Data-parallel Rasterization of Micropolygons with Defocus and Motion Blur

Kayvon Fatahalian, Edward Luong, Solomon Boulos, Pat Hanrahan
Stanford University

Kurt Akeley
Microsoft Research

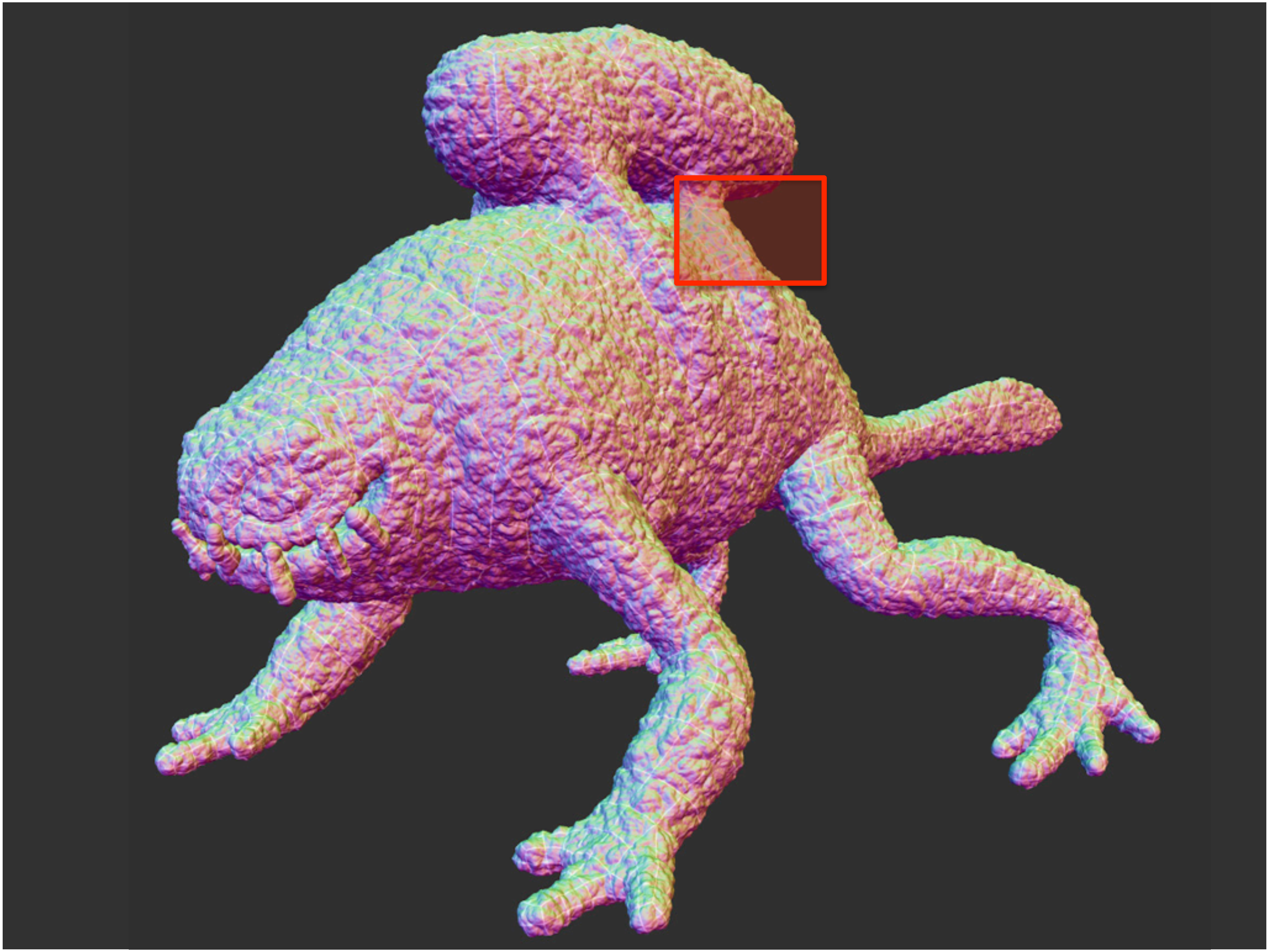
William R. Mark
Intel Corporation

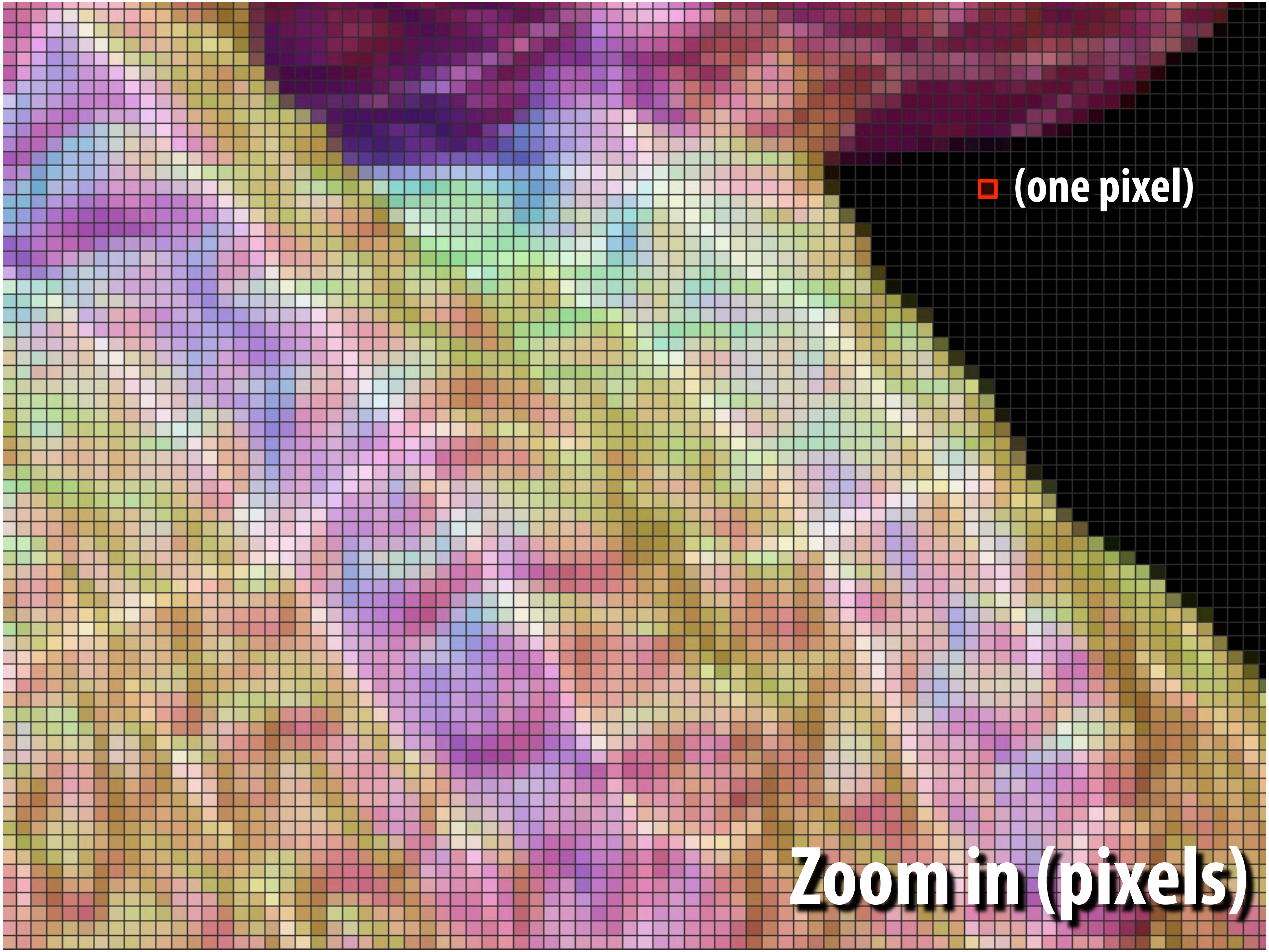
Detailed surfaces

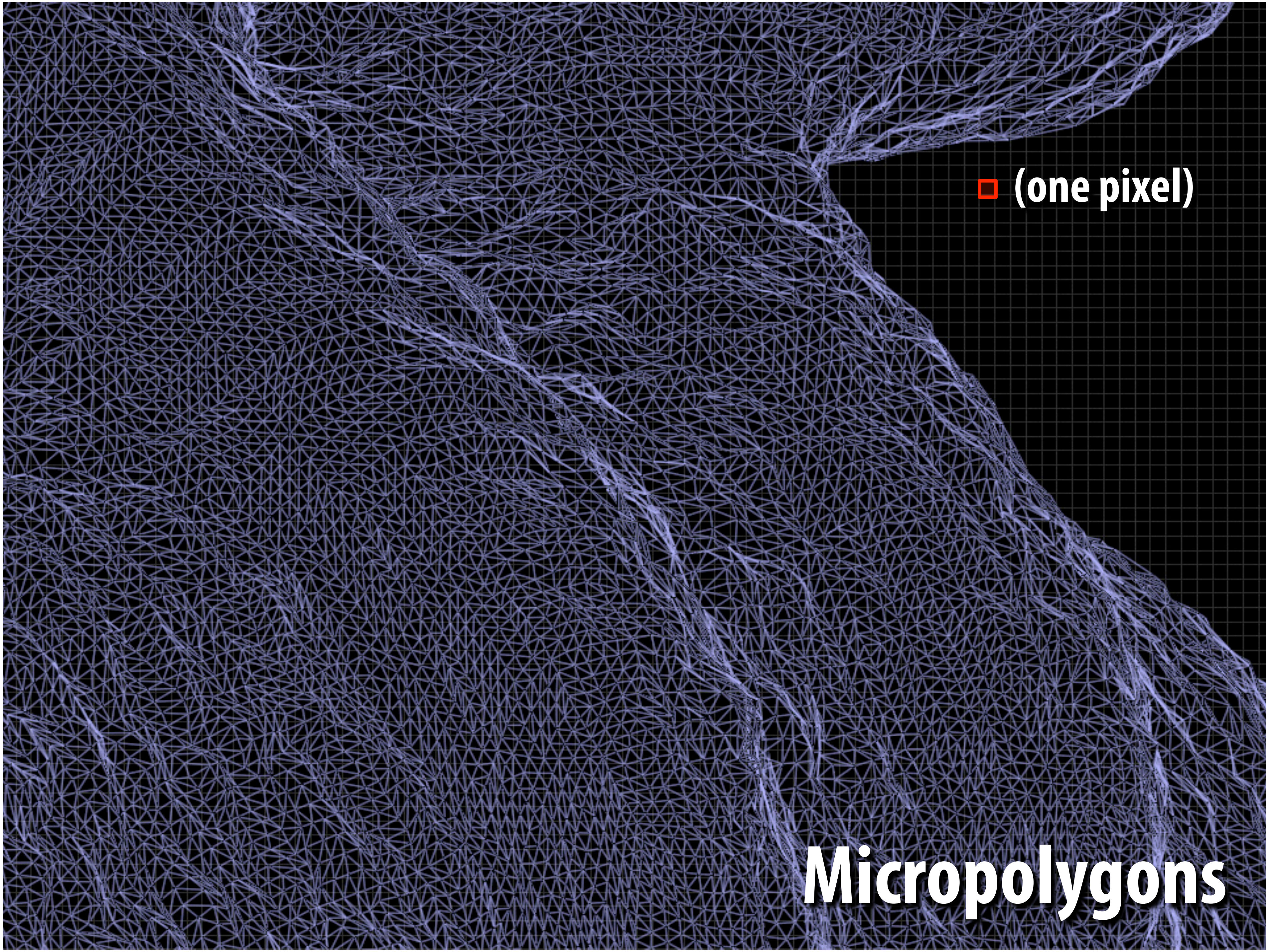
Credit: DreamWorks Pictures, Shrek 2 (2004)

Credit: Pixar Animation Studios, Toy Story 2 (1999)

Credit: Crytek

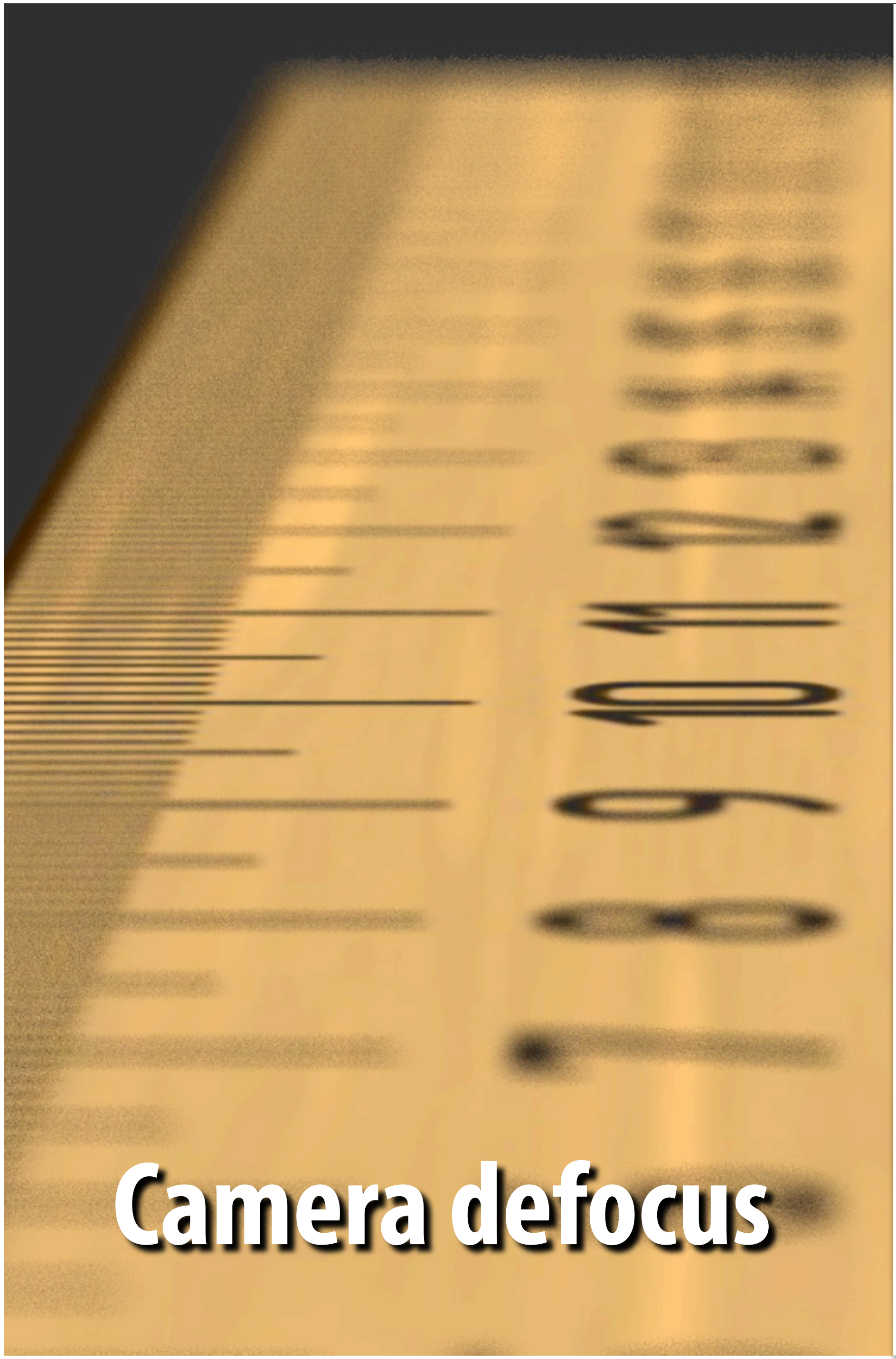



(one pixel)
Zoom in (pixels)

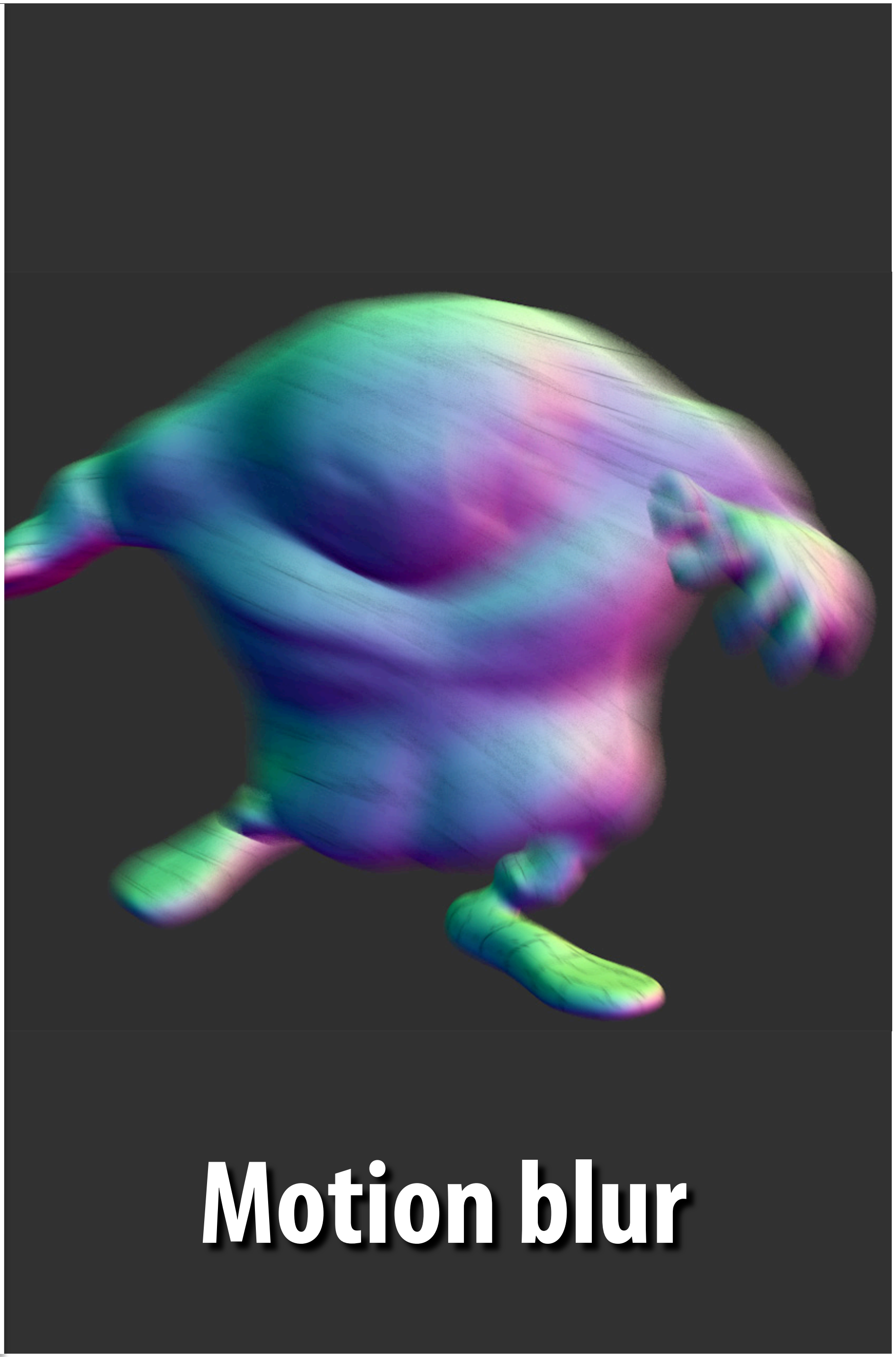


□ (one pixel)

Micropolygons



Camera defocus



Motion blur

Rendering goals

Highly detailed surfaces (micropolygons)

Accurate camera defocus and motion blur

[Future] real-time system

How do we evolve the real-time graphics pipeline to enable efficient micropolygon rendering?

This talk: rasterizing micropolygons

How is micropolygon-sample coverage computed efficiently?

How expensive are motion blur and defocus?

