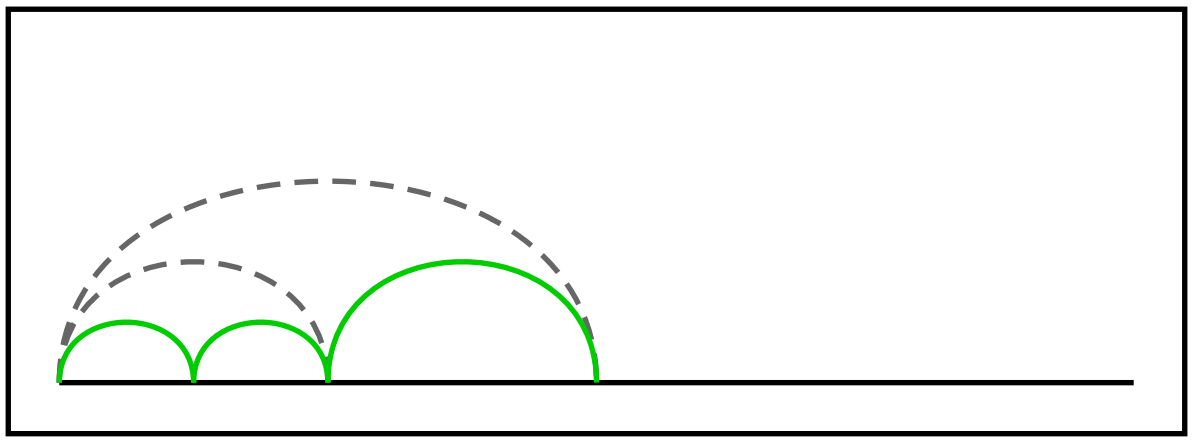
$\int_{[O]}$ – Outer Integral:



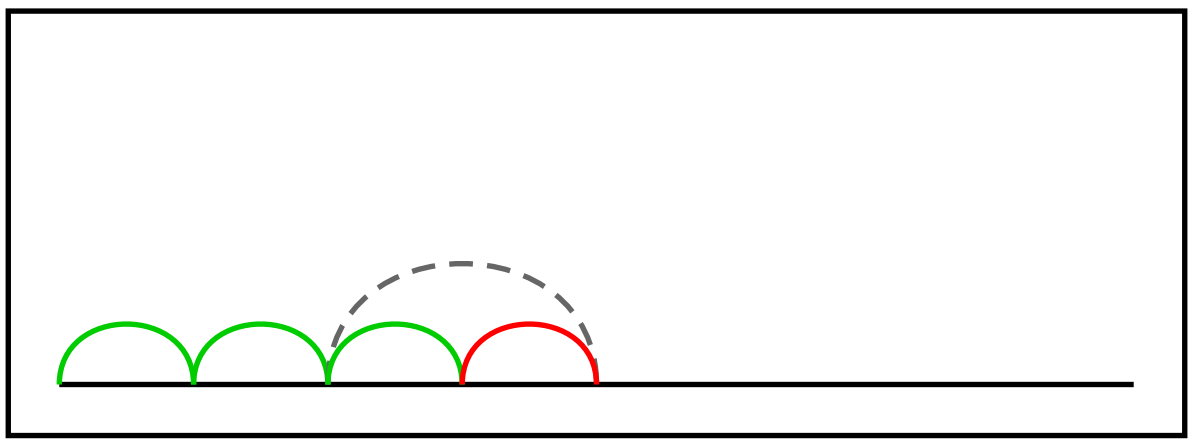
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



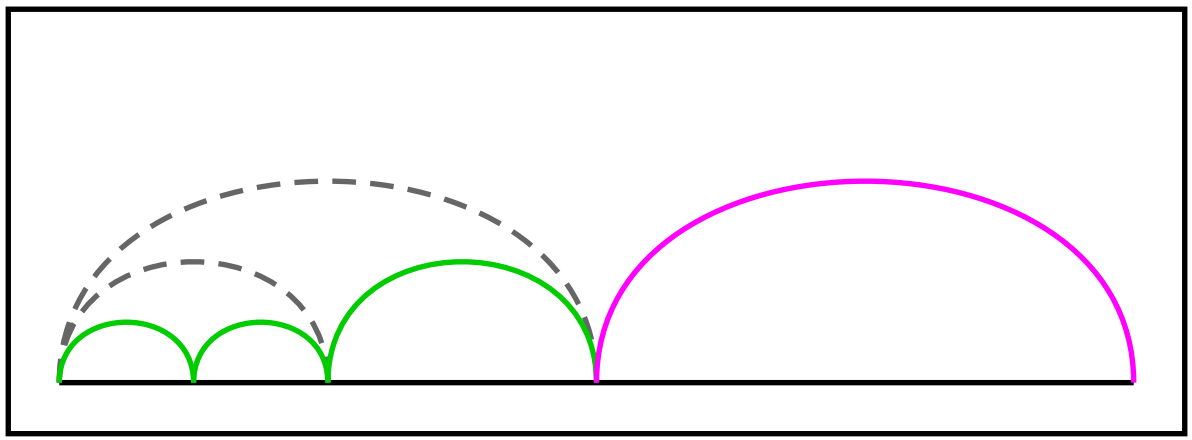
$\int_{[O]}$ – Outer Integral:



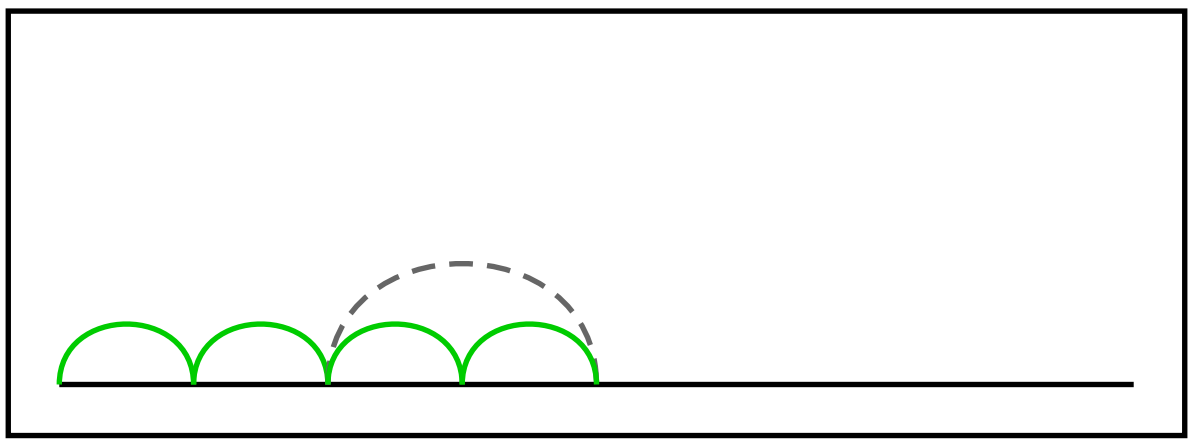
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



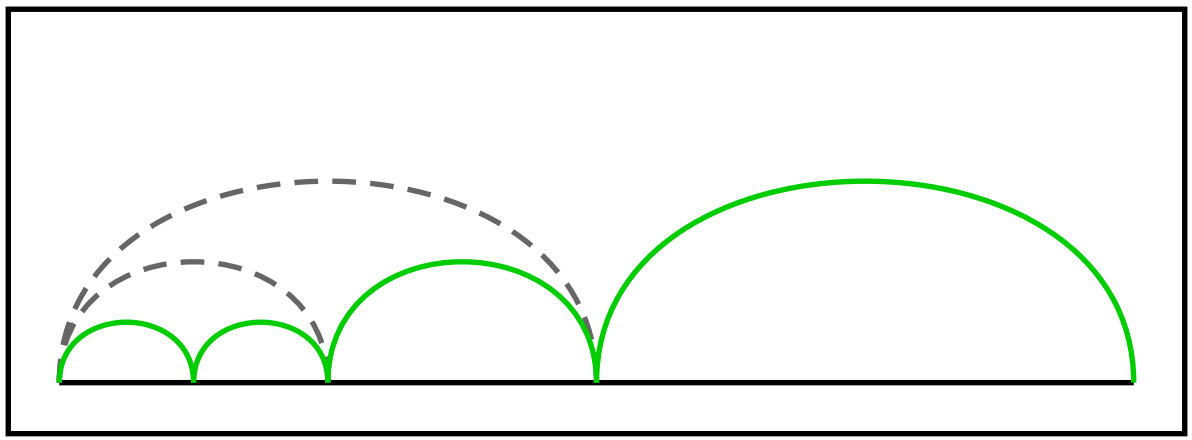
$\int_{[O]}$ – Outer Integral:



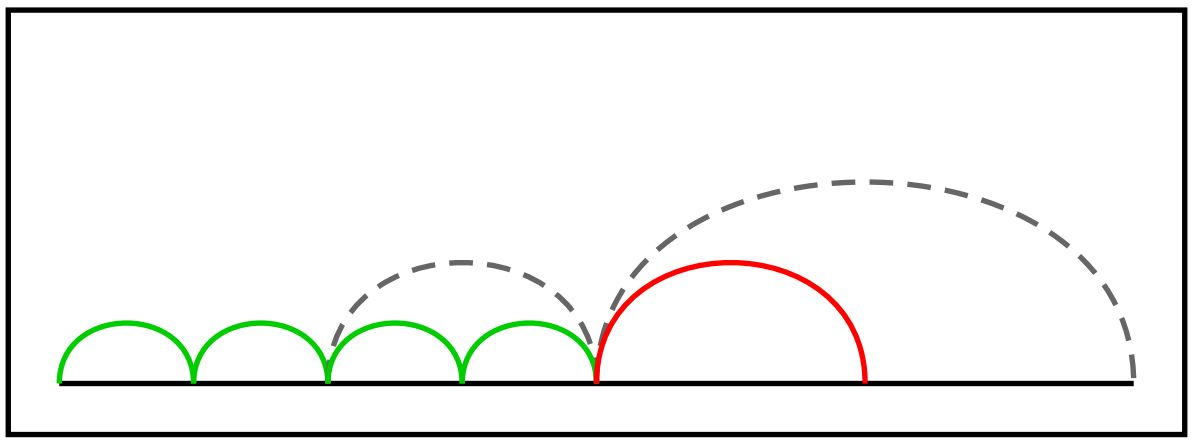
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



