Filmic SMAA

Sharp Morphological and Temporal Antialiasing

Jorge Jimenez – Graphics R&D Technical Director (Activision Blizzard)

Advances in Real-Time Rendering course

SIGGRAPH2016

ACTIVISION | BLIZZARD

Notice for Offline Reading

- Hidden slides in this presentation
- Slide show mode won’t show them
- Don’t miss the speaker notes, most slides have them
- Movies typically contain a red/green bar on the bottom
  - Red means without a feature
  - Green means with it
  - Movie progress bar might hide this bar if not seen in fullscreen mode

SMAA Goals

- Sharpness

SMAA Goals

- Sharpness
- Robustness

SMAA Goals

- Sharpness
- Robustness
- Performance
  - 0.75-1.05ms on a PS4@1080p
  - Includes most of what is included in this presentation

What is Filmic SMAA?

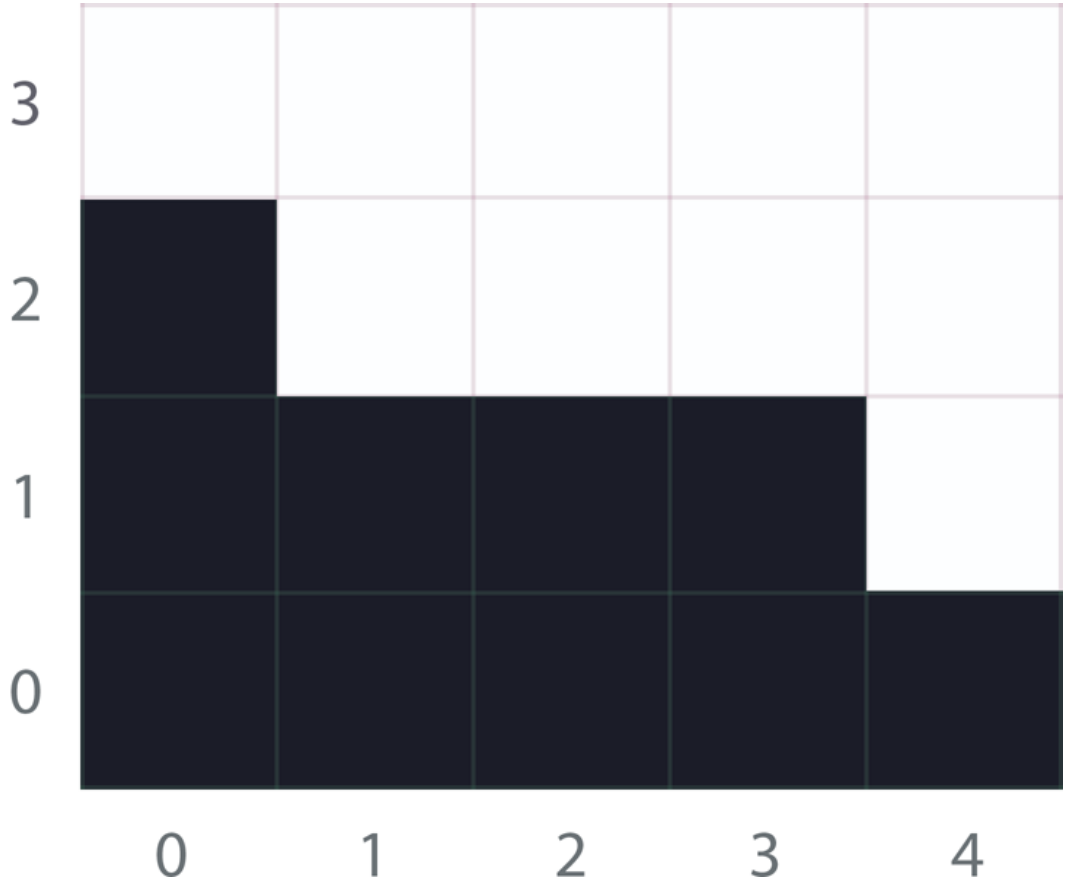
- Morphological antialiasing (SMAA 1x)
  - + Temporal supersampling (SMAA T2x)
  - + Spatial multisampling (SMAA S2x) – *not practical for our games*
  - Combined (SMAA 4x) – *not practical for our games*

What is Filmic SMAA?

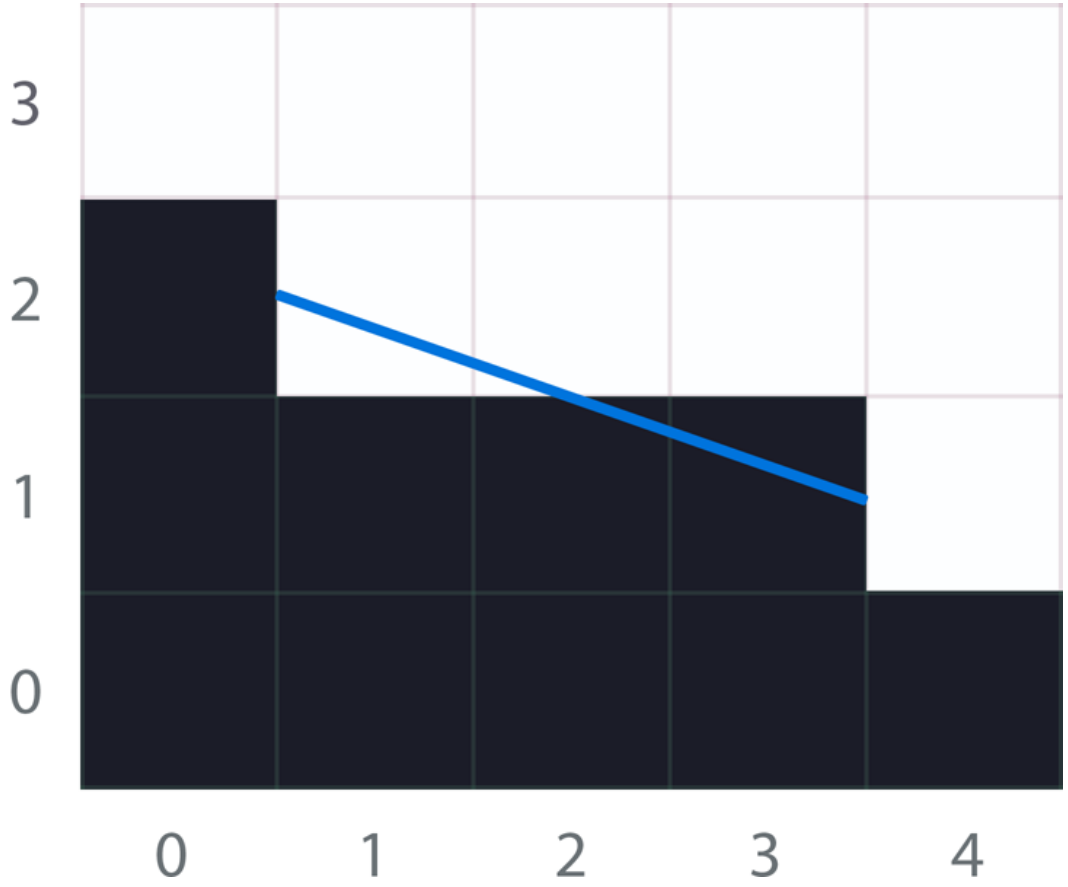
- We added a new component
  - Filmic filtering (*Filmic* variants)
    - Filmic SMAA 1x
    - Filmic SMAA T2x – used for PS4 and XB1
    - …
- Temporal filtering ≠ temporal supersampling

Morphological

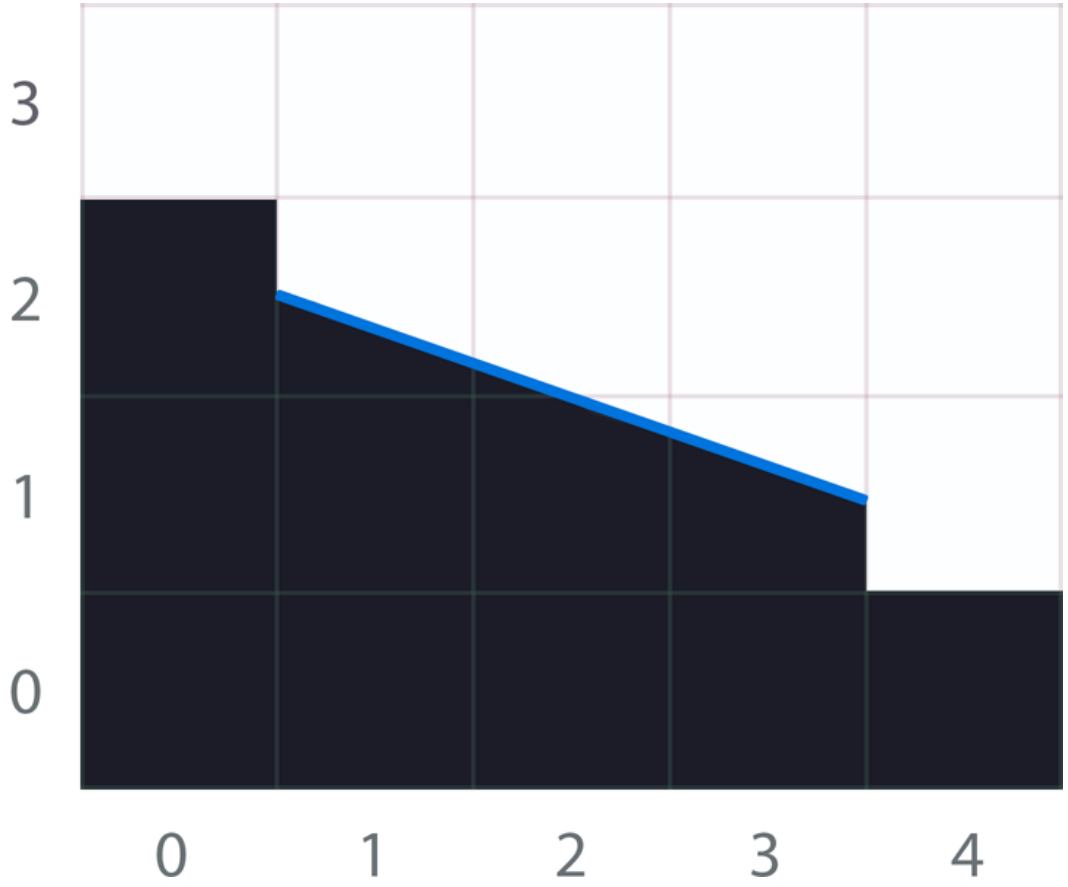
Morphological Antialiasing Basics



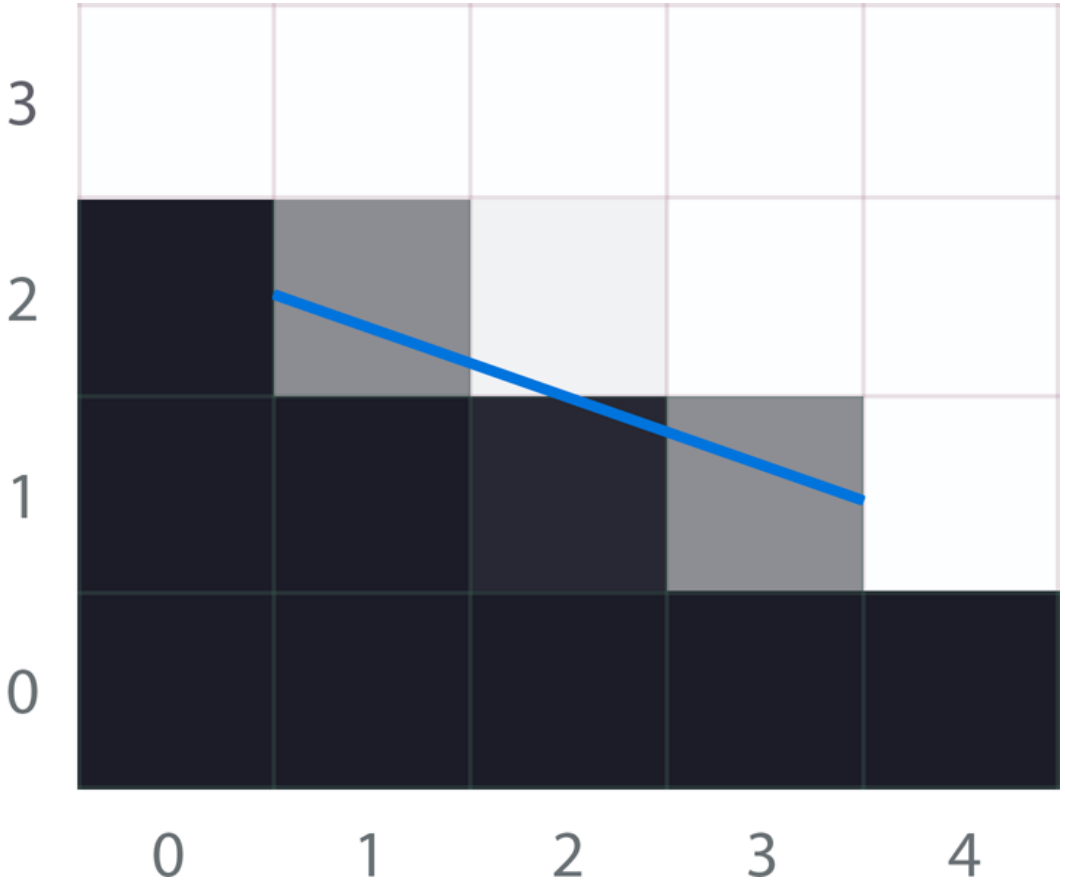
Morphological Antialiasing Basics



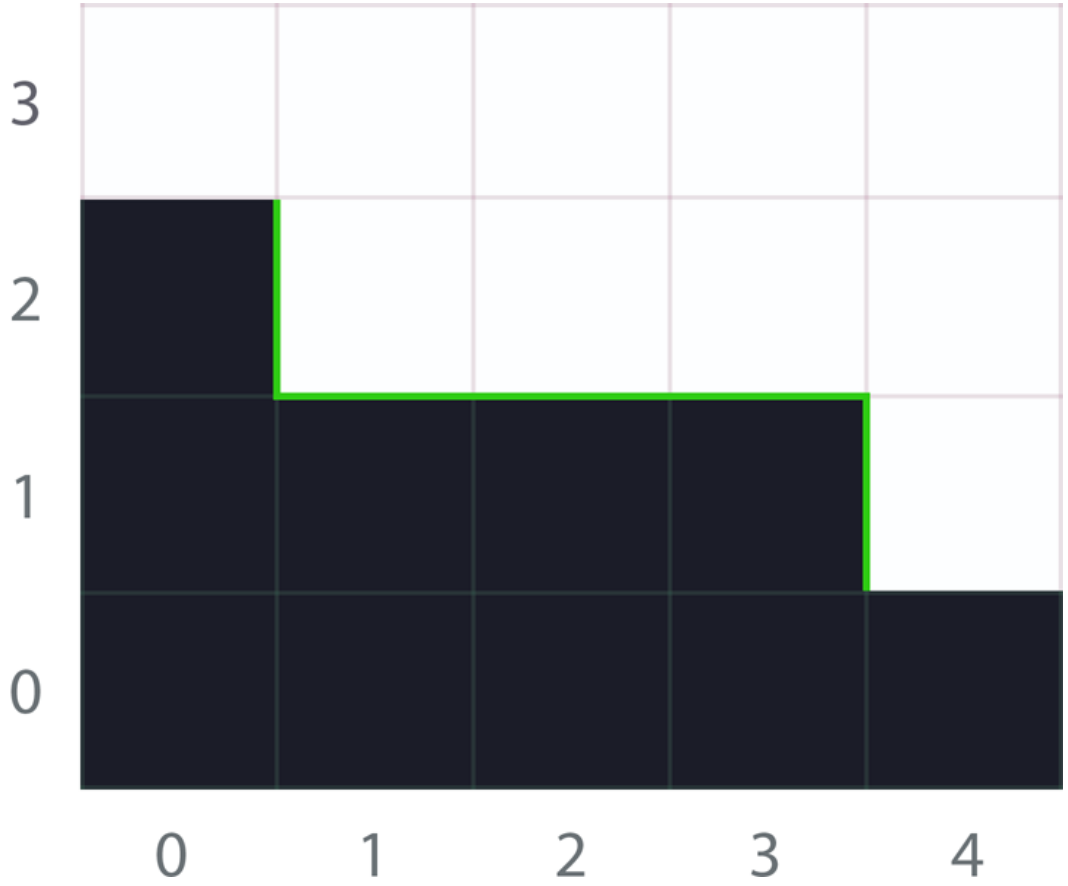
Morphological Antialiasing Basics



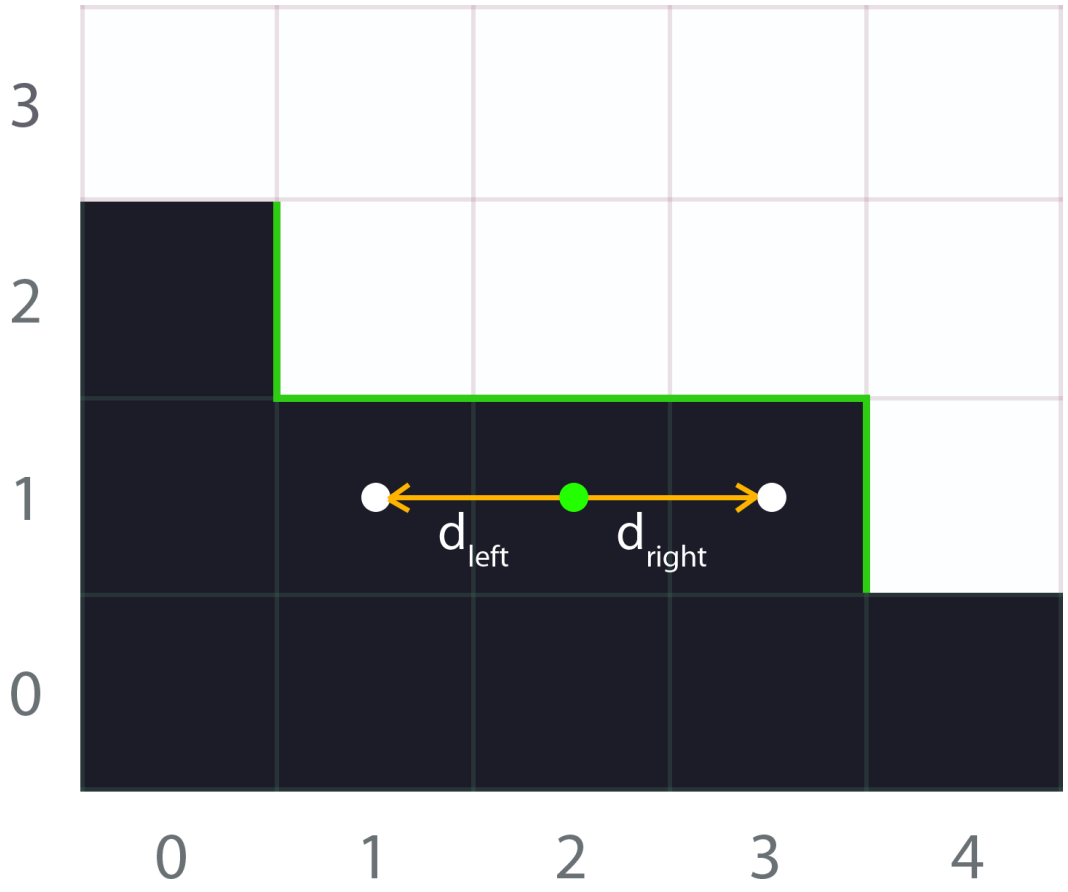
Morphological Antialiasing Basics



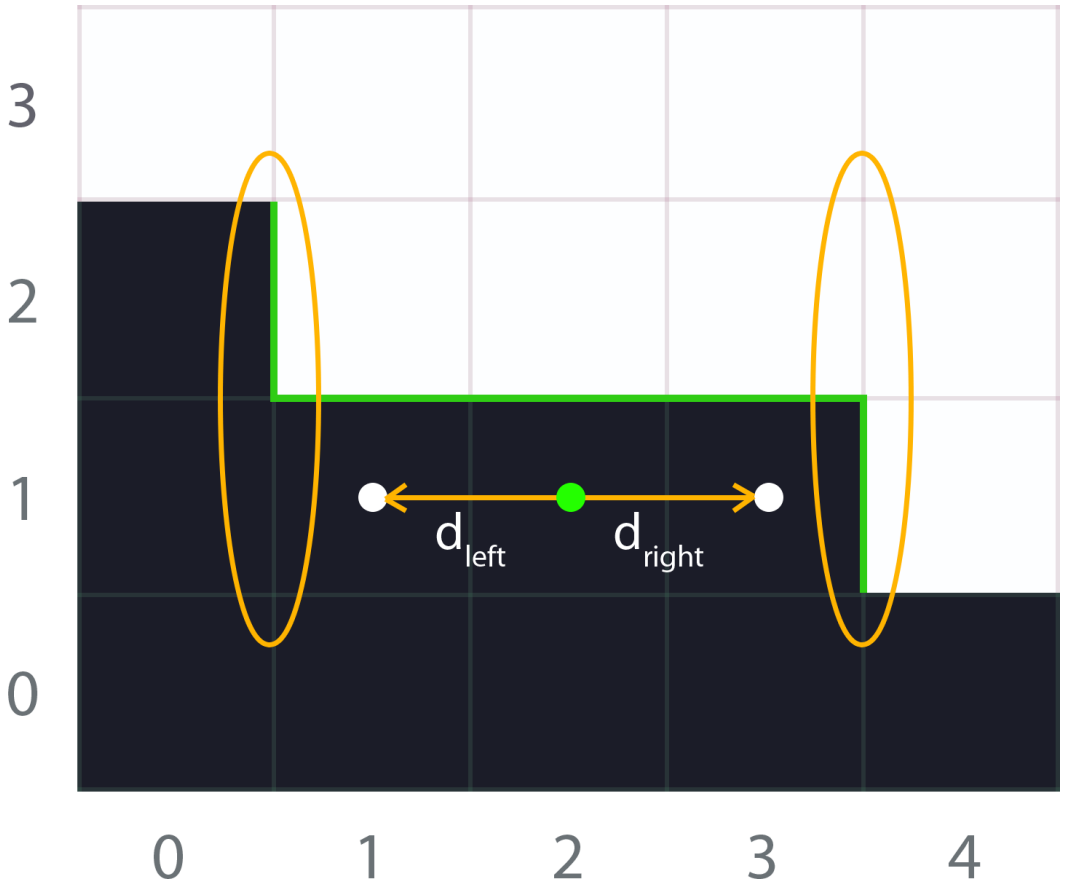
Morphological Antialiasing Basics



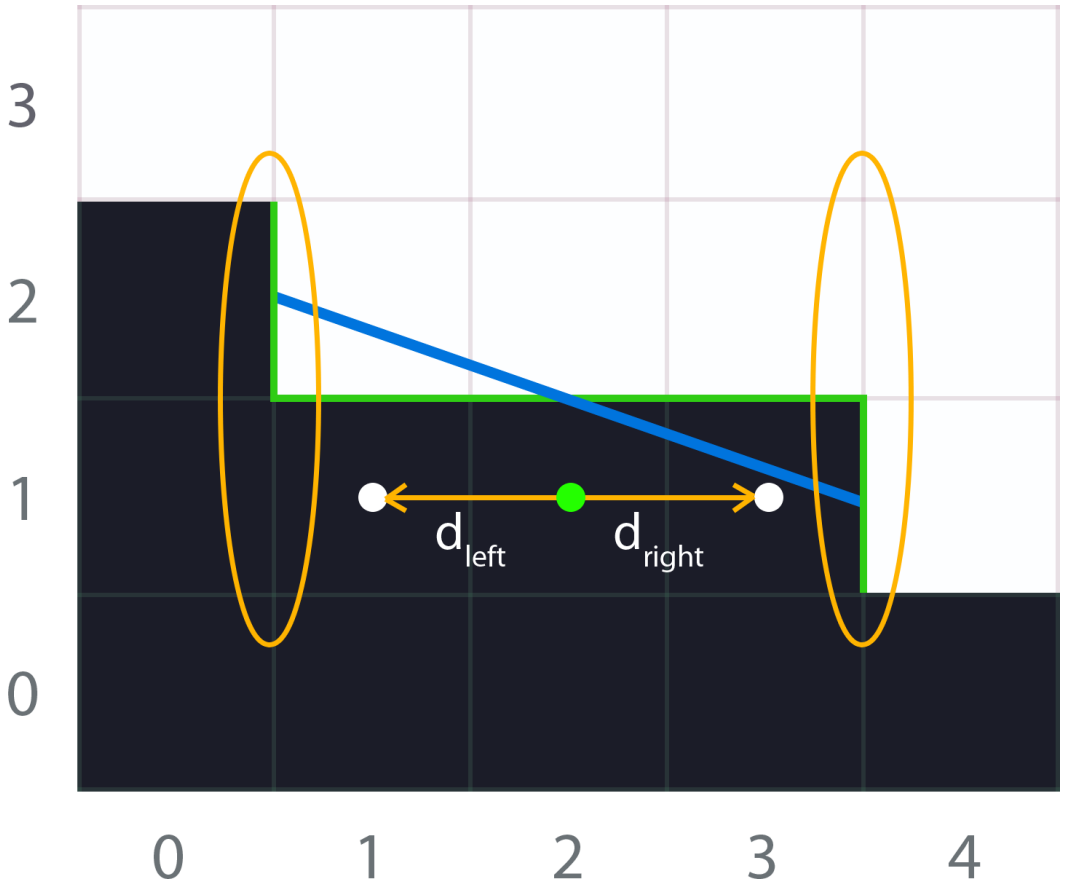
Morphological Antialiasing Basics



Morphological Antialiasing Basics



Morphological Antialiasing Basics



Morphological Antialiasing Improvements

Quality

- Morphological edge suppression
- Silhouette line detection
- U-shape smoothing

Performance

- Deferred queues
- Using the LDS
- Dual edge blending

Morphological Antialiasing Improvements

Quality

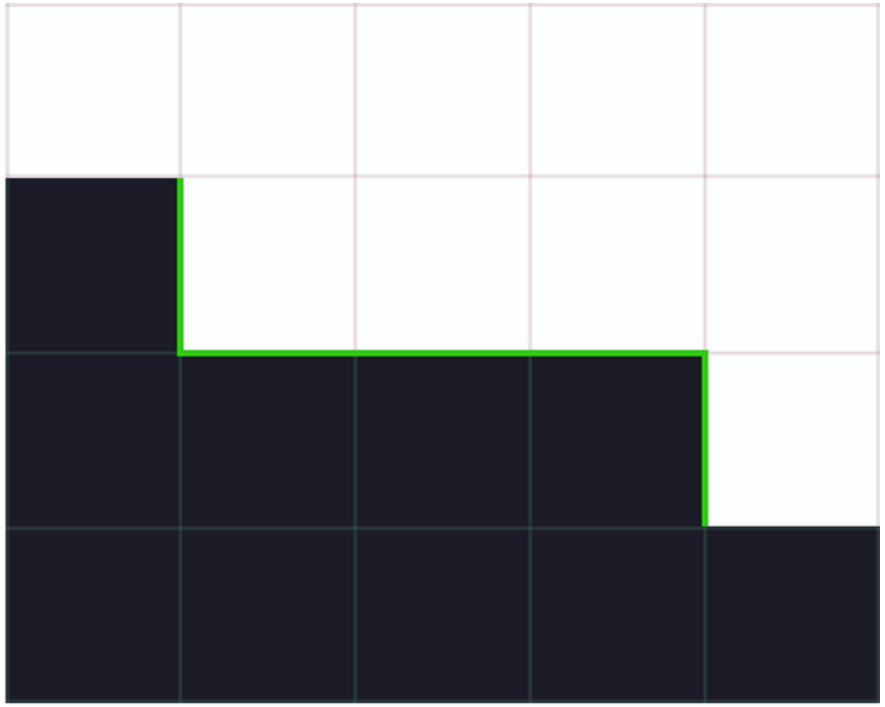
- Morphological edge suppression
- Silhouette line detection
- U-shape smoothing

