Adaptive Scalable Texture Compression

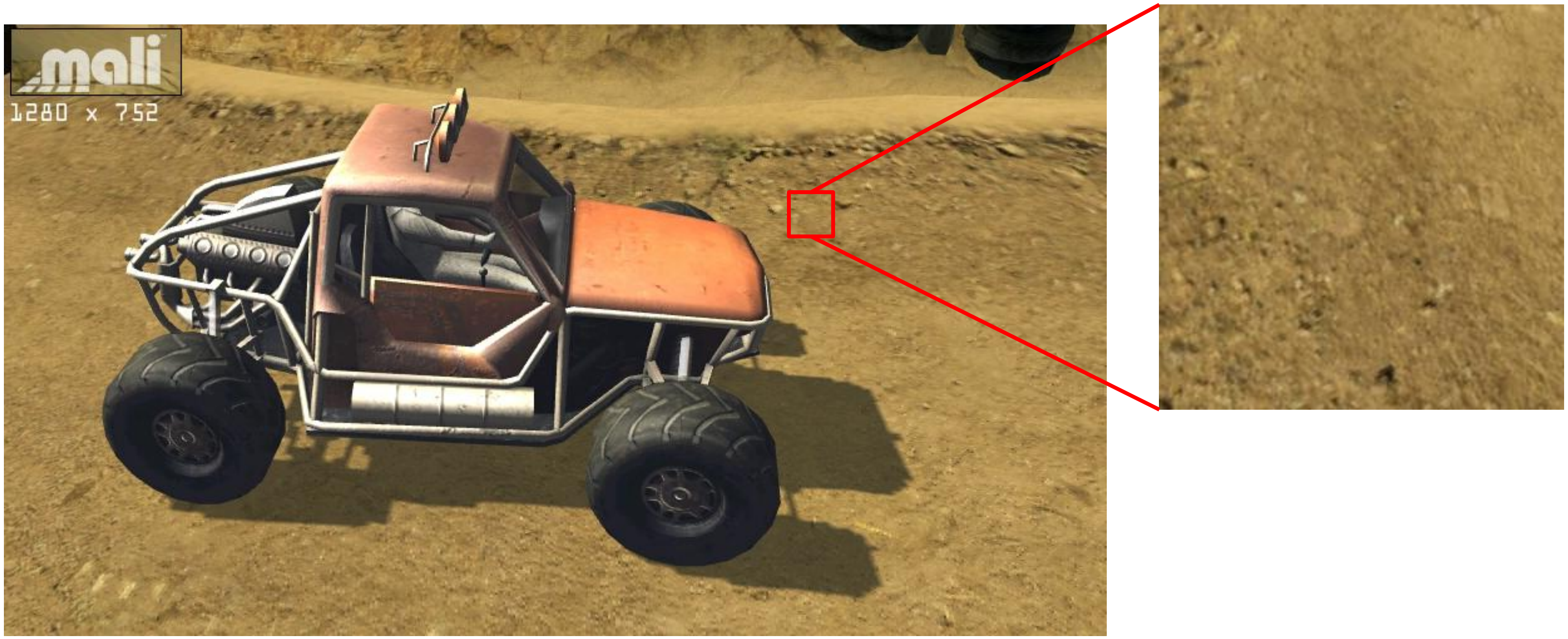
J. Nystad[1], A. Lassen[1], A. Pomianowski[2], S. Ellis[1], T. Olson[1]

[1]ARM [2]AMD

High Performance Graphics 2012

Motivation

Textures are fundamental in modern graphics

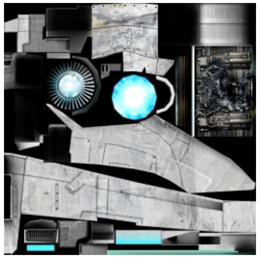


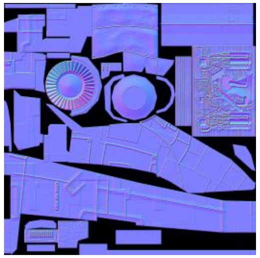
But textures are big…

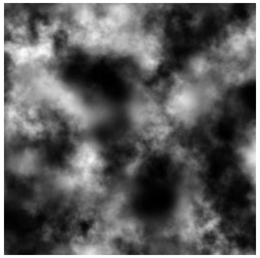
- Major contributors to memory bandwidth and power consumption
- Solution: Texture Compression [e.g. Knittel et al 96, Beers et al 96]

Texture Use Cases

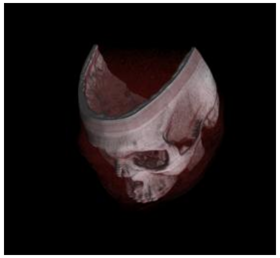
Textures are used for many different things:


Reflectance

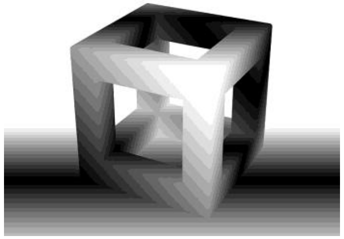

Normals


Height


Illuminance


Density (3D)


