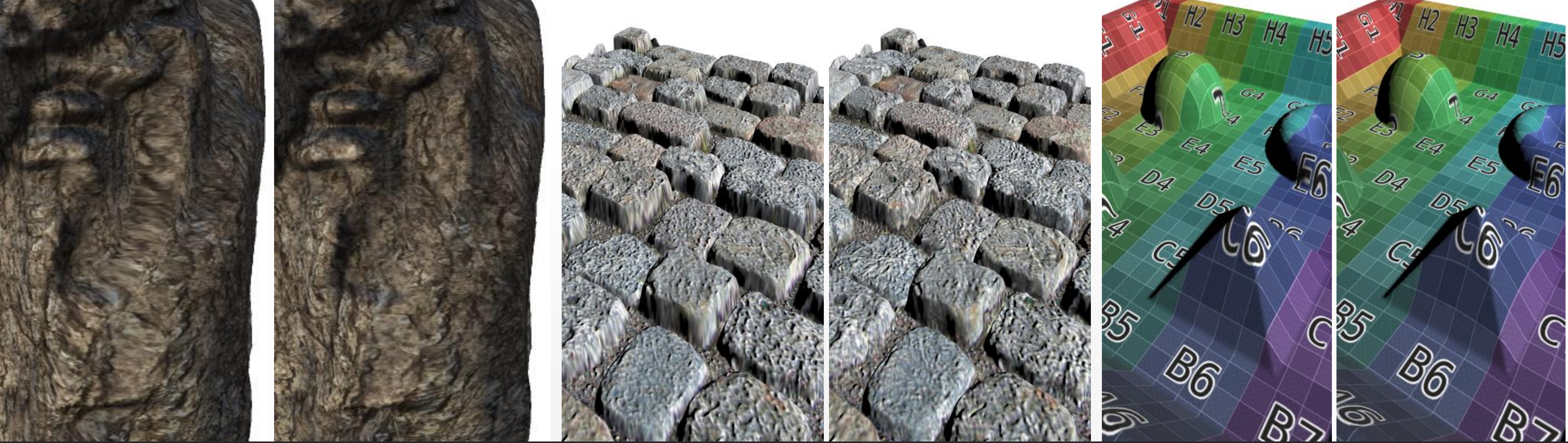


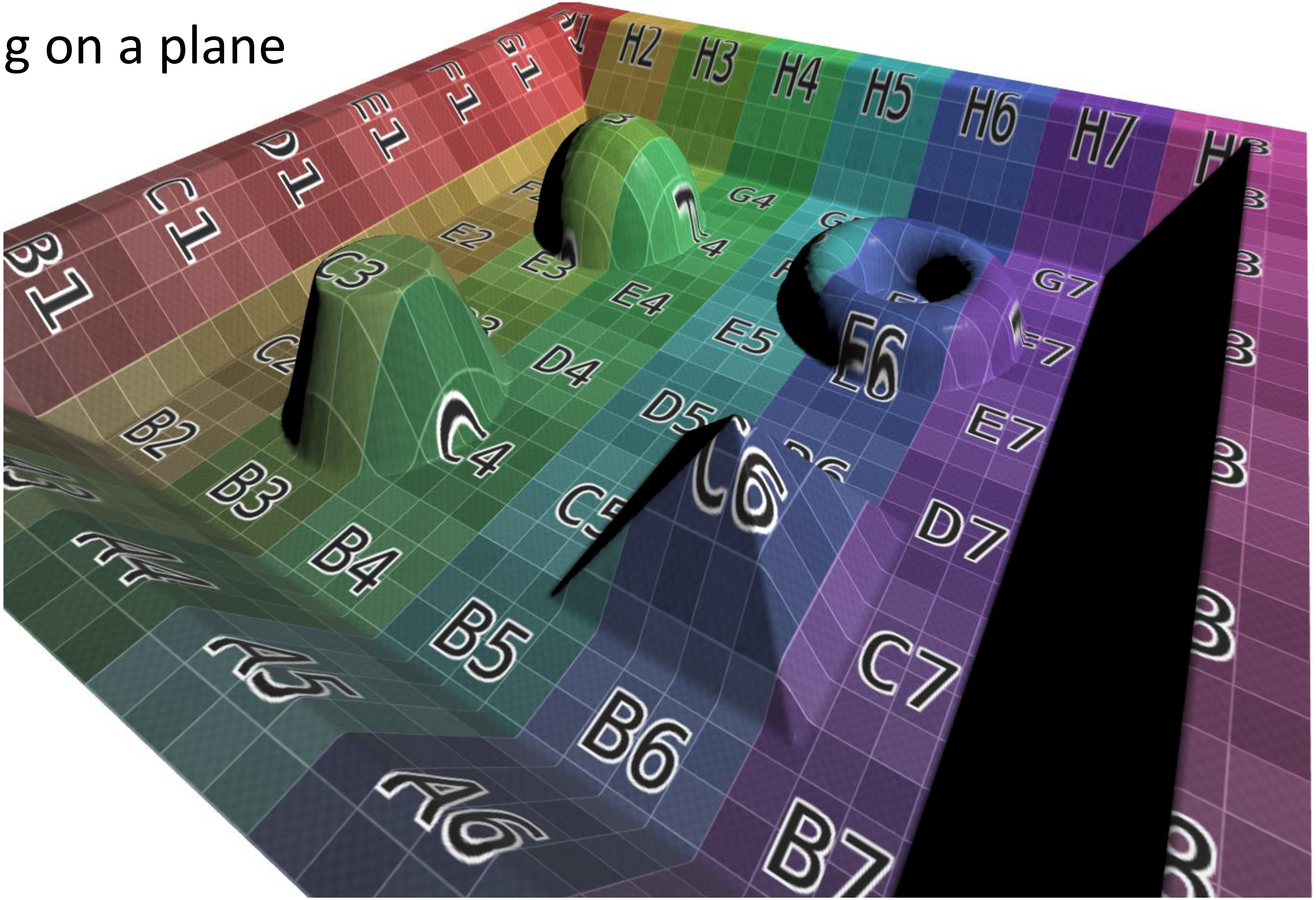
Distortion-Free Displacement Mapping

Tobias Zirr [1] and Tobias Ritschel [2]

[1] Karlsruhe Institute of Technology
[2] University College London

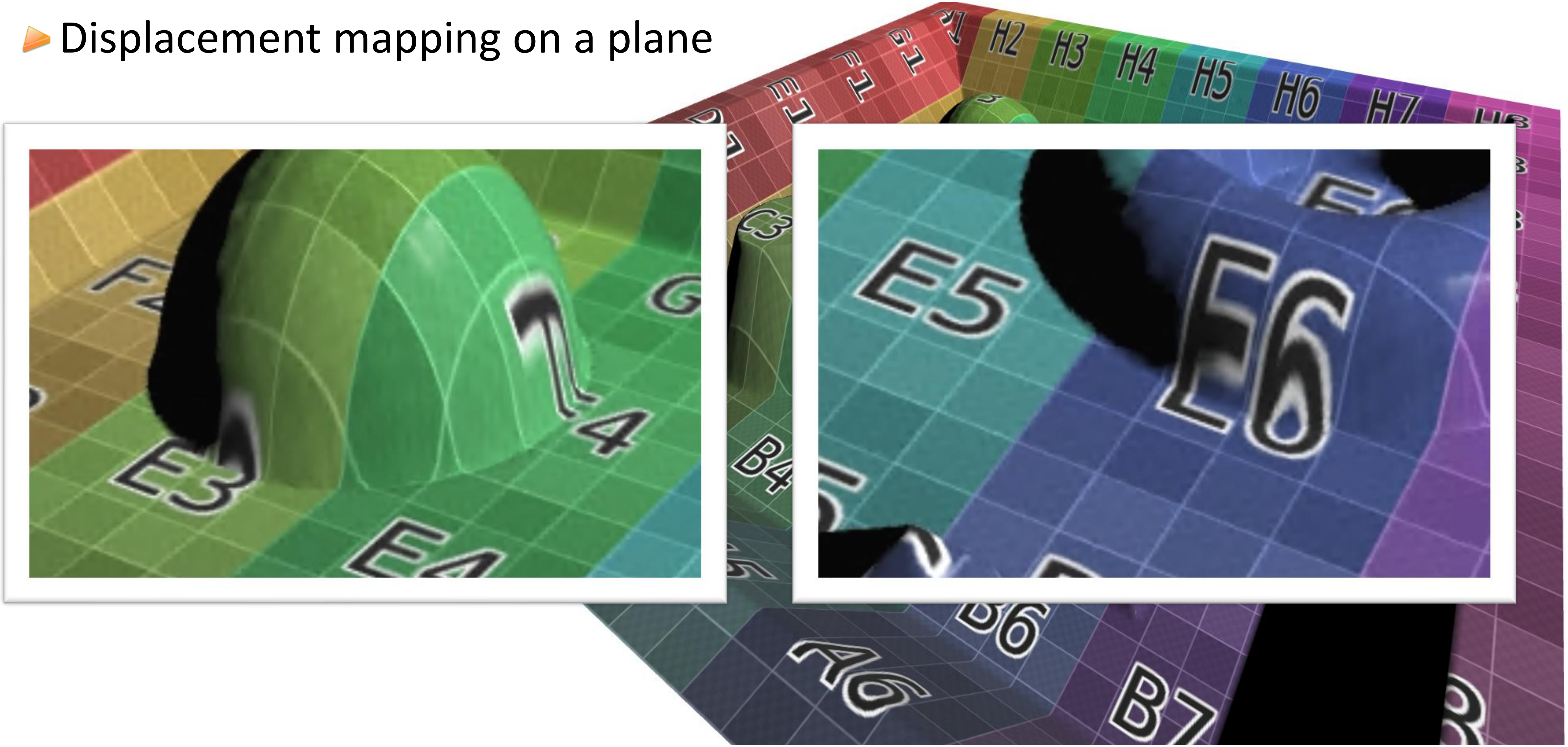
Displacement Mapping causes stretching

- Displacement mapping on a plane



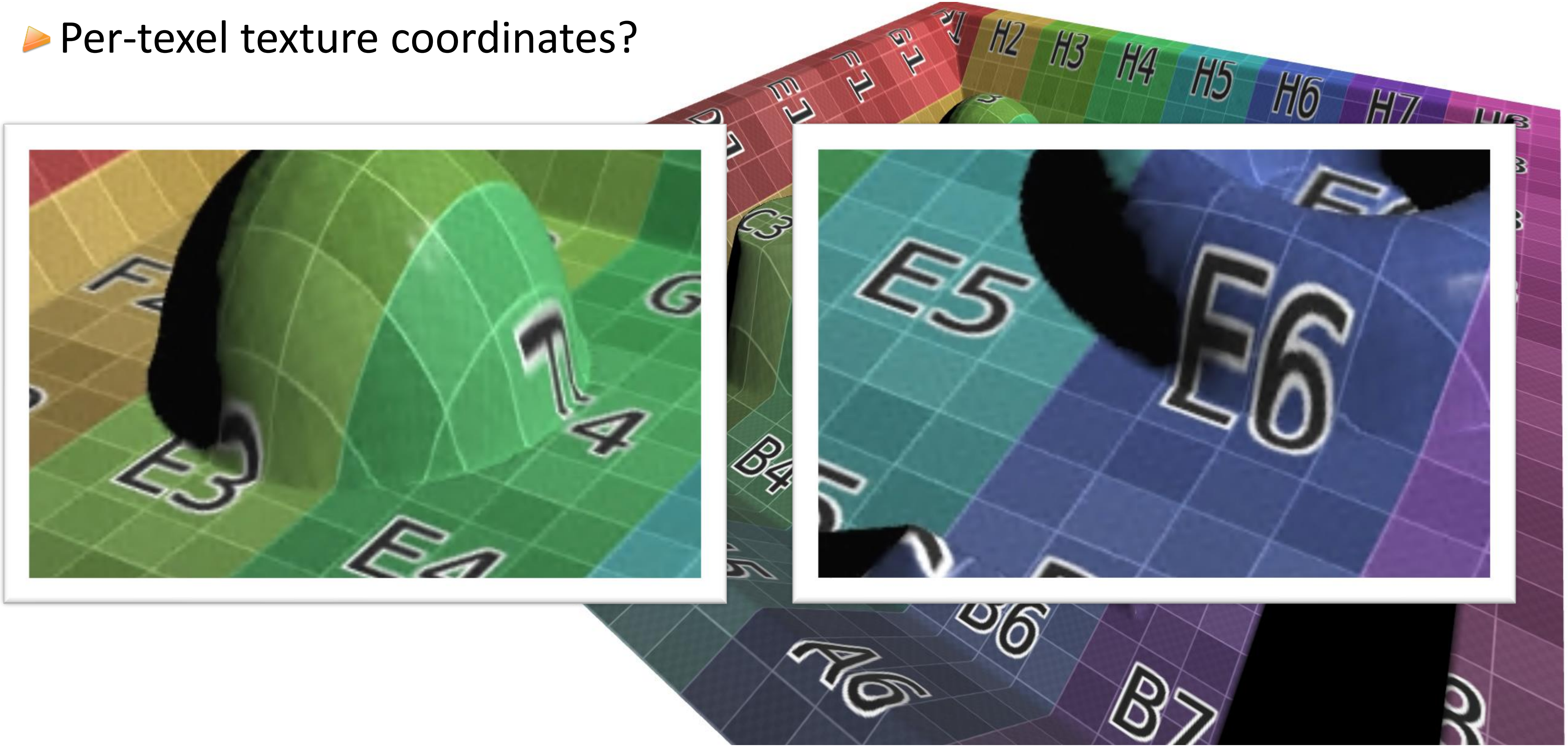
Displacement Mapping causes stretching

- Displacement mapping on a plane



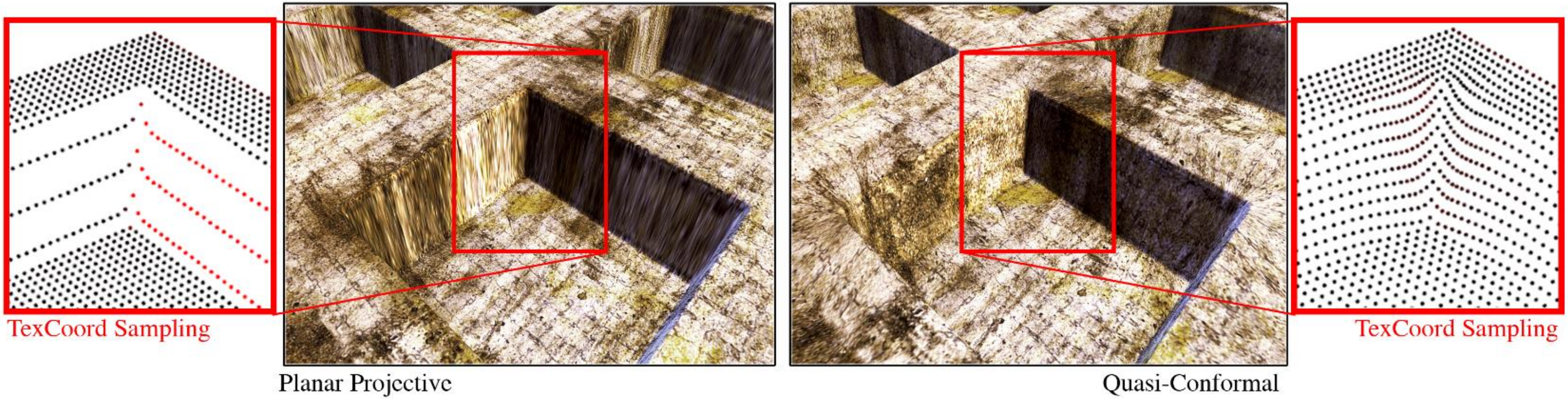
Can we have proper texture unwrapping?

- Per-texel texture coordinates?



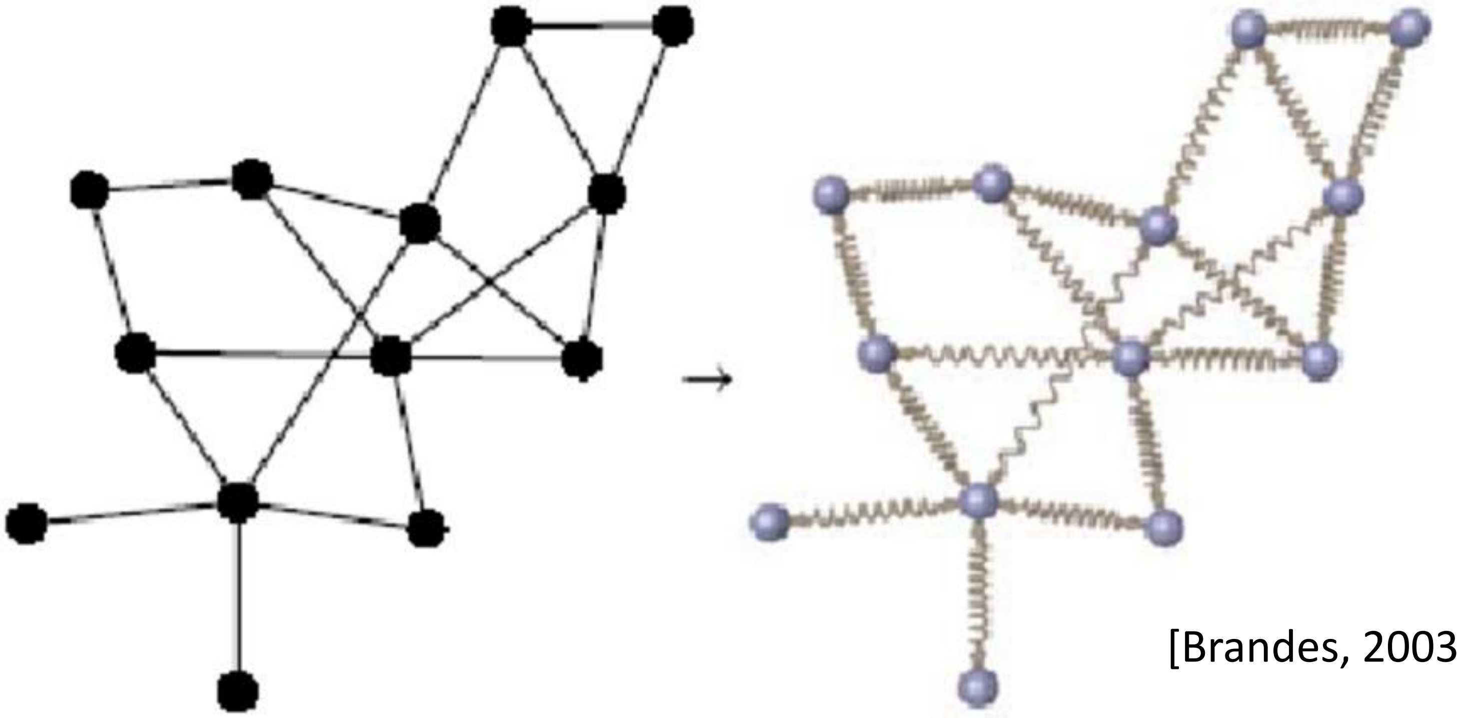
Previous Work

- **“Indirection Mapping for Quasi-Conformal Relief Texturing”** [McGuire and Whitson, 2008]
  - Spring relaxation on mesh vertices (Iterative, Hierarchical)
  - Resampling of texture coordinates



Optimization by Physical Analogies?

