Texture Compression in Memory- and Performance-Constrained Embedded Systems

Jens Ogniewski
Information Coding Group
Linköpings University

LiU expanding reality

Outline

Background / Motivation

DXT / PVRTC texture compression

the ePUMA Platform

Decoding speed comparison

Encoding

Quality comparison

Conclusion / Future work

Background / Motivation

- Modern embedded systems (smartphones, (mobile) multimedia players, settop-boxes etc.) are supposed to handle more and more multimedia applications
  - Including computer graphics
- These systems are designed towards low cost, low power consumption
  - The number of electronical components and their complexity need to be restricted

=> small, shared memories, shared memory bus

=> avoid memory accesses

Background / Motivation

- Use texture compression for graphic purposes
  - Minimize usage of memory and its bus
  - Enable high quality 3D graphics

- Requested characteristic:
  - Low decoding complexity, high decoding speed
  - Random access
  - Lossy compression acceptable
  - Encoding speed minor issue

DXT

- Based on S3TC, the first commercial texture compression available
- Uses vector compression
  - n Vectors are represented by a smaller number of m vectors
- Divide texture into 4x4 blocks
- Each pixel in this block is represented by 1 of 4 different color vectors
- 2 of these vectors are “given” vectors, the other 2 are linearly interpolated, i.e.

$$c3 = \frac{1}{3} * c1 + \frac{2}{3} * c2$$
$$c4 = \frac{1}{3} * c2 + \frac{2}{3} * c1$$

DXT

- The two “given” vectors are directly encoded, using RGB 565
  => each of these two uses 16 bits
- Each pixel needs 2 bit to select which of the four colors should represent it
  => 32 bits need for indexing
- 64 bits are needed to encode one 4x4 block
  - Compression factor of 6 (compared to uncompressed RGB888)
- Different DXT versions available
  - Only differ in how they handle transparency
  - Not further discussed here

PVRTC

- Used by Imaginations Technology
  - Building highly successful GPUs for embedded systems
- Makes use of the inbuilt linear interpolation hardware
- Encode 2 images whose height and width are both ¼ of the height / width of the original image
- Decoding: upscale these two images to the original resolution using linear interpolation
- Each pixel can again choose between four values:
  - The values on its position in either of the upscaled images
  - A linear combination of these two different values
    - Weights of ⅜ and ⅝ are used

PVRTC

- The two images are encoded in RGB555 format
  - Again, 16 bits are used to encoded one color vector
  - 1 bit is reserved to signalize the use of transparency
- Again, 2 bits are needed by each pixel for indexing

=> same compression rate as DXT

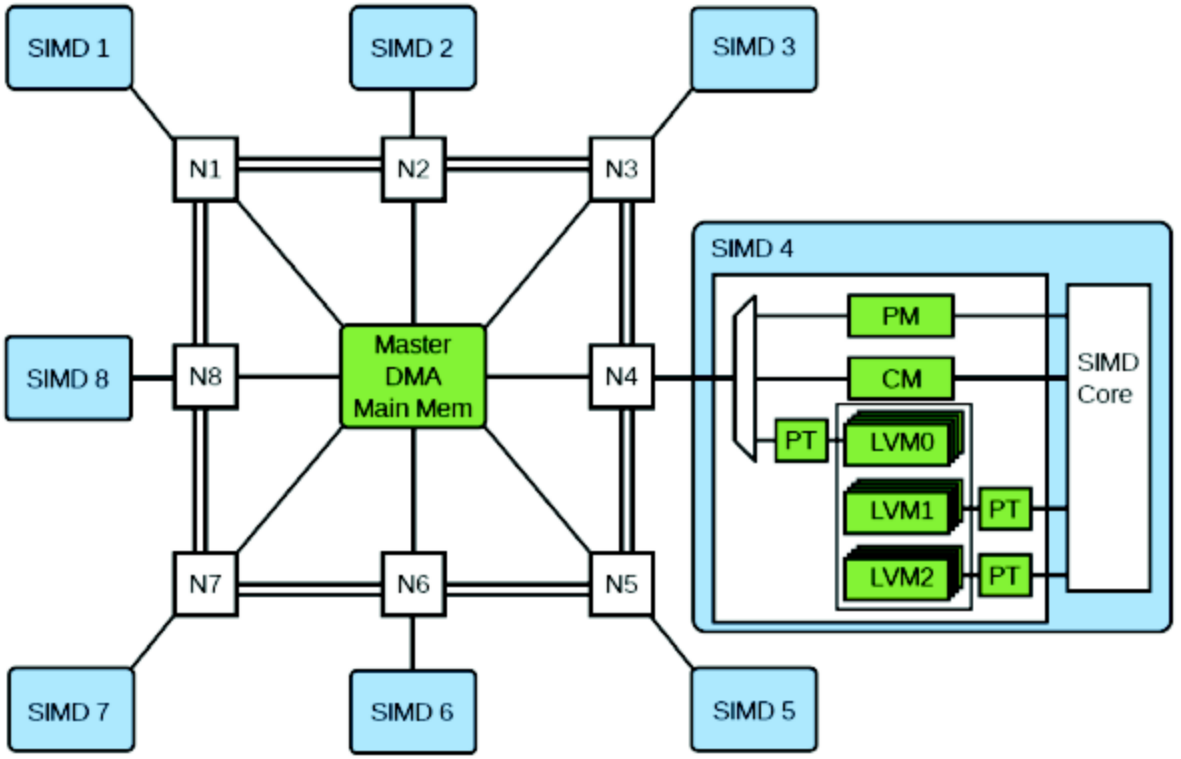
- But more complicated decoding process
- Worse random access
  - 8 color vectors need to be loaded to be able to decode one block, i.e. 160 bits for one block (instead of only 64 as DXT)
- Mode with a compression rate of 12 also available
  - Not considered here