Contributions

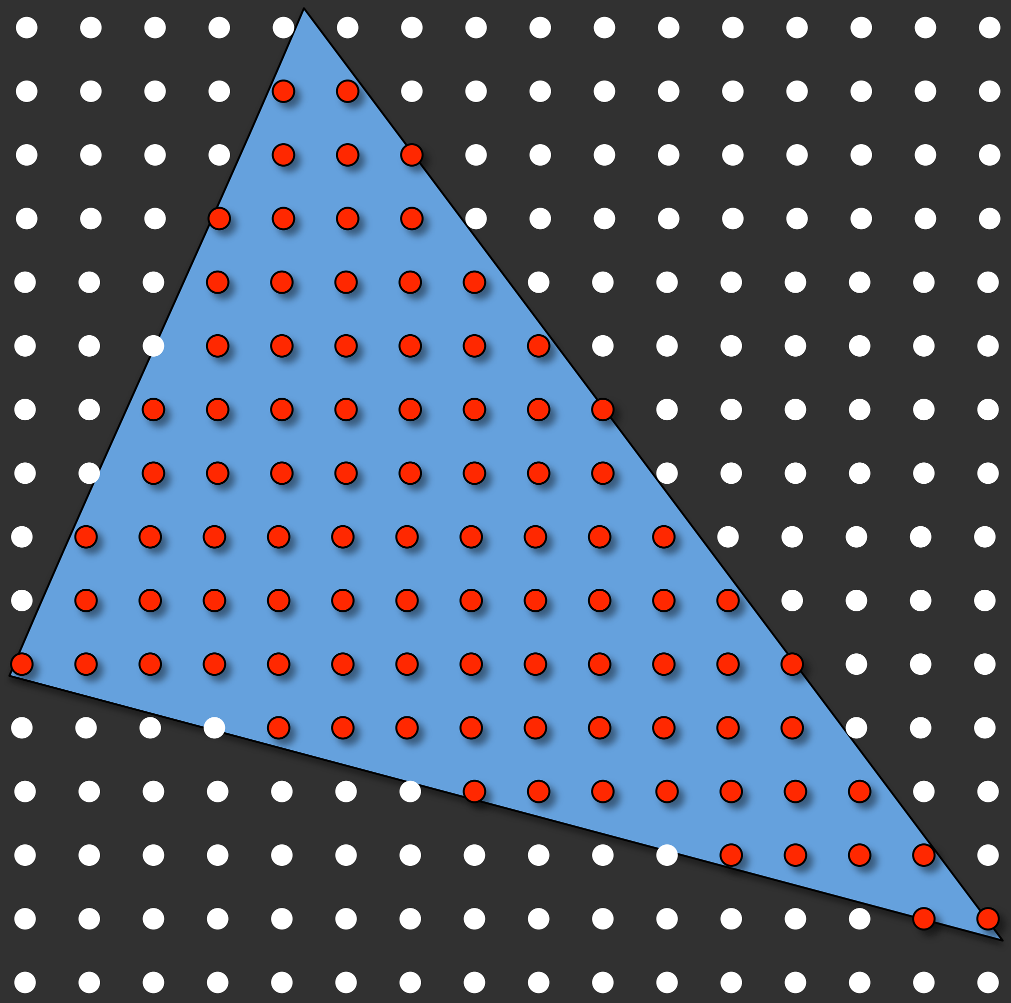
Design and analysis of three data-parallel algorithms for micropolygon rasterization

- Re-optimize rasterization for micropolygon workloads
  - NOBLUR
- Extend rasterizer to support camera defocus and motion blur
  - INTERVAL: vector implementation of Pixar algorithm
  - INTERLEAVE: leverage interleaved sampling for better perf

BACKGROUND

(no motion, no defocus)

Rasterization



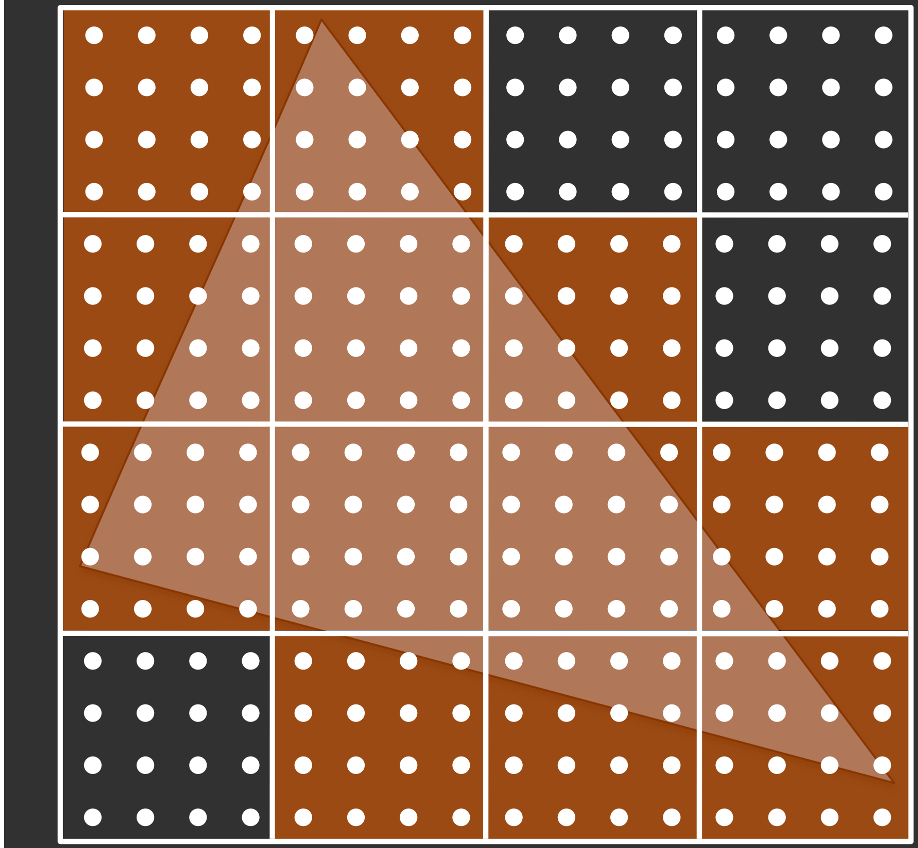
Step 1: per-polygon preprocessing (setup)