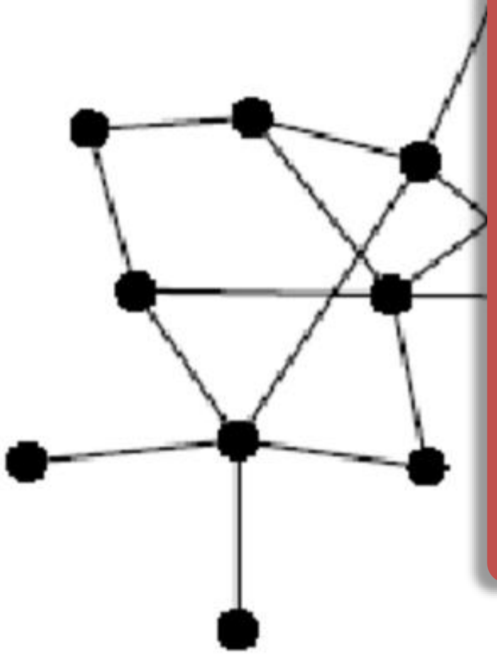
- Optimization by physical models *can* give good results
  - Previous work [McGuire et al., 2008]
  - Also in graph embedding:



[Brandes, 2003]

Optimization by Physical Analogies?

- Optimization by p d results
  - Previous work [I
  - Also in graph em



[Brandes, 2003]

Problems:

- Unknown step sizes
  - Oscillations?
  - Annealing?!
  - Good convergence speed?!
- Indirectly defined energy
  - Guarantees / objectives?!

Optimization by Energy Minimzation

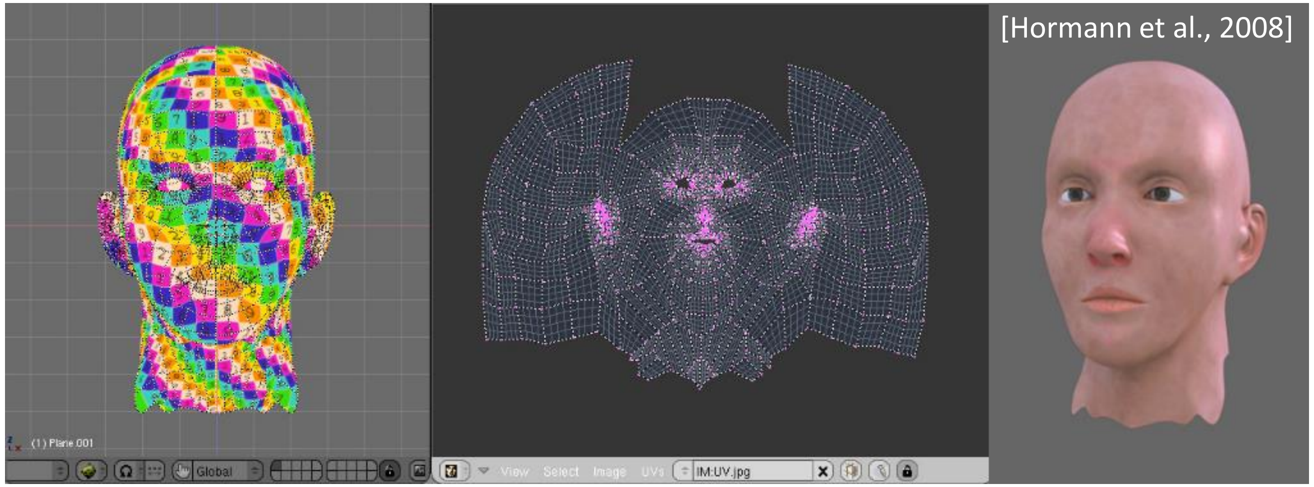
- Energy minimization is more flexible and controllable
  - Directly formulate your problem
    - E.g., metric on per-pixel texture coordinates
    - No resampling required (vs. previous work)

Optimization by Energy Minimzation

- Energy minimization is more flexible and controllable
  - Directly formulate your problem
    - E.g., metric on per-pixel texture coordinates
    - No resampling required (vs. previous work)
  - Flexible constraints
    - Boundaries
    - Artist Intervention
    - Tradeoffs like angle vs. area
    - Avoid invalid results like collapsed triangles

Conformal vs. Isometric Mappings

- Conformal mappings
  - Preserve angles
  - Guaranteed existence for “disks”
    - May require area stretching

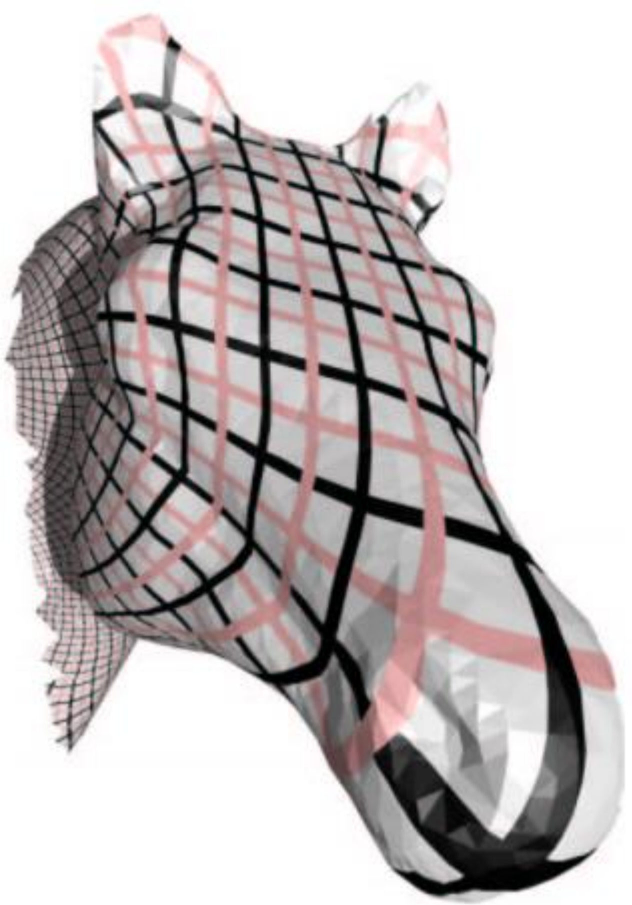


Least-Squares Conformal Mapping (LCSM) in Blender

[Hormann et al., 2008]

Conformal mappings

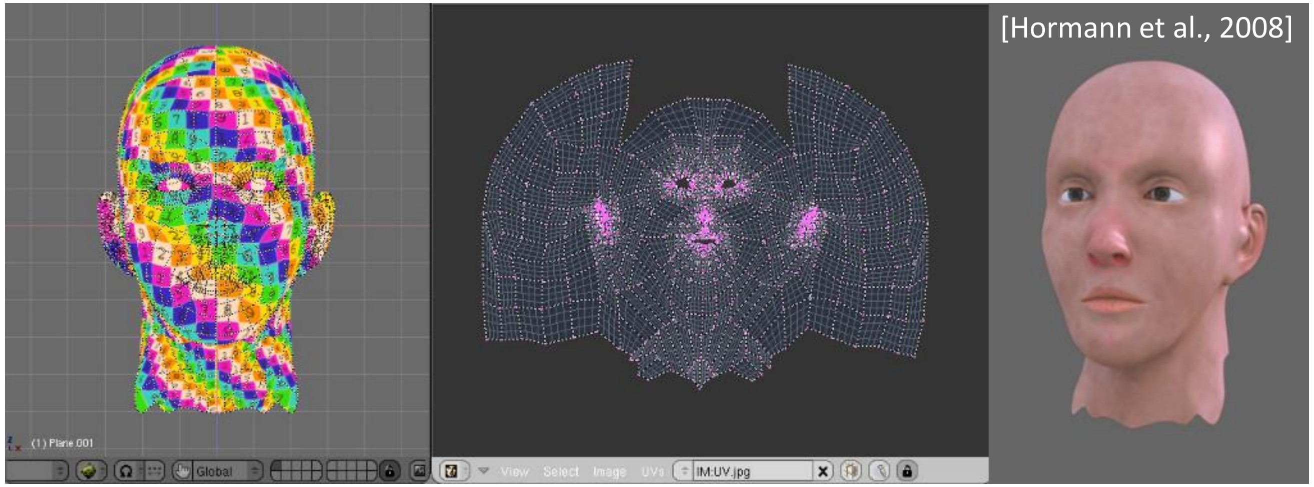
- Angle preservation



Conformal vs. Isometric Mappings

- Conformal mappings
  - Preserve angles
  - Guaranteed existence for “disks”
    - May require area stretching

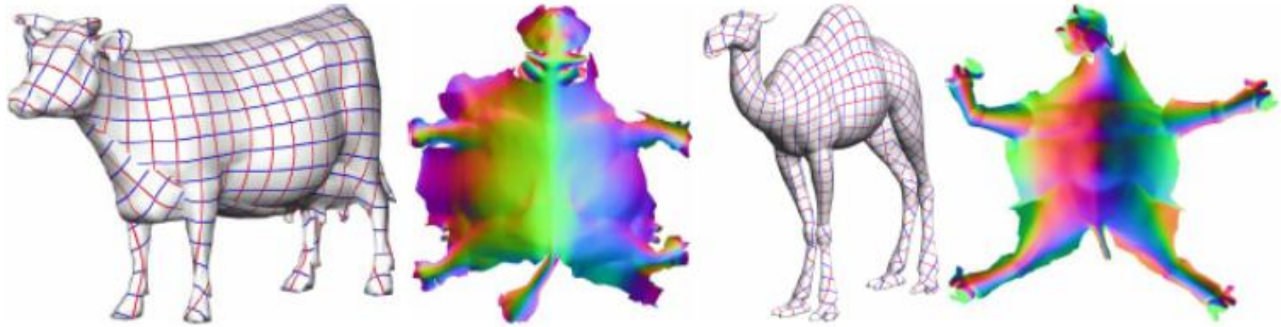
- Isometric mappings
  - Also preserve area
  - No general existence guarantee
    - Approach as closely as possible



[Hormann et al., 2008]

Least-Squares Conformal Mapping (LCSM) in Blender

[Hormann et al., 2008]

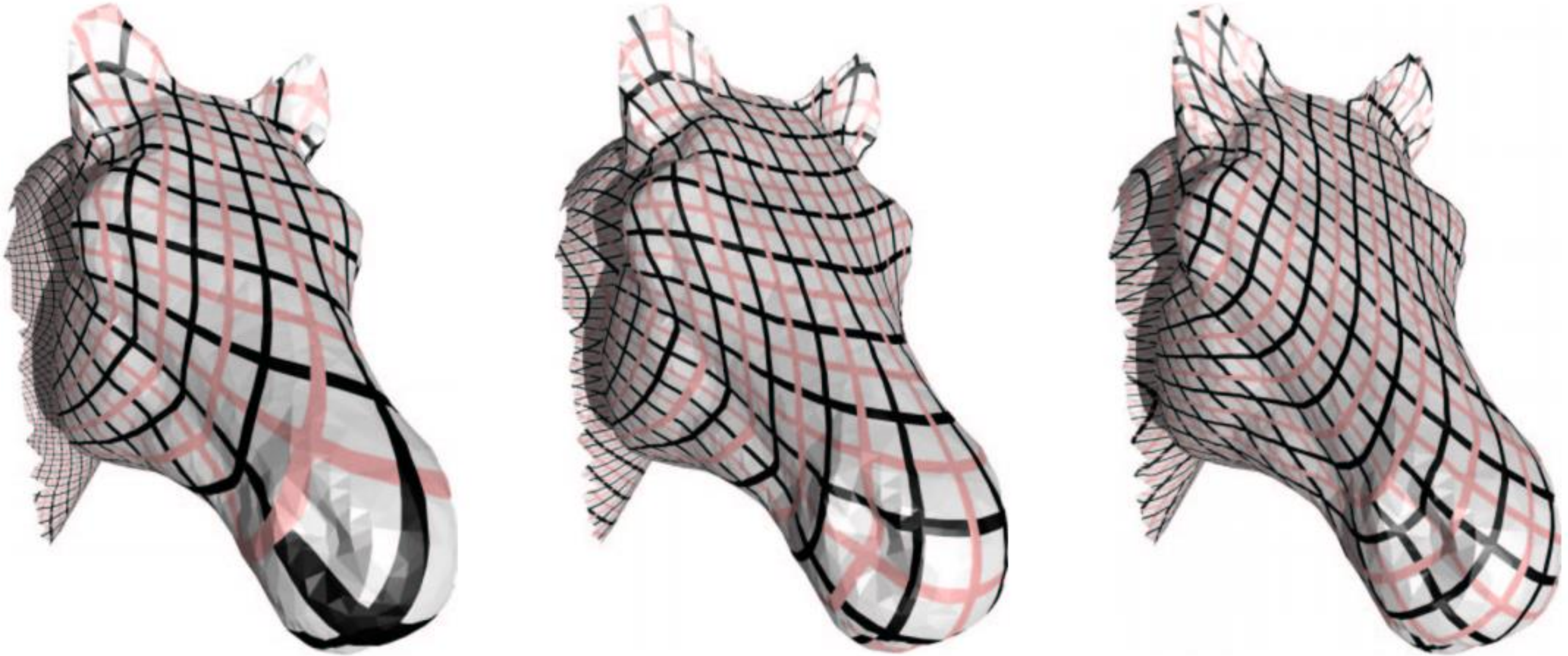


L2 Stretch Minimizing Parameterization [Sander et al., 2001]

An Adaptable Surface Parameterization [Degener et al., 2003]

- Controllable Area Preservation (via energy exponent θ)

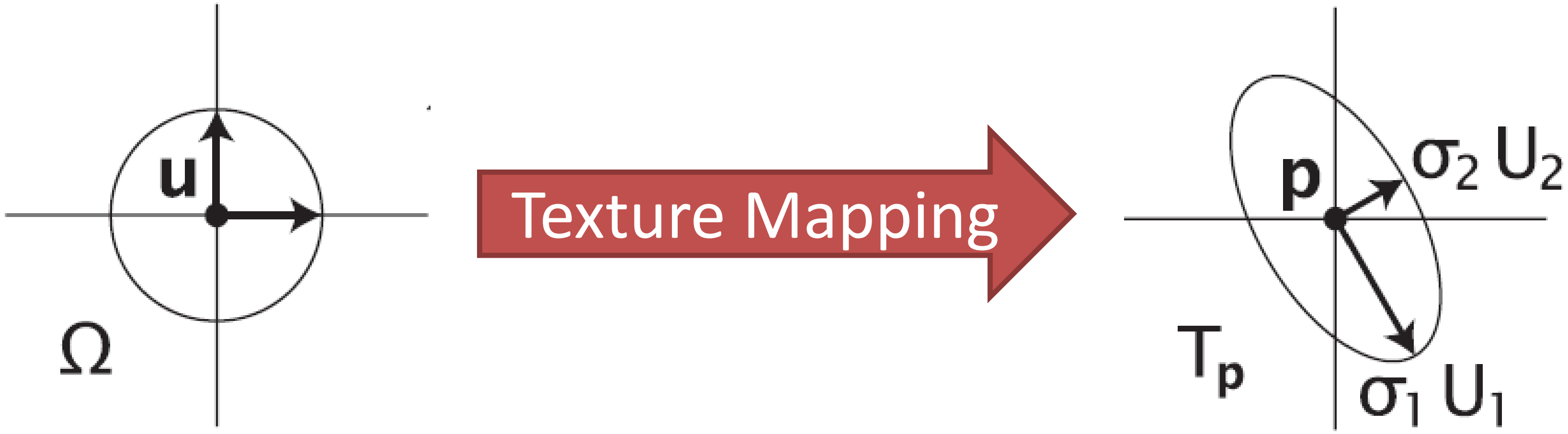
$\theta = 0.3, \theta = 1.0, \theta = 3.0$



Undistortion Energy [Degener et al., 2003]

Texture Space

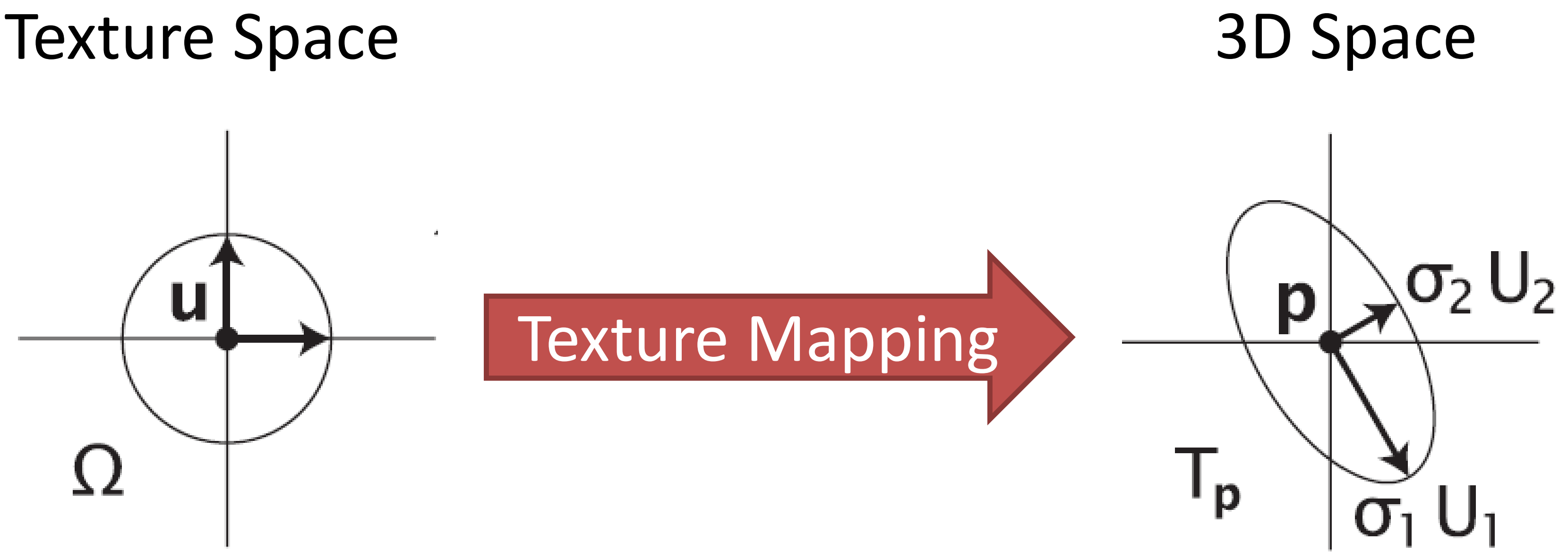
3D Space



Texture Mapping

[Hormann et al., 2008]