Performance

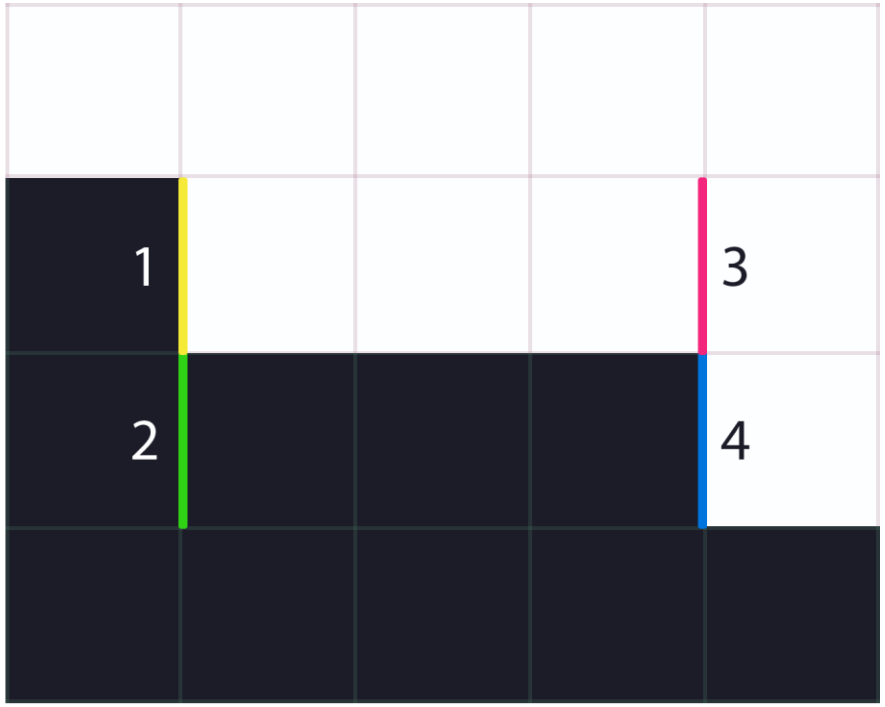
- Deferred queues
- Using the LDS
- Dual edge blending

Morphological:
Quality

Recap: Local Contrast Edge Suppression

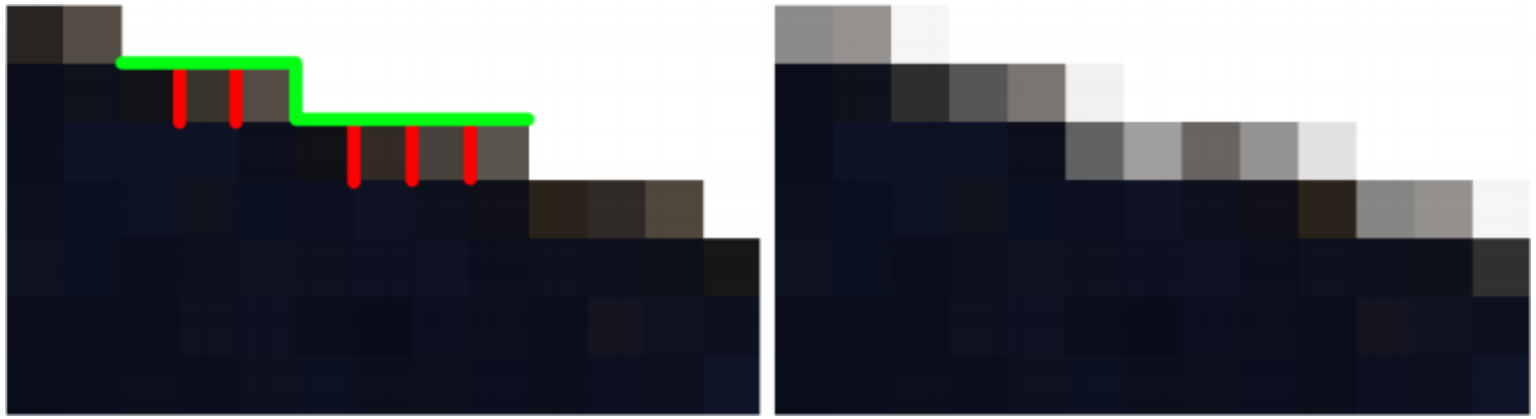


Searching for ends



We stop at Crossing Edges

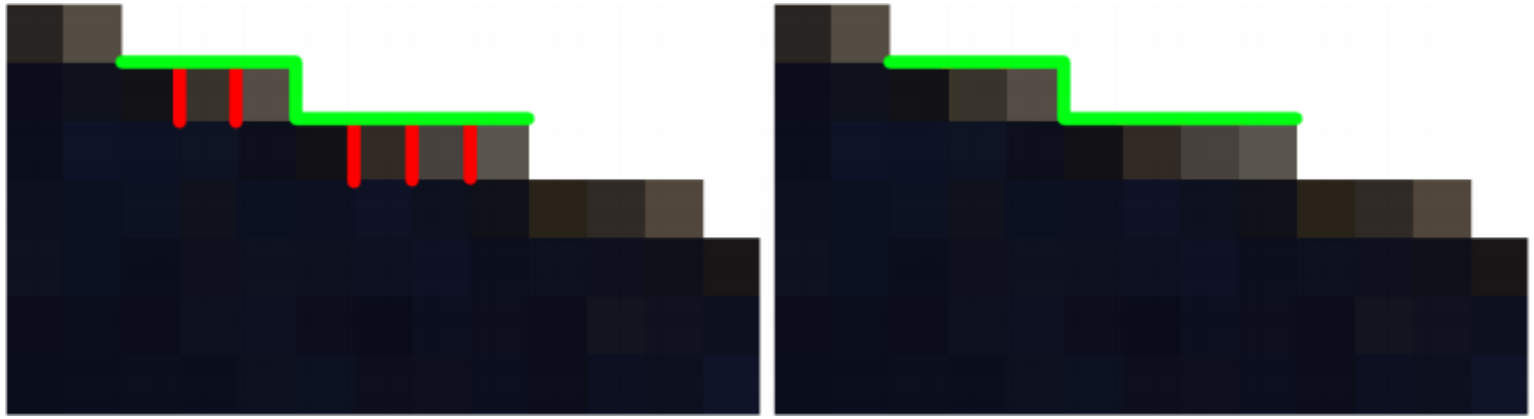
Recap: Local Contrast Edge Suppression



Input + Edge Overlay

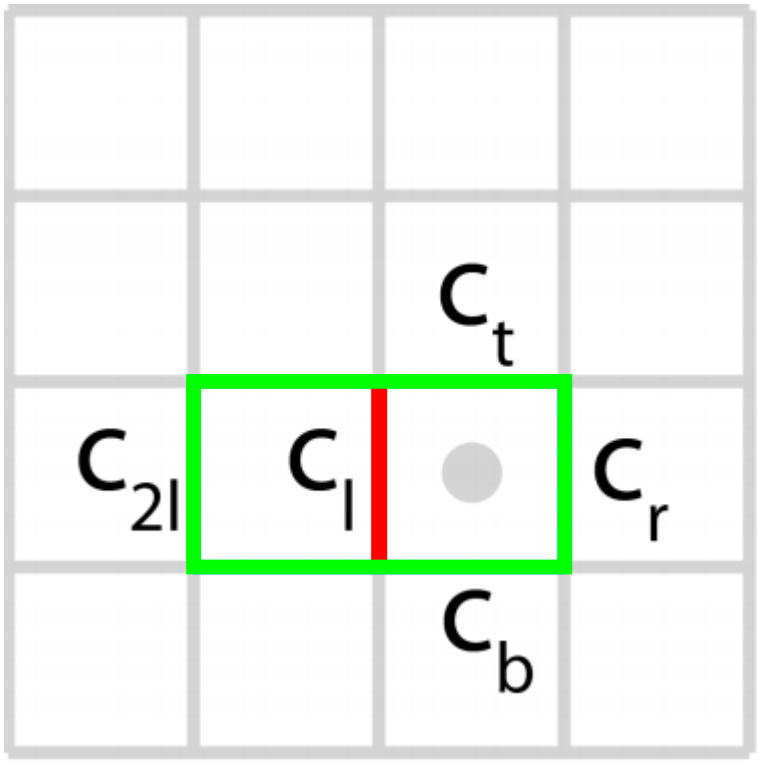
Morphological Output

Recap: Local Contrast Edge Suppression



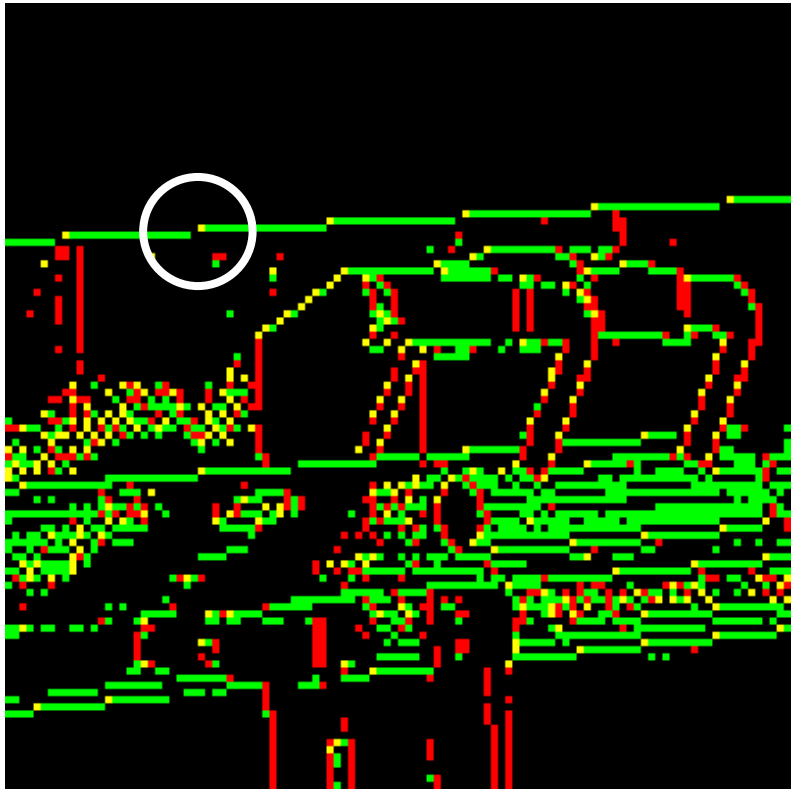
Before local contrast detection

After local contrast detection



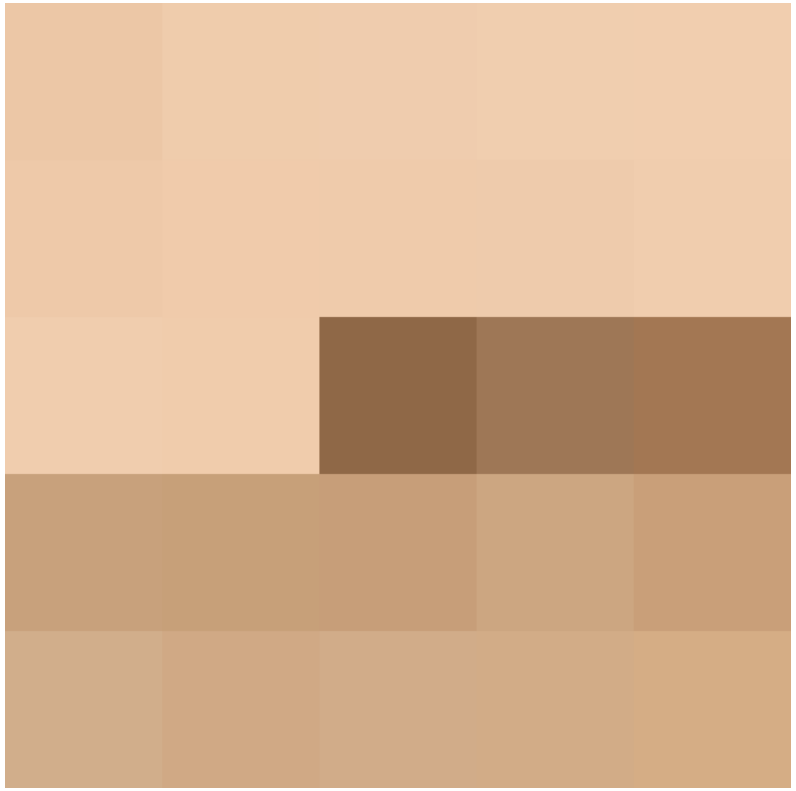
Failure Case

Input

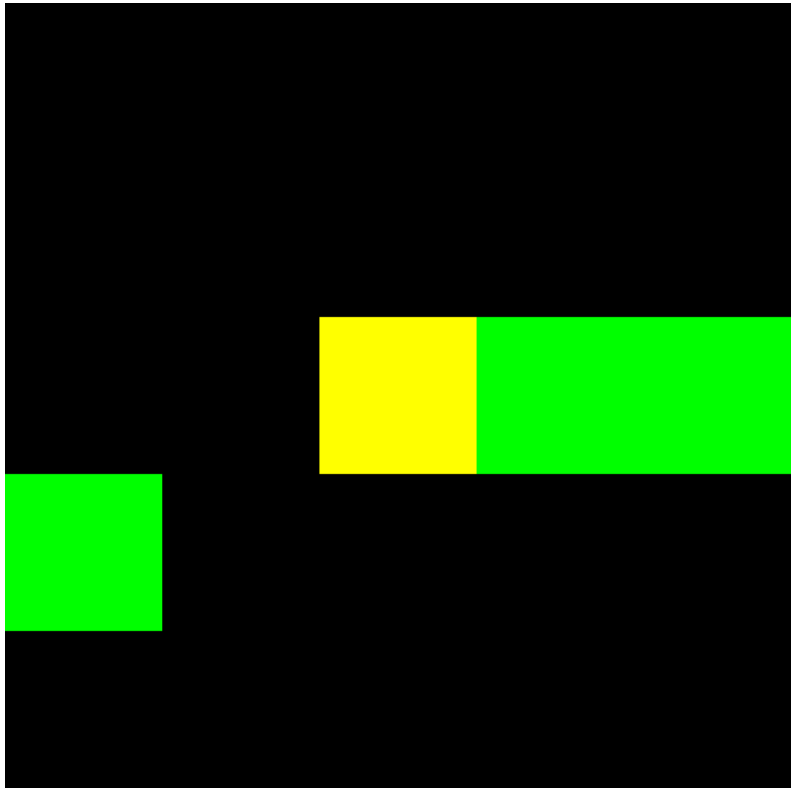


Edges

Failure Case



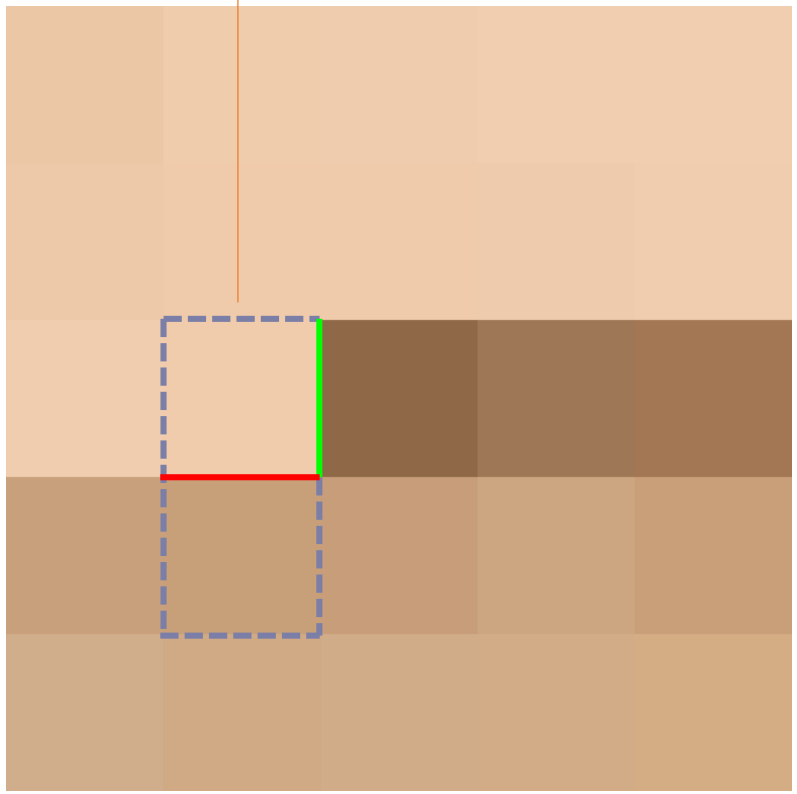
Input



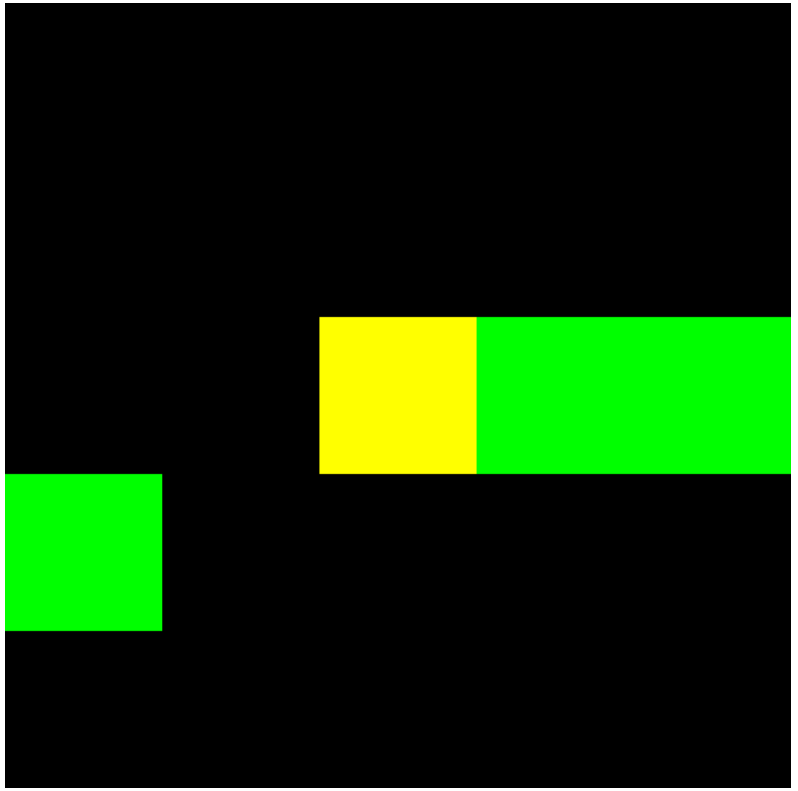
Edges

Failure Case

Local Contrast Edge Suppression



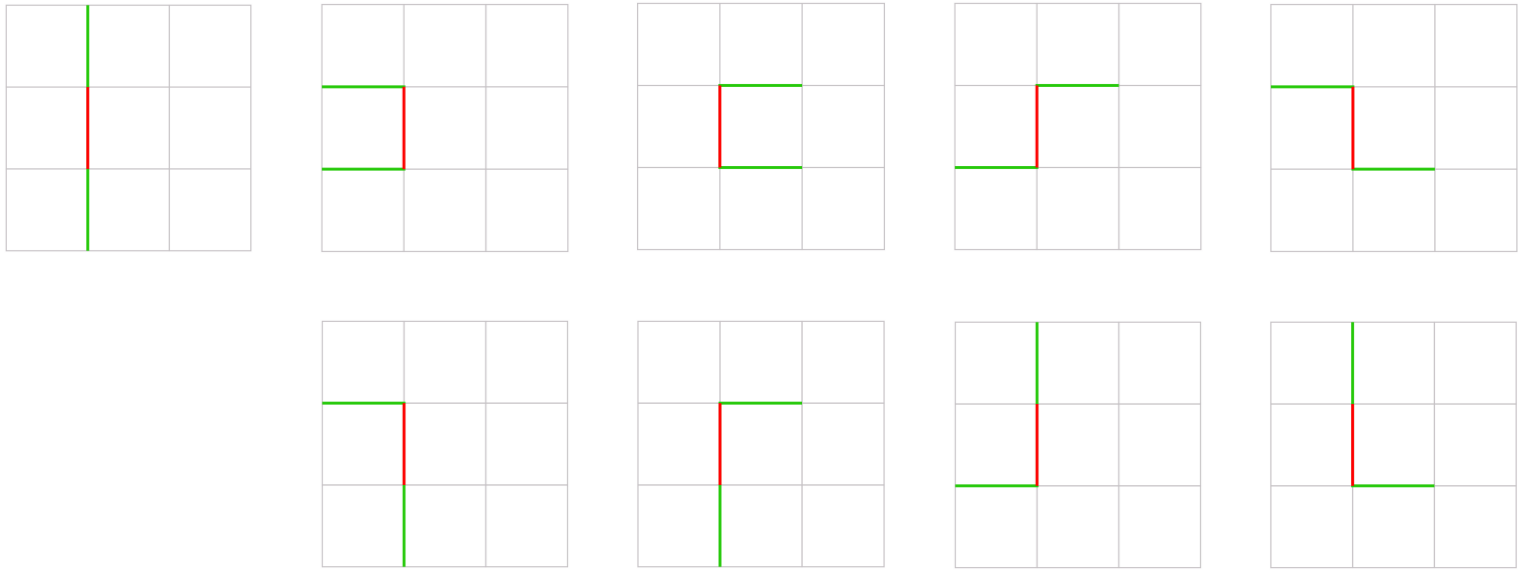
Input



Edges

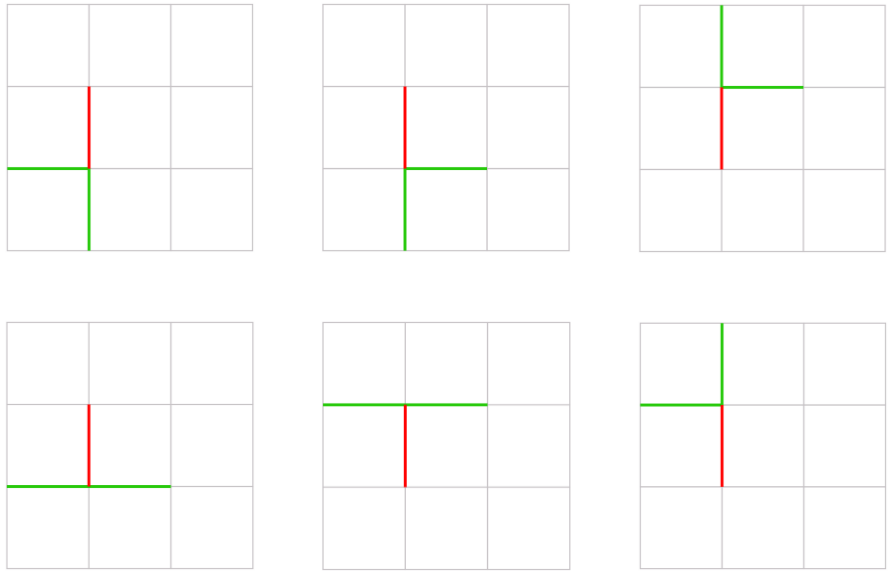
Morphological Edge Suppression

- Search for all patterns that **can be passing** through current edge



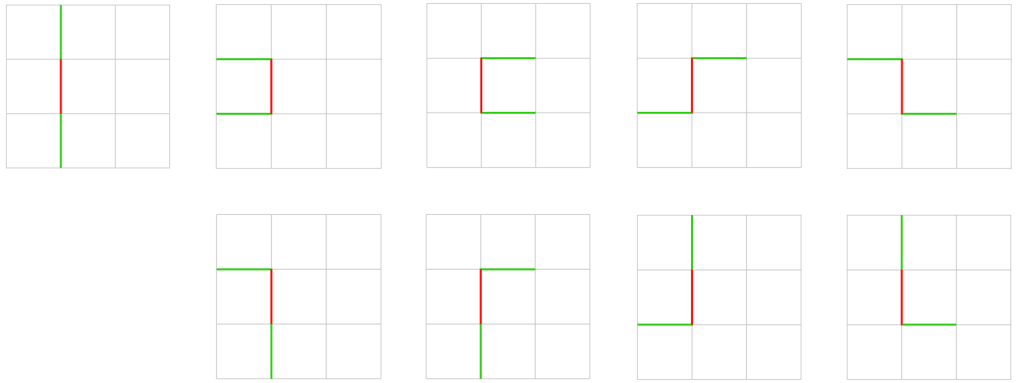
Morphological Edge Suppression

- Search for all patterns that **do not pass** through current edge



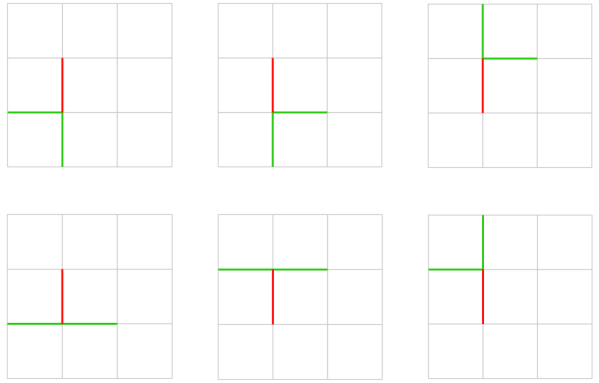
Morphological Edge Suppression

- Check which one has higher score (intensity)
- Suppress edge if a pattern that **do not** pass through current edge wins



