SIGGRAPH ASIA 2020 VIRTUAL
Pre-recorded sessions: From 4 December 2020
Live sessions: 10 – 13 December 2020
SA2020.SIGGRAPH.ORG
#SIGGRAPHAsia | #SIGGRAPHAsia2020

QuickETC2: Fast ETC2 Texture Compression using Luma Differences

Jae-Ho Nah, LG Electronics
nahjaeho@gmail.com

Introduction & Related Work

Texture Compression

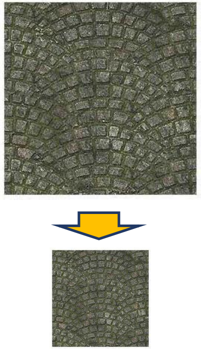
- For high-quality rendering, a large amount of high-resolution textures in an app is now common
- Let's think their compression burden in the following example
  - 5,000 4K×4K-sized uncompressed textures = 83G pixels
  - Assumed encoding speed: 1M pixels/s
  - Time required for compression: 23.3 hours!
- Slow texture compression can be a bottleneck in S/W development
  - Increase the necessity of fast encoders

Real-time Texture Compression

- In some scenarios, **real-time** texture compression is required
  - Due to limited time budgets, a huge amount of textures, or a response speed

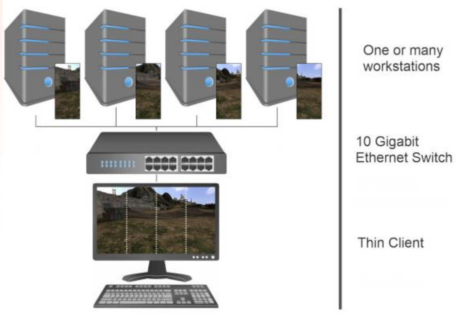
3D reconstruction
[Easterbrook et al., CVMP 2010]

GIS tools
[Krajcevski and Manocha, i3D/JCGT 2014]



Texture resizing
[Nah et al., SIGGRAPH 2018]

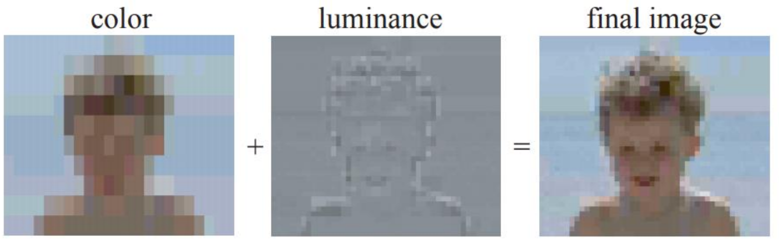
In-game video capture
[Kemen, OpenGL Insights]



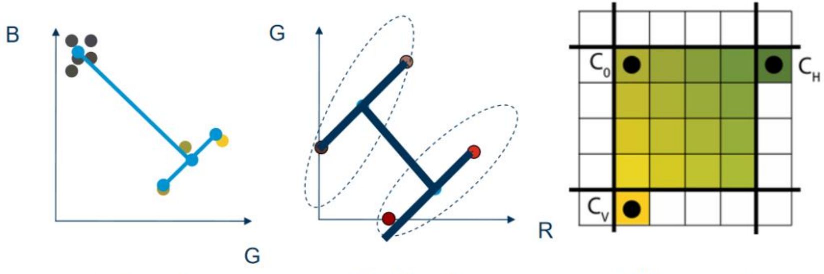
In-home streaming
[Pohl et al., FedCSIS 2017]

ETC Codecs

- Standard texture codecs
  - Microsoft BC1-7 (Desktop), ETC1/ETC2/EAC (Android), PVRTC (iOS) & ASTC (Android/iOS)
- ETC1 [Ström and Akenine-Möller, GH 2005]
  - OpenGL ES 2.0 standard
  - Two base chrominance + per-pixel luminance
  - 6:1 compression ratio



- ETC2/EAC [Ström and Petersson, GH 2007]
  - OpenGL ES 3.0 standard
  - Three additional modes: T, H & planar
  - Less block & banding artifacts
  - Alpha support (EAC)



(a) T-mode (b) H-mode (c) Planar mode

SIGGRAPH ASIA 2020 VIRTUAL

ETC Compressors

ETCPACK [Ericsson 2005-2018]

- Reference encoder
- Fast & slow modes
- Integrated into
  - Mali Texture Compression Tool
  - PVRTexTool
  - AMD Compressonator
  - Unity (normal option)

Etc2Comp [Google and Blue Shift 2016-2017]

- Faster multi-threaded encoder
- Fine quality control
- Integrated into
  - Unity (best option)

etcpak [Taudul and Jungmann 2013-2020]