The ePUMA Platform

- Embedded Parallel DSP with Unique Memory architecture
- Aimed for low cost, low power embedded systems
  - Designed mostly for low energy consumption
- Expected to handle 3D graphics as well
  - But no dedicated texture memory, no hardware support for interpolation
  - Also limited memory and memory bandwidth
- Texture compression needed

The ePUMA Platform

- 8 SIMD cores, one master processor
- Communication between the cores via ringbus
- Via DMA otherwise
  - Also used to access the main memory



The ePUMA Platform

- Each SIMD:
  - 8 16bit datapaths, can be used as 4 32bit datapaths instead
    - 8bit not supported yet, but will be in future implementations
  - 80 kB memory
    - Can hold 6 64x64 textures, 26 32x32 textures
    - The number of textures may be further reduced by memory alignment, mipmapping, or other data that needs to be stored

=> Dire need for fast, efficient texture compression

Decoding Speed (in cycles)

Task	DXT1	PVRTC
Unpacking Calculation of color the vectors Waiting for pipeline to finish **total**	3 3 8 **14**	3 99 8 **110**

- DXT does not take full advantage of SIMD parallelism

=> in real application difference might be even more pronounced

Encoding

- 3 different encoder:
  - PVRTC reference encoder (provided by Imaginations Technology)
  - SQUISH: DXT, using a clusterfit approach, an open-source implementation of NVIDIAs reference encoder
  - Line matching: an own encoder for DXT

Encoding

- Line matching
  - Consider the color-values of the block as points in the RGB color space
  - The 4 color-vectors used in DXT form a line in this space

    Then:
  - Find line in color space with minimizes the sum of distances between the line and the color values
    - Using a standard singular-value-decomposition
  - Search for candidates of the two directly encoded vectors on this line
  - Do a local search around the final candidates to find a(local) optimum
    - Needed due to approximations and rounding errors during the SVD and the search along the line
    - Also if color values in the block are very similar
    - Removing this step does not significantly reduce the objective quality or the encoder runtime