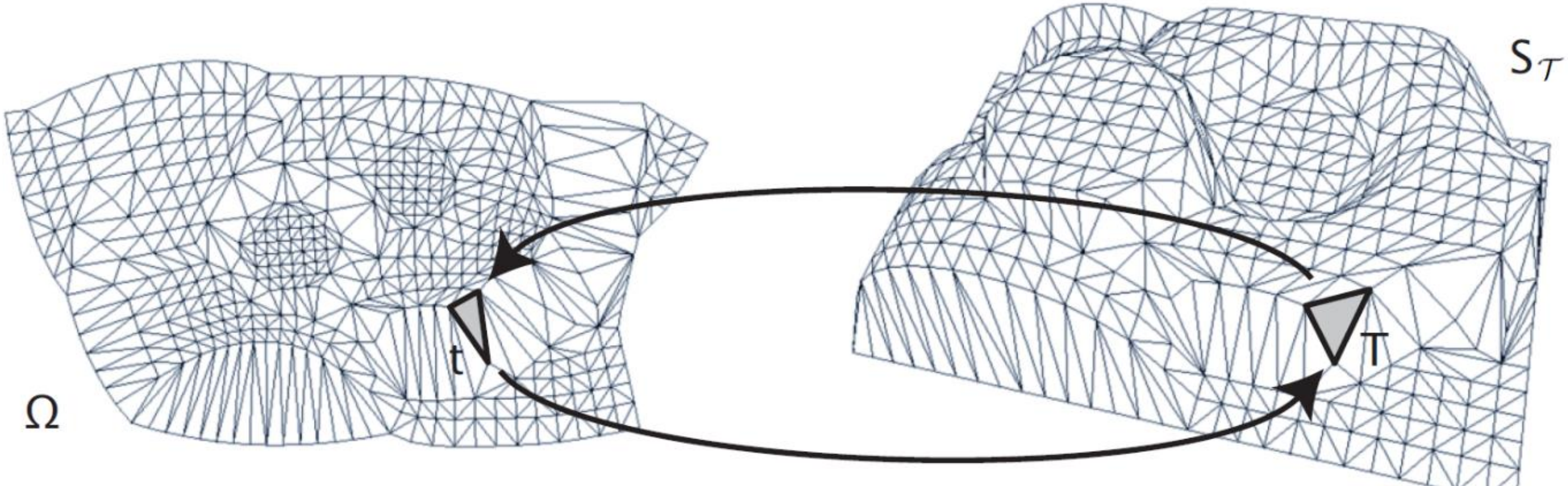
Undistortion Energy [Degener et al., 2003]


Texture Space

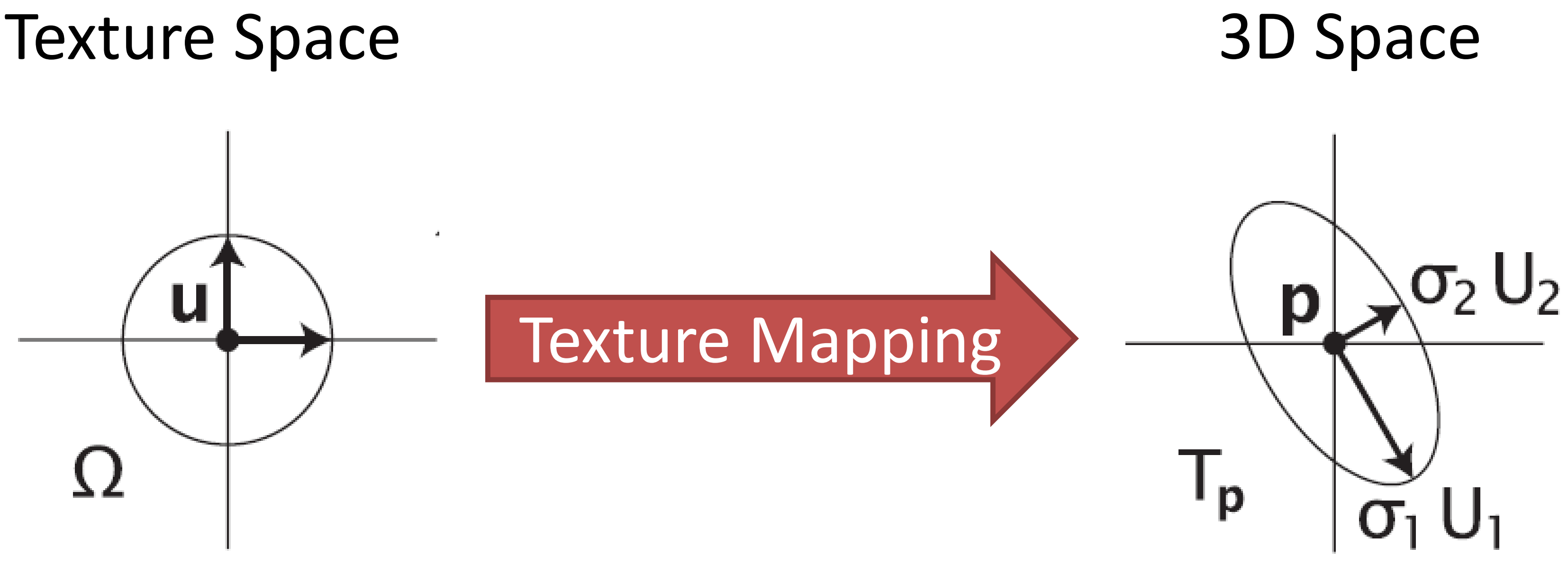
Texture Mapping

3D Space

[Hormann et al., 2008]



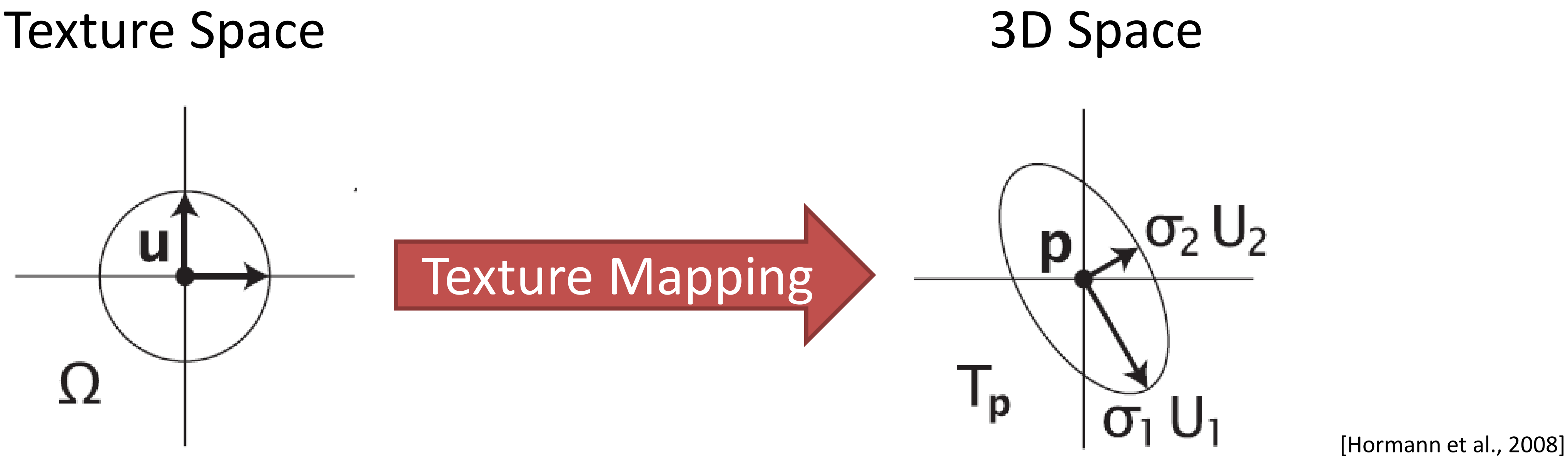
Undistortion Energy [Degener et al., 2003]



[Hormann et al., 2008]

- σ_1 longest, σ_2 shortest stretch
  - Ratio: angle preservation
  - Product: change in area

Undistortion Energy [Degener et al., 2003]



[Hormann et al., 2008]

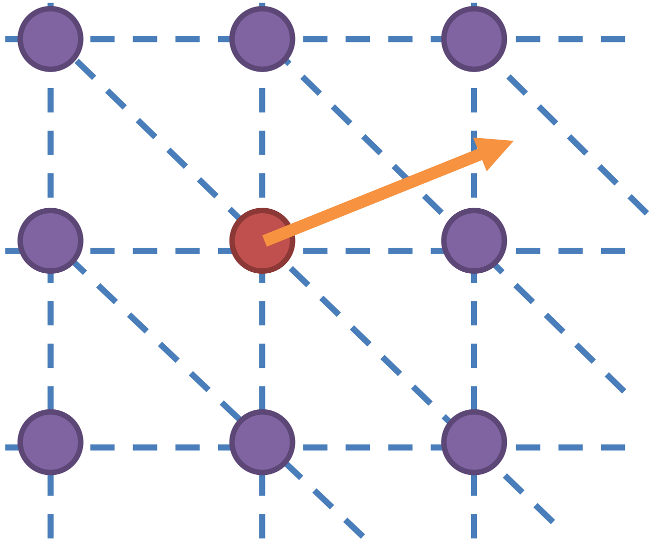
Degener et al.:

$$\left(\frac{\sigma_1}{\sigma_2}+\frac{\sigma_2}{\sigma_1}\right)\cdot\left(\sigma_1\sigma_2+\frac{1}{\sigma_1\sigma_2}\right)^{\theta}$$

- σ_1 longest, σ_2 shortest stretch
  - Ratio: angle preservation
  - Product: change in area

Local Energy Optimization

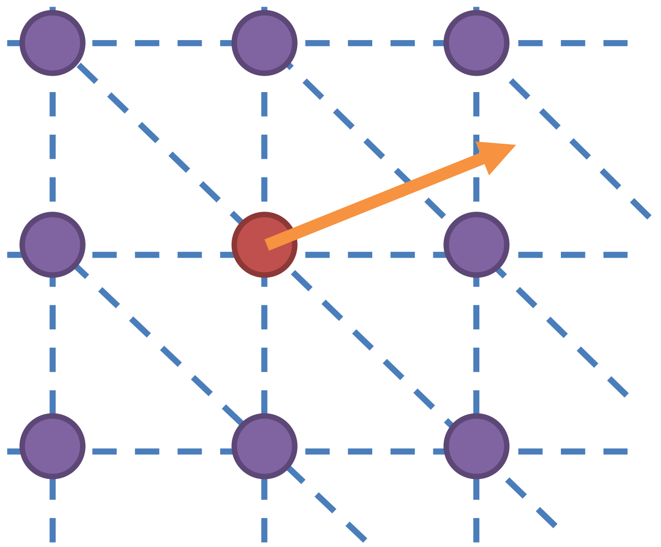
- Massively parallel gradient descent (GD) per texel



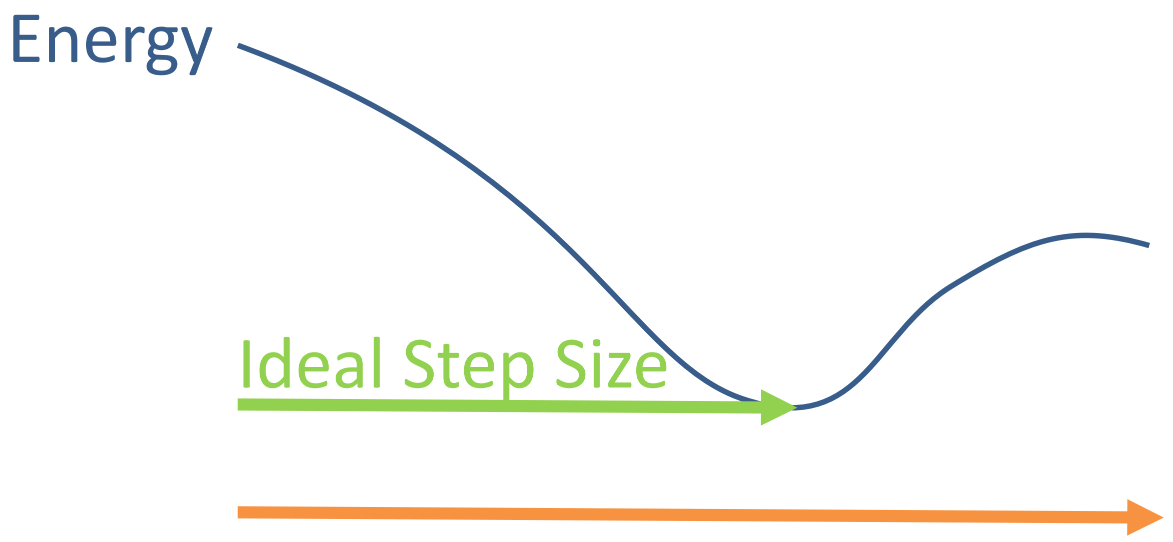
Gradient -> direction of highest local change

Local Energy Optimization

- Massively parallel gradient descent (GD) per texel
  - Find local optimum along gradient line for each texel

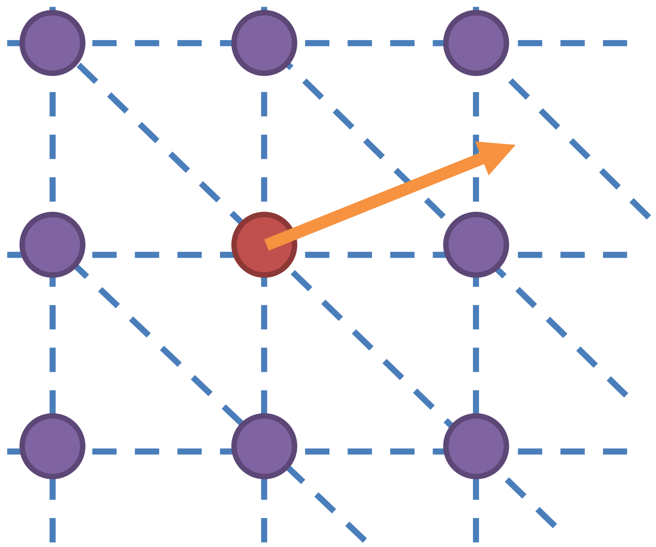


Gradient = direction of highest local change

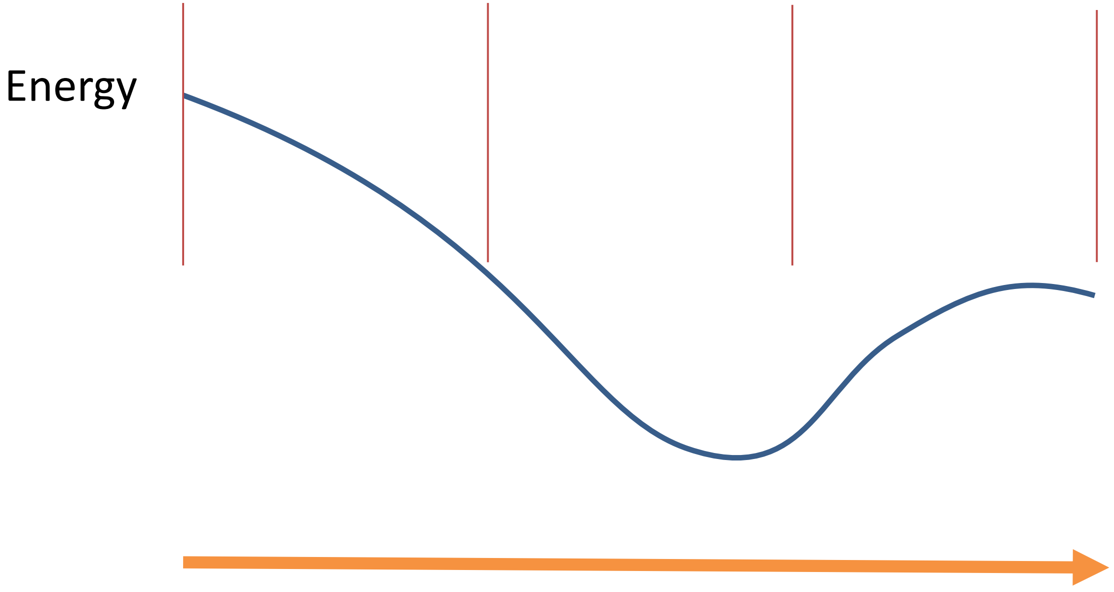


Local Energy Optimization

- Massively parallel gradient descent (GD) per texel
  - Find local optimum along gradient line for each texel



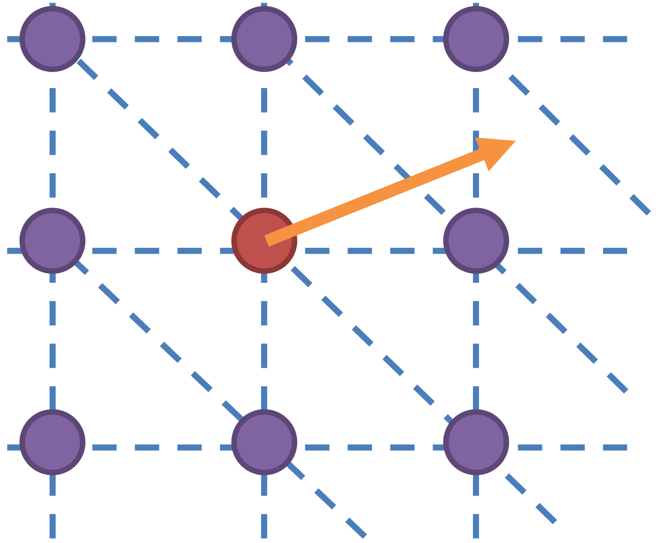
Gradient = direction of highest local change



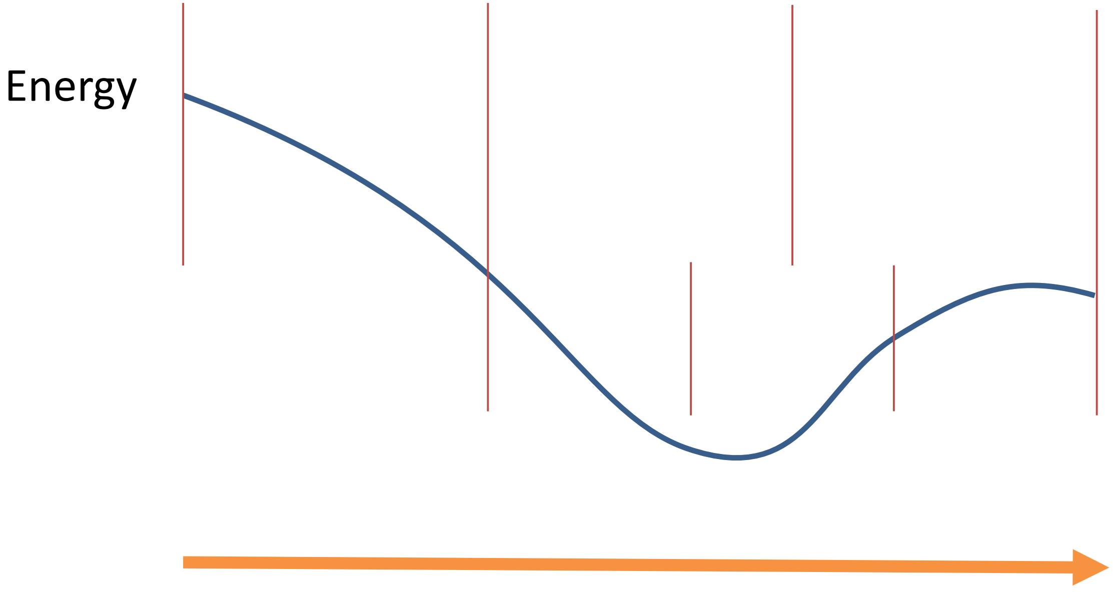
Ternary line search of local optimum

Local Energy Optimization

- Massively parallel gradient descent (GD) per texel
  - Find local optimum along gradient line for each texel