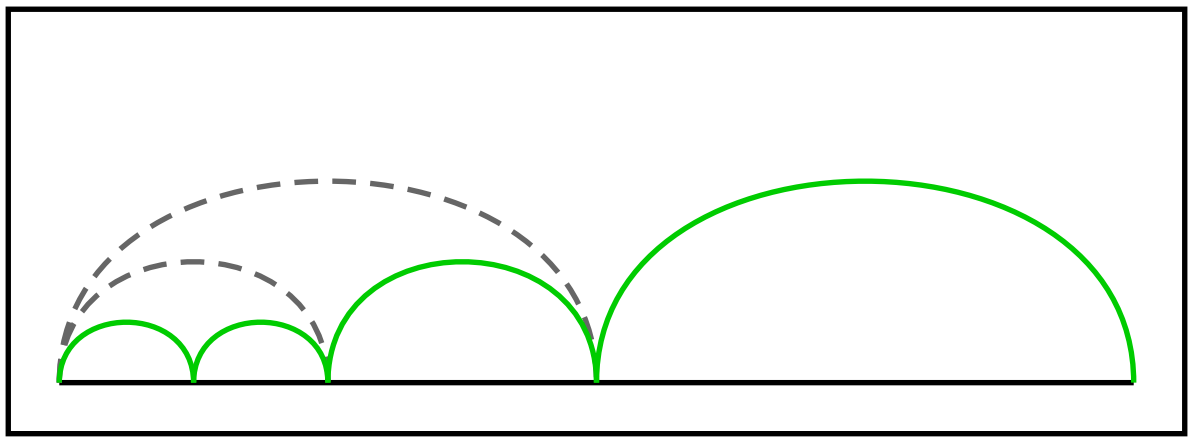
$\int_{[O]}$ – Outer Integral:



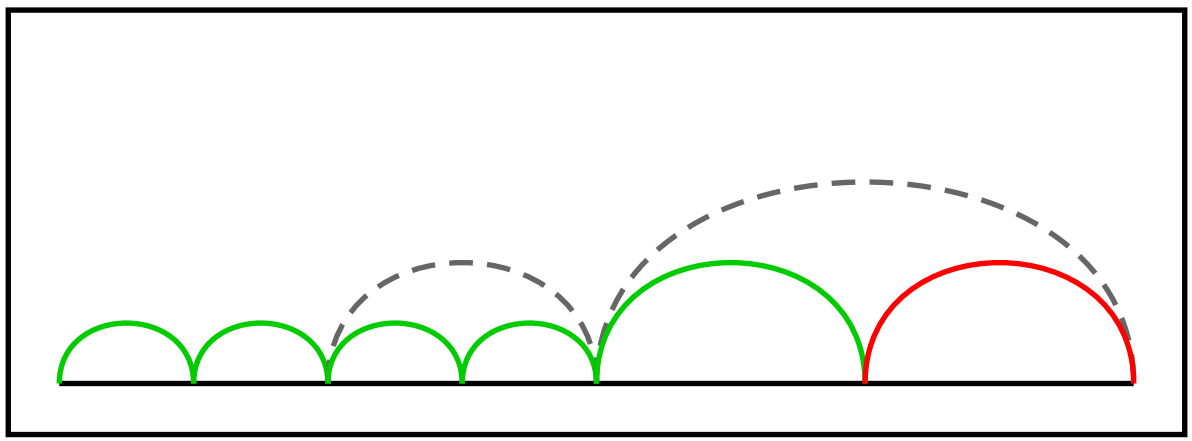
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



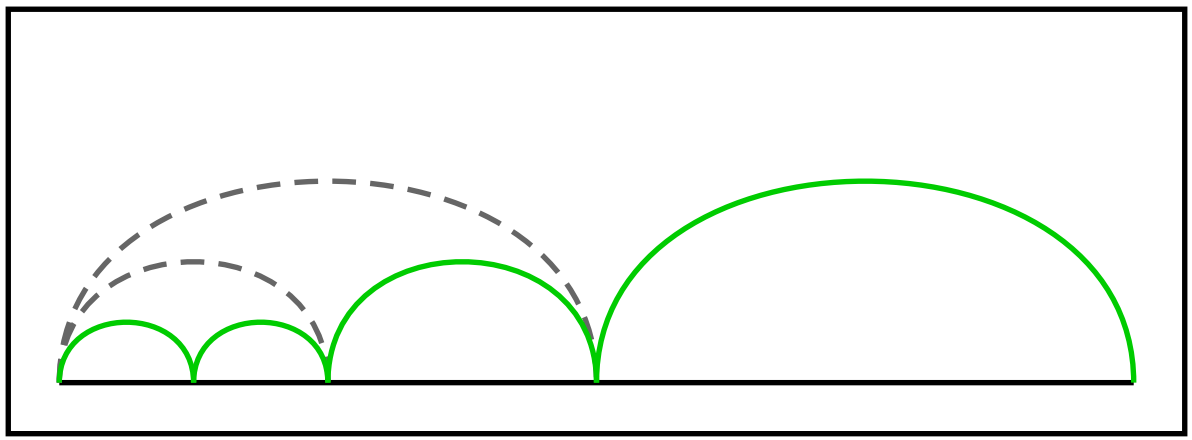
$\int_{[O]}$ – Outer Integral:



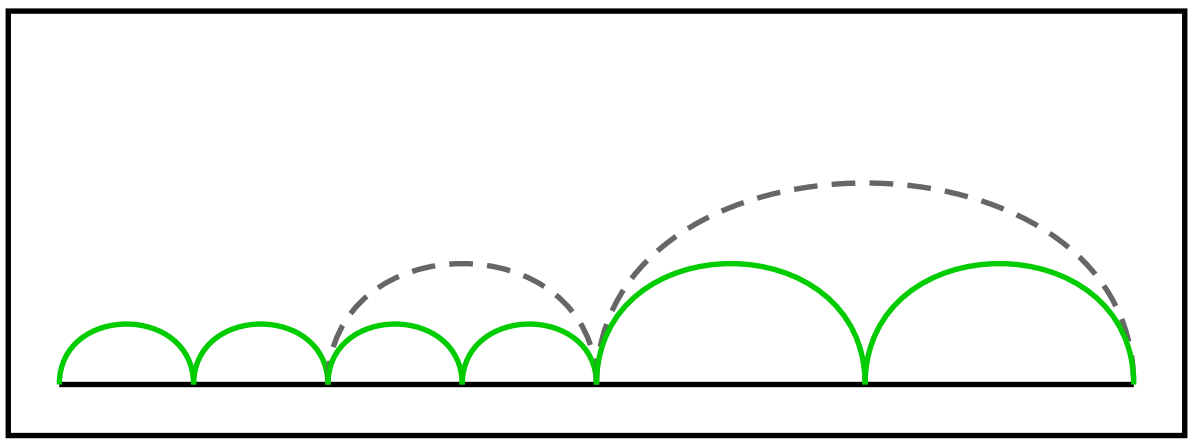
Computing VRI with our iterative process

Calculate the both intergral in consonance:

$\int_{[I]}$ – Inner Integral:



$\int_{[O]}$ – Outer Integral:



Computing VRI - In summary

- Inner integral – $\int_{[I]}$:
  - The process ends when a first valid step is found.
  - Returns approximation of the integral and size of last step.
  - If zero, the outer integral doesn't need to be computed.

- Outer integral – $\int_{[O]}$:
  - Iterates along all over interval defined by h_{inn}.
  - Returns approximation of the integral and size of last step.
  - For each outer integral evaluation, the inner integral is approximated using a non-adaptive Simpson's rule.
  - Integrates all color channels ($RGBA$).

Proposed algorithm to compute VRI

Input: $t, \Delta t, h_0, \epsilon$
Output: VRI $(t, \Delta t)$

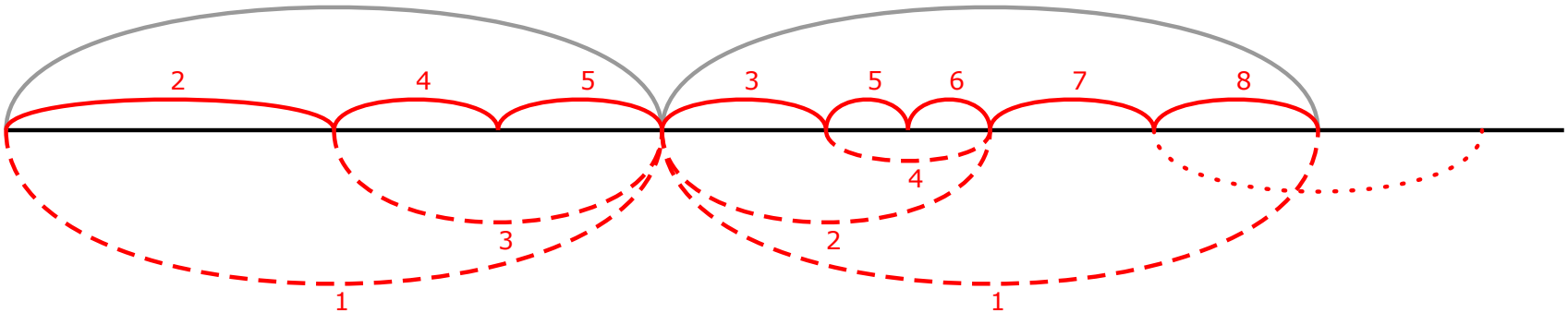
$I = [R = 0, G = 0, B = 0, A = 0]$
$t_{final} = t + \Delta t$
$h_{inn} = \min(h_0, t_{final} - t)$

while $t < t_{final}$ **do**
  $I_{inn}, h_{inn} = \int_{[I]}(t, t_{final} - t, h_{inn})$
  if $I_{inn} \neq 0$ **then**
    $h_{out} = h_{inn}$
    $I_{out}, h_{out} = \int_{[O]}(t, h_{inn}, h_{out})$
    $I = I + I_{out}$
  end if
end while
return I

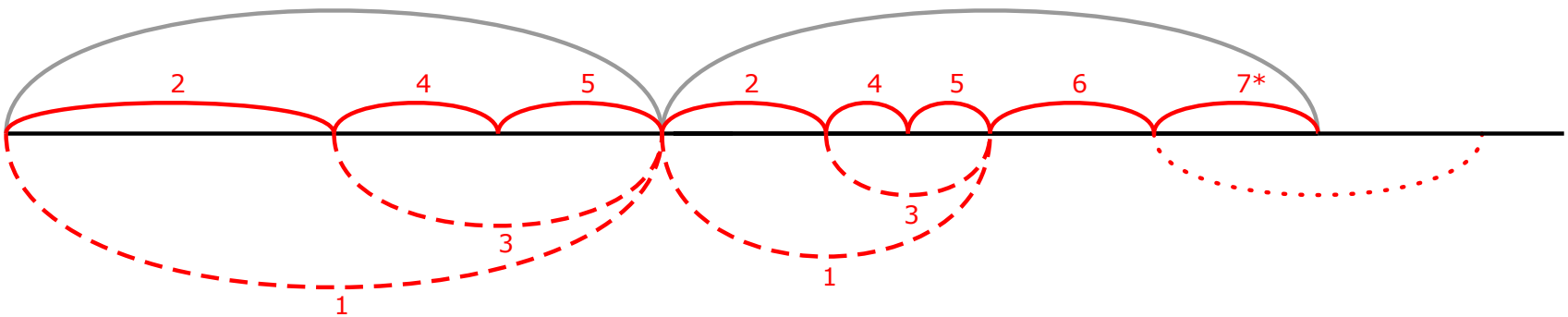
Coupled/Decoupled strategy

Calculate VRI using Coupled and Decoupled strategy:

Coupled strategy:



Decoupled strategy:



Our proposal with decoupled strategy

Input: $t, \Delta t, h_0, \epsilon$
Output: VRI $(t, \Delta t)$

$I = [R = 0, G = 0, B = 0, A = 0]$
$t_{final} = t + \Delta t$
$h_{inn} = \min(h_0, t_{final} - t)$
$h_{out} = h_{inn}$
while $t < t_{final}$ **do**
  $I_{inn}, h_{inn} = \int_{[I]}(t, t_{final} - t, h_{inn})$
  if $I_{inn} \neq 0$ **then**