- Ultra-fast, multi-threaded, SIMD-optimized encoder
- Partial ETC2 support (planar only)
- Integrated into
  - Unity (fast option)

QuickETC2

- Goals
  - Fastest ETC2 compression speed
  - Full ETC2 support (T, H, and planar) for high quality

- Built upon etcpak 0.7

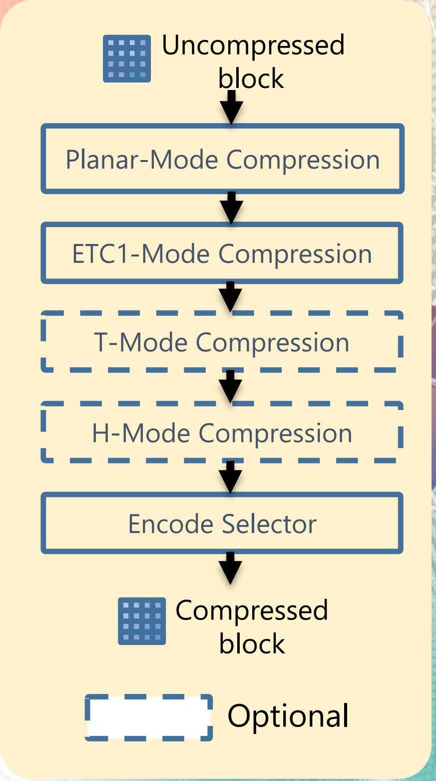
- Two contributions
  - Early compression-mode decision (up to a 3X speedup)
  - Fast T-/H-mode compression algorithm (up to +1dB PSNR)
  - SIMD (SSE/AVX2) optimized

Early Compression-Mode Decision

Traditional ETC2 Encoding

- ETC2 compression on existing encoders
  - Sequentially performs multiple compression in all (supported) ETC1/2 modes (etcpak does not support T- & H-modes)
  - Finally selects a block with the lowest error
  - ETC2 encoding is 1.5X-6X slower than ETC1 encoding

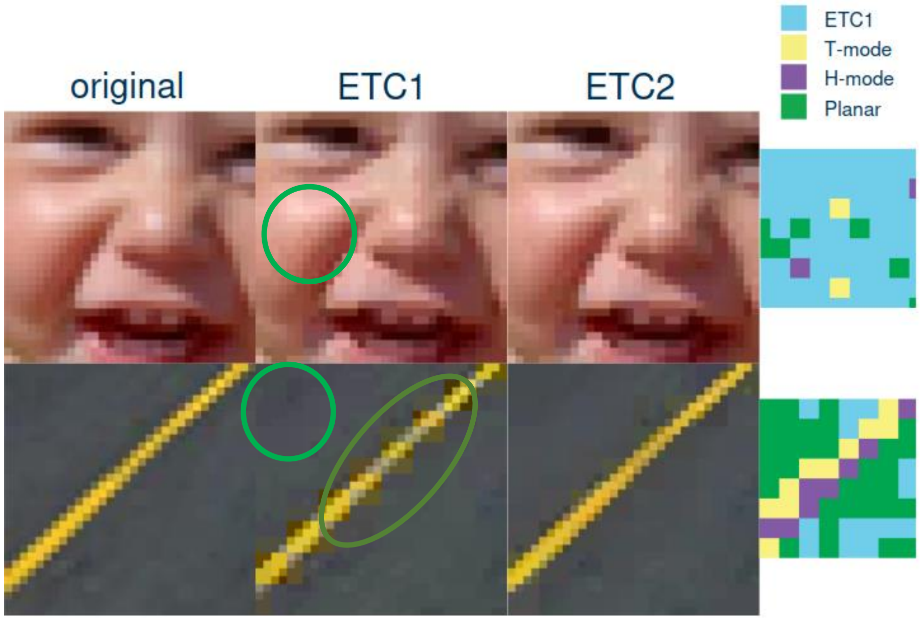
- Our question
  - Can we avoid these duplicated tests for a speedup?