- Clip, back face cull, compute edge equations

- Make point-in-polygon tests cheap

Step 2: compute candidate sample set

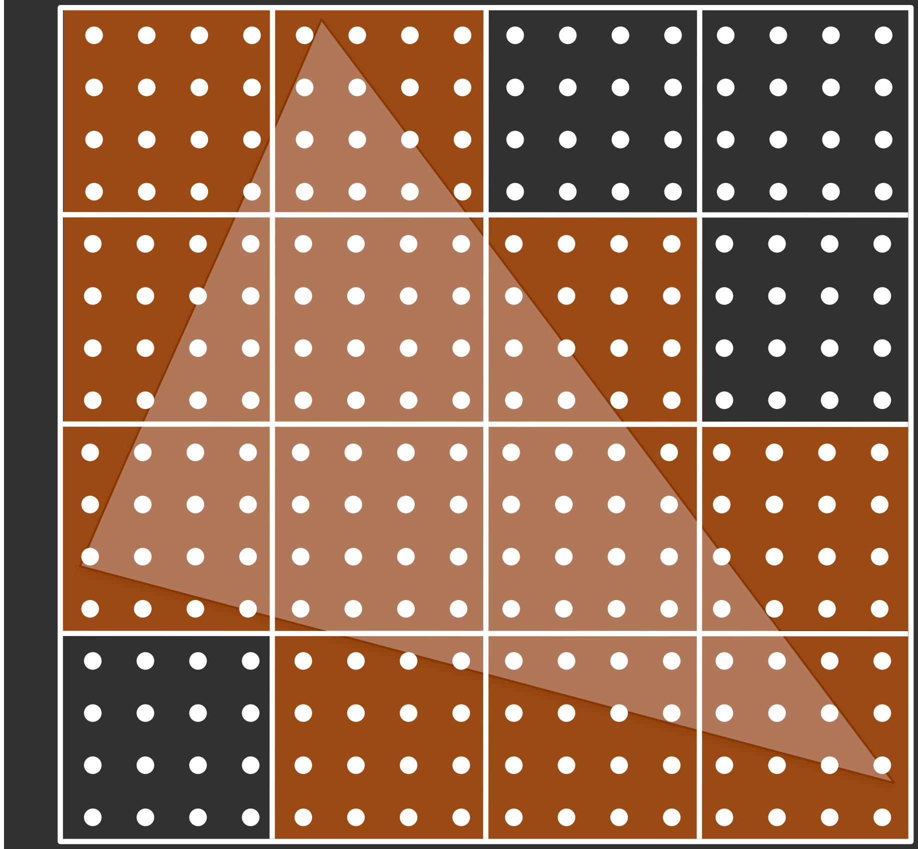
- Coarse reject/accept of samples



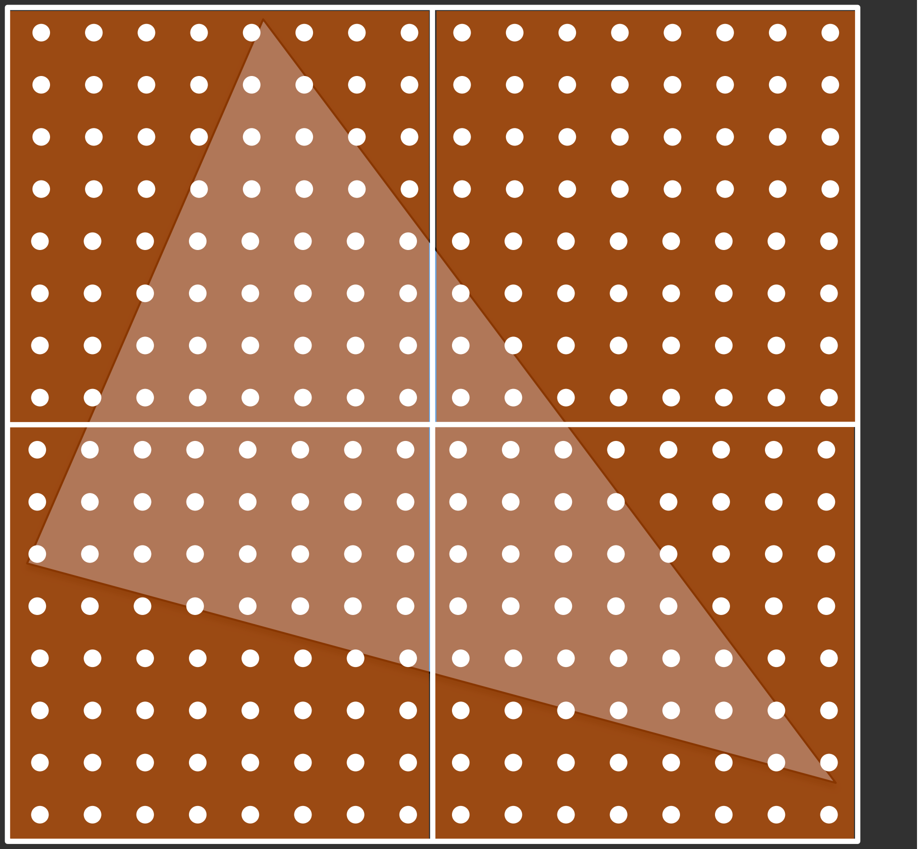
Coarse uniform grid

Step 2: compute candidate sample set

- Coarse reject/accept of samples



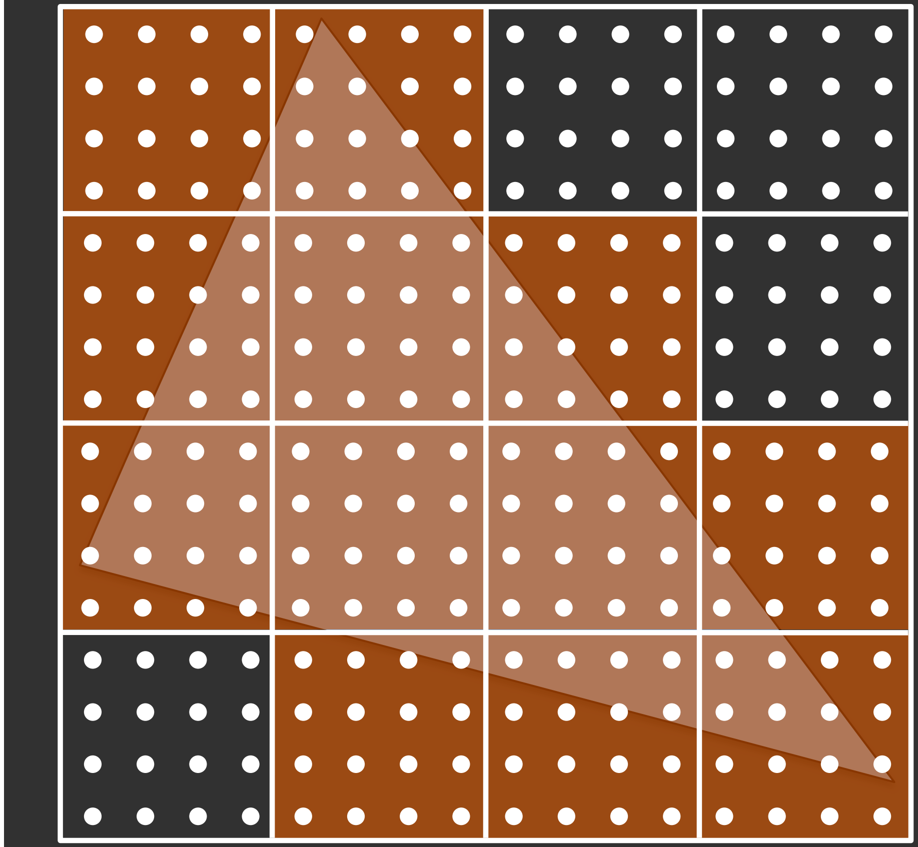
Coarse uniform grid



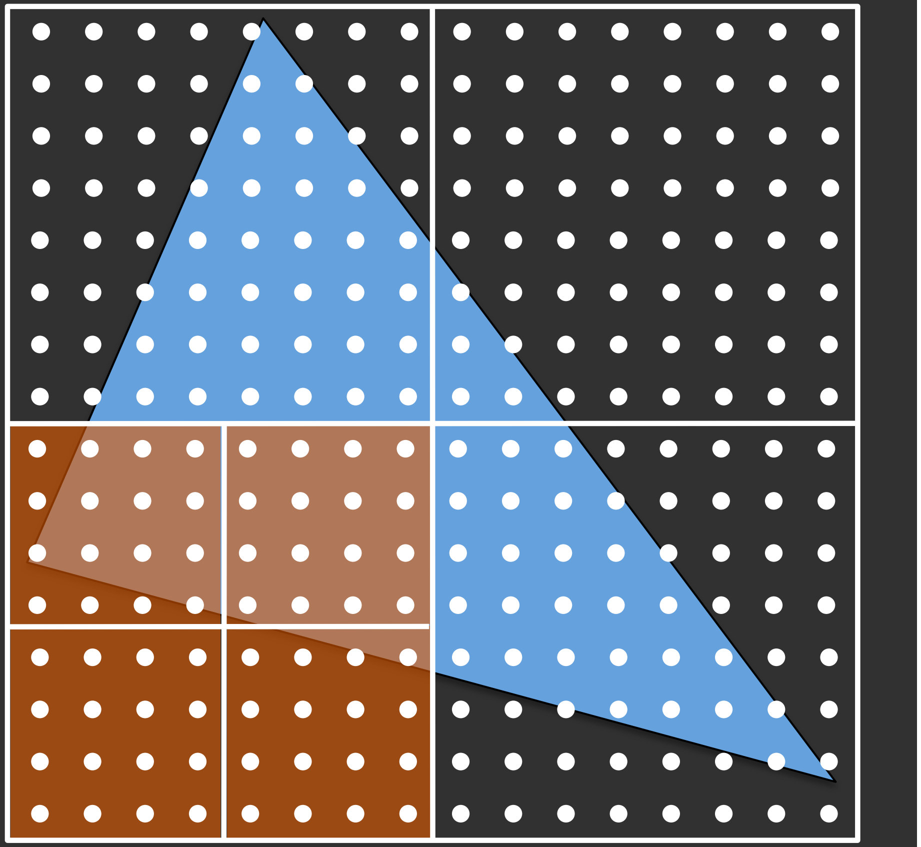
Hierarchical descent

Step 2: compute candidate sample set

- Coarse reject/accept of samples



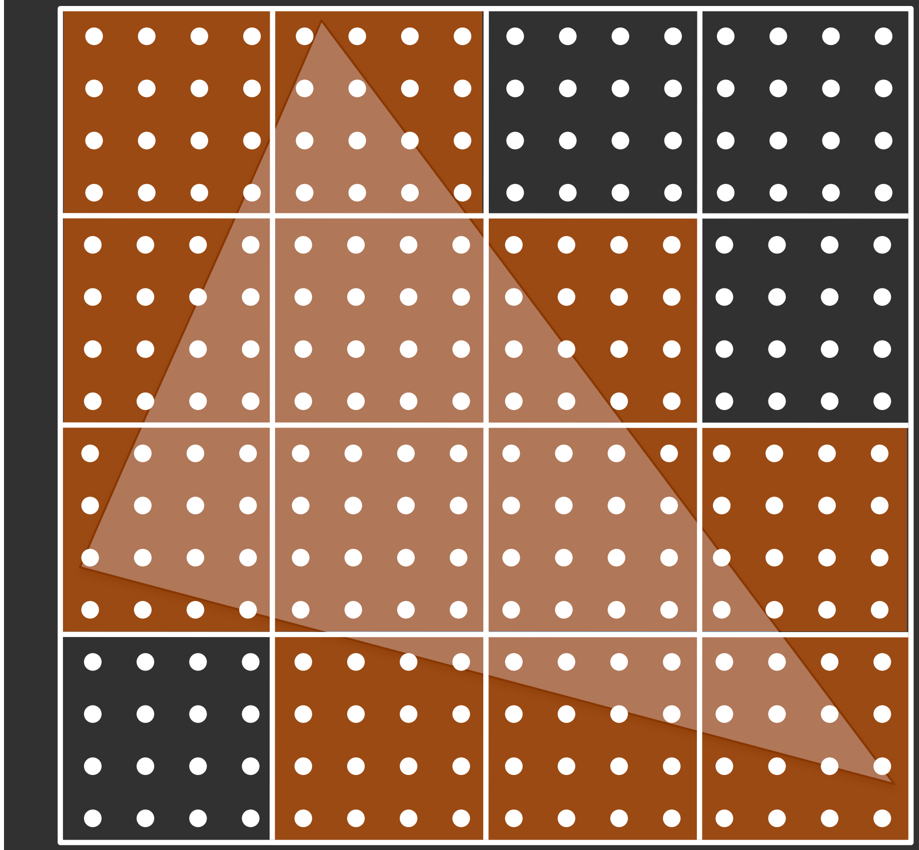
Coarse uniform grid



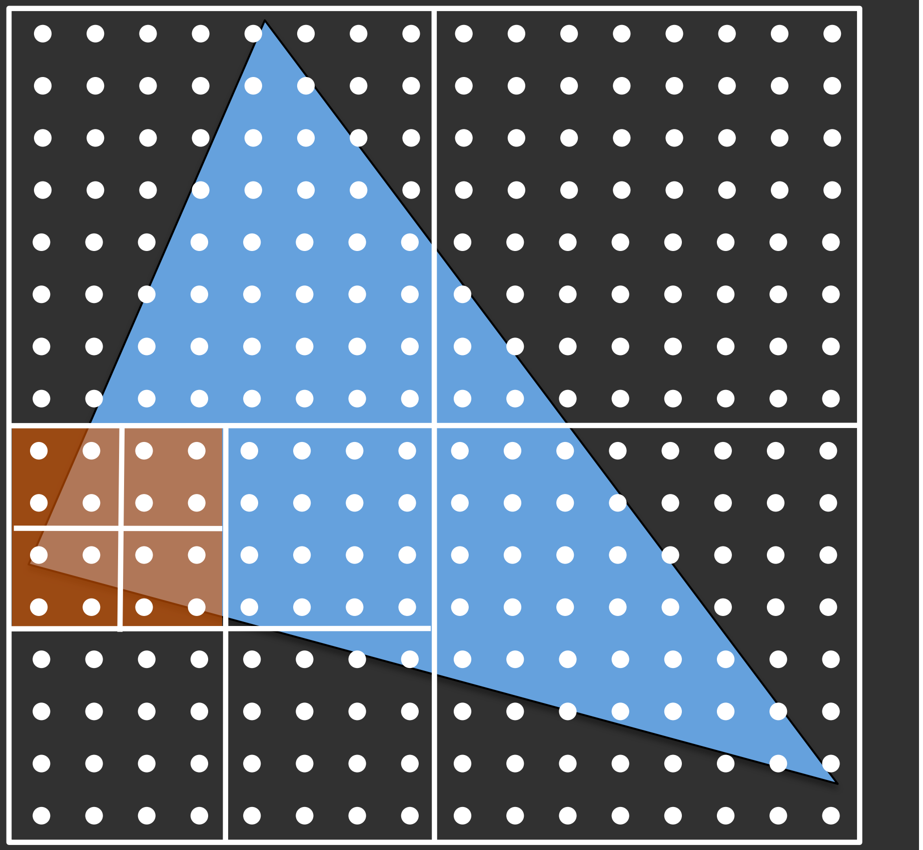
Hierarchical descent

Step 2: compute candidate sample set

- Coarse reject/accept of samples



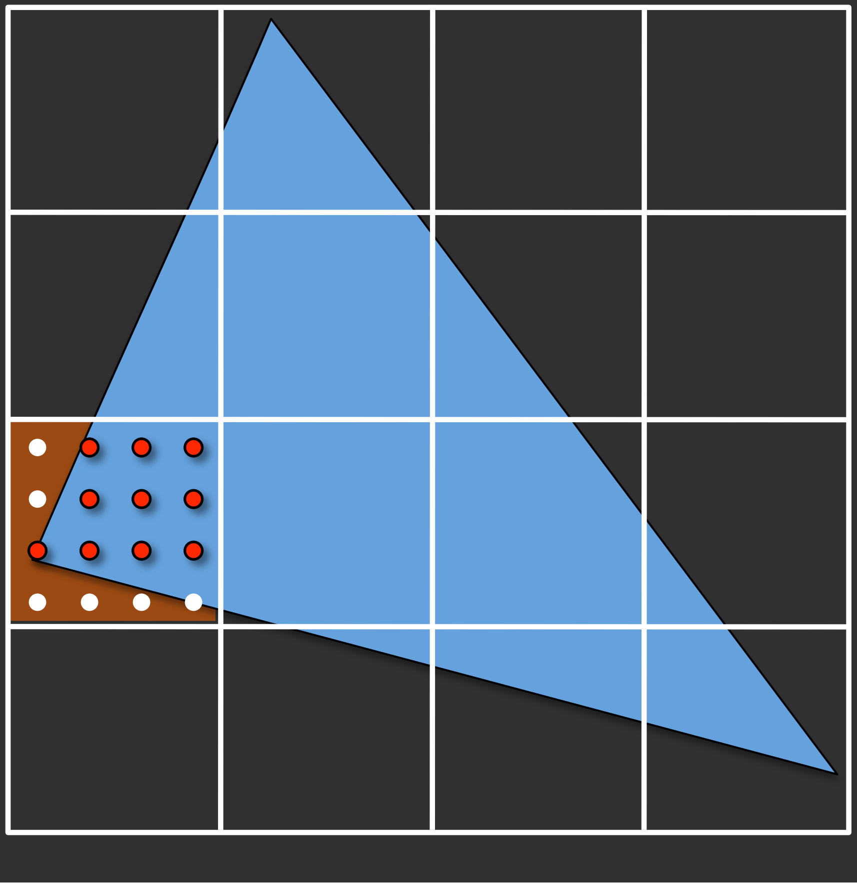
Coarse uniform grid



Hierarchical descent

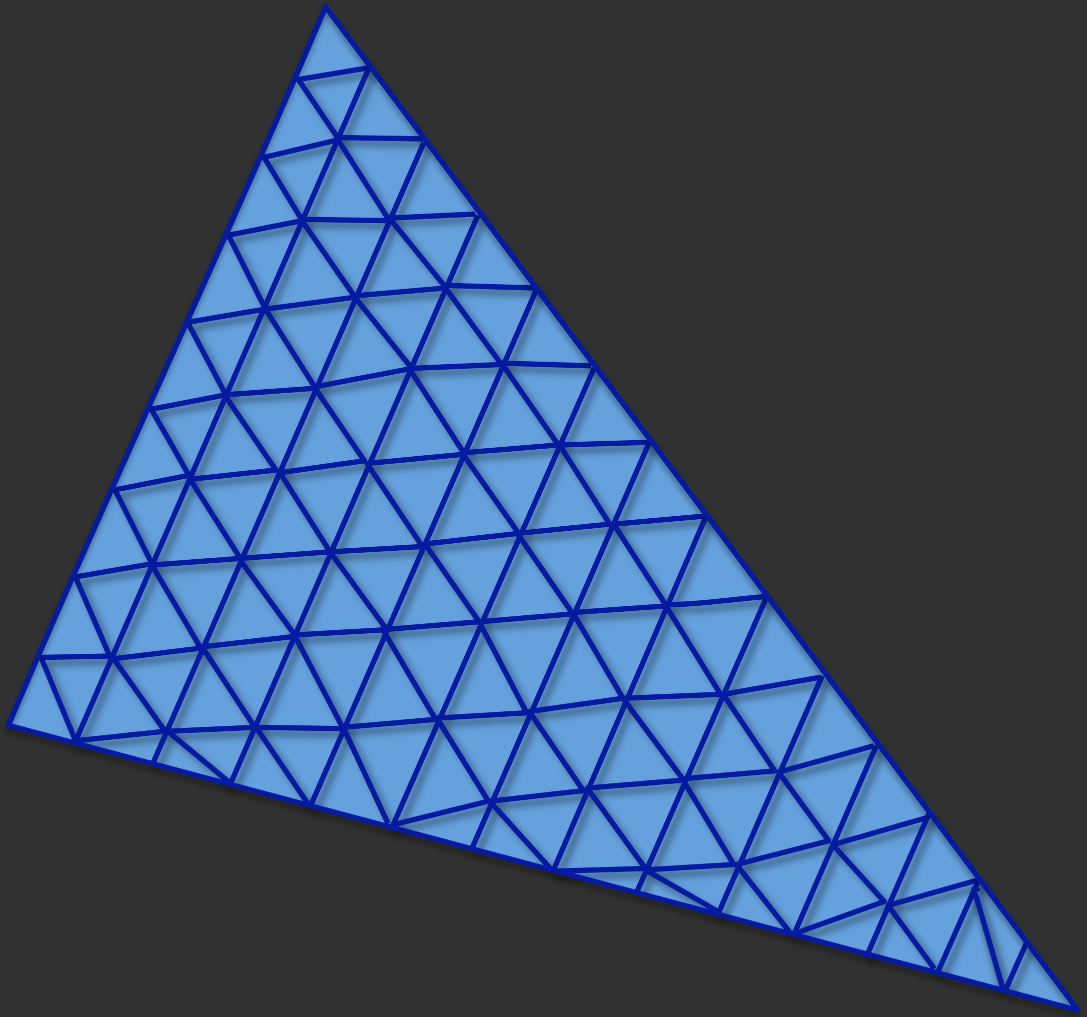
Step 3: point-in-polygon tests

- Test "stamp" of samples against polygon simultaneously (data-parallel)

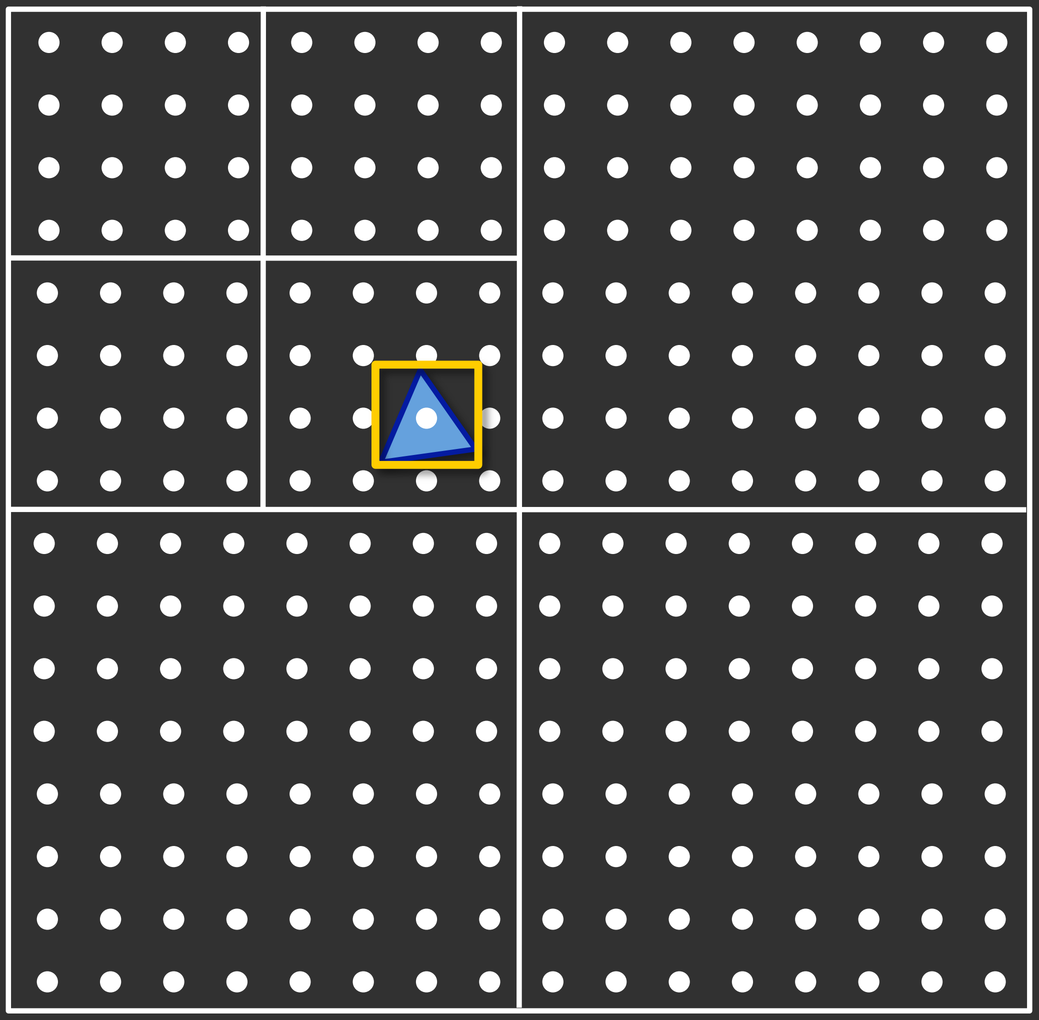


[Pineda 88]
[Fuchs 89]
[Greene 96]
[Seiler 08]

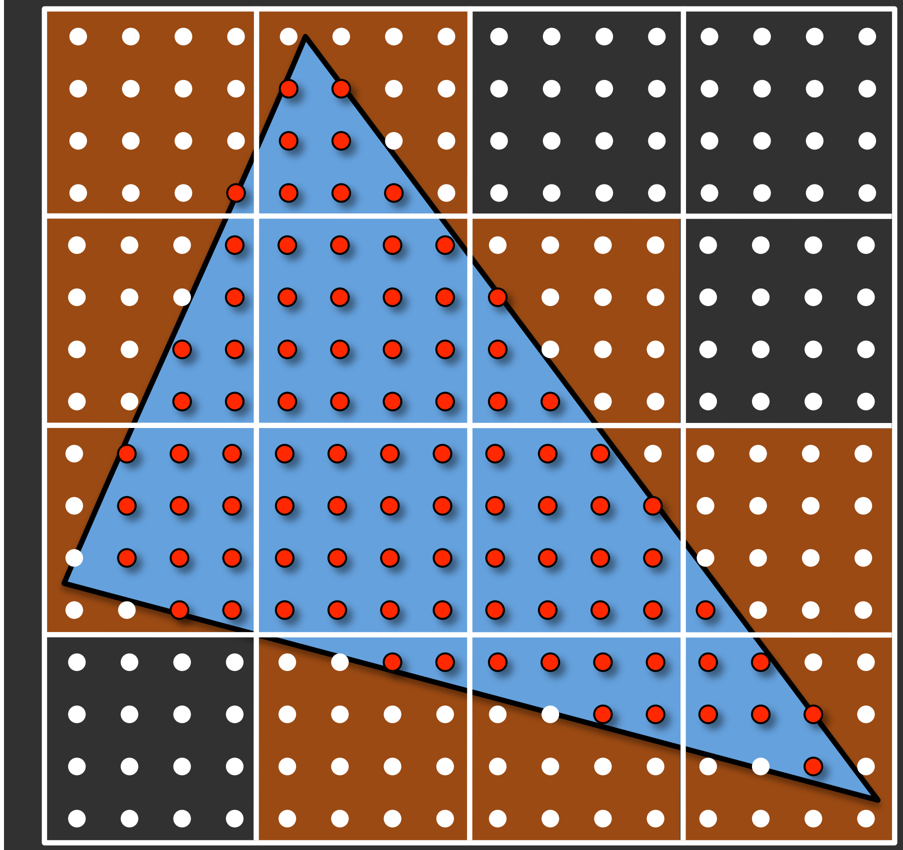
Micropolygons: more polygons = more setup



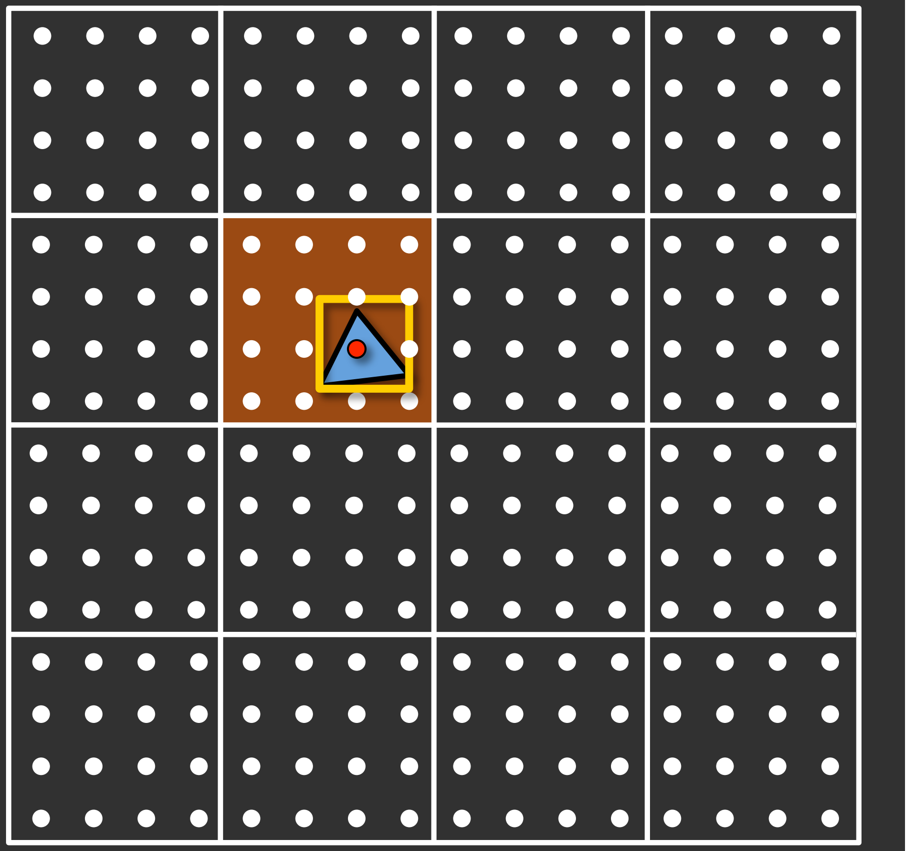
Micropolygons: coarse reject not useful



Micropolygons: large stamps yield low efficiency



47% of tested samples inside triangle



6% of tested samples inside triangle

ALGORITHM #1: NOBLUR

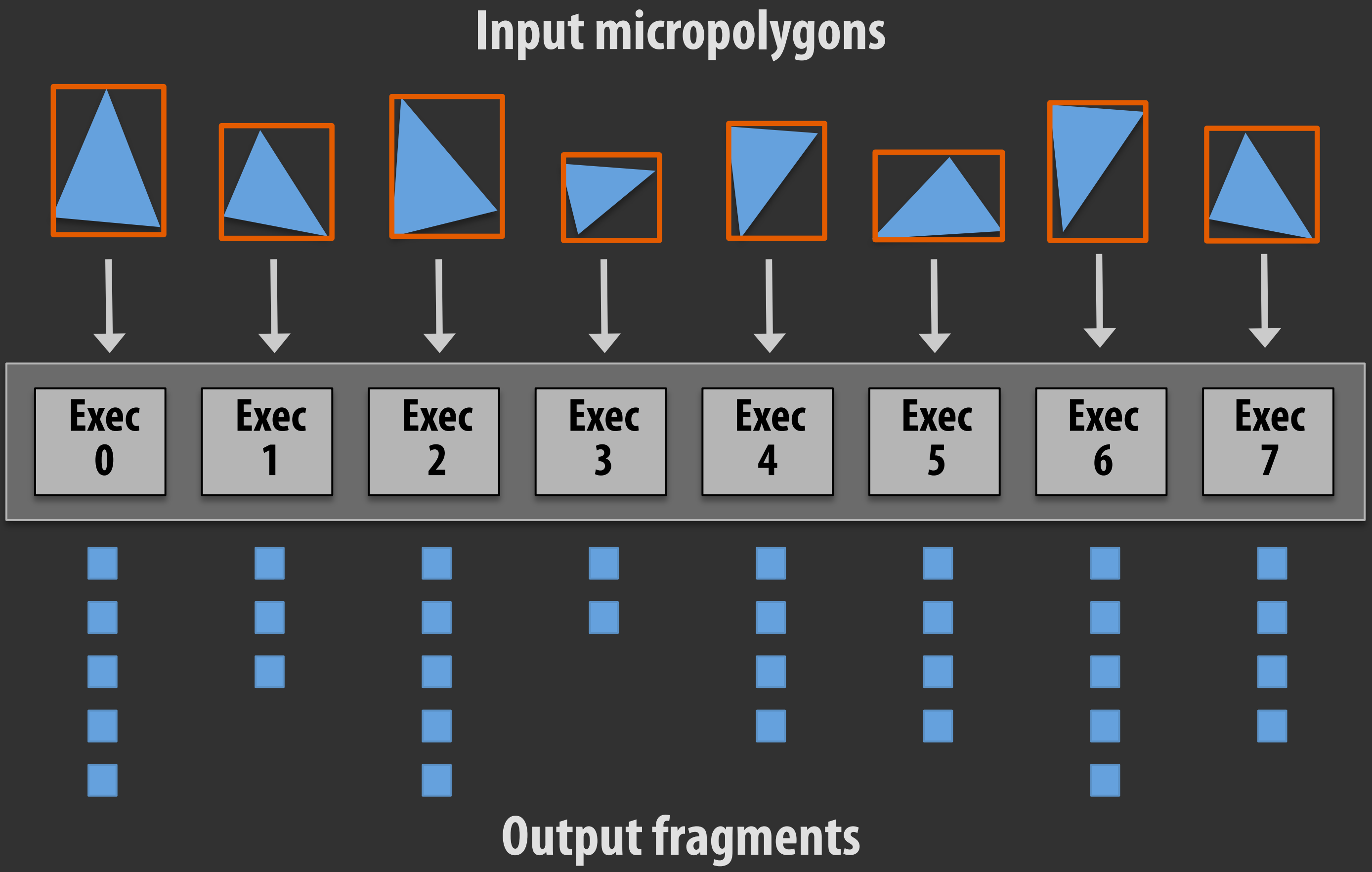
(no motion, no defocus)

NOBLUR

```
        For each MP
--------------------------------------------
Setup       Cull backfacing
--------------------------------------------
Bound       Compute subpixel bbox of MP
--------------------------------------------
            For each sample in bbox
Test           Test MP-sample coverage
```

NOBLUR parallelization

- Rasterize many micropolygons simultaneously



NOBLUR parallelization

```
        For each MP                                          PARALLEL

Setup   Cull backfacing

Bound   Compute subpixel bbox of MP

Test    For each sample in bbox
           Test MP-sample coverage                        UTILIZATION?
```

MOTION BLUR AND DEFOCUS

Motion blur and defocus

- **Many 2D-techniques for approximating blur**

  [Sung 02]

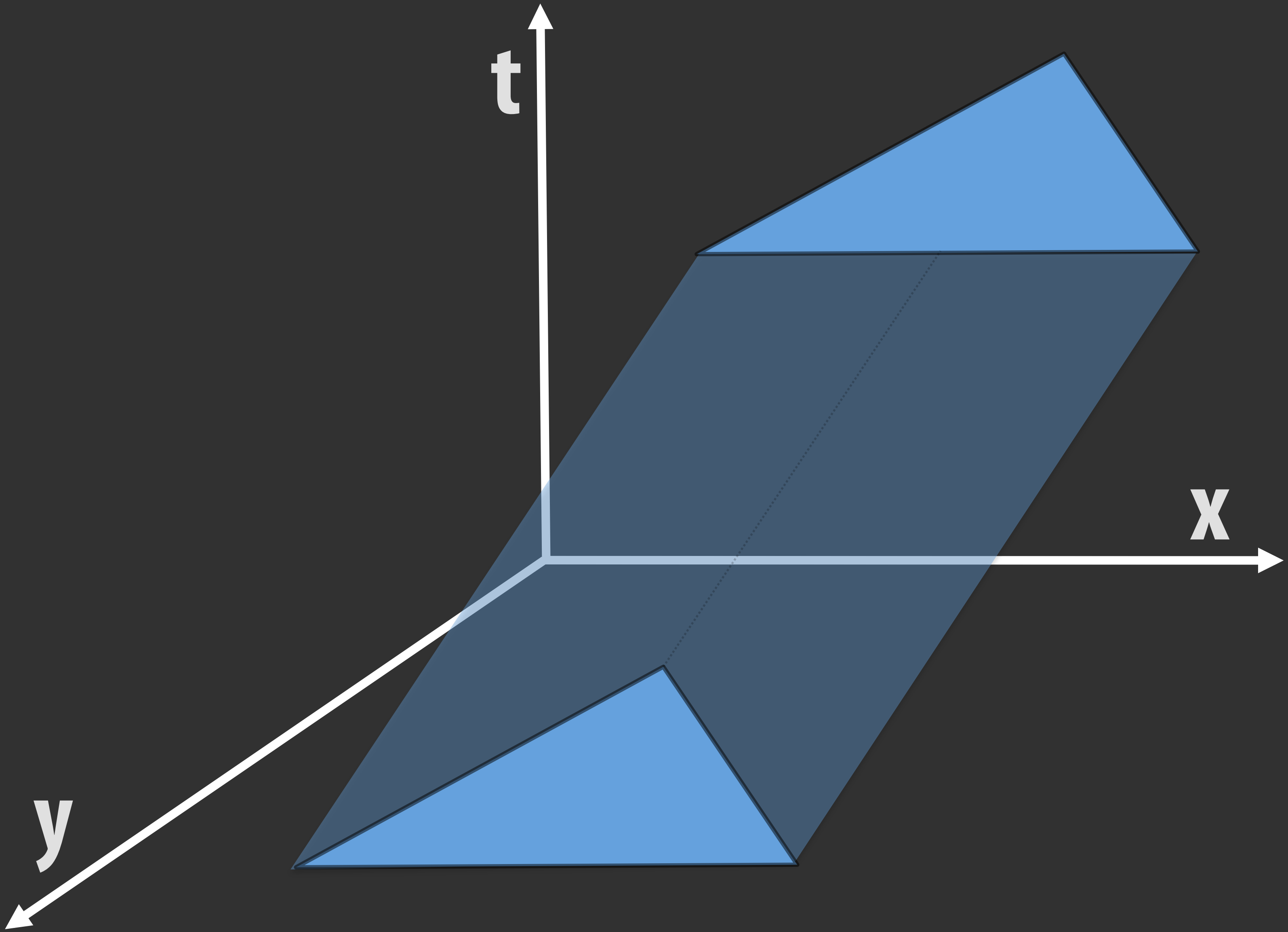
  [Demers 04]

