SIGGRAPH THINK BEYOND

2020 S2020.SIGGRAPH.ORG

QUICK-ETC2

How to Finish ETC2 Compression within 1ms*

* 1K×1K size

RECORDING POLICY

It is important to recognize that many of the words, images, sounds, objects, and technologies presented at SIGGRAPH are protected by copyrights or patents. They are owned by the people who created them. Please respect their intellectual-property rights by refraining from making recordings from your device or taking screenshots. If you are interested in the content, feel free to reach out to the contributor or visit the ACM SIGGRAPH Digital library after the event, where the proceedings will be made available.

JAE-HO NAH

PROFESSIONAL

SWP LAB, CTO, LG ELECTRONICS

- BS, MS & PhD from Yonsei University (2012)
- 15+ tech papers in TOG, TVCG, CGF, etc.
- Reviewers for SIGGRAPH, EG, HPG, etc.
- Research interests: ray tracing, GPU architectures, rendering algorithms, etc.

SIGGRAPH THINK BEYOND
2020 S2020.SIGGRAPH.ORG

INTRO

Texture compression

ETC codecs & encoders

Introduction to our project

TEXTURE COMPRESSION

- For high-quality rendering,

  huge textures in a game are 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

Feb 7, 2017, 10:58am EST

Fallout 4's Ridiculously Huge, 58 GB HD Texture Pack Has Arrived

Paul Tassi Senior Contributor

Games

News and opinion about video games, television, movies and the internet.

This article is more than 3 years old.

(Photo: Bethesda Softworks)

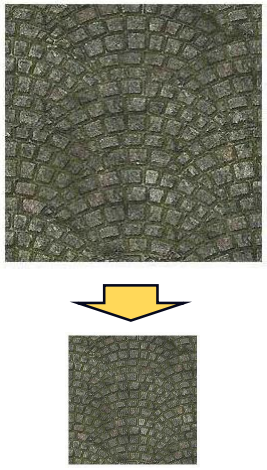
(source: Forbes)

REAL-TIME TEXTURE COMPRESSION

- In some scenarios, **REAL-TIME** texture compression is required

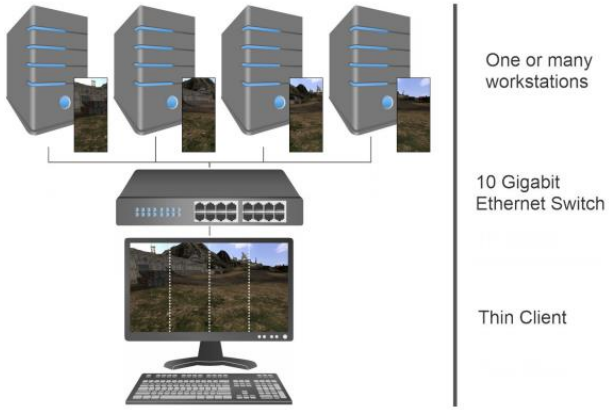
3D reconstruction
[Meerits, PhD thesis]

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



Texture resizing
[Nah et al., SIGGRAPH 2018]

Procedural engine
[Kemen, OpenGL Insights]

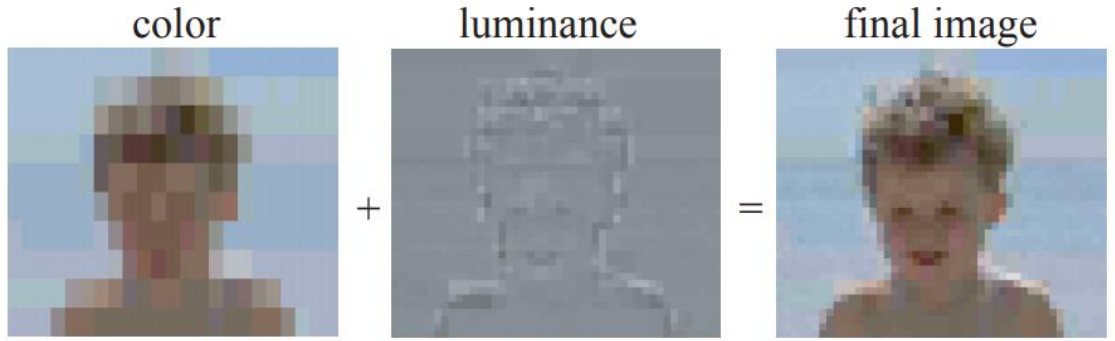


In-home streaming
[Pohl et al., FedCSIS 2017]