Lighting environment

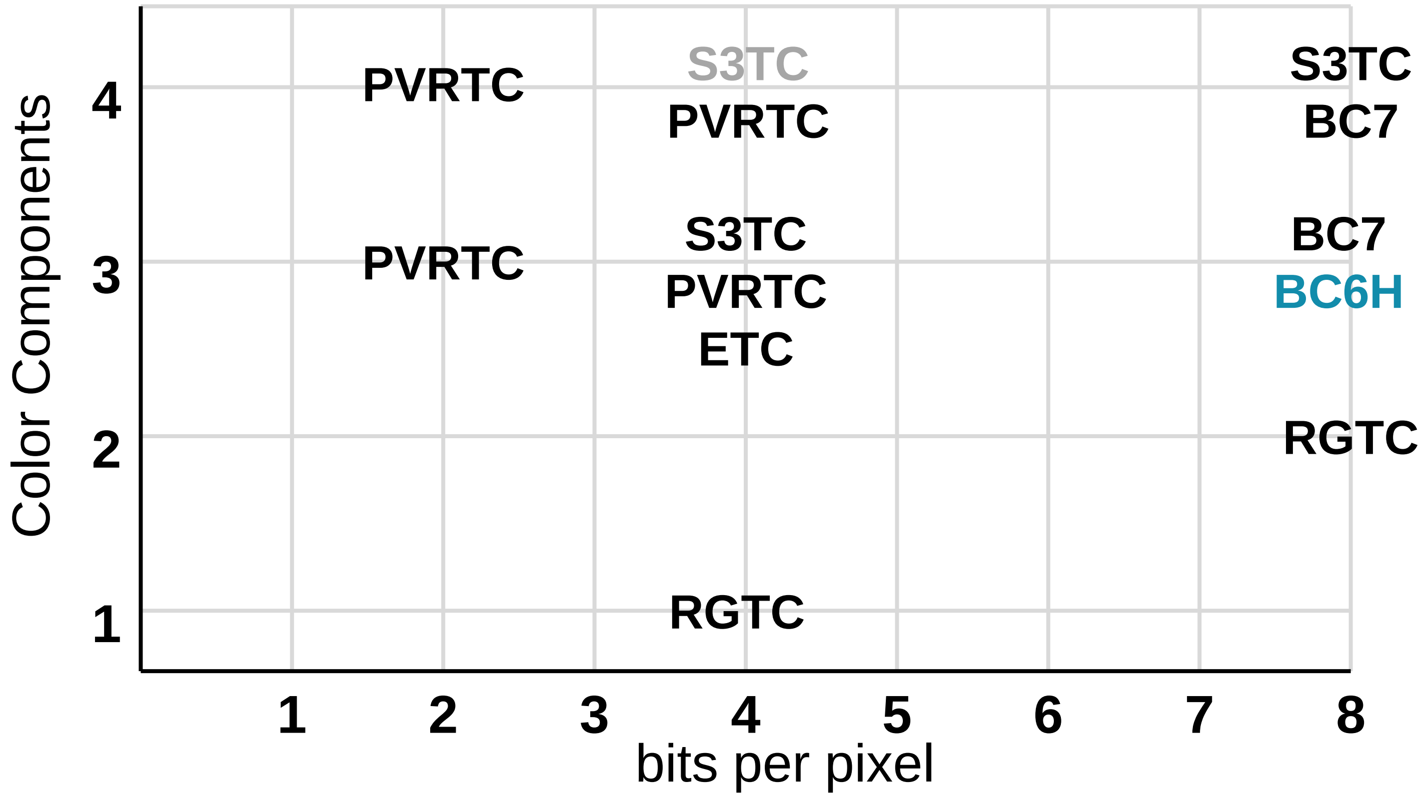

Depth

…and each use case has different requirements

- Number of color components
- Dynamic range (LDR vs HDR)
- Dimensionality (2D vs 3D)
- Quality

The Problem

No existing format addresses all use cases



Our Solution

Adaptive Scalable Texture Compression

Design Goals

- Cover the widest possible range of use cases
- High quality

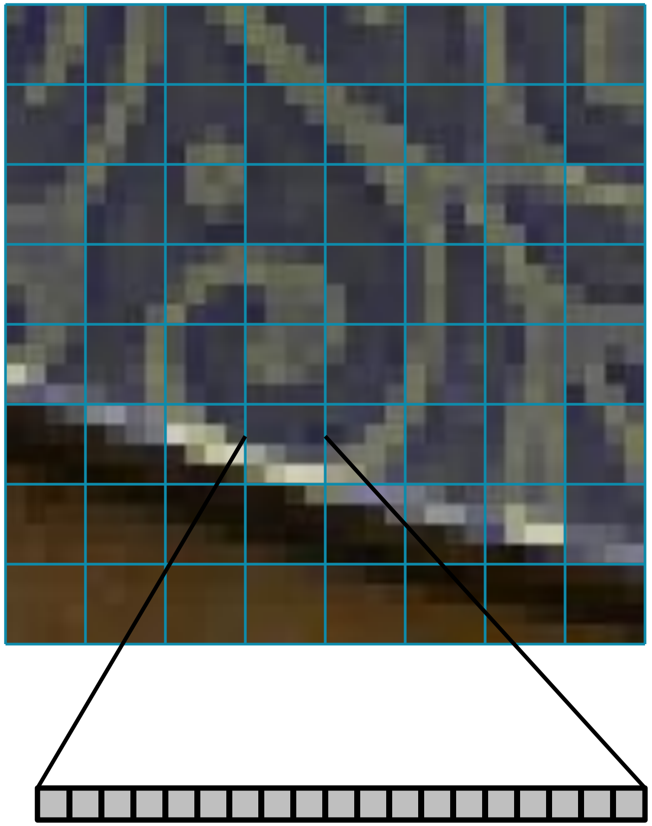
Functionality

- *Adaptive:* # color components, dynamic range specified per-block
- *Scalable:* from 8bpp down to <1bpp in fine steps
- Orthogonal: 1 to 4 color components at any bit rate
- General: both 2D and 3D, both LDR and HDR
- Area-efficient, hardware-friendly