    $I_{out}, h_{out} = \int_{[O]}(t, h_{inn}, h_{out})$
    $I = I + I_{out}$
  end if
end while
return I

Proposed algorithm to compute VRI

Input: $t, \Delta t, h_0, \epsilon$
Output: VRI $(t, \Delta t)$

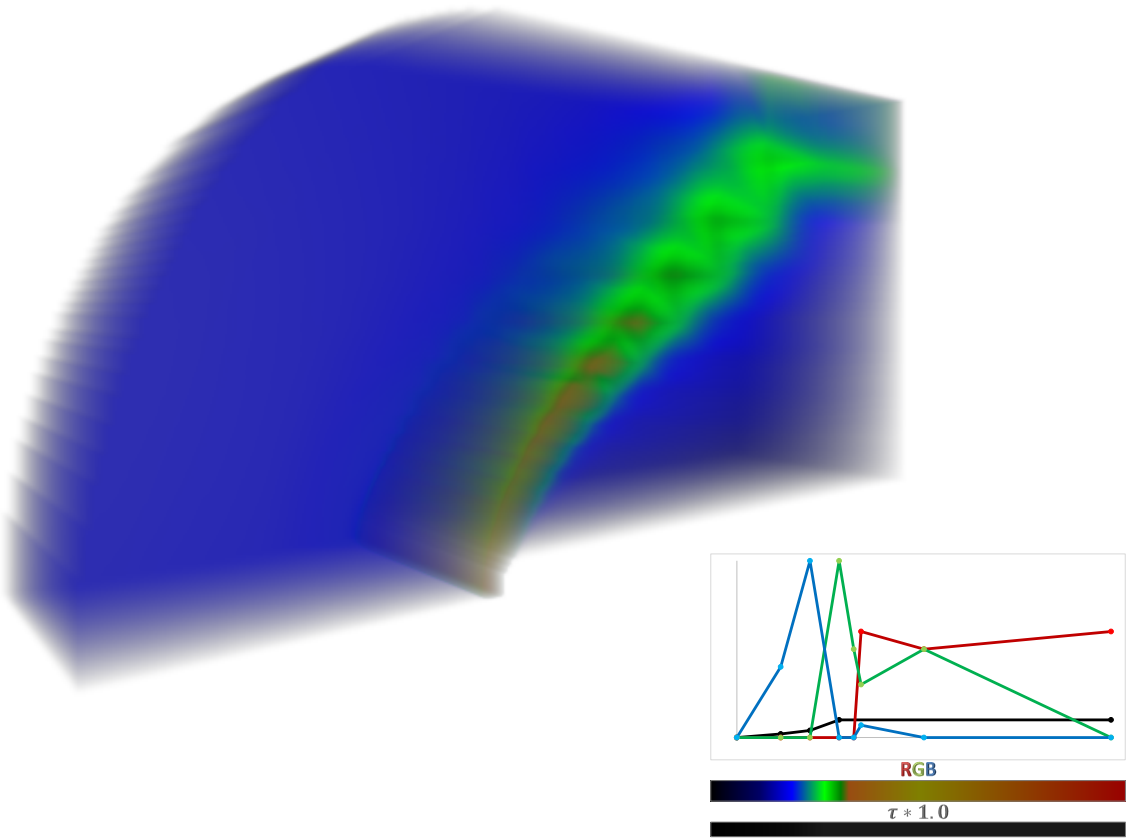
$I = [R = 0, G = 0, B = 0, A = 0]$
$t_{final} = t + \Delta t$
$h_{inn} = \min(h_0, t_{final} - t)$
$h_{out} = h_{inn}$ {only on decoupled}
while $t < t_{final}$ **do**
  $I_{inn}, h_{inn} = \int_{[I]}(t, t_{final} - t, h_{inn})$
  if $I_{inn} \neq 0$ **then**
    $h_{out} = h_{inn}$ {only on coupled}
    $I_{out}, h_{out} = \int_{[O]}(t, h_{inn}, h_{out})$
    $I = I + I_{out}$
  end if
end while
return I

Results – Comparisons

- Riemann Summation: $[Riemann_h / R_h]$
  - $h = 0.5, 0.4, 0.3, 0.2, 0.1$ voxel units
- Our Proposal: $[Coupled_\epsilon / C_\epsilon$ e $Decoupled_\epsilon / D_\epsilon]$
  - $h_0 = 0.5$ voxel units
  - $h_{min} = 0.1$ voxel units
  - $h_{max} = 2.0$ voxel units
  - $\epsilon = 0.01, 0.005, 0.001$
- Ground Truth: $Riemann_{0.01}$

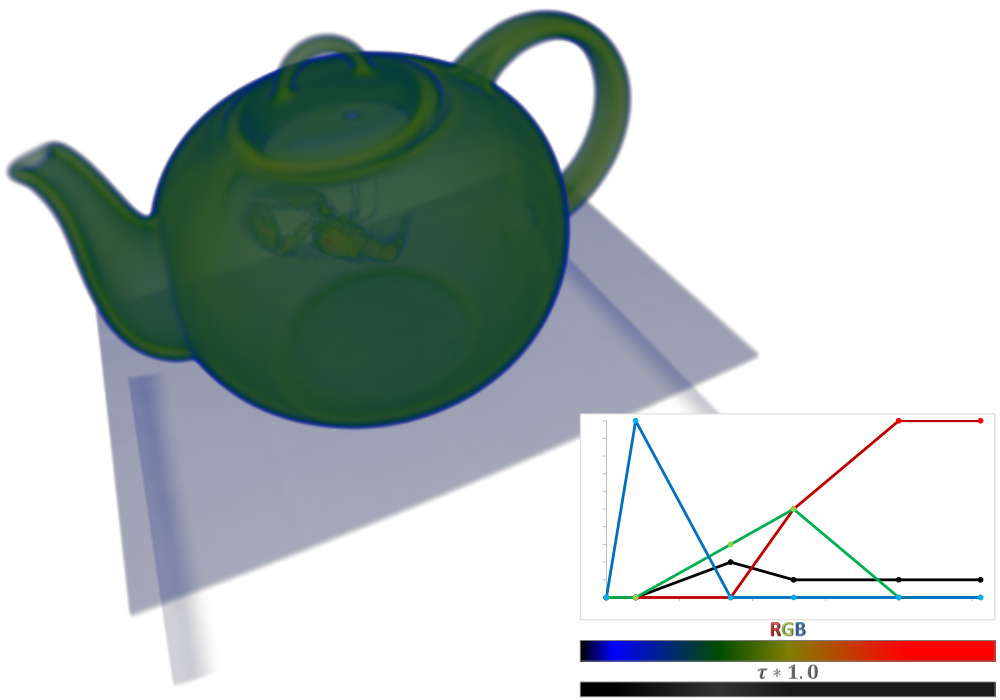
Models – Blunt Fin

256x128x64 voxels, 268207 evaluated rays



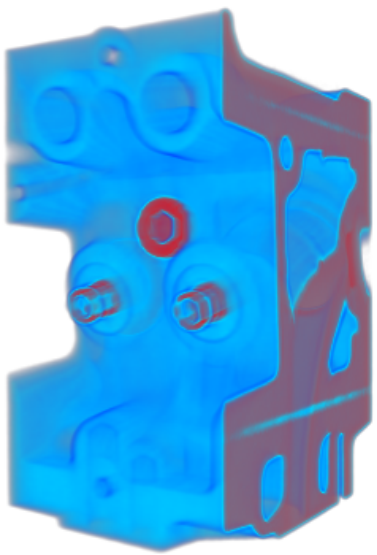
Models – Boston Teapot

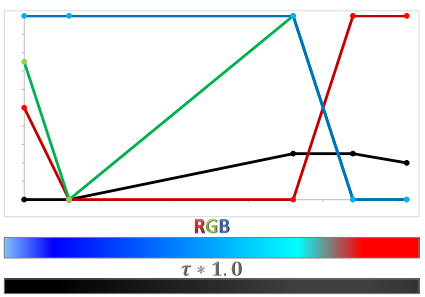
256x256x178 voxels, 270101 evaluated rays



Models – Engine

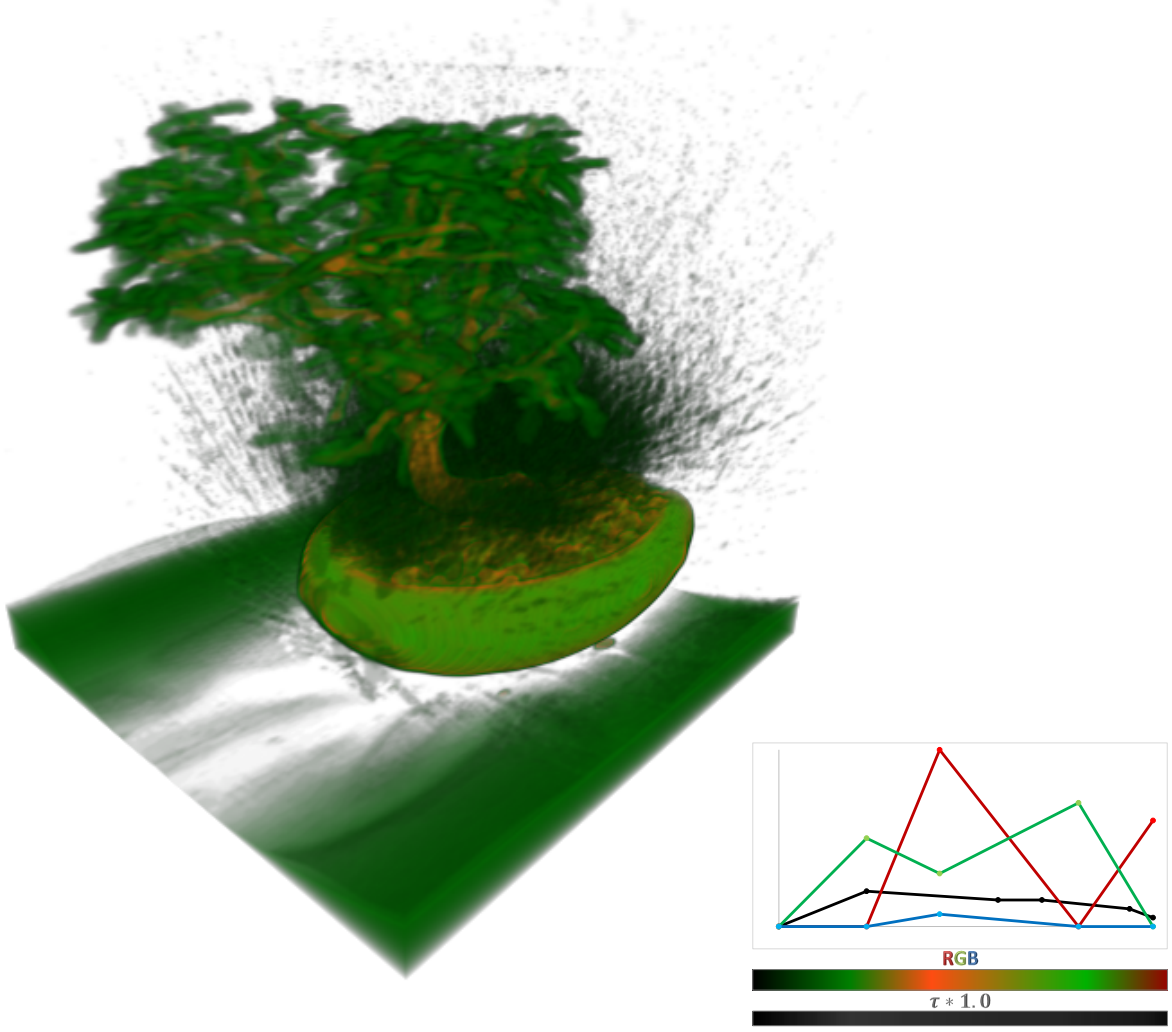
256x256x128 voxels, 177373 evaluated rays