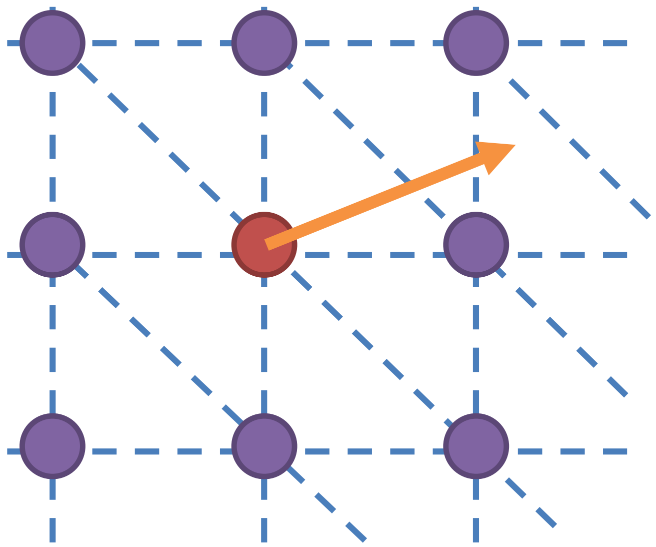
Gradient = direction of highest local change



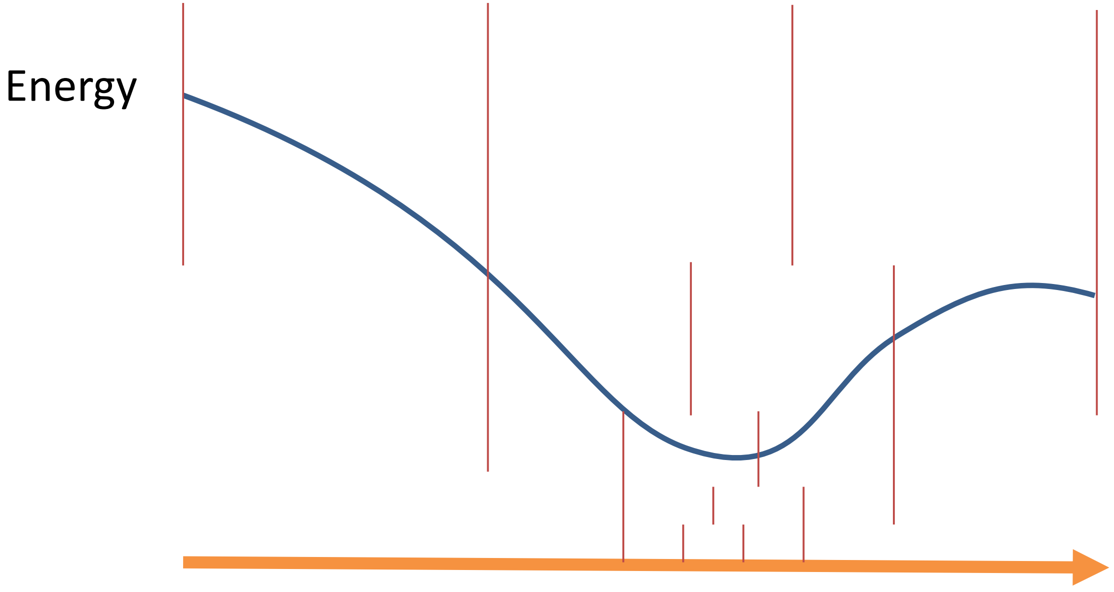
Ternary line search of local optimum

Local Energy Optimization

- Massively parallel gradient descent (GD) per texel
  - Find local optimum along gradient line for each texel



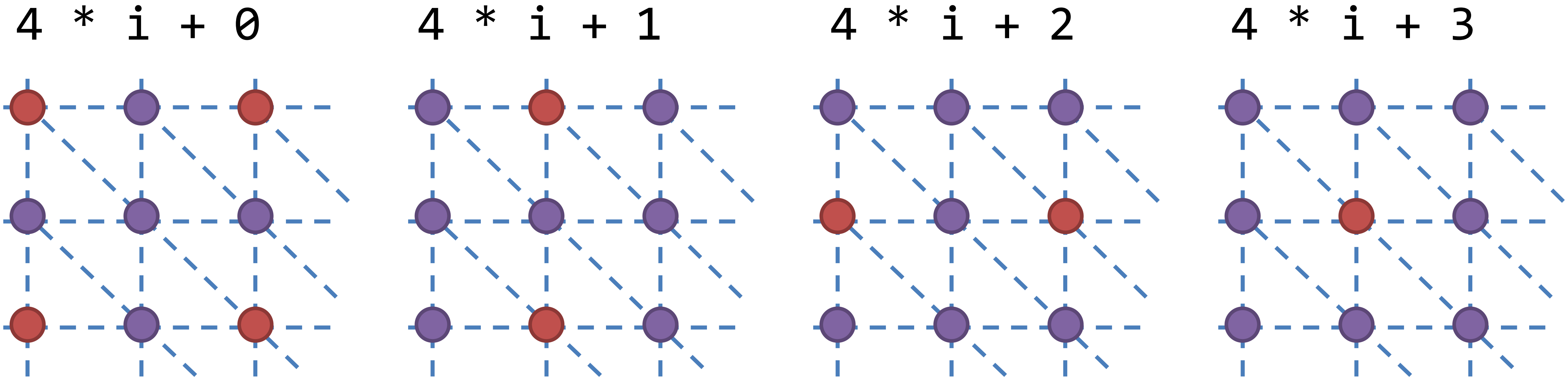
Gradient = direction of highest local change



Ternary line search of local optimum

Ensuring Convergence

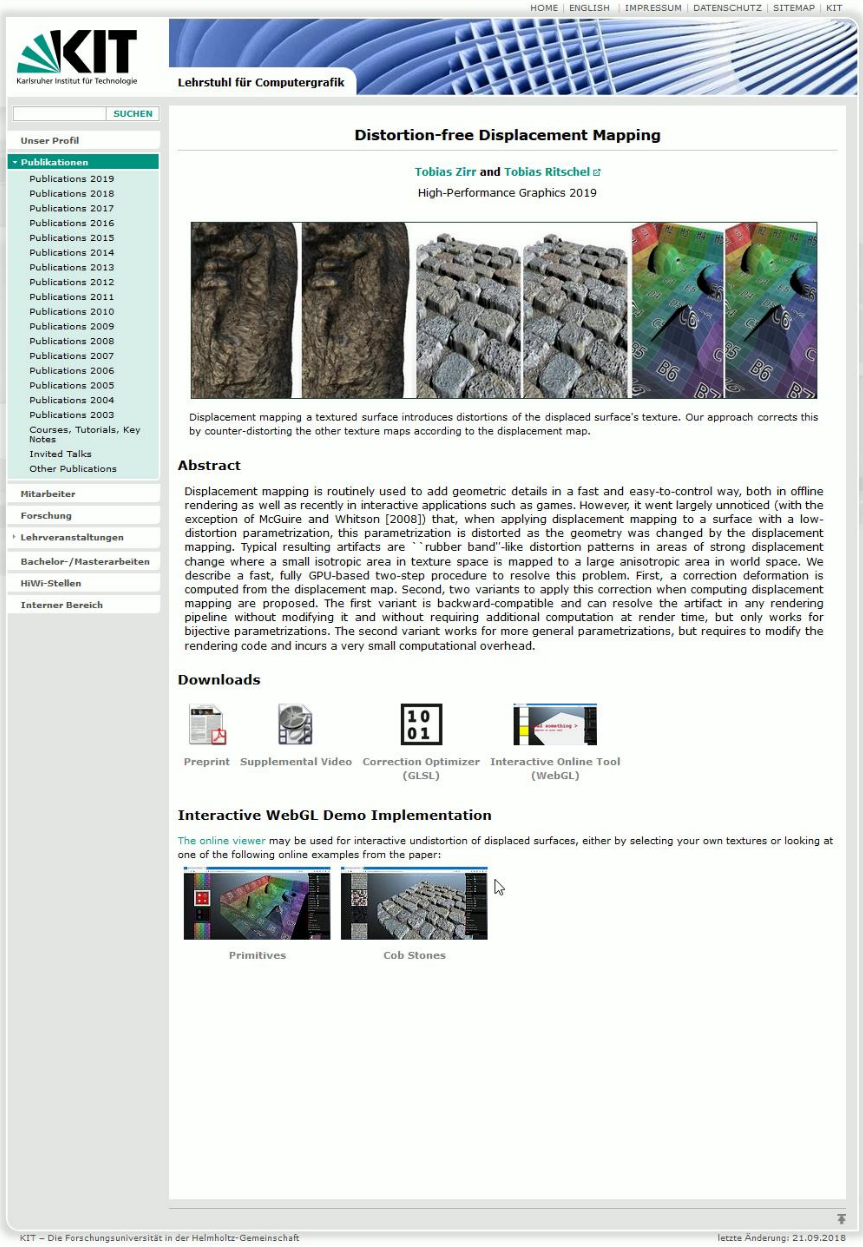
- Parallel local optimum for constant neighborhood (otherwise oscillations!)
    - One thread per correction texel **quad**
    - Split across four iterations



GLSL implementation can run interactively ☺

HPG 2019

- [MOVIE]



Distortion-free Displacement Mapping

Tobias Zirr and Tobias Ritschel

High-Performance Graphics 2019

Displacement mapping a textured surface introduces distortions of the displaced surface's texture. Our approach corrects this by counter-distorting the other texture maps according to the displacement map.

Abstract

Displacement mapping is routinely used to add geometric details in a fast and easy-to-control way, both in offline rendering as well as recently in interactive applications such as games. However, it went largely unnoticed (with the exception of McGuire and Whitson [2008]) that, when applying displacement mapping to a surface with a low-distortion parametrization, this parametrization is distorted as the geometry was changed by the displacement mapping. Typical resulting artifacts are ``rubber band''-like distortion patterns in areas of strong displacement change where a small isotropic area in texture space is mapped to a large anisotropic area in world space. We describe a fast, fully GPU-based two-step procedure to resolve this problem. First, a correction deformation is computed from the displacement map. Second, two variants to apply this correction when computing displacement mapping are proposed. The first variant is backward-compatible and can resolve the artifact in any rendering pipeline without modifying it and without requiring additional computation at render time, but only works for bijective parametrizations. The second variant works for more general parametrizations, but requires to modify the rendering code and incurs a very small computational overhead.

Downloads

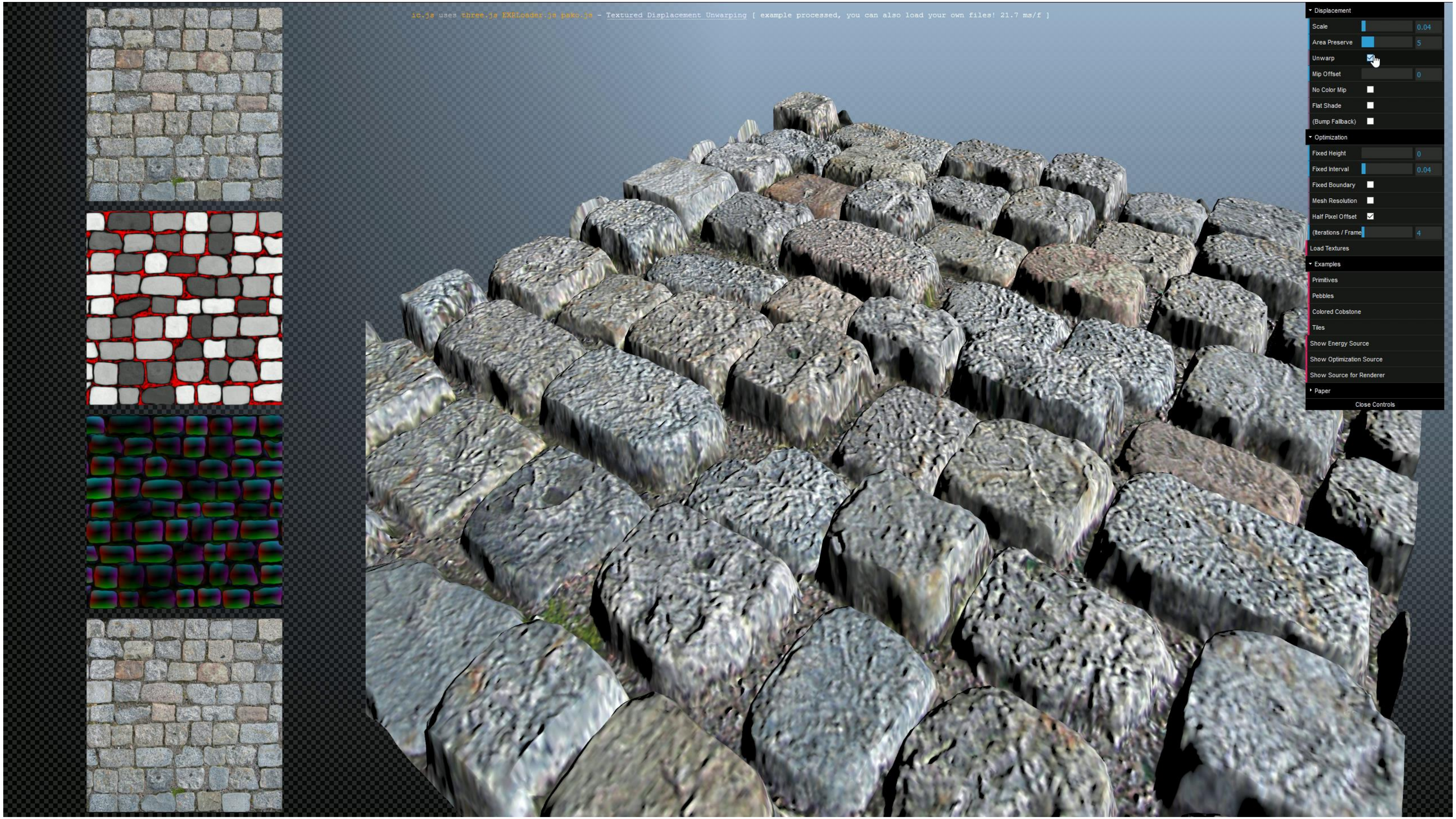
Interactive WebGL Demo Implementation

The online viewer may be used for interactive undistortion of displaced surfaces, either by selecting your own textures or looking at one of the following online examples from the paper:

KIT – Die Forschungsuniversität in der Helmholtz-Gemeinschaft

letzte Änderung: 21.09.2018

GLSL implementation can run interactively ☺



ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [example processed, you can also load your own files! 21.7 ms/f]

▾ Displacement

Scale 0.04

Area Preserve 5

Unwarp ☑

Mip Offset 0

No Color Mip □

Flat Shade □

(Bump Fallback) □

▾ Optimization

Fixed Height 0

Fixed Interval 0.04

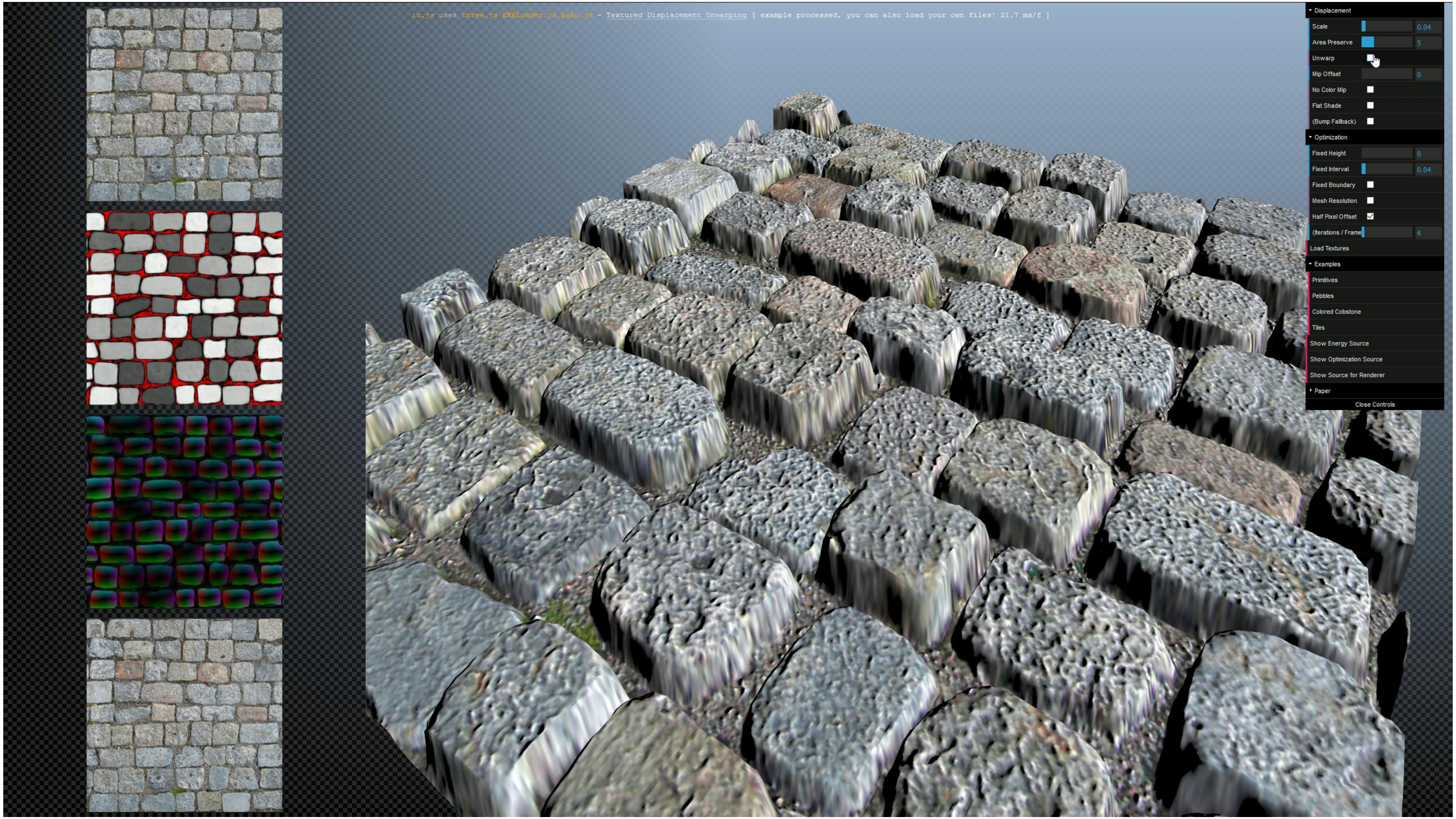
Fixed Boundary □

Mesh Resolution □

Half Pixel Offset ☑

(Iterations / Frame 4

GLSL implementation can run interactively ☺



ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [example processed, you can also load your own files! 21.7 ms/f]