Related Work: The Standard Paradigm

Block-based, fixed-rate

- BTC [Delp & Mitchell 79]
- S3TC / DXTn [Iourcha et al 99]
- BPTC / BC6H+BC7 [Microsoft]
- ETC1 / ETC2 [Ström et al 05,07]
- …many others, including ASTC



Block Contents

- Color space(s)
- Per-texel color selectors
- Control information

Key Advantage

- Can decode any texel in constant time with one memory access

Other Approaches

Vector Quantization [Beers et al 96]

- Better quality
- Not hardware-friendly due to need for codebooks

Variable-rate coding [Inada and McCool 06]

- Better quality
- Requires multiple memory references, special cache architecture

PVRTC [Fenney 03]

- Reduced block artifacts
- Requires multiple memory references

Representing bounded integer values

Problem: Given sequences of equiprobable values in the range [0..N-1], find an efficient encoding that…

- Provides random access with compact decode hardware
- Works for many values of N

Standard solution: packed binary

- Efficient (optimal) for $N = 2^k$

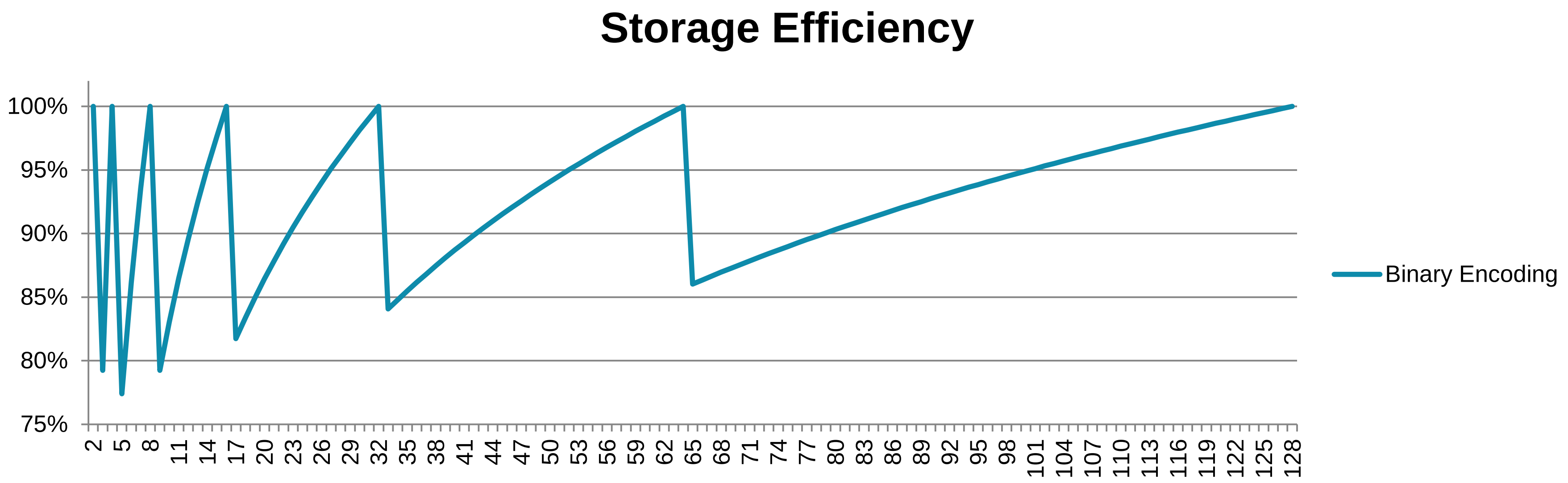
New solution: *bounded integer sequence encoding* (BISE)

- Optimal for $N = 2^k$
- Near optimal for $N = 3\times2^k$, 5×2^k

Storage Efficiency

Equiprobable values in range [0..N-1] stored in B bits/value

- Each value contains $\log_2(N)$ bits of information
- Storage efficiency is $\log_2(N)/B$

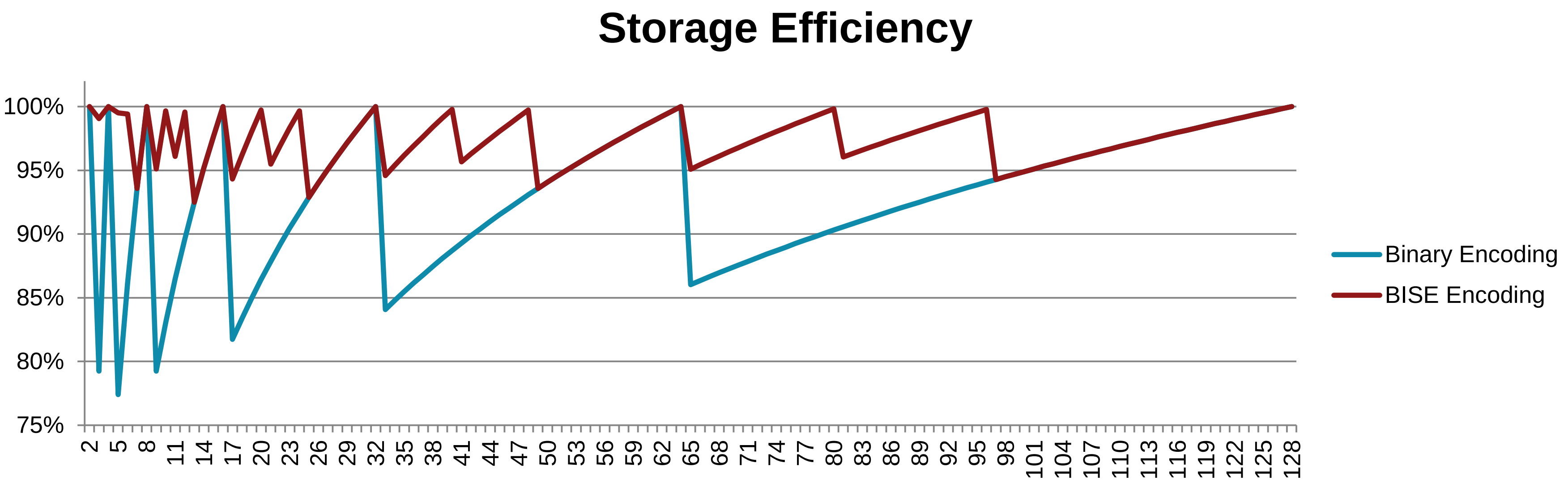


Binary encoding provides widely spaced operating points

Storage Efficiency

Equiprobable values in range [0..N-1] stored in B bits/value

- Each value contains $\log_2(N)$ bits of information
- Storage efficiency is $\log_2(N)/B$