Models – Bonsai

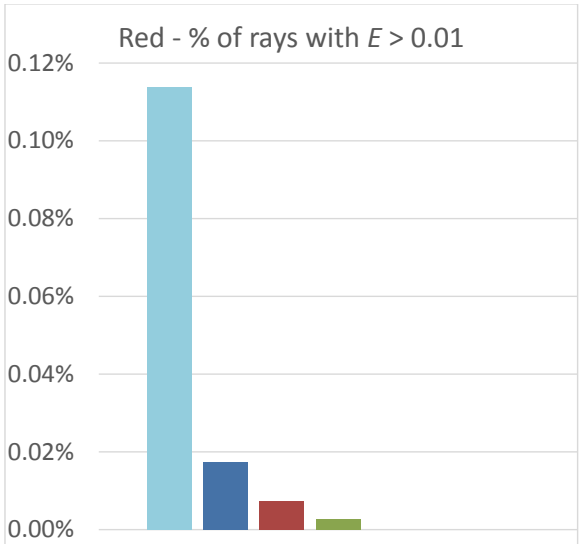
256x256x256 voxels, 223092 evaluated rays

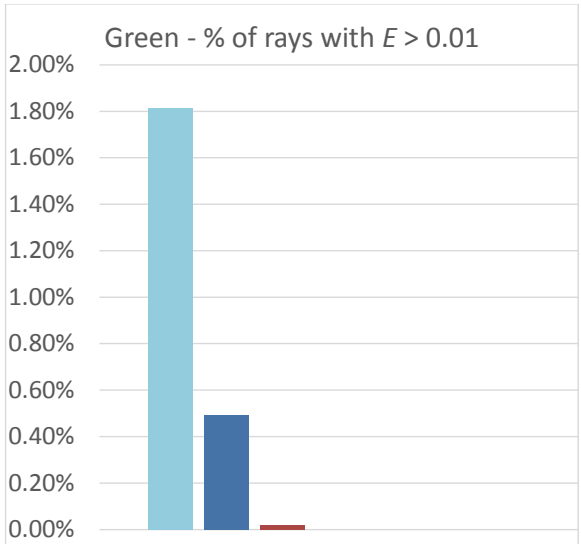


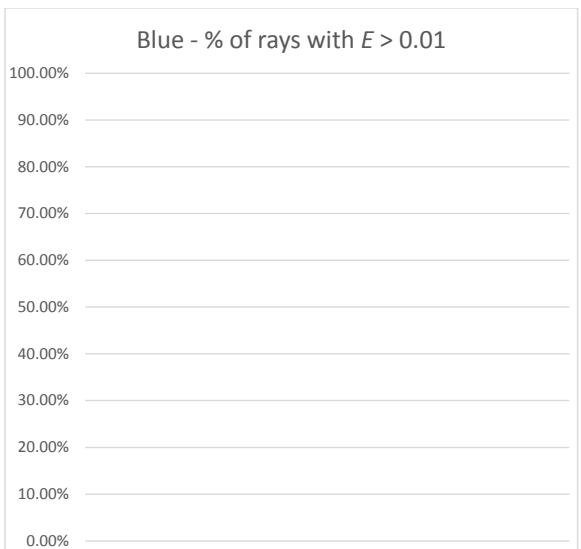
Precision results – Bonsai ($\epsilon = 0.01$)

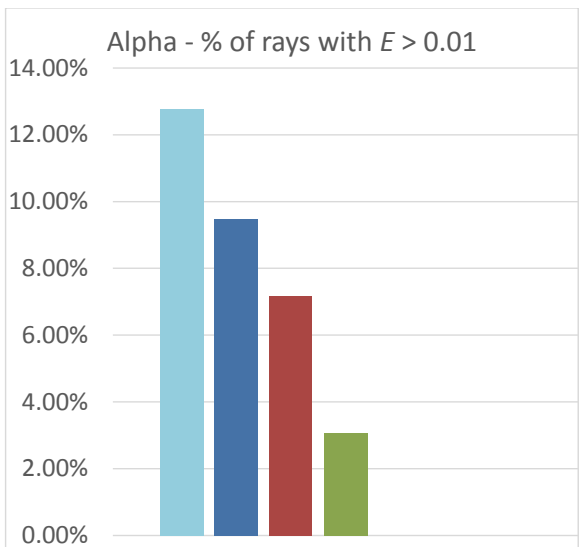
Percentage of rays that exceeds the defined tolerance.

- Riemann 0.5
- Riemann 0.4
- Riemann 0.3
- Riemann 0.2
- Riemann 0.1
- Coupled 0.01
- Decoupled 0.01

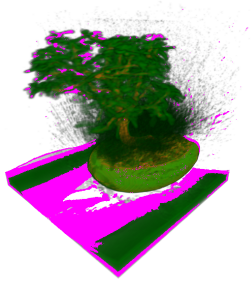




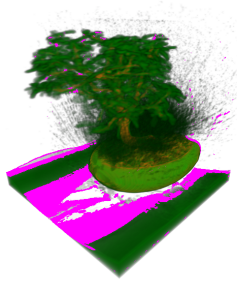




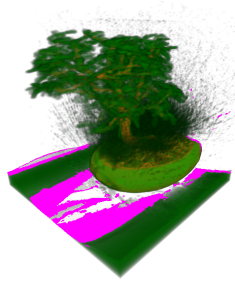
Precision Results – Bonsai ($\epsilon = 0.01$)



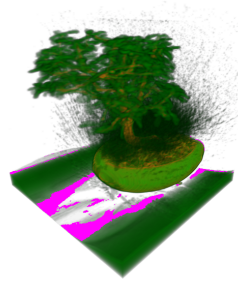
$Riemann_{0.5}$



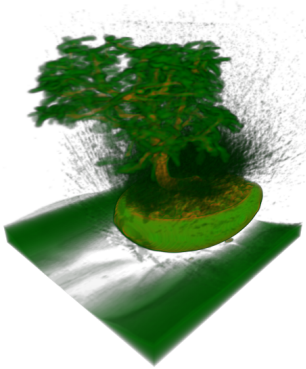
$Riemann_{0.4}$



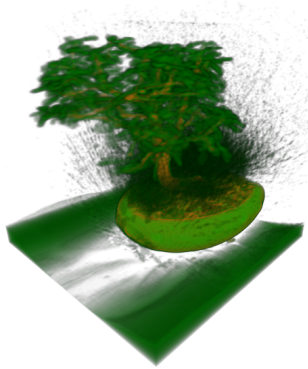
$Riemann_{0.3}$



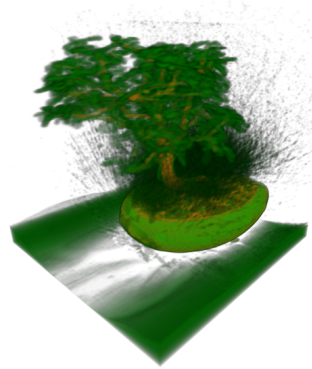
$Riemann_{0.2}$



$Riemann_{0.1}$



$Coupled_{0.01}$

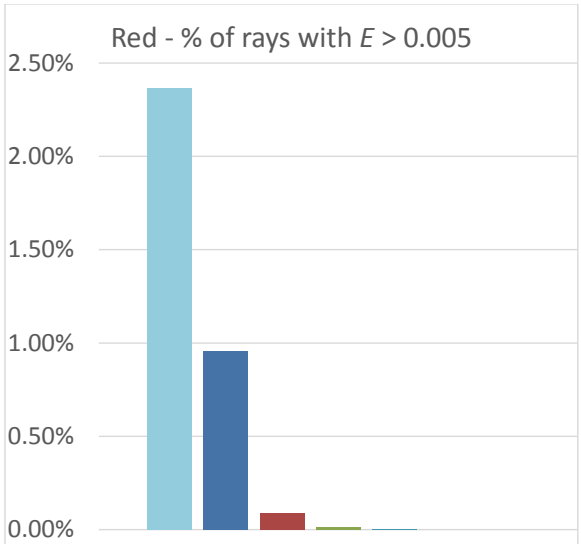


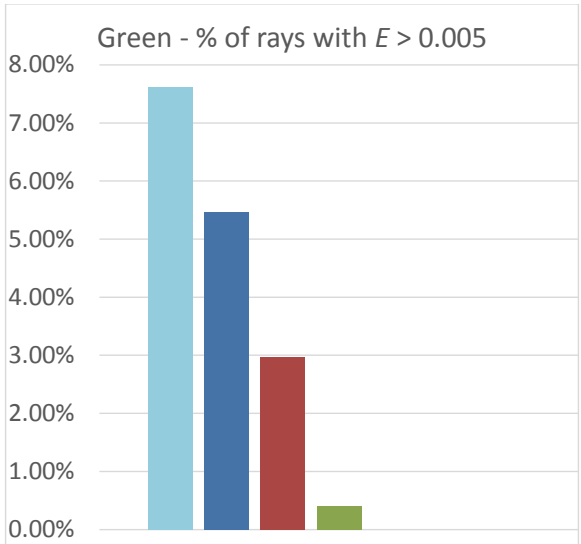
$Decoupled_{0.01}$

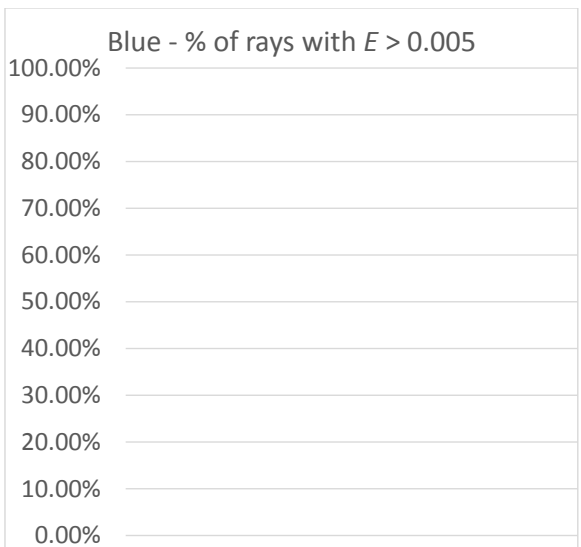
Precision Results – Bonsai ($\epsilon = 0.005$)

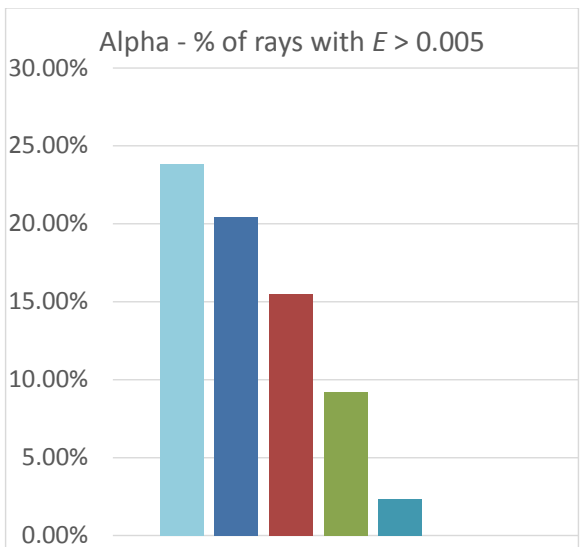
Percentage of rays that exceeds the defined tolerance.