- **Stochastic point sampling**

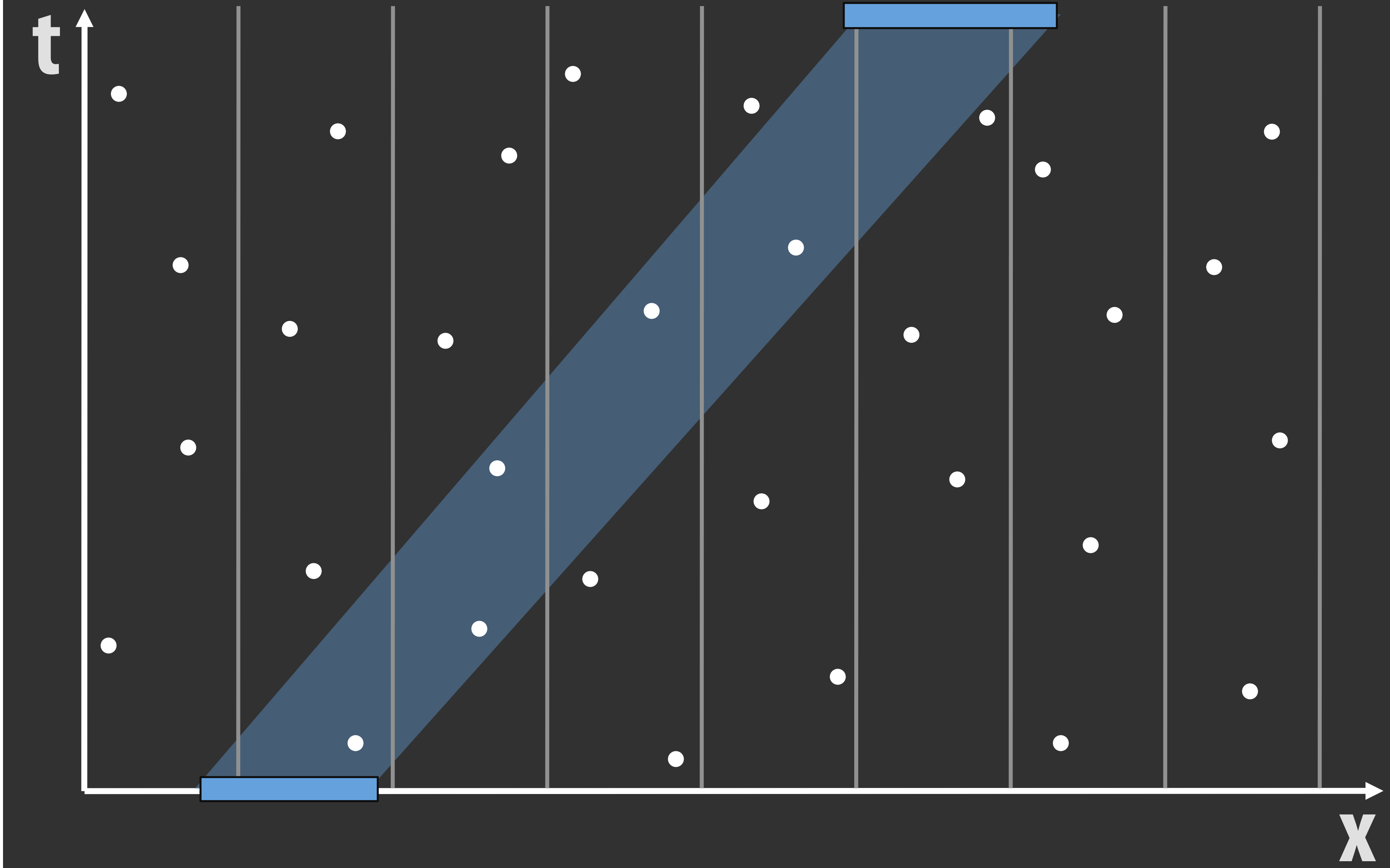
  [Cook 84, Cook 86]

  [Akenine-Moller 07]

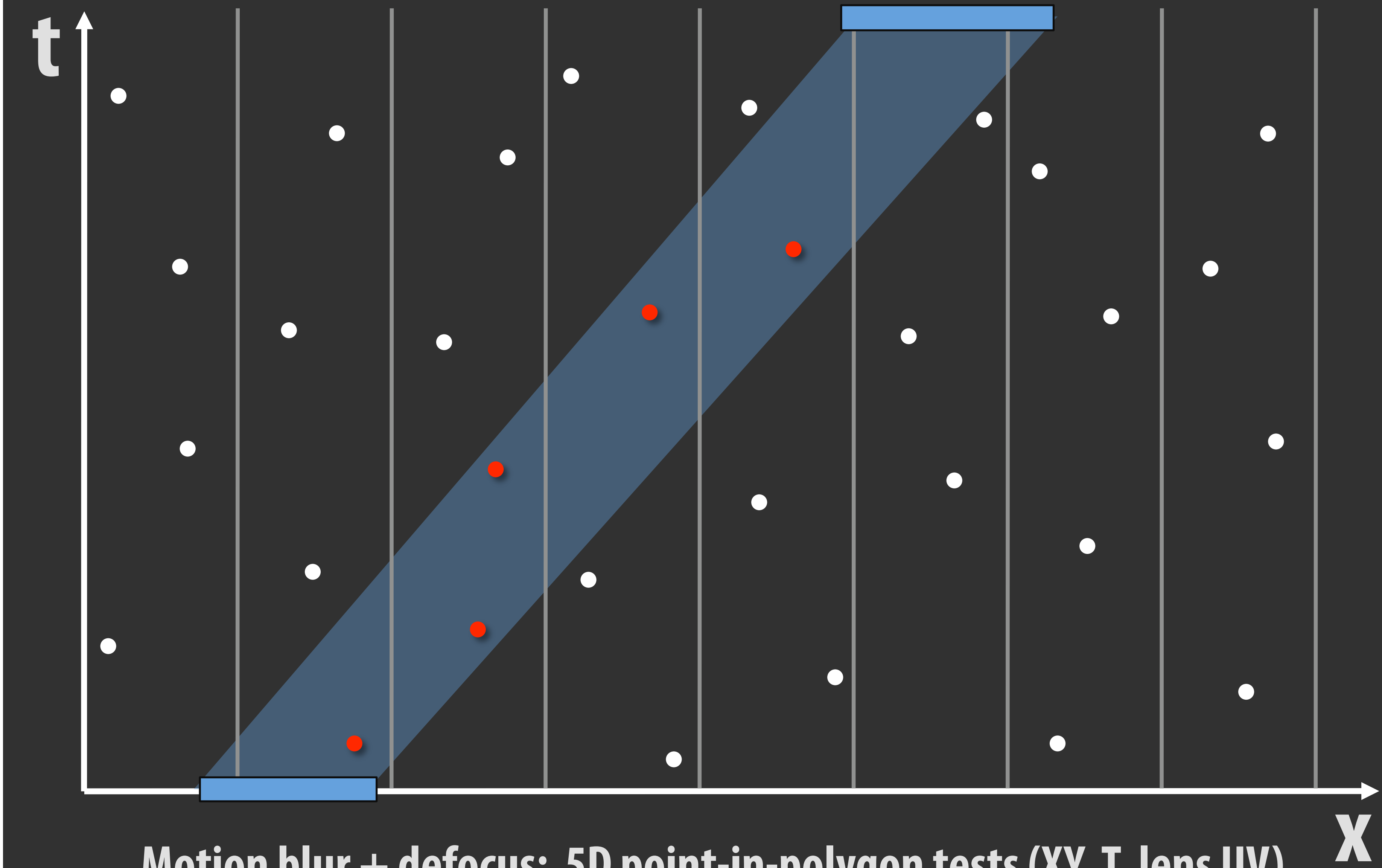
Moving micropolygon



X,T plane

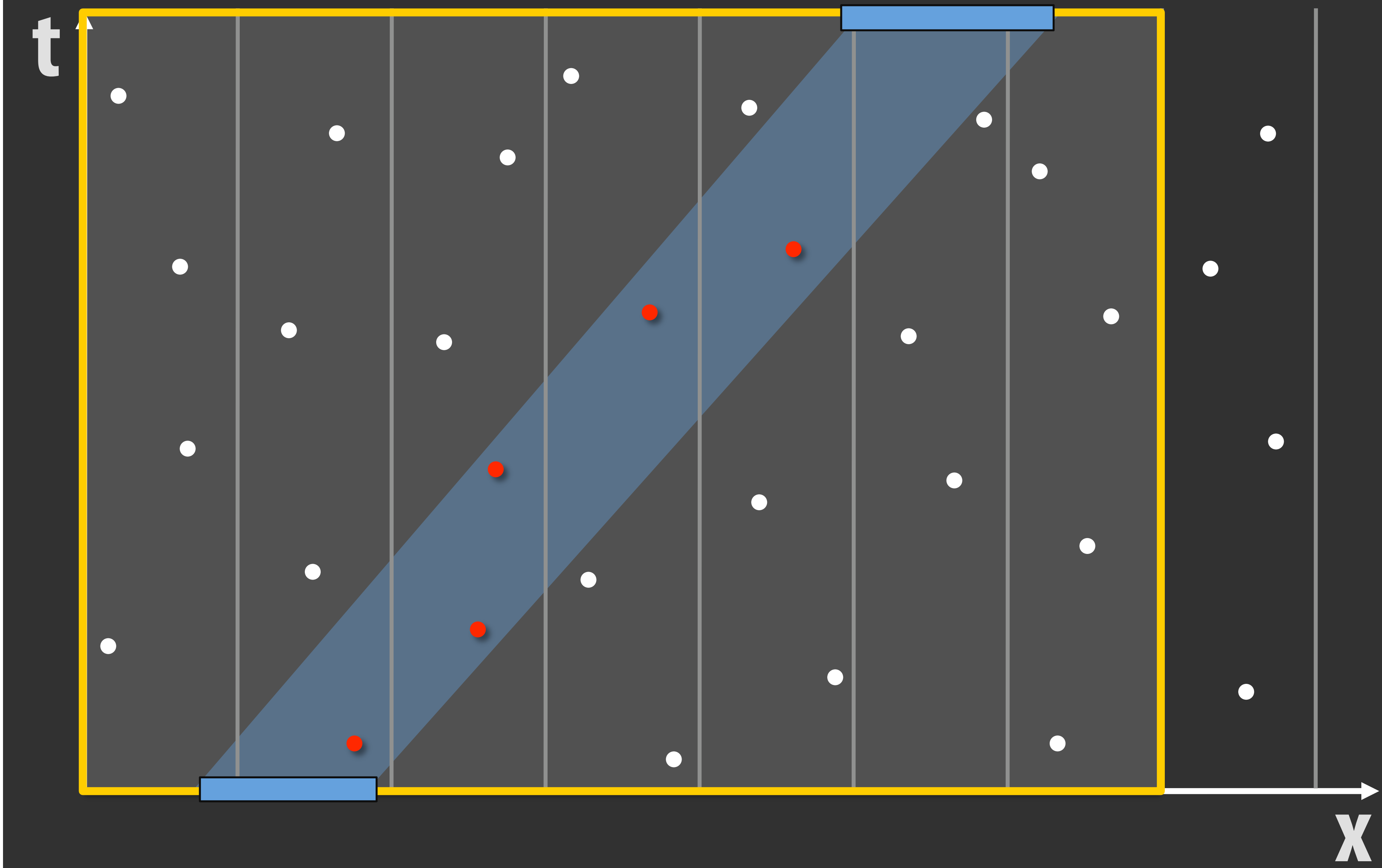


X,T plane



Motion blur + defocus: 5D point-in-polygon tests (XY, T, lens UV)

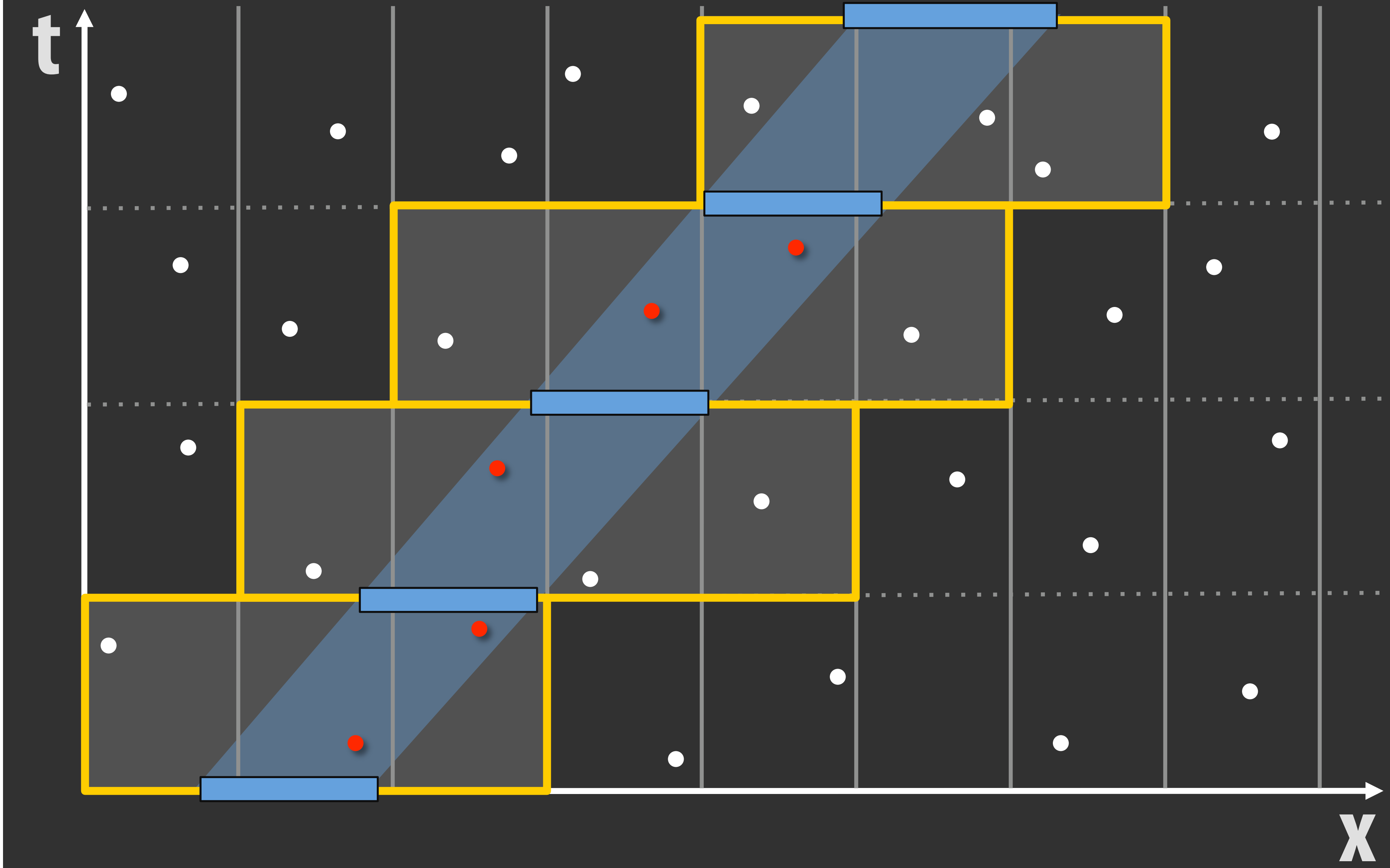
Candidate samples



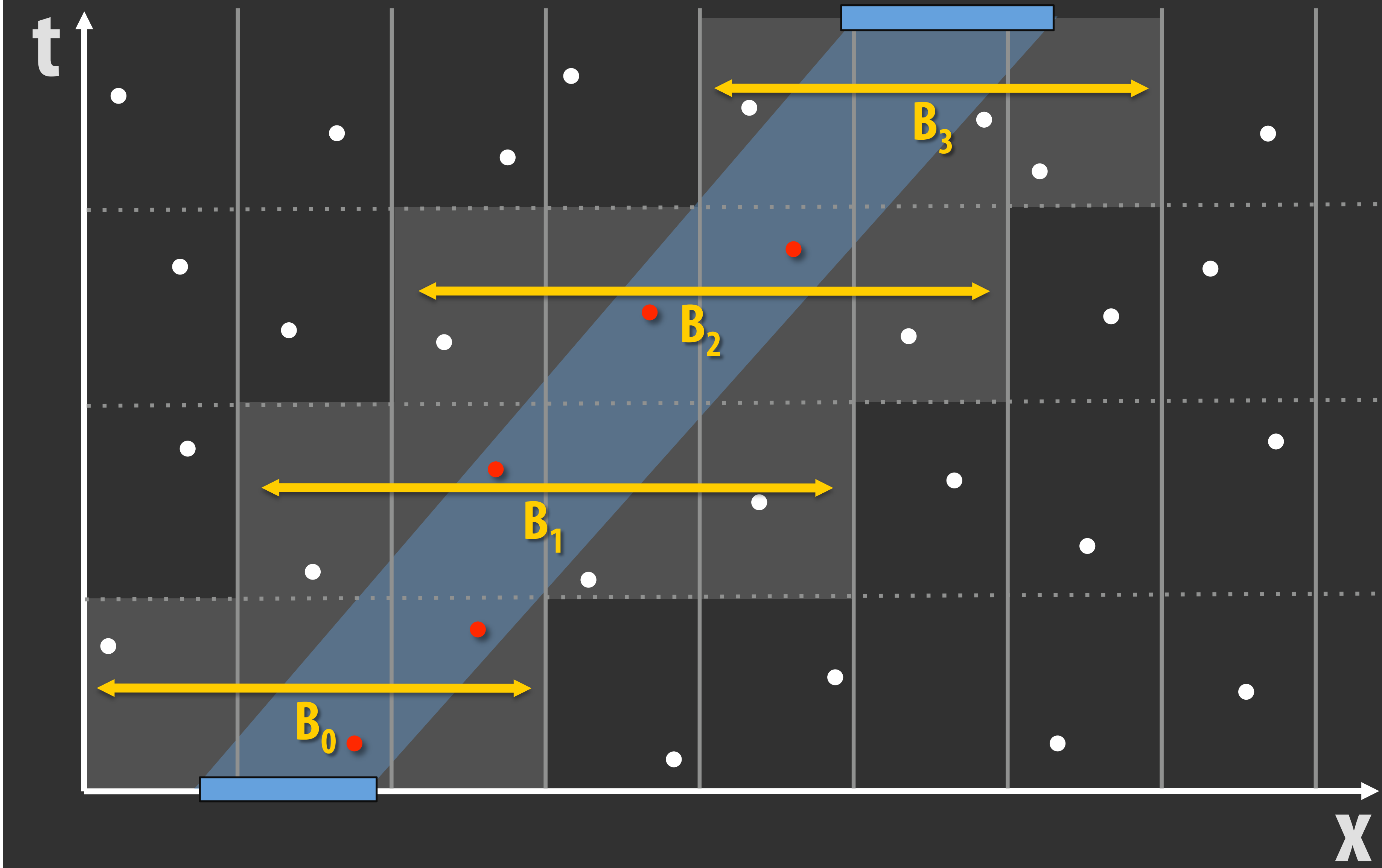
ALGORITHM #2: INTERVAL

[Cook 90]