BISE adds two optimal value ranges between each pair of powers of two

ASTC Bit Rates

Standard block-based paradigm

- Generalized to 3D
- Unusually large number of block sizes

2D Bit Rates				3D Bit Rates			
4x4	8.00 bpp	10x5	2.56 bpp	3x3x3	4.74 bpp	5x5x4	1.28 bpp
5x4	6.40 bpp	10x6	2.13 bpp	4x3x3	3.56 bpp	5x5x5	1.02 bpp
5x5	5.12 bpp	8x8	2.00 bpp	4x4x3	2.67 bpp	6x5x5	0.85 bpp
6x5	4.27 bpp	10x8	1.60 bpp	4x4x4	2.00 bpp	6x6x5	0.71 bpp
6x6	3.56 bpp	10x10	1.28 bpp	5x4x4	1.60 bpp	6x6x6	0.59 bpp
8x5	3.20 bpp	12x10	1.07 bpp				
8x6	2.67 bpp	12x12	0.89 bpp				

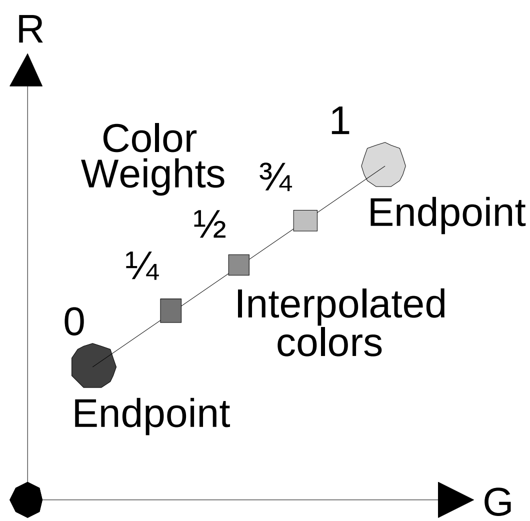
Color spaces and color selectors

Color spaces defined by pairs of color endpoints

- cf S3TC, PVRTC, BPTC
- Endpoints can be LDR or HDR, 1 to 4 color components

Per-texel weights interpolate between the endpoints

- Number of values a weight can have is variable
- Interpolation is linear for LDR, pseudo-logarithmic for HDR



Texel Weights

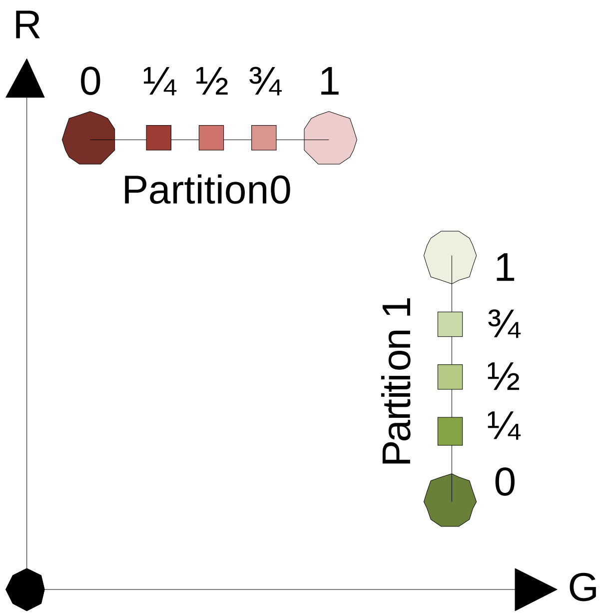
0	0	¼	¾
0	0	¼	1
¼	¼	½	1
¾	1	1	1

stored with block

Partitions and Multiple Color Spaces

Each block has an optional partition function (cf BPTC)

- Function maps each texel in the block to a partition
- Each partition has its own color space