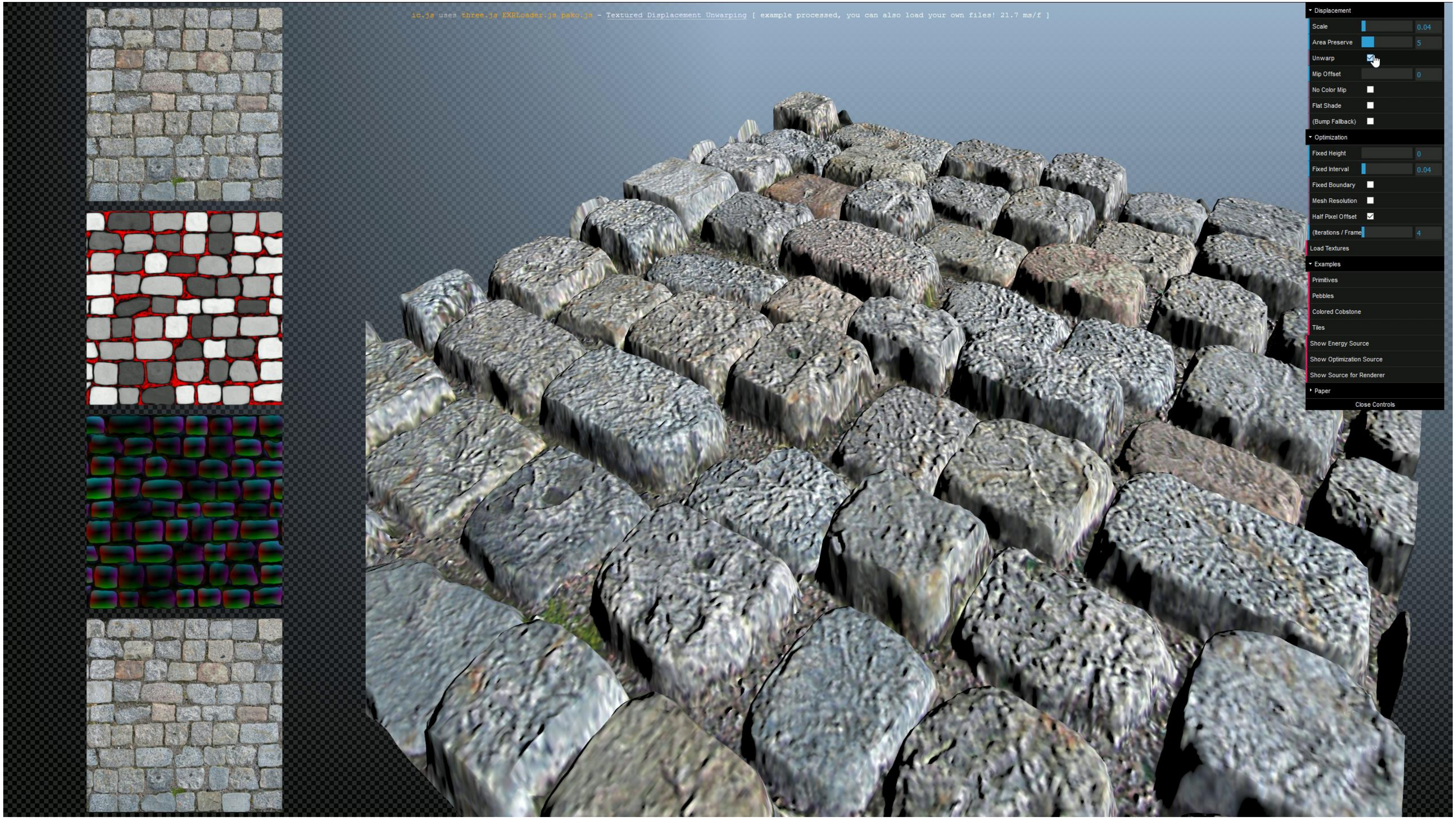
▾ Displacement

- Scale 0.04
- Area Preserve 5
- Unwarp □
- Mip Offset 0
- No Color Mip □
- Flat Shade □
- (Bump Fallback) □

▾ Optimization

- Fixed Height 0
- Fixed Interval 0.04
- Fixed Boundary □
- Mesh Resolution □
- Half Pixel Offset ☑
- (Iterations / Frame 4

GLSL implementation can run interactively ☺

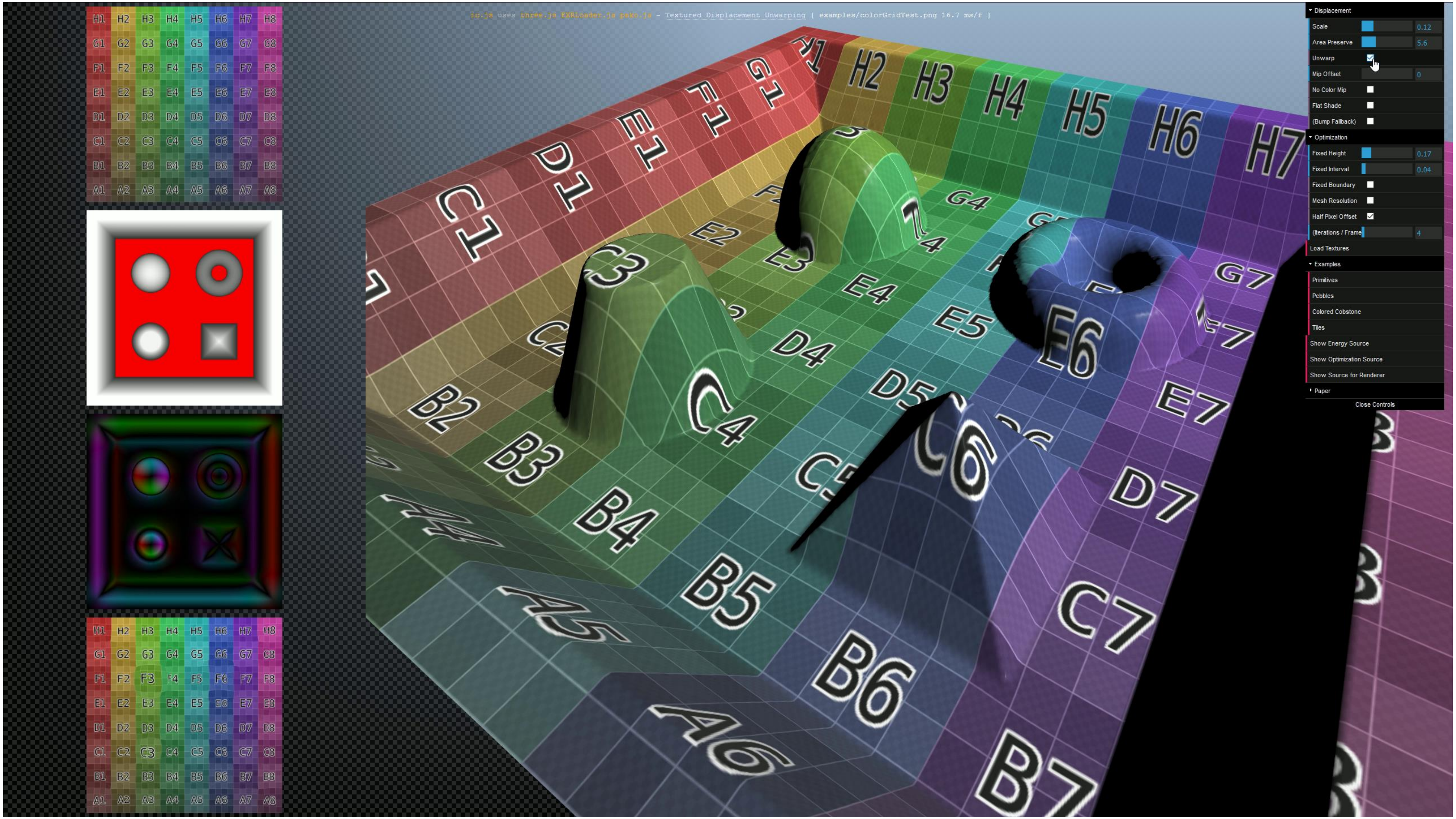


ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [example processed, you can also load your own files! 21.7 ms/f]

Displacement
- Scale 0.04
- Area Preserve 5
- Unwarp ☑
- Mip Offset 0
- No Color Mip □
- Flat Shade □
- (Bump Fallback) □

Optimization
- Fixed Height 0
- Fixed Interval 0.04
- Fixed Boundary □
- Mesh Resolution □
- Half Pixel Offset ☑
- (Iterations / Frame 4

GLSL implementation can run interactively ☺



ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [examples/colorGridTest.png 16.7 ms/f]

▾ Displacement
- Scale 0.12
- Area Preserve 5.6
- Unwarp ☑
- Mip Offset 0
- No Color Mip □
- Flat Shade □
- (Bump Fallback) □

▾ Optimization
- Fixed Height 0.17
- Fixed Interval 0.04
- Fixed Boundary □
- Mesh Resolution □
- Half Pixel Offset ☑
- (Iterations / Frame 4

Load Textures

▾ Examples
- Primitives
- Pebbles
- Colored Cobstone
- Tiles

Show Energy Source

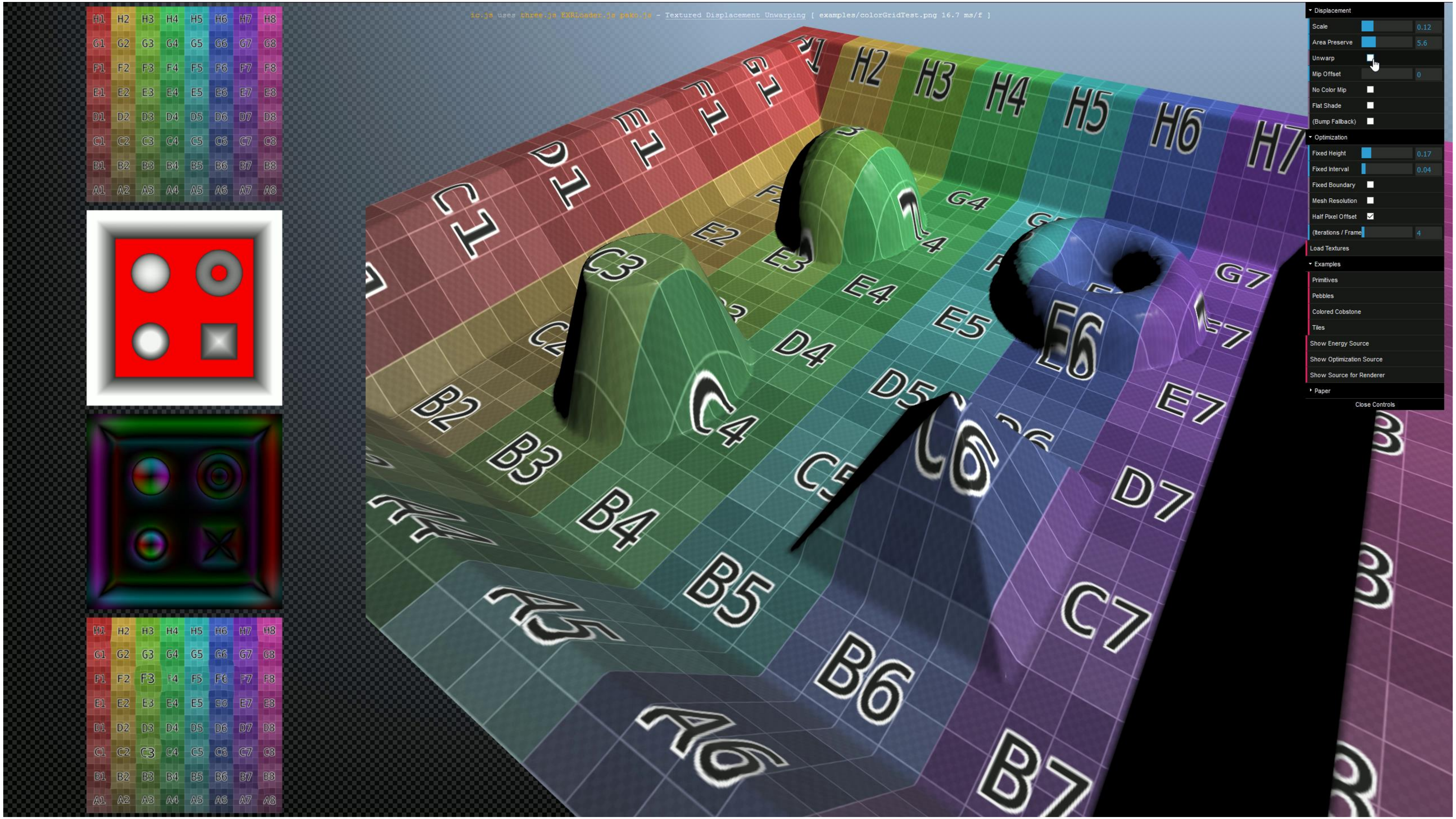
Show Optimization Source

Show Source for Renderer

▸ Paper

Close Controls

GLSL implementation can run interactively ☺



ic.js uses three.js EXRLoader.js pako.js – Textured Displacement Unwarping [examples/colorGridTest.png 16.7 ms/f]

▾ Displacement
- Scale 0.12
- Area Preserve 5.6
- Unwarp ☐
- Mip Offset 0
- No Color Mip ☐
- Flat Shade ☐
- (Bump Fallback) ☐

▾ Optimization
- Fixed Height 0.17
- Fixed Interval 0.04
- Fixed Boundary ☐
- Mesh Resolution ☐
- Half Pixel Offset ☑
- (Iterations / Frame 4

Load Textures

▾ Examples
- Primitives
- Pebbles
- Colored Cobstone
- Tiles

Show Energy Source

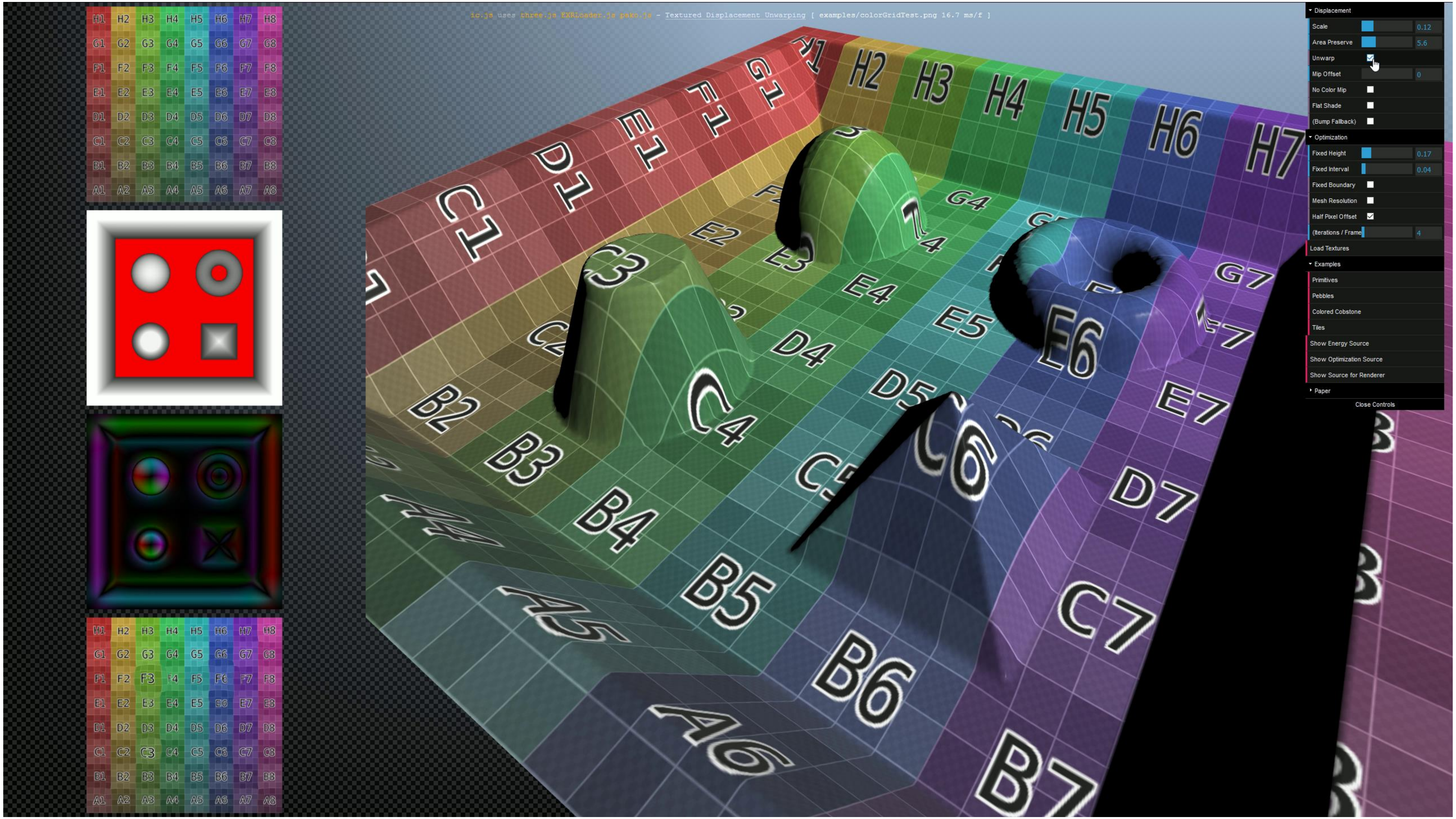
Show Optimization Source

Show Source for Renderer

▸ Paper

Close Controls

GLSL implementation can run interactively ☺



ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [examples/colorGridTest.png 16.7 ms/f]

▾ Displacement

- Scale 0.12
- Area Preserve 5.6
- Unwarp ☑
- Mip Offset 0
- No Color Mip □
- Flat Shade □
- (Bump Fallback) □

▾ Optimization

- Fixed Height 0.17
- Fixed Interval 0.04
- Fixed Boundary □
- Mesh Resolution □
- Half Pixel Offset ☑
- (Iterations / Frame 4