- Riemann 0.5
- Riemann 0.4
- Riemann 0.3
- Riemann 0.2
- Riemann 0.1
- Coupled 0.005
- Decoupled 0.005

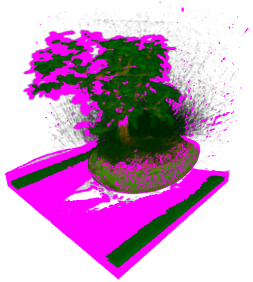




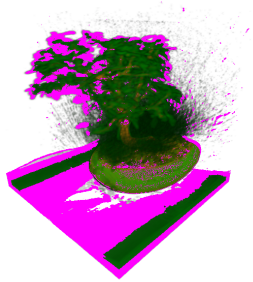




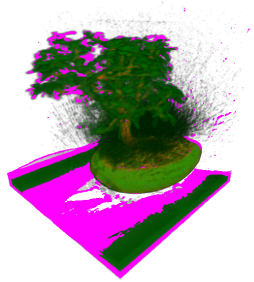
Precision Results – Bonsai ($\epsilon = 0.005$)



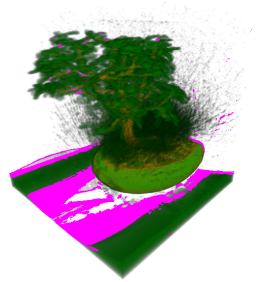
$Riemann_{0.5}$



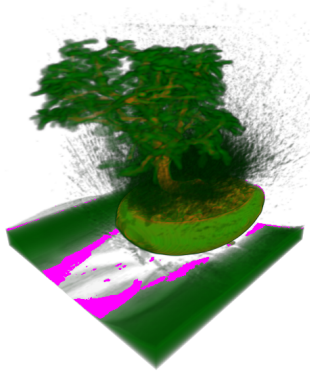
$Riemann_{0.4}$



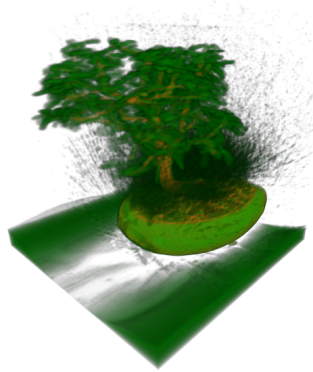
$Riemann_{0.3}$



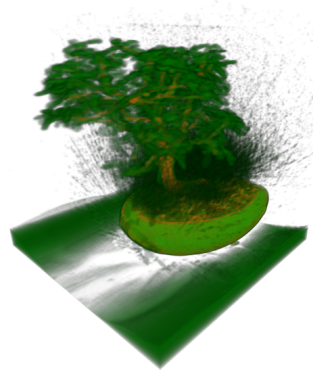
$Riemann_{0.2}$



$Riemann_{0.1}$



$Coupled_{0.005}$

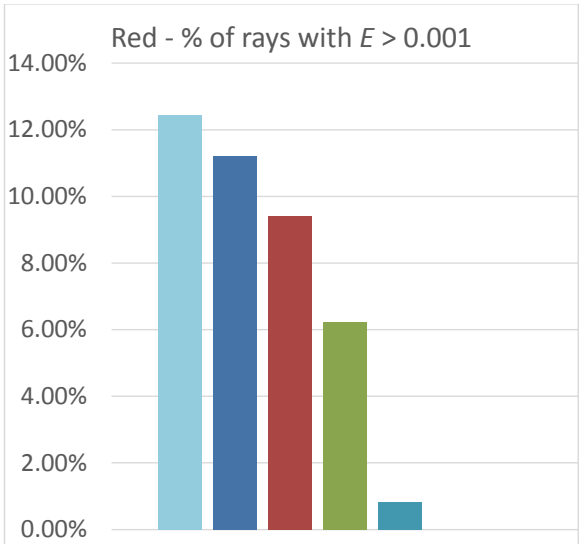


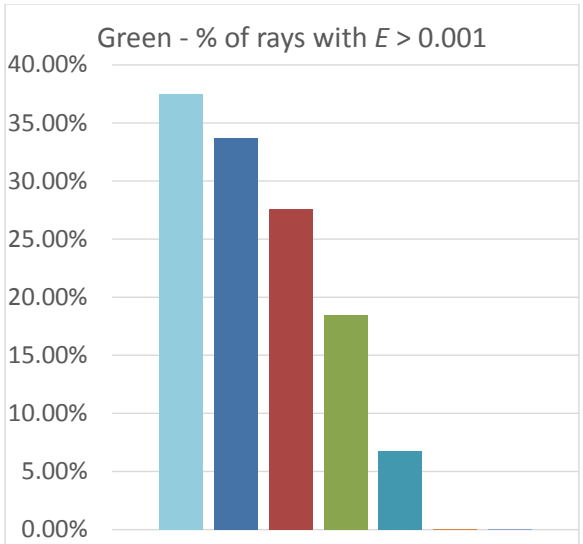
$Decoupled_{0.005}$

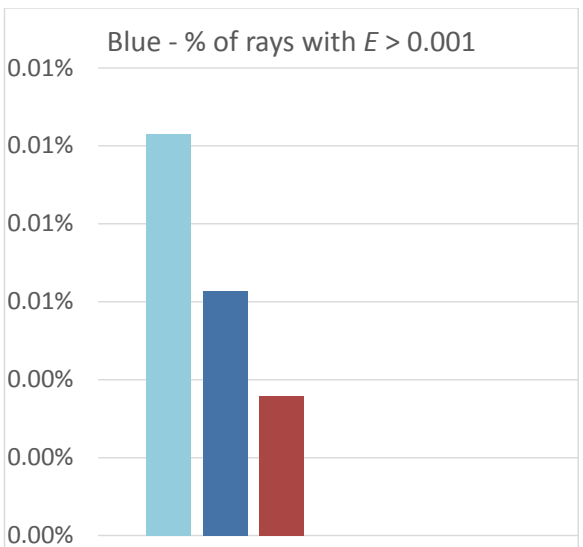
Precision Results – Bonsai ($\epsilon = 0.001$)

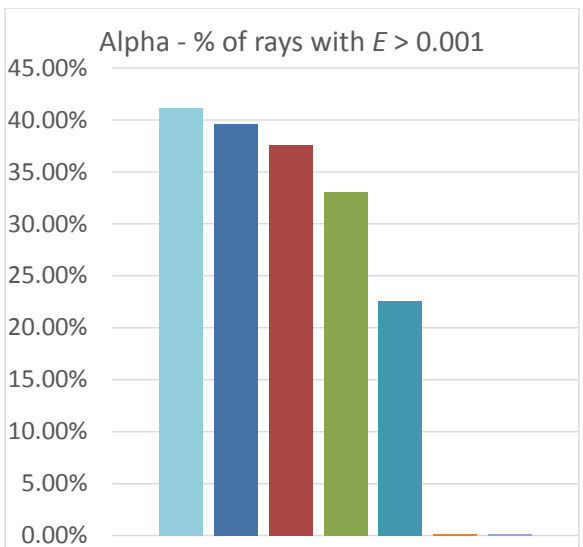
Percentage of rays that exceeds the defined tolerance.

- Riemann 0.5
- Riemann 0.4
- Riemann 0.3
- Riemann 0.2
- Riemann 0.1
- Coupled 0.001
- Decoupled 0.001

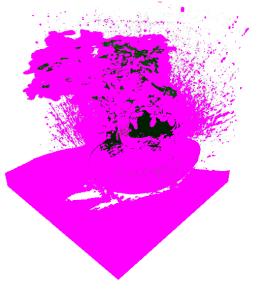




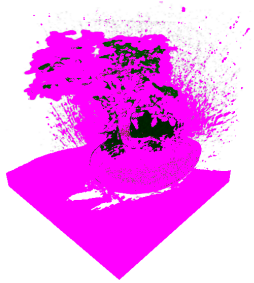




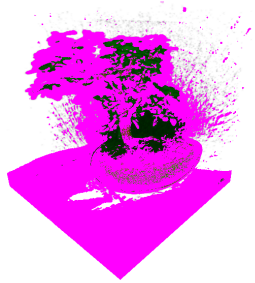
Precision Results – Bonsai ($\epsilon = 0.001$)



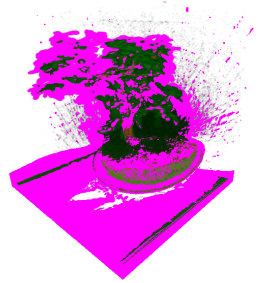
$Riemann_{0.5}$



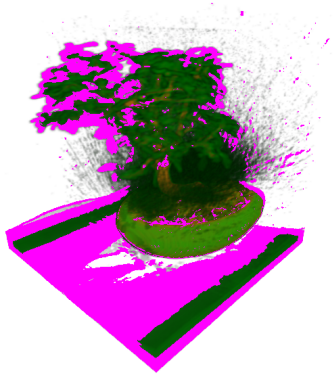
$Riemann_{0.4}$



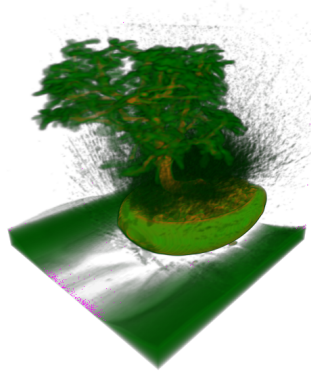
$Riemann_{0.3}$



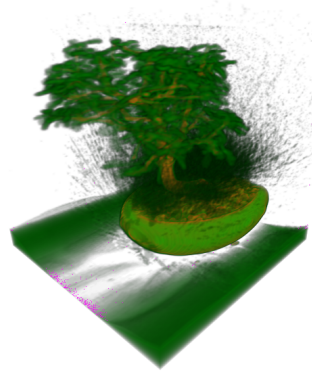
$Riemann_{0.2}$



$Riemann_{0.1}$



$Coupled_{0.001}$



$Decoupled_{0.001}$

Performance Results – CPU (seconds)

- Higher performance.
- Higher accuracy.