Our Observation

- Three ETC2 modes assist ETC1 in different ways
  - Planar: improves gradients in low-contrast regions
  - T & H: reduce block artifacts in high-contrast regions

- Thus, we expect that
  - We can determine proper compression mode(s) in advance to avoid duplicated tests



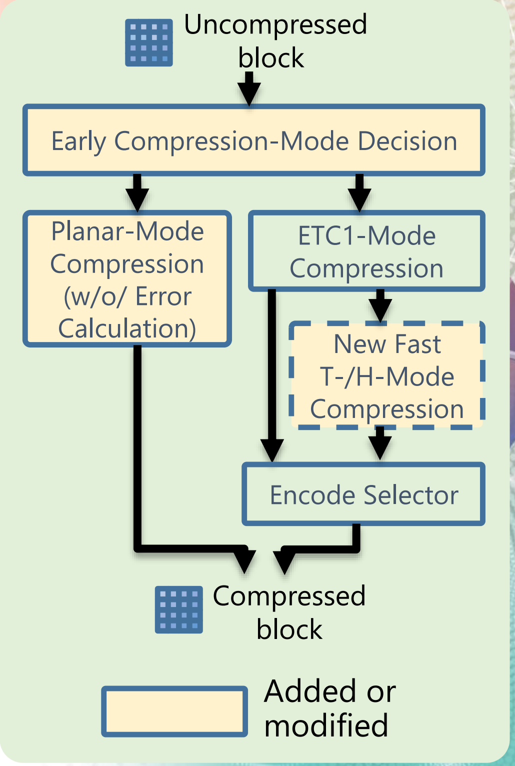
[Ström and Petersson, GH 2007]

Early Compression-mode Decision

- Key idea: block classification according to luma differences (LDs)
  - $Y = 0.299R + 0.587G + 0.114B$

LD range*	Corner pixel check	Compression mode
[0.00, 0.03]	N/A	Planar
(0.03, 0.09]	M and m in opposite corners (M: MaxLuma m: MinLuma)	Planar
(0.03, 0.09]	other cases	ETC1
(0.09, 0.38)	N/A	ETC1
[0.38, 1.00]	N/A	ETC1 & T/H

* Optimal thresholds were determined by our experiments



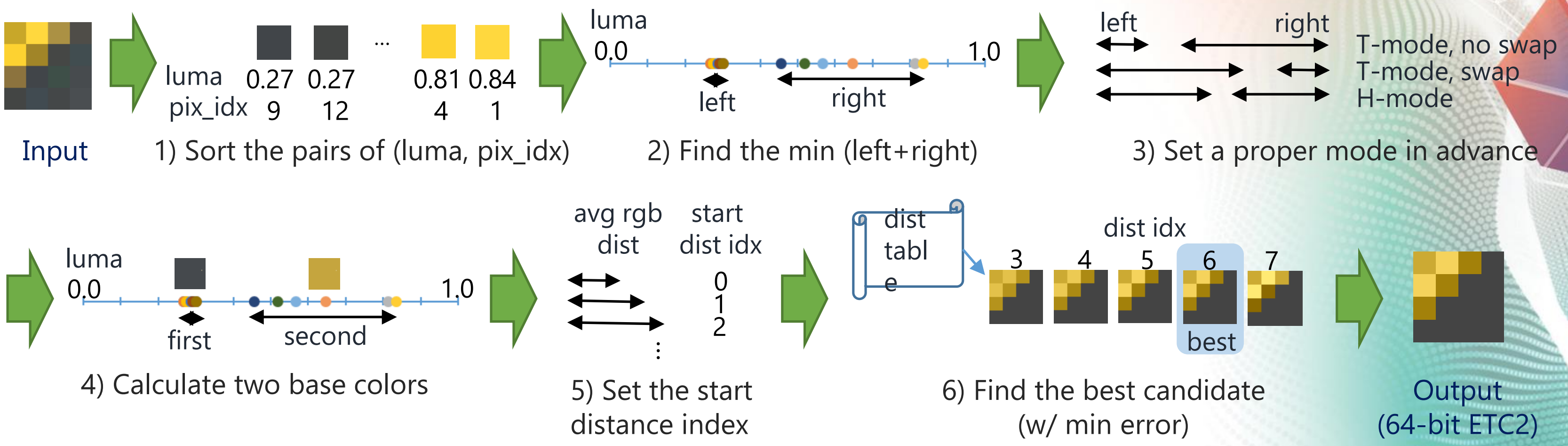
SIMD Optimizations

- Our early compression-mode decision is simple but…
  - Can be overhead because it should be performed on all blocks
  - By utilizing AVX2/SSE, we can access 16 pixels together
- Calculating luma differences
  - The value of a 256-bit luma variable (16x16bits) is calculated from three 128-bit RGB variables
  - The luma variable is converted into an 128-bit variable (16x8bits)
  - We utilize _mm_min_epu8() to find the min/max luma values quickly
- Checking corner pixels
  - The corner index pairs {(0, 15) & (3, 12)} and the pixel indices corresponding to the min/max values are compared by _mm_cmpeq_epi16()