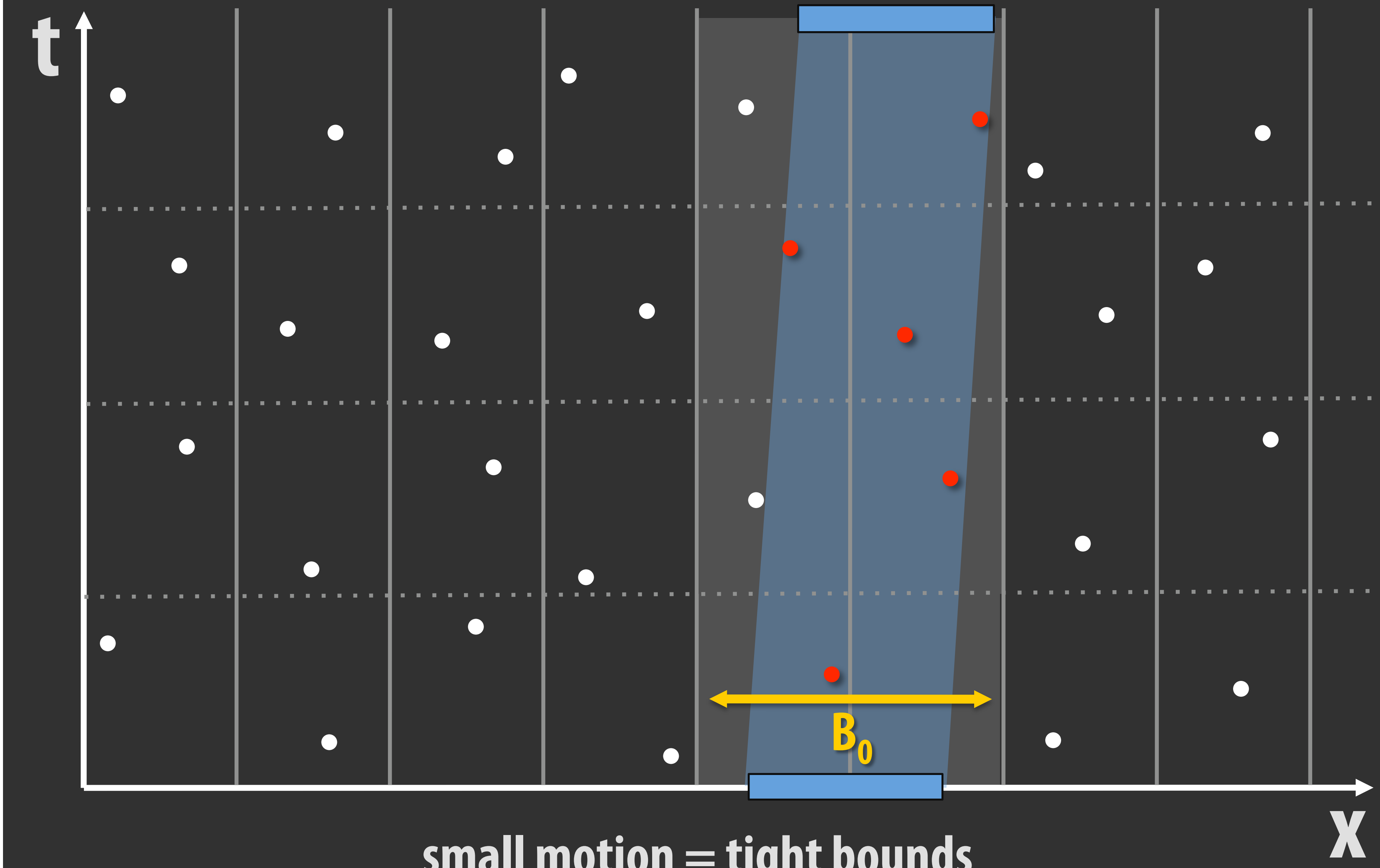
INTERVAL (4 time intervals)



INTERVAL (4 time intervals)

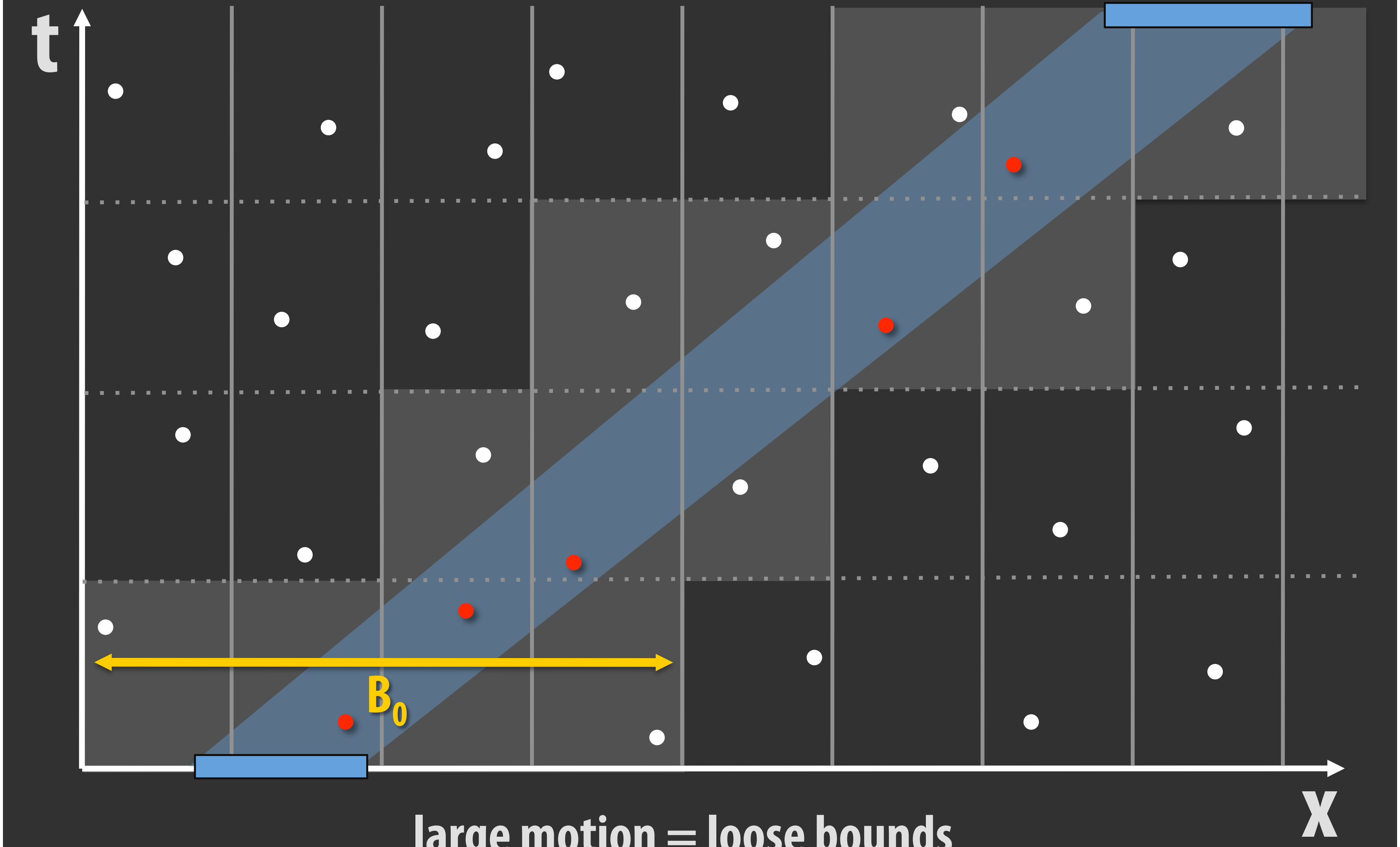


INTERVAL



small motion = tight bounds

INTERVAL



large motion = loose bounds

INTERVAL

```
         For each MP
--------------------------------------------------------
Setup       ...
--------------------------------------------------------
            For each time interval
Bound         Compute MP bbox over interval
--------------------------------------------------------
              For each sample in interval and in bbox
Test            Position MP at sample T
                Test MP-sample coverage
```

INTERVAL parallelization

```
        For each MP                                      PARALLEL

Setup       ...

Bound       For each time interval                       PARALLEL
              Compute MP bbox over interval

Test          For each sample in interval and in bbox
                Position MP at sample T
                Test MP-sample coverage              UTILIZATION?
```

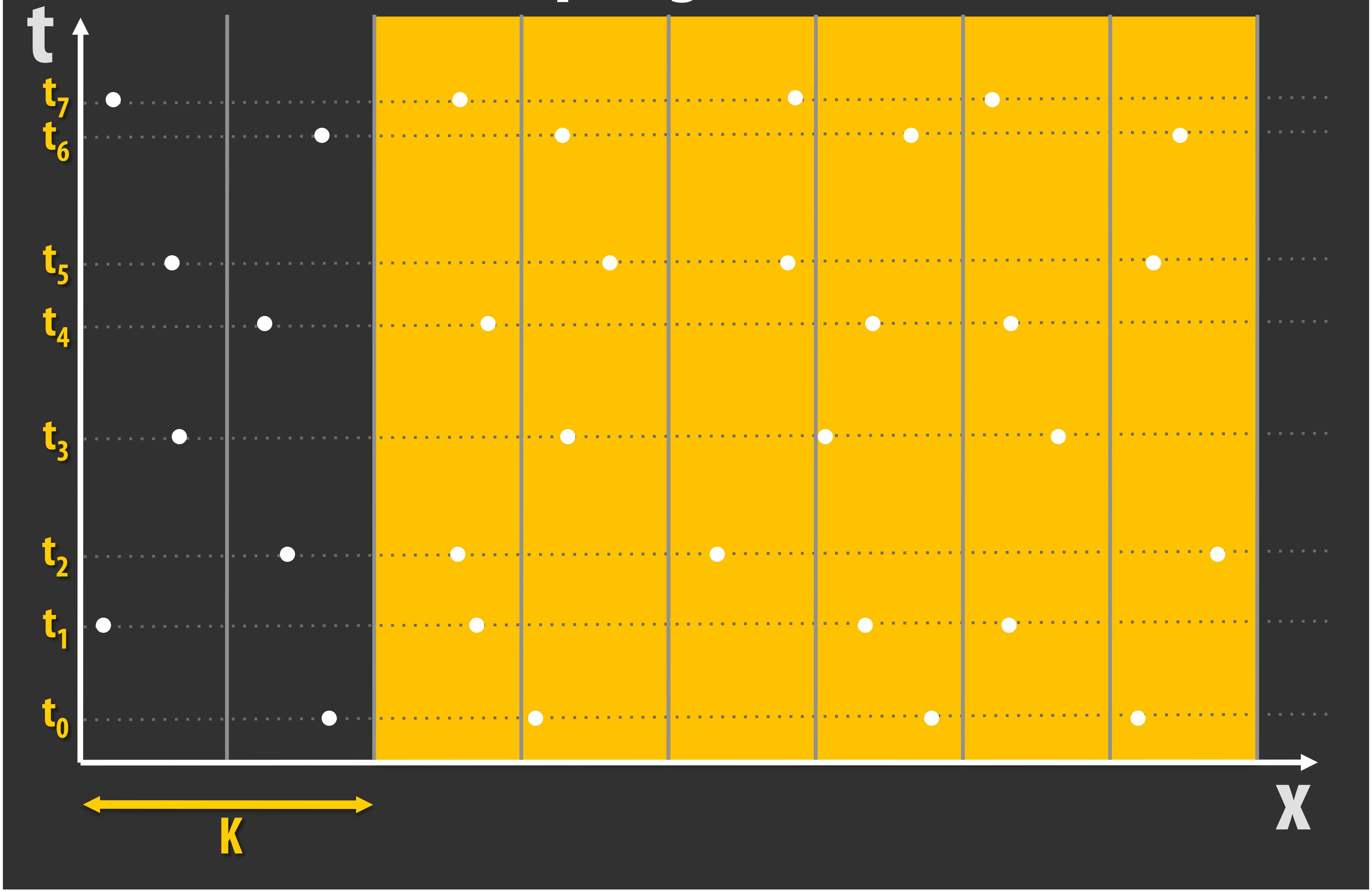
ALGORITHM #3: INTERLEAVE

INTERLEAVE: main idea

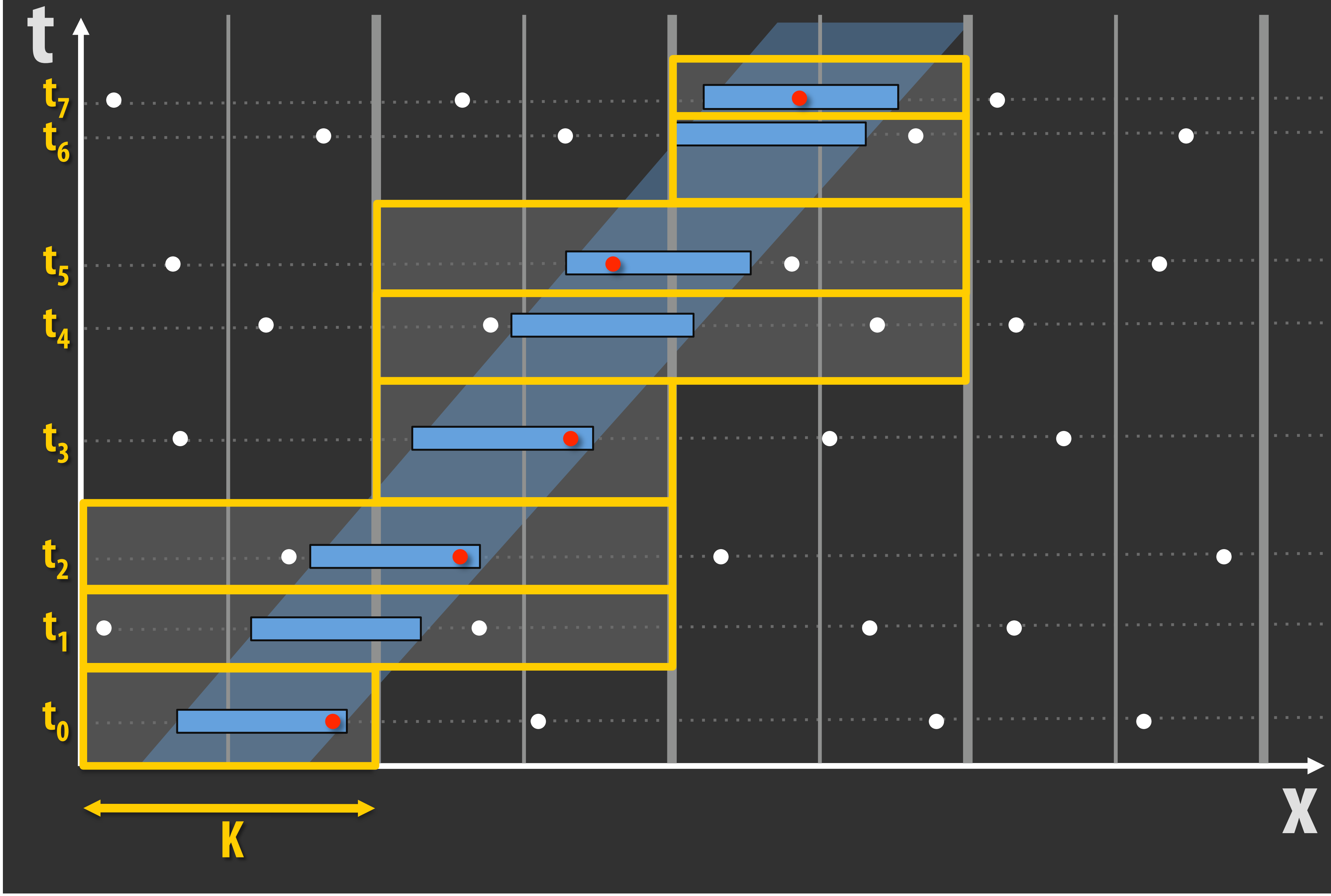
- Limit the number of unique times (or lens positions) used to sample coverage

Interleaved sampling

[Mitchell 91]
[Keller 01]



INTERLEAVE

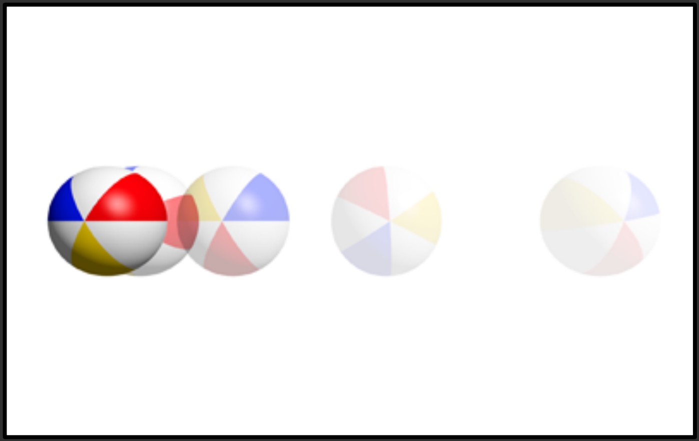


INTERLEAVE parallelism

```
        For each MP
------------------------------------------------------------
Setup      ...
------------------------------------------------------------
          For each unique time T                   PARALLEL
Bound        Position MP at T
             Compute MP bbox at T
------------------------------------------------------------
Test         For each tile in bbox                 UTILIZATION?
               Test MP-sample coverage
```

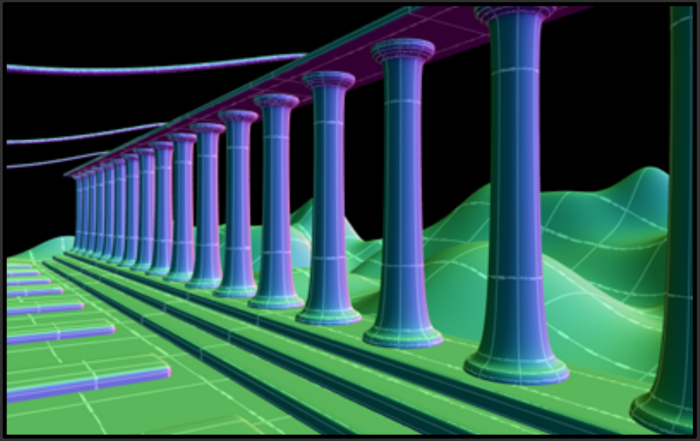
EVALUATION

Test scenes



Ball Roll

Soccer Jump



Columns

Talking

1728 x 1080 resolution, ½-pixel area triangle micropolygons

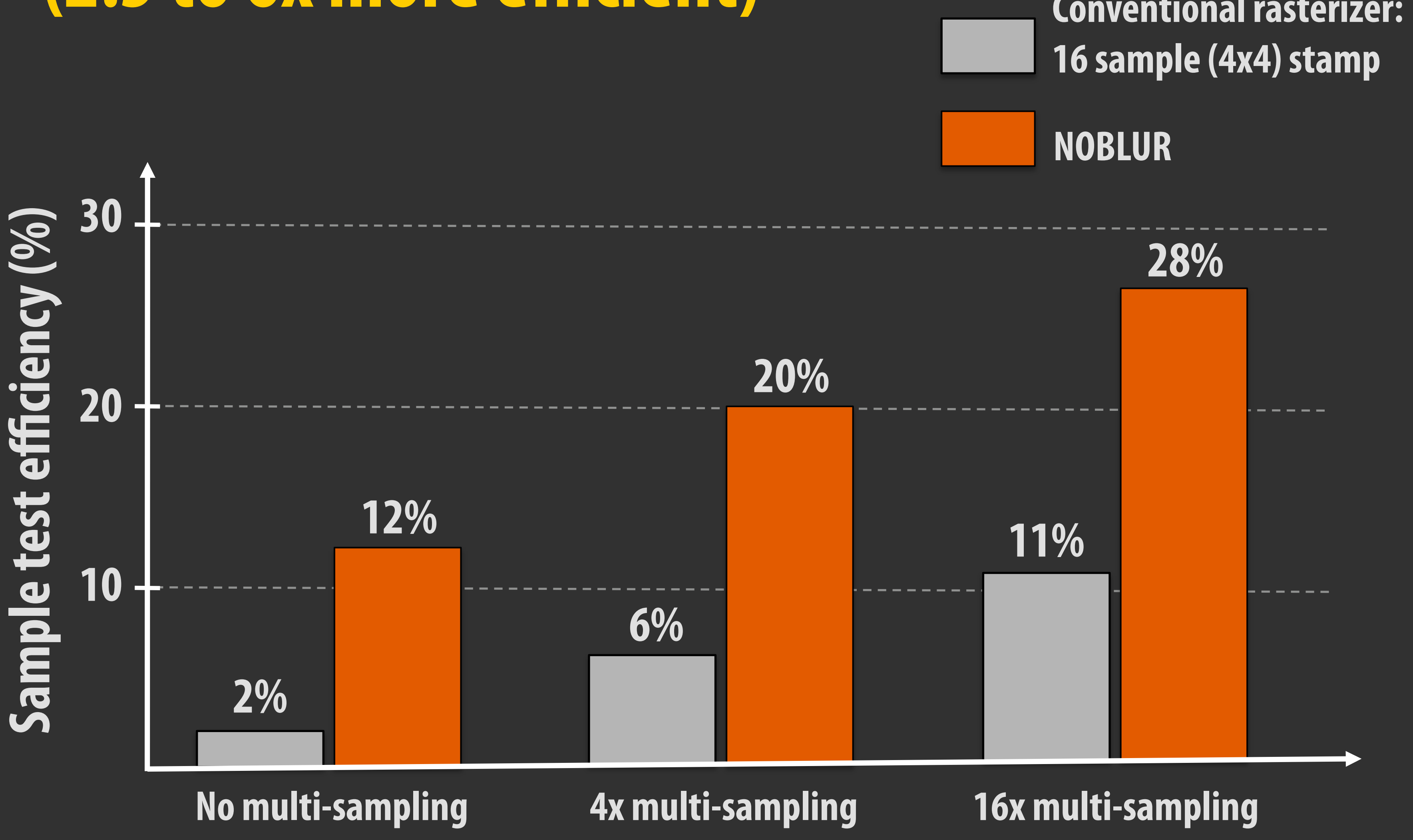
How efficient is NOBLUR?

- What fraction of sample tests generate fragments?