Patterns that **pass** through current edge

VS



Patterns that **do not** pass through current edge

Local Contrast Edge Suppression

Close up #1

Close up #2

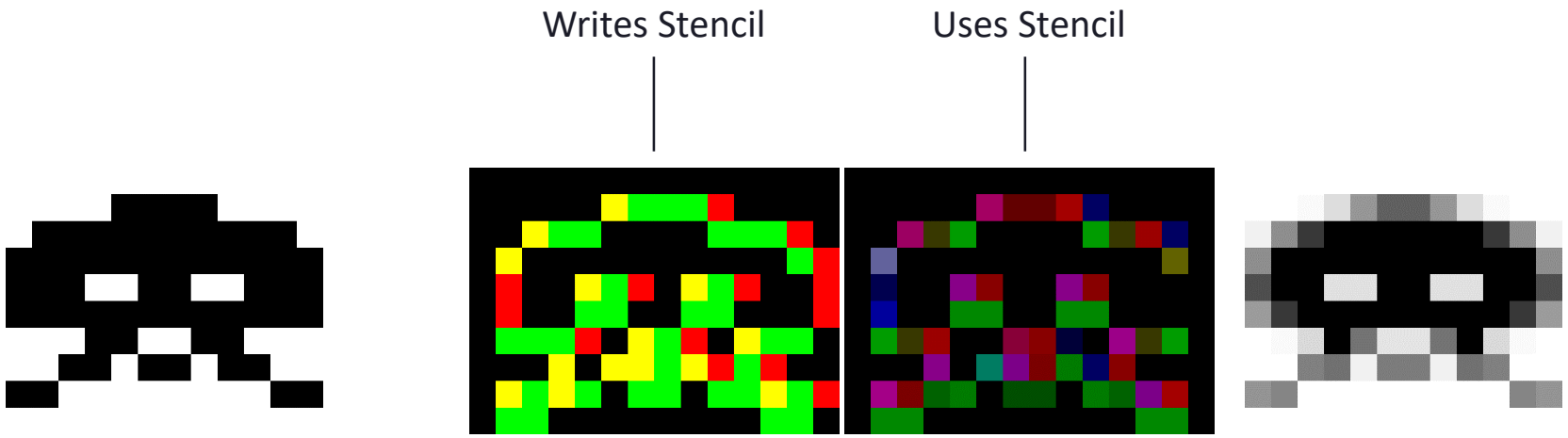
Morphological Edge Suppression

Close up #1

Close up #2

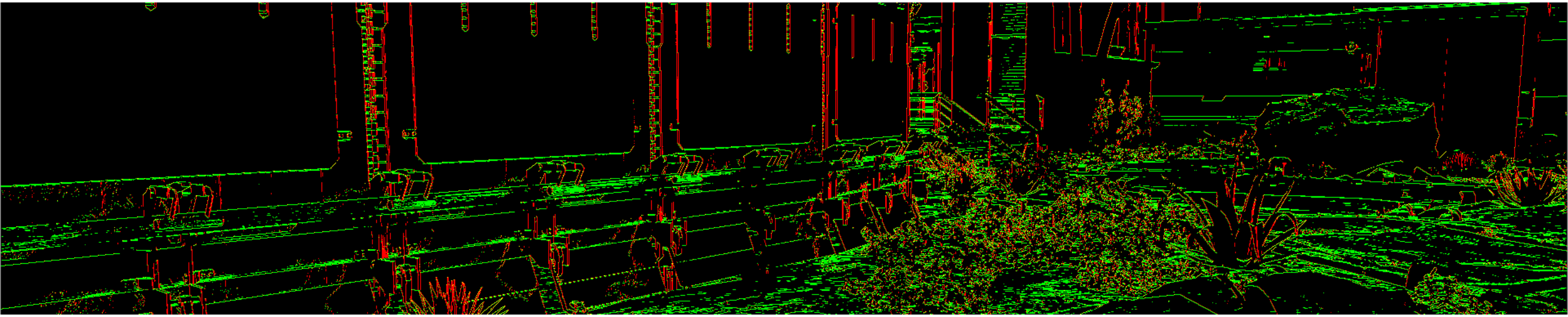
Morphological:
Performance

SMAA Morphological Multi Pass Approach



Pixel Shader Performance Inefficiencies

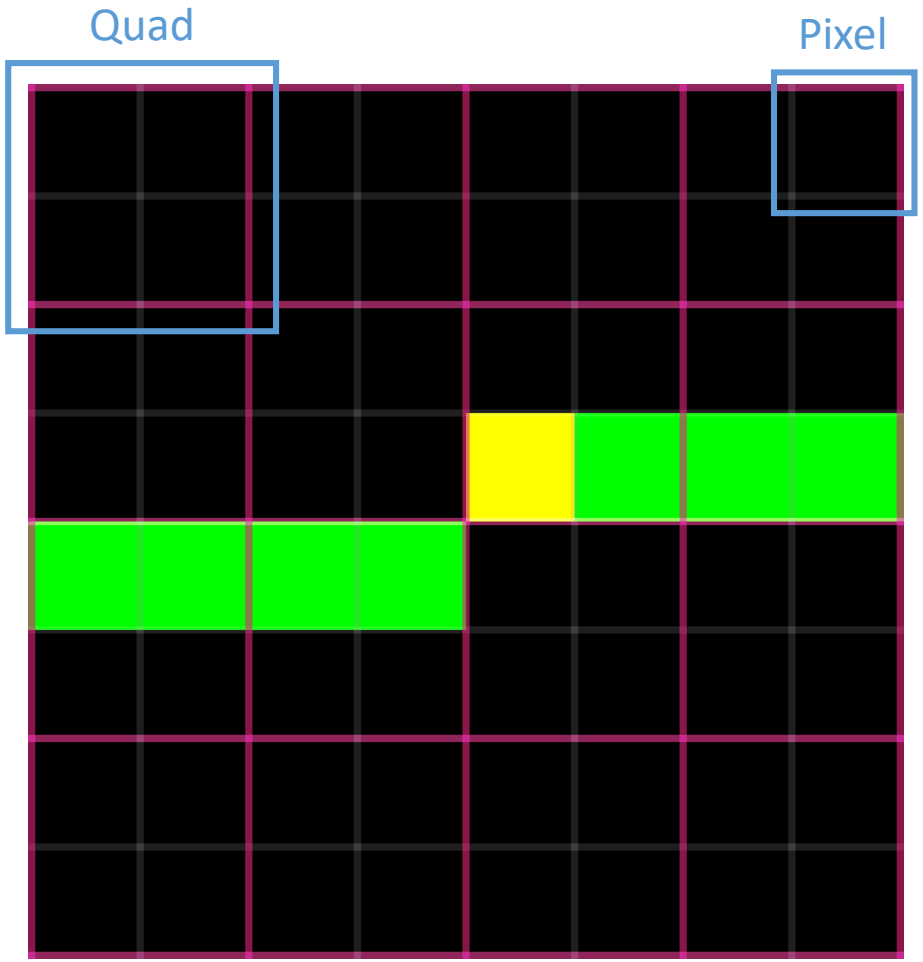
- Stenciling an edge detection texture is inefficient
- PS always scheduled in 2x2 quads
- Partial quads can fill a wave w/few active threads



Edge Detection Output

Stencil Culling Inefficiencies

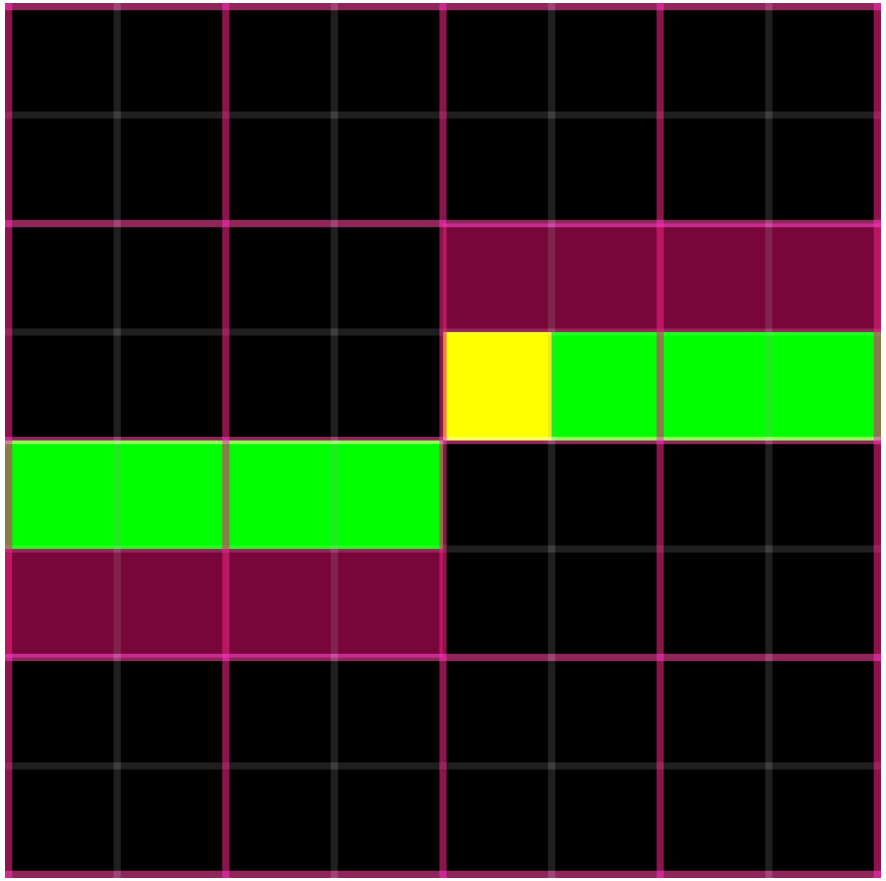
- Pixel shader runs in quads
  - For derivative support
- Stencil pass for antialiasing where we have edges
  - Yellow or green
- Pixel shader waves are packed from raster generated quads



8x8 Screen Tile

Stencil Culling Inefficiencies

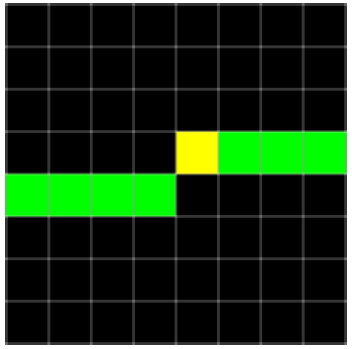
- A quad can be culled only if all its 4 samples are rejected
- All pixels in purple are packed in a PS wave but do nothing
  - Wastes efficiency



8x8 Screen Tile

Deferred Queues

- This problem was solved by **James Stanard's** MiniEngine FXAA, in the context of branching
  - https://github.com/Microsoft/DirectX-Graphics-Samples/blob/master/MiniEngine/Core/Shaders/FXAAPass1CS.hlsli
  - https://github.com/Microsoft/DirectX-Graphics-Samples/blob/master/MiniEngine/Core/Shaders/FXAAPass2CS.hlsli
- Detected edges are appended to an append/consume buffer (about 2500 edges on a typical scene)
- An indirect dispatch is executed to consume this buffer
- This allows for all threads to do actual work

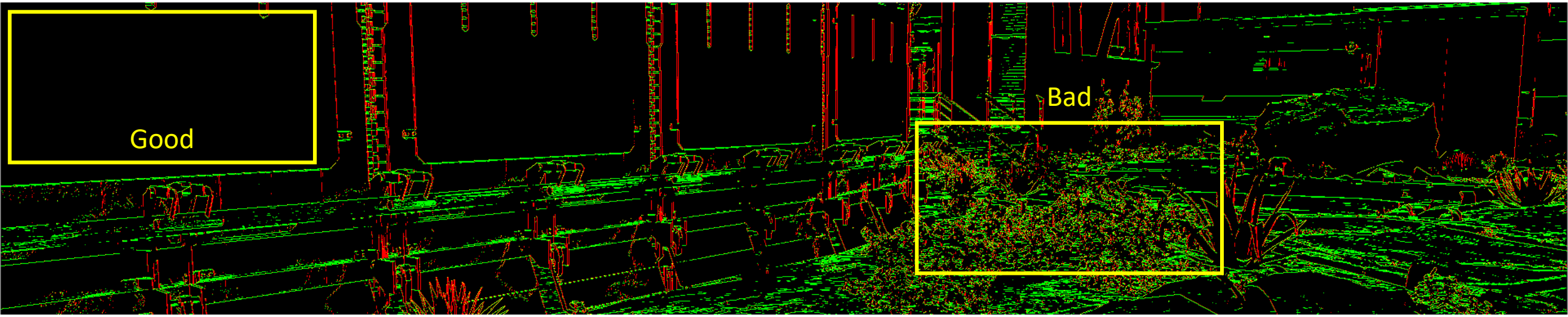


8x8 Screen Tile

64x1 Compute Shader Indirect Dispatch

Deferred Queues

- Avoids reading the fullscreen stencil buffer
- Hi-stencil not always optimal due to scattered edge distribution in some cases



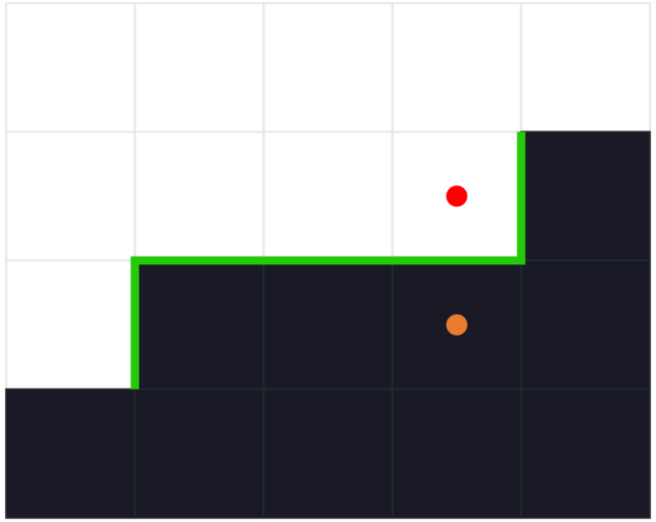
Edge Detection Output

Deferred Queues

- We adapted SMAA to use a similar approach
  - Significantly improved the performance
- Also has application for temporal
  - More on this later on

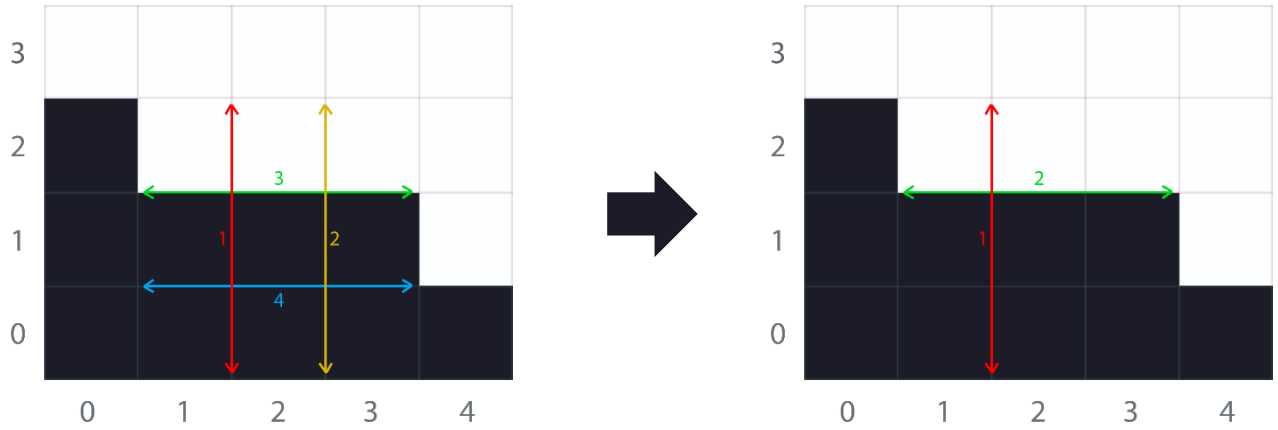
Repeated Pattern Searches

- Both red and orange pixels will search for the same pattern
- Same pattern search can **in theory** antialias both pixels



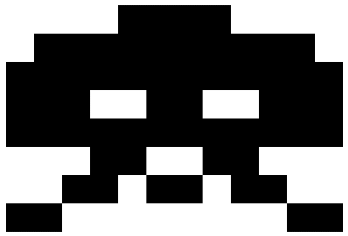
How SMAA Dealt with This Problem with PS

- Up to 4 lines can pass through a pixel
- **Solution: pixels share edges, don’t repeat calculations!**

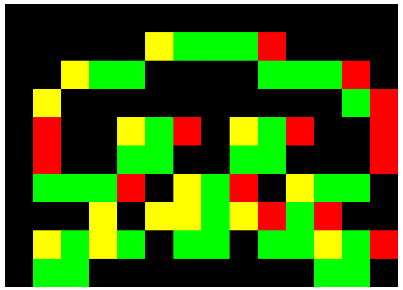


How SMAA Dealt with This Problem with PS

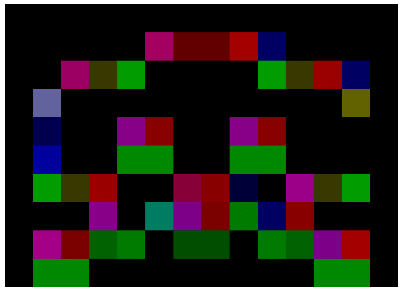
Split Pattern Search and Antialiasing



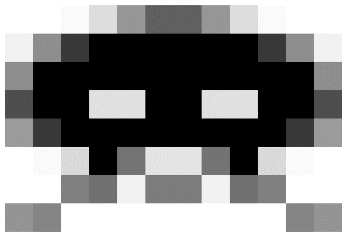
Original Image



1st Pass
Edge Calculation



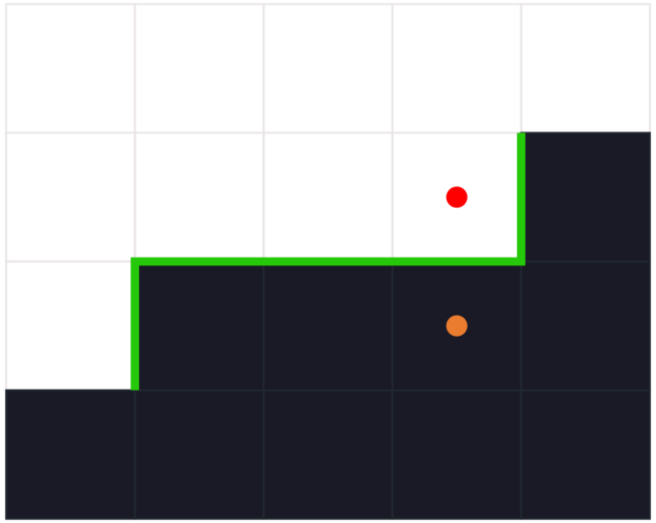
2nd Pass
Pattern Search



3rd Pass
Antialiasing

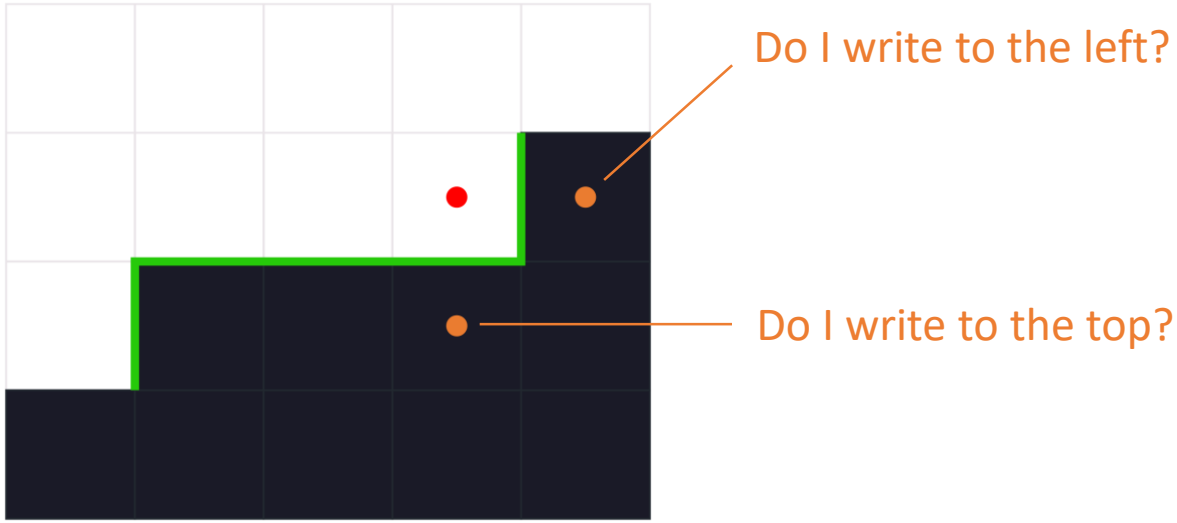
How SMAA Solves This Problem with CS

- **Modern Era:** we can now scatter values by using RW textures
- Pattern search for the orange pixel and antialias both orange and red pixels
- Each pixel searches top or left boundaries only (the one with higher contrast)



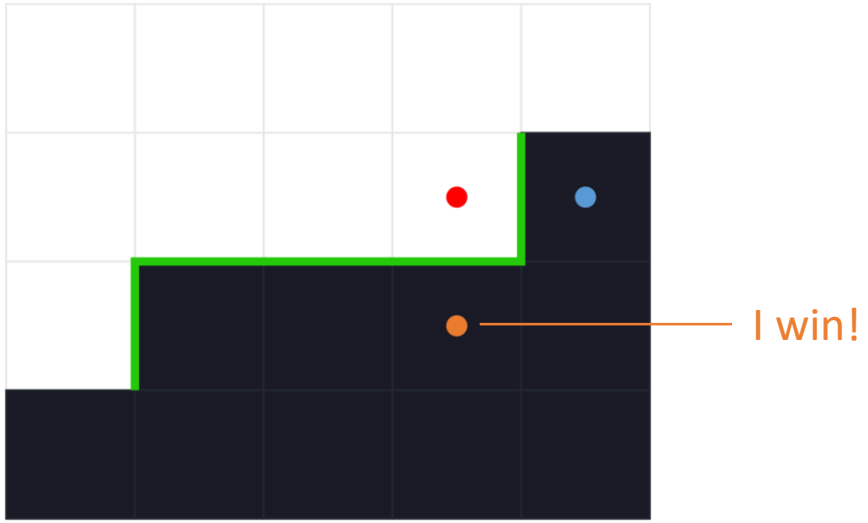
How SMAA Solves This Problem with CS

- **Problem:** race conditions!
- Both orange pixels and the own red could write into the red position
- Each frame a different one can win (flickers)



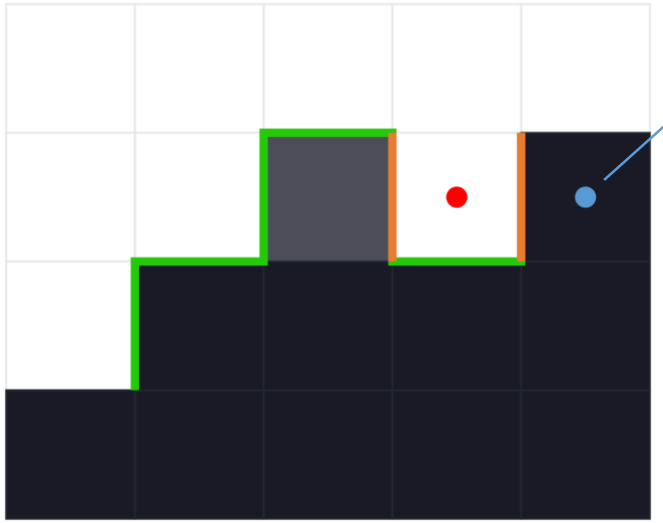
How SMAA Solves This Problem with CS

- **First step:** serialize execution
- Vertical edges then horizontal



How SMAA Solves This Problem with CS

- **Problem:** we can have race conditions within a direction
  - Red and blue and write to the red position
  - Remember: both look to the edge in their left
- **Second step:** get edges to agree during edge detection
  - Checking neighbor decision

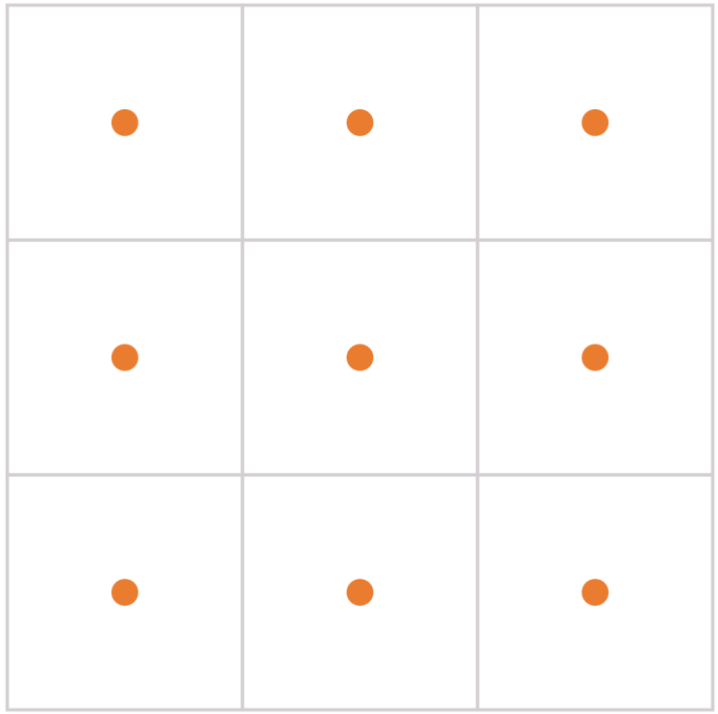


Performance Recap

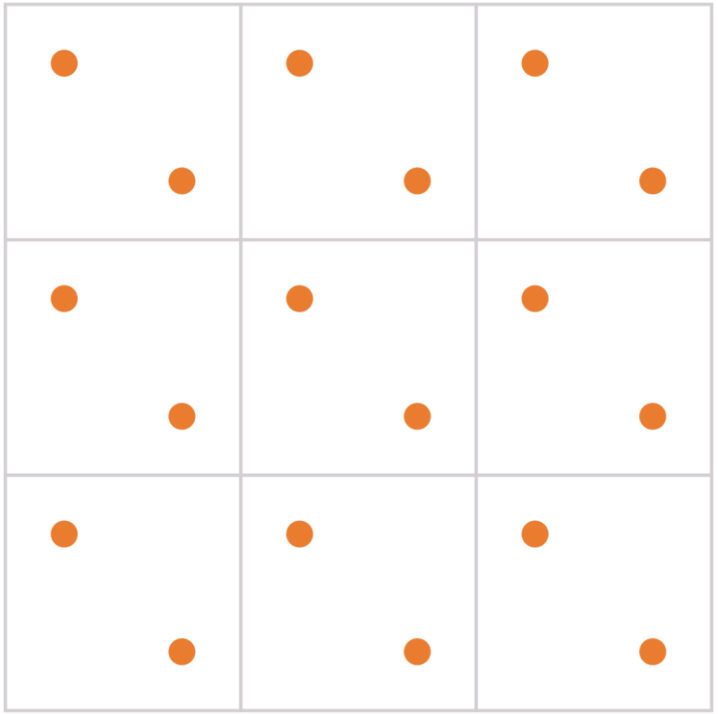
- PS4@1080p
- Deferred queues + Dual edge blending
  - 0.16ms
- LDS:
  - 0.14ms
- Brought performance back from (note that varies per scene):
  - 0.774ms to 0.475ms
- Faster than FXAA

