Avoiding Texture Seams by Discarding Sample Taps

Robert Toth
Intel, Advanced Rendering Technology
2014-03-15

Part I
Seams: Atlases and Ptex

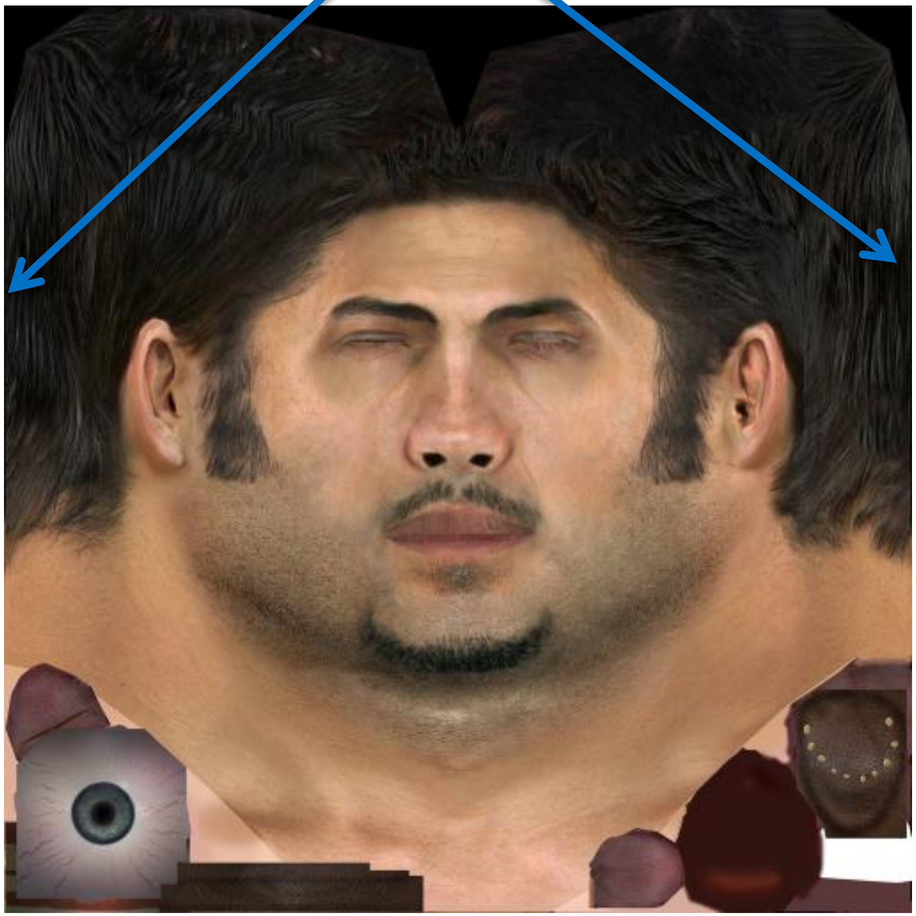
Texture Seams

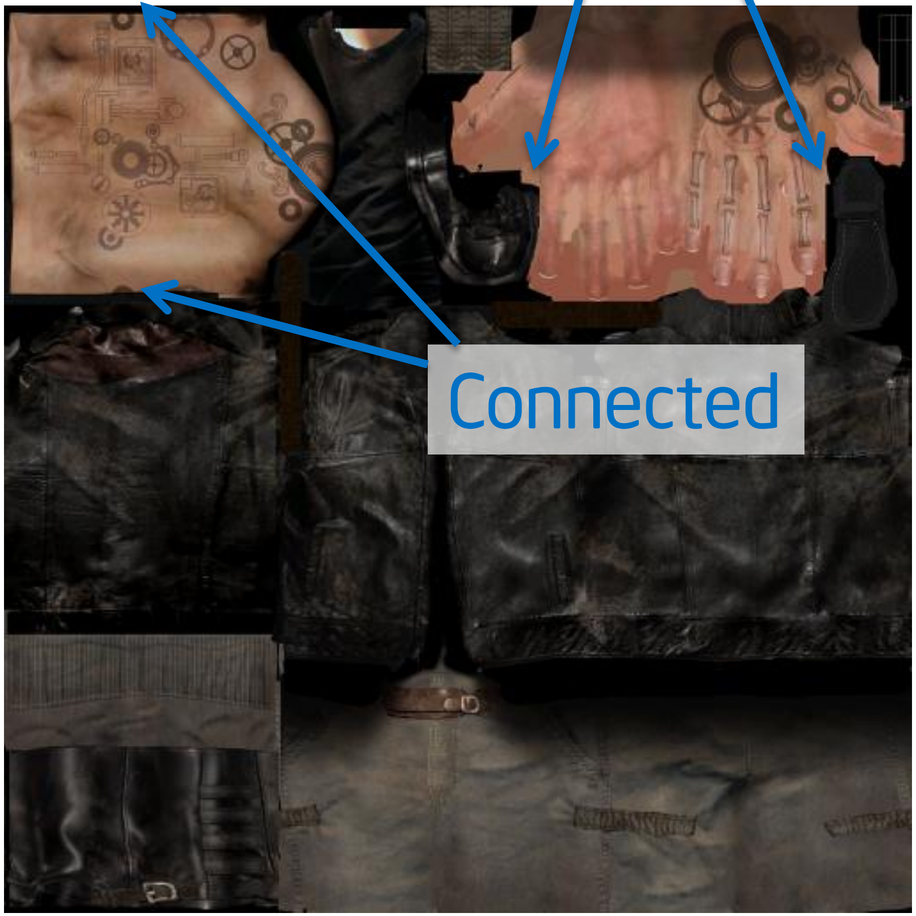
Atlas

Source: Microsoft DirectX SDK

Texture Seams

Atlas

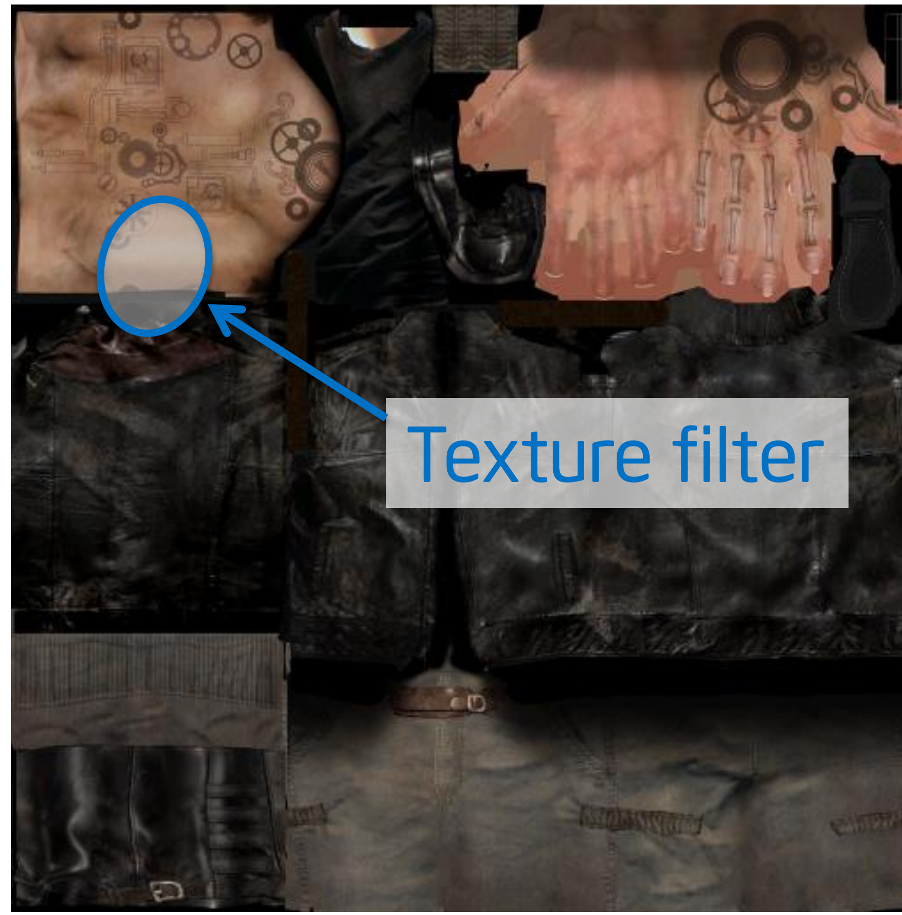




Source: Microsoft DirectX SDK

Texture Seams

Atlas

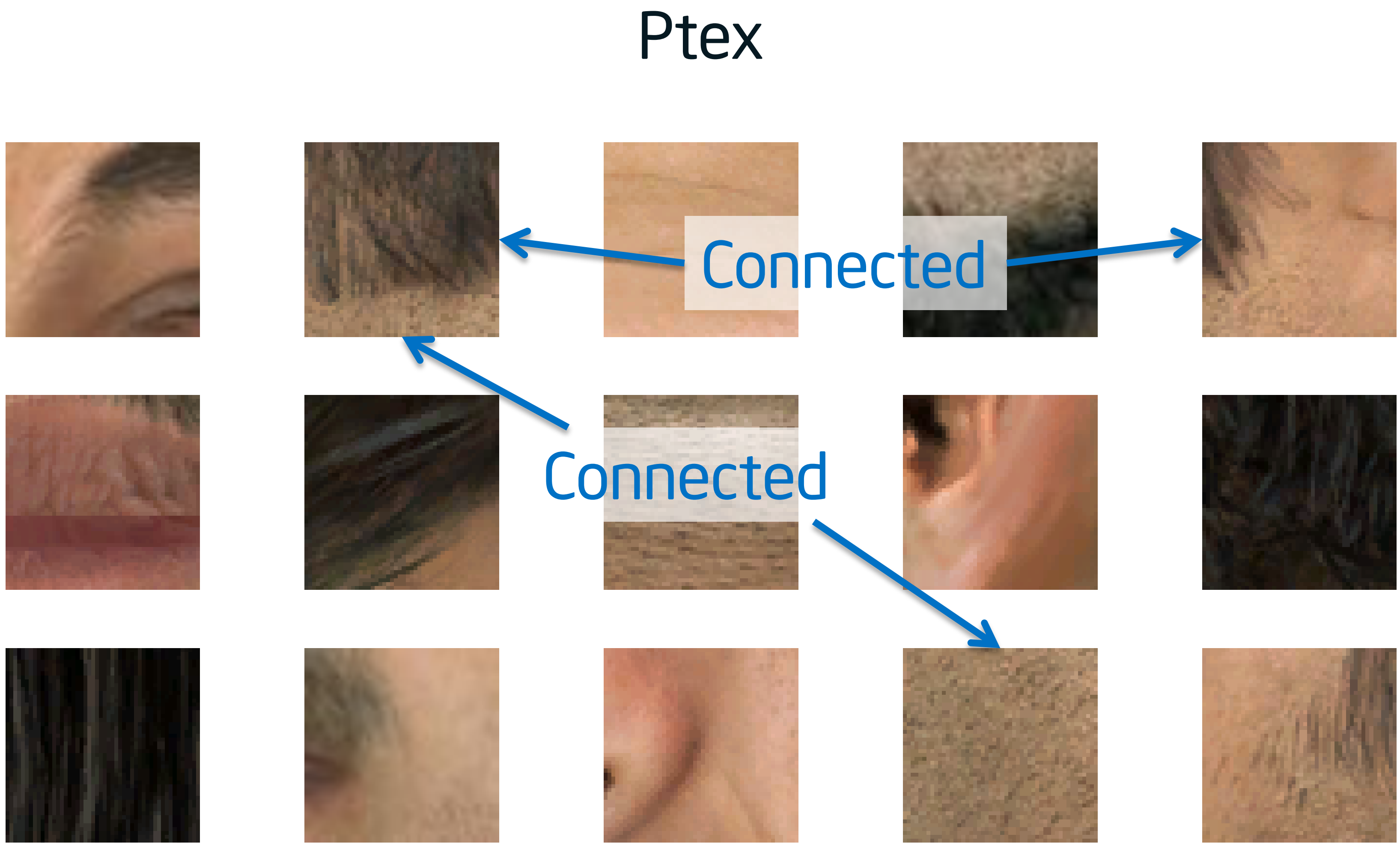


Source: Microsoft DirectX SDK

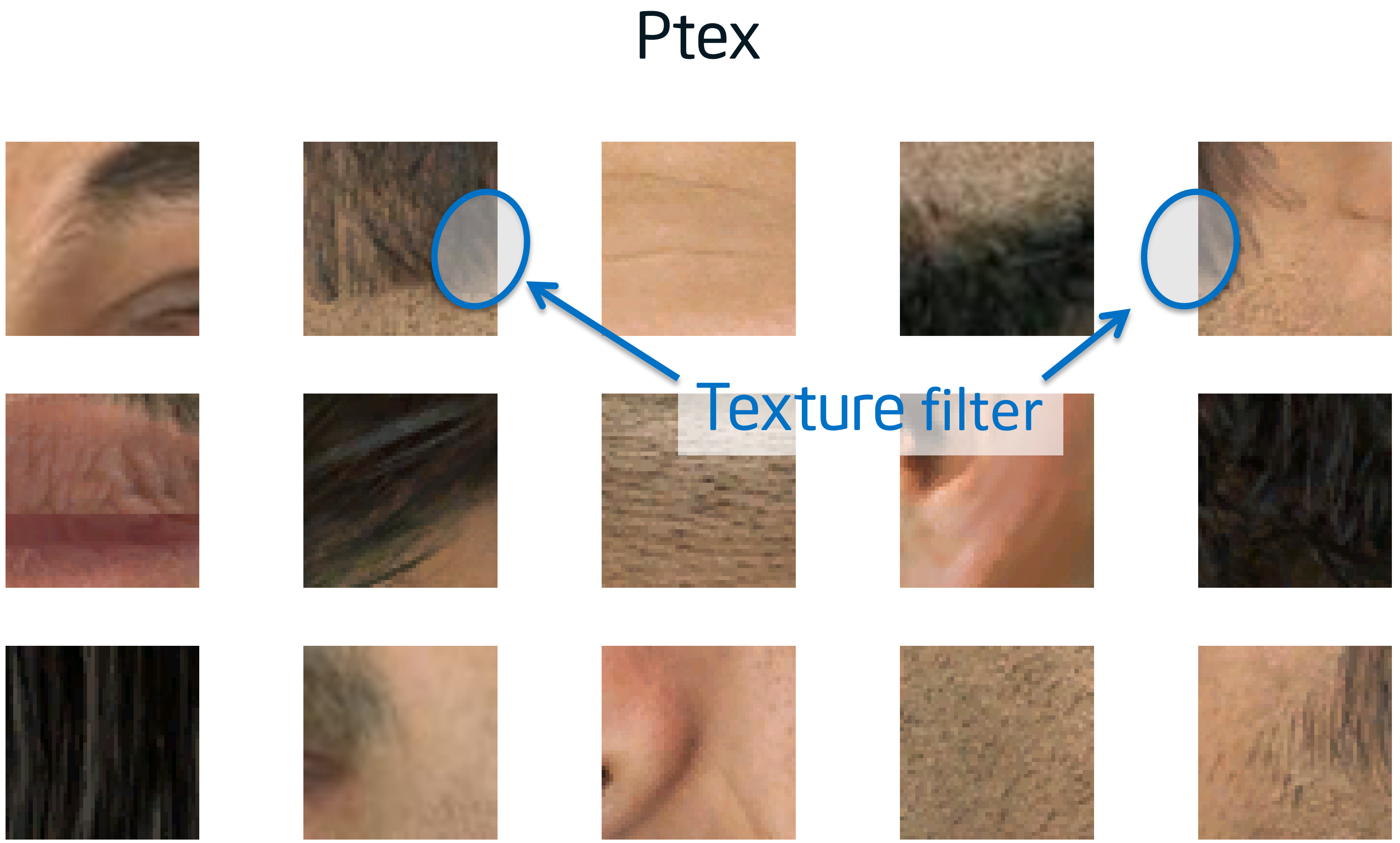
Texture Seams

Ptex

Texture Seams



Texture Seams



Realtime Ptex implementations

Algorithm	Wide filter	Memory	Lookups
McDonald, Burley: SIGGRAPH 2011 Real-time Ptex (Per-Face Texture Mapping)	Yes	Large	1
Kim, Hillesland, Hensley: SIGGRAPH Asia 2011 A Space-efficient and hardware-friendly Implementation of Ptex	No	Small	1
McDonald: GDC 2013 Eliminating Texture Waste: Borderless Ptex	Yes*	Small	5/10

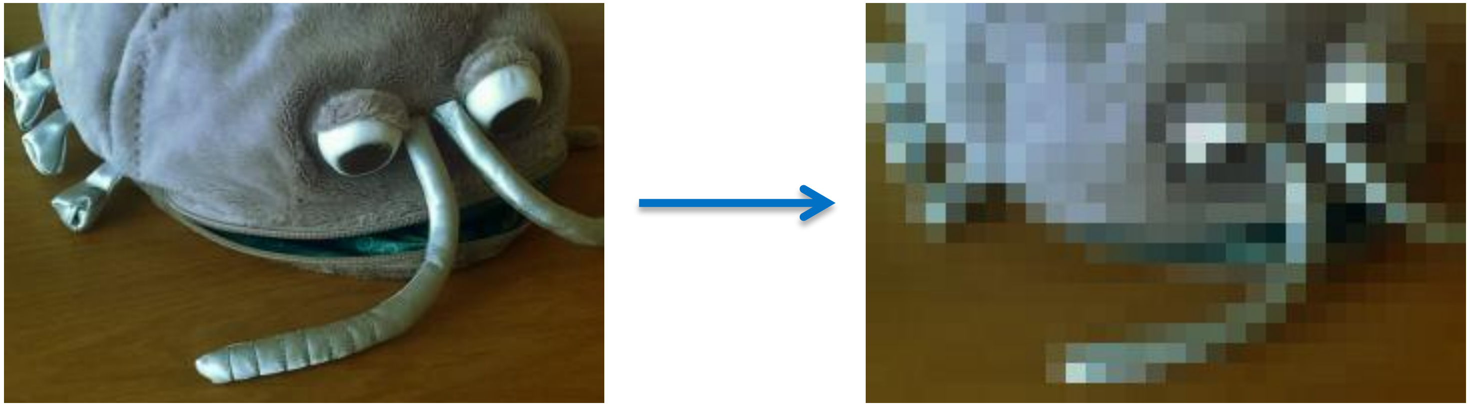
*: over edges only, not corners

Part II
Analysis: What, and Why?

Goal

Need to determine pixel colors

- Scene is a continuous signal
- Display has finite number of pixels



Two interpretations

Interpretation 1:

- This is a sampling and reconstruction problem!

Interpretation 2:

- This is an error minimization problem!