Luma-based T-/H-Mode Compression

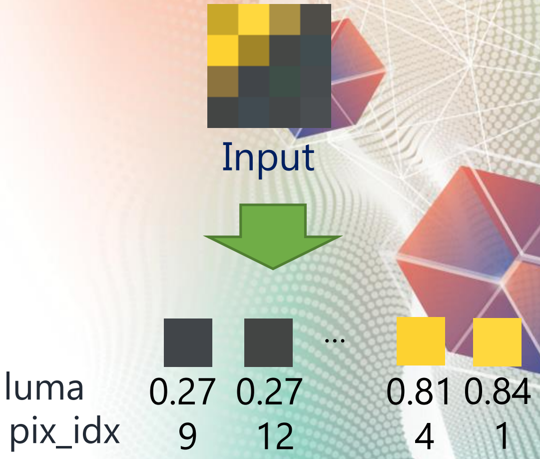
Overview

- Key ideas
  - Faster clustering by replacing the 3D RGB space with the 1D luma space
  - Reduction in the number of base-color pairs, compression modes & distance candidates
- Algorithm overview



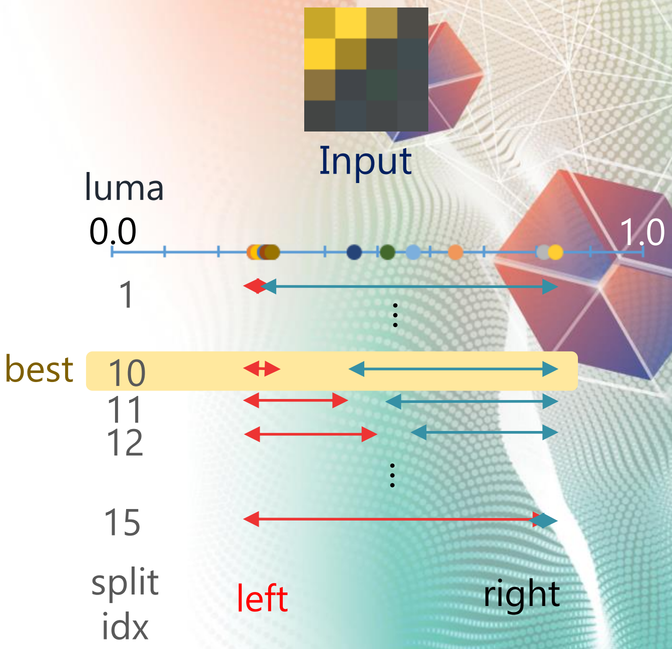
1) Sort the Pairs of (luma, pix_idx)

- The initial step for base-color calculation on the luma space
- Reuse the luma values calculated in the early compression-mode decision step
- Sorting of the pairs of a luma value and a pixel index in a block
  - In ascending order of luma values
  - Results in a single 1D line



2) Find the Min (left+right)

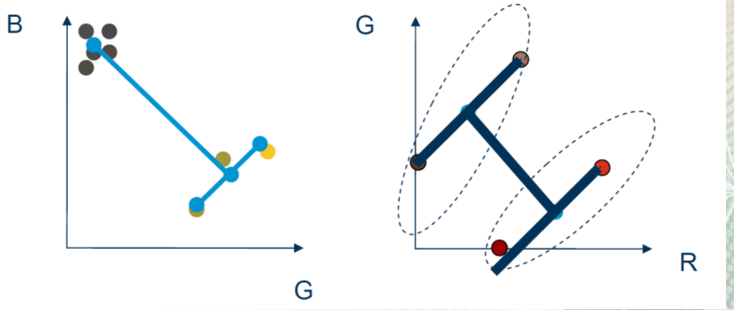
- Simple clustering based on the sorted luma values
- Find the min value of the 15 summed luma differences (LDs)
  - Summed LD = LD_L + LD_R + BF (bonus factor)
  - An iterator sweeps the line from left to right
- Small bonus factors added to both ends of the line
  - Prevent a situation that the longer cluster covers too large color range
  - Reduce a possibility of selecting the left- or right-most position; a “zero” difference can be incorrect after quantization



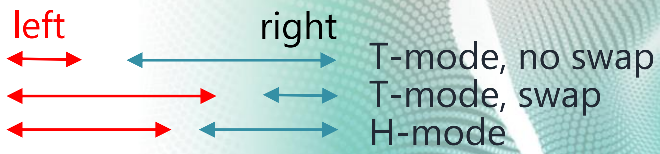
3) Set a Proper Mode in Advance

- Brute-force approach needs 3X iterations for the following modes
  - T-mode with swapping of the 1st and 2nd base colors
  - T-mode without the swapping
  - H-mode

- Instead, we can set a proper mode in advance according to LDs
  - $2LD_L \leq LD_R$ → T-mode, no swap
  - $LD_L \geq 2LD_R$ → T-mode, swap
  - Otherwise → H-mode