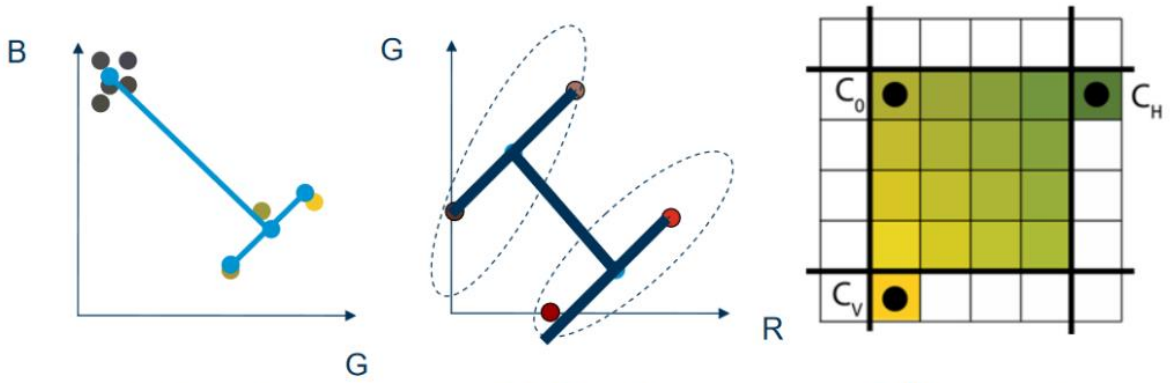
ETC CODECS

- 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 colors + per-pixel luminance
  - 4x2 or 2x4 sub-blocks
  - 6:1 compression ratio



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



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

ETC COMPRESSORS

ETCPACK [Ericsson 2005-2018]

- Reference encoder
- Fast & exhaustive 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)

OUR PROJECT: QUICK-ETC2

- Goals
  - Fastest ETC2 compression speed
  - Full ETC2 support (T, H, and planar) for high quality
  - SIMD optimization (SSE/AVX2)
- Built upon etcpak 0.6.2
- Two contributions
  - Early compression-mode decision (up to 2.5X speedup)
  - Fast T-/H-mode compression algorithm (up to +1dB PSNR)

SIGGRAPH THINK BEYOND
2020 S2020.SIGGRAPH.ORG

EARLY COMPRESSION-MODE DECISION

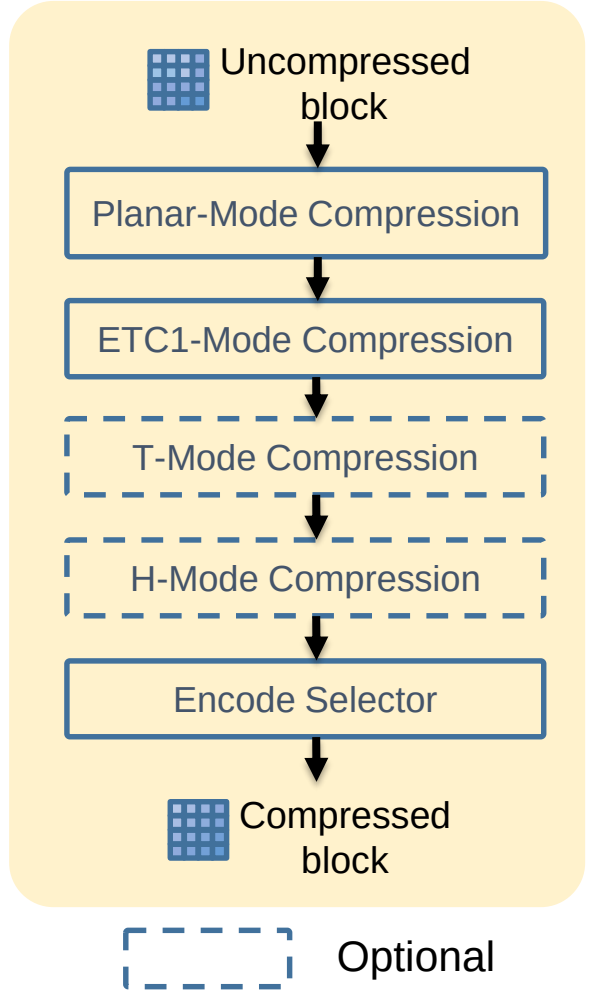
Traditional ETC2 encoding

Our approach

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 1.5-6X slower than ETC1 encoding