Encoding

- Squish
  - Uses principal axis instead of line which minimizes distances

Quality Comparison

Squish – PSNR: 31.24, SSIM: 0.984

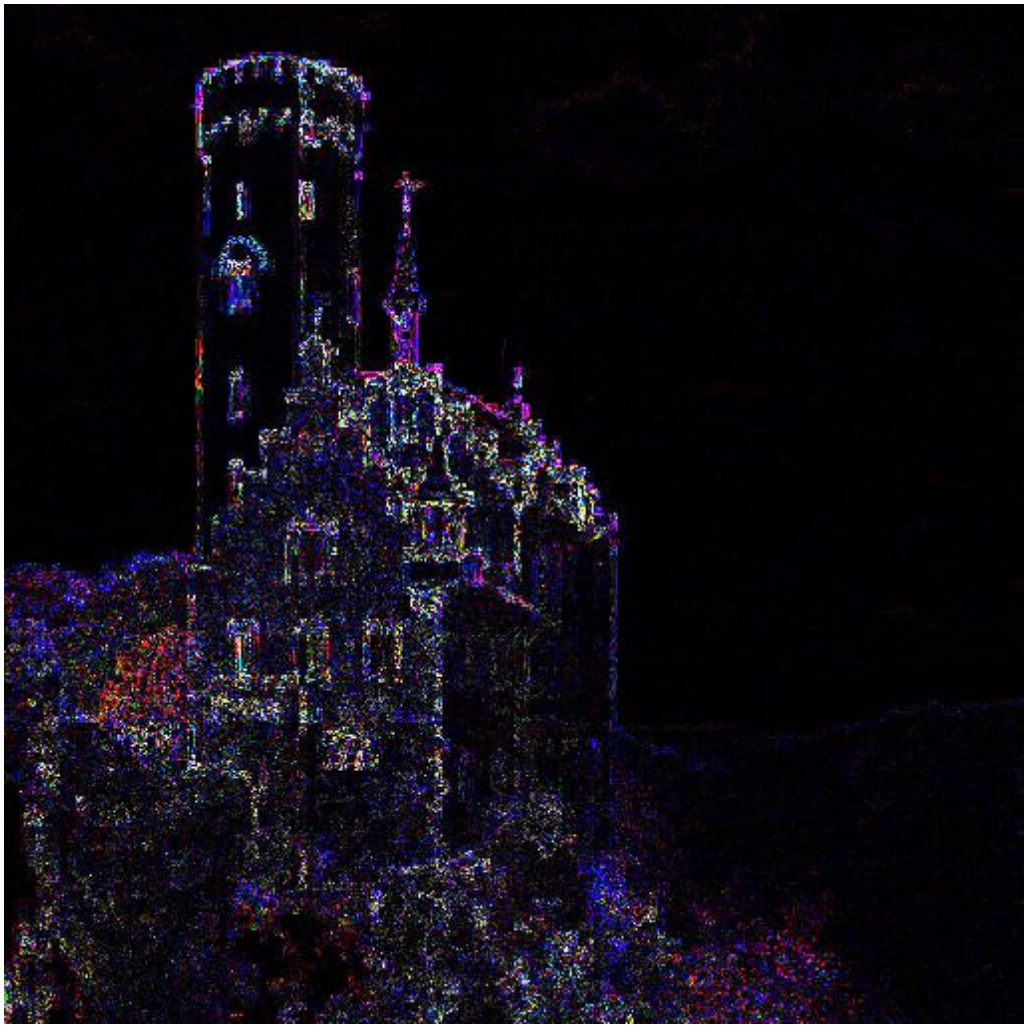
Quality Comparison

Line matching – PSNR: 31.51, SSIM: 0.985

Quality Comparison

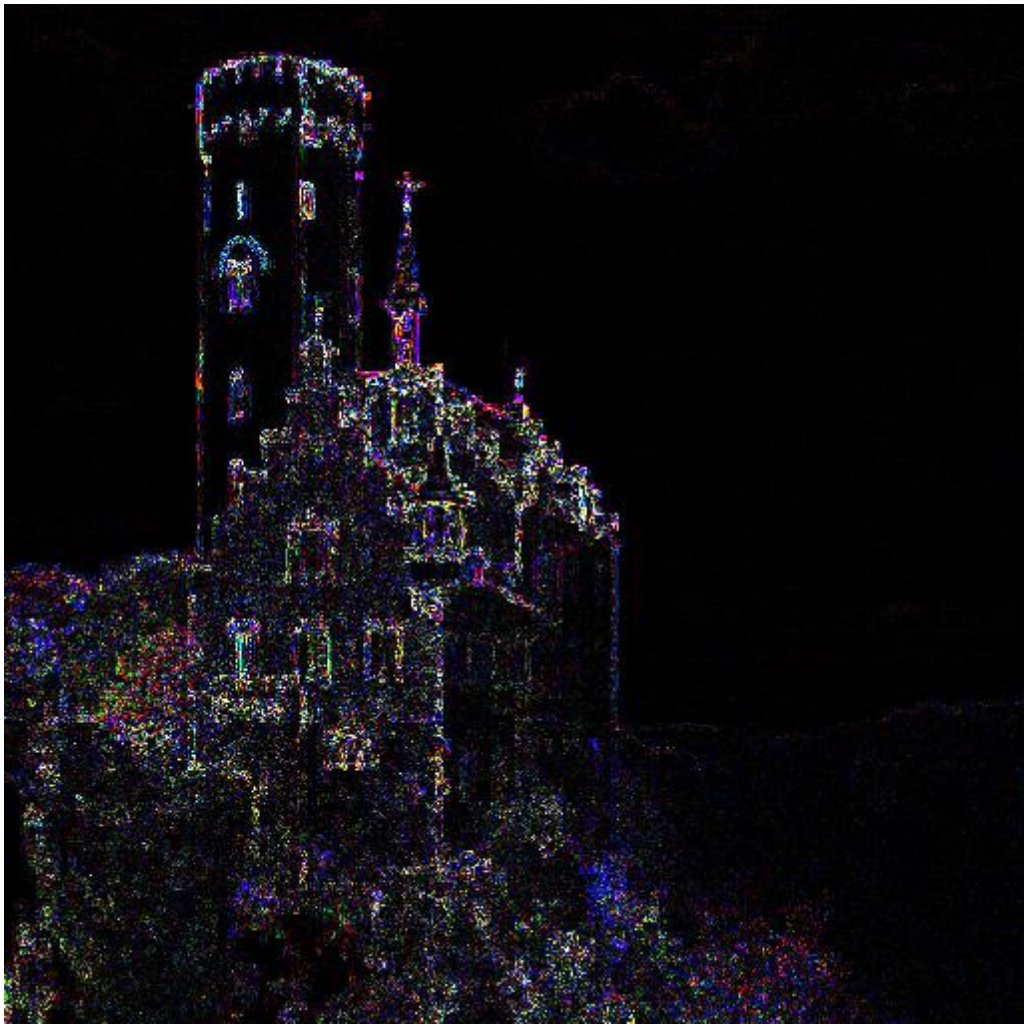
PVRTC – PSNR: 30.63, SSIM: 0.982

Quality Comparison



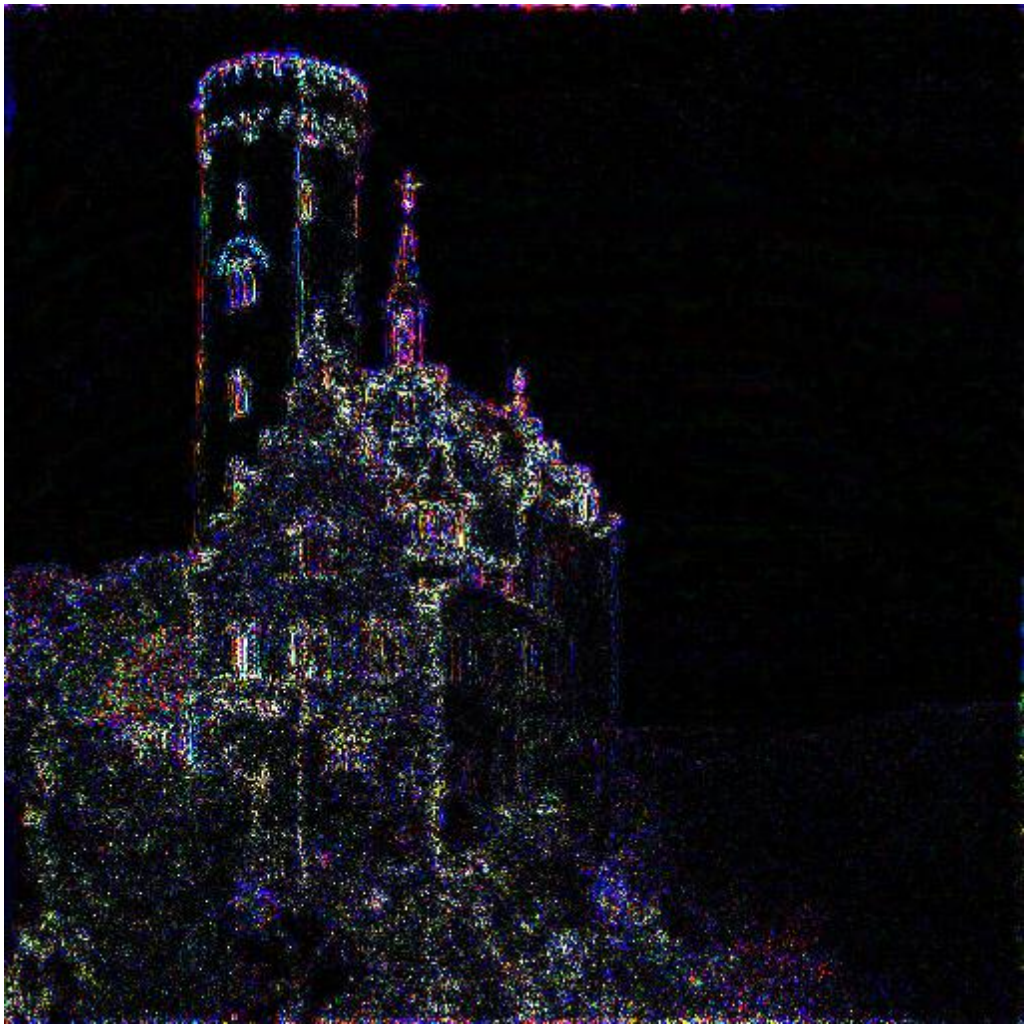
Squish – PSNR: 31.24, SSIM: 0.984

Quality Comparison



Line matching – PSNR: 31.51, SSIM: 0.985

Quality Comparison



PVRTC – PSNR: 30.63, SSIM: 0.982

Quality Comparison, Results (PSNR)

Task	Squish	Line matching	PVRTC
Bark	27.86	28.04	27.86
Brick	30.94	31.21	30.73
Buildings	30.73	30.59	29.78
Clouds	34.60	35.51	34.45
Fabric	25.21	25.20	24.82
Flowers	29.62	29.86	29.98
Food	26.50	26.61	27.18
Grass	23.96	23.95	23.62
Images	30.47	30.71	30.47
Leaves	26.46	26.70	26.81