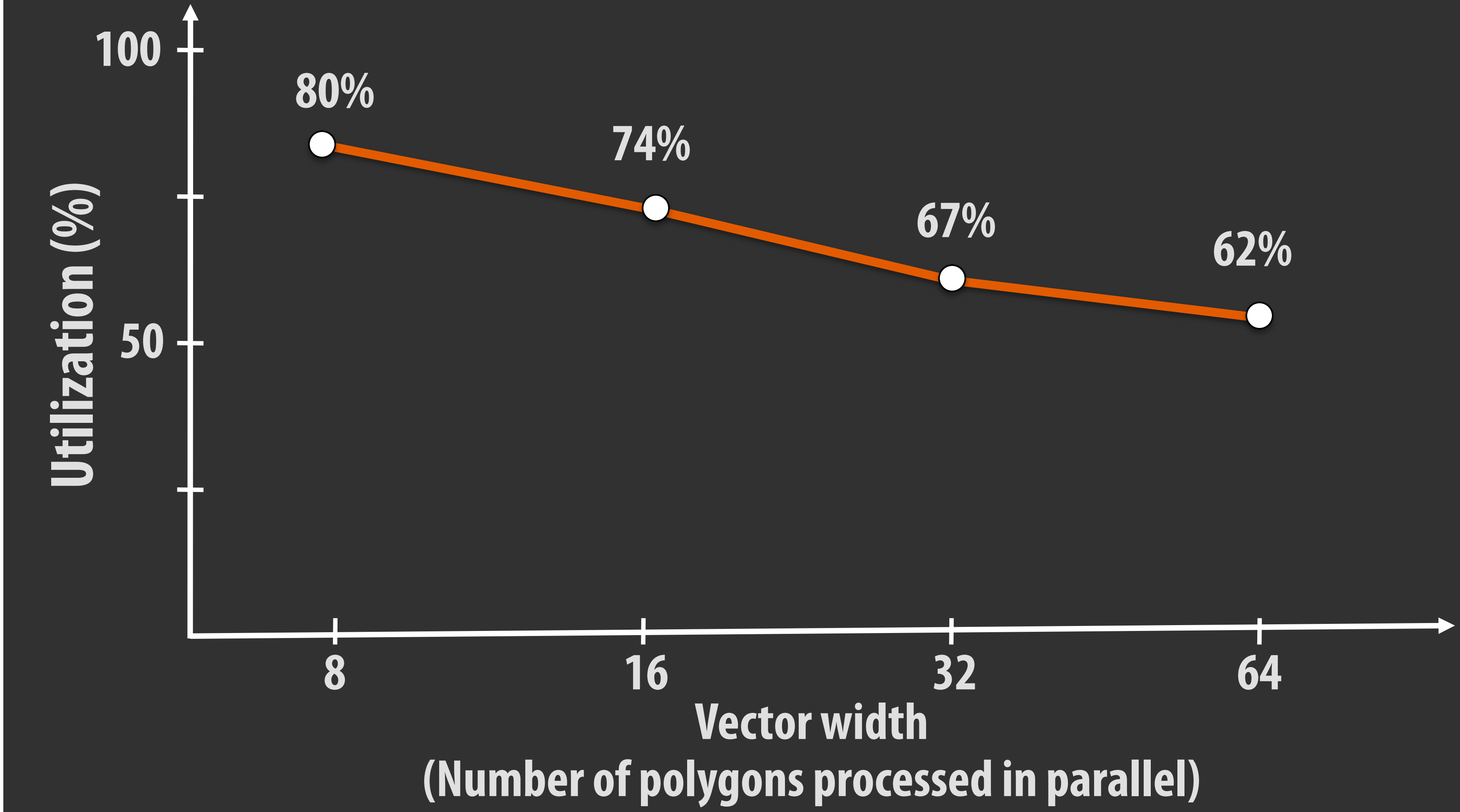
- Does parallelization across polygons efficiently utilize vector processing?

NOBLUR increases sample test efficiency

(2.5 to 6x more efficient)



NOBLUR sustains high vector utilization



Micropolygon rasterization is expensive

Primary visibility computation:

1080p resolution, 30 Hz

4x multi-sampling

Simple scene (10 M micropolygons)

Estimated cost of GPU SW implementation:

Approximately 1/3 of high-end GPU

Fixed-function micropolygon rasterization is appealing

- How much do motion blur and camera defocus cost?
- What is relative performance of INTERVAL, INTERLEAVE under varying amounts of motion or defocus?

Soccer jump

16x multi-sampling

INTERVAL:	16 time intervals
INTERLEAVE:	64 unique times

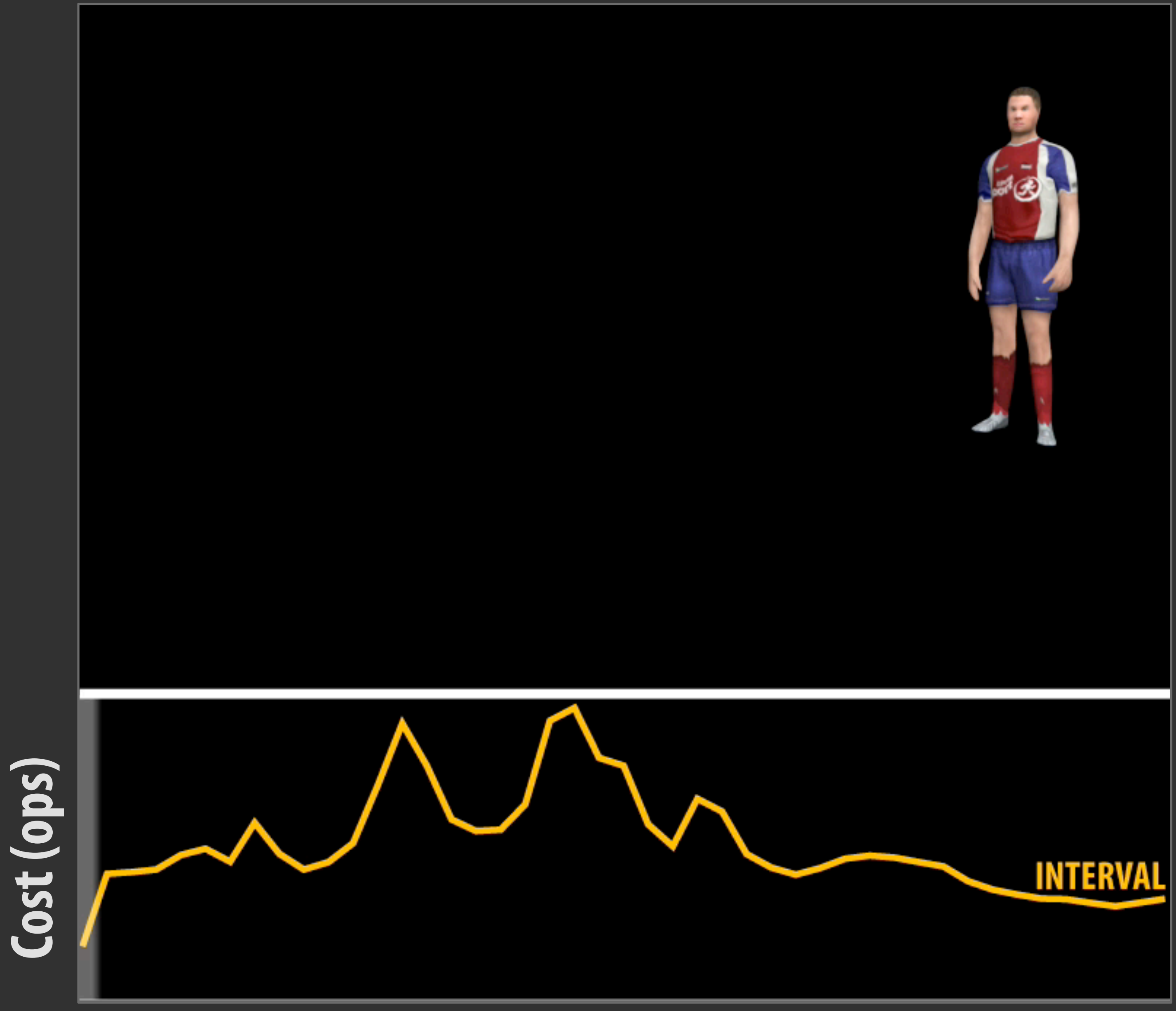
Enabling motion/defocus blur costs 3 to 7x more

- Point-in-polygon tests are more expensive
- INTERVAL, INTERLEAVE perform more tests than NOBLUR

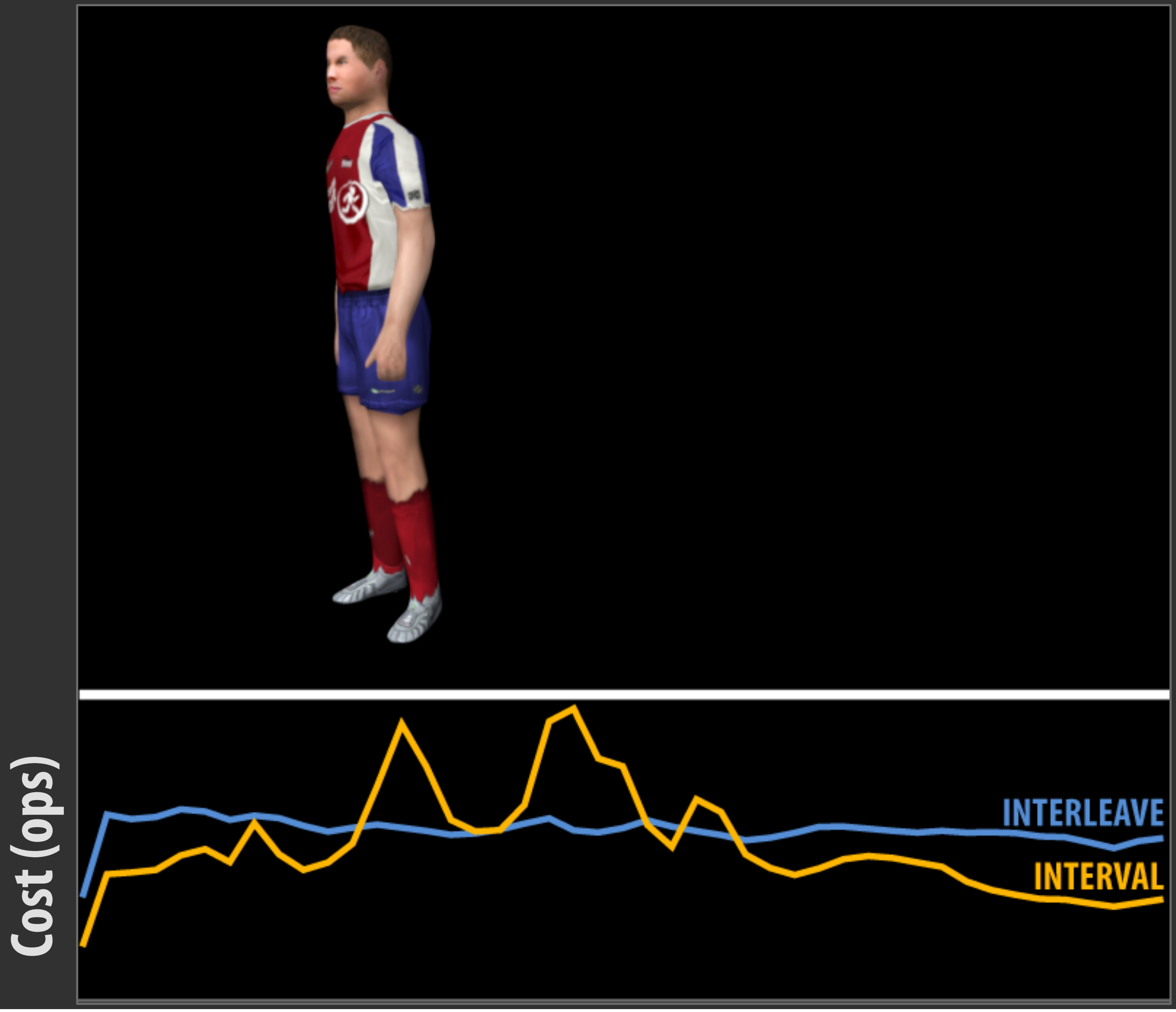
Sample test efficiency
(stationary geometry, perfect focus) →

NOBLUR	28%
INTERVAL	11%
INTERLEAVE	5%

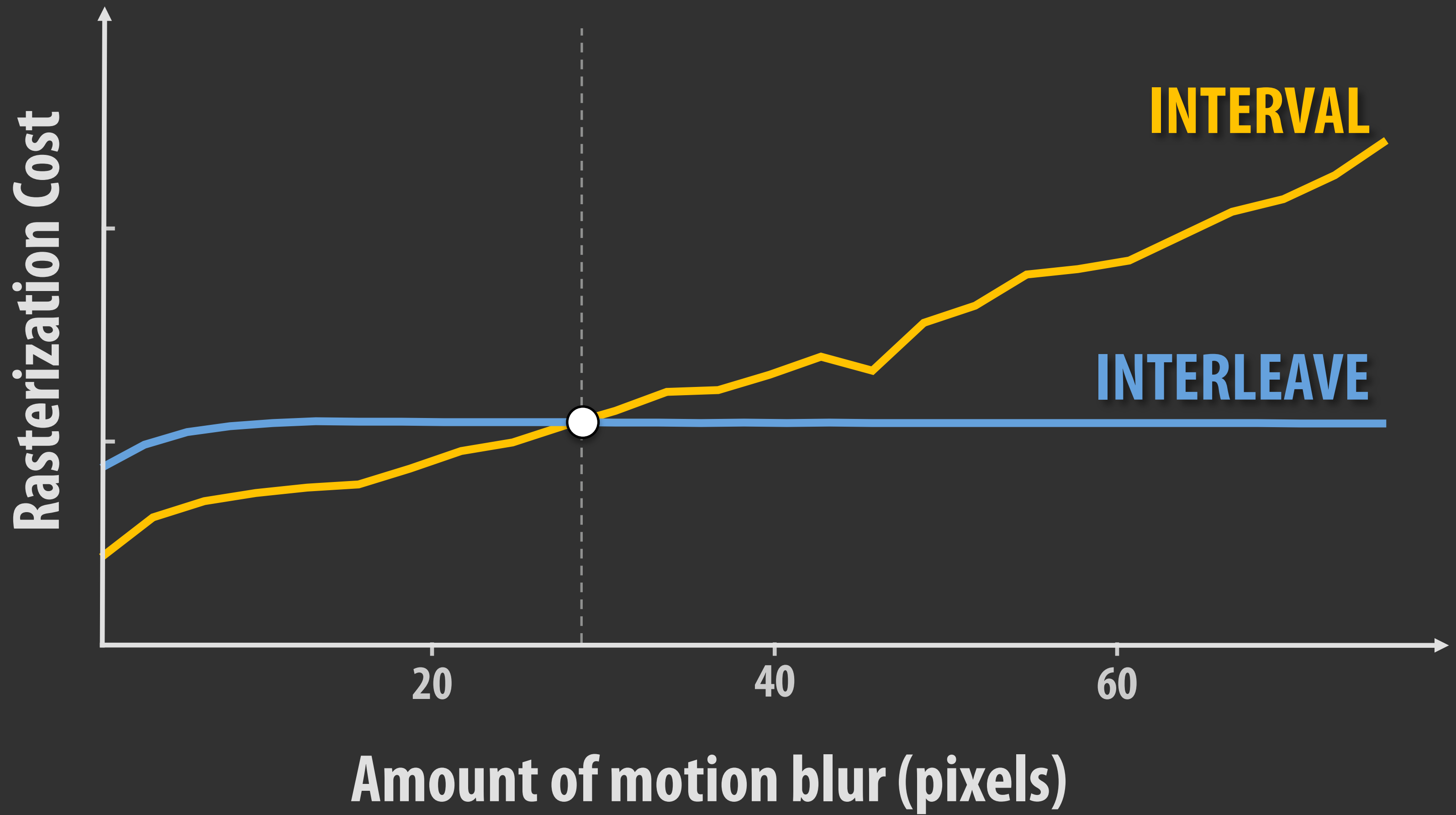
INTERVAL's performance varies with motion



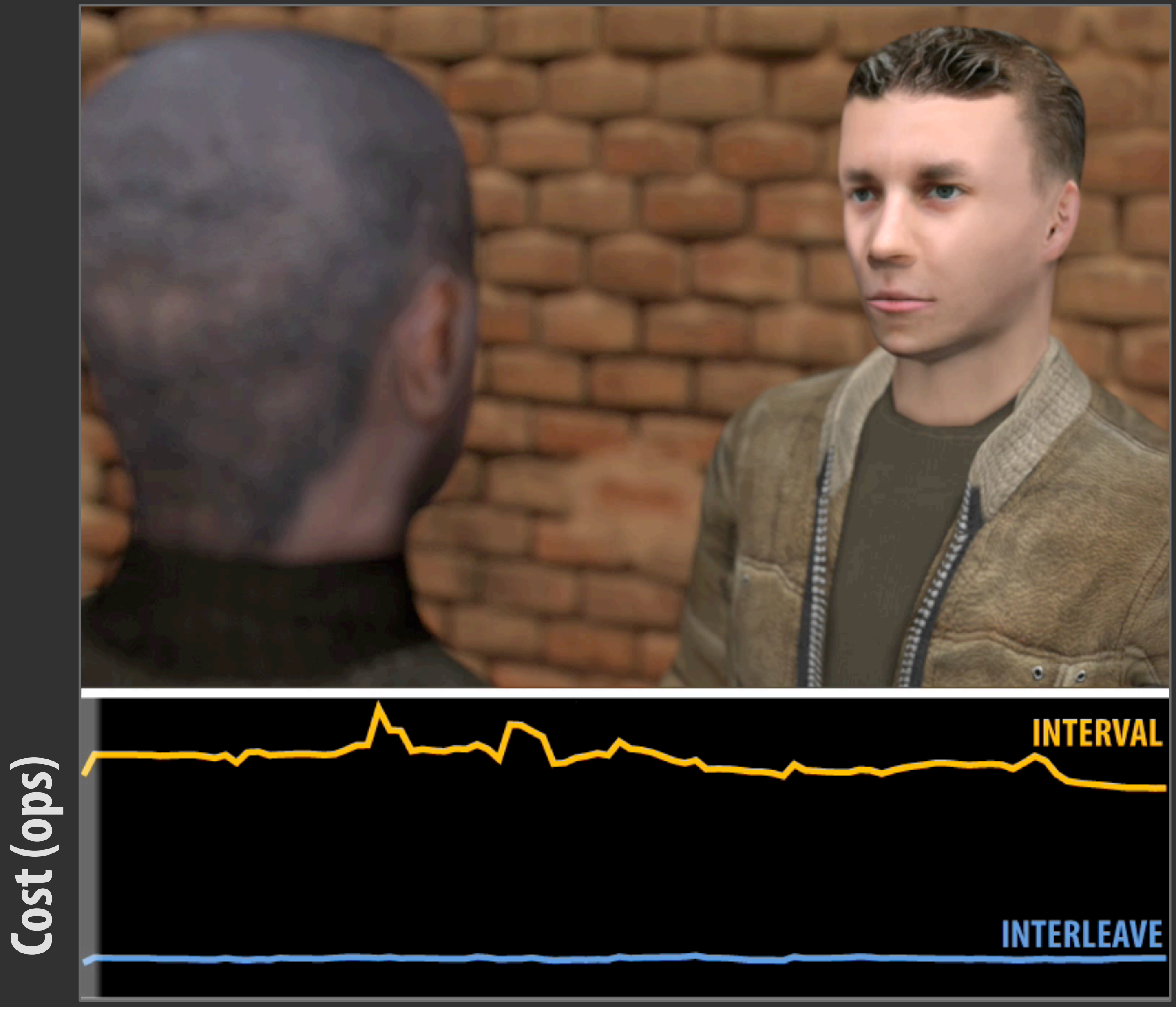
INTERLEAVE more efficient than INTERVAL at high motion