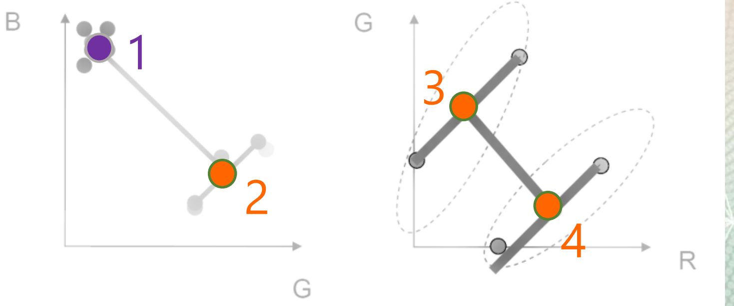


T-mode H-mode
[Ström and Petersson, GH 2007]



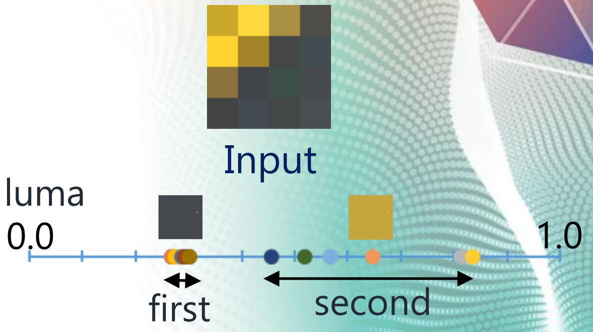
4) Calculate Two Base Colors

- Two base colors → Four paint colors
  - Different strategies for the two following cases
- Ranged paint colors
  - 2nd base color in the T-mode & both base colors in the H mode (Points 2-4): symmetric ranges from the midpoint
  - Pick the midpoint RGB color of both ends of each cluster
  - Clamp its RGB444 color to [1, 14] to prevent a halved range
- Base color = Paint color
  - 1st base color in the T-mode (Point 1): a single color point
  - Average all the RGB colors in the cluster
  - Clamp its RGB444 color to [0, 15]



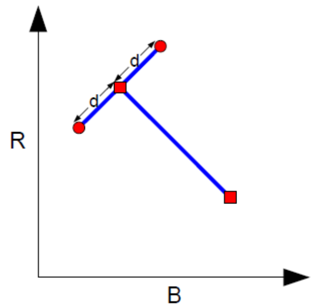
T-mode H-mode

[Ström and Petersson, GH 2007]



5) Set the Start Distance Index

- Distance *d*
  - RGB difference between a base color and two related paint colors



[Ström and Petersson, GH 2007]

- Start-distance-index optimization
  - Preset an optimal start index using the avg RGB distance
  - Skip unnecessary error-calculation iterations
  - A flip version of the T-/H-distance table

  3 levels earlier for conservative compression

Distance Index	Distance *d*
0	3
1	6
2	11
3	16
4	23
5	32
6	41
7	64

T-/H-distance table (in the ETC2 spec)

Average RGB Distance	Start Distance Index
0~16	0
17~23	1
24~32	2
33~41	3
42~	4

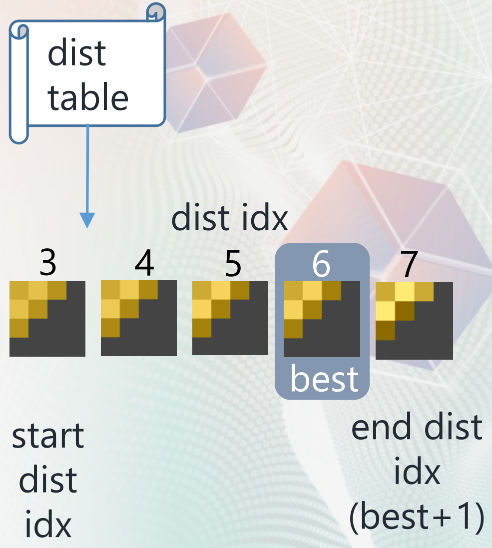
Start-distance-index table

6) Find the Best Distance Candidate

- Iterations to find the optimal distance candidate
  1) Calculate errors between the pixel and paint colors with the current distance
  2) Select the best paint color with the minimum error
  3) At the end of an iteration, update the up-to-date minimum block error

- End-distance-index optimization
  - Stop further iterations if the current iteration does not decrease the error
  - Based on the V-curve pattern of error values