Load Textures

▾ Examples

- Primitives
- Pebbles
- Colored Cobstone
- Tiles

Show Energy Source

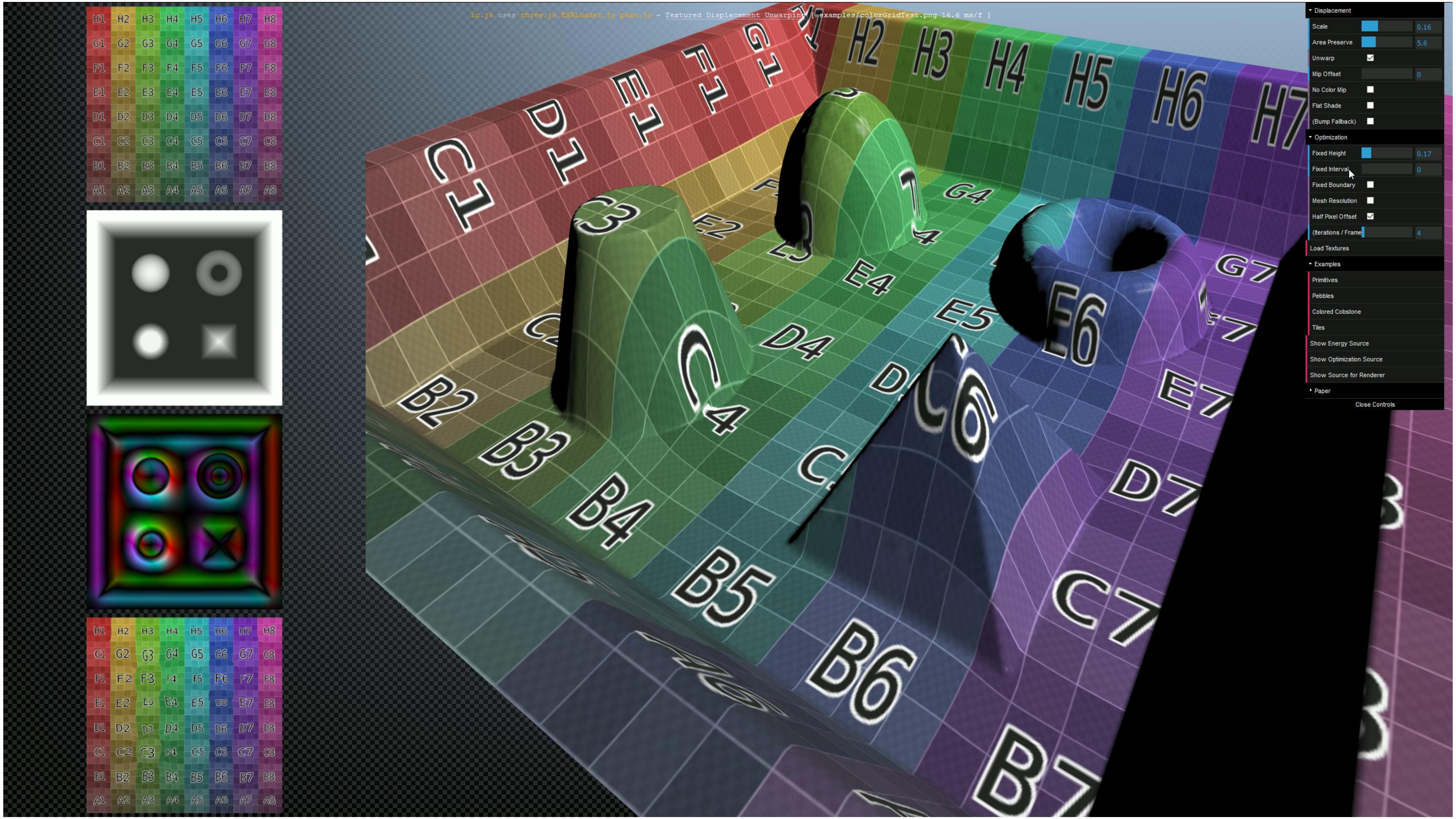
Show Optimization Source

Show Source for Renderer

▸ Paper

Close Controls

GLSL implementation can run interactively ☺



```
ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [ examples/colorGridTest.png 16.6 ms/f ]
```

▾ Displacement
- Scale 0.16
- Area Preserve 5.6
- Unwarp ☑
- Mip Offset 0
- No Color Mip □
- Flat Shade □
- (Bump Fallback) □

▾ Optimization
- Fixed Height 0.17
- Fixed Interval 0
- Fixed Boundary □
- Mesh Resolution □
- Half Pixel Offset ☑
- (Iterations / Frame 4

Load Textures

▾ Examples
- Primitives
- Pebbles
- Colored Cobstone
- Tiles

Show Energy Source

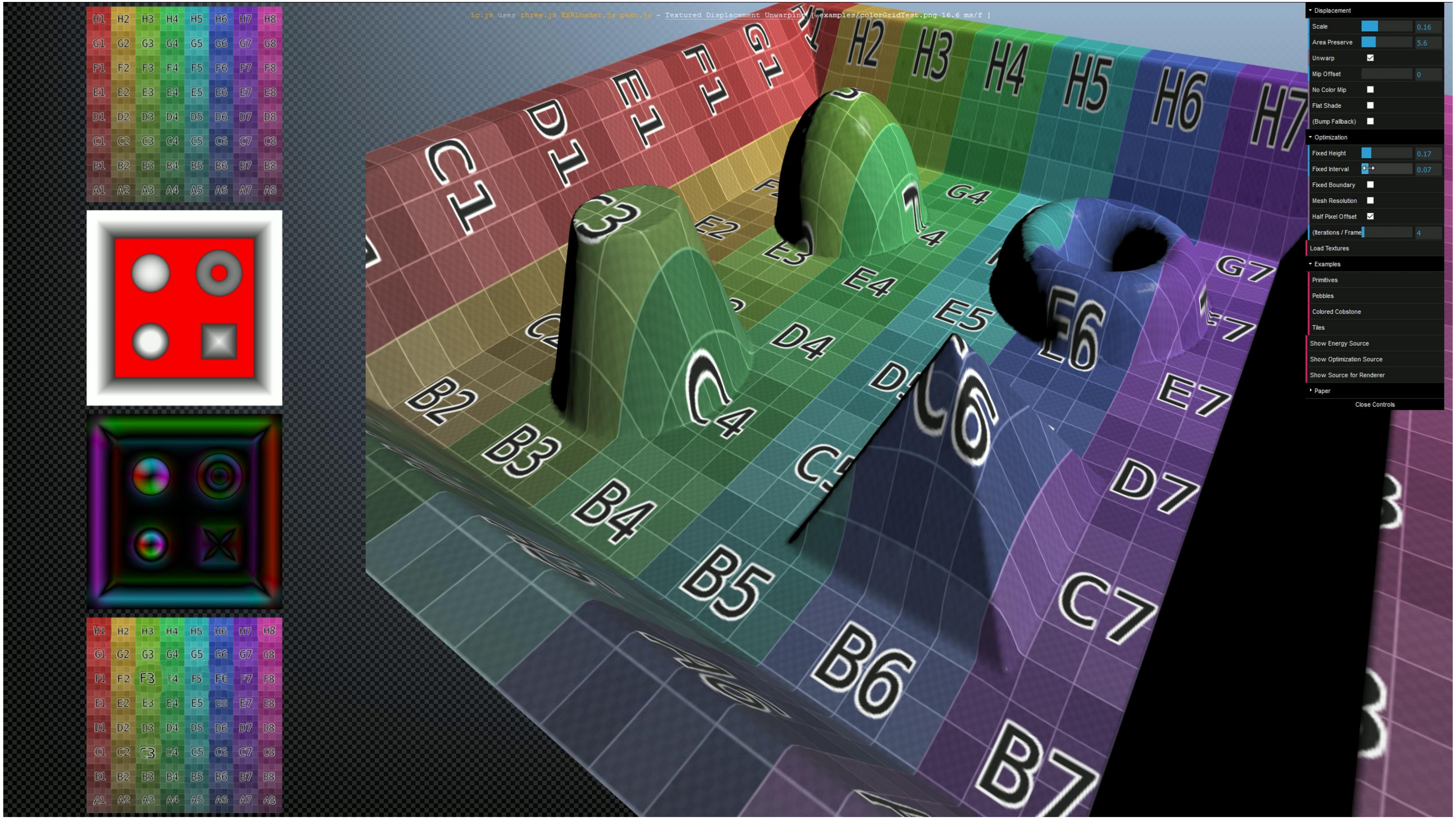
Show Optimization Source

Show Source for Renderer

▸ Paper

Close Controls

GLSL implementation can run interactively ☺

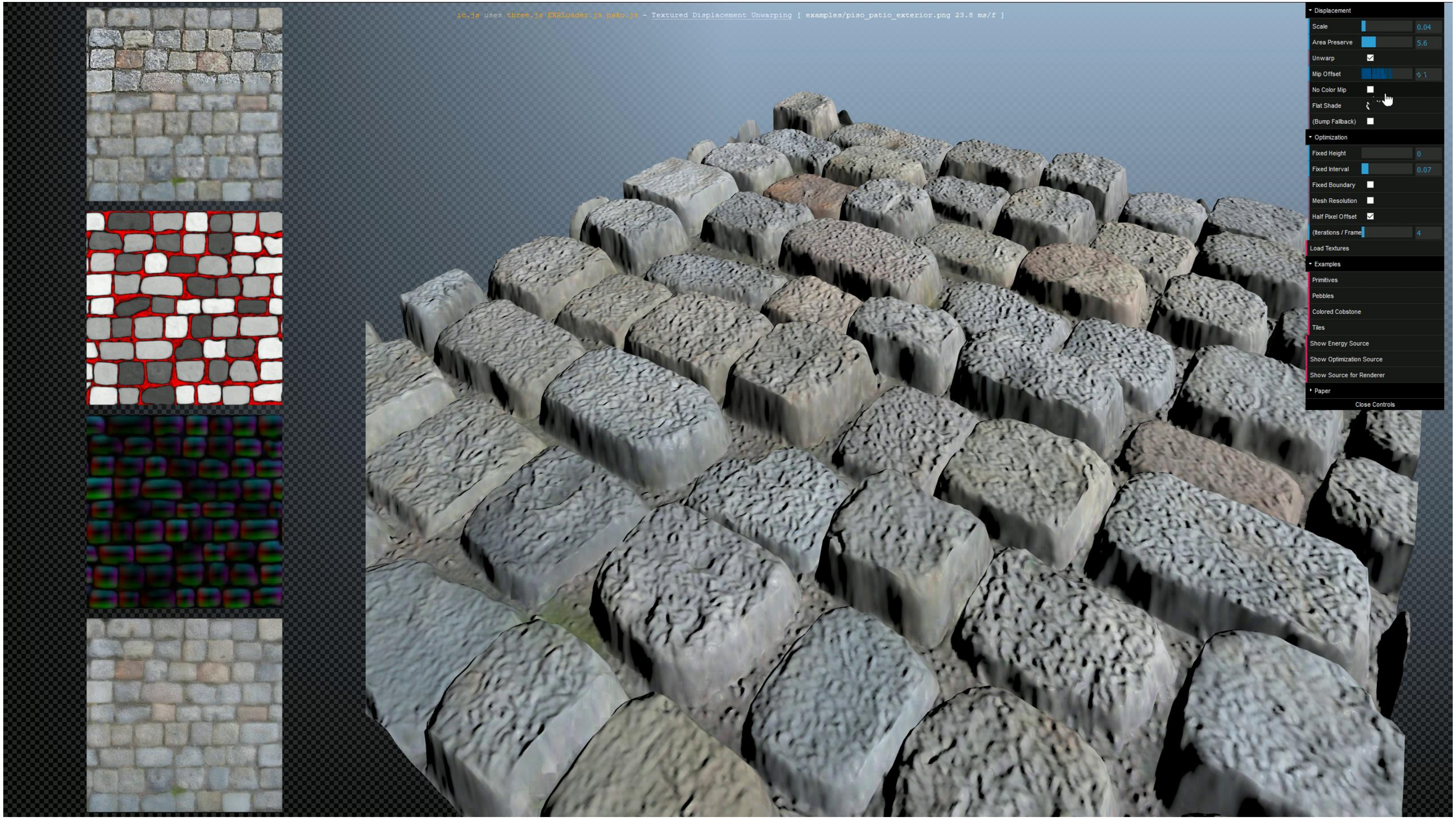


ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [examples/colorGridTest.png 16.6 ms/f]

- Displacement
  - Scale: 0.16
  - Area Preserve: 5.6
  - Unwarp ☑
  - Mip Offset: 0
  - No Color Mip □
  - Flat Shade □
  - (Bump Fallback) □
- Optimization
  - Fixed Height: 0.17
  - Fixed Interval: 0.07
  - Fixed Boundary □
  - Mesh Resolution □
  - Half Pixel Offset ☑
  - (Iterations / Frame: 4
- Load Textures
- Examples
  - Primitives
  - Pebbles
  - Colored Cobstone
  - Tiles
- Show Energy Source
- Show Optimization Source
- Show Source for Renderer
- Paper

Close Controls

GLSL implementation can run interactively ☺



ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [examples/piso_patio_exterior.png 23.8 ms/f]

▾ Displacement

Scale 0.04

Area Preserve 5.6

Unwarp ☑

Mip Offset 0.1

No Color Mip □

Flat Shade

(Bump Fallback) □

▾ Optimization

Fixed Height 0

Fixed Interval 0.07

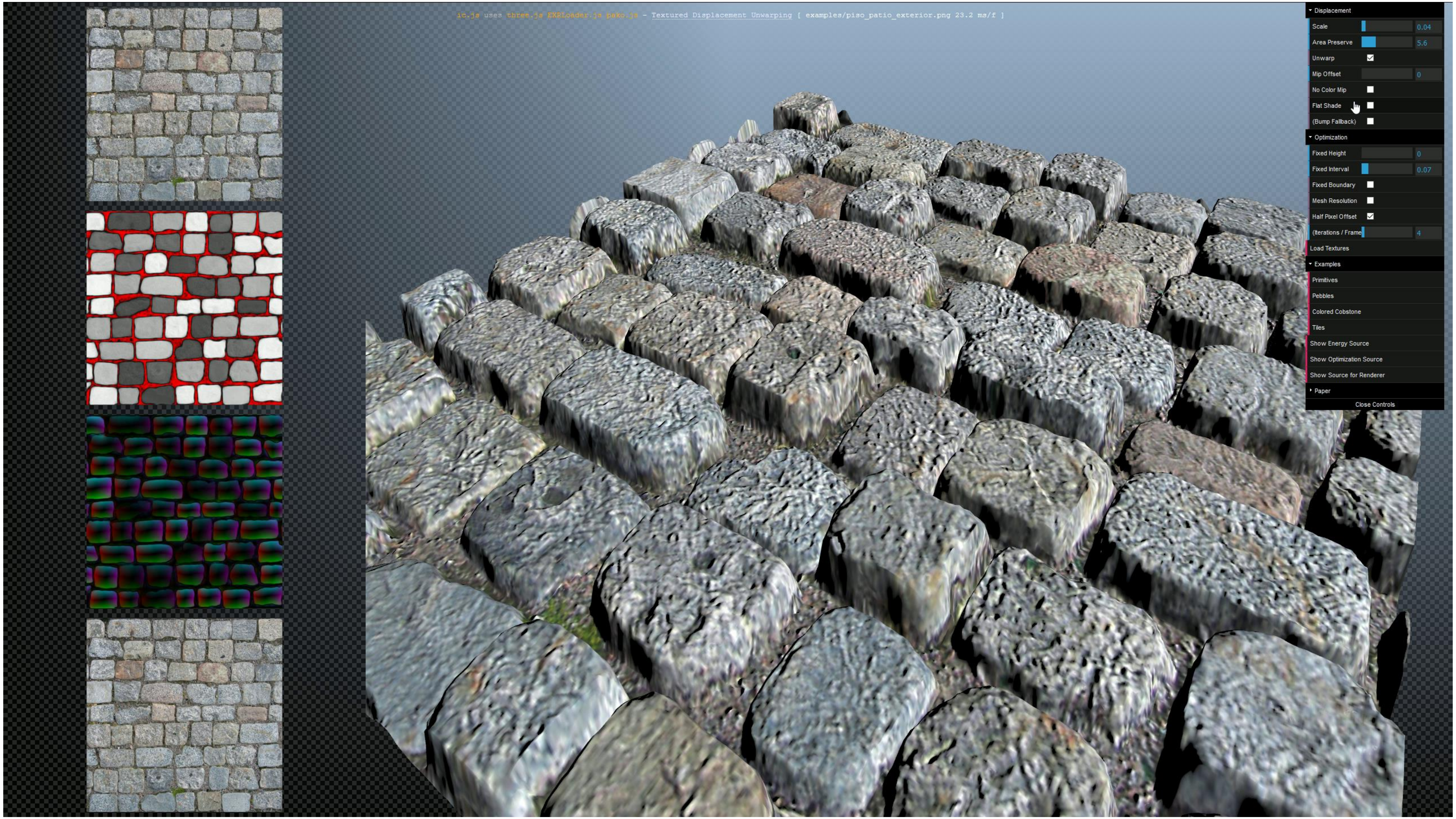
Fixed Boundary □

Mesh Resolution □

Half Pixel Offset ☑

(Iterations / Frame 4

GLSL implementation can run interactively ☺



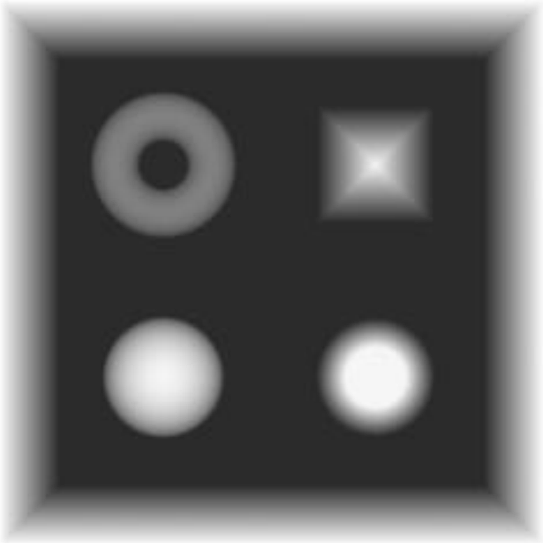
ic.js uses three.js EXRLoader.js pako.js – Textured Displacement Unwarping [examples/piso_patio_exterior.png 23.2 ms/f]

Displacement
- Scale 0.04
- Area Preserve 5.6
- Unwarp ☑
- Mip Offset 0
- No Color Mip □
- Flat Shade □
- (Bump Fallback) □

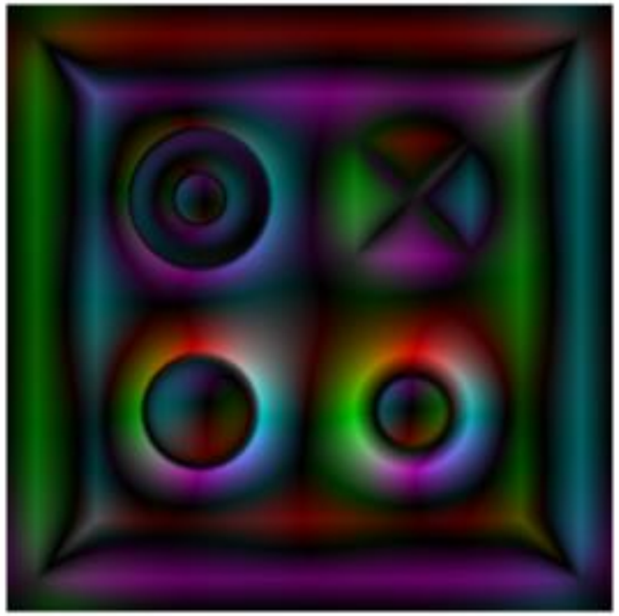
Optimization
- Fixed Height 0
- Fixed Interval 0.07
- Fixed Boundary □
- Mesh Resolution □
- Half Pixel Offset ☑
- (Iterations / Frame 4

Store relative per-texel coordinates!