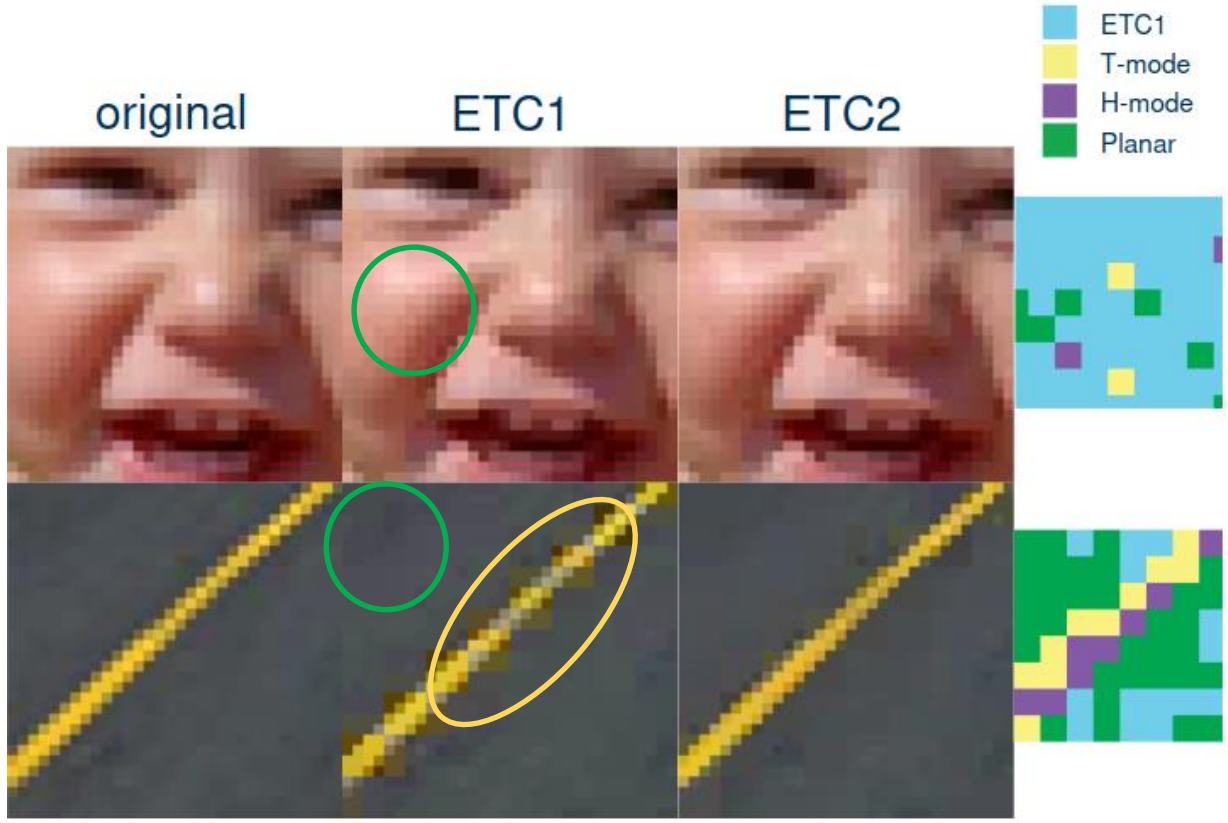
- Our question
  - Can we avoid these duplicated tests for a speedup?