Temporal

Temporal Antialiasing Toolset:
Jittering

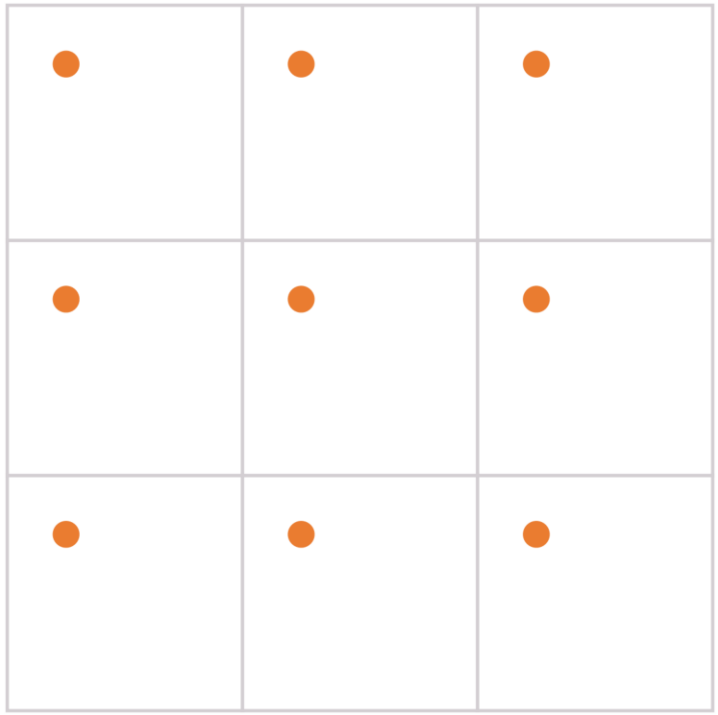


Regular Grid

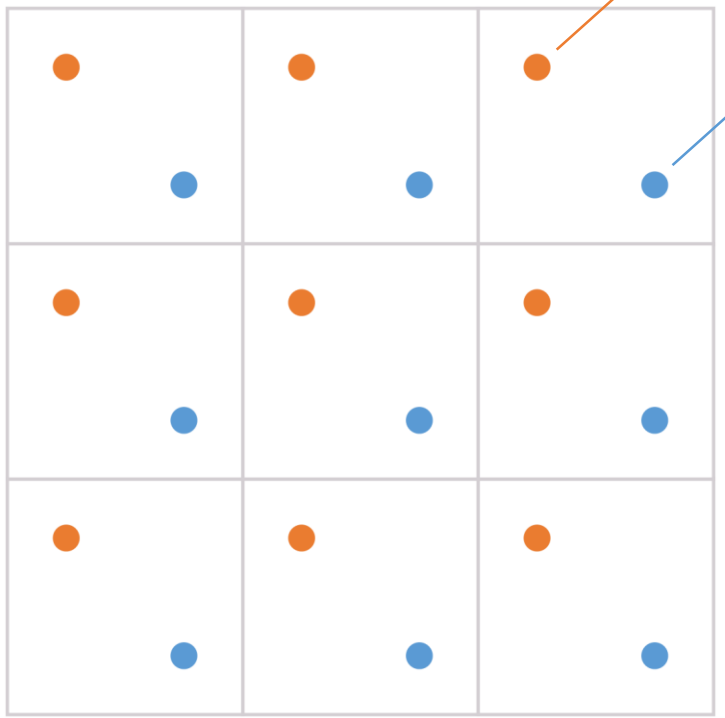


SSAA 2x

Temporal Antialiasing Toolset:
Jittering



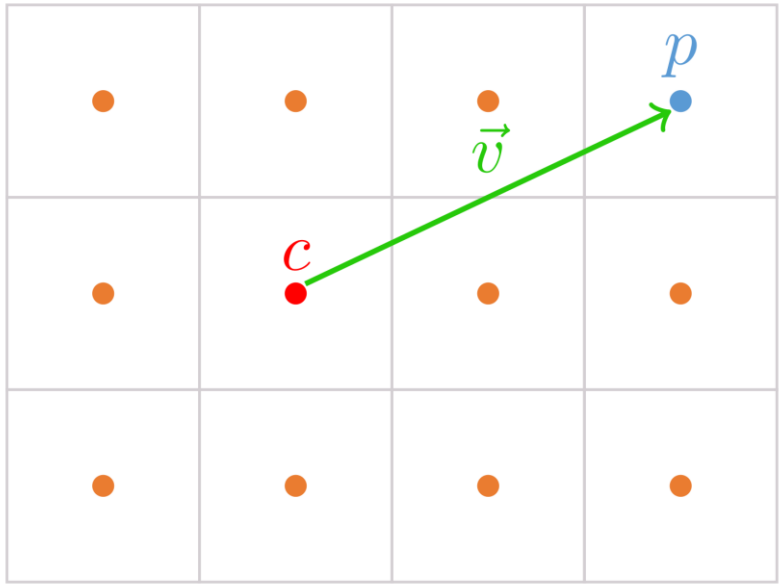
Current Frame



SMAA T2x

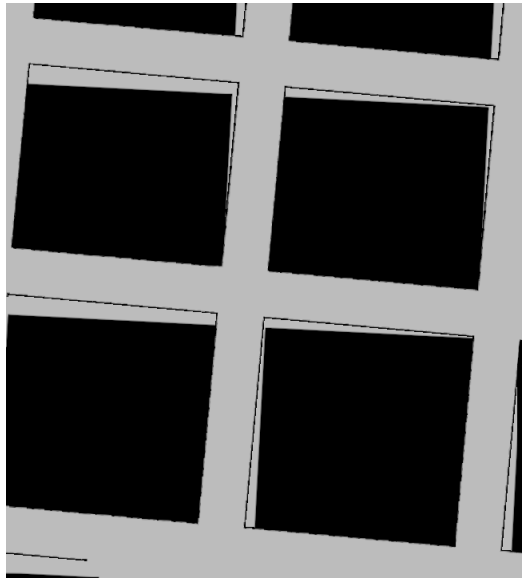
Temporal Antialiasing Toolset:
Temporal Reprojection

- For motion:
  - Reproject current red pixel to previous frame position using motion vectors

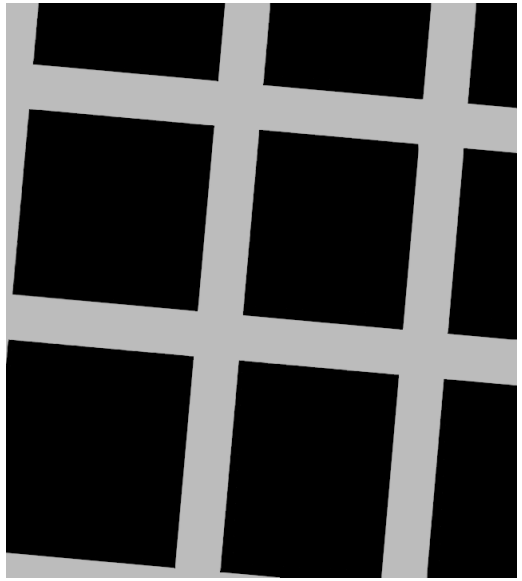


Temporal Antialiasing Toolset:
Disocclusion and Velocity Weighting

- **Disocclusion:** pixels in the new frame not present in previous one
- **Solution:** don't blend if velocities are too different
  - Don't work for pixels without velocities (alpha blended)



Without Velocity Weighting

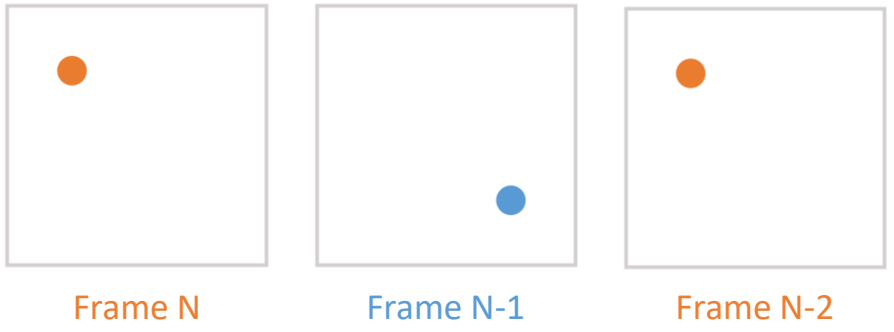


With Velocity Weighting

[Sousa2011][Jimenez2012]

Temporal Antialiasing Toolset:
Disocclusion and Color Weighting

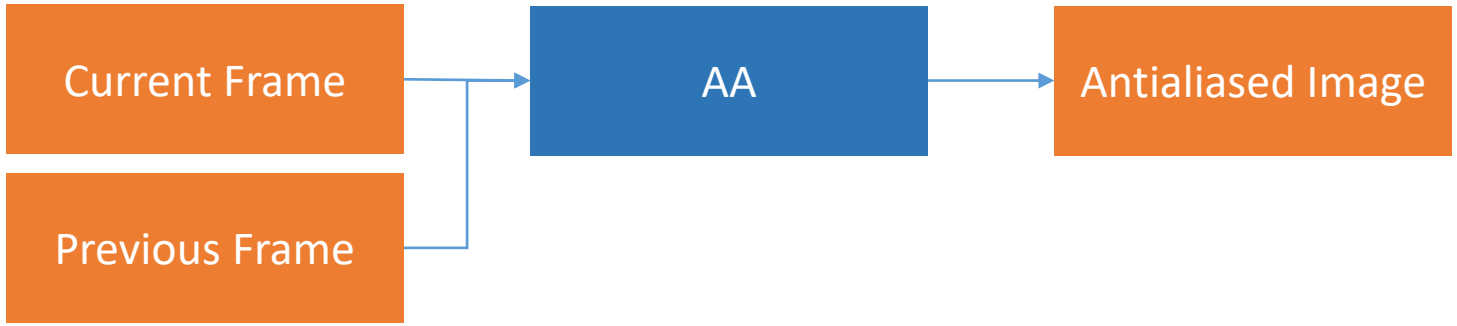
- Color information can also be used for disocclusions
  - Together with velocity weighting
- Can’t compare current and previous frame
  - Different jitters: would often reject even on static images
- **Solution:** Compare Frame N with Frame N-2 [Drobot2014]
  - Mitigates ghosting on objects without velocities (alpha blended)



[Sousa2011][Jimenez2012]

Temporal Antialiasing Toolset:
Exponential History / Exponential Moving Average

- SMAA T2x and similar techniques use two frames
  - $c_{aa} = 0.5c_i + 0.5\ c_{i-1}$



Temporal Antialiasing Toolset:
Exponential History / Exponential Moving Average

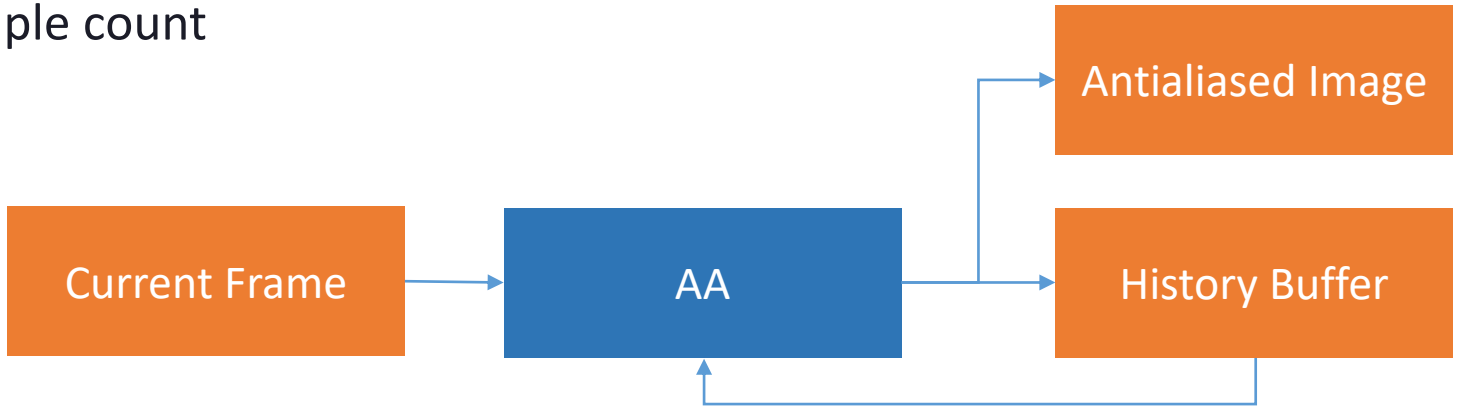
- We can leverage an exponential accumulation buffer:

Blend Factor

$$h_{i+1} = \alpha c_i + (1 - \alpha)\, h_{i-1}$$

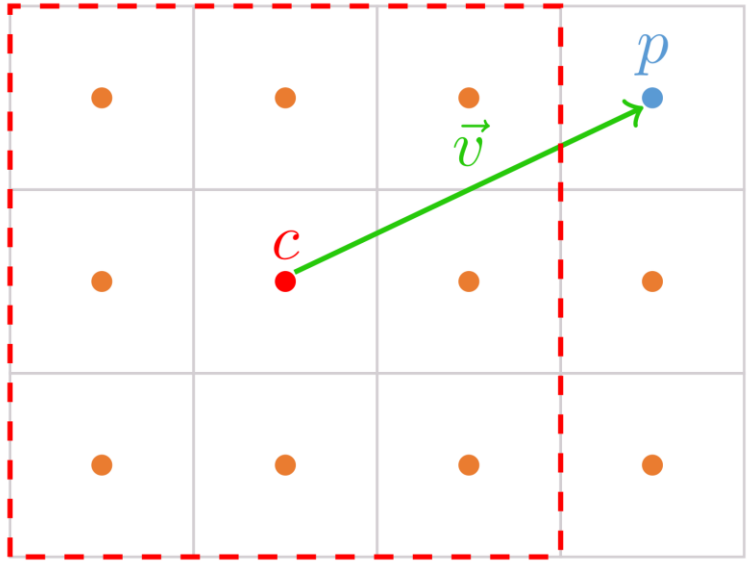
Next exponential history sample — Current frame sample — Previous exponential history sample

- Similar to moving average of multiple frames [Karis2004]
- Increase effective subsample count
- Feedback loop

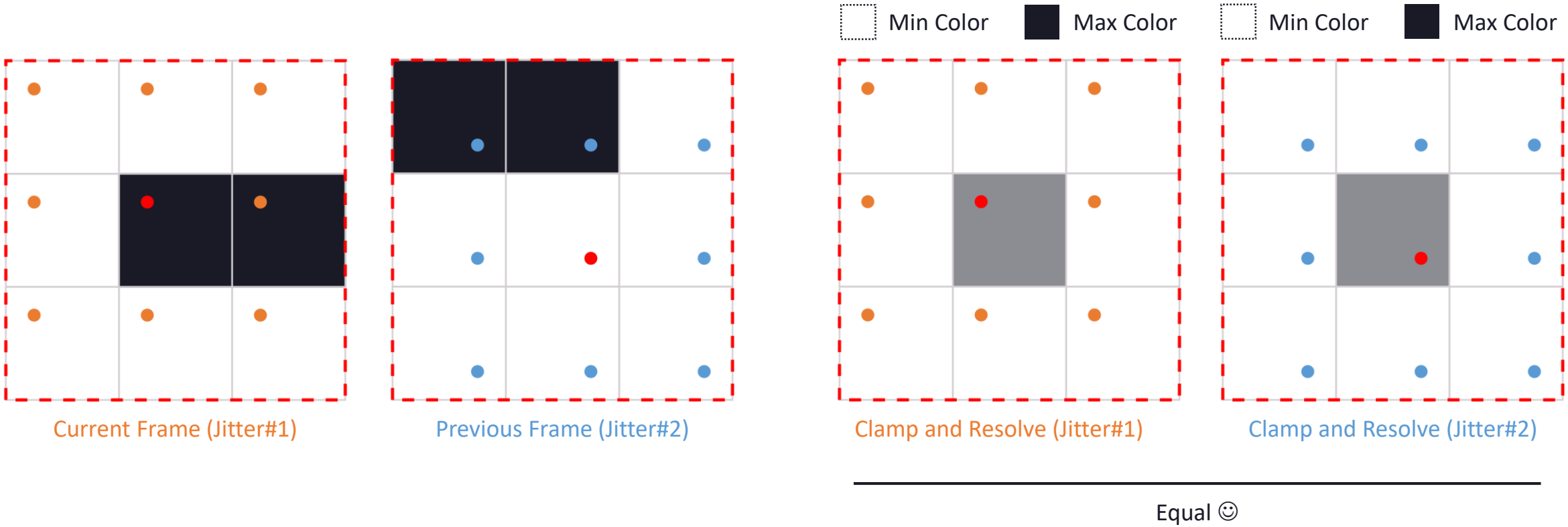


Temporal Antialiasing Toolset: Neighborhood Clamp

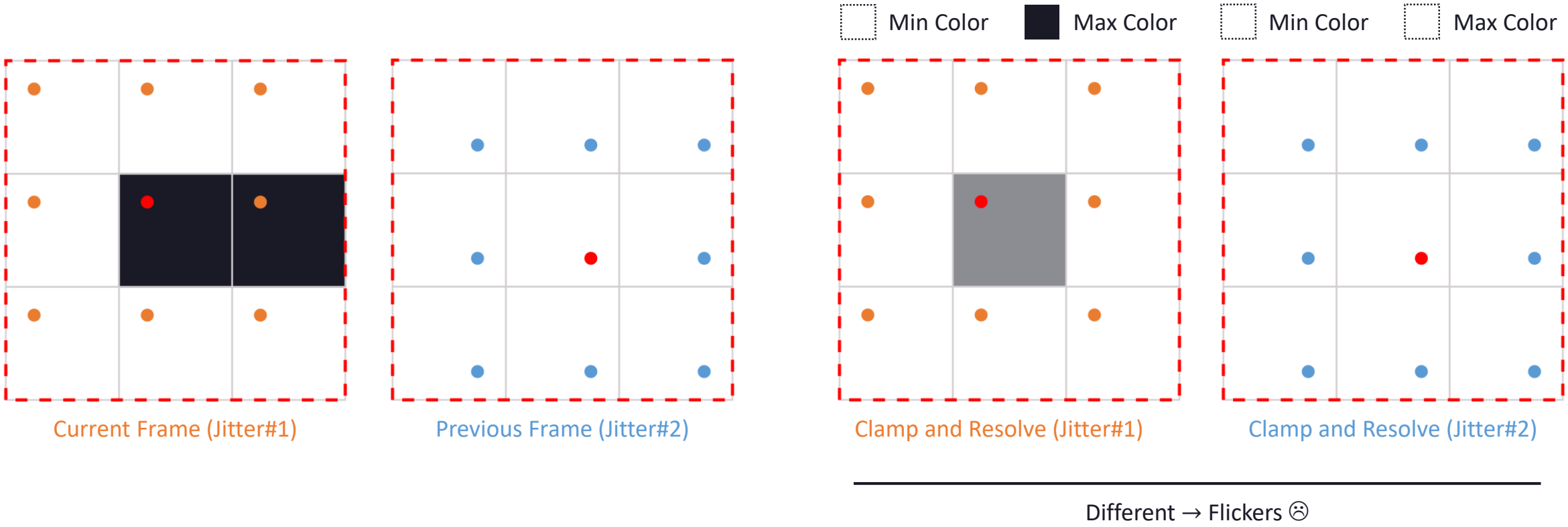
- Exponential accumulation buffer techniques typically use **neighborhood clamping** for disocclusion [Lottes2011]
- On its basic form:
  - $p = \max(\min(p, n_{max}), n_{min})$
- n_{min} and n_{max} and are min and max colors in the 3x3 dashed neighborhood



Temporal Antialiasing Toolset:
Neighborhood Clamp Flickering Problem

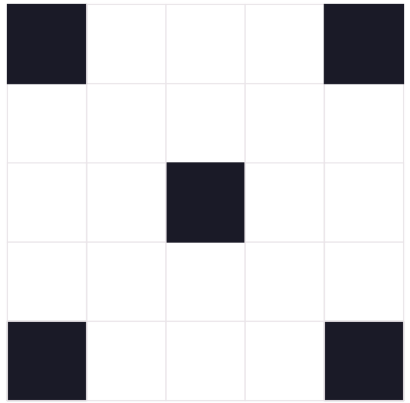


Temporal Antialiasing Toolset:
Neighborhood Clamp Flickering Problem

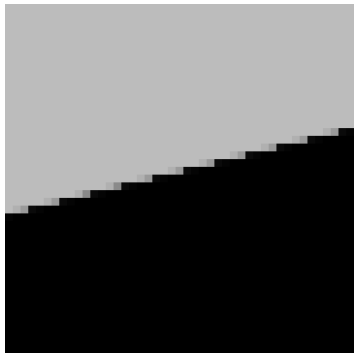


Temporal Antialiasing Toolset: Nearest Velocity

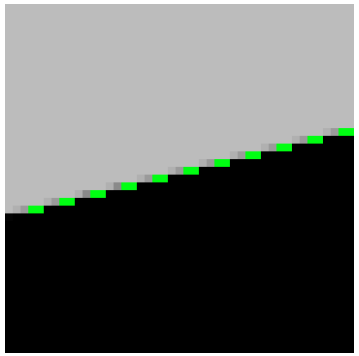
- Exponential history is smooth and antialiased
- Fresh frame velocities are aliased
  - Will introduce aliasing when reprojecting
- **Solution:** take front most neighborhood velocity [Karis2014]



Velocity Buffer



Antialiasing Output



Pixels that lost AA

Temporal Antialiasing Toolset: More!

- Shaped neighborhood YCoCg clipping [Karis2014]
- Soft neighborhood clamping [Drobot2014]
- Higher order resampling [Drobot2014]
- Variance clipping [Salvi2016]
- ...

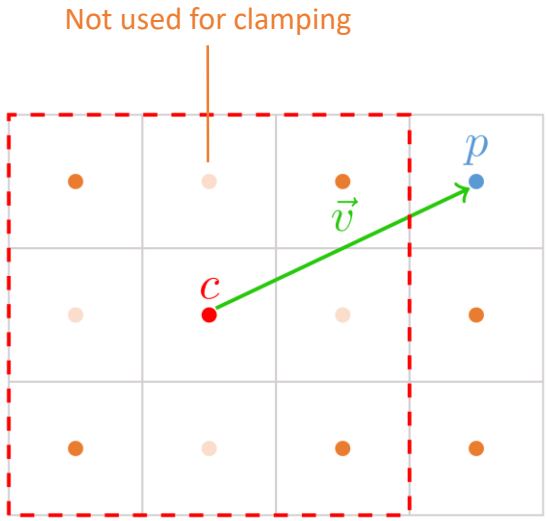
Previous Work

SMAA 1TX [Sousa2013]

- SMAA morphological for edges
- No subpixel jitter
- Simplified neighborhood clamping
  - Using diagonals only
- Spatial-contrast exponential history
  - Blend with history more strongly on high contrast areas
  - Helps to keep texture details in motion

$$c_{max} = \max(c_{TL}, c_{TR}, c_{BL}, c_{BR})$$
$$c_{min} = \min(c_{TL}, c_{TR}, c_{BL}, c_{BR})$$
$$c_{acc} = clamp(c_{acc}, c_{min}, c_{max})$$
$$w_k = \left|(c_{TL} + c_{TR} + c_{BL} + c_{BR}) * 0.25 - c_M\right|$$
$$w_{rgb} = clamp\left(\frac{1}{K_{lowfreq} * (1 - w_k) + K_{hifreq} * w_k}, 0, 1\right)$$
$$c = c_M * (1 - w_{rgb}) + c_{acc} * w_{rgb}$$

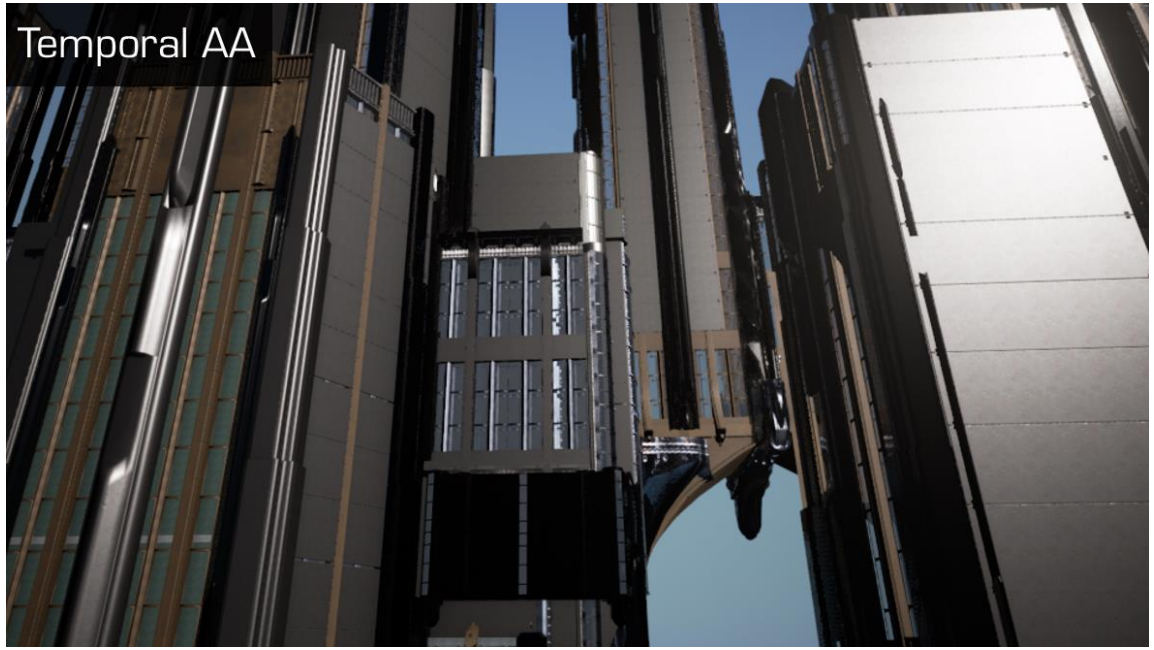
Spatial-Contrast Exponential History



Simplified neighborhood clamping

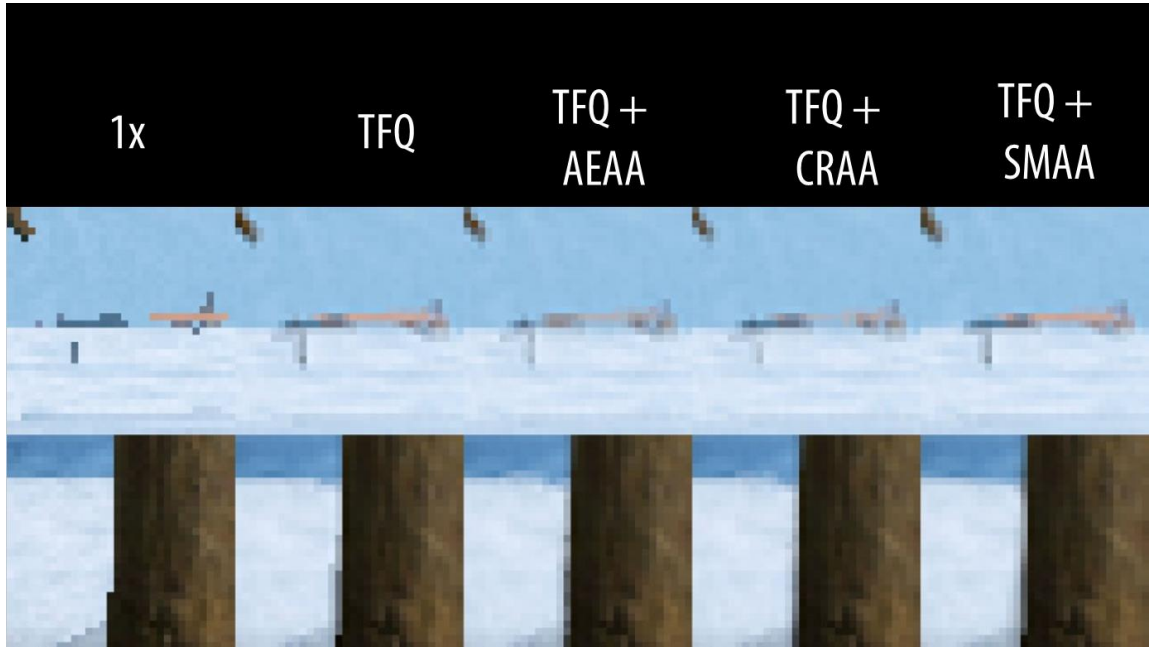
Unreal Engine 4 AA [Karis2014]

- 8x subpixel jittering
  - Halton(2,3)
- Exponential history
- Advanced neighborhood clamping
  - Shaped neighborhood YCoCg clipping
  - Tighter clamp (reduced ghosting)
- Nearest velocity
- Distance to clamp