- We don’t store per-texel coordinates, but per-texel offsets!
  - Small magnitude (compact quantization)
  - Works across texture tiling borders (preserving pixel derivatives)

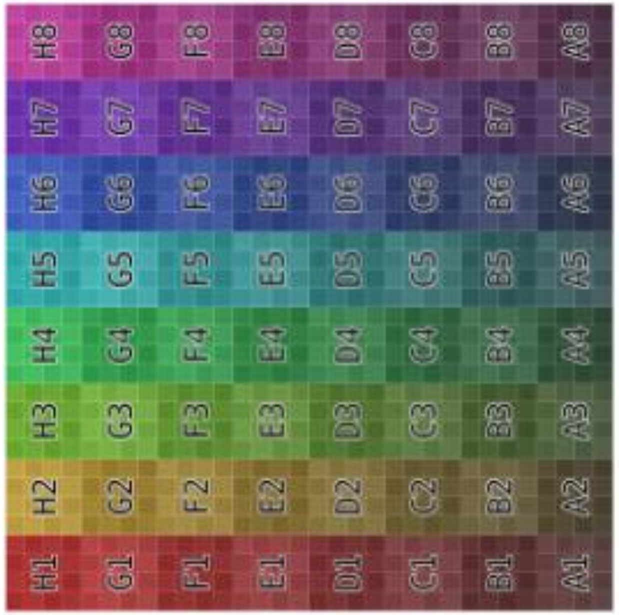


Displacement D



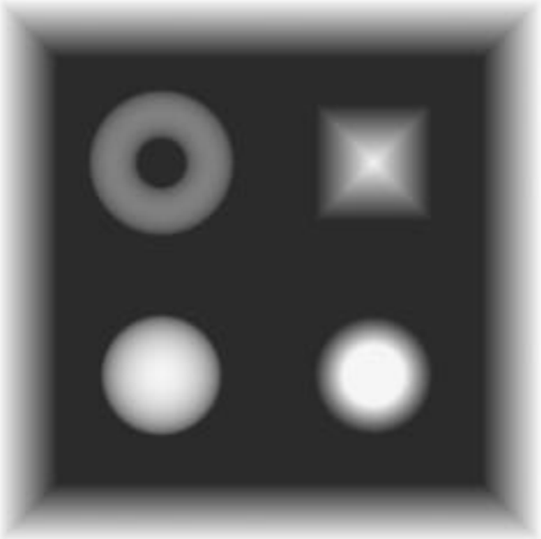
Per-texel texcoord offsets

Store relative per-texel coordinates!

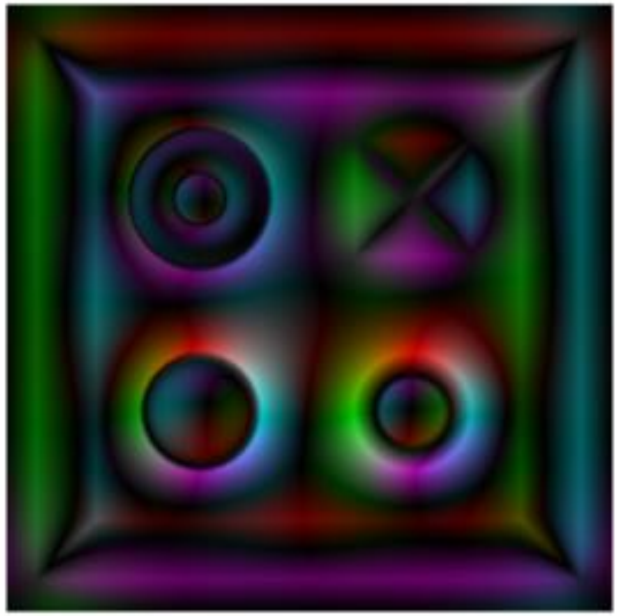


Albedo A

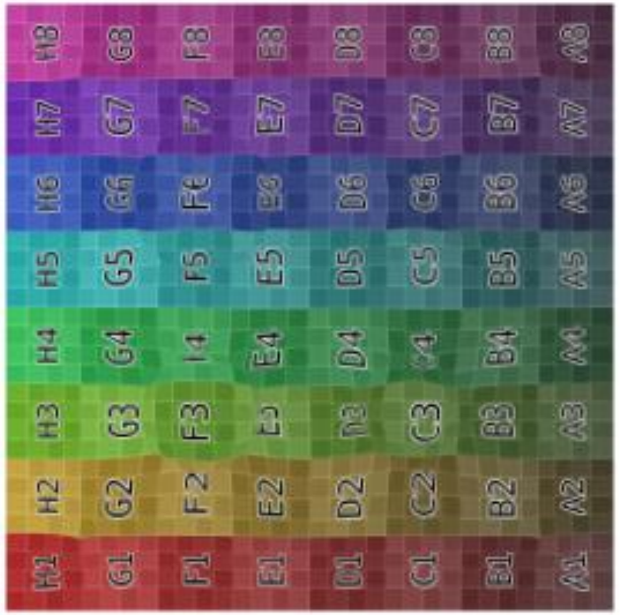
Linear texcoord + per-texel offset



Displacement D



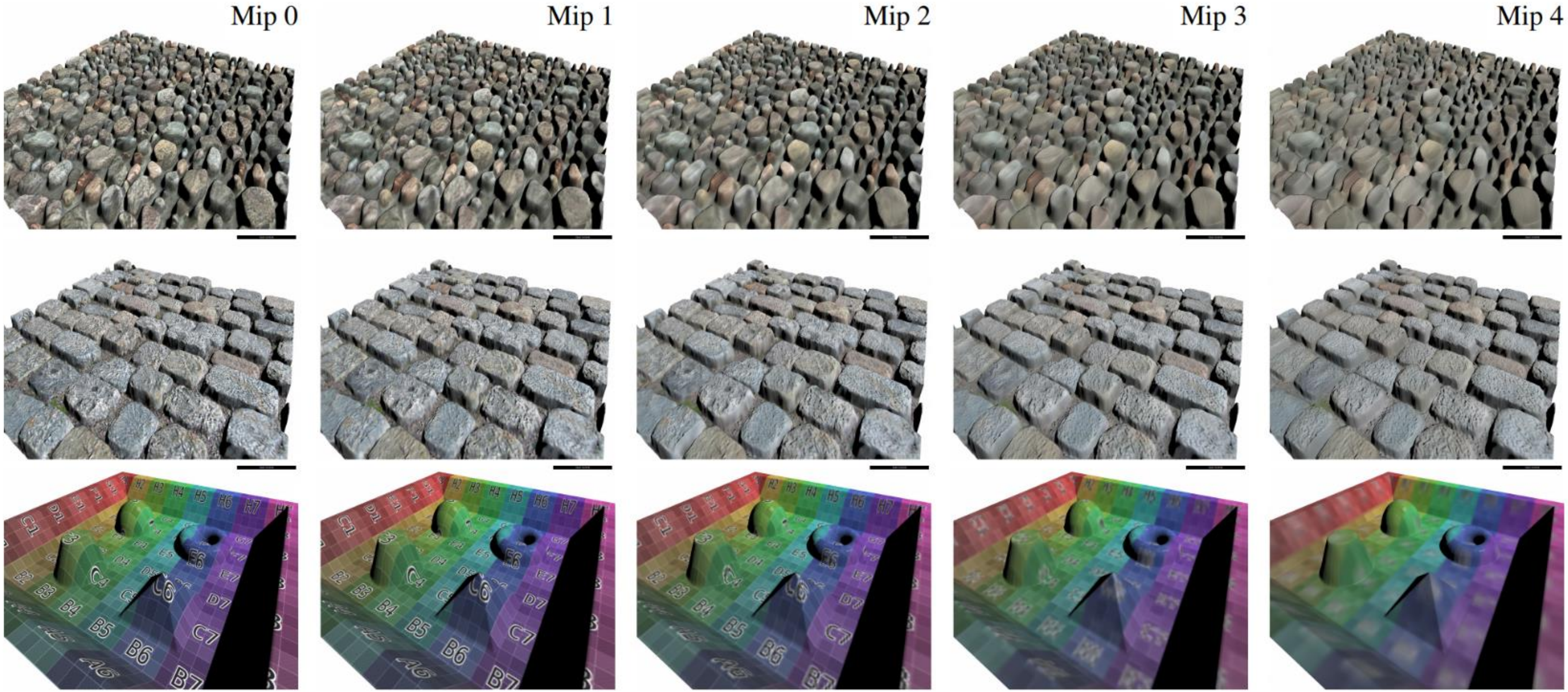
Per-texel texcoord offsets



Warped A

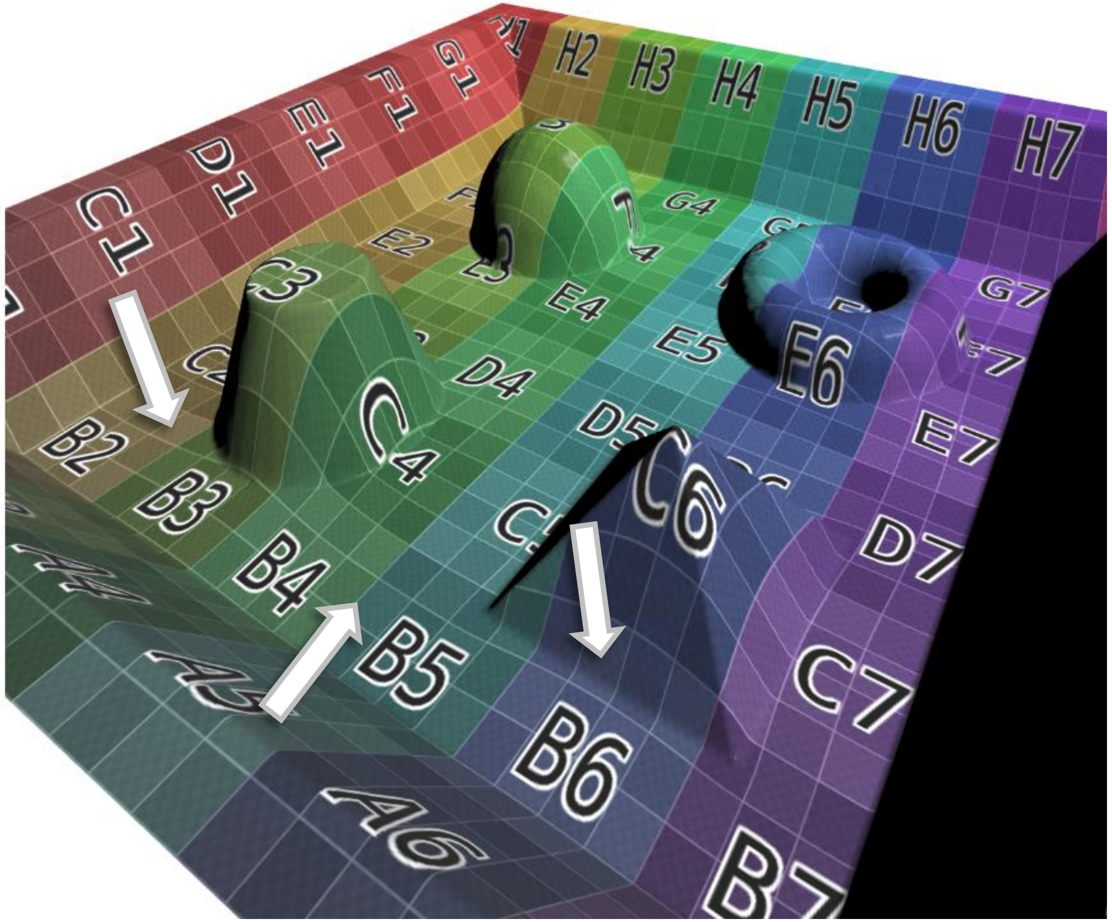
Mip Mapping

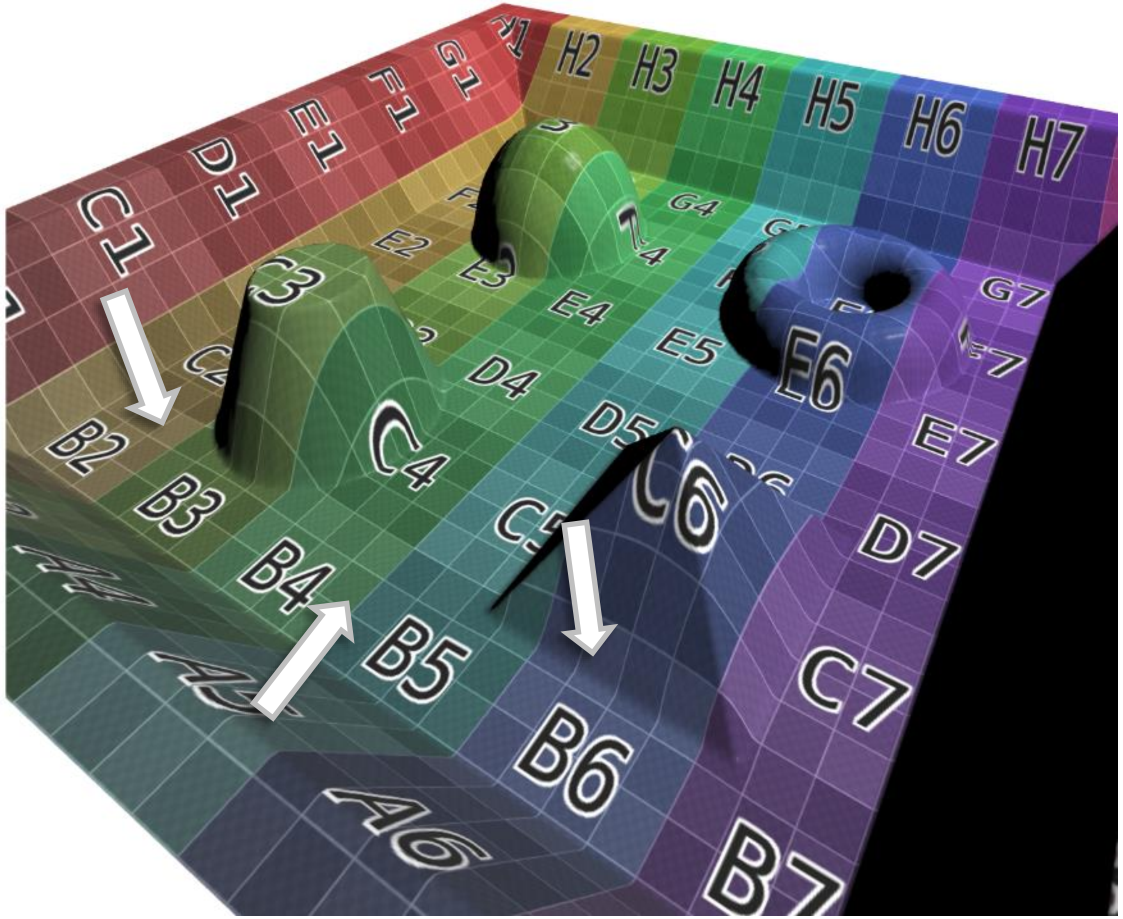
- Simplest solution works OK:
  - Mip Maps for both Correction Offsets & Color Textures



Fixation Energy

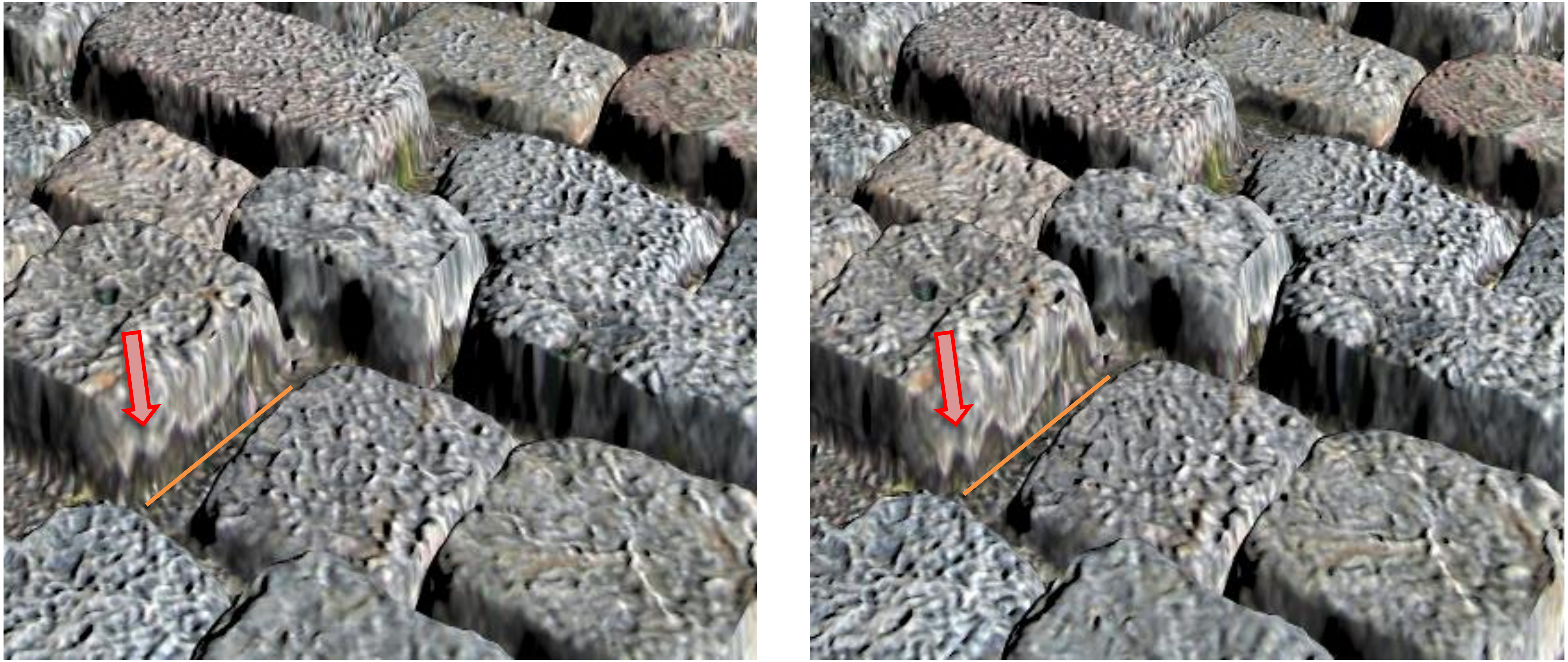
- Optimization can lead to unwanted distortion
  - Perturbed straight lines