OUR OBSERVATION

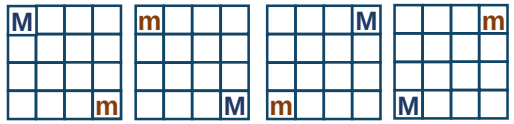
- Three ETC2 modes assist ETC1 in different ways
  - Planar: improves gradation in low-contrast regions
  - T & H: reduce block artifacts in high-contrast regions

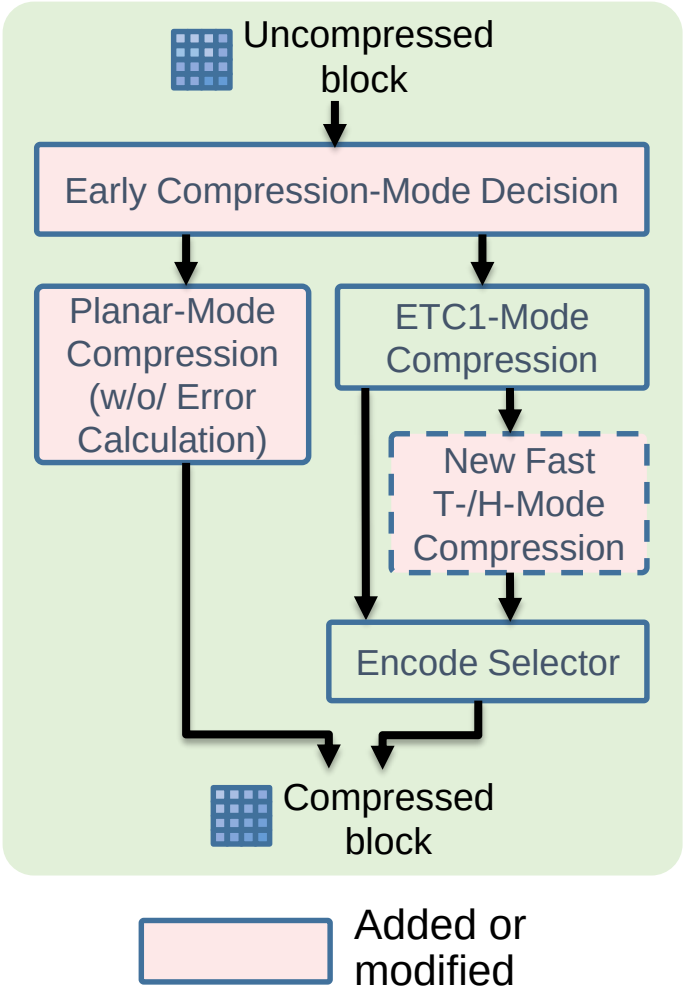
- Thus, we expect that
  - We can determine proper compression mode(s) in advance to avoid duplicated tests



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, 0.02]	N/A	Planar
	(0.02, 0.15]	 **M**: MaxLuma **m**: MinLuma	Planar
		other cases	ETC1
	(0.15, 0.40)	N/A	ETC1
	[0.40, 1.00]	N/A	ETC1 & T/H



SIMD OPTIMIZATIONS

- Our early compression-mode decision is simple but…
  - Can be overhead because they should be performed on all blocks
  - By utilizing AVX2/SSE, we can access 16 pixels together without loop iterations

- SIMD implementation for calculating the luma difference
  - The value of a 256-bit luma variable (16 X 16bits) is calculated from three 128-bit RGB variables
  - The luma variable is converted into a 128-bit variable (16 X 8bits)
  - Now, we can utilize _mm_min_epu8() to quickly find the min/max luma values

- SIMD implementation for 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()

SIGGRAPH 2020 S2020.SIGGRAPH.ORG THINK BEYOND

LUMA-BASED T-/H-MODE COMPRESSION

Traditional T/H compression

Our approach

TRADITIONAL ETC2 T-/H-MODE COMPRESSION

- **procedure** ETC2_TH (pix)
  1. {c1, c2} ← FindBaseColors (pix)
  2. {blockT, errorT} ← CompressBlockT (pix, c1 ,c2)
  3. {blockH, errorH} ← CompressBlockH (pix, c1 ,c2)
  4. **return** (errorT < errorH)? {blockT, errorT} : {blockH, errorH}