Texel Weights

0	0	¼	½
0	0	¼	¾
¼	¼	½	1
¾	1	1	1

stored with block

Partition Function

1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1

Maps texels to partitions

Partition Functions

Need lots of partition functions

- Too many to store as tables

Procedural partition functions

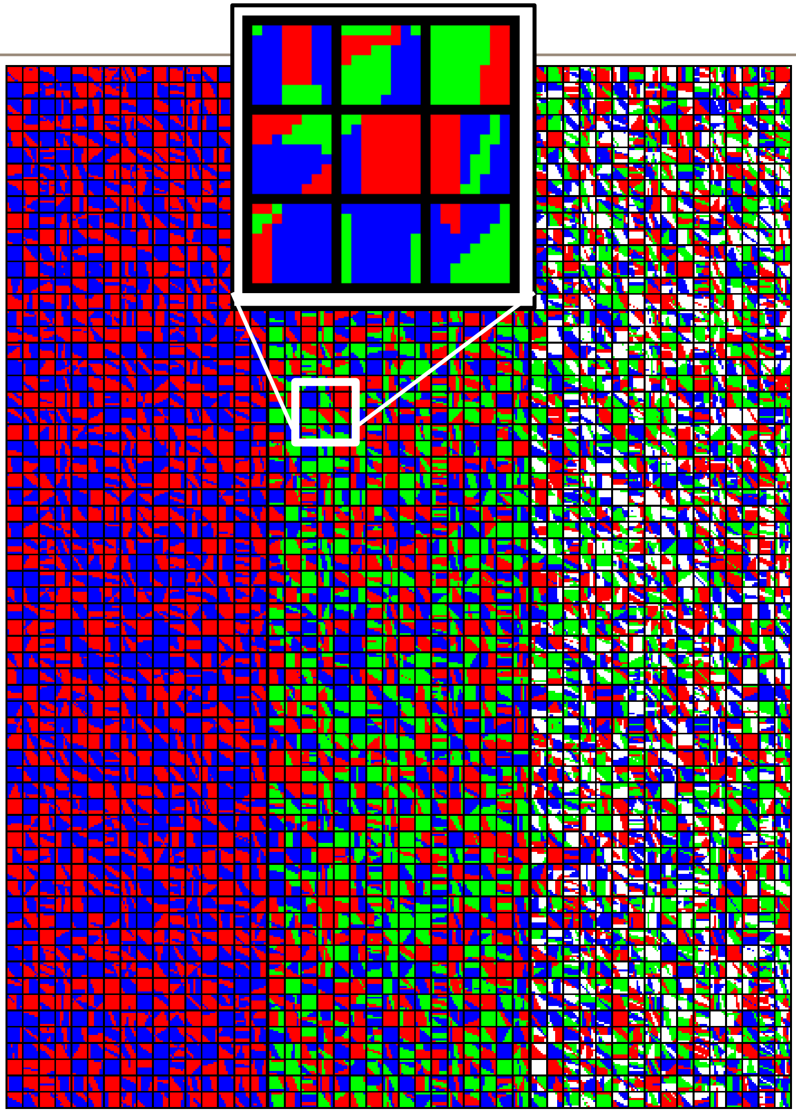
- Selected by 10-bit per-block index plus # of partitions
- Derived from HW random number generator

Advantage

- 3072 functions

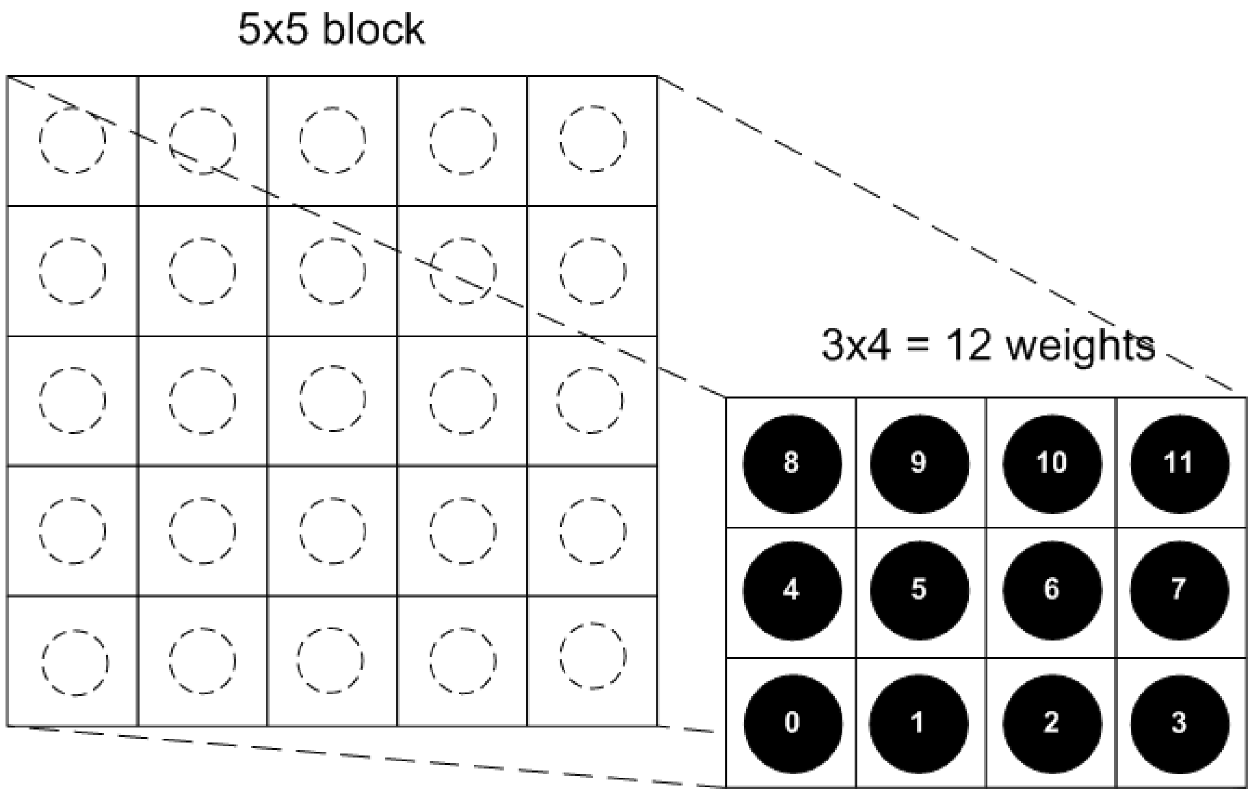
Disadvantage

- Functions are suboptimal



Partition patterns for 8x8 block size (false colored to show partition ID)

Computing Per-Texel Weights



Scaling Infill

- Color weights for a block are specified as MxN arrays
- Weights obtained by bilinear (2D) or simplex (3D) interpolation