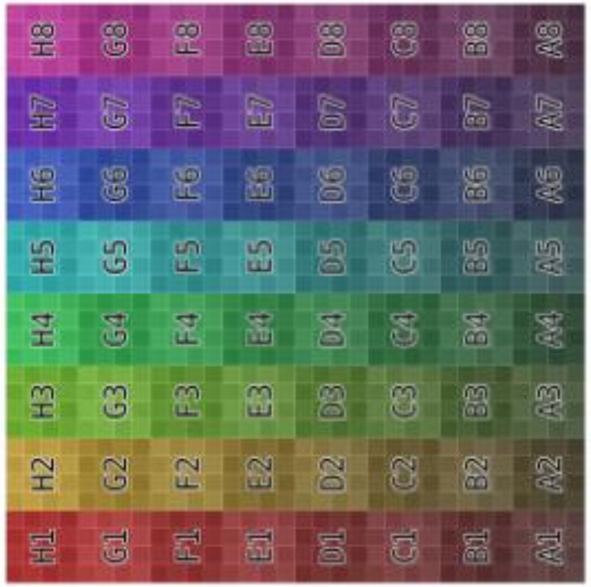
Fixation Energy

- Optimization can lead to unwanted distortion
  - Correlations between texture and geometry

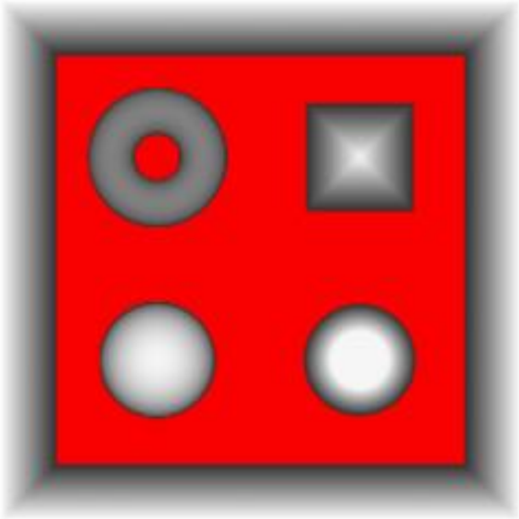


Fixation Energy

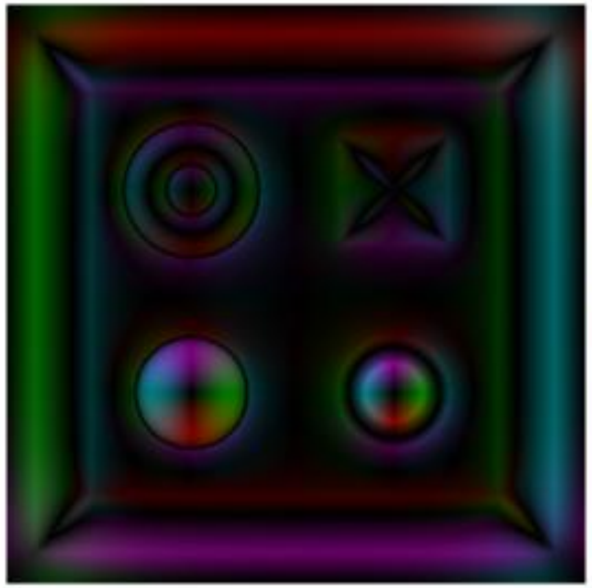
- Optimization can lead to unwanted distortion
  - Correlations between texture and geometry
  - Perturbed straight lines
- We allow to mark parts of textures fixed by an additional fixation energy term



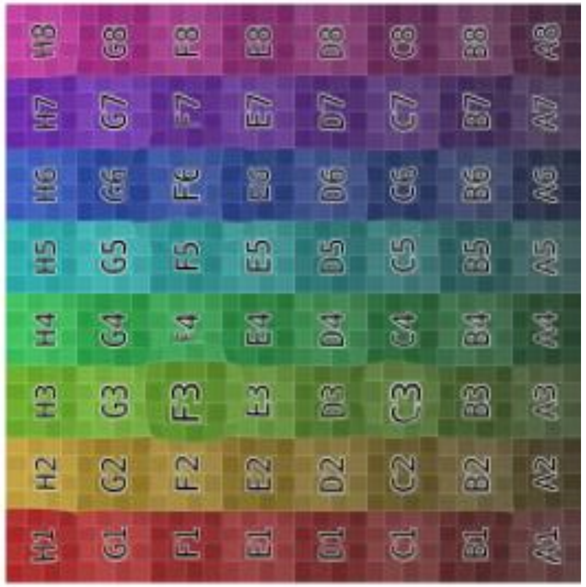
Albedo A



Displacement D, Fixation



Correction Offsets c(u)

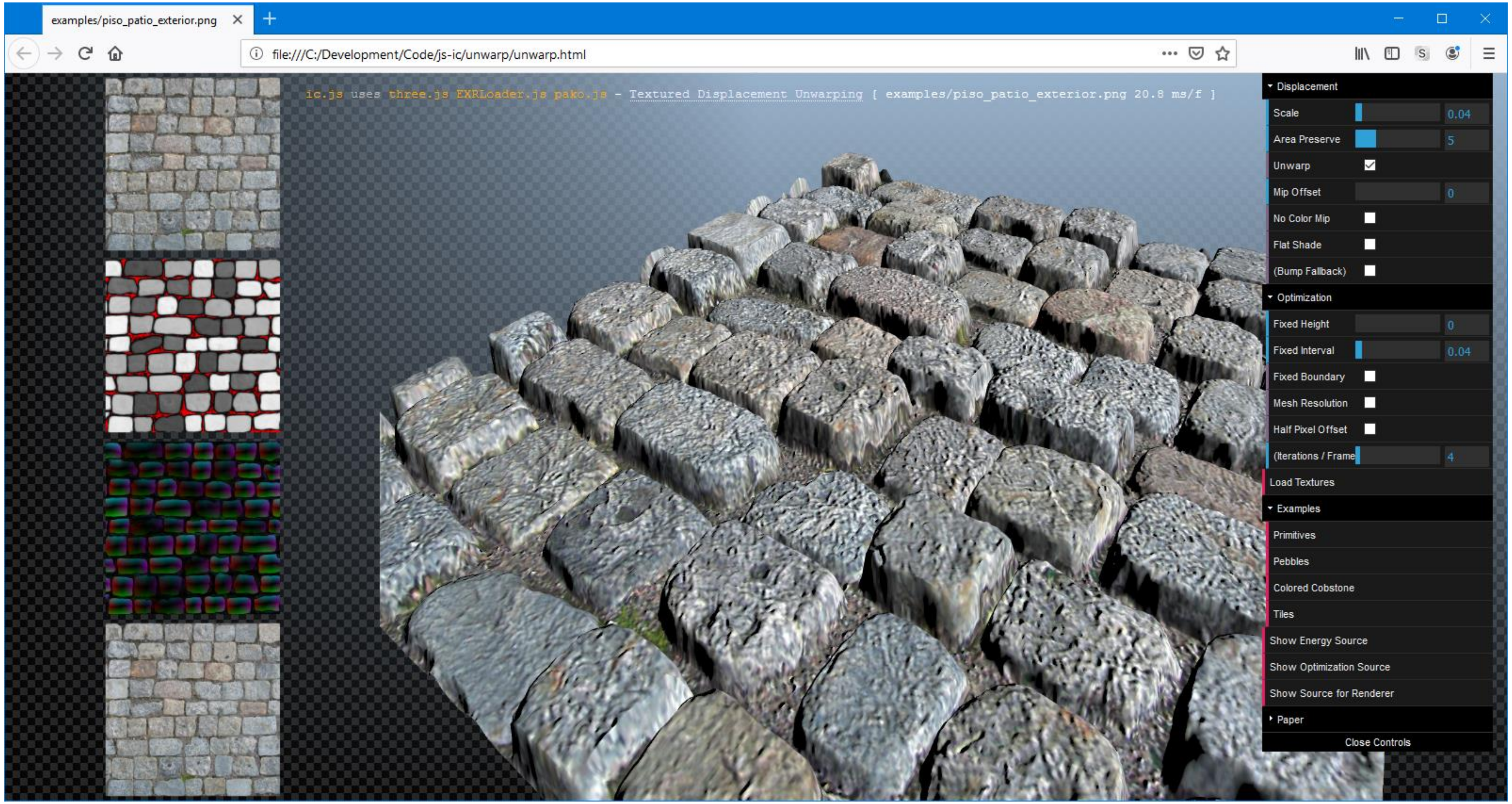


Corrected A (u + c(u))

Example Code / Interactive Demo

HPG 2019

- Example implementation: https://github.com/tszirr/ic.js/tree/master/unwarp



examples/piso_patio_exterior.png

file:///C:/Development/Code/js-ic/unwarp/unwarp.html

ic.js uses three.js EXRLoader.js pako.js - Textured Displacement Unwarping [examples/piso_patio_exterior.png 20.8 ms/f]

Displacement
- Scale 0.04
- Area Preserve 5
- Unwarp ☑
- Mip Offset 0
- No Color Mip □
- Flat Shade □
- (Bump Fallback) □

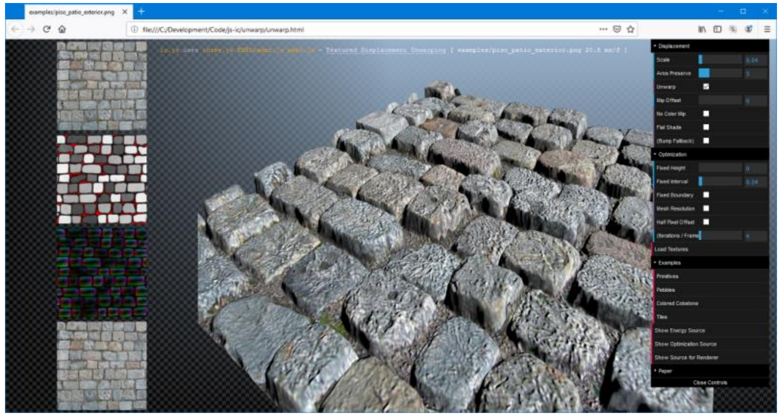
Optimization
- Fixed Height 0
- Fixed Interval 0.04
- Fixed Boundary □
- Mesh Resolution □
- Half Pixel Offset □
- (Iterations / Frame 4

Thank you!

- Questions?

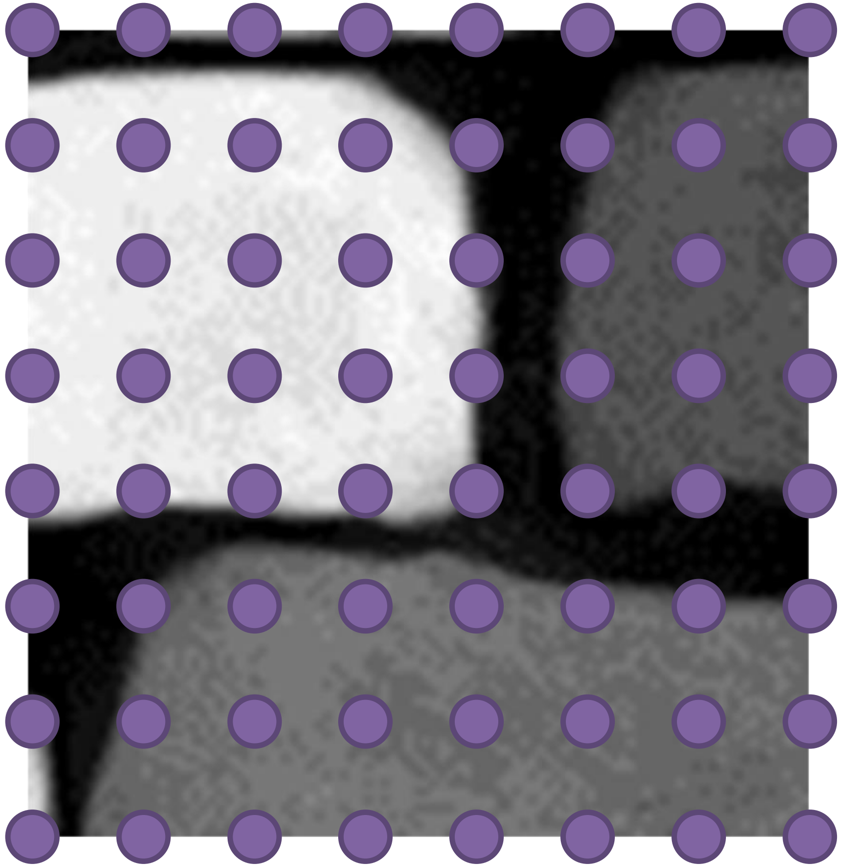
- Contact
  - **Tobias Zirr**
    tobias.zirr@alphanew.net
    Twitter: @alphanew
    http://www.alphanew.net
  - **Tobias Ritschel**
    http://www.homepages.ucl.ac.uk/~ucactri

Project sources:
http://www.alphanew.net

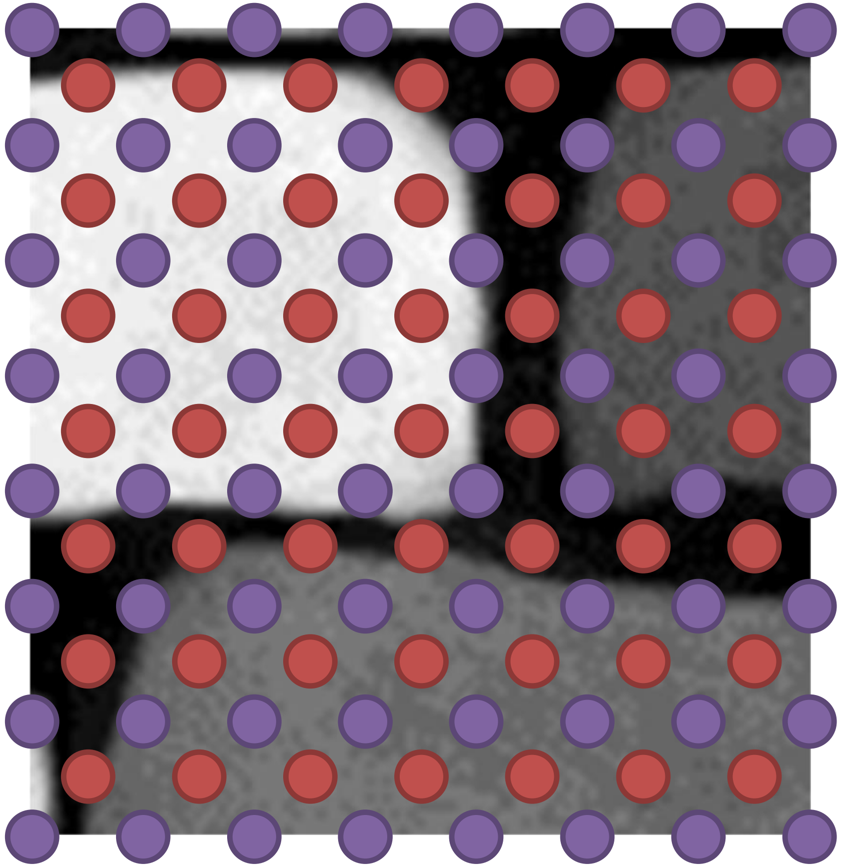


Implementation Tips

- Default evaluation of displacement: on texel boundaries

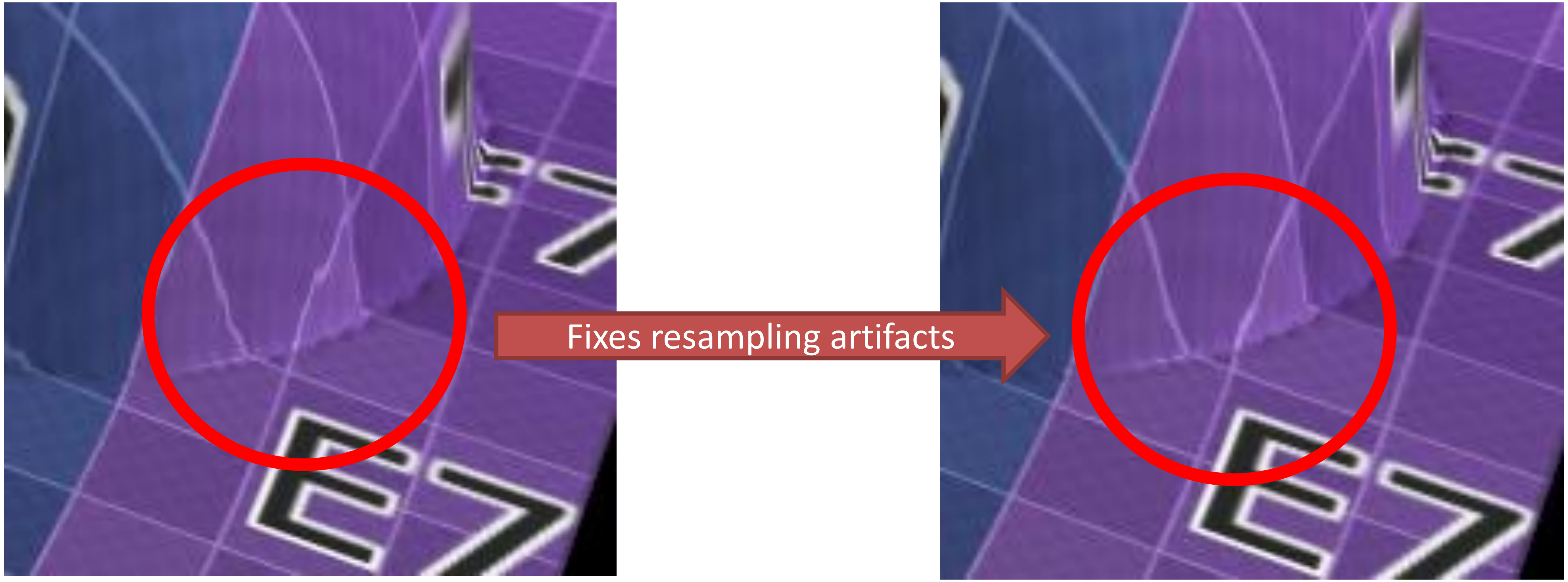


- Default evaluation of correction offsets: on texel centers



Correct with Half Texel Offset

- Match undistortion with displacement grid
  - Use half texel offset
    - Apply during correction optimization
    - Undo during correction evaluation



Fixes resampling artifacts