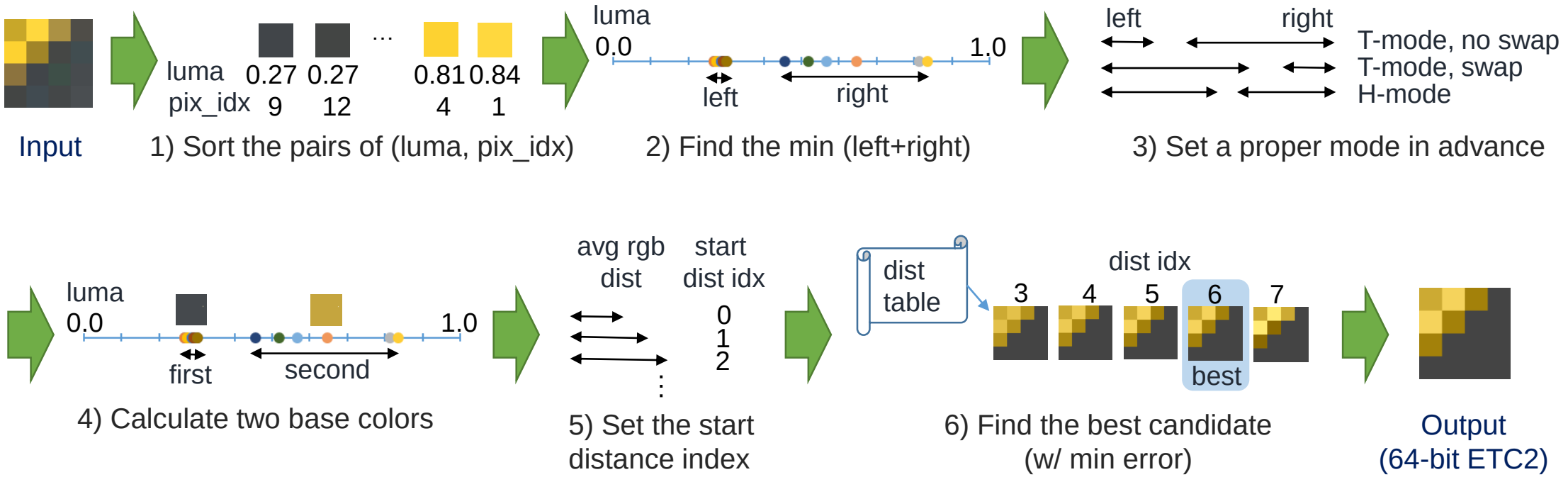
- Computational cost of ETC2_TH (pix)
  - $C_{TH} = N_{BC} \left(F_{BC} \cdot C_{BC} + N_{Mode} \cdot N_{Dist} \cdot C_{EC} \right)$

N_{BC}	F_{BC}	C_{BC}	N_{Mode}	N_{Dist}	C_{EC}
# of base-color pairs (BCPs)	Outputs 1 or 2 1: T&H share same BCPs 2: otherwise	Cost of BC calculation	# of comp modes	# of distance candidates	Cost of error calculation

  - C_{TH} (ETCPACK$_{FAST}$) = $3 (2 \cdot C_{BC} + 3 \cdot 8 \cdot C_{EC}) = 6C_{BC} + 72C_{EC}$

OUR APPROACH

- Key idea
  - 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



Input

1) Sort the pairs of (luma, pix_idx)

2) Find the min (left+right)

3) Set a proper mode in advance

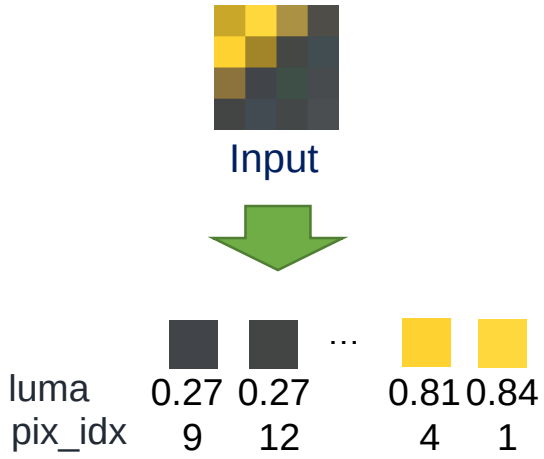
4) Calculate two base colors

5) Set the start distance index

6) Find the best candidate (w/ min error)

