Seamless Compressed Textures

Andrea Maggiordomo and **Marco Tarini**
Università degli Studi di Milano "La Statale"

PROBLEM

Texture mapping requires UV-maps, and UV-maps (in general) require texture seams; texture seams (in general) cause small visual artifacts in rendering; these can be prevented by careful, slight modifications of a few texels around the seam.
Unfortunately, GPU-based texture compression schemes are lossy and introduce their own slight modifications of texture values, nullifying that effort.
The result is that texture compression may reintroduce the visual artifacts at seams.
We modify a standard texture compression algorithm to make it aware of texture seams, resulting in compressed textures that still prevent the seam artifacts.

RELATED WORK 1

The problem of **visual artifacts at texture seams** has been identified [7] and countered using several approaches. The most obvious countermeasure is to limit the number of seams and carefully place them in areas where the artifacts will be less evident or less disturbing. This difficult to formalize objective is implicitly sought by many digital artists constructing the UV-map, and, incidentally, is one of the characteristics making automatic UV-map construction differ from the ones authored by digital artists. Automatic approaches have been proposed. [5] constructs UV-maps by restricting texture seams to the axis-aligned case, which is artifact-free; [3] tweaks both an existing UV-map and an existing texture image to minimize artifacts. In our work, we consider the technique [6], which is a simple approach that achieves similar results by only adjusting the texel values. Unfortunately, all these approaches rely on careful selection of texel values, which are impacted by texture compression.

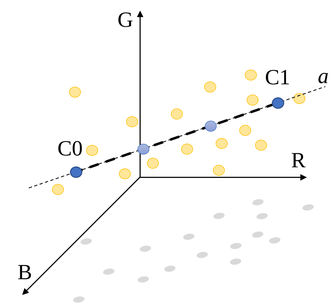
RELATED WORK 2

Texture images consume GPU-RAM, which is a scarce resource, making **texture compression** essential.
Compression schemes for textures differ from other image compression schemes in that individual texel values must be accessible (during per-fragment processing) in constant time, while keeping the image compressed in GPU-RAM.
As a consequence, texture compression schemes are lossy, with a fixed compression ratio, and a comparably unfavorable trade-off between quality loss and space occupancy. Examples include simple schemes such as palette (or color-map) compression, quantization of RGB channels, and more sophisticated schemes such as 3Dc, A8L8 (often employed for normal maps), and DXT schemes (S3TC) [2].
Here. we consider the case of the DXT1 schema, which is an S3TC schema designed for RGB images, popular in video games etc, featuring a 1:6 compression ratio. There are several algorithms to compress an image under this schema, like [1], which we adapt for our purposes.

METHODOLOGY

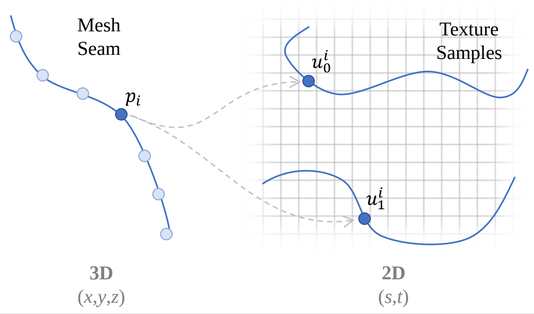
Standard DXT1 compression
- Block-based Encoding (4×4 texel blocks)
- 2 Colors (16-bit RGB 5:6:5)
- 4 possible linear interpolations
- 64 bits per block (32 + 2×16)

For each 4×4 *texel block:*
(1) Consider 16 texels as pts in RGB space
(2) Find 1st principal axis *a*
(3) Project pixels on *a*
(4) Find extremes C0, C1 of segment
(5) Quantize per-texel parametric position on segment



Standard Seam Masking
- Makes texture color match along mesh seams
- Considering texture bilinear interpolation

Given input texture `I`*:*
(1) Sample mesh seam-edges at uniform discrete intervals p_i
(2) Compute color conservation equations
(3) Compute seamless equations
(4) Solve min $E_C + \alpha E_S$ (we use $\alpha = 2$); as a global Least Squares system with texels as 3D variables



New Compressed Seam Masking
- Makes texture color match along mesh seams
- Considering texture bilinear interpolation *and* DXT1 decoding

Given input texture `I`*:*
(1) Compute a seam-masked texture with *Standard Seam Masking*
(2) Compress it in DXT1 using any *standard DXT1 compression*
(3) Solve global LS system anew, but with **per-block colors** C0, C1 as 3D variables

linear functions in C0, C1 (for a given block encoding)