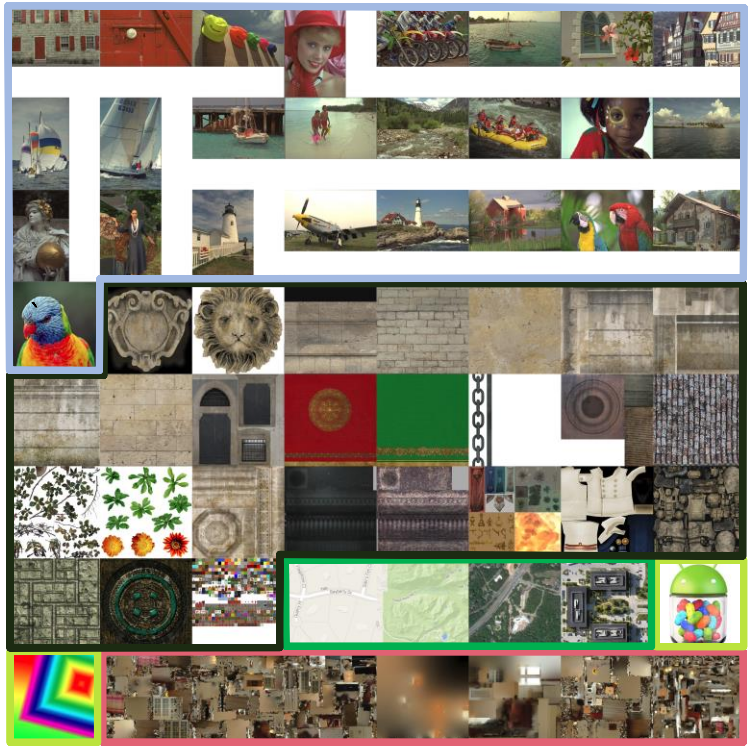
- SIMD optimization
  - Process all 16 pixels together & avoid inner pixel iterations
  - Use the perceptual error metric with the halved scaling factors in etcpak



Experiments and Results

SIGGRAPH ASIA 2020 VIRTUAL

Test Images



- 55 RGB + 9 RGBA textures
- Size: 256X256 ~ 8192X8192

- Photos (No. 1-25)
  - Kodak Lossless True Color Image Suite & Lorikeet
- Game textures (No. 26-51)
  - Crytek Sponza, FasTC & Vokselia Spawn (Minecraft)
- GIS maps (No. 52-55)
  - Google Maps & Cesium
- Synthesized images (No. 56-57)
  - Android Jelly & Gradient
- Captured images for 3D reconstruction (No.58-64)
  - Bedroom