Regardless: integrate scene modulated by filter function, $f(x)$

Integrate

$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

Integrate

$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

Pixel color value

Integrate

$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

Integrate over each contributing surface

Integrate

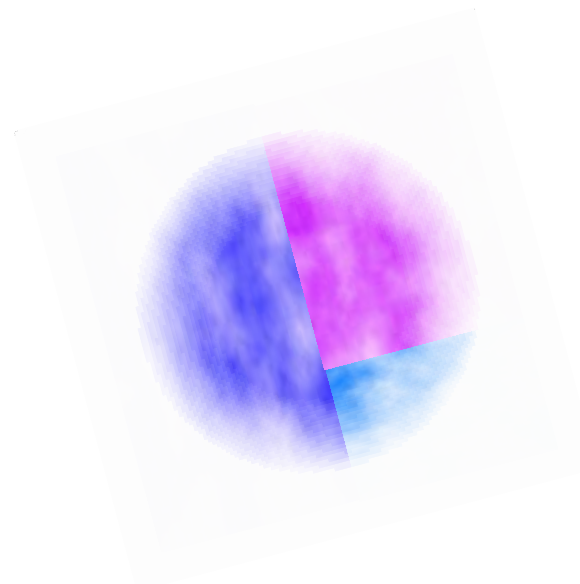
$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x}))f(\boldsymbol{x})d\boldsymbol{x}$$

Pixel filter function

Integrate

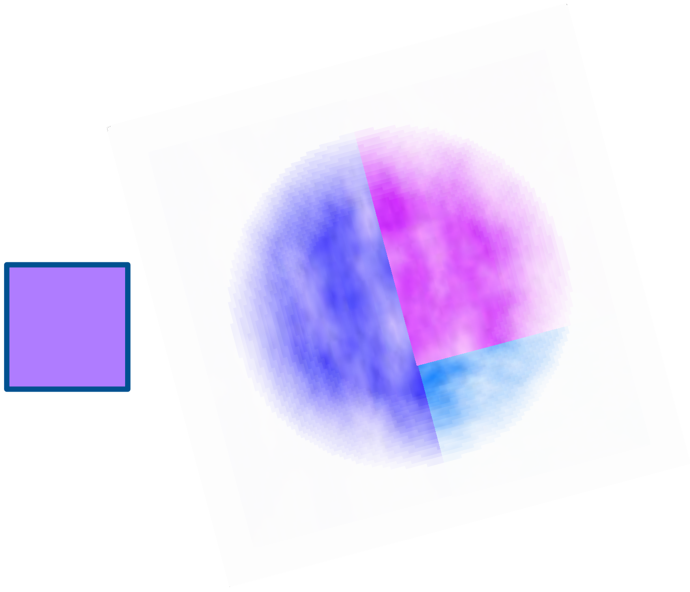
$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x}))f(\boldsymbol{x})d\boldsymbol{x}$$

Texture Texcoords



Integrate

$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x}))f(\boldsymbol{x})d\boldsymbol{x}$$



Integrate

$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

MSAA:

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

Integrate

$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x}))f(\boldsymbol{x})d\boldsymbol{x}$$

Resolve filter

MSAA:

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x})d\boldsymbol{x}$$

Many† visibility samples
-> Riemann integral

† a few

Integrate

$$P = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x}))f(\boldsymbol{x})d\boldsymbol{x}$$