Bonsai

	time	error in %			
		R	G	B	A
Riemann ($h = 0.5$)	16.23	2.3645	7.6170	0	23.8579
Riemann ($h = 0.4$)	20.03	0.9548	5.4722	0	20.4068
Riemann ($h = 0.3$)	26.31	0.0883	2.9643	0	15.4645
Riemann ($h = 0.2$)	38.94	0.0121	0.4016	0	9.2177
Riemann ($h = 0.1$)	76.80	0.0004	0.0000	0	2.3614
Coupled ($\epsilon = 0.005$)	23.61	0.0000	0.0000	0	0.0000
Decoupled ($\epsilon = 0.005$)	23.72	0.0000	0.0000	0	0.0000

Performance Results – GPU (frame rate)

- Adaptiveness was less effective on performance.

	Blunt Fin	Bonsai
Riemann ($h = 0.5$)	603	68
Riemann ($h = 0.4$)	494	55
Riemann ($h = 0.3$)	390	42
Riemann ($h = 0.2$)	270	28
Riemann ($h = 0.1$)	143	**14**
Coupled ($\epsilon = 0.005$)	367	23
Decoupled ($\epsilon = 0.005$)	335	22
Coupled ($\epsilon = 0.001$)	226	**9**
Decoupled ($\epsilon = 0.001$)	205	**9**

Performance Results – GPU (frame rate)

- Tunning: Aims to achieve better performance on GPUs.
- Variables of step-size range control tunned:
  - $h_{min} = 0.4$ (0.1 before).
  - $h_{max} = 4.0$ (2.0 before).

	Blunt Fin	Bonsai	Tunned Bonsai
Riemann ($h = 0.5$)	603	68	68
Riemann ($h = 0.4$)	494	55	55
Riemann ($h = 0.3$)	390	42	42
Riemann ($h = 0.2$)	270	28	28
Riemann ($h = 0.1$)	143	14	**14**
Coupled ($\epsilon = 0.005$)	367	23	-
Decoupled ($\epsilon = 0.005$)	335	22	-
Coupled ($\epsilon = 0.001$)	226	9	**19**
Decoupled ($\epsilon = 0.001$)	205	9	**18**

Final considerations

- The proposed algorithm maintains the error under a predefined tolerance, using the additional parameters to control each step.
- The CPU implementation maintains good performance and accuracy, compairing with riemann summation.
- On GPU, the parameters can be tuned to achieve a competitive result.
- Allows balacing between accuracy and performance by changing the parameters (ϵ, h_{min}, h_{max}).
- The proposed algorithm adapts itself based on the given transfer function.

Future work

- Unstructured meshes.
- Add color perception principles to better control the quality of the images.
- Illumination algorithm.
- How to better evaluate the perceptual error?
- How the tolerance of the inner integral affects the final image?

Thank You!

Accurate Volume Rendering based on Adaptive Numerical Integration

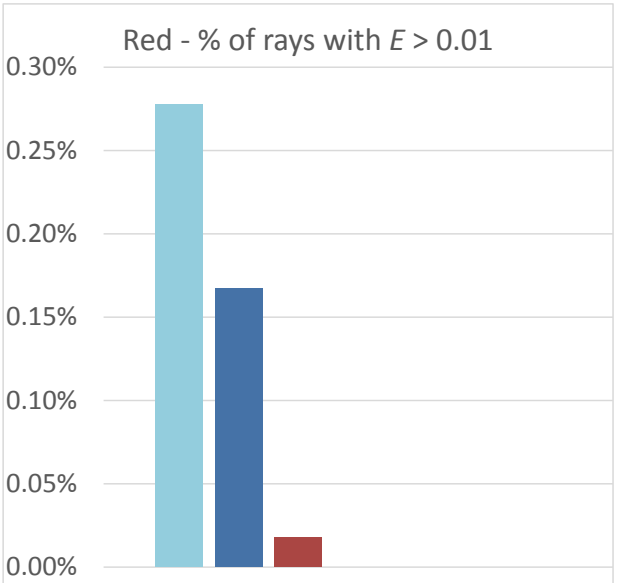
Leonardo Quatrin Campagnolo
Waldemar Celes Filho
Luiz Henrique de Figueiredo

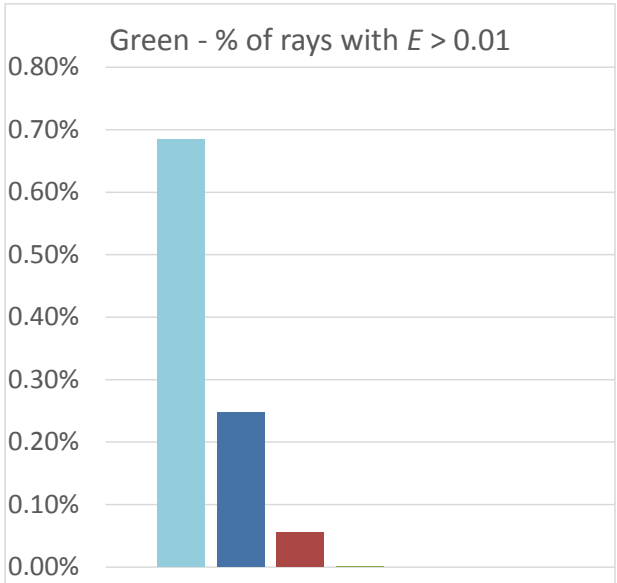
Sponsored by CNPq & TecGraf.

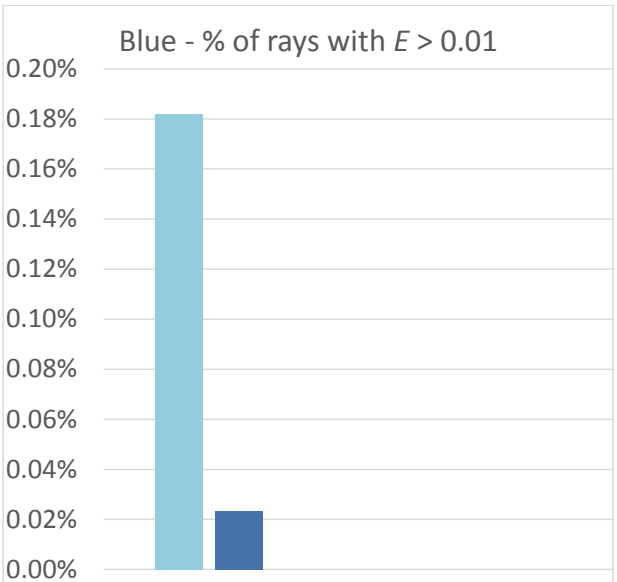
Questions?

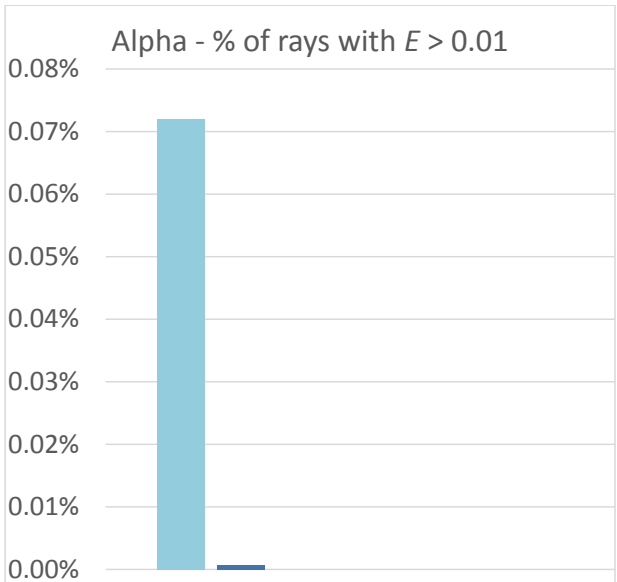
Precision Results – Blunt Fin ($\epsilon = 0.01$)

- Riemann 0.5
- Riemann 0.4
- Riemann 0.3
- Riemann 0.2
- Riemann 0.1
- Coupled 0.01
- Decoupled 0.01



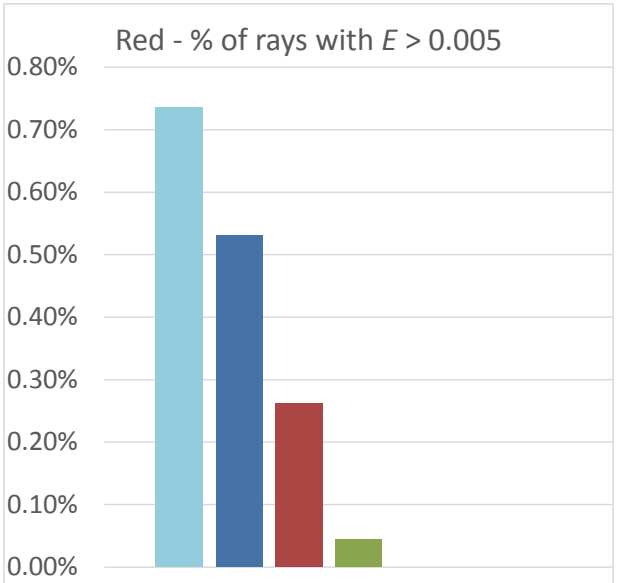


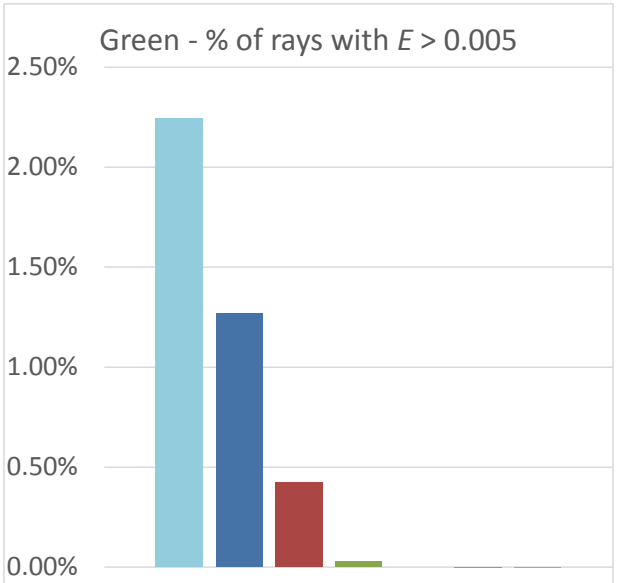


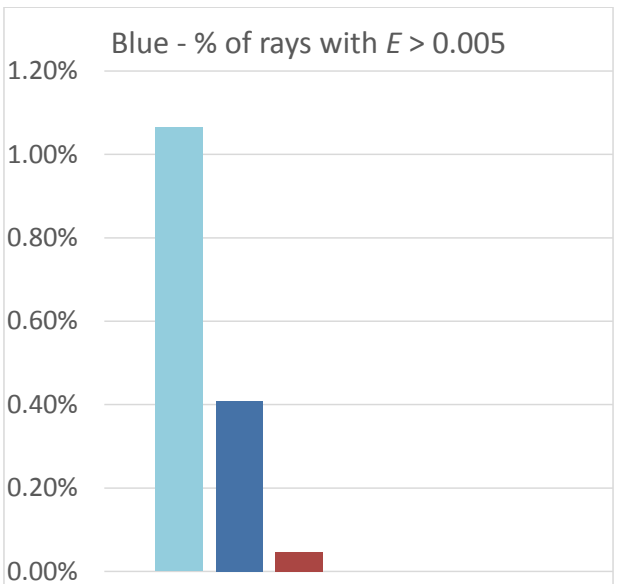


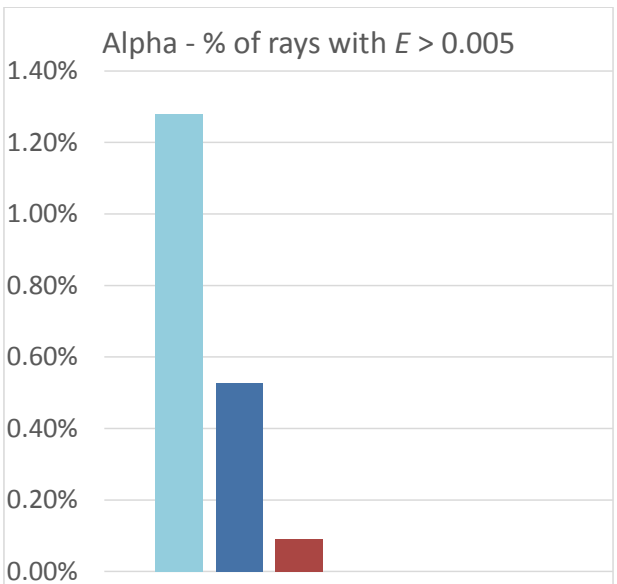
Precision Results – Blunt Fin ($\epsilon = 0.005$)