Block Encoding

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Index Mode											PC		Partition Index (if needed)										Color Mode								

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Color Endpoint Data (variable width) *Fill direction →*																															

64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
¦ Extra Color Mode Data ¦																															

96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
←Fill direction Color Weight Data (variable width)																															

Index Mode

- Color weight array dimensions
- Range of values used for weights

Partition Information

- Partition count
- Partition function ID

Color Space Mode(s)

- Number of channels
- Dynamic range
- Color endpoint encoding

Color Endpoint Data

Color Weights

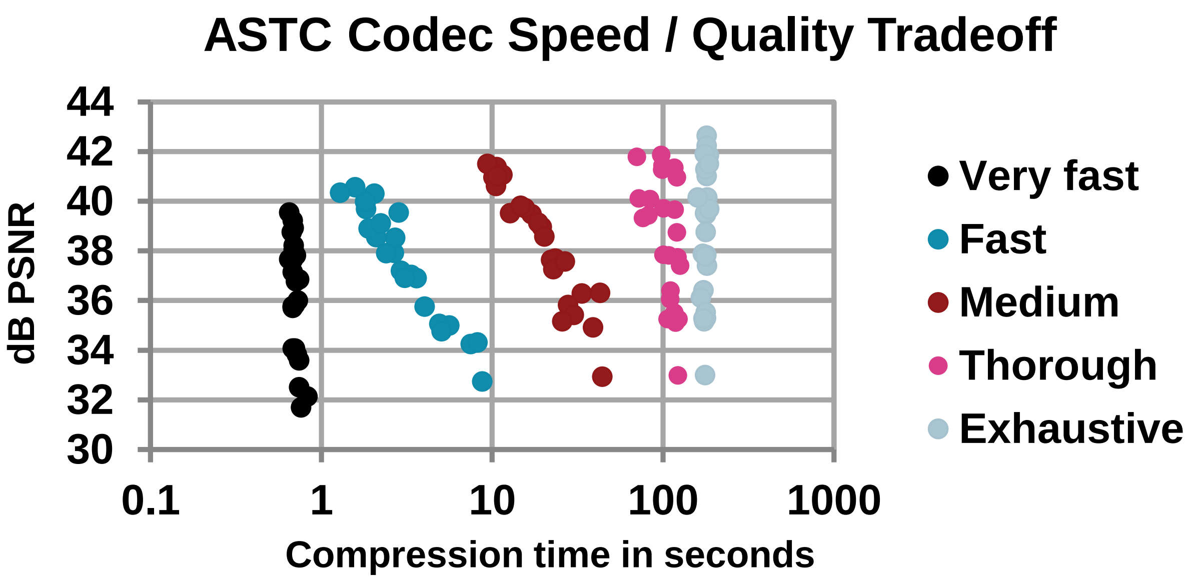
Implementation

Implemented in synthesizable RTL

- About 2x the size of our BPTC implementation

Experimental codec

- Branch-and-bound search
- Choice of heuristics to control speed/quality tradeoff

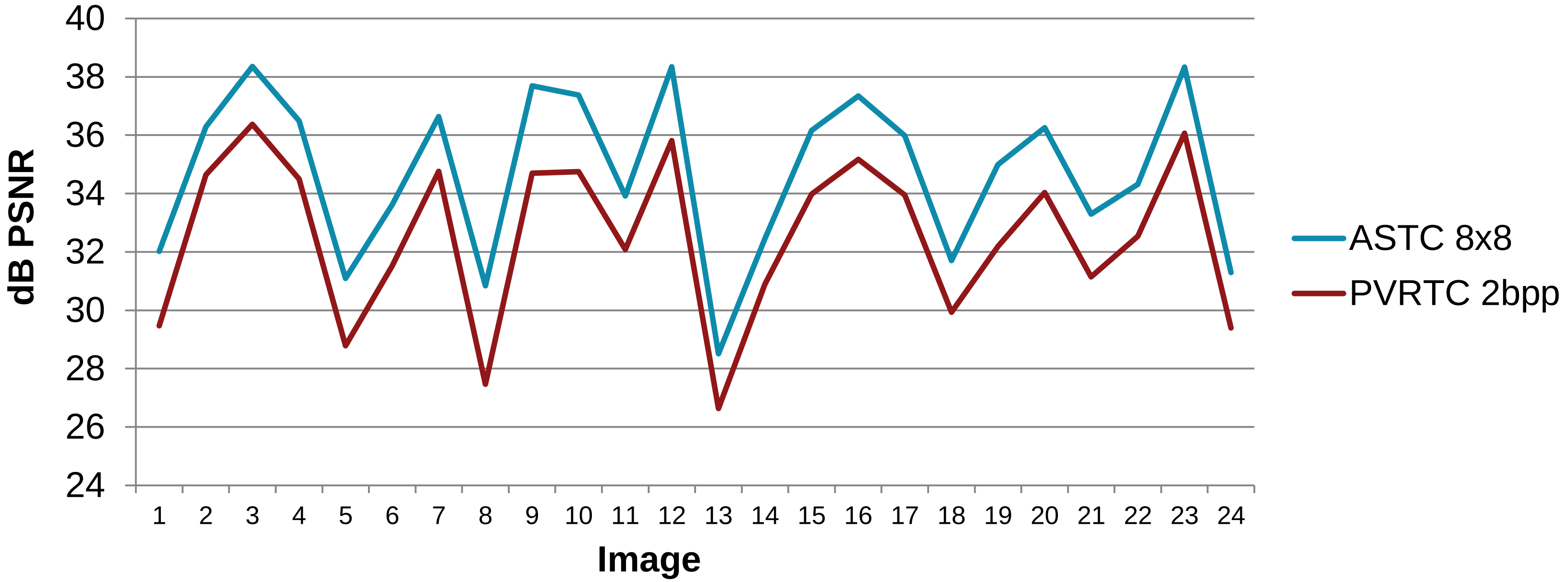


Quality Comparison – RGB LDR 2bpp

"Kodak" test set

- 24 natural RGB images
- PSNR comparison

ASTC vs PVRTC 2bpp:

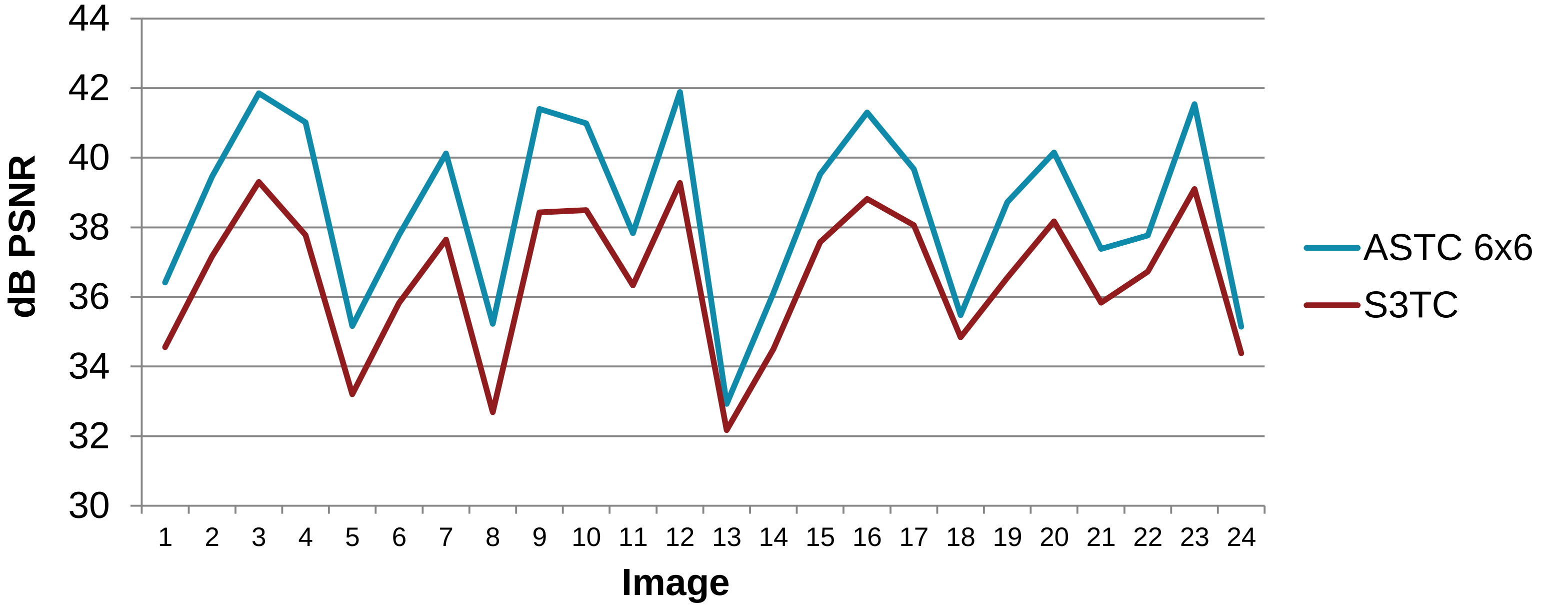


Quality Comparison – RGB LDR 4bpp

"Kodak" test set

- 24 natural RGB images
- PSNR comparison

ASTC at 3.56 bpp vs S3TC at 4bpp:



Quality Comparison – RGB LDR 8bpp

“Kodak” test set

- 24 natural PSNR images
- PSNR comparison

ASTC vs BC7 at 8bpp:

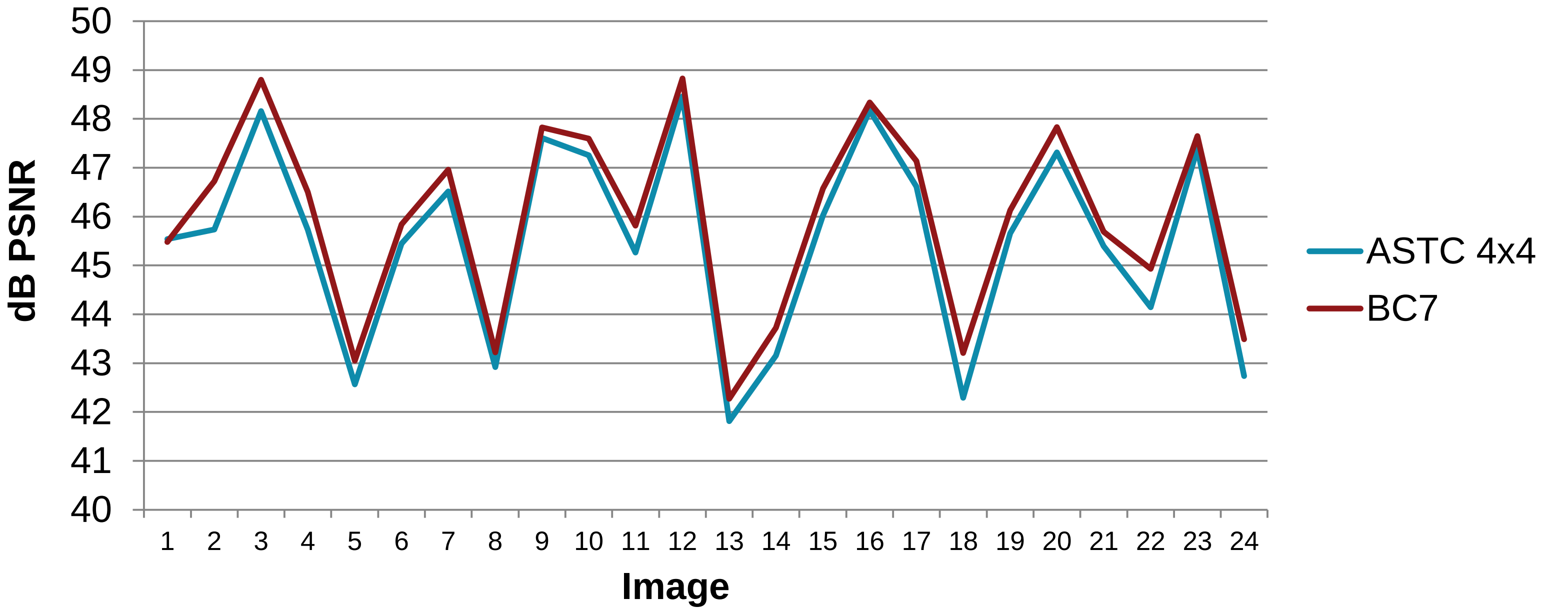


Image Comparisons – RGB LDR 2bpp

original



original



Image Comparisons – RGB LDR 2bpp

ASTC 8x8

PVRTC 2bpp

Image Comparisons – RGB LDR 4bpp

original

original

Image Comparisons – RGB LDR 4bpp

ASTC 6x6

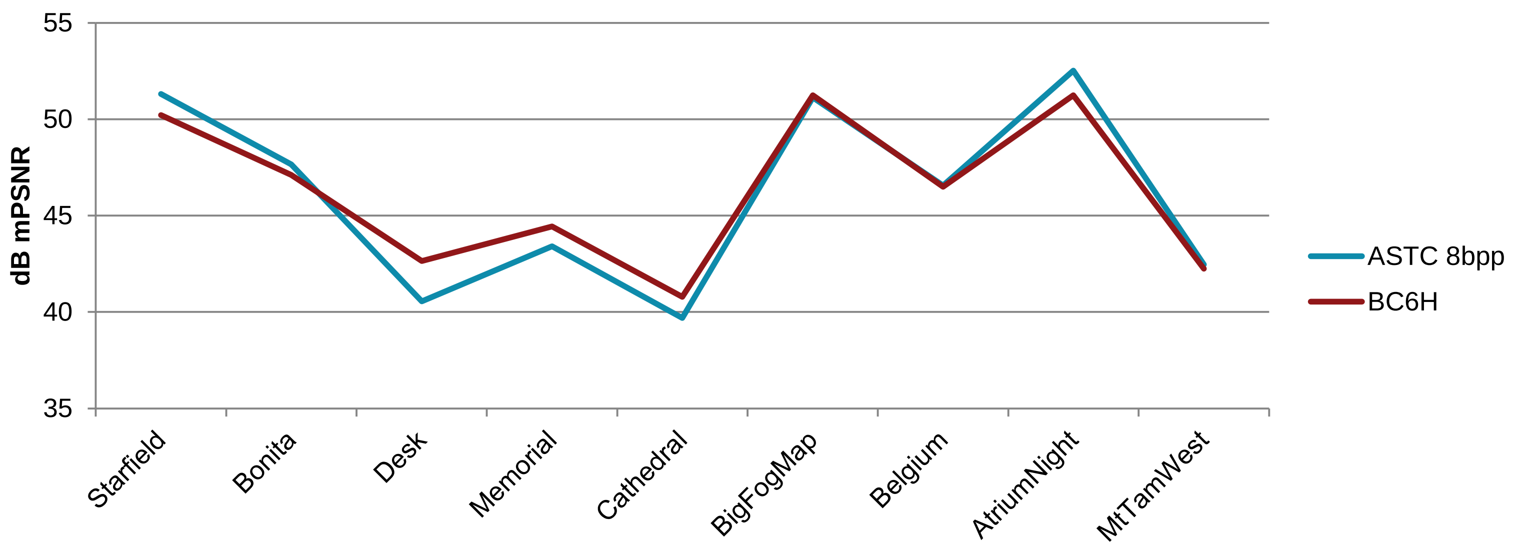
S3TC (4bpp)

Quality Comparison – RGB HDR

OpenEXR example images

- mPSNR comparison
- Using exposure ranges from Munkberg et al 2006

ASTC 8 bpp vs BC6H 8bpp:



Contributions

Novel techniques

- Bounded Integer Sequence Encoding
- Scaling Infill
- Procedural Partition Functions

A new texture compression format: ASTC

- Unprecedented flexibility
  - Wide range of bit rates
  - Orthogonal choice of number of color components
  - LDR and HDR, 2D and 3D
- Very high quality
  - As good or better than formats in commercial use

Future Work

Encoder Improvements

- HDR
- Block artifact reduction

Quality evaluation / improvement on other use cases

- Normals
- 3D texture applications

Codec speed improvements

- Embeddable encoder

Acknowledgements

Valuable discussions and feedback:

- Konstantine Iourcha, Cass Everitt, Nick Penwarden, Jacob Ström, Walt Sullivan, and many others
- The HPG reviewers

Image Credits

- http://en.wikipedia.org/wiki/File:CTSkullImage.png
- http://en.wikipedia.org/wiki/File:Cubic_Structure_and_Floor_Depth_Map_with_Front_and_Back_Delimitation.jpg
- http://en.wikipedia.org/wiki/File:Heightmap.png
- http://r0k.us/graphics/kodak/