Pixel shader

MSAA:

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x})d\boldsymbol{x}$$

Pass the problem on to the developer →

Pass the problem on to the texture sampler

Solution

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

Solution: Simple

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^S = \int_{\Omega_P} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

Solution: Simple

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^S = \int_{\Omega_P} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

Extrapolation

Solution: Simple

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^S = \int_{\Omega_P} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

$$P' = \sum_i c_i^S \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} \neq P$$

Solution: Traverse

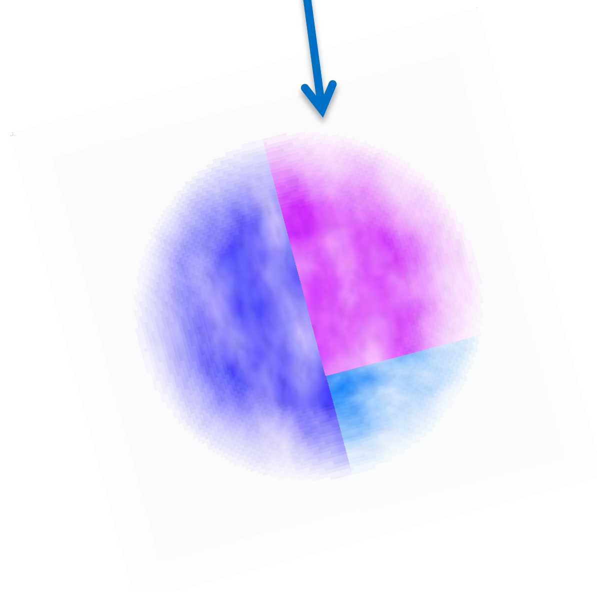
$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

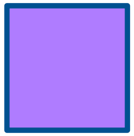
$$c_i^T = \sum_j \int_{\Omega_j} t_j(\boldsymbol{u}_j(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}$$

Solution: Traverse

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^T = \sum_j \int_{\Omega_j} t_j(\boldsymbol{u}_j(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$





Solution: Traverse

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

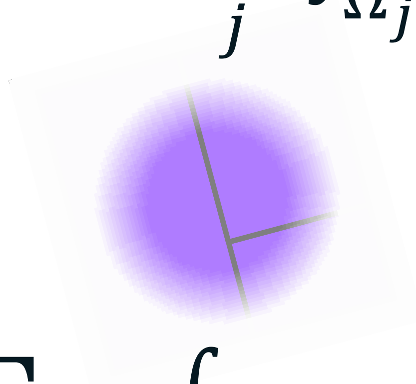
$$c_i^T = \sum_j \int_{\Omega_j} t_j(\boldsymbol{u}_j(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$

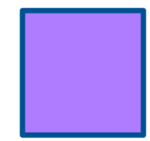
$$P' = \sum_i c_i^T \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = P \sum_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = P$$

Solution: Traverse

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x})d\boldsymbol{x}$$

$$c_i^T = \sum_j \int_{\Omega_j} t_j(\boldsymbol{u}_j(\boldsymbol{x}))f(\boldsymbol{x})d\boldsymbol{x} = P$$





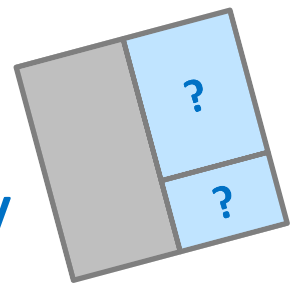
$$P' = \sum_i c_i^T \int_{\Omega_i} f(\boldsymbol{x})d\boldsymbol{x} = P\sum_i \int_{\Omega_i} f(\boldsymbol{x})d\boldsymbol{x} = P$$

Solution: Traverse

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x})\,d\boldsymbol{x}$$

$$c_i^T = \sum_j \int_{\Omega_j} t_j\big(\boldsymbol{u}_j(\boldsymbol{x})\big) f(\boldsymbol{x})\,d\boldsymbol{x} = P$$

Connectivity



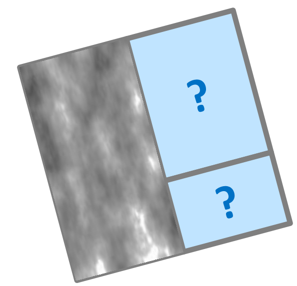
$$P' = \sum_i c_i^T \int_{\Omega_i} f(\boldsymbol{x})\,d\boldsymbol{x} = P \sum_i \int_{\Omega_i} f(\boldsymbol{x})\,d\boldsymbol{x} = P$$

Solution: Traverse

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^T = \sum_j \int_{\Omega_j} t_j(\boldsymbol{u}_j(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$

Neighboring textures



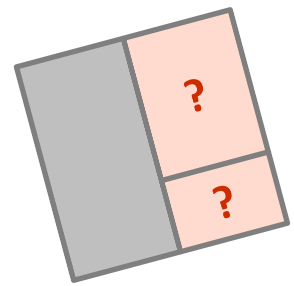
$$P' = \sum_i c_i^T \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = P \sum_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = P$$

Solution: Traverse

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x})\,d\boldsymbol{x}$$

$$c_i^T = \sum_j \int_{\Omega_j} t_j\big(\boldsymbol{u}_j(\boldsymbol{x})\big) f(\boldsymbol{x})\,d\boldsymbol{x} = P$$

Curvature



$$P' = \sum_i c_i^T \int_{\Omega_i} f(\boldsymbol{x})\,d\boldsymbol{x} = P \sum_i \int_{\Omega_i} f(\boldsymbol{x})\,d\boldsymbol{x} = P$$

Solution: Traverse

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^T \approx \sum_j \int_{\Omega_j} t_j(\boldsymbol{u}_j(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$

$$P' = \sum_i c_i^T \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} \approx P \sum_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = P$$

New solution: Discard

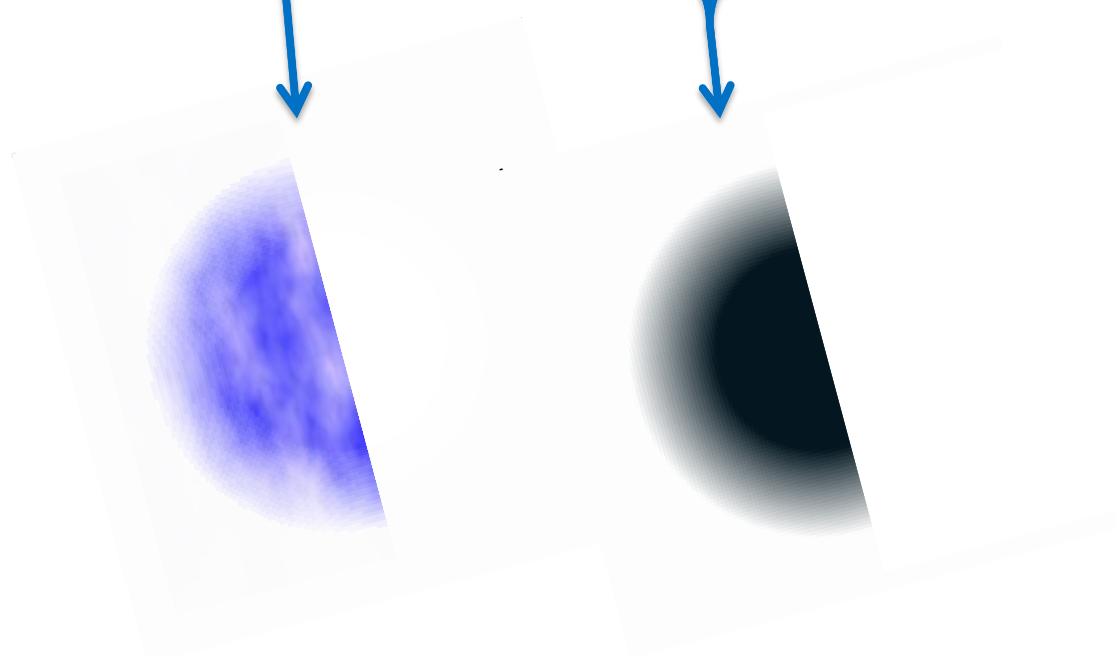
$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^D = \frac{\int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}}{\int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}}$$

New solution: Discard

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^D = \frac{\int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}}{\int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}}$$

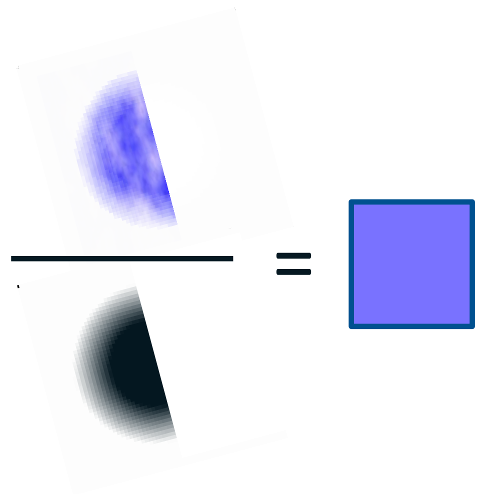


New solution: Discard

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^D = \frac{\int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}}{\int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}}$$

Normalize filter weight



New solution: Discard

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

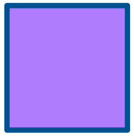
$$c_i^D = \frac{\int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}}{\int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}}$$

$$P' = \sum_i c_i^D \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$

New solution: Discard

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^D = \frac{\int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}}{\int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}}$$



$$P' = \sum_i c_i^D \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$

New solution: Discard

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^D = \frac{\int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}}{\int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}}$$

Resolve filter

Texture filter

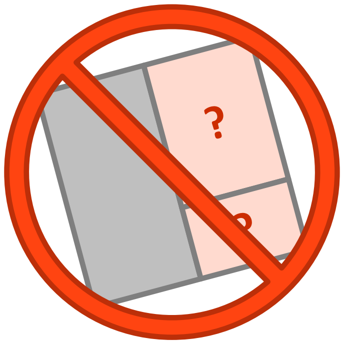
$$P' = \sum_i c_i^D \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$

New solution: Discard

$$P' = \sum_i c_i \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}$$

$$c_i^D = \frac{\int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x}}{\int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x}}$$

Local information



$$P' = \sum_i c_i^D \int_{\Omega_i} f(\boldsymbol{x}) d\boldsymbol{x} = \sum_i \int_{\Omega_i} t_i(\boldsymbol{u}_i(\boldsymbol{x})) f(\boldsymbol{x}) d\boldsymbol{x} = P$$

New solution: Discard

Restrict all the edges?

New solution: Discard

Restrict all the edges?