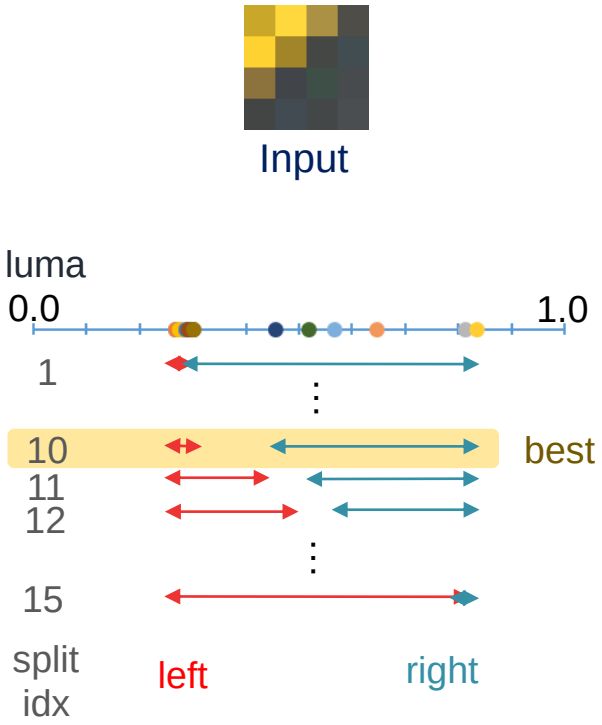
Output (64-bit ETC2)

1) SORT THE PAIRS OF (LUMA, PIX_IDX)



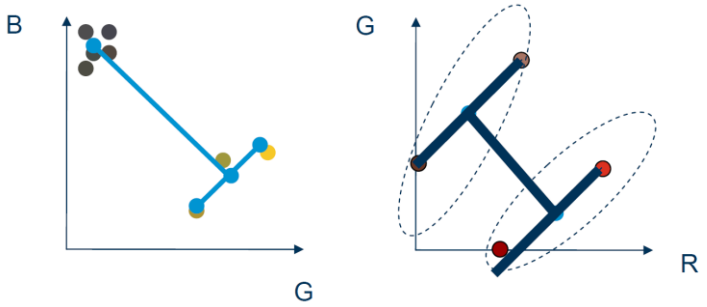
- The first step for base-color calculation on the luma space
- No need for calculating the luma values of each pixel again
  - Already calculated in the early compression-mode decision step
- Sorting of the pairs of a luma value and a pixel index in a block
  - In ascending order of luma values
  - Results in a single 1D line

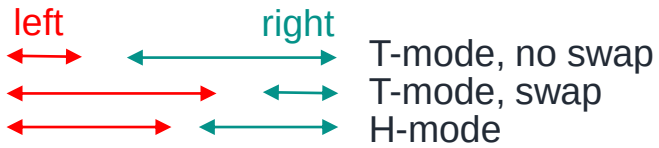
2) FIND THE MIN (LEFT+RIGHT)



- Simple clustering based on the sorted luma values
- Calculate the min value of the 15 summed luma differences (LDs)
  - Summed LD = LD_L + LD_R
  - An iterator sweeps the line from left to right
- Small bonus factors added to both ends of the line
  - 8 for 1st & 15th pairs (luma format: 8-bit fixed)
  - 4 for 2nd & 14th pairs
  - 2 for 3rd & 13th pairs
  - Prevent a situation that the longer cluster covers too large color ranges
  - Reduce a possibility of selecting the left- or right-most position as the split point; a “zero” difference of the shorter cluster is incorrect after RGB444 quantization