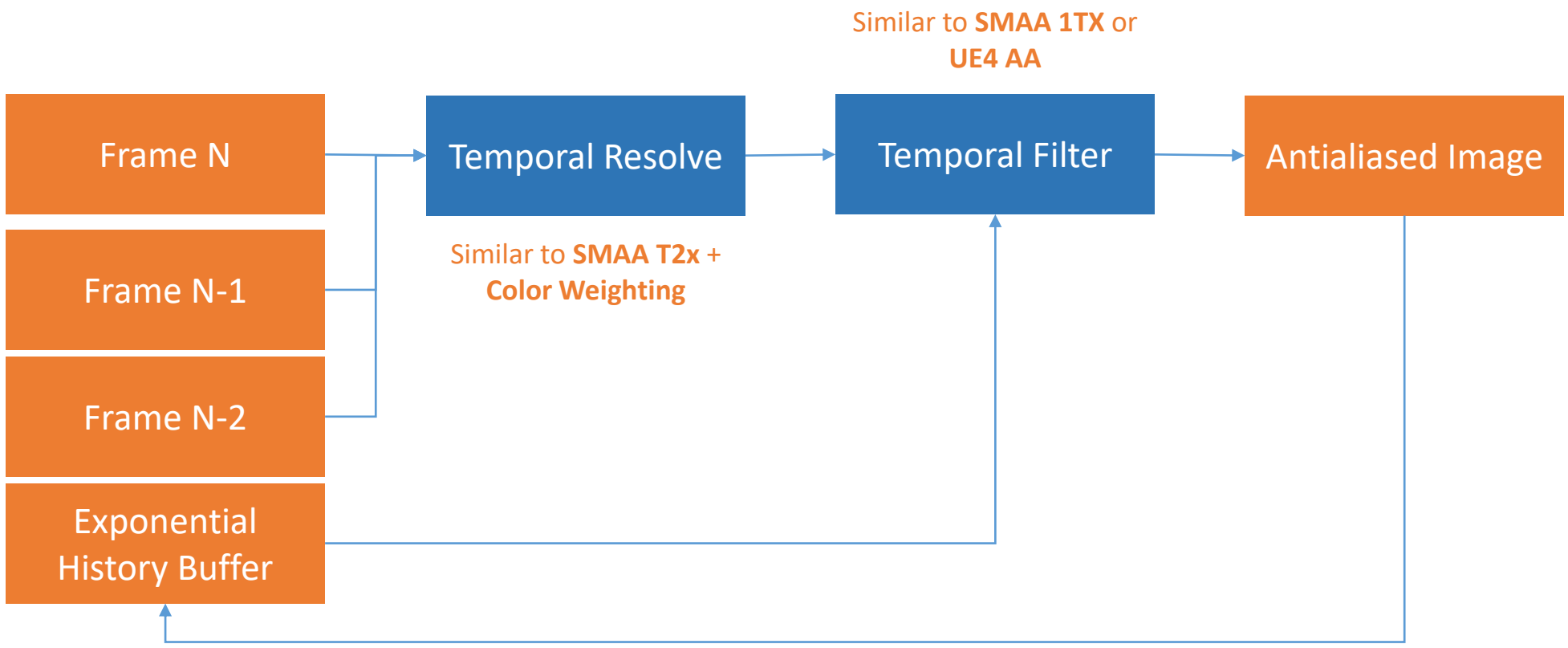


HRAA [Drobot2014]

- For temporal, hybrid approach between SMAA T2x and Unreal Engine 4 AA
- CRAA or SMAA Morphological for edges
- 2x subpixel jittering
  - Using flipquads for 4x effective rate
- Disocclusions:
  - Velocity weighting
  - Introduces color flow (weighting)
- Exponential history **after** the temporal 2x resolve
- Advanced neighborhood clamping
  - Soft clamp, only accepts colors near min and max
- Higher order resampling for sharper results



HRAA [Drobot2014]



Key Takeaway:
Temporal Filter is Decoupled from AA

- Using an exponential history is a process that can be decoupled from supersampling with subpixel jitters
  - Will be called temporal filter from now on
- It has the effect to temporally filter (or blur) an image
- We can think that as a side effect it can blur or average temporal subpixel jittering
- Even when not jittering, objects in motion naturally fall in different subsample positions each frame
  - Objects in motion will have AA even if no jitter is used

Filmic SMAA T2x

SMAA T2x

- **Subsamples:** 2x
- **Edges:** Morphological
- **Temporal filter:** No

- Ghosting on objects without velocities
- Stable in static images

Filmic SMAA T2x

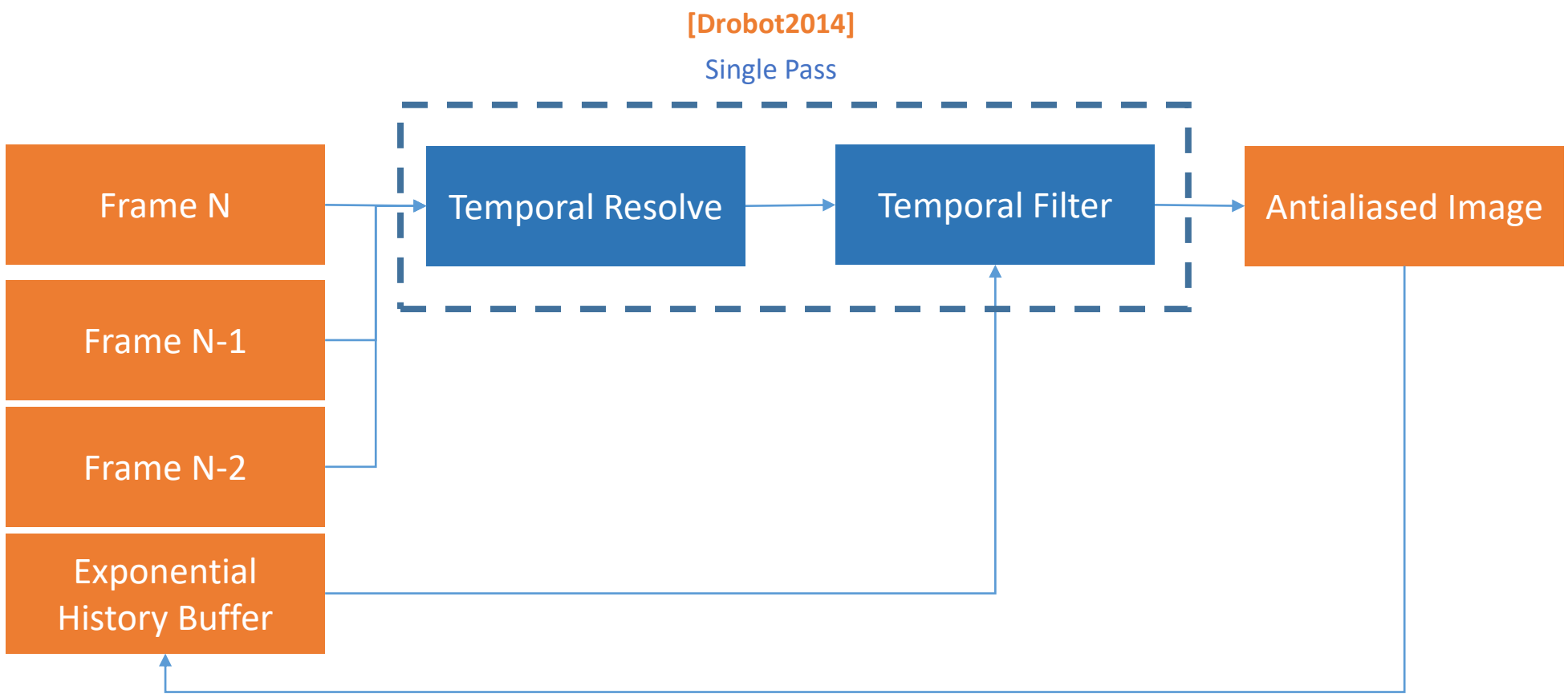
- **Subsamples:** 2x (Optional Contrast-Aware Quincunx for ~4x)
- **Edges:** Morphological
- **Temporal filter:** Yes

- Reduced ghosting
- Tight performance budget
- Sharp
- Stable in static images

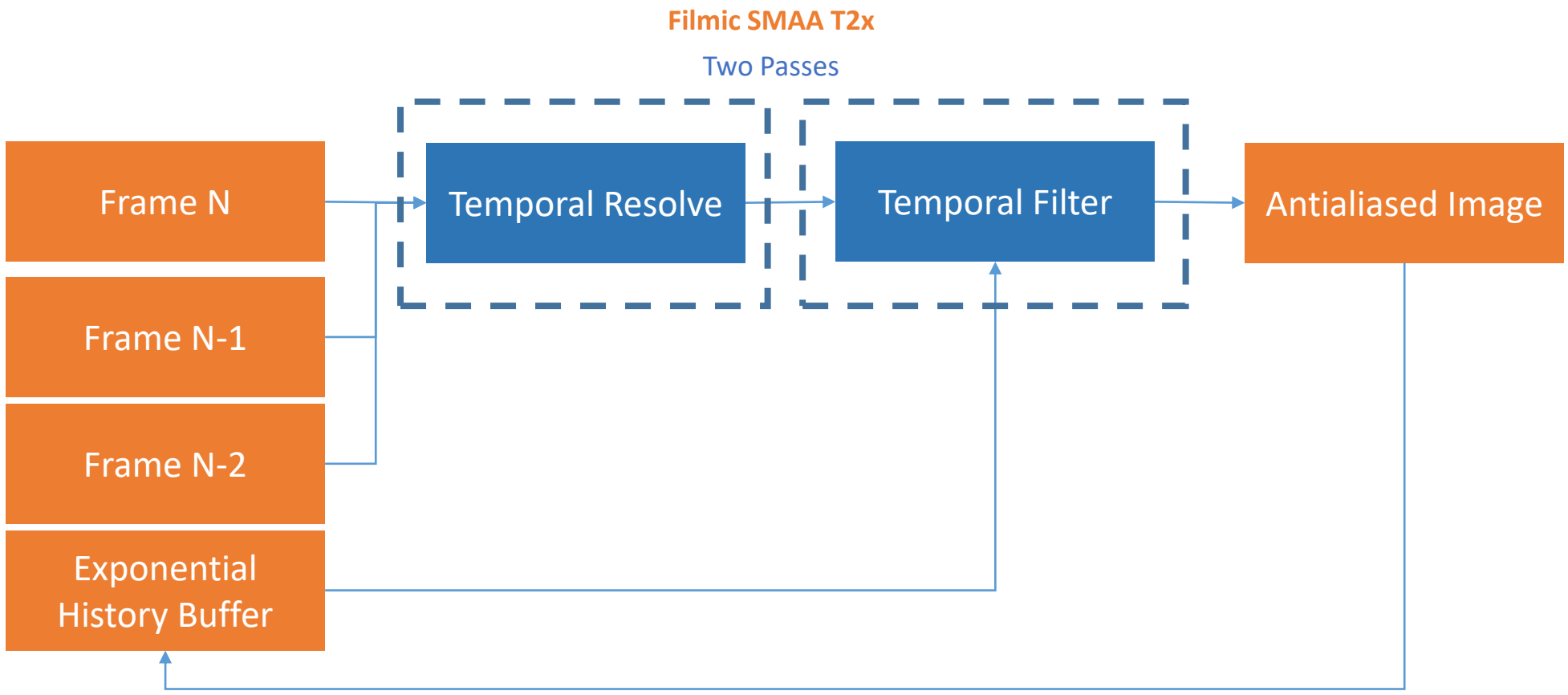
Temporal Antialiasing Improvements

- Separate temporal supersampling and temporal filtering
- Temporal filtering sharpness
- Temporal spatial contrast Tracking
- FPS-aware temporal filtering
- Extending neighborhood clamping with depth testing
- Improved temporal supersampling resampling
- Improved temporal Quincunx
- Supersampling derivatives
- Improved color weighting
- Low-profile velocity buffers
- Faster closest velocity
- Temporal upsampling

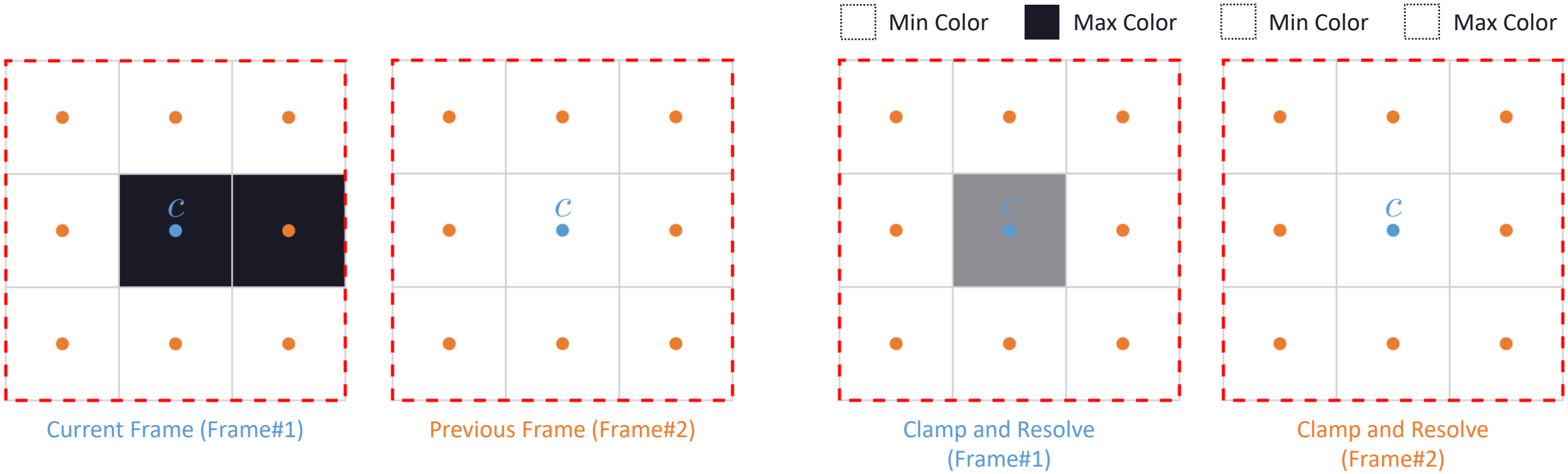
Separate Temporal Supersampling and Temporal Filtering

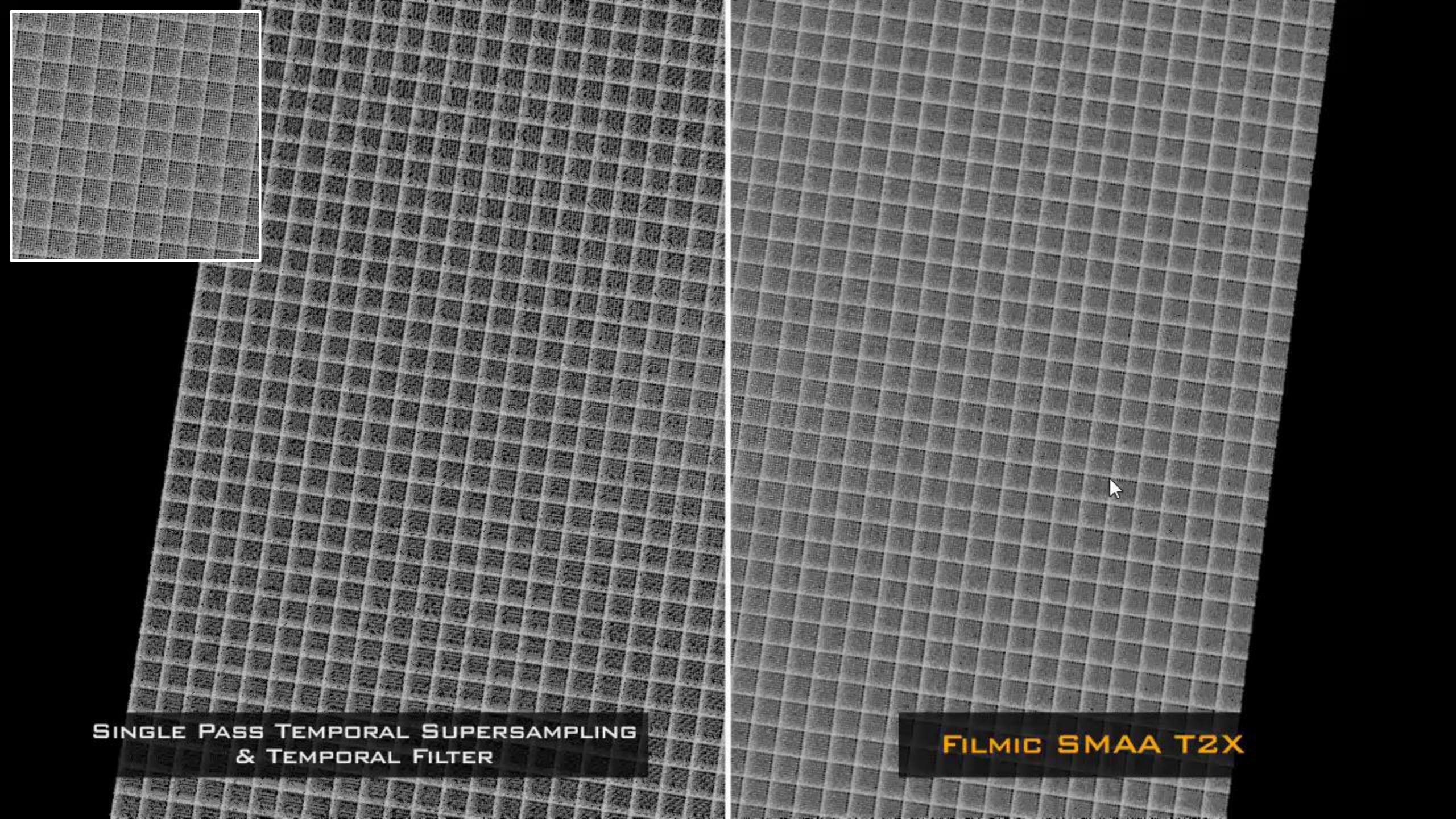


Separate Temporal Supersampling and Temporal Filtering



Temporal Antialiasing Toolset:
Neighborhood Clamp




Single Pass Temporal Supersampling
& Temporal Filter
Filmic SMAA T2X

Temporal Filtering Sharpness
Temporal Contrast

- SMAA 1TX [Sousa2013] lowers α in low spatial contrast areas
- Inspired by this, we instead lower α in low temporal contrast areas
- No filtering is performed on *temporal edges* below a threshold
  - Won't cause subpixel jitter/flicker because input already stabilized before temporal filtering
  - *Resets* the accumulation buffer when current color is similar (prevents numerical diffusion)
  - Similar in spirit to [Drobot2014] acceptance metric
- *Temporal edges* added to a deferred queue
  - Similar to morphological

$$h_{i+1} = \alpha c_i + (1 - \alpha)\, h_{i-1}$$

$$c_{\max} = \max(c_{TL}, c_{TR}, c_{BL}, c_{BR})$$
$$c_{\min} = \min(c_{TL}, c_{TR}, c_{BL}, c_{BR})$$
$$c_{acc} = clamp(c_{acc}, c_{\min}, c_{\max})$$
$$w_k = \left|(c_{TL} + c_{TR} + c_{BL} + c_{BR}) * 0.25 - c_M\right|$$
$$w_{rgb} = clamp\left(\frac{1}{K_{lowfreq} * (1 - w_k) + K_{hifreq} * w_k}, 0, 1\right)$$
$$c = c_M * (1 - w_{rgb}) + c_{acc} * w_{rgb}$$

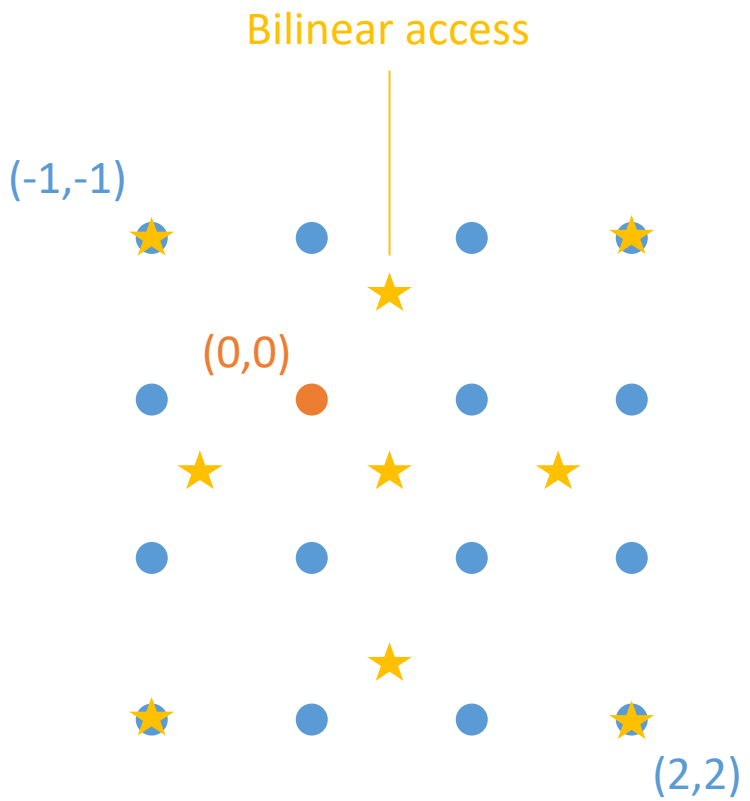
[Sousa2013] Spatial-Contrast Exponential History

Temporal Filtering Sharpness
Bicubic Resampling

- Bicubic filtering reduces numerical diffusion error on the exponential history
  - Catmull-Rom yields the best results
  - Optimized version uses 9 samples
  - Thanks to Bart Wronski for sharing the idea
- We only apply Catmull-Rom on temporal edges
  - When consuming the deferred queue
  - Still expensive
- Higher order resampling is an alternative option [Drobot2014]

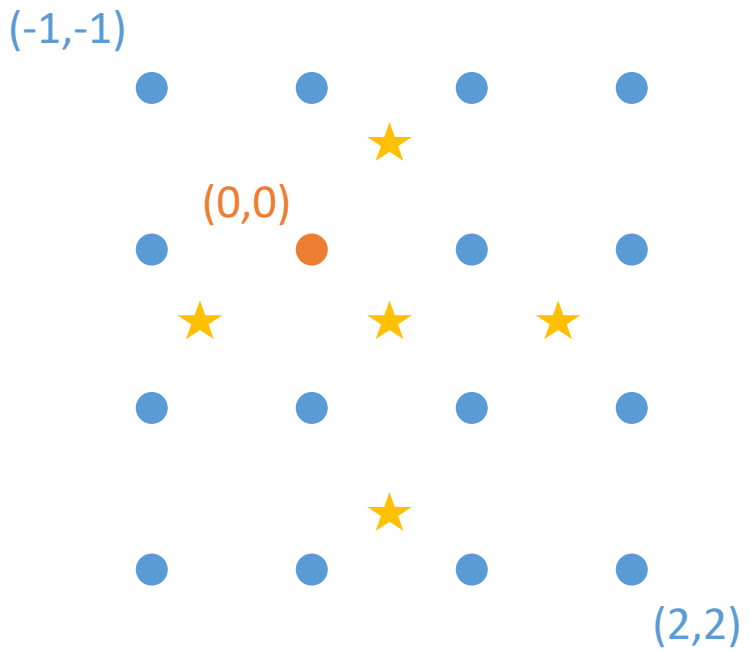
Temporal Filtering Sharpness
Bicubic Resampling

- Optimized Catmull-Rom uses 9 bilinear samples to process the 4x4 area
  - http://vec3.ca/bicubic-filtering-in-fewer-taps/
  - http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter20.html



Temporal Filtering Sharpness
Bicubic Resampling

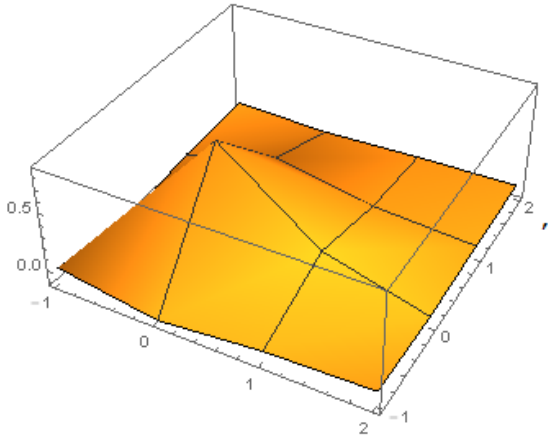
- Ignoring the 4 corners yields very similar results
- Reduces from 9 to 5 taps



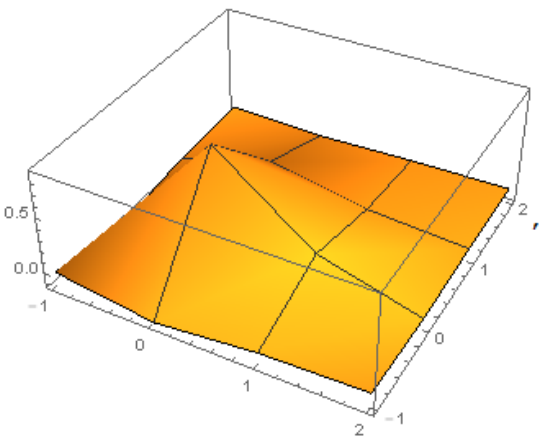
Temporal Filtering Sharpness
Bicubic Resampling

- Small numerical error
- Possible applications in other areas

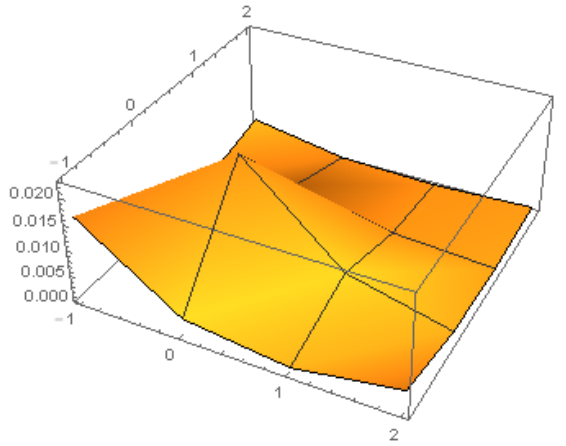
$$\left\{\begin{pmatrix} 0.016384 & -0.118784 & -0.029696 & 0.004096 \\ -0.118784 & 0.861184 & 0.215296 & -0.029696 \\ -0.029696 & 0.215296 & 0.053824 & -0.007424 \\ 0.004096 & -0.029696 & -0.007424 & 0.001024 \end{pmatrix}, \begin{pmatrix} 0. & -0.121905 & -0.0304762 & 0. \\ -0.121905 & 0.88381 & 0.220952 & -0.0304762 \\ -0.0304762 & 0.220952 & 0.0552381 & -0.00761905 \\ 0. & -0.0304762 & -0.00761905 & 0. \end{pmatrix}, \begin{pmatrix} 0.016384 & 0.00312076 & 0.00078019 & 0.004096 \\ 0.00312076 & 0.0226255 & 0.00565638 & 0.00078019 \\ 0.00078019 & 0.00565638 & 0.0014141 & 0.000195048 \\ 0.004096 & 0.00078019 & 0.000195048 & 0.001024 \end{pmatrix},$$