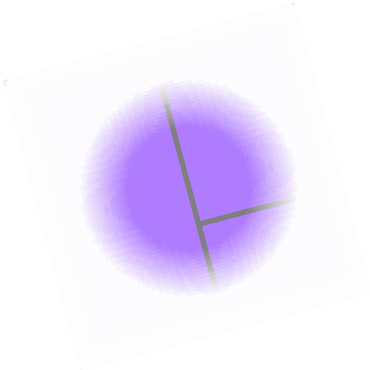
- No, only texture boundaries

Interiors are less problematic

Sampler would need edge information

John McDonald [2013] * (Traverse)

```
C = lookup(texture, texcoord(p))

W = lookup(one, texcoord(p))

for each neighboring patch K:

        C += lookup(K.texture, K.texcoord(p))

        W += lookup(K.one, K.texcoord(p))

return C/W
```



* Eliminating Texture Waste: Borderless Ptex
https://developer.nvidia.com/gdc-2013/

New solution (Discard)

```
C = lookup(texture, texcoord(p))

W = lookup(one, texcoord(p))

//for each neighboring patch K:

//          C += lookup(K.texture, K.texcoord(p))

//          W += lookup(K.one, K.texcoord(p))

return C/W
```

Part III
Really?

Assumptions

Traverse

Surface is not curved

Texture is not stretched

Discard

Resolve filter and texture filter are the same

Assumptions

Traverse

Surface is not curved

Texture is not stretched

Discard

Resolve filter and texture filter are the same

They should be be!

Assumptions

Traverse

Surface is not curved

Texture is not stretched

Discard

Resolve filter and texture filter are the same

Quality factors

Traverse

Texture filter

Resolve filter (silhouettes)

Geometry/texture curvature

Discard

Texture filter

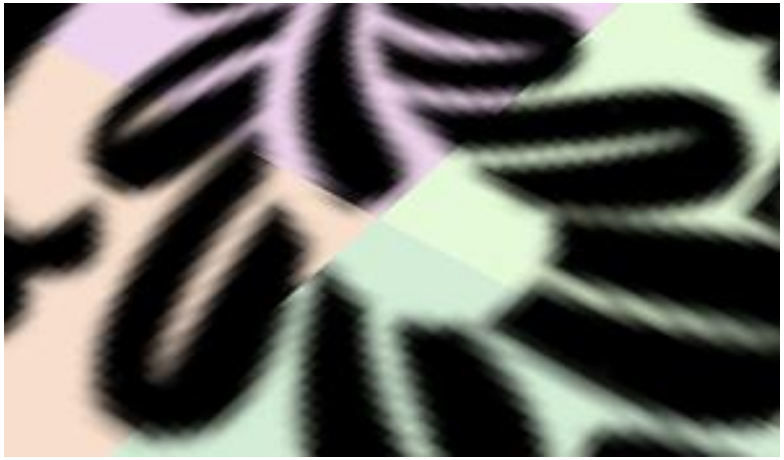
Resolve filter

Achilles heel

DX/GL do not specify texture content all the way to the texture boundary

- Bad for magnification

Desired appearance

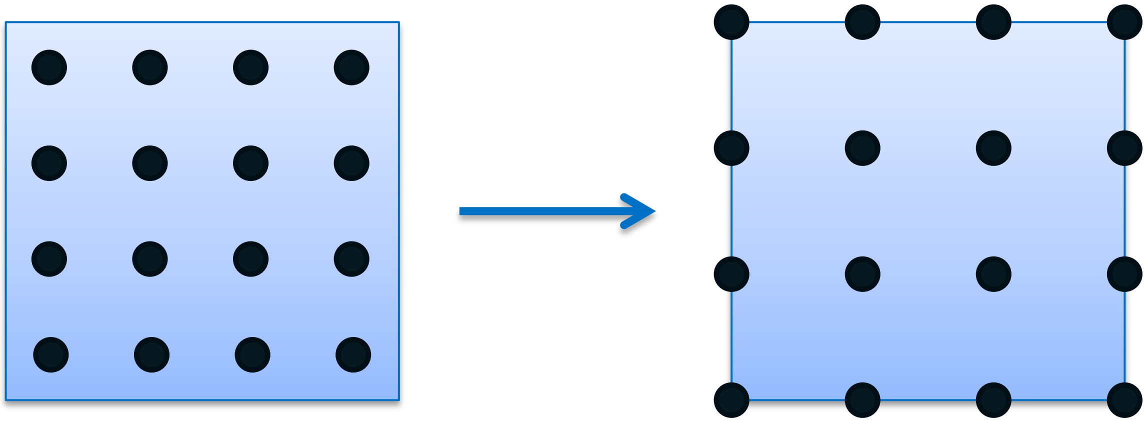


Appearance without texture data at border

Achilles heel

Solved by Purnomo et al., 2004 *

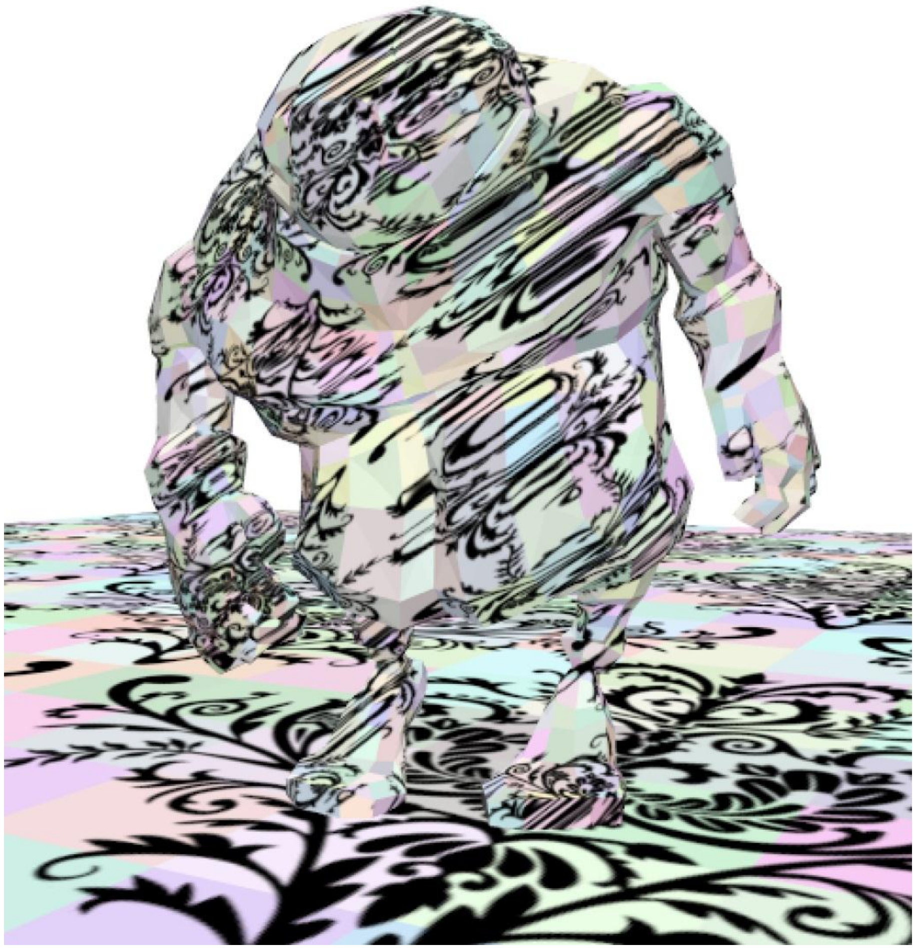
Still not in DirectX/OpenGL APIs



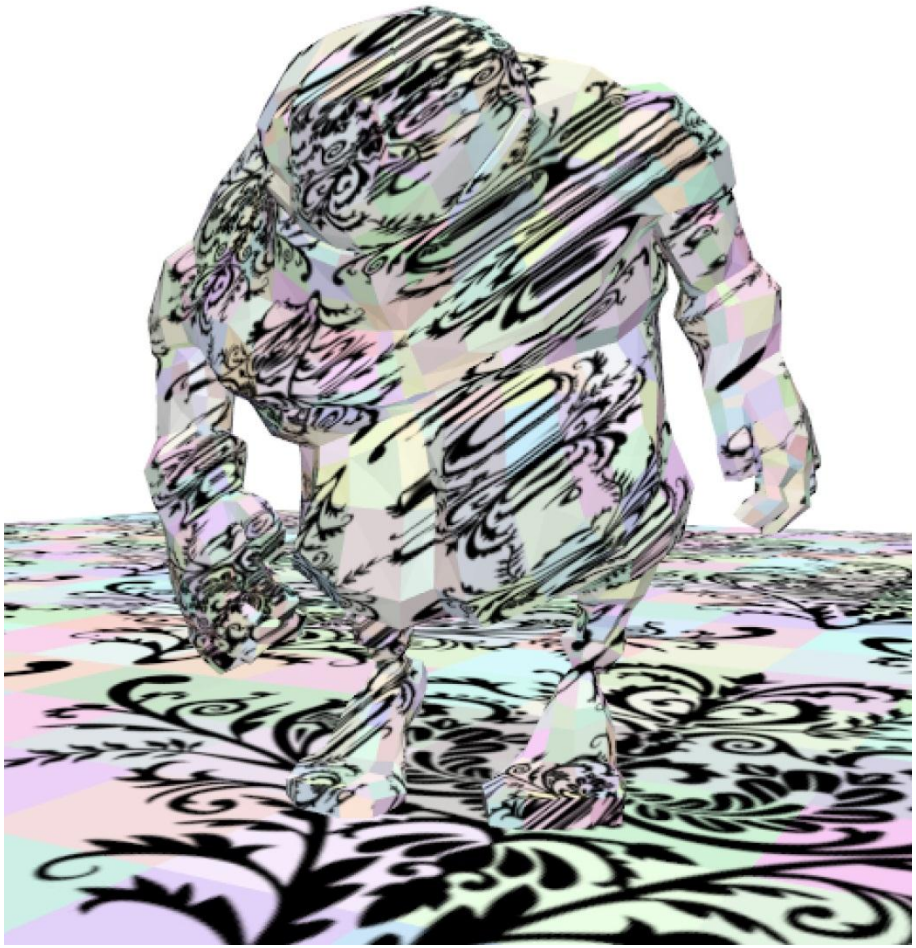
* B. Purnomo, J. Cohen and S. Kumar, "Seamless Texture Atlases"
ACM SIGGRAPH/Eurographics SGP 2004