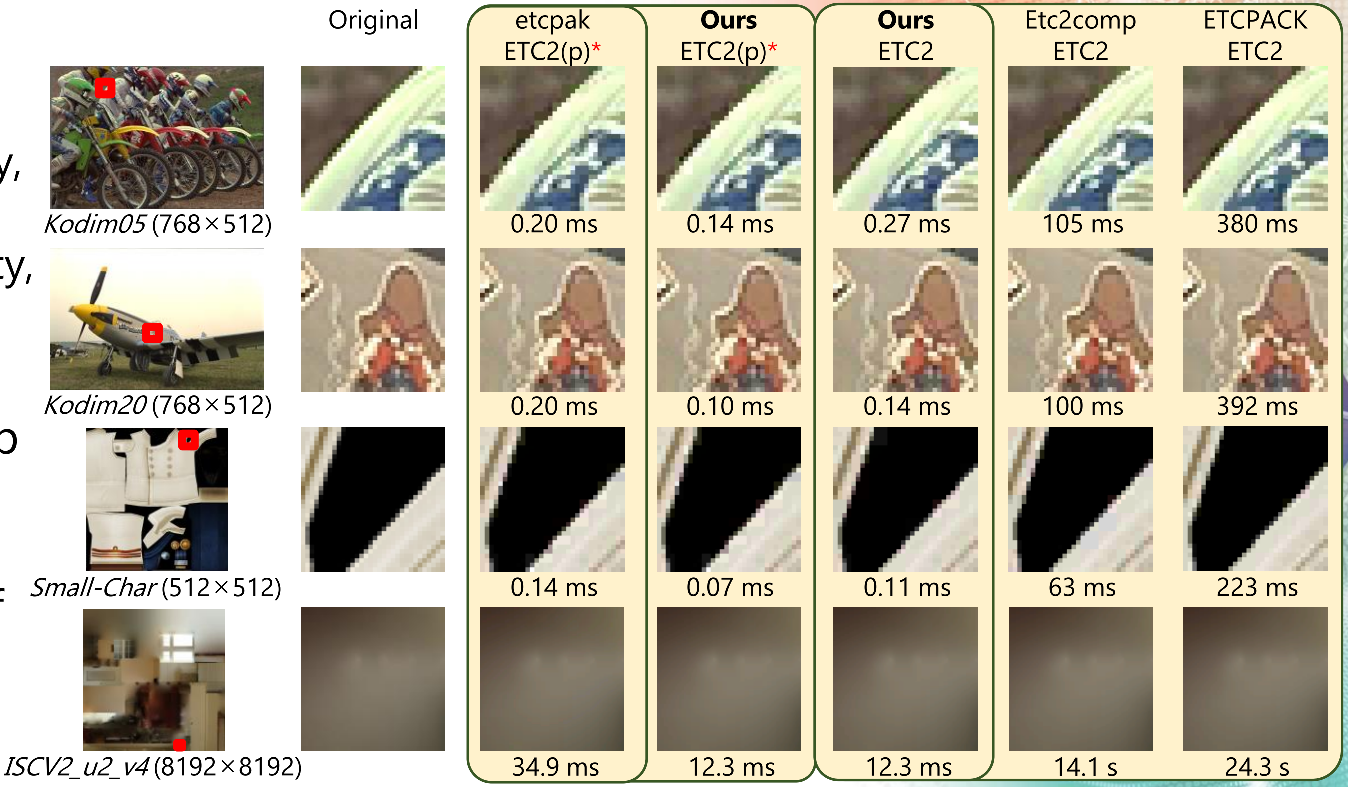
H/W & S/W Setup

- Test hardware
  - AMD Ryzen 7 3700X@3.6GHz 8-core (with hyper-threading) CPU
- Encoder settings (w/ fastest options)
  - etcpak 0.7: (partial*) ETC2
  - QuickETC2 (ours): partial* ETC2, full ETC2
  - Etc2Comp: effort = 0 (fastest) & error metric = rgba
  - ETCPACK 4.0.1: fast perceptual

* Partial ETC2 = ETC1+Planar

Quality & Performance Comparison on Four Reference Test Images

- Compared to etcpak
  - ETC2(p): Similar quality, 1.4~2.8X speed
  - Full ETC2: Better quality, 0.7~2.8X speed
- Compared to Etc2Comp & ETCPACK
  - Comparable quality
  - Two to three orders of magnitude faster



Quality & Performance Comparison on the 64 Test Images

Higher is better

Encoder & codec		etcpak ETC2(p)	**Ours** ETC2(p)	**Ours** ETC2	Etc2Comp ETC2	ETCPACK ETC2
Quality	PSNR (dB)	37.01	37.20	37.35	36.12*	38.35
	SSIM	0.959	0.959	0.959	0.940*	0.967
Performance	Mpixels/s	1911	3189	2600	4.0	1.3

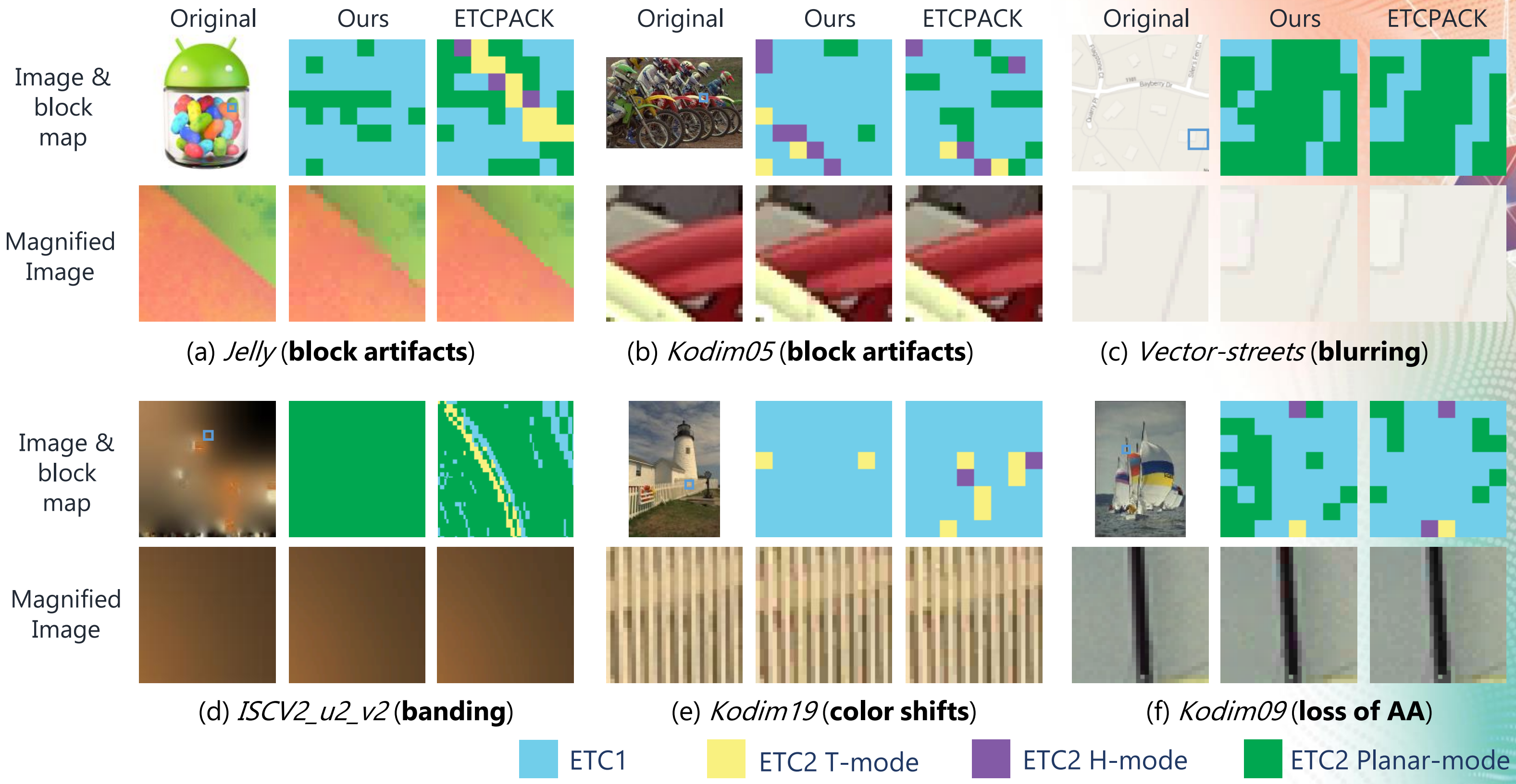
- Compared to etcpak
  - ETC2(p): +0.19dB PSNR w/ 1.67X performance
  - ETC2 : +0.34dB PSNR w/ 1.36X performance