9 Samples



5 Samples



Error

$$\}$$

Shader Code

```
float3 SMAAFilterHistory(SMAATexture2D colorTex, float2 texcoord, float4 rtMetrics)
{
    float2 position = rtMetrics.zw * texcoord;
    float2 centerPosition = floor(position - 0.5) + 0.5;
    float2 f = position - centerPosition;
    float2 f2 = f * f;
    float2 f3 = f * f2;

    float c = SMAA_FILMIC_REPROJECTION_SHARPNESS / 100.0;
    float2 w0 =        -c  * f3 +  2.0 * c         * f2 - c * f;
    float2 w1 =  (2.0 - c) * f3 - (3.0 - c)        * f2         + 1.0;
    float2 w2 = -(2.0 - c) * f3 + (3.0 -  2.0 * c) * f2 + c * f;
    float2 w3 =         c  * f3 -                c * f2;

    float2 w12 = w1 + w2;
    float2 tc12 = rtMetrics.xy * (centerPosition + w2 / w12);
    float3 centerColor = SMAASample(colorTex, float2(tc12.x, tc12.y)).rgb;

    float2 tc0 = rtMetrics.xy * (centerPosition - 1.0);
    float2 tc3 = rtMetrics.xy * (centerPosition + 2.0);
    float4 color = float4(SMAASample(colorTex, float2(tc12.x, tc0.y )).rgb, 1.0) * (w12.x * w0.y ) +
                   float4(SMAASample(colorTex, float2(tc0.x,  tc12.y)).rgb, 1.0) * (w0.x  * w12.y) +
                   float4(centerColor,                                      1.0) * (w12.x * w12.y) +
                   float4(SMAASample(colorTex, float2(tc3.x,  tc12.y)).rgb, 1.0) * (w3.x  * w12.y) +
                   float4(SMAASample(colorTex, float2(tc12.x, tc3.y )).rgb, 1.0) * (w12.x * w3.y );
    return color.rgb * rcp(color.a);
}
```

Extending Neighborhood Clamping with Depth Testing

- Neighborhood clamping do not remove all ghosting
- High frequency changes on the neighborhood window perform the clamp differently
  - Ghosting persists
  - Usually only seen in motion

Extending Neighborhood Clamping with Depth Testing

- Performing a low pass on the neighborhood window helps
  - Blurring neighbors with bilinear filtering
  - Introduces back aliasing

```
neighborhood[0] = SMAASample(colorTex, mad(SMAA_RT_METRICS.xy, 0.6 * float2(-1.0, -1.0), texcoord)).rgb;
neighborhood[1] = SMAASample(colorTex, mad(SMAA_RT_METRICS.xy, 0.6 * float2(-1.0,  1.0), texcoord)).rgb;
neighborhood[2] = SMAASample(colorTex, mad(SMAA_RT_METRICS.xy, 0.6 * float2( 1.0, -1.0), texcoord)).rgb;
neighborhood[3] = SMAASample(colorTex, mad(SMAA_RT_METRICS.xy, 0.6 * float2( 1.0,  1.0), texcoord)).rgb;
```

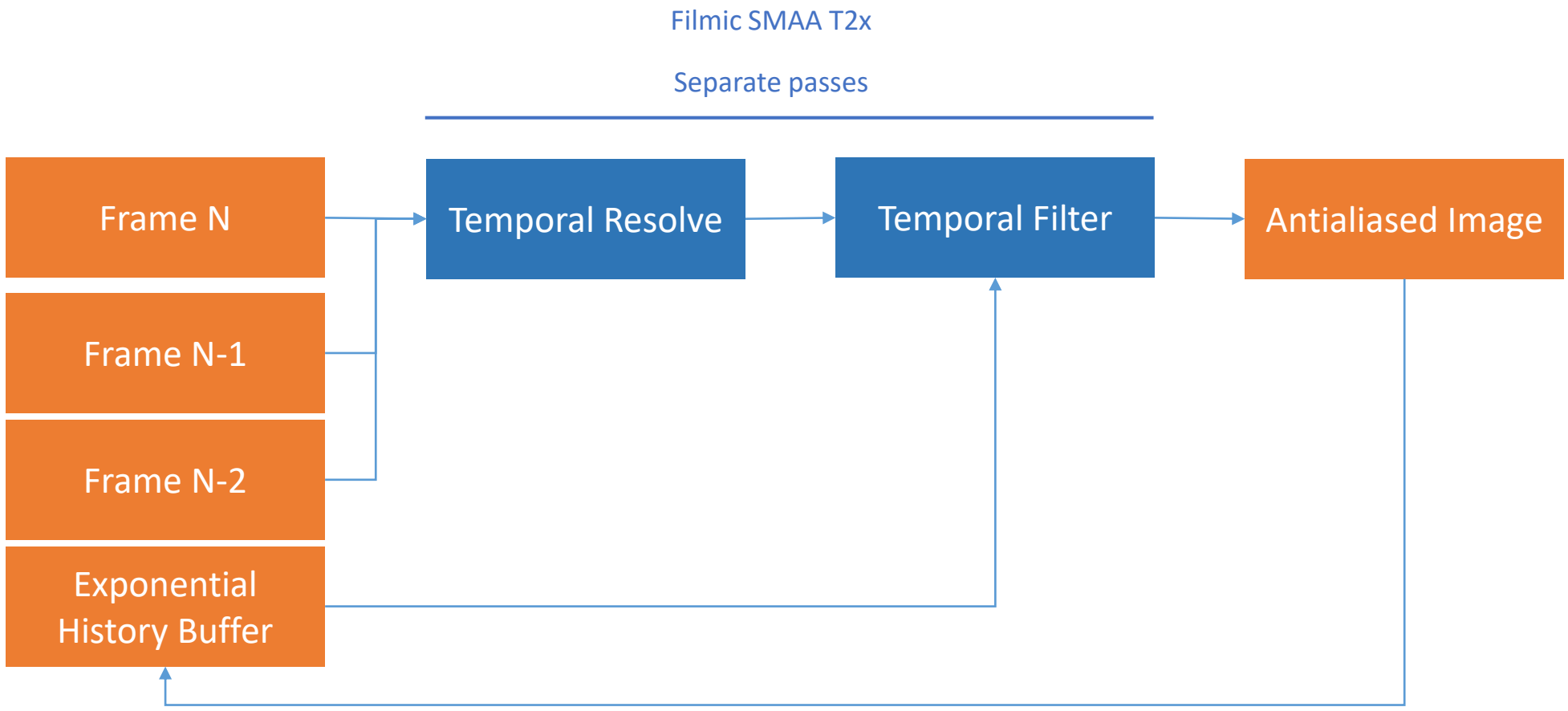
- [Salvi2016] proposed using variance for a tighter clamp/clip
  - In our technique/testbed [Salvi2016] + clipping introduced back some aliasing

- Using variance or a low pass on the neighborhood reduce the min/max window
  - Less ghosting but potentially more aliasing

- To keep maximum temporal smoothing effect, we use depth testing to detect disocclusions
  - Compare current and previous frame depth
  - Half resolution nearest depth

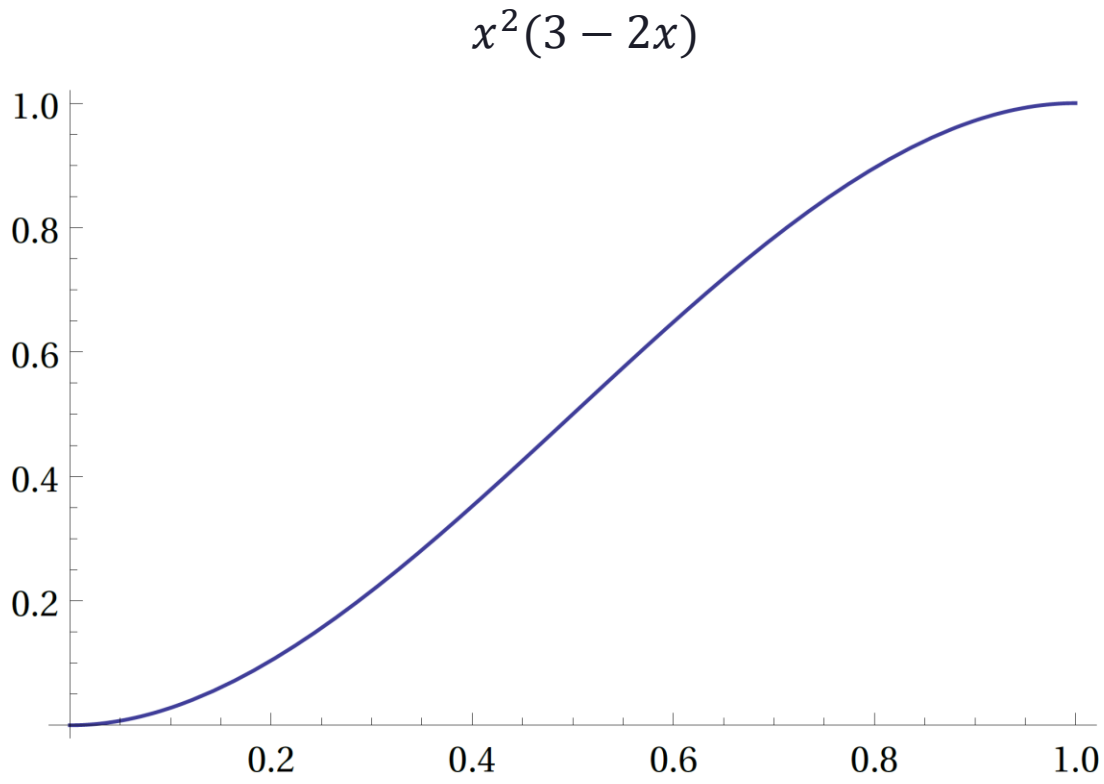
- “Responsive AA” for alpha blended objects [Karis2014]
  - With depth testing and “Responsive AA” we mitigate all cases without harming the temporal filtering effect on opaque objects

Improved Temporal Supersampling Resampling

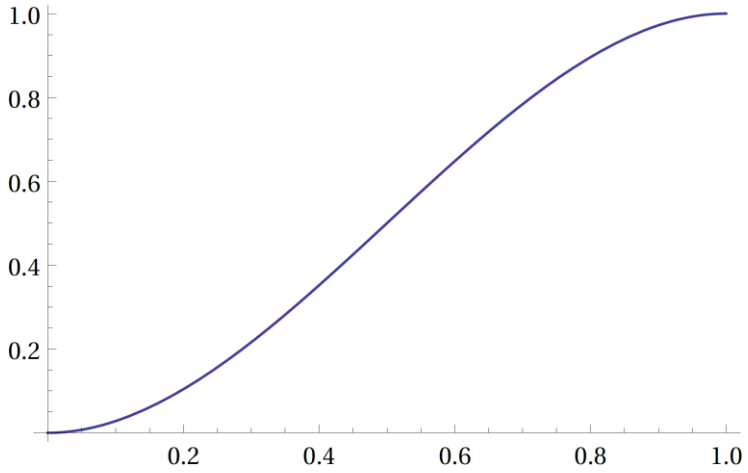


Improved Temporal Supersampling Resampling

- Temporal supersampling also resamples
  - Bicubic filtering nice, but overkill (no numerical diffusion on supersampling: not recursive)
- Iñigo Quilez improved texture interpolation
  - Change fractional part of `texcoord` to `smoothstep`
  - http://www.iquilezles.org/www/articles/texture/texture.htm
- Snaps `texcoord` to centers
- Thanks to Xianchun Wu for the idea

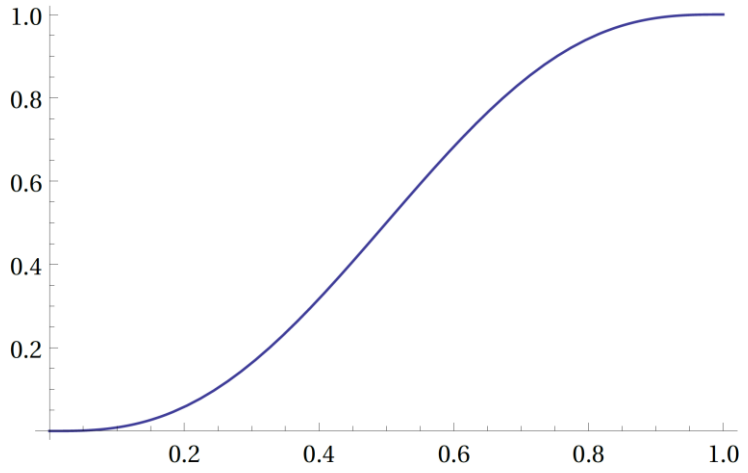


Improved Temporal Supersampling Resampling



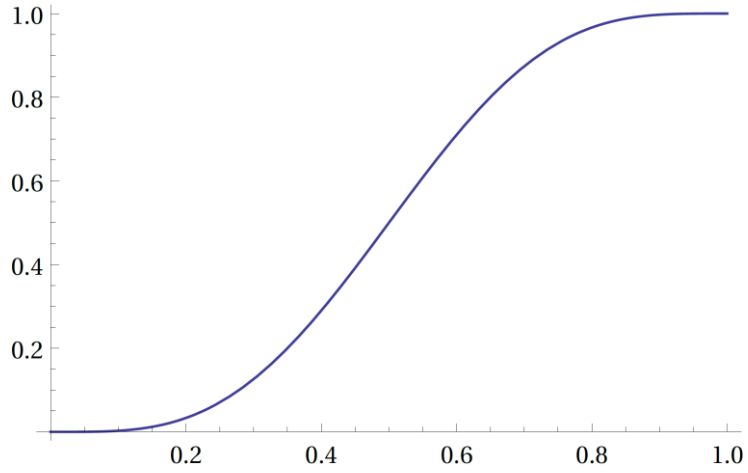
Smoothstep

$x^2(3-2x)$



Smootherstep

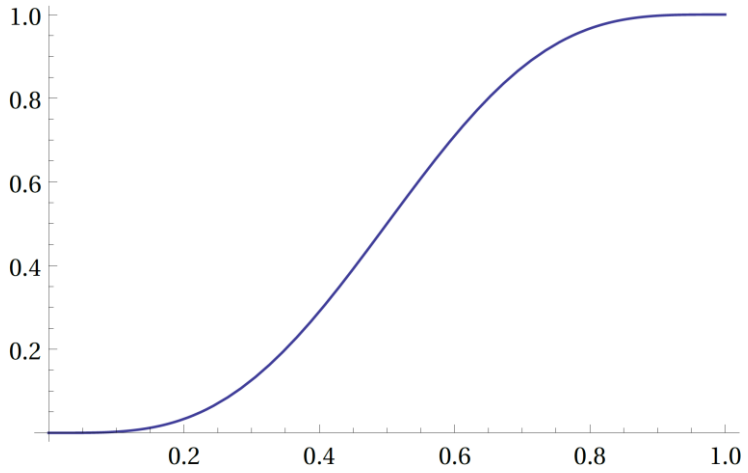
$x^3(x(6x-15)+10)$



Smoothereststep

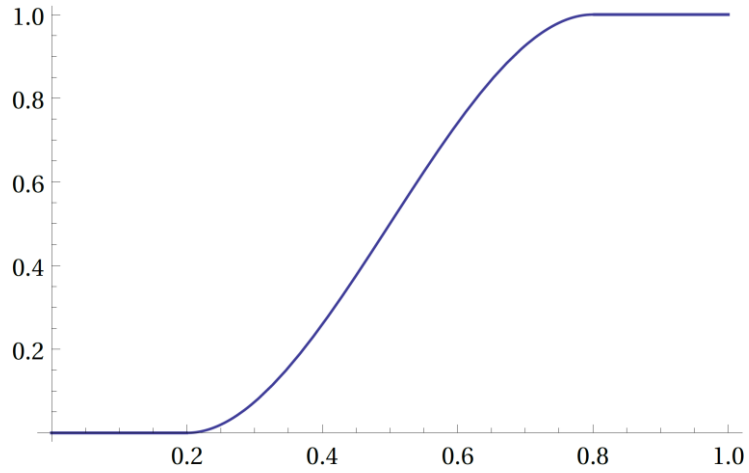
$-20x^7+70x^6-84x^5+35x^4$

Improved Temporal Supersampling Resampling



Smootheststep

$$-20x^7 + 70x^6 - 84x^5 + 35x^4$$



Fast Smootheststep

$$u = \max(\min((x - 0.2)/(1 - 0.4),1),0)$$
$$u^2(3 - 2u)$$

Improved Temporal Supersampling Resampling

Current Frame

Improved Temporal Supersampling Resampling

Previous Frame

Resampled with bilinear

Improved Temporal Supersampling Resampling

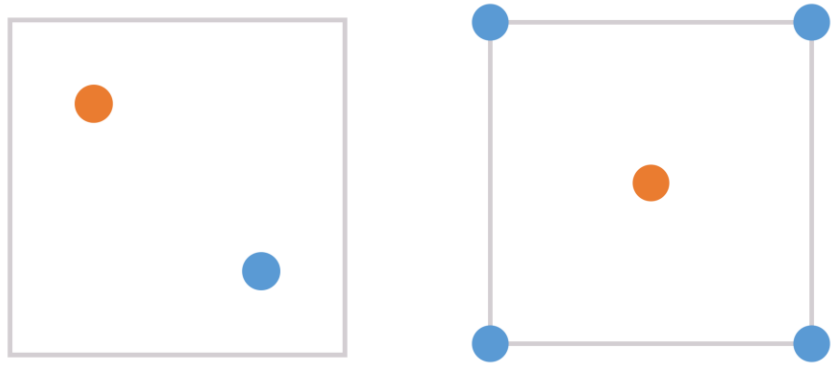
Previous Frame

Resampled with fast smoothststep

Improved Temporal Quincunx

Basic Quincunx

- Better subsample coverage ~4x
- Fast *0.5px wide resolve*
- Softens the texture details
  - Infamous in the previous gen
  - Not so bad at 1080p
- Used for temporal AA in Crysis 2 [Sousa2011]
- Flip quads a good alternative [Drobot2014]
  - Different stability, performance and sharpness tradeoff
  - We went for the less efficient but faster and sharper Quincunx



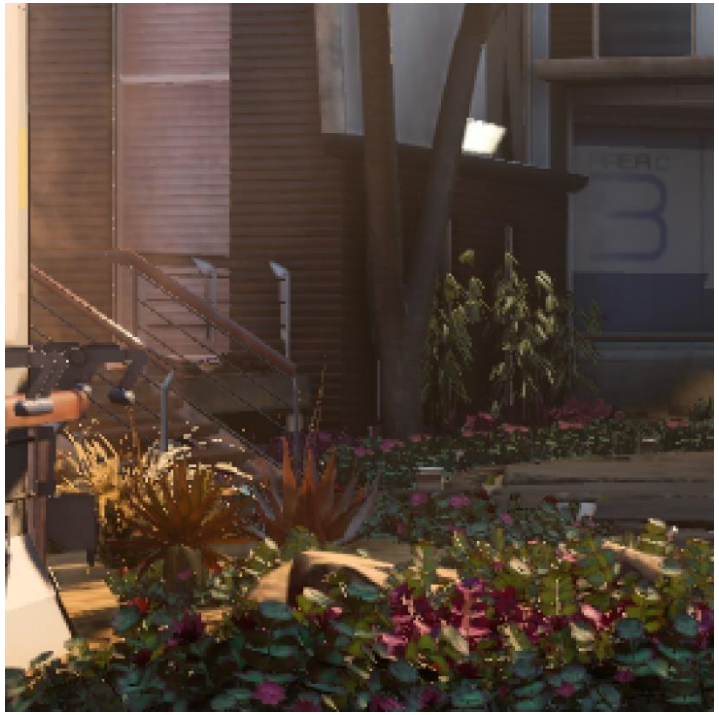
Rotated Grid 2x

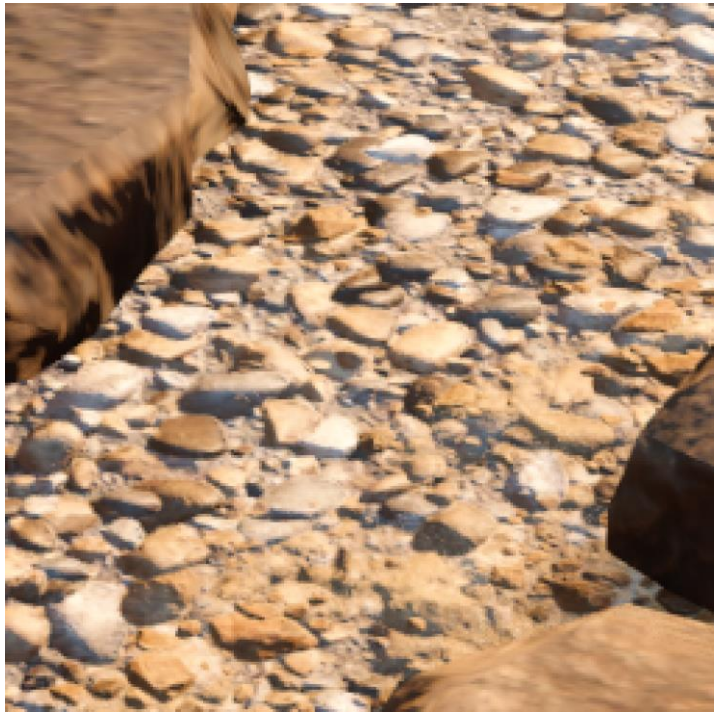
Quincunx

Improved Temporal Quincunx

Basic Quincunx

Rotated Grid 2x





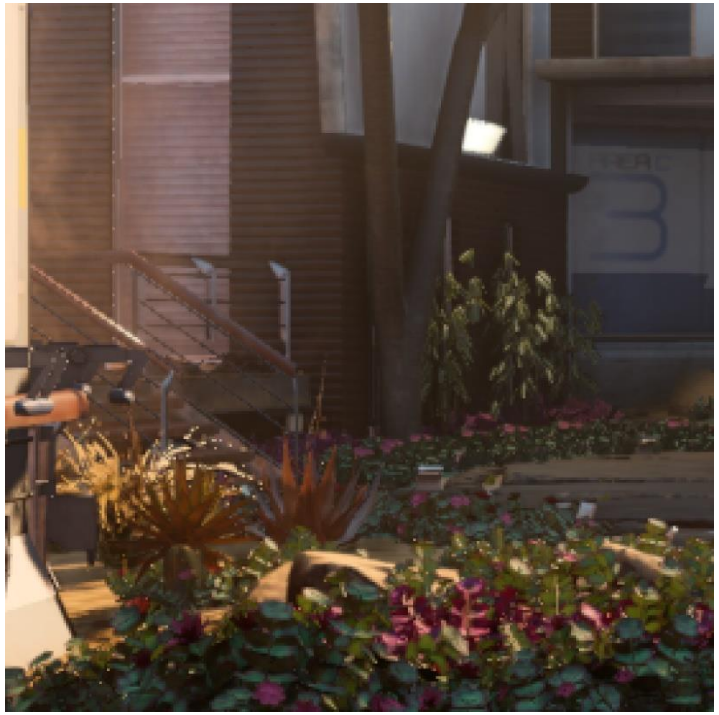
Better Subsample Coverage ☺

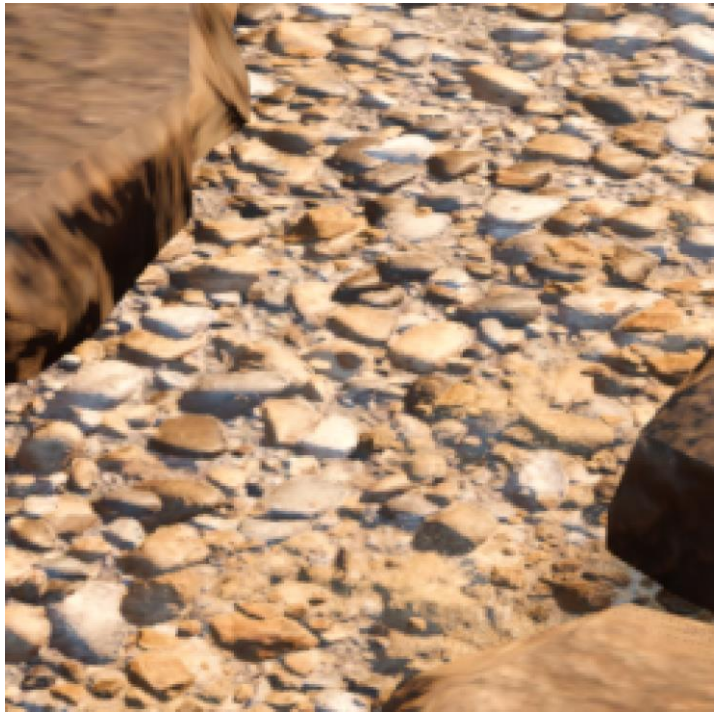
Softer Texture Details ☹

Improved Temporal Quincunx

Basic Quincunx

Quincunx





Better Subsample Coverage ☺

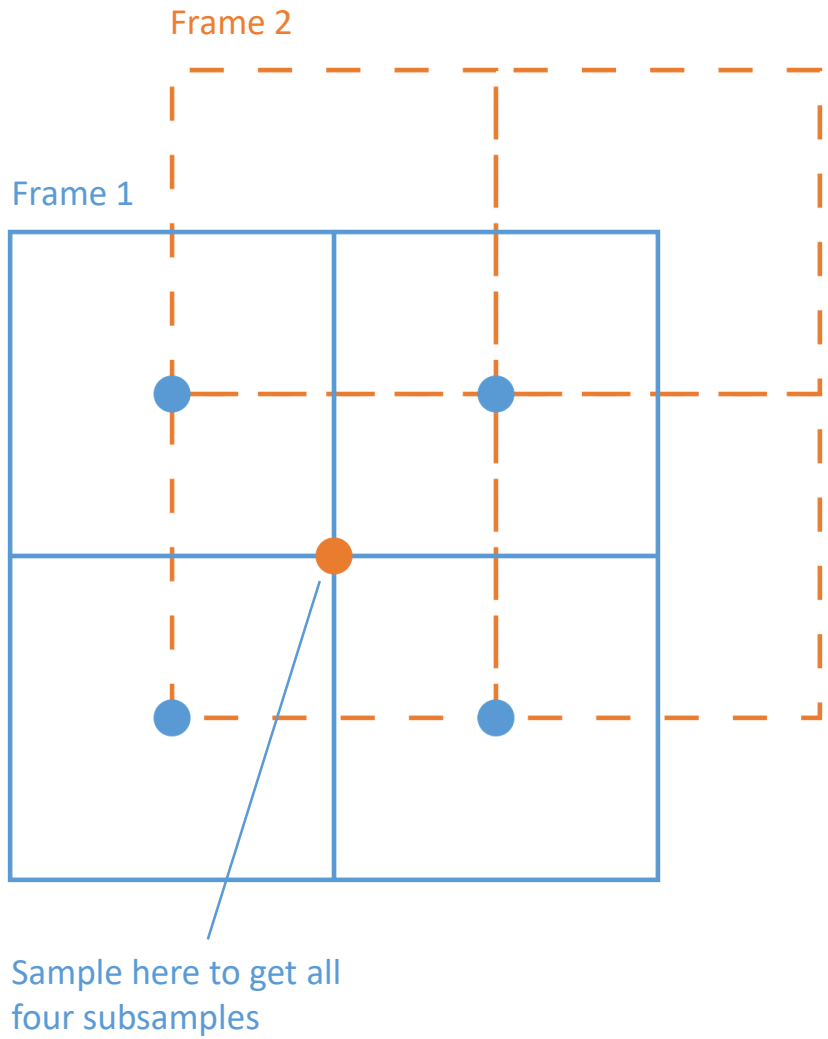
Softer Texture Details ☹

Improved Temporal Quincunx
Faster Quincunx

- Quincunx definition

$0.5 \times$ ● $+ \; 0.5 \times$ ●

- Orange is a single sample
- We can fetch all blue with bilinear
- Quincunx becomes:
  - Texcoord offset
  - Same performance as 2x