Results

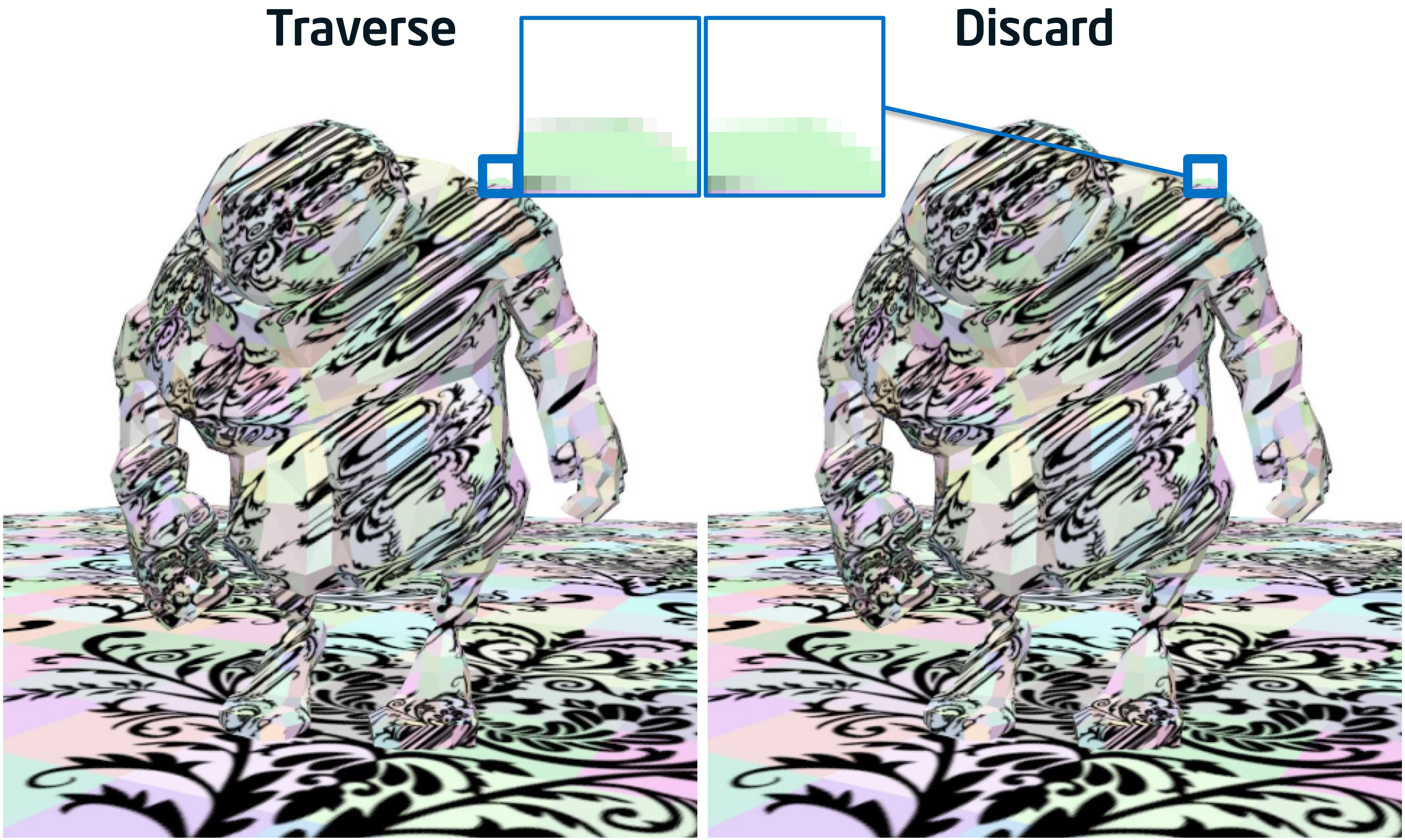
Traverse



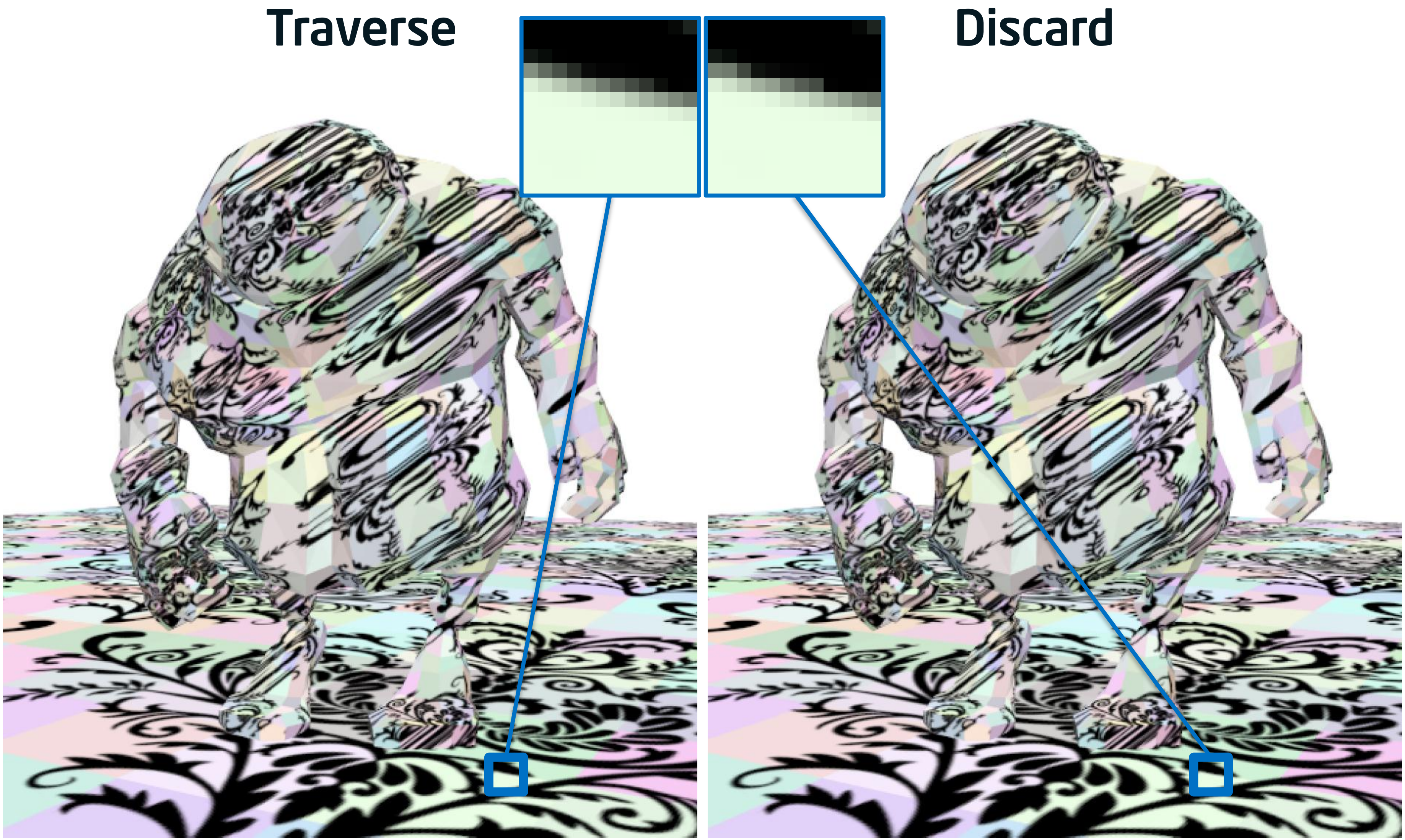
Discard



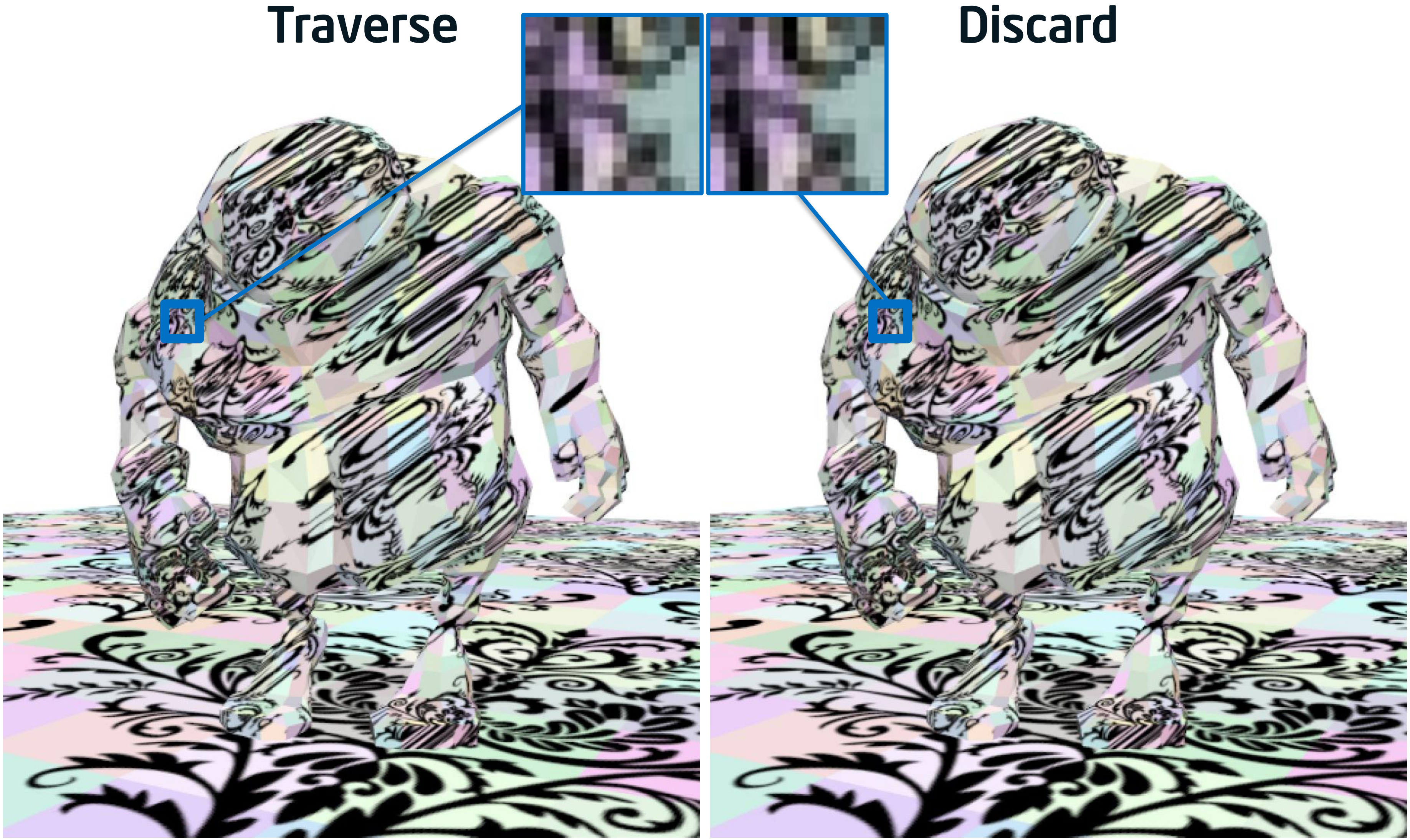
Results

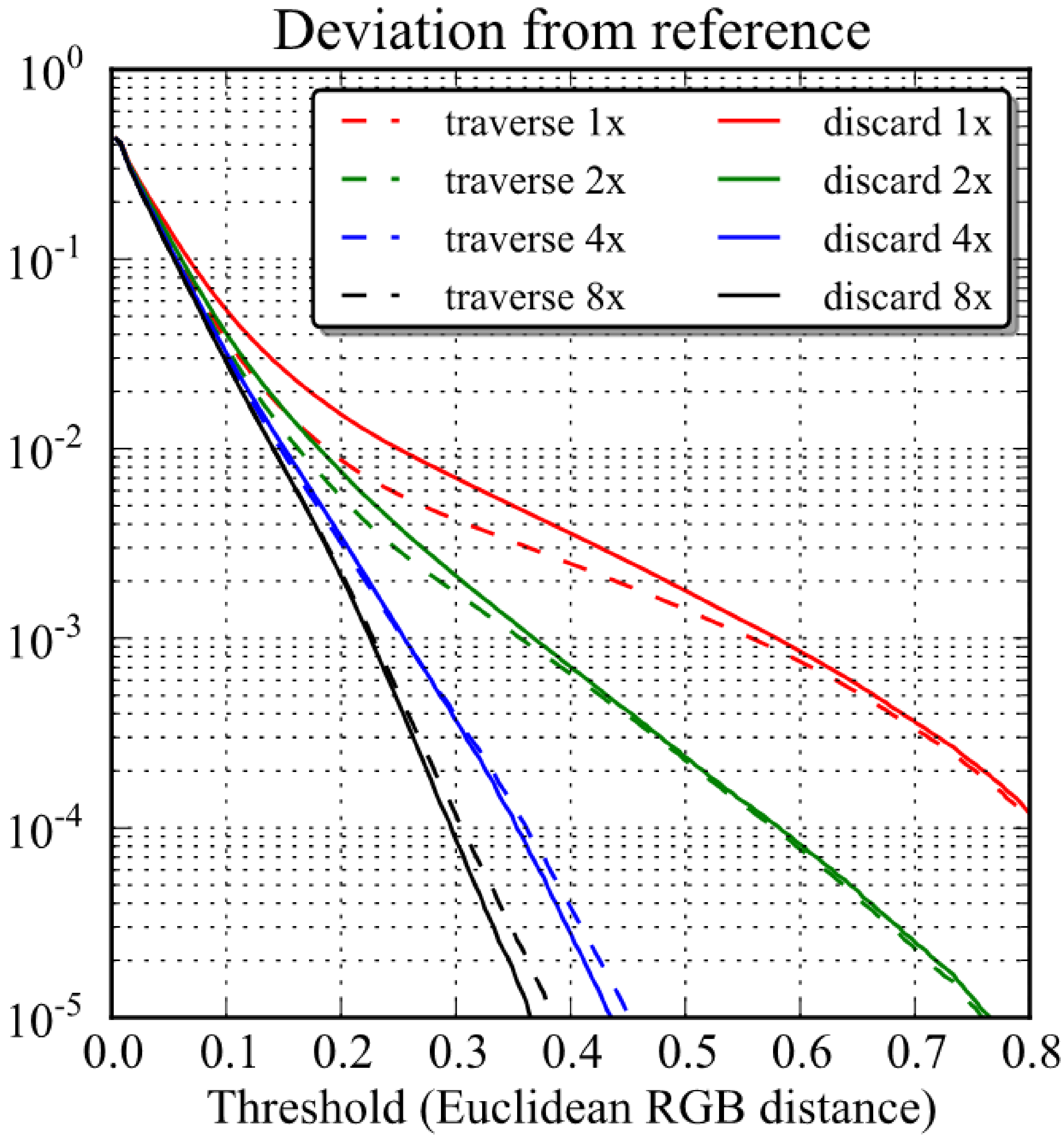


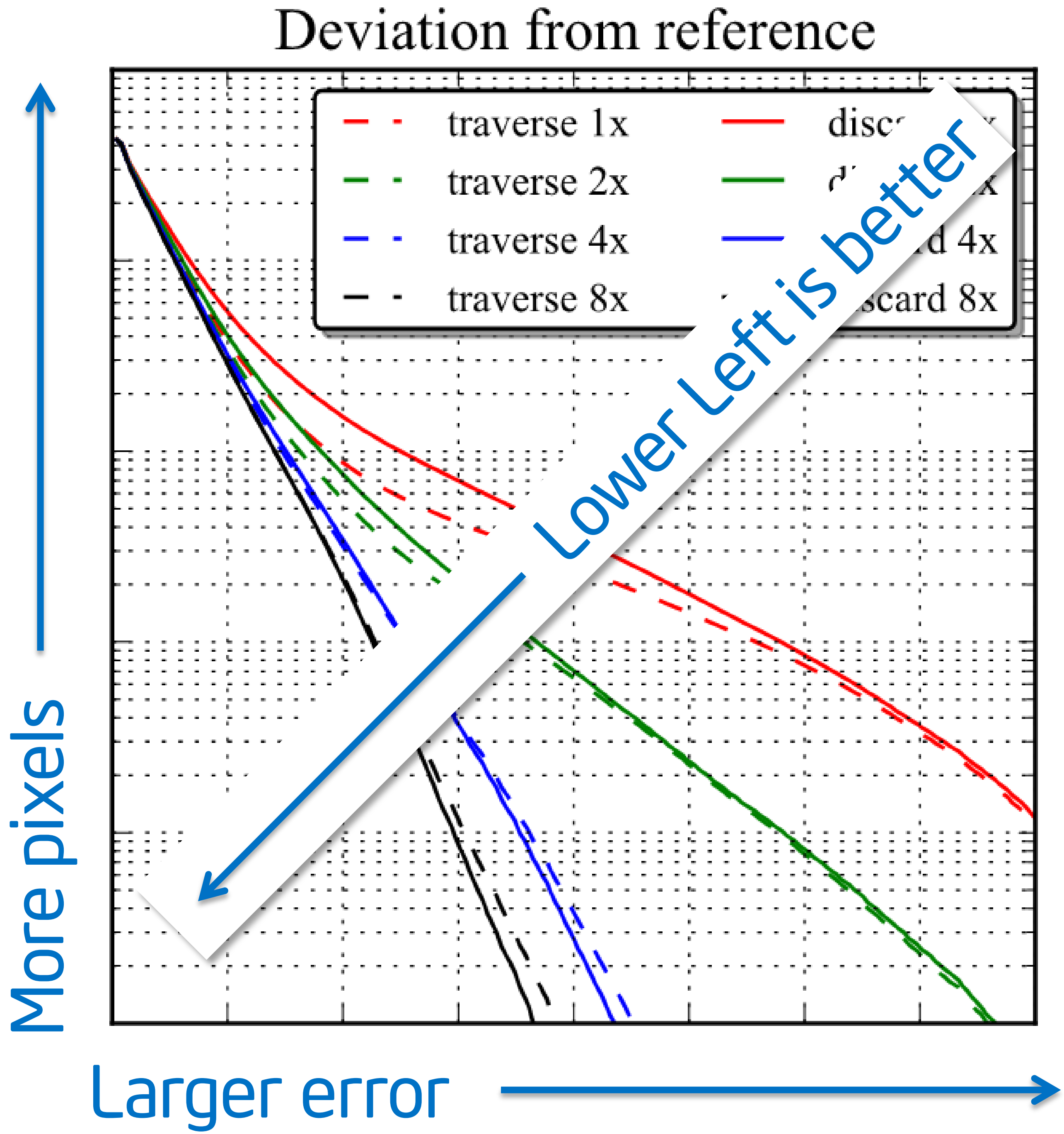
Results