Similar results to those in the previous slide

- Compared to Etc2Comp
  - Comparable quality
  - 650X performance
- Compared to ETCPACK
  - Lower quality (-1dB PSNR & -0.008 SSIM)
  - 2000X performance

* Note that several PSNR/SSIM drops occurred in Etc2Comp because of its RGBA compression policy; it ignores the original RGB colors at fully transparent pixels

Artifact Analysis



Occur in only QuickETC2

Commonly occur in ETC2

Concluding Remarks

Concluding Remarks

- Two approaches for fast ETC2 compression
  - Early compression-mode decision scheme
  - New T-/H-mode compression algorithm
  - Exploit the luma difference of a block for faster processing
- Future work
  - Quality & speed improvement - ETC1, EAC & early compression-mode decision
  - ARM Neon & GPU porting
- Full source code is attached
  - Can be directly applied to etcpak 0.7

SIGGRAPH ASIA 2020 VIRTUAL

References

- Patrick Cozzi and Christophe Ricco (Eds). 2012. **OpenGL Insights**. CRC Press. [link]
- Ericsson. 2018. **ETCPACK**. [link]
- Google Inc. and Blue Shift Inc. 2017. **Etc2Comp - Texture to ETC2 compressor**. [link]
- Pavel Krajcevski and Dinesh Manocha. 2014. **Real-Time Low-Frequency Signal Modulated Texture Compression using Intensity Dilation**. i3D 2014. [link]
- Pavel Krajcevski and Dinesh Manocha. 2014**. Fast PVRTC Texture Compression using Intensity Dilation**. JCGT 3, 4. [link]
- Jim Easterbrook, Oliver Grau, and Peter Schubel. 2010. **A system for distributed multi-camera capture and processing**." CVMP 2010. [link]
- Jae-Ho Nah, Byeongjun Choi, and Yeongkyu Lim. 2018. **Classified Texture Resizing for Mobile Devices**. ACM SIGGRAPH 2018 Talks. [link]
- Jacob Ström and Tomas Akenine-Möller. 2005. **iPACKMAN: High-Quality, Low-Complexity Texture Compression for Mobile Phones**. GH 2005. [link]
- Jacob Ström and Martin Pettersson. 2007. **ETC 2: Texture Compression using Invalid Combinations**. GH 2007. [link]
- Bartosz Taudul and Daniel Jungmann. 2020. **etcpak - The Fastest ETC Compressor on the Planet**. [link]
- Daniel Pohl, Daniel Jungmann, Bartosz Taudul, Richard Membarth, Harini Hariharan, Thorsten Herfet, and Oliver Grau. 2017. **The Next Generation of In-Home Streaming: Light Fields, 5K, 10 GbE, and Foveated Compression**. IEEE FedCICS 2017. [link]