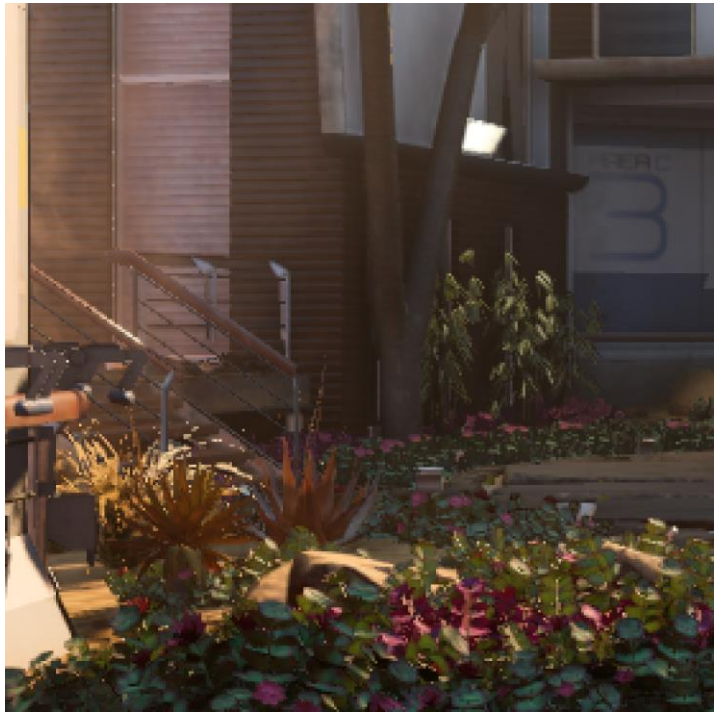
Improved Temporal Quincunx
Contrast-Aware Quincunx

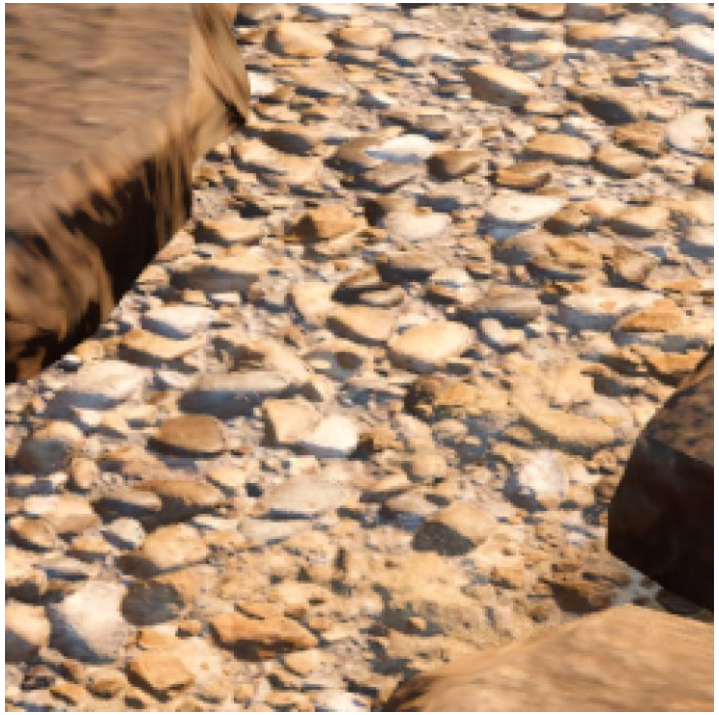
- **Idea:** lerp the texcoord offsets according to local contrast
  - Low contrast (texture details): 2x
  - High contrast (edges): Quincunx
- Use Quincunx subsamples to determine contrast
  - Refetch with 2x offsets if low contrast
  - Cheap: ~0.02ms on PS4@1080

Improved Temporal Quincunx

Contrast-Aware Quincunx

Rotated Grid 2x





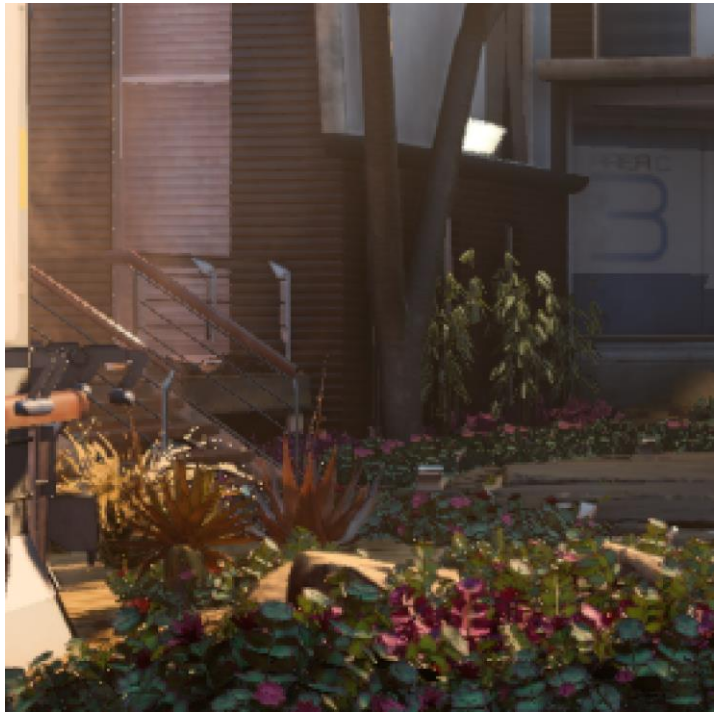
Better Subsample Coverage ☺

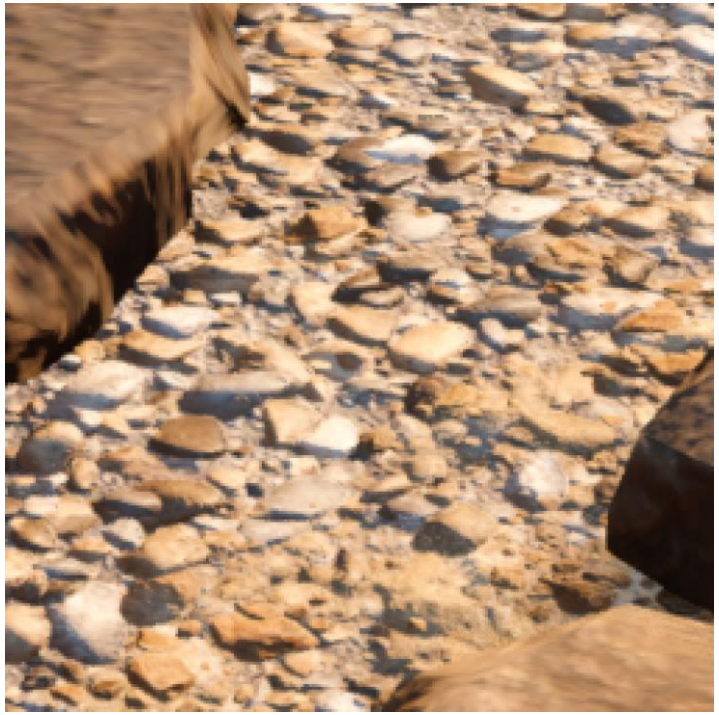
Sharp Texture Details ☺

Improved Temporal Quincunx

Contrast-Aware Quincunx

Contrast-Aware Quincunx



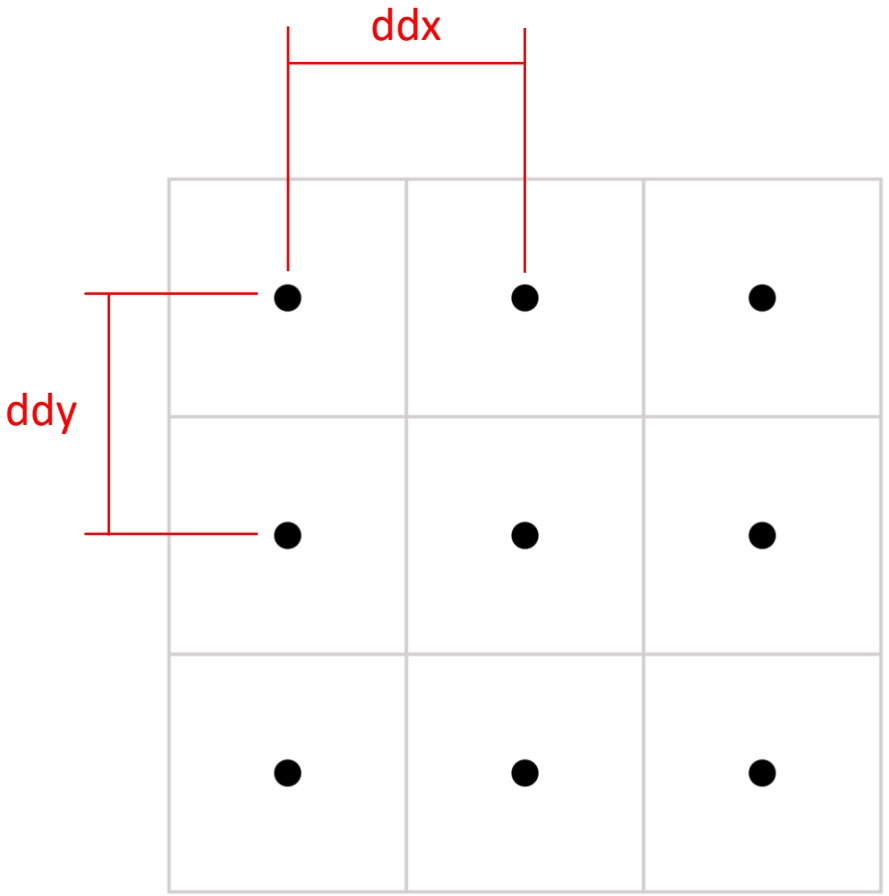


Better Subsample Coverage ☺

Sharp Texture Details ☺

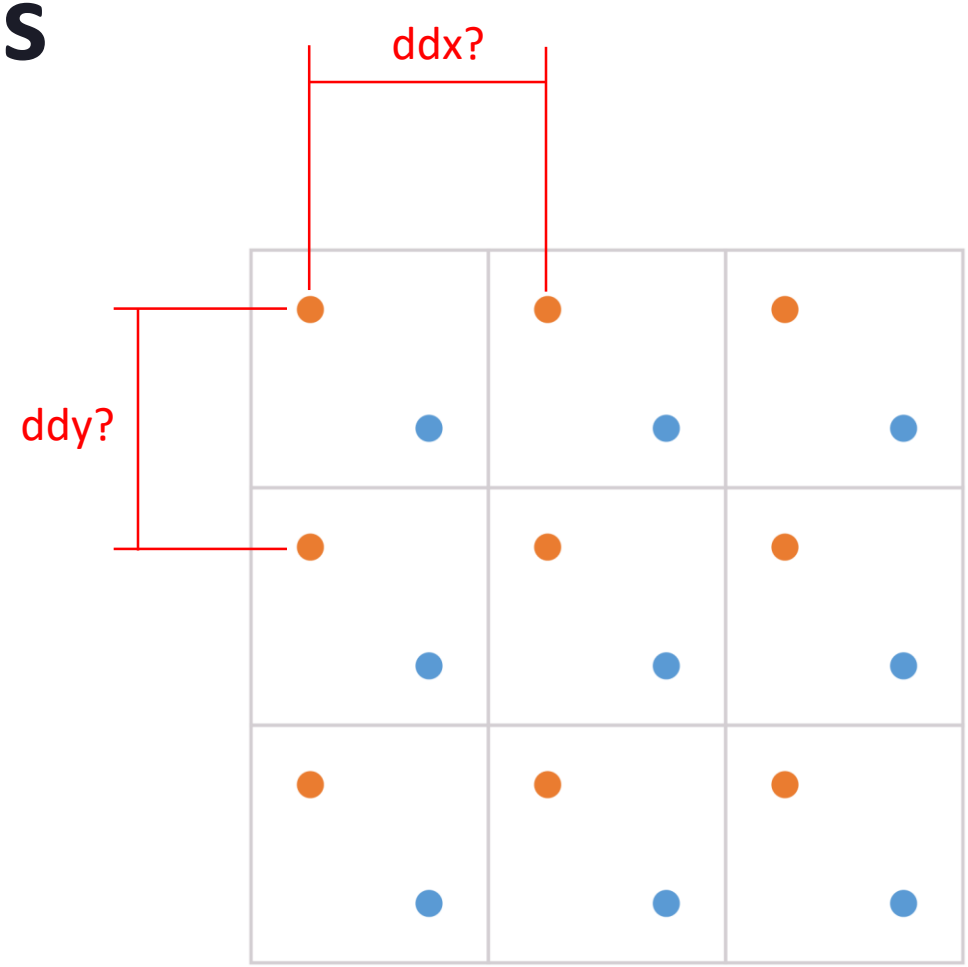
Supersampling Derivatives

- Derivatives calculated for 1x uniform grid



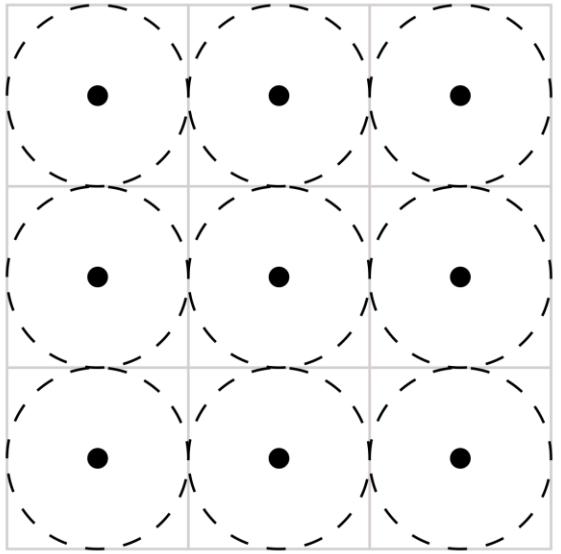
Supersampling Derivatives

- For 2x derivatives between subsamples smaller
  - Blurrier results
- Grid is rotated
  - What ddx/ddy to use?

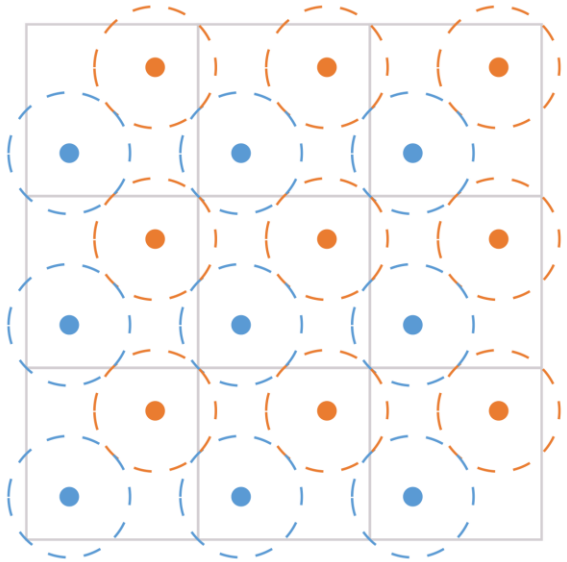


Supersampling Derivatives

- If we think of derivatives as filtering radius, we have a tradeoff:
  - Minimal overlap (blur): 0.7071
  - Minimal holes (aliasing): 1.0
- Adjust gradients by diagonal distance:
  - `SampleGrad(..., 0.7071 * ddx, 0.7071 * ddy)`
- Faster and identical results:
  - `SampleBias(..., log2(0.7071))`
  - Better: use texture sampler LOD bias
- Thanks to Peter-Pike Sloan and Angelo Pesce for the ideas
- **Sharper results**



Uniform Grid 1x

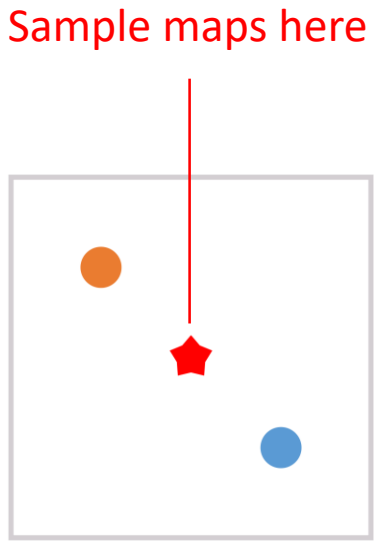


Rotated Grid 2x

Supersampling Derivatives

- Alternatively, unjitter the diffuse and normal map texture coordinates
  - Using the barycentric position of the subsamples (as in MSAA)
  - Loses shading antialiasing
  - A good tradeoff is to only do it for the diffuse
  - Free

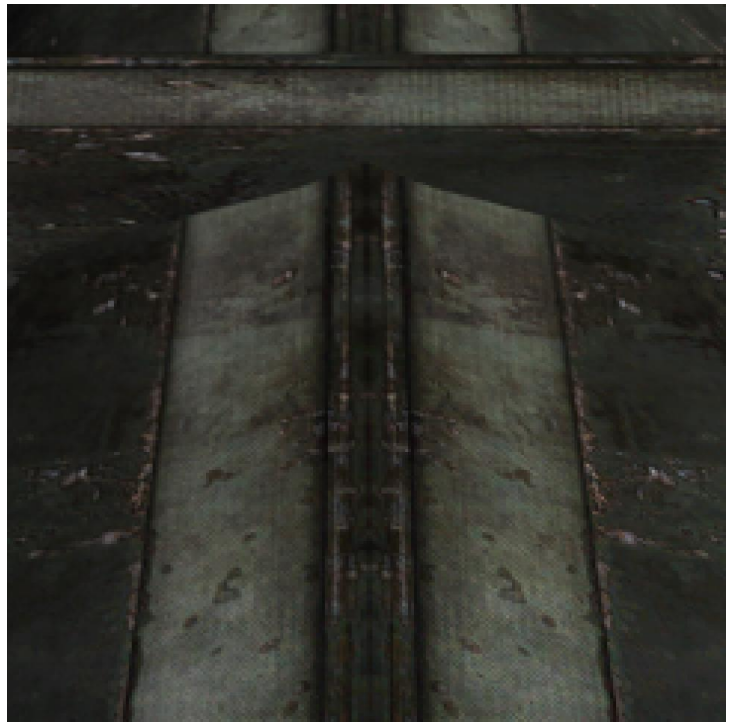
```
input.texcoord += ddx(input.texcoord) * subpixelOffset.x;
input.texcoord -= ddy(input.texcoord) * subpixelOffset.y;
```



Supersampling Derivatives

Uniform Grid 1x

Supersampling Derivatives



Rotated Grid 2x

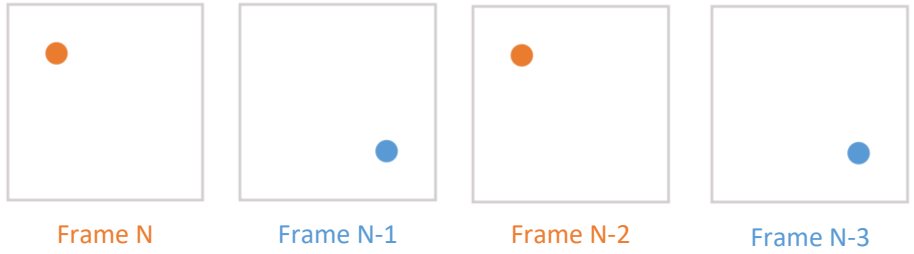
Supersampling derivatives

Rotated Grid 2x +
Barycentric Position

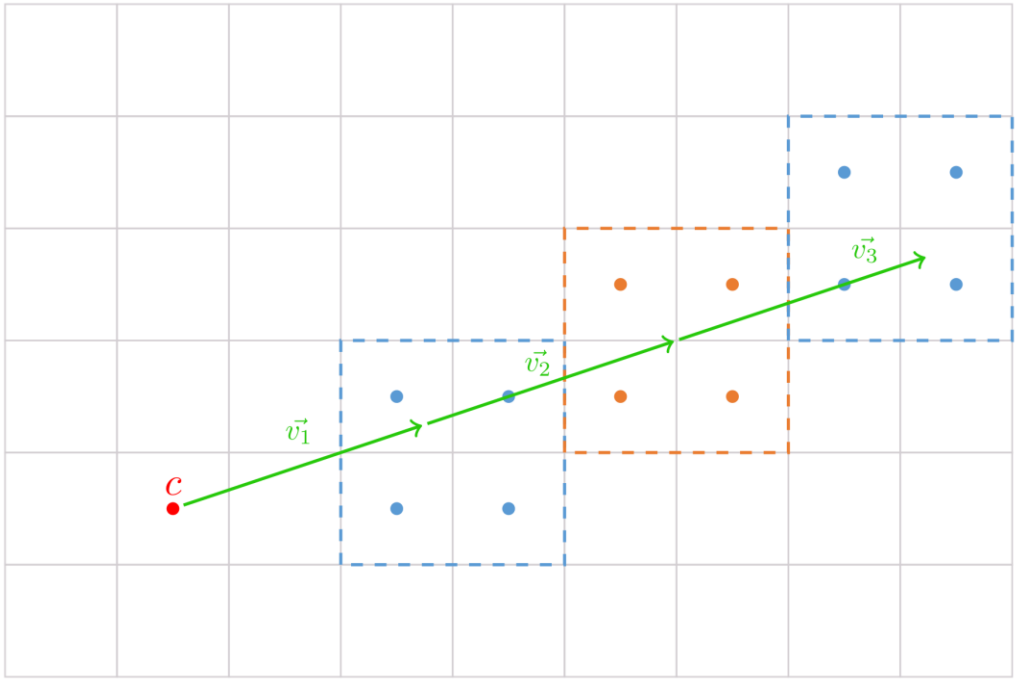
Supersampling Derivatives

Rotated Grid 2x +
SampleGrad

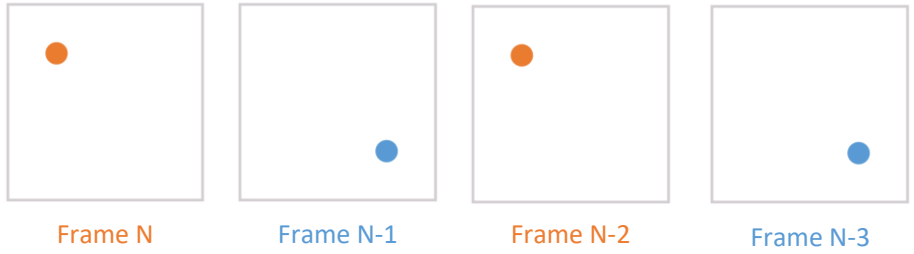
Improved Color Weighting
Faster



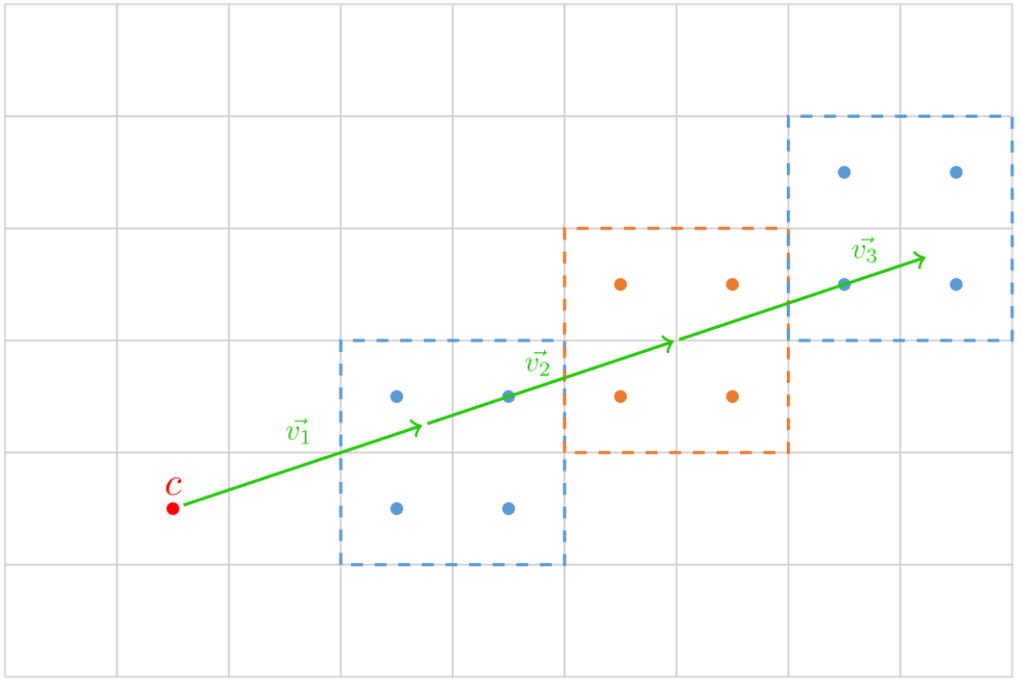
- **Remember:** used to reject previous frame subsamples for super sampling [Drobot2014]
- Sum of absolute differences (SAD):
  - Current frame 3x3 block
  - Previous frame 3x3 block
- We store luma for each frame
  - Compare 2x2 blocks using Gather
  - Reduces 18 samples to 2



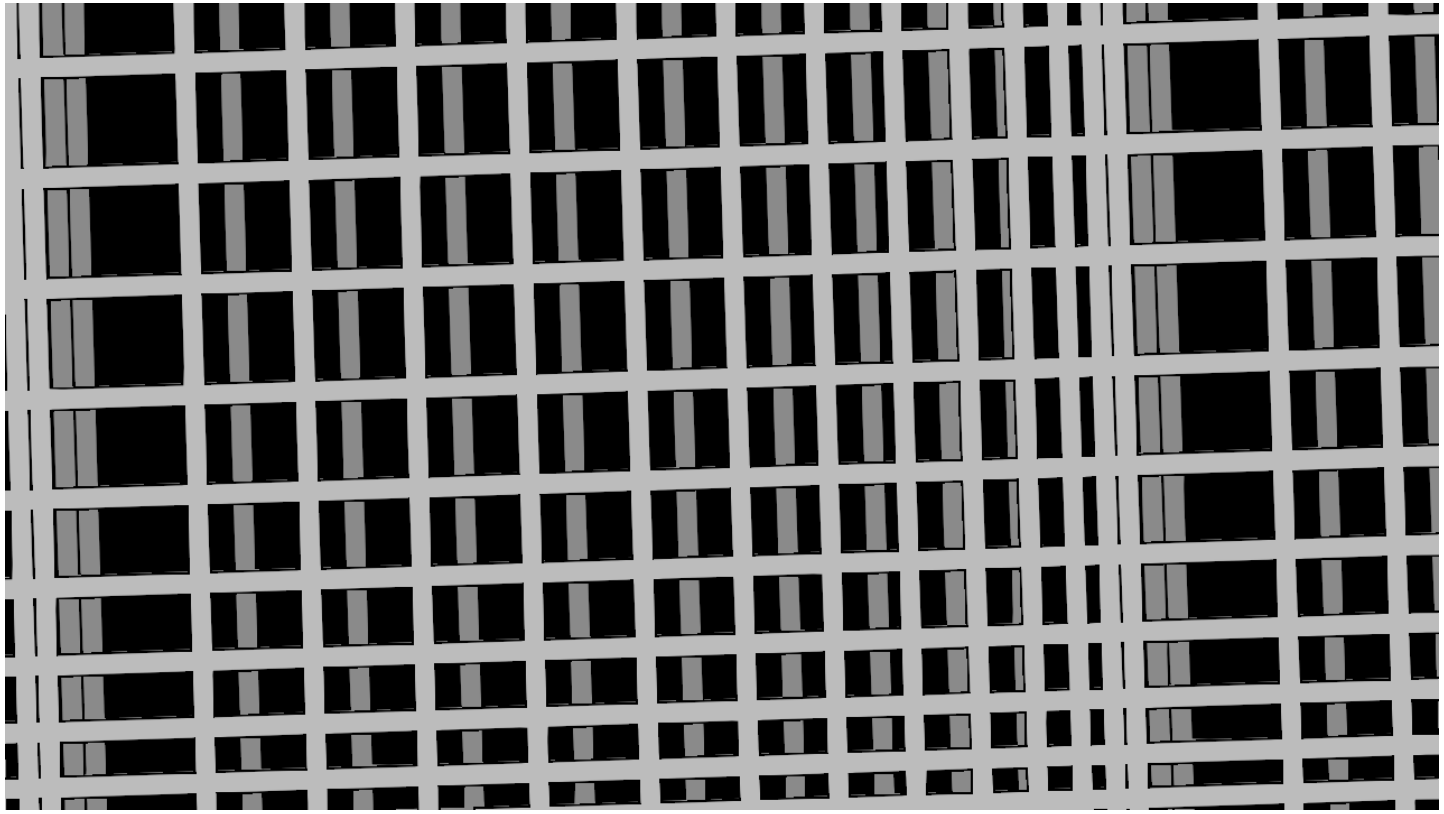
Improved Color Weighting
Extended



- We do two comparisons instead of one:
  - Frame N and N-2
  - Frame N-1 and N-3