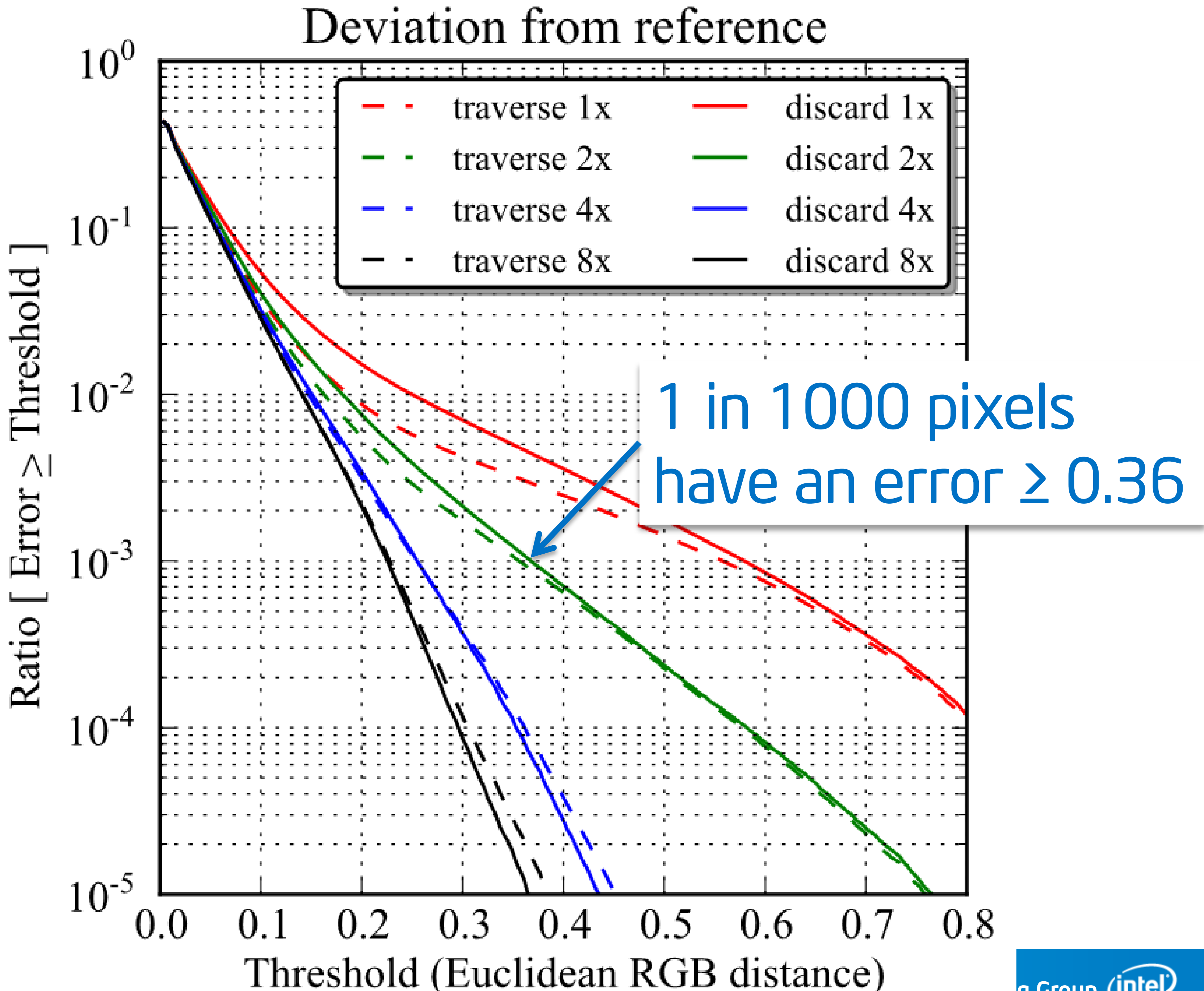


Results

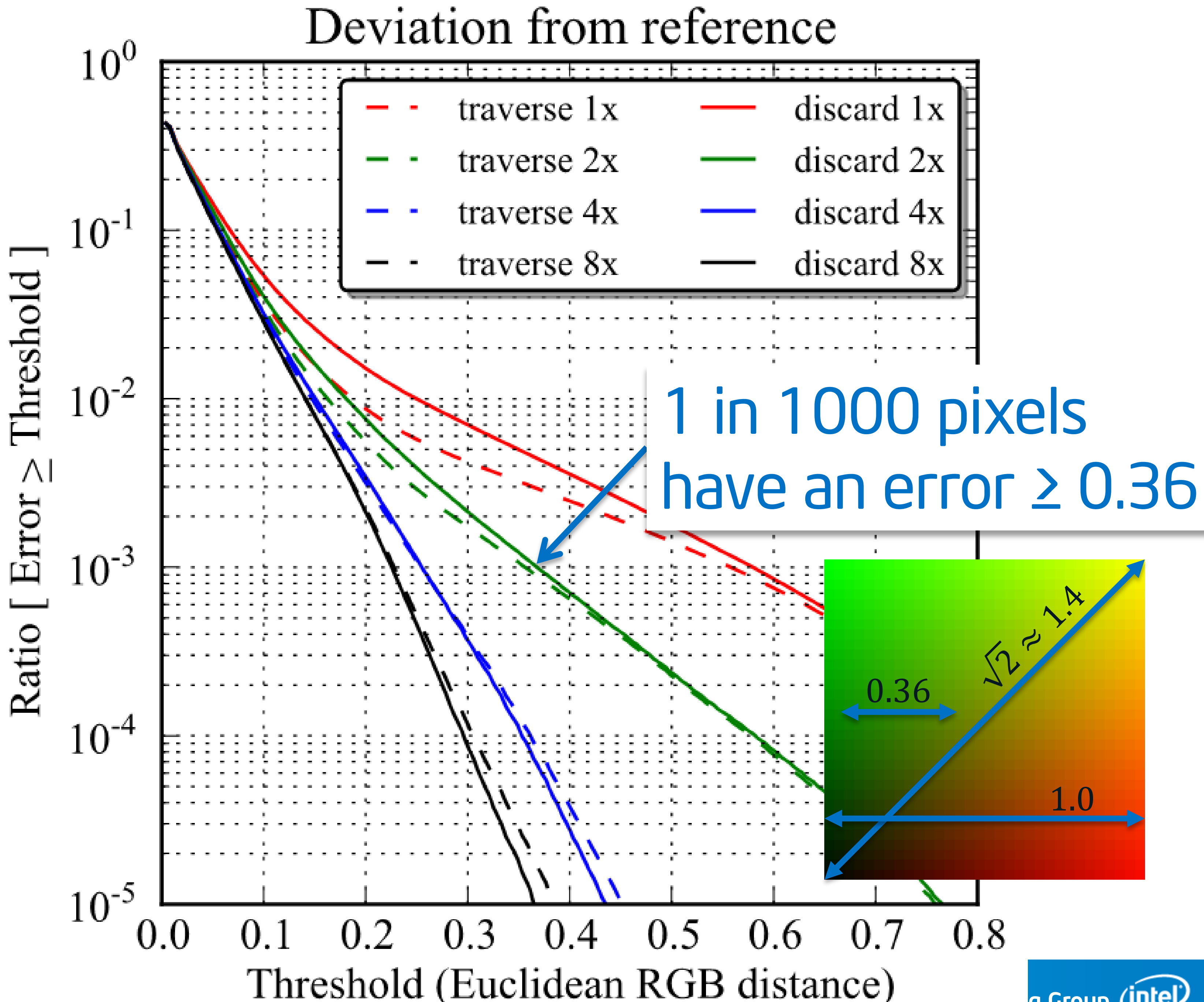


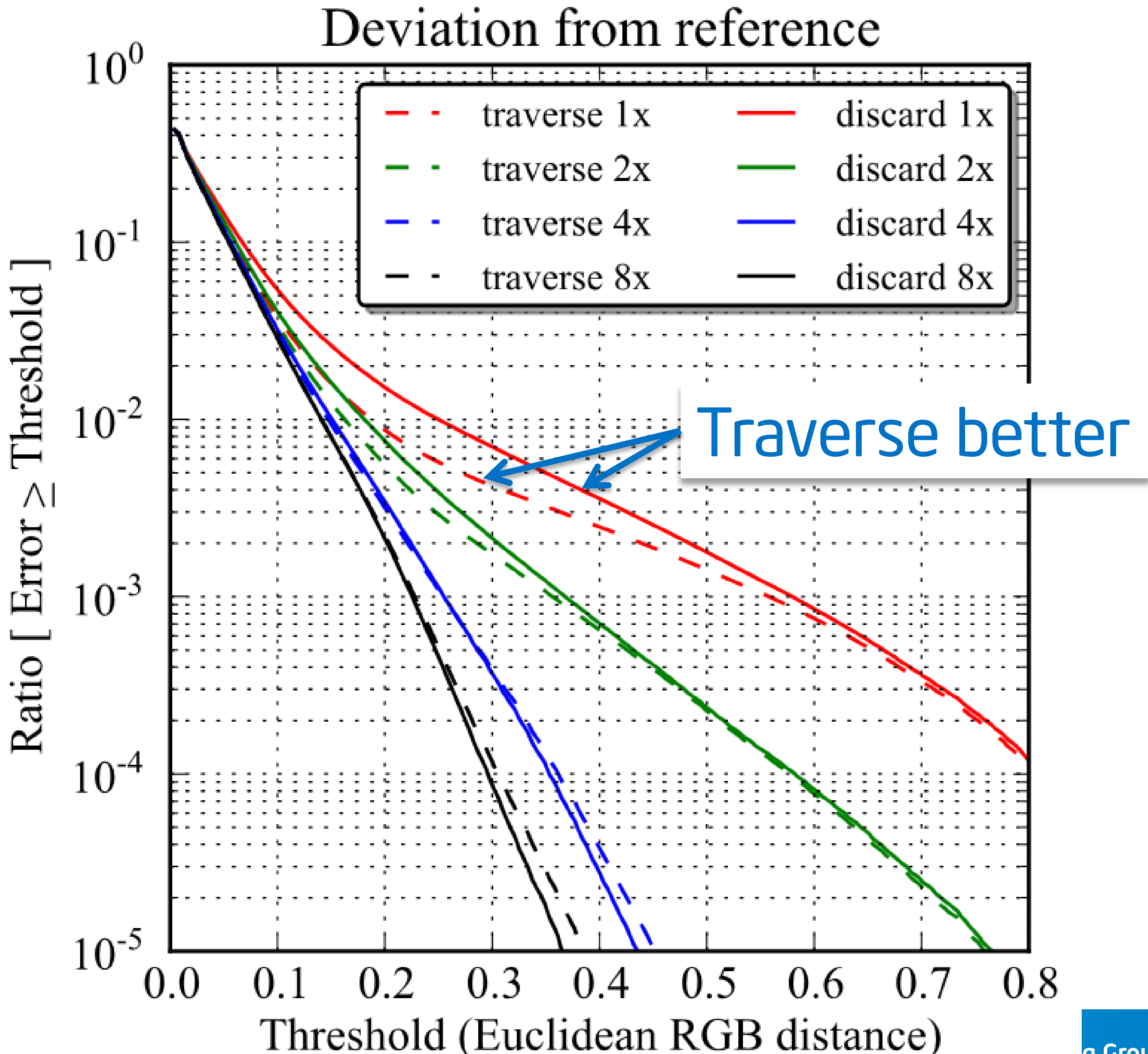

Deviation from reference
traverse 1x
traverse 2x
traverse 4x
traverse 8x
discard 1x
discard 2x
discard 4x
discard 8x