Metal	22.63	22.54	21.90
Misc	29.05	29.26	28.90
Paintings	26.81	27.16	27.70
Sand	29.07	29.21	29.46
Stone	28.36	28.64	28.48
Terrain	33.98	34.59	34.20
Tile	30.42	30.66	30.25
Water	31.81	32.23	31.65
WheresWaldo	25.59	25.81	26.30
Wood	31.32	31.77	31.77
total	**28.39**	**28.60**	**28.45**

Quality Comparison, Results (SSIM)

Task	Squish	Line matching	PVRTC
Bark	0.980	0.981	0.980
Brick	0.980	0.981	0.978
Buildings	0.983	0.983	0.978
Clouds	0.971	0.974	0.967
Fabric	0.978	0.980	0.974
Flowers	0.982	0.983	0.982
Food	0.981	0.981	0.982
Grass	0.981	0.981	0.980
Images	0.973	0.975	0.976
Leaves	0.981	0.983	0.982
Metal	0.977	0.977	0.972
Misc	0.980	0.981	0.980
Paintings	0.974	0.976	0.977
Sand	0.979	0.981	0.981
Stone	0.974	0.978	0.973
Terrain	0.981	0.984	0.985
Tile	0.980	0.981	0.977
Water	0.980	0.982	0.979
WheresWaldo	0.981	0.983	0.984
Wood	0.982	0.984	0.984
total	**0.979**	**0.980**	**0.979**

Conclusion

- Texture compression is essential for embedded systems for 3D graphic applications
- Decoding of DXT in realtime is possible on the ePUMA platform, enabling the use of 3D graphics albeit the comparably small local memory
- PVRTC does not deliver an improved quality over DXT with the current standard-encoder
- The presented line matching encoder for DXT delivered the overall best quality of all presented encoder, in terms of both PSNR and SSIM
- Even if the PVRTC standard-encoder could be improved, a possible quality gain will probably still not justify the much higher decoding complexity

Future Work

- Optimize encoder (especially towards speed, removal of blocking artifacts)
- Transparency
- Comparison with Ericsson texture compression
- Optimized Texture compression scheme
- Longterm goal: a full 3D renderer on ePUMA

Questions?


Thank you very much!
www.liu.se