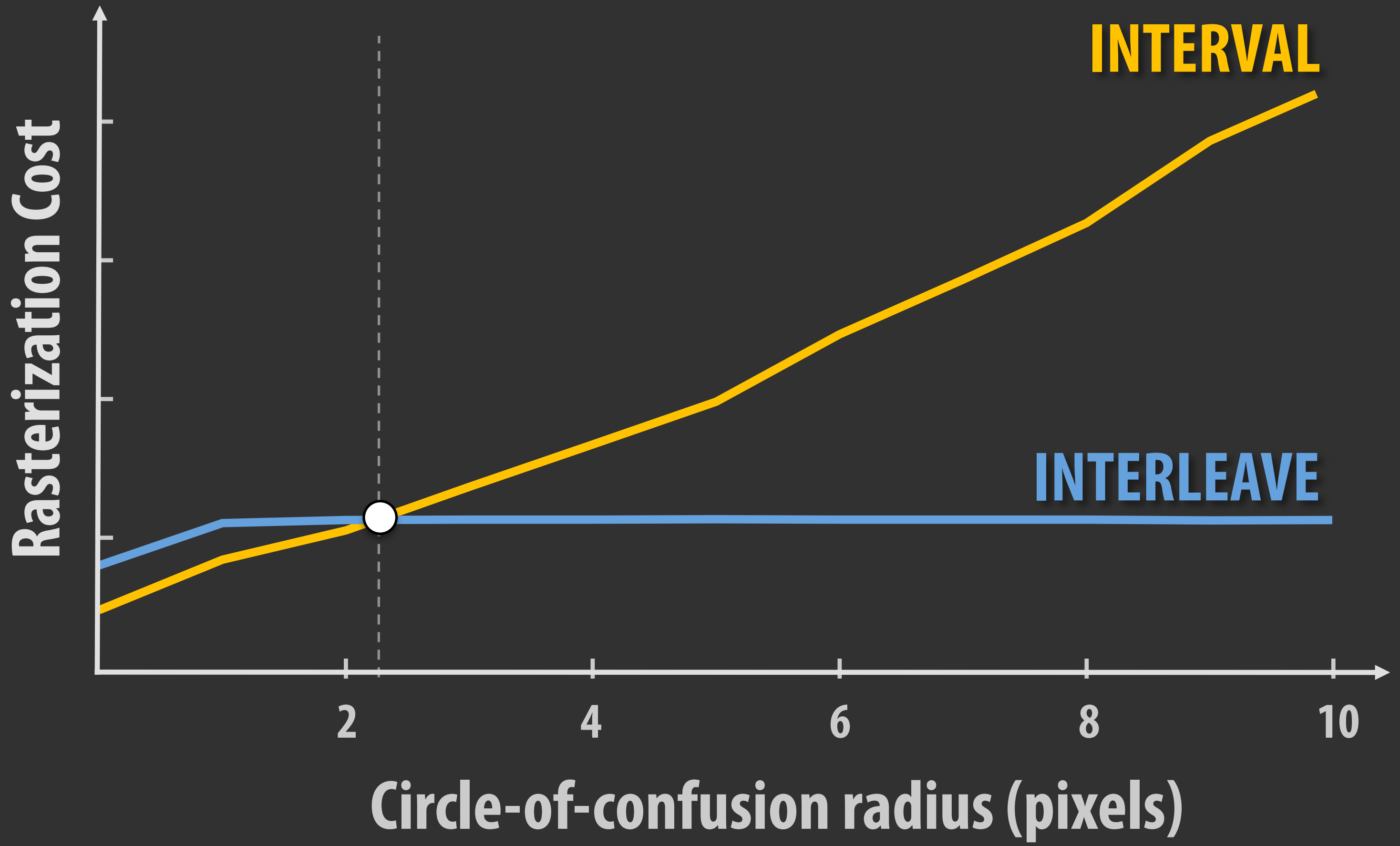
~30 pixels of motion blur equates performance



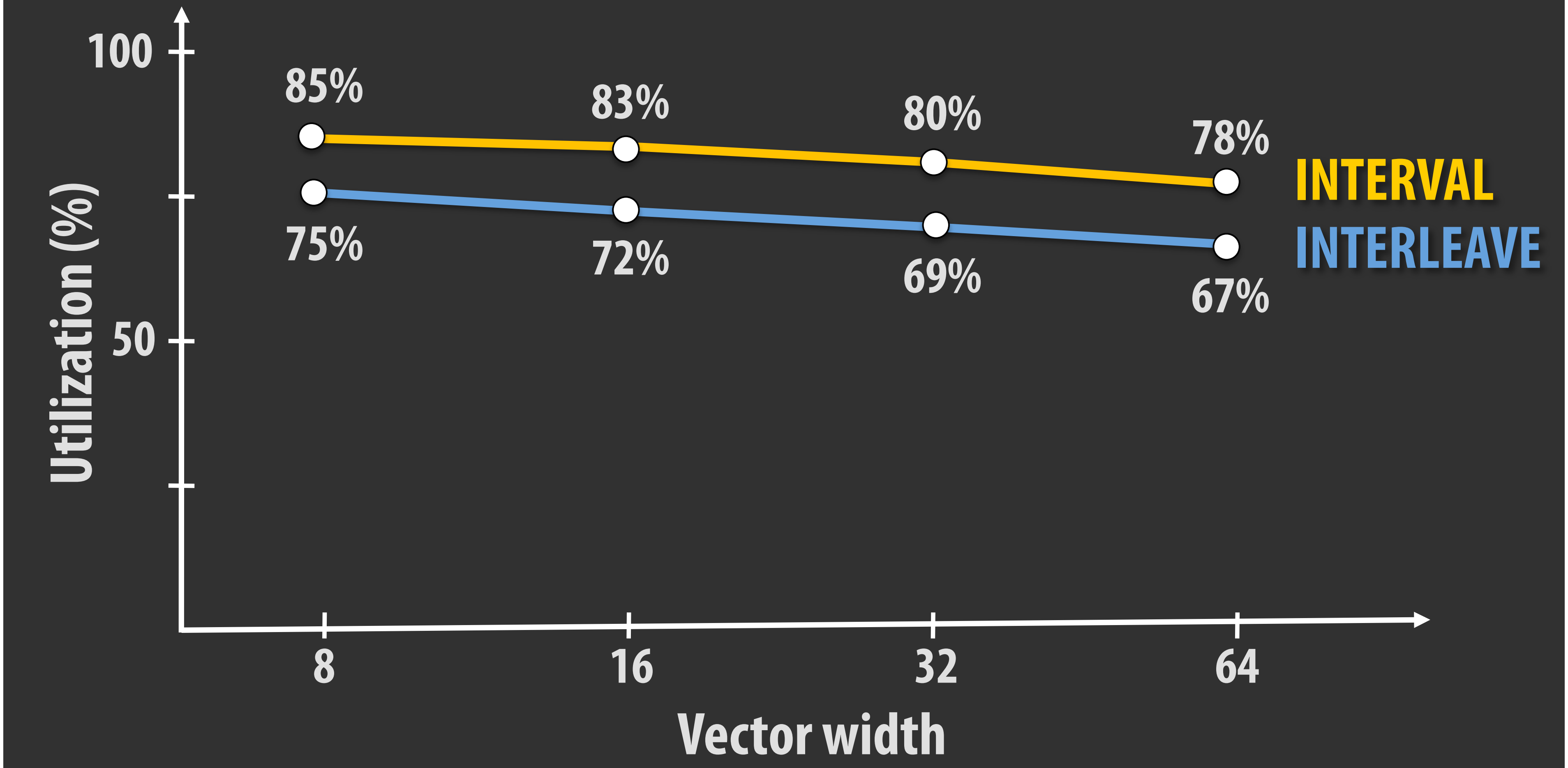
INTERVAL's costs increase sharply with defocus



~2 pixel defocus blur radius equates performance



INTERVAL/INTERLEAVE sustain high utilization



SUMMARY

Re-optimizing rasterization: NOBLUR

- Parallelize across micropolygons
- More efficient than conventional rasterization techniques
  - Especially at low sampling rates
- Utilizes wide vector processing well

- Even with these improvements, micropolygon rasterization is expensive

Extension to motion blur / defocus

- Costs 3 to 7x more in flops
- INTERVAL more efficient until motion is large
- INTERLEAVE more efficient under high motion, moderate to high defocus
- Both algorithms are inefficient
  - Only 1 in 20 polygon-sample tests generate hits

How does real-time graphics pipeline evolve to enable <u>efficient</u> micropolygon rendering?

How should surfaces be tessellated into micropolygons?

How can micropolygons be rasterized efficiently?

How is occlusion-culling best implemented?

Should the pipeline shade like GPUs or like REYES?

Special thanks to

Intel Foundation Ph.D. Fellowship Program

Intel Larrabee Research Grant Program

National Science Foundation Graduate Research Program

Stanford Pervasive Parallelism Lab
(SUN, AMD, NVIDIA, IBM, Intel, HP, NEC)