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Threshold (Euclidean RGB distance)

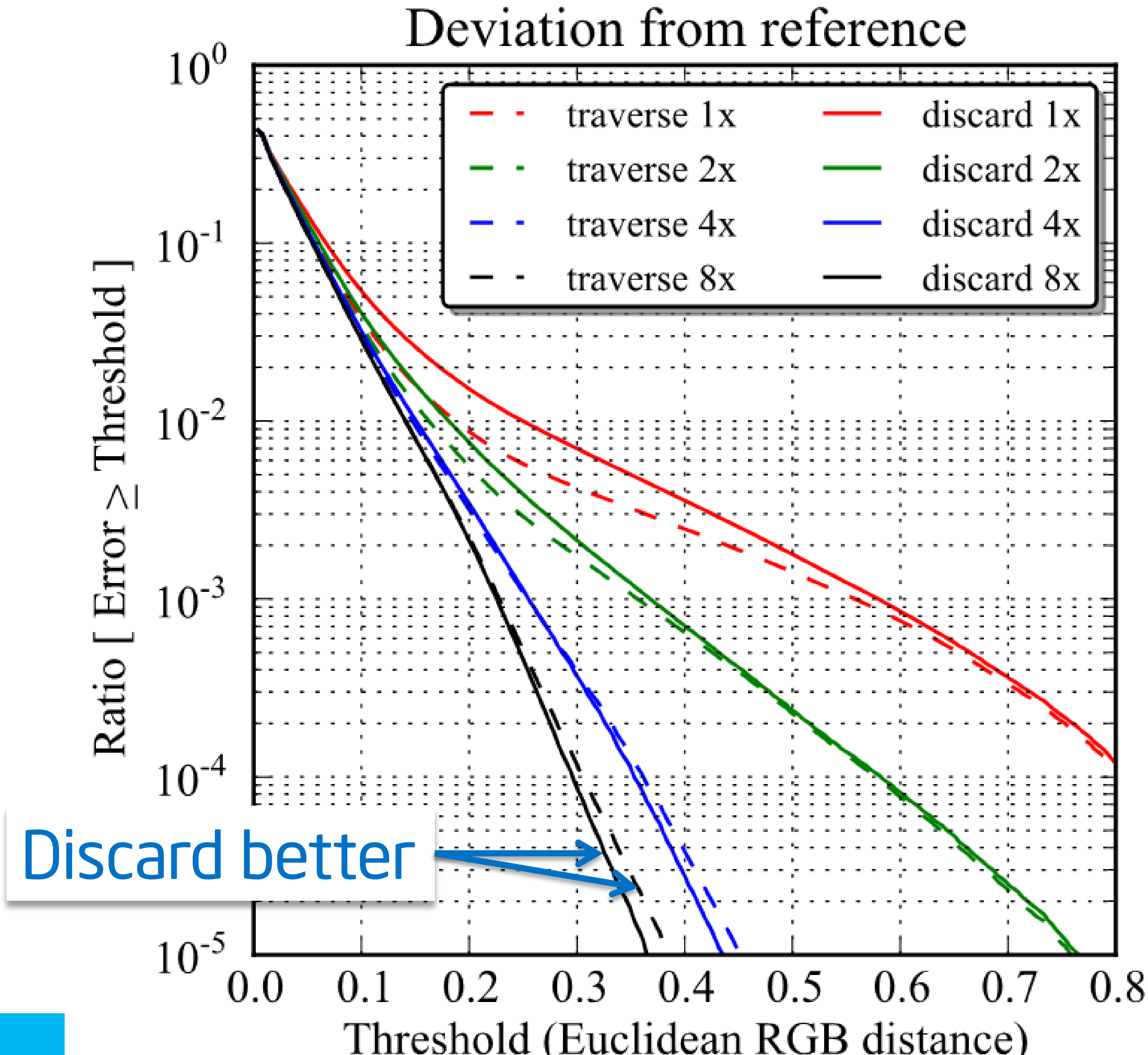

Deviation from reference
traverse 1x
traverse 2x
traverse 4x
traverse 8x
Lower Left is better
More pixels
Larger error