Color conservation equations (1 per texel)

```
texel(i, j) = I(i, j).rgb
```

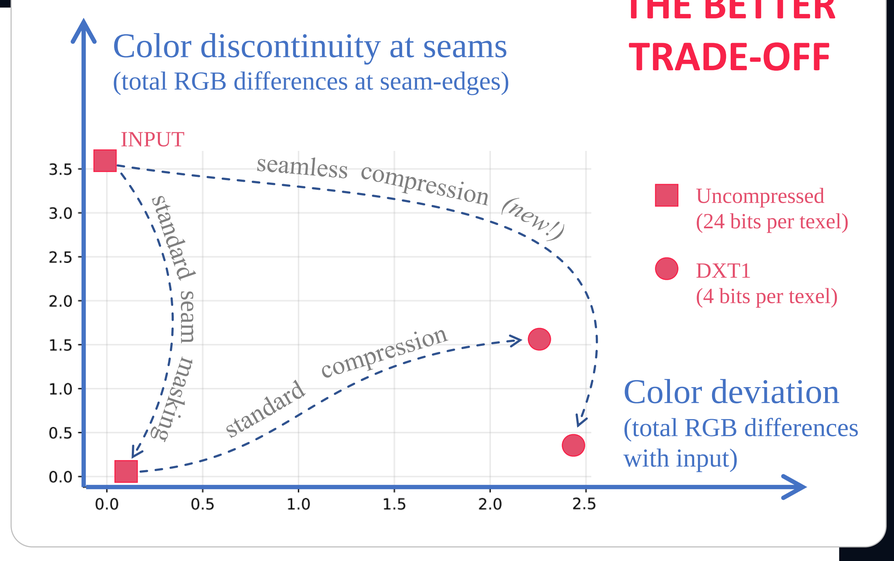
Seam energy equations (1 per sample p_i)

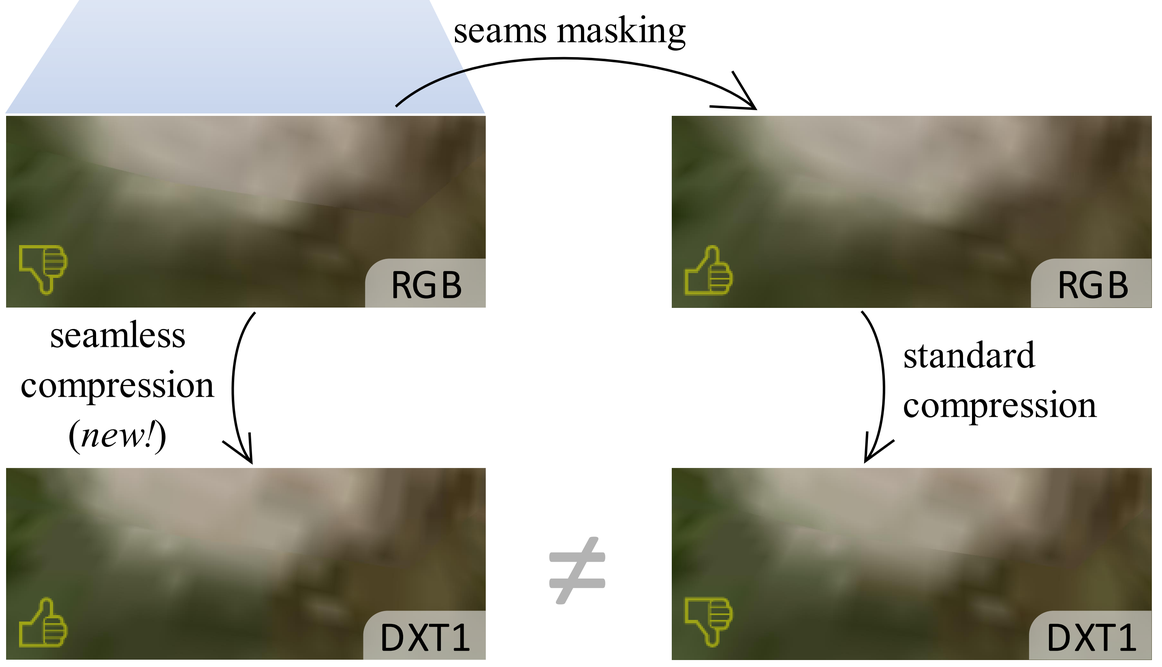
```
texture2D(u_0^i) = texture2D(u_1^i)
```

bilinear combinations of texel(i,j)

RESULTS


DATASET FROM [4]





Can you spot the artifact at the texture seam?
Get our implementation from GitHub: https://github.com/maggio-a/seamless-compressed-textures

REFERENCES

[1] BROWN S.: Dxt compression techniques (squish). blog post - https://www.sjbrown.co.uk/posts/ dxt-compression-techniques/, Jan 2007.
[2] IOURCHA K. I., NAYAK K. S., HONG Z.: System and method for fixed-rate block-based image compression with inferred pixel values, Sept. 21 1999. US Patent 5,956,431.
[3] LIU S., FERGUSON Z., JACOBSON A., GINGOLD Y.: Seamless: Seam erasure and seam-aware decoupling of shape from mesh resolution. ACM Transactions on Graphics (TOG) 36, 6 (Nov. 2017), 216:1–216:15. doi:10.1145/3130800.3130897.
[4] MAGGIORDOMO A., PONCHIO F., CIGNONI P., TARINI M.: Real-world textured things: A repository of textured models generated with modern photo-reconstruction tools. Computer Aided Geometric Design 83 (2020), 101943. doi:https://doi.org/10.1016/j.cagd.2020.101943.
[5] RAY N., NIVOLIERS V., LEFEBVRE S., LEVY B.: Invisible seams. In EUROGRAPHICS Symposium on Rendering Conference Proceedings (2010), Lawrence J., Stamminger M., (Eds.), Eurographics, Eurographics Association. 1
[6] SEBASTIAN SYLVAN: Fixing texture seam with linear leastsquare. blog post - https://www.sebastiansylvan.com/post/LeastSquaresTextureSeams/, 2015.
[7] YUKSEL C., LEFEBVRE S., TARINI M.: Rethinking texture mapping. Comput. Graph. Forum 38, 2 (2019), 535– 551. URL: https://doi.org/10.1111/cgf.13656, doi: 10.1111/cgf.13656.

UNIVERSITÀ DEGLI STUDI DI MILANO

EG'22 TheEvent