- Riemann 0.5
- Riemann 0.4
- Riemann 0.3
- Riemann 0.2
- Riemann 0.1
- Coupled 0.005
- Decoupled 0.005



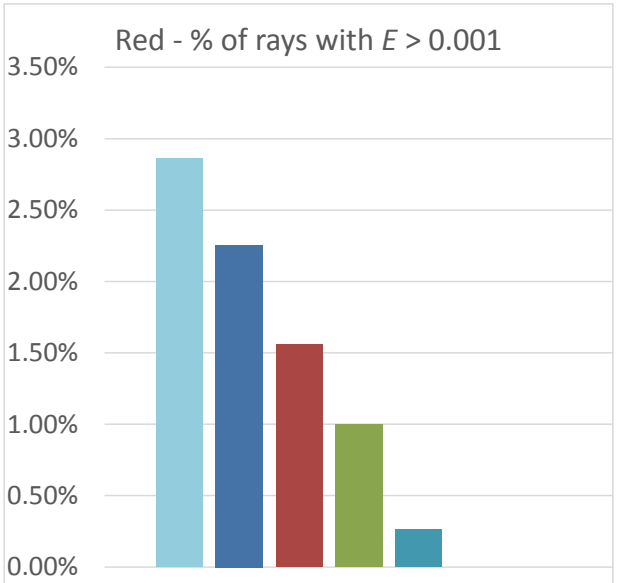


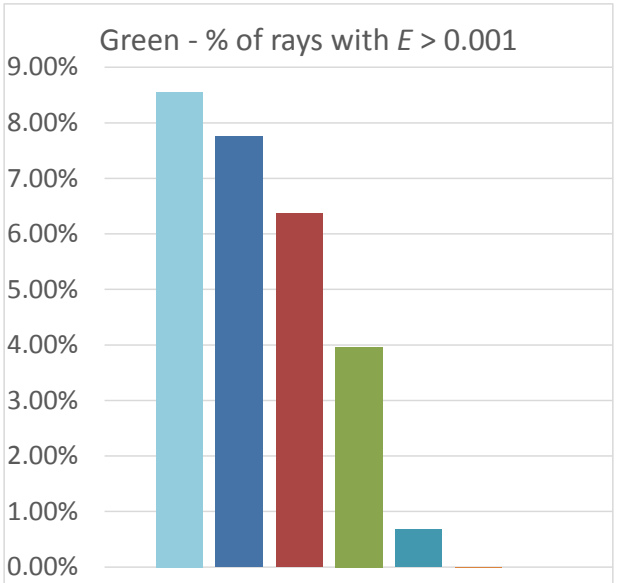


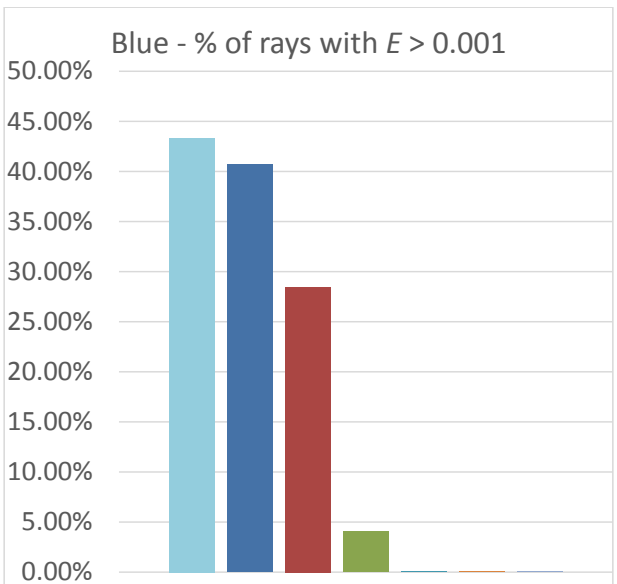


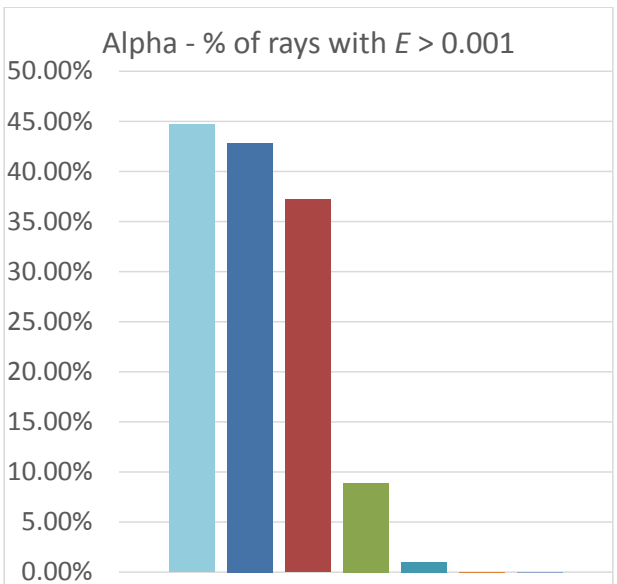
Precision Results – Blunt Fin ($\epsilon = 0.001$)

- Riemann 0.5
- Riemann 0.4
- Riemann 0.3
- Riemann 0.2
- Riemann 0.1
- Coupled 0.001
- Decoupled 0.001

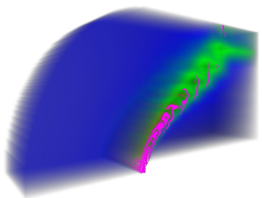




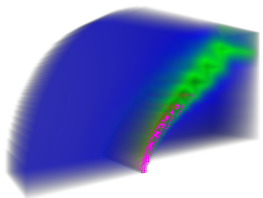




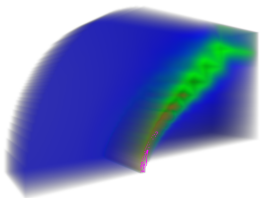
Precision Results – Blunt Fin ($\epsilon = 0.01$)



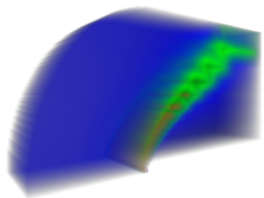
$R_{0.5}$



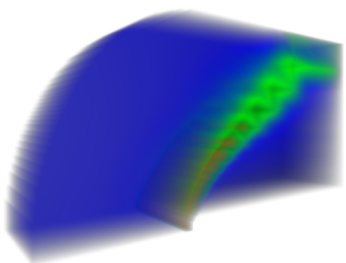
$R_{0.4}$



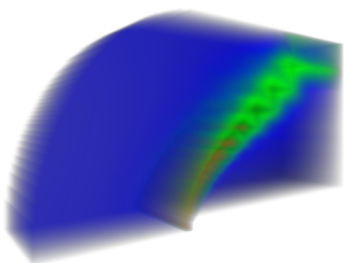
$R_{0.3}$



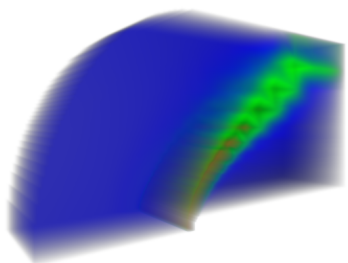
$R_{0.2}$



$R_{0.1}$

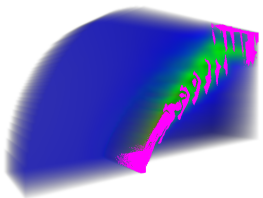


$C_{0.01}$

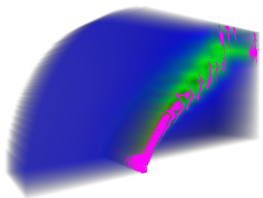


$D_{0.01}$

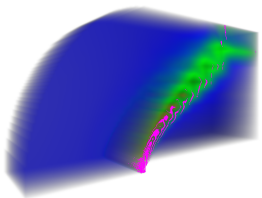
Precision Results – Blunt Fin ($\epsilon = 0.005$)



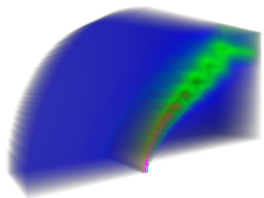
$R_{0.5}$



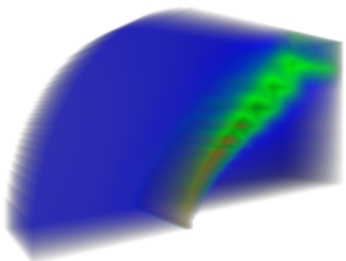
$R_{0.4}$



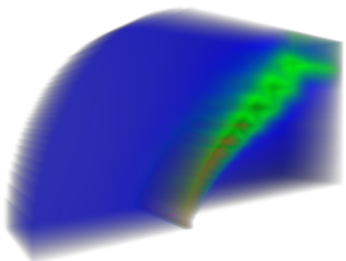
$R_{0.3}$



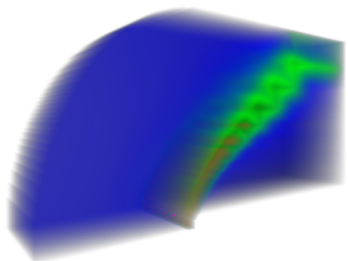
$R_{0.2}$



$R_{0.1}$



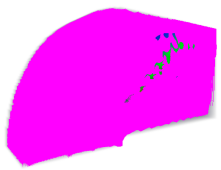
$C_{0.005}$



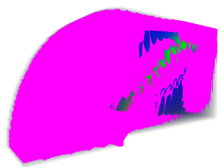
$D_{0.005}$

Precision Results – Blunt Fin ($\epsilon = 0.001$)