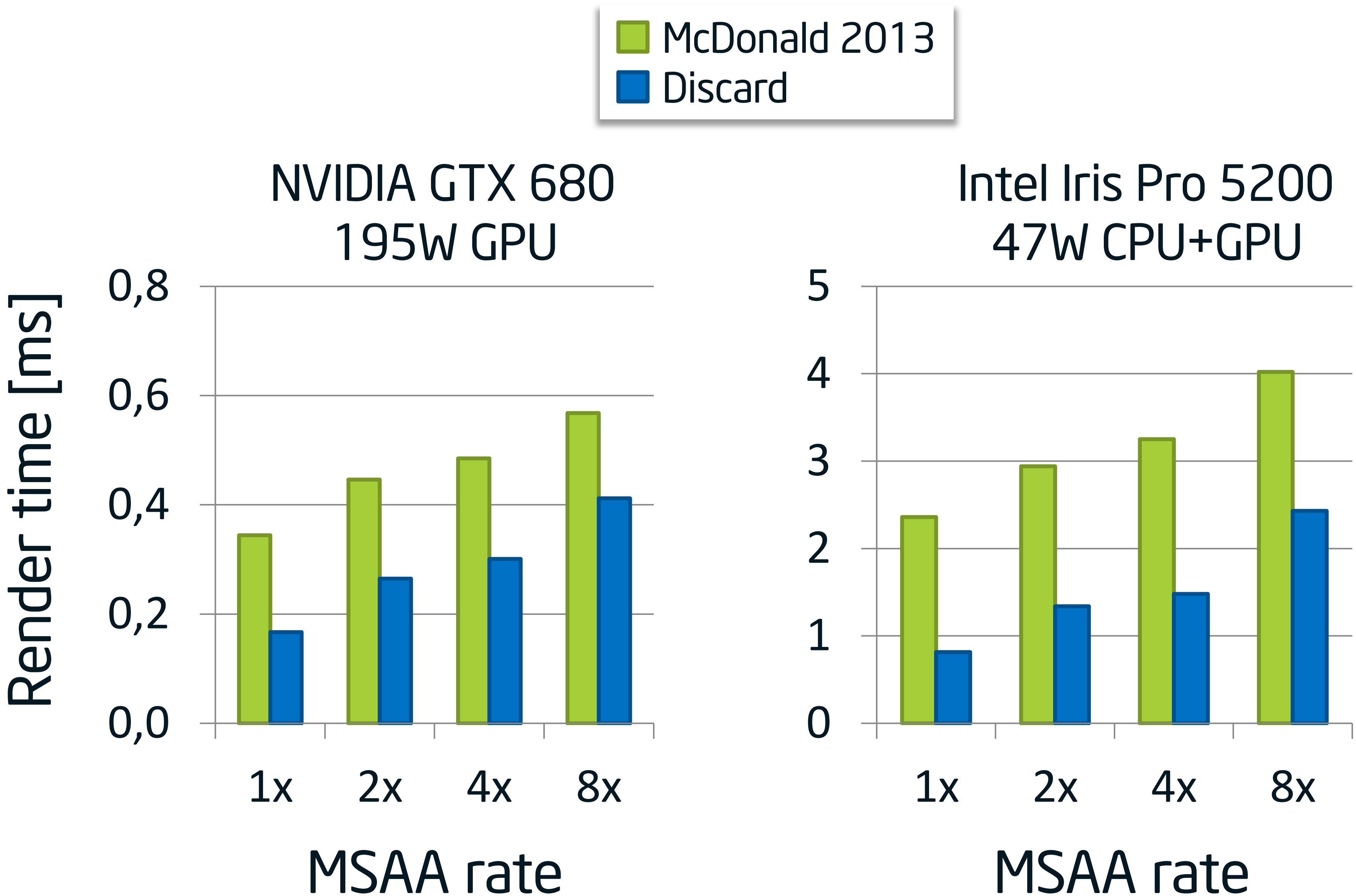
Deviation from reference
traverse 1x
traverse 2x
traverse 4x
traverse 8x
discard 1x
discard 2x
discard 4x
discard 8x
1 in 1000 pixels
have an error ≥ 0.36
Ratio [Error ≥ Threshold]
Threshold (Euclidean RGB distance)

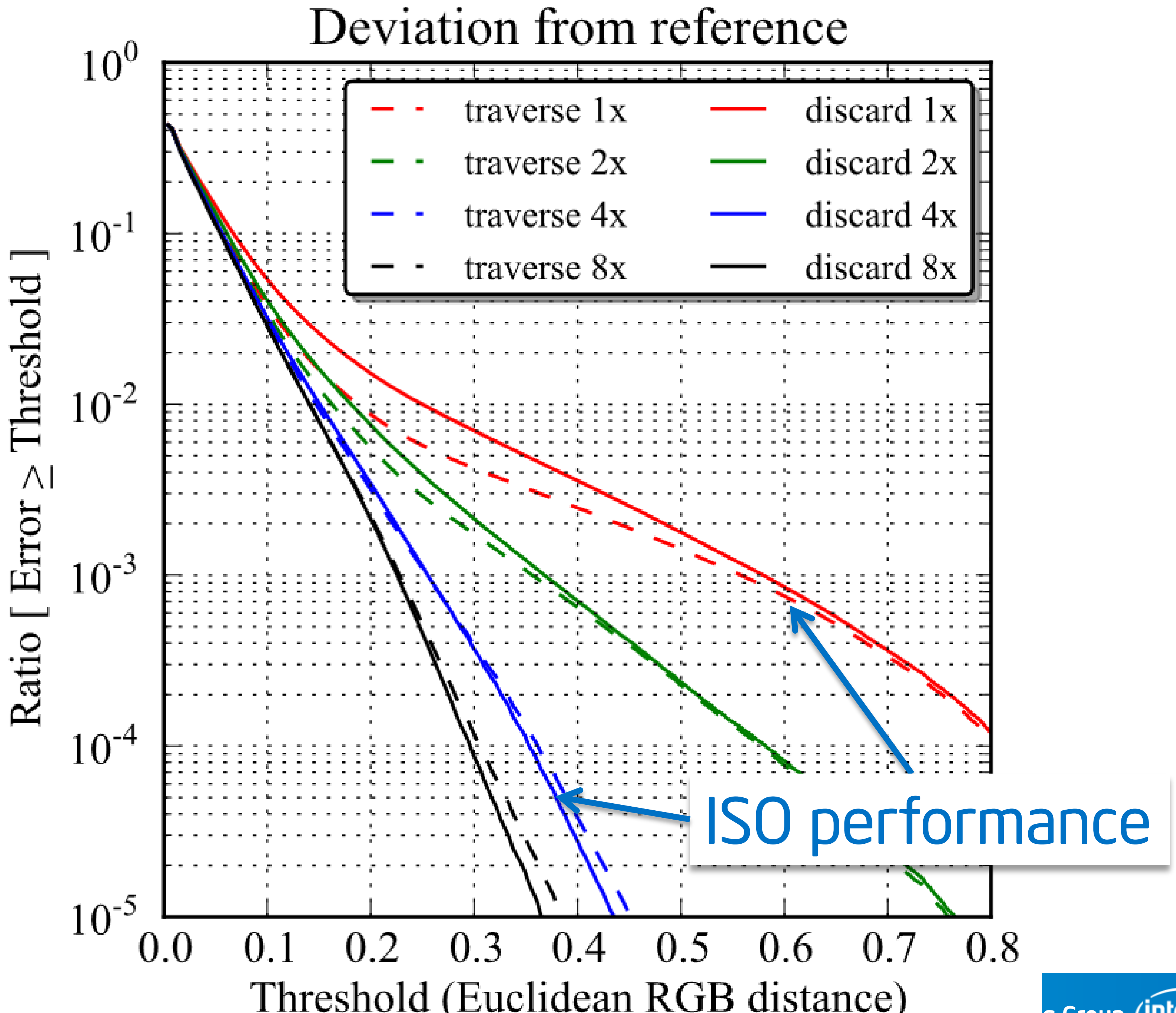

Deviation from reference
Ratio [Error ≥ Threshold]
Threshold (Euclidean RGB distance)
traverse 1x
traverse 2x
traverse 4x
traverse 8x
discard 1x
discard 2x
discard 4x
discard 8x
1 in 1000 pixels
have an error ≥ 0.36
0.36
√2 ≈ 1.4
1.0


Deviation from reference
traverse 1x
traverse 2x
traverse 4x
traverse 8x
discard 1x
discard 2x
discard 4x
discard 8x
Ratio [Error ≥ Threshold]
Threshold (Euclidean RGB distance)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Traverse better


Deviation from reference
traverse 1x
traverse 2x
traverse 4x
traverse 8x
discard 1x
discard 2x
discard 4x
discard 8x
Ratio [Error ≥ Threshold]
Threshold (Euclidean RGB distance)
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Discard better

Performance




Deviation from reference
traverse 1x
traverse 2x
traverse 4x
traverse 8x
discard 1x
discard 2x
discard 4x
discard 8x
Ratio [Error ≥ Threshold]
Threshold (Euclidean RGB distance)
ISO performance

Realtime Ptex implementations

Algorithm	Wide filter	Memory	Lookups
McDonald, Burley: SIGGRAPH 2011 Real-time Ptex (Per-Face Texture Mapping)	Yes	Large	1
Kim, Hillesland, Hensley: SIGGRAPH Asia 2011 A Space-efficient and hardware-friendly Implementation of Ptex	No	Small	1
McDonald: GDC 2013 Eliminating Texture Waste: Borderless Ptex	Yes*	Small	5/10

*: over edges only, not corners

Realtime Ptex implementations

Algorithm	Wide filter	Memory	Lookups
McDonald, Burley: SIGGRAPH 2011 Real-time Ptex (Per-Face Texture Mapping)	Yes	Large	1
Kim, Hillesland, Hensley: SIGGRAPH Asia 2011 A Space-efficient and hardware-friendly Implementation of Ptex	No	Small	1
McDonald: GDC 2013 Eliminating Texture Waste: Borderless Ptex	Yes*	Small	5/10

*: ~~over edges only, not corners~~

Realtime Ptex implementations

Algorithm	Wide filter	Memory	Lookups
McDonald, Burley: SIGGRAPH 2011 Real-time Ptex (Per-Face Texture Mapping)	Yes	Large	1
Kim, Hillesland, Hensley: SIGGRAPH Asia 2011 A Space-efficient and hardware-friendly Implementation of Ptex	No	Small	1
McDonald: GDC 2013 Eliminating Texture Waste: Borderless Ptex	Yes*	Small	5/10
Toth: JCGT 2013 Avoiding Texture Seams by Discarding Filter Taps	Yes	Small	1/2

*: ~~over edges only, not corners~~

Thank you for listening

intel®