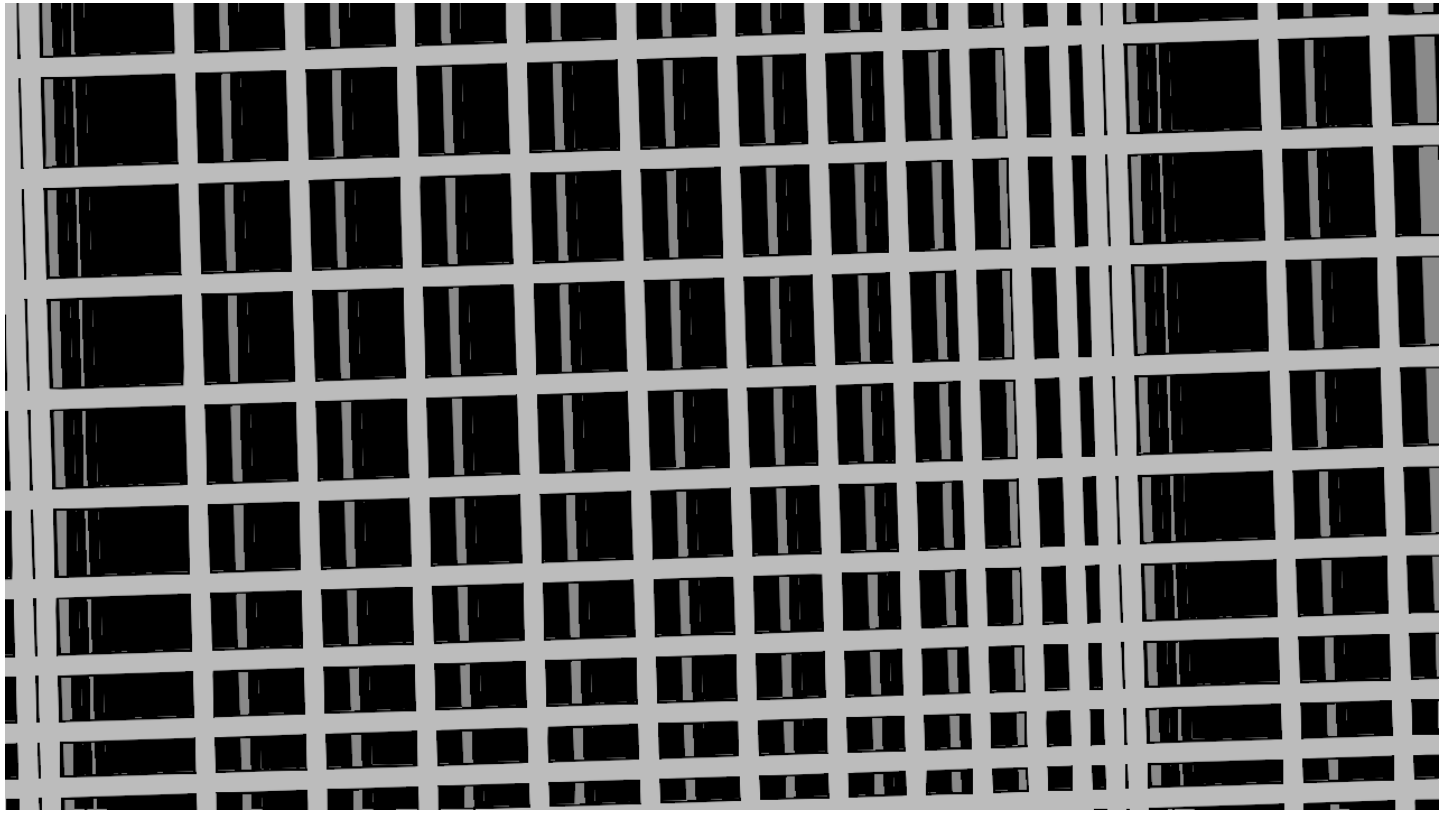


Improved Color Weighting
Extended



No color weighting

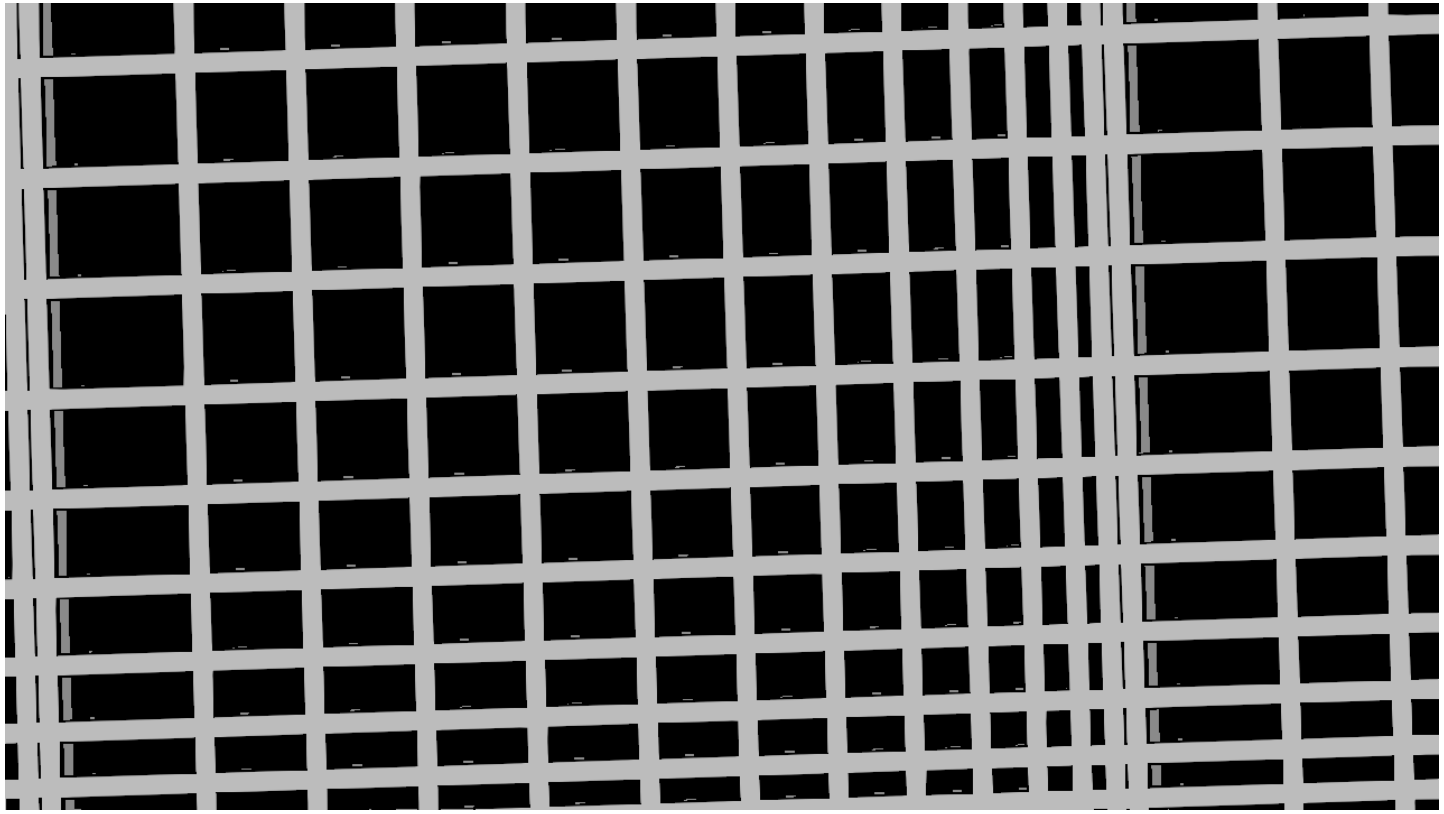
Improved Color Weighting
Extended



Color weighting

Frame N and N-2

Improved Color Weighting
Extended



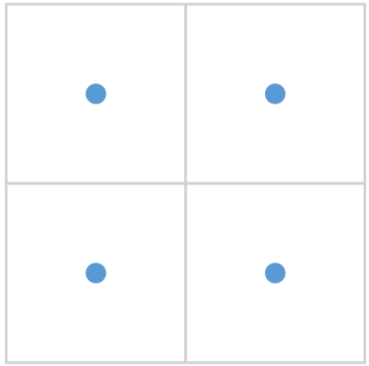
Color weighting

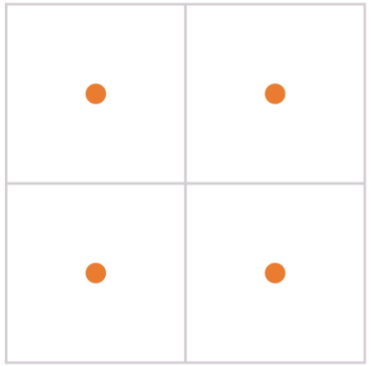
Frame N and N-2 and Frame N-1 and N-3

Improved Color Weighting
Improved Pixel Matching

- SAD works very well for translations
  - Not perfectly suited for rotations or perspective transforms

$$\sum abs(\quad - \quad)$$

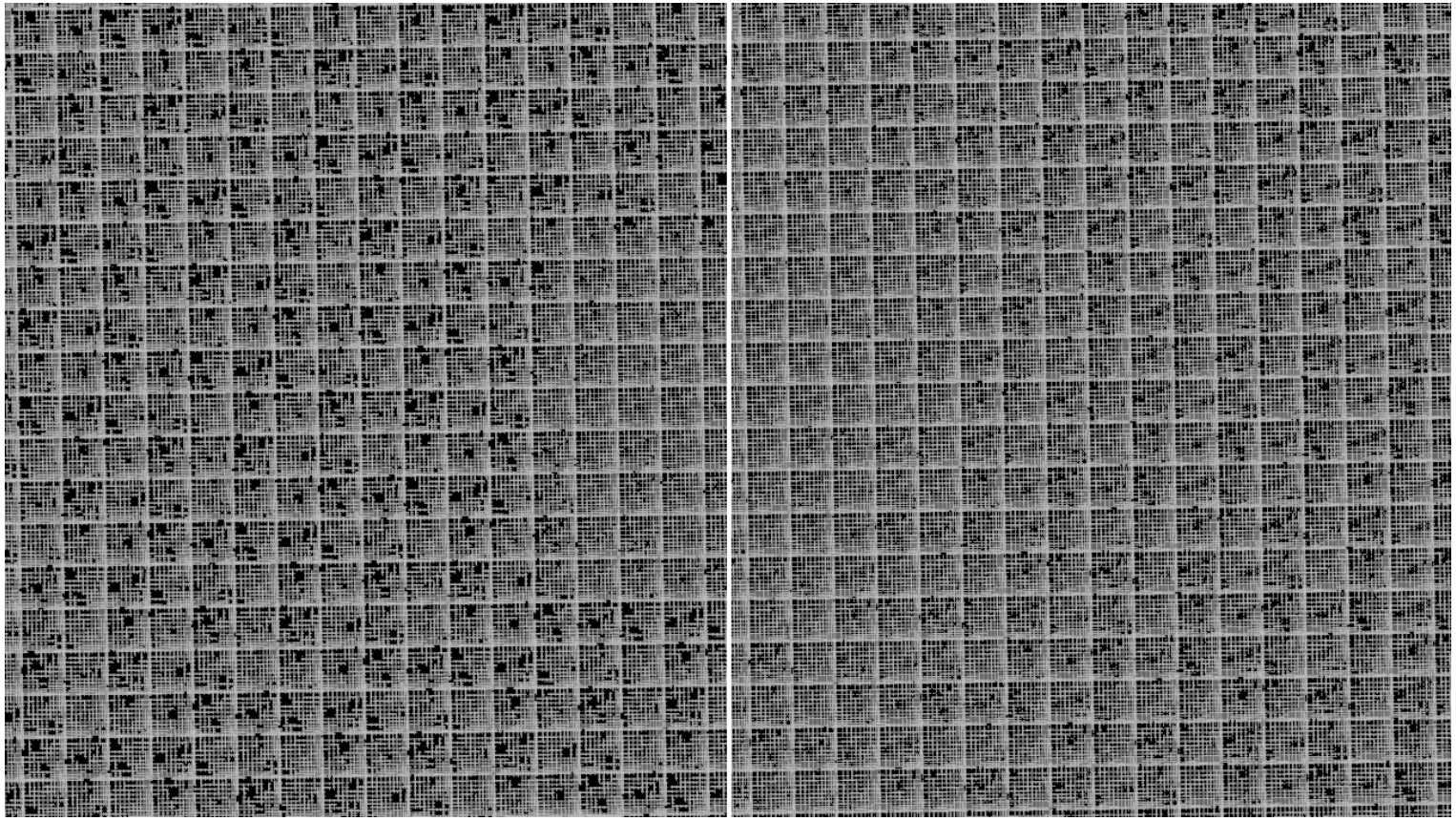




Improved Color Weighting
Improved Pixel Matching

- We instead search for the closest match for each pixel

Improved Color Weighting
Improved Pixel Matching



SAD

Closest Match

Low-Footprint Velocity Buffers

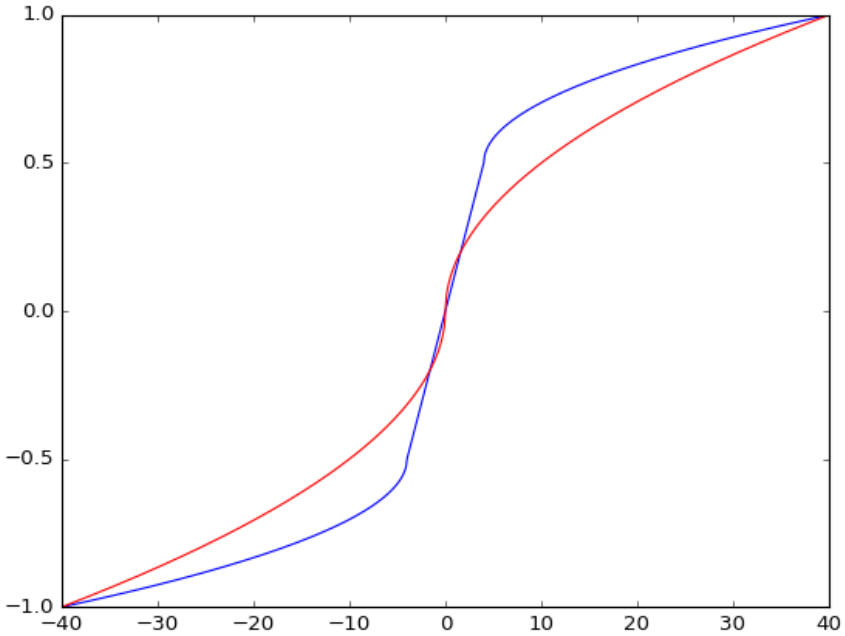
- Temporal supersampling:
  - Typically memory bound
  - Require a lot of buffers
- Velocity buffer
  - 7.91MB for 1080p@16 bit
- Reducing its memory footprint desirable

Low-Footprint Velocity Buffers

- 8-bit Gamma [Sousa2013]
- We opted for a two-piece linear function:
  - 0..40 pixels range
  - Not completely linear: smooth transition
  - Constant precision over a longer range than a gamma

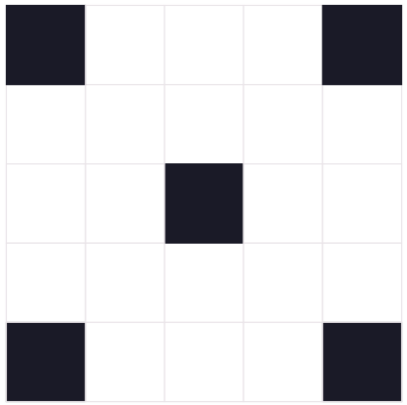
8-bit Gamma

Two-piece linear



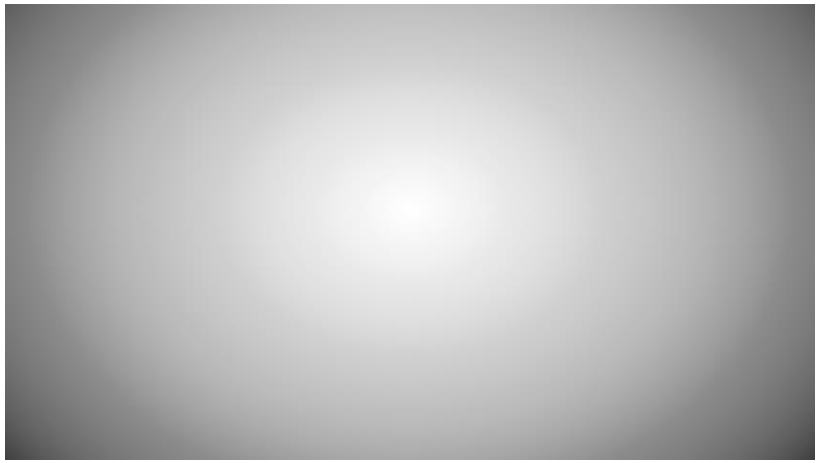
Faster Closest Velocity and Halfres Velocities

- **Idea:** precalculate nearest velocity
- Dynamic velocities typically composited with camera ones in a full screen pass
- Composite and downsample to half resolution using compute shader
  - Pick closest to camera velocity
  - Amortized if async
- Original 10-sample closest velocity reduced to:
  - 2 gathers for the velocity (`GatherRed` and `GatherGreen`)
  - 1 gather for depth
- Half resolution velocities and 8-bit:
  - Reduce velocity buffers footprint from 7.91MB to 0.98MB



Exploiting the Human Vision System

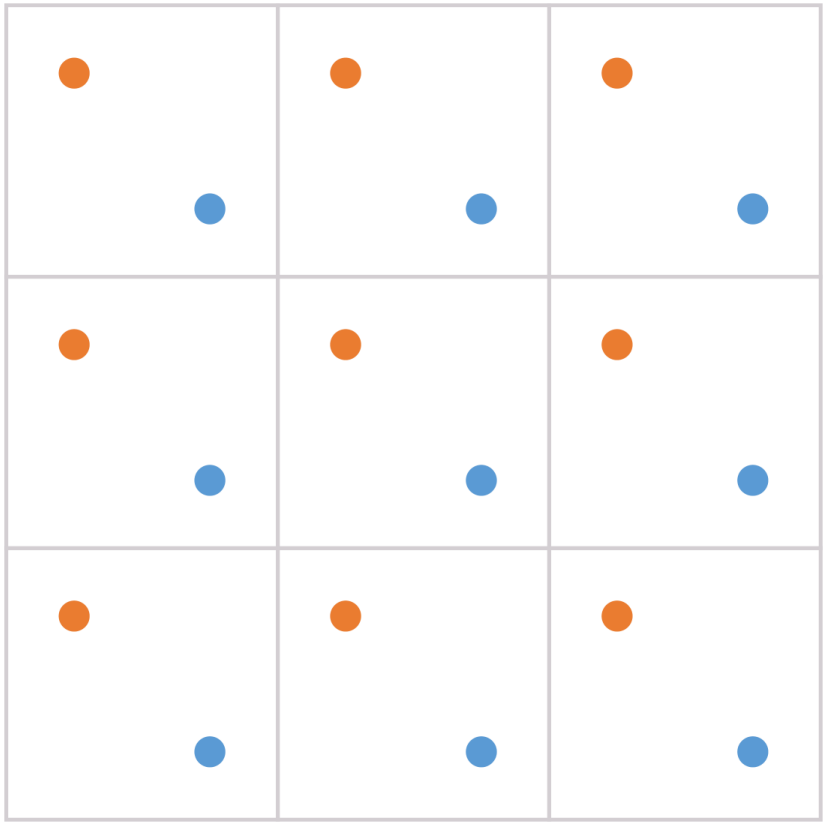
- In First Person Shooters (FPS) we typically look at the center of the screen
  - Exploit that!
  - Lower morphological/temporal filter quality on sides
  - Works particularly well with deferred queues



Quality Value
(white is higher)

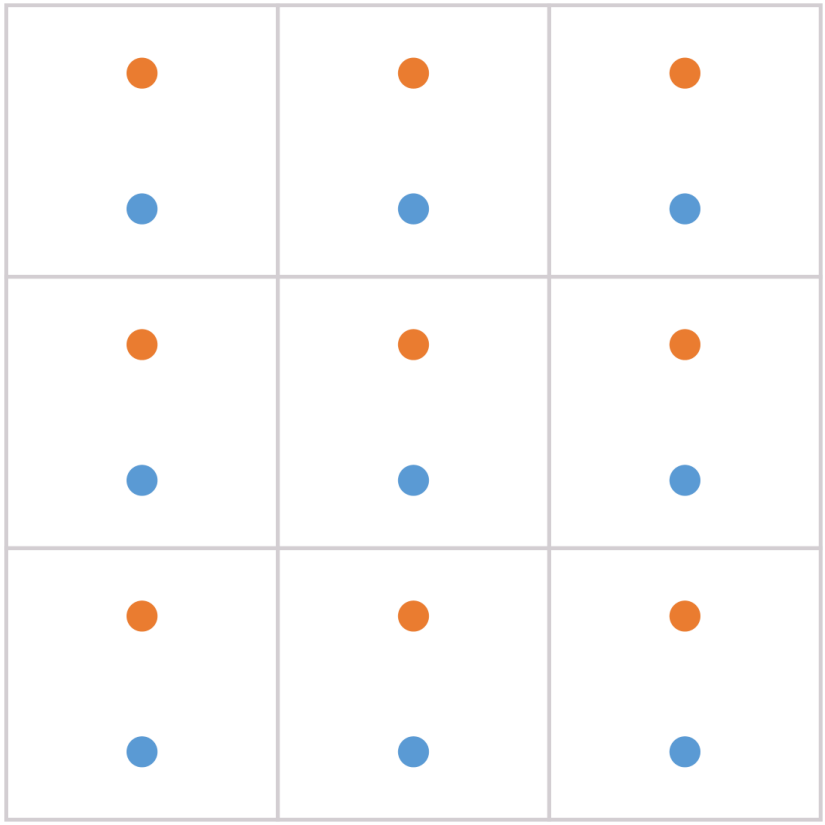
Temporal Upsampling

- So far we presented
  - Temporal supersampling
  - Temporal filtering



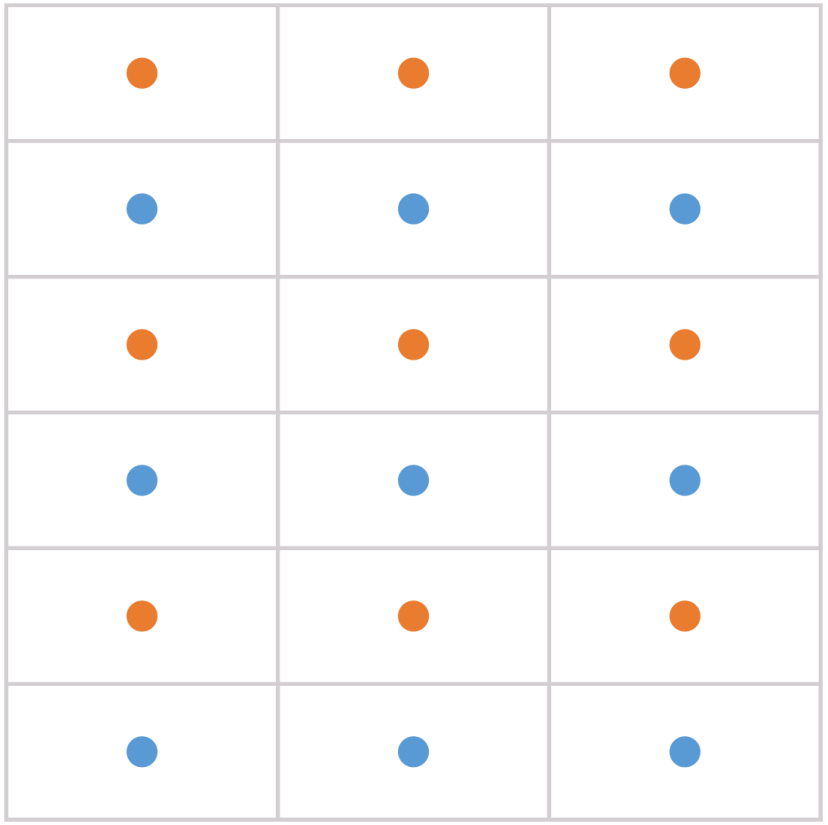
Temporal Upsampling

- Rearrange the subsamples



Temporal Upsampling

- Don’t average the subsamples
  - Output 2x resolution
- We then have
  - Temporal upsampling [Valient2014]
  - Temporal filtering



Conclusions

- Filmic SMAA T2x Highlights:
  - Sharpness
  - Robustness
  - Performance

Q&A - Acknowledgements

Special thanks to the course organizer Natasha Tatarchuk

- Angelo Pesce
- Bart Wronski
- Christer Ericson
- Dimitar Lazarov
- Eran Rich
- James Stanard
- Jennifer Velazquez
- Josean Checa
- Michael Vance
- Michal Drobot
- Paul Edelstein
- Peter-Pike Sloan
- Tiago Sousa
- Unai Landa
- Wade Brainerd
- Xianchun Wu

References

- [Sousa2011] Anti-Aliasing Methods in CryENGINE 3
- [Sousa2013] CryENGINE3 Graphics Gems
- [Jimenez2012] SMAA: Enhanced Subpixel Morphological Antialiasing
- [Drobot2014] Hybrid Reconstruction Antialiasing
- [Karis2004] High Quality Temporal Supersampling
- [Lottes2011] Temporal Supersampling (blog post, no longer available)
- [Salvi2016] An Excursion in Temporal Supersampling
- [Valient2014] Taking Killzone: Shadow Fall Image Quality Into the Next Generation