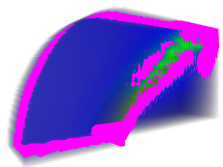
$R_{0.5}$



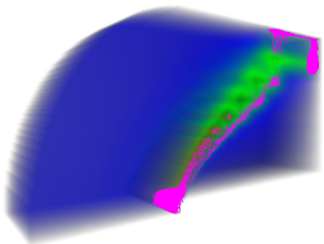
$R_{0.4}$



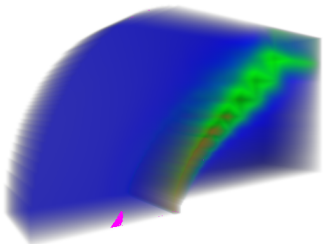
$R_{0.3}$



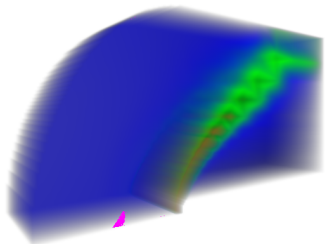
$R_{0.2}$



$R_{0.1}$



$C_{0.001}$



$D_{0.001}$

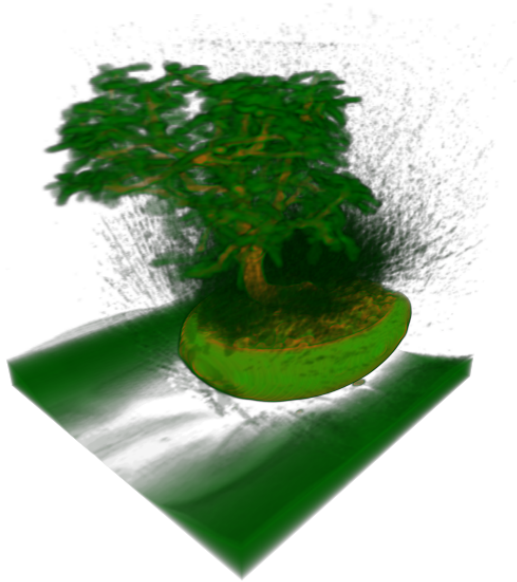
Performance Results – CPU (seconds)

Blunt Fin

	time	error in %			
		R	G	B	A
Riemann ($h = 0.5$)	6.53	0.7352	2.2449	1.0645	1.2796
Riemann ($h = 0.4$)	8.05	0.5306	1.2688	0.4083	0.5261
Riemann ($h = 0.3$)	10.52	0.2625	0.4228	0.0477	0.0910
Riemann ($h = 0.2$)	15.47	0.0436	0.0261	0.0000	0.0000
Riemann ($h = 0.1$)	30.33	0.0000	0.0000	0.0000	0.0000
Coupled ($\epsilon = 0.005$)	9.30	0.0000	0.0007	0.0000	0.0000
Decoupled ($\epsilon = 0.005$)	9.55	0.0000	0.0007	0.0000	0.0000

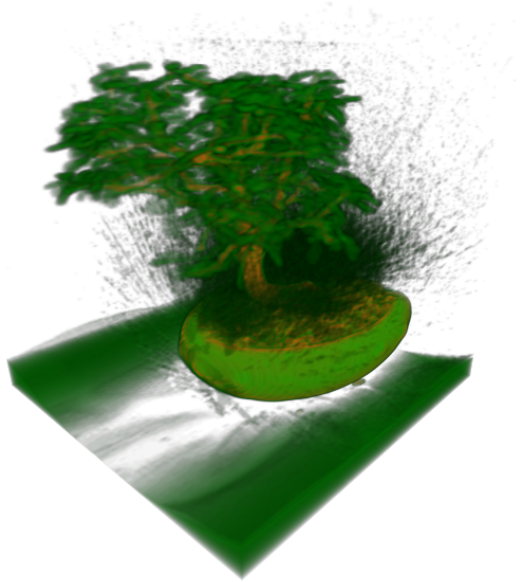
Tunning – Bonsai

- Parameter tunning
  - $h_{min} = 0.1$
  - $h_{max} = 2.0$



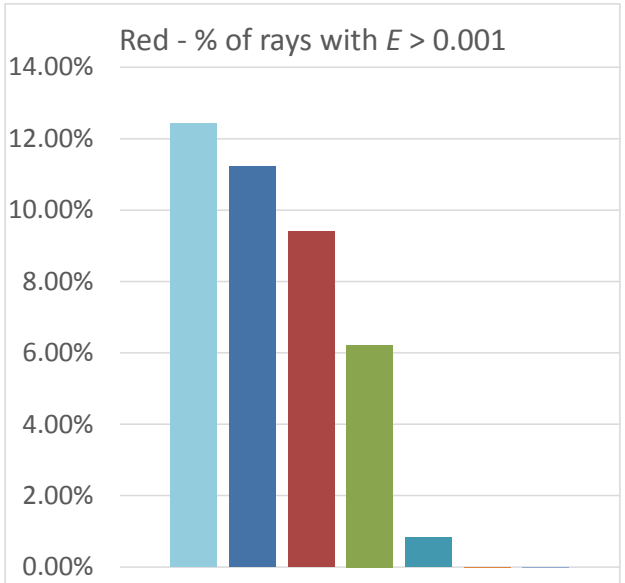
Tunning – Bonsai

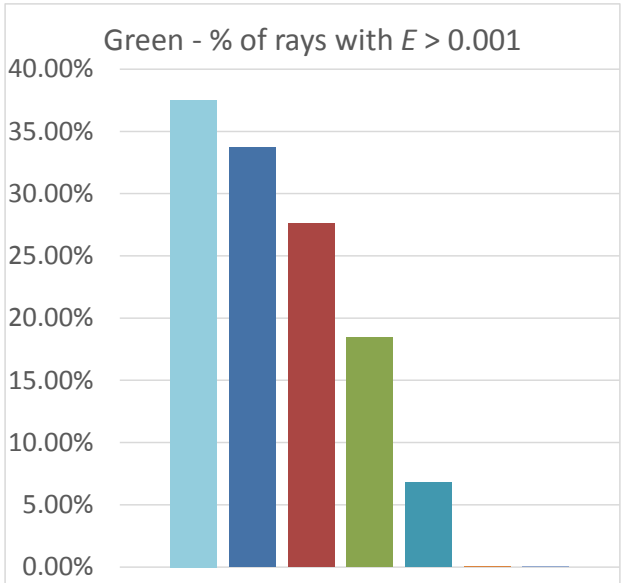
- Parameter tunning
  - $h_{min} = 0.4$ (0.1)
  - $h_{max} = 4.0$ (2.0)

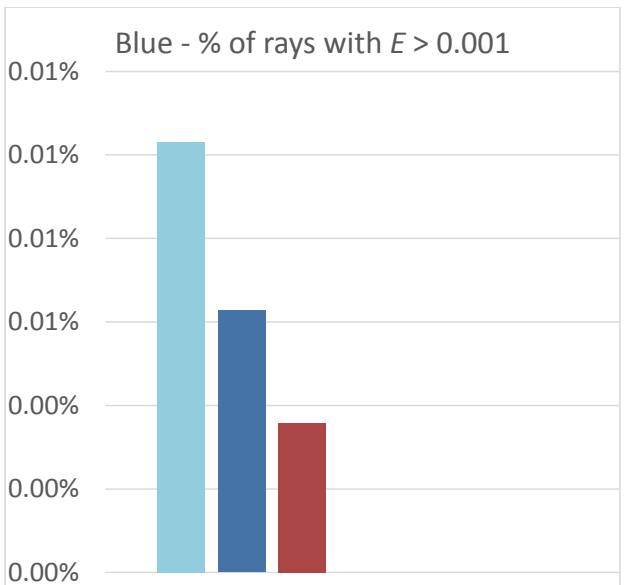


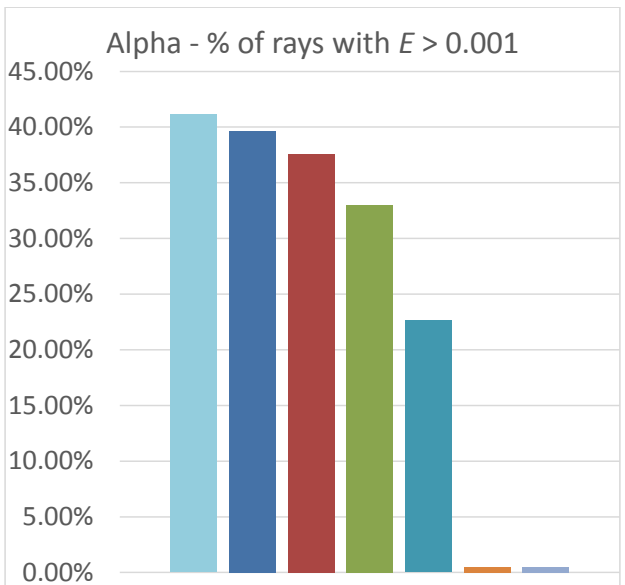
Precision Results – Bonsai ($\epsilon = 0.001$)

- Riemann 0.5
- Riemann 0.4
- Riemann 0.3
- Riemann 0.2
- Riemann 0.1
- Coupled 0.001
- Decoupled 0.001

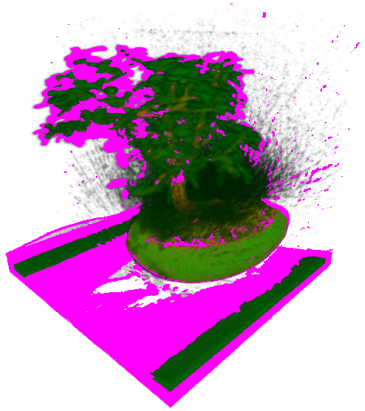




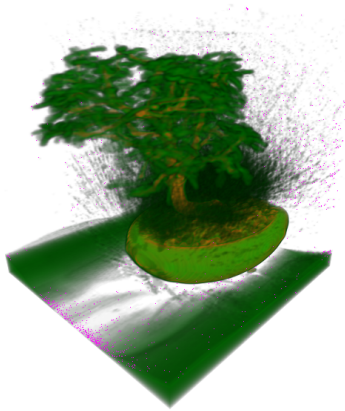




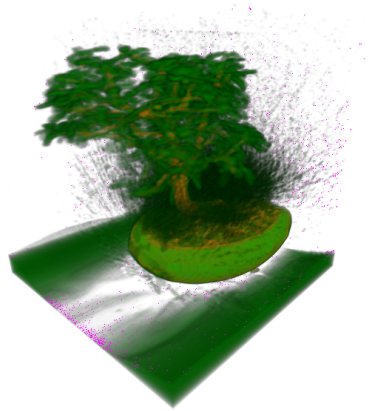
Precision Results – Bonsai ($\epsilon = 0.001$)



$R_{0.1}$



$C_{0.001}$



$D_{0.001}$

Performance Results – CPU/GPU

	CPU (s)	GPU (fps)
Riemann ($h = 0.1$)	76.80	14
Coupled ($\epsilon = 0.001$)	26.77	19
Decoupled ($\epsilon = 0.001$)	26.91	18