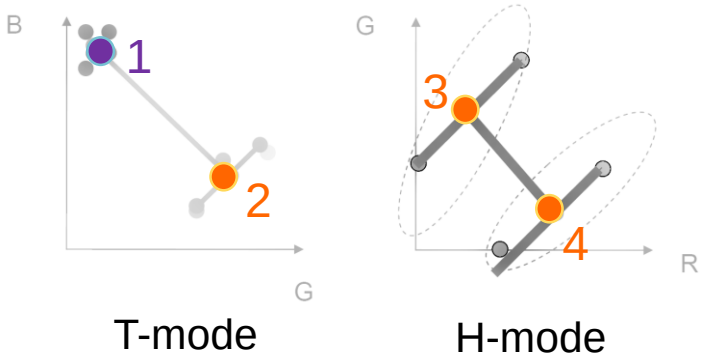
3) SET A PROPER MODE IN ADVANCE

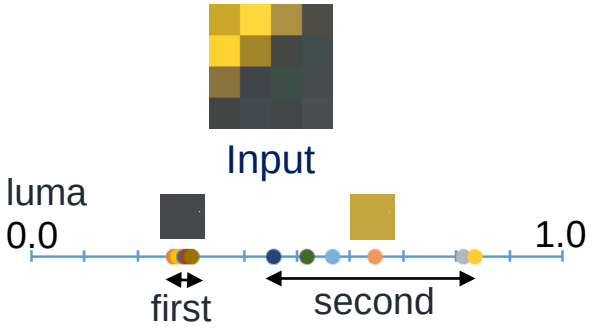




- Brute-force approach needs 3X iterations for the following modes (ETCPACK & Etc2Comp)
  - T-mode with swapping of the 1st and 2nd base colors
    (the 2nd base color is located on the upper horizontal line of “T” character)
  - T-mode without the swapping
  - H-mode
- Instead, we can set a proper mode in advance according to LD ranges
  - $2LD_L \leq LD_R$ → T-mode, no swap
  - $LD_L \geq 2LD_R$ → T-mode, swap
  - Otherwise → H-mode

4) CALCULATE TWO BASE COLORS





- Ranged paint color
  - 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
- Averaged 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]

5) SET THE START DISTANCE INDEX

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

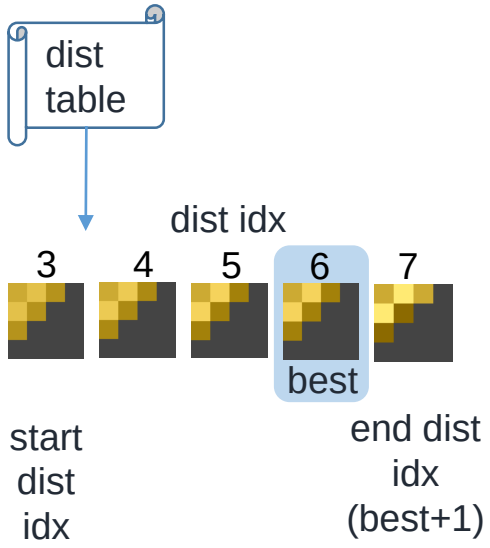
Distance table for ETC2 T & H modes

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

Table for start-distance indices

- Start-distance-index optimization
  - Inspired by the ETC2 T/H distance table
  - The optimal start index can be determined in advance
  - Using the average RGB distances of the two clusters
  - Skip unnecessary error-calculation iterations
  - Three-level earlier start for conservative compression

6) FIND THE BEST CANDIDATE (W/ MIN ERROR)



- Iterations to find the optimal distance candidate
  1) Compare the pixel colors and the paint colors (w/ the current distance value)
  2) Obtain the luma error
  3) Select the best paint color with the minimum error
  4) 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
  - This is possible because the pattern of error values is usually V-curves

- SSE/AVX2 optimization
  - All 16 pixels are processed together to avoid inner pixel iterations
  - Convert 16 RGB888 colors into three __m256i variables
  - Use the perceptual error metric with the halved scaling factors in etcpak (38, 76, and 14 for each RGB channel) to prevent overflows of signed int16

COST ESTIMATION

- Let's return to the cost equation and compare ours with ETCPACK
  - $C_{TH} = N_{BC} \left(F_{BC} \cdot C_{BC} + N_{Mode} \cdot N_{Dist} \cdot C_{EC} \right)$

    N_{BC}: # of base-color pairs (BCP)

    F_{BC}: Outputs 1 or 2 1: T&H share same BCP 2: otherwise

    C_{BC}: Cost of BC calculation

    N_{Mode}: # of comp modes

    N_{Dist}: # of distance candidates

    C_{EC}: Cost of error calculation
  - C_{TH} (ours) $= 1 \left(1 \cdot C'_{BC} + 1 \cdot N'_{Dist} \cdot C'_{EC} \right) = C'_{BC} + N'_{Dist} \cdot C'_{EC}$

    C'_{BC}: Reduced C_{BC} (3D RGB → 1D luma)

    N'_{Dist}: Reduced N_{Dist} ($2 \le N'_{Dist} \le 8$)

    C'_{EC}: Reduced C_{EC} (w/ SIMD optimizations)
  - C_{TH} ($ETCPACK_{FAST}$) $= 6C_{BC} + 72 C_{EC}$

Greatly reduced cost

SIGGRAPH THINK BEYOND
2020 S2020.SIGGRAPH.ORG