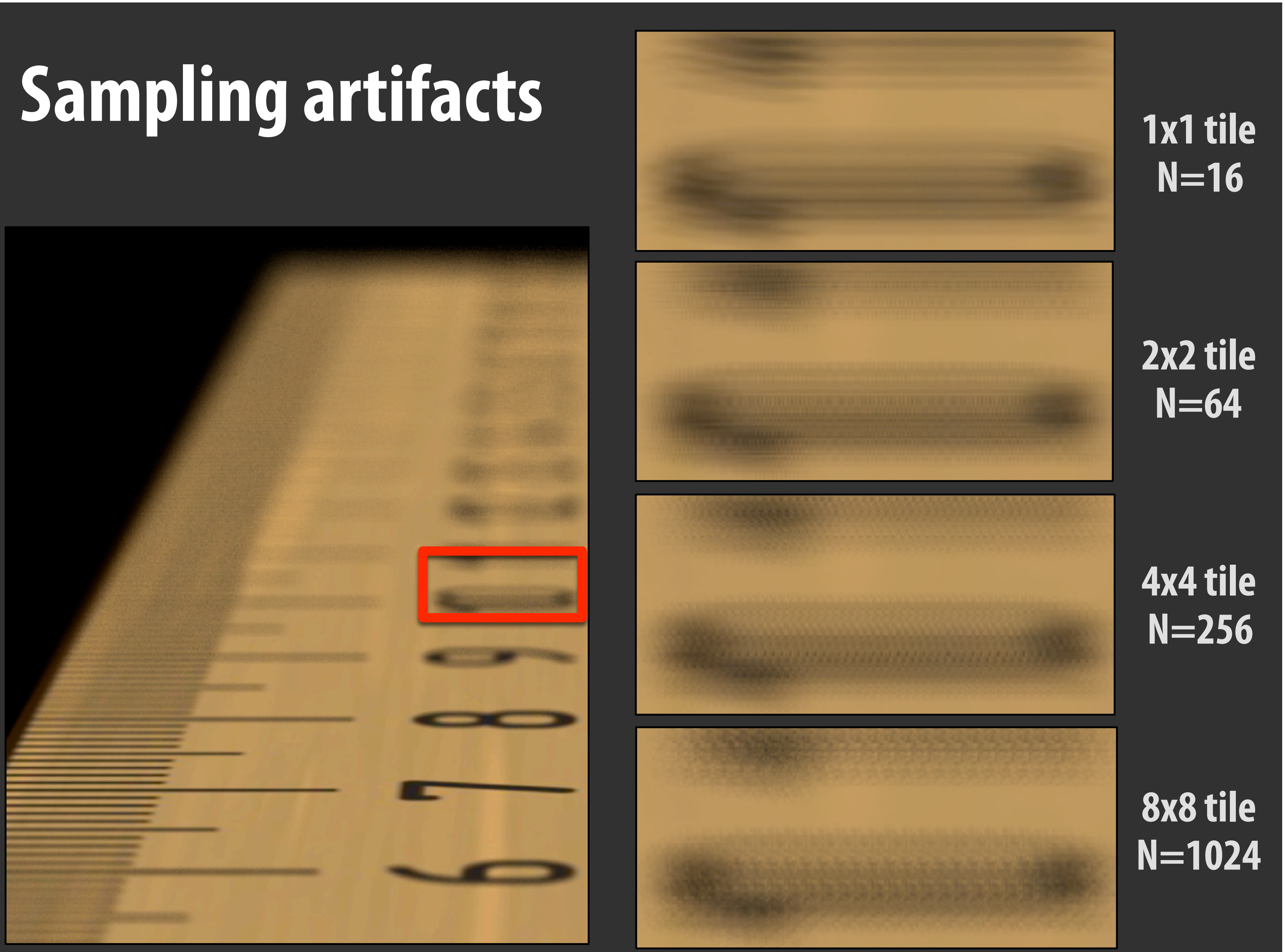
Mike Doggett, Lund University

Mike Houston, AMD

Tim Purcell, NVIDIA

ADDITIONAL SLIDES

Sampling artifacts

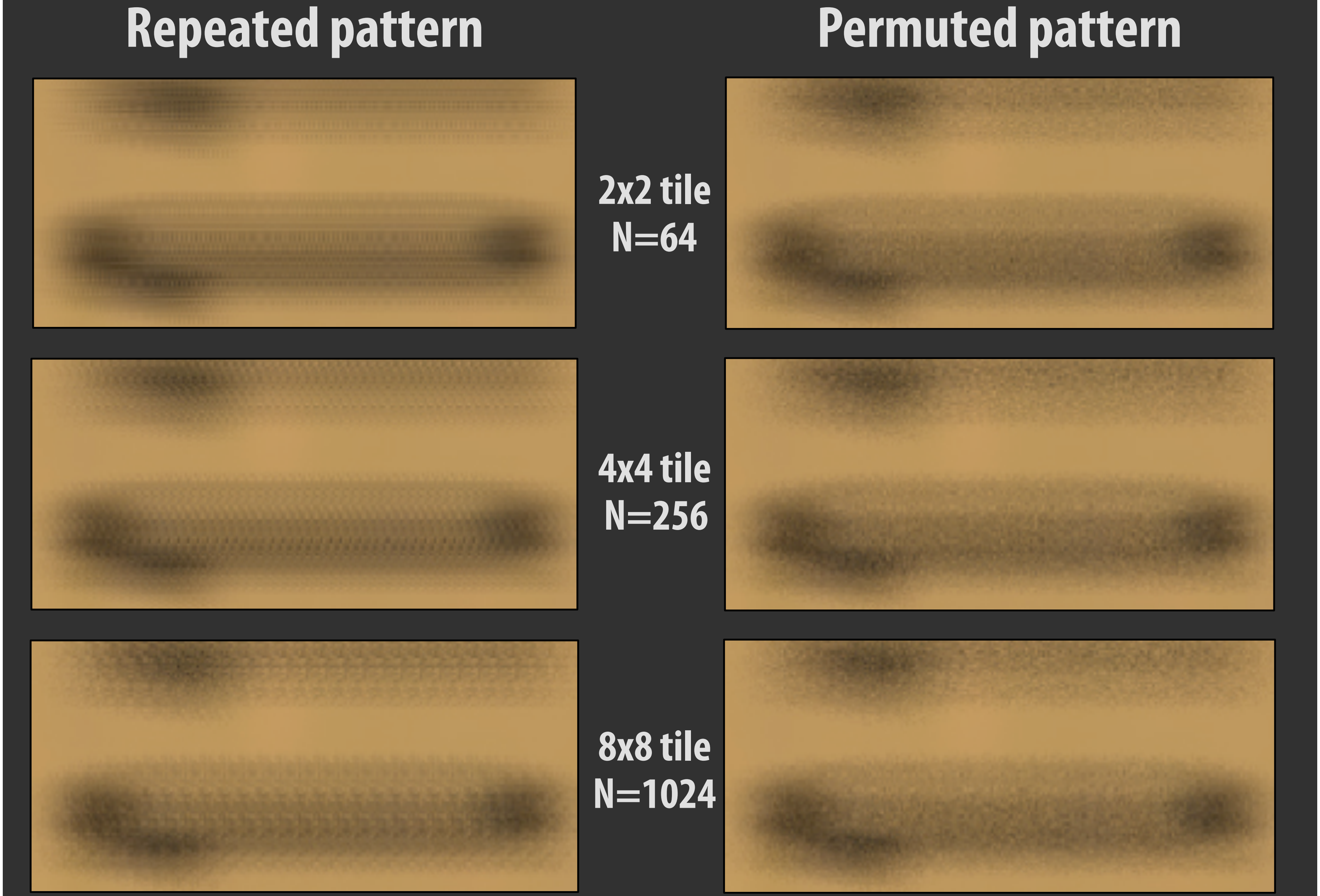


1x1 tile
N=16

2x2 tile
N=64

4x4 tile
N=256

8x8 tile
N=1024



Repeated pattern

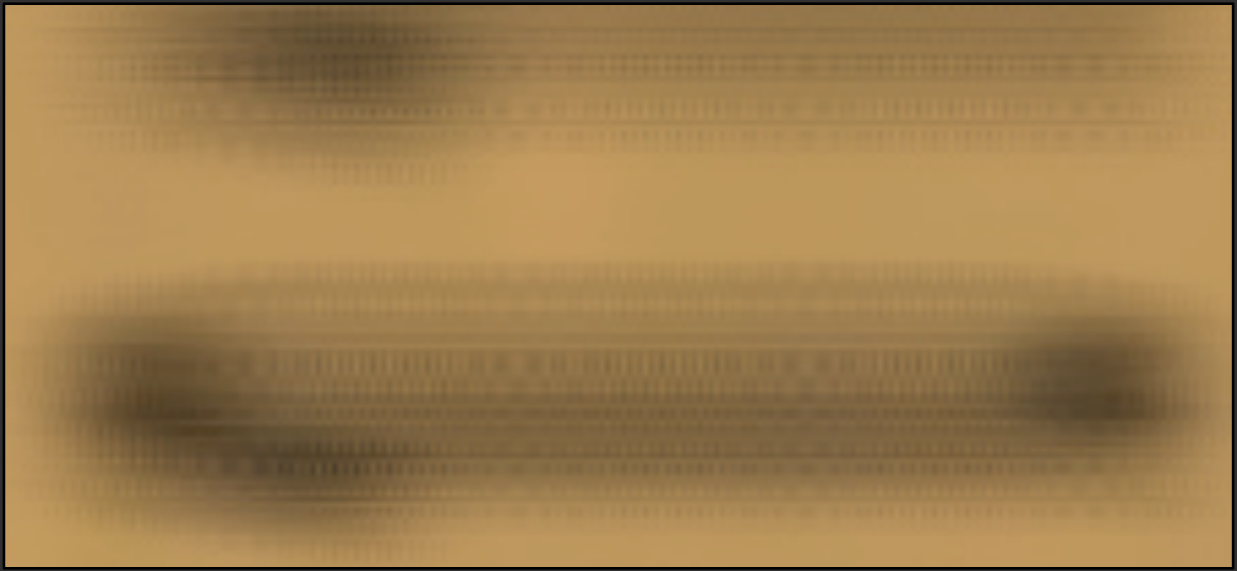
Permuted pattern

2x2 tile
N=64

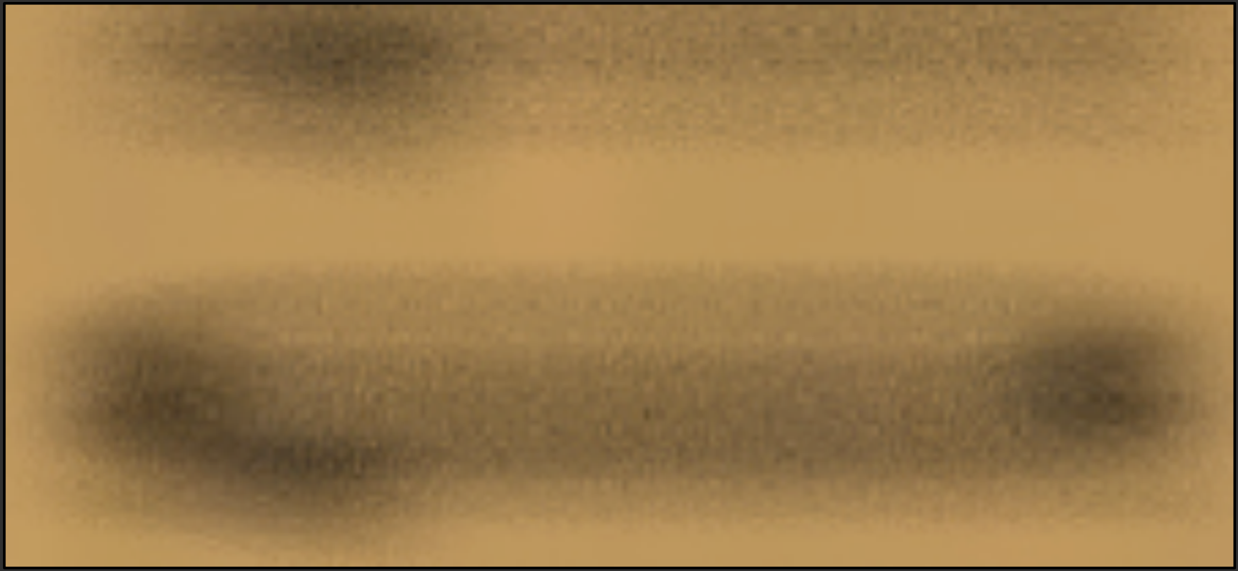
4x4 tile
N=256

8x8 tile
N=1024

INTERLEAVE 2x2 tile, N=64



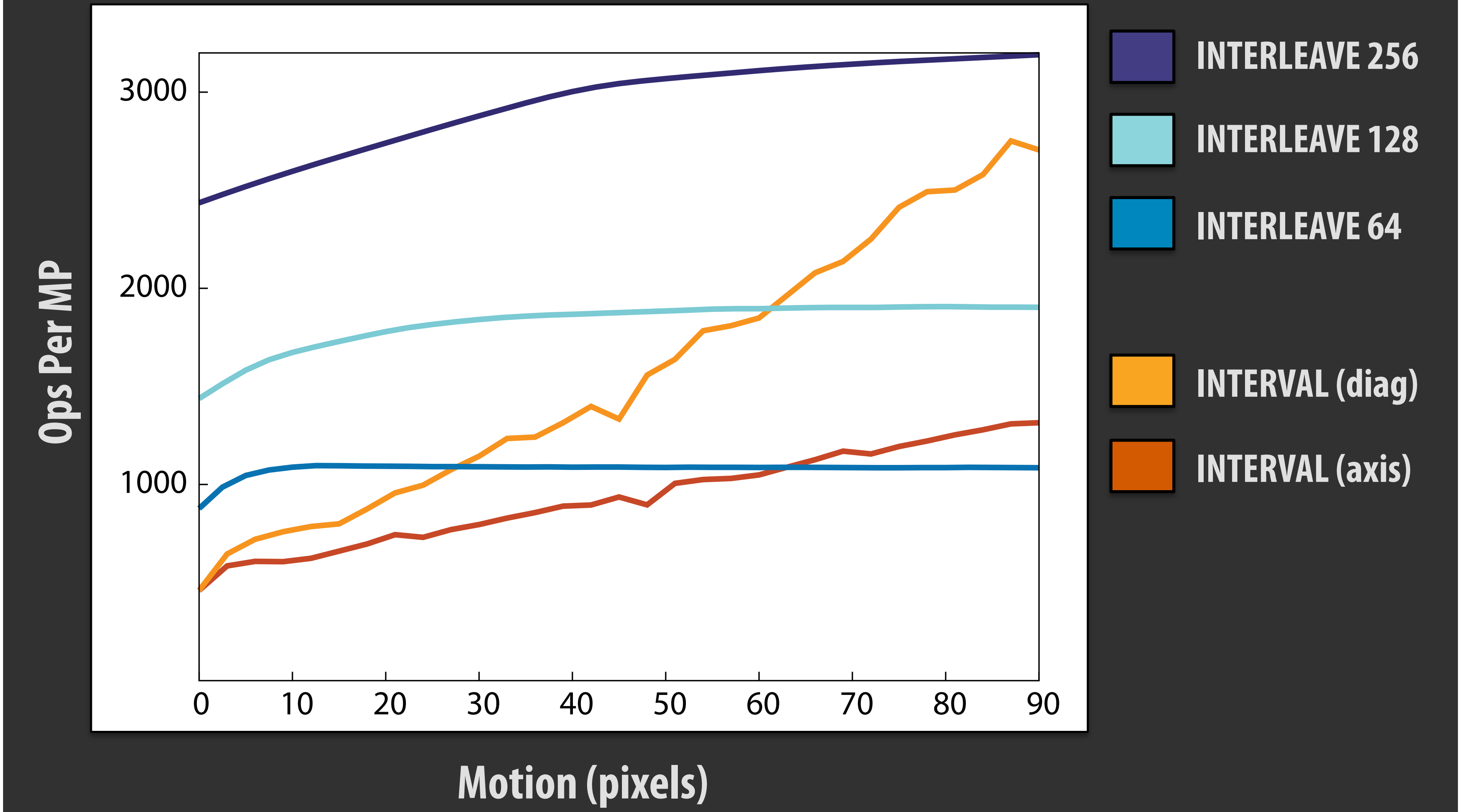
Repeated pattern



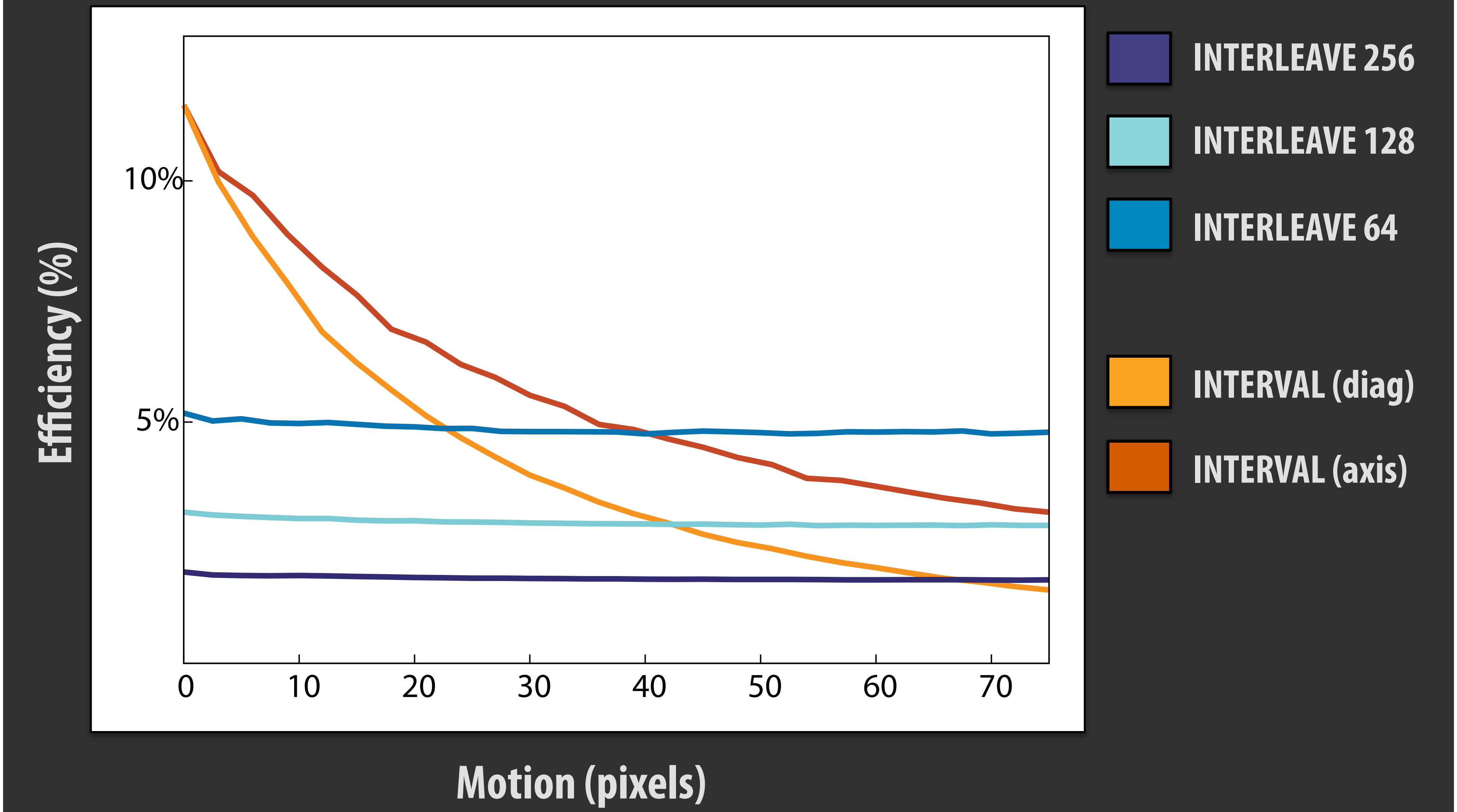
Permuted pattern

Motion blur

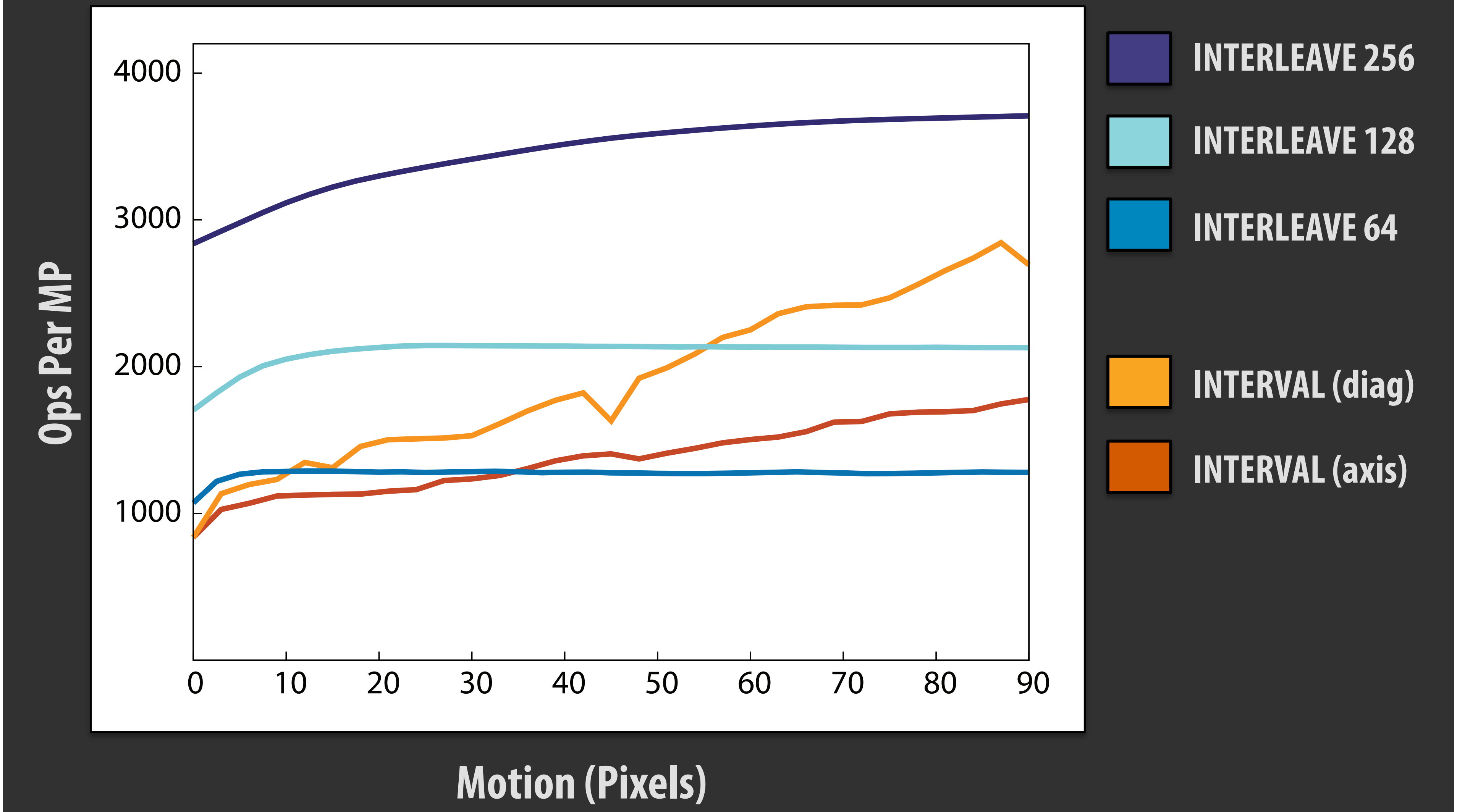
16x MSAA: motion blur



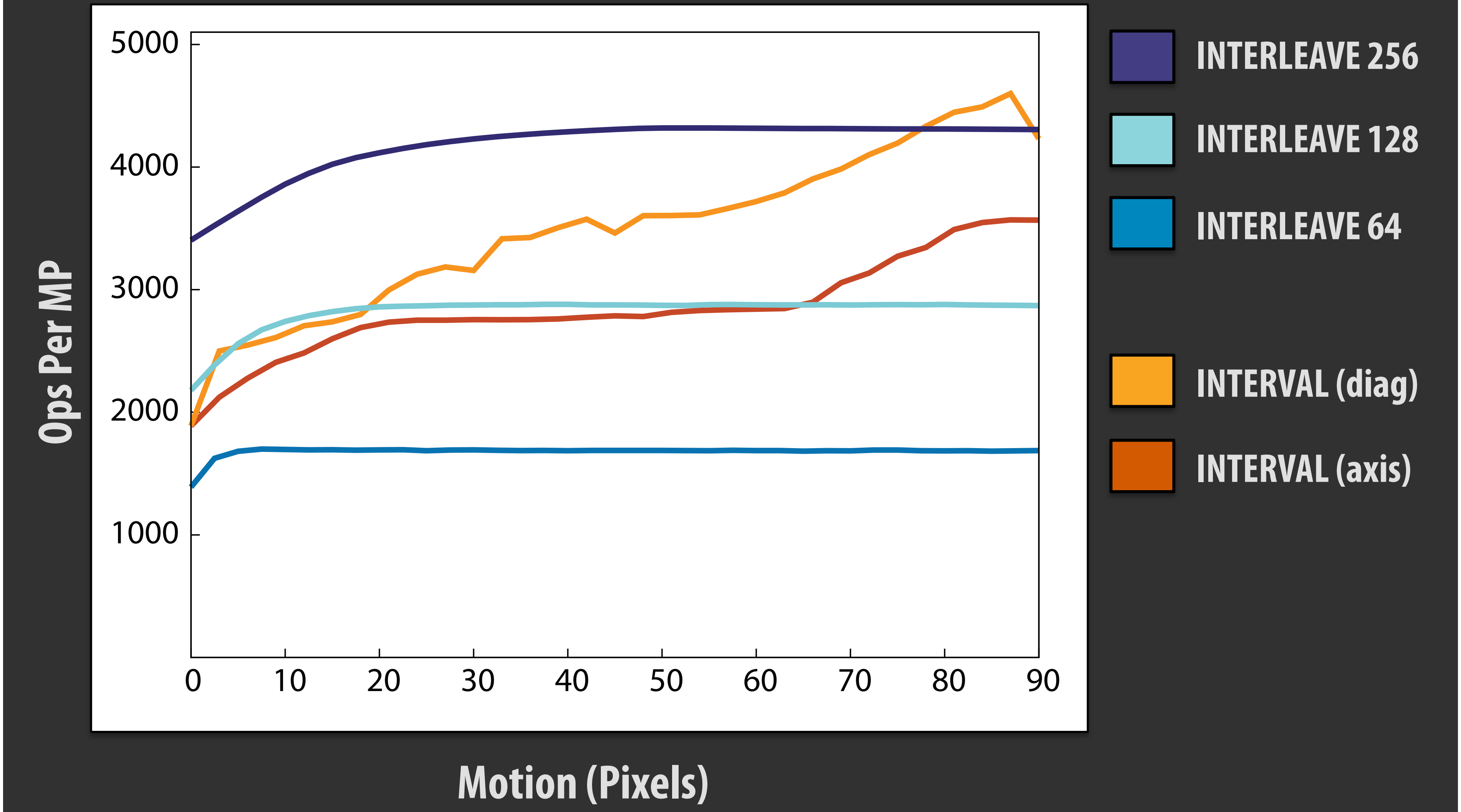
16x MSAA: motion blur (efficiency)



32x MSAA: motion blur

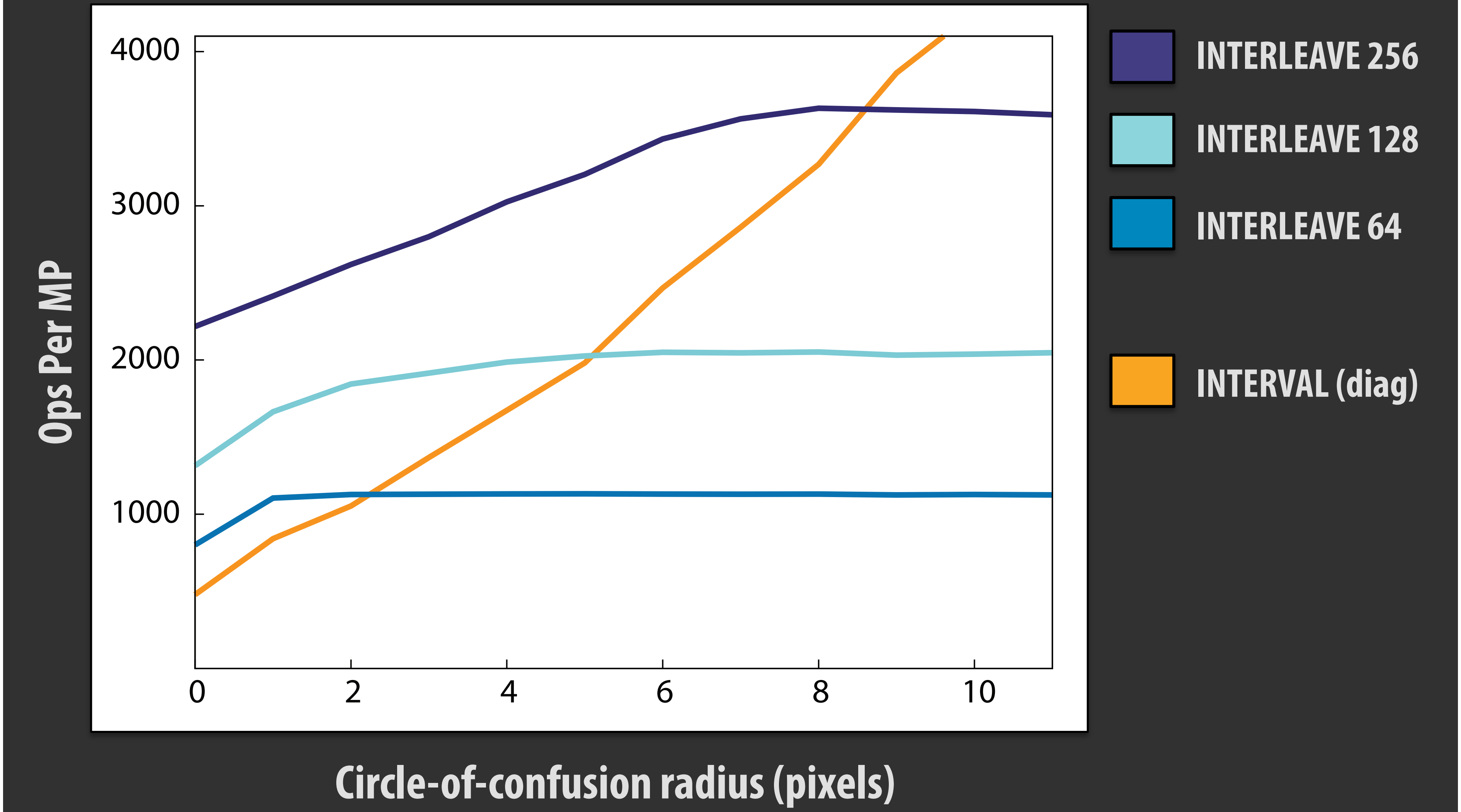


64x MSAA: motion blur

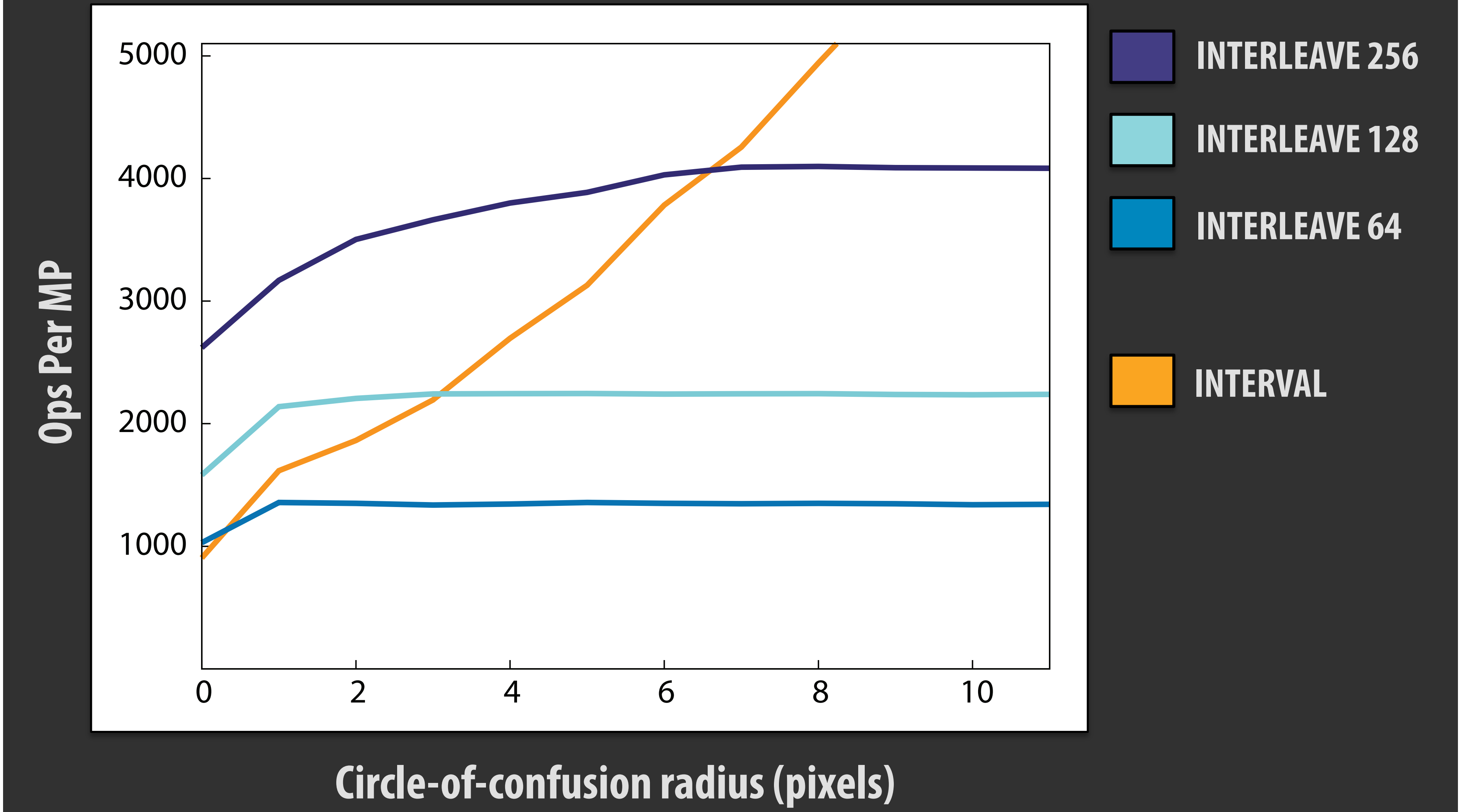


Defocus blur

16x MSAA: defocus blur



32x MSAA: defocus blur


Ops Per MP
Circle-of-confusion radius (pixels)
INTERLEAVE 256
INTERLEAVE 128
INTERLEAVE 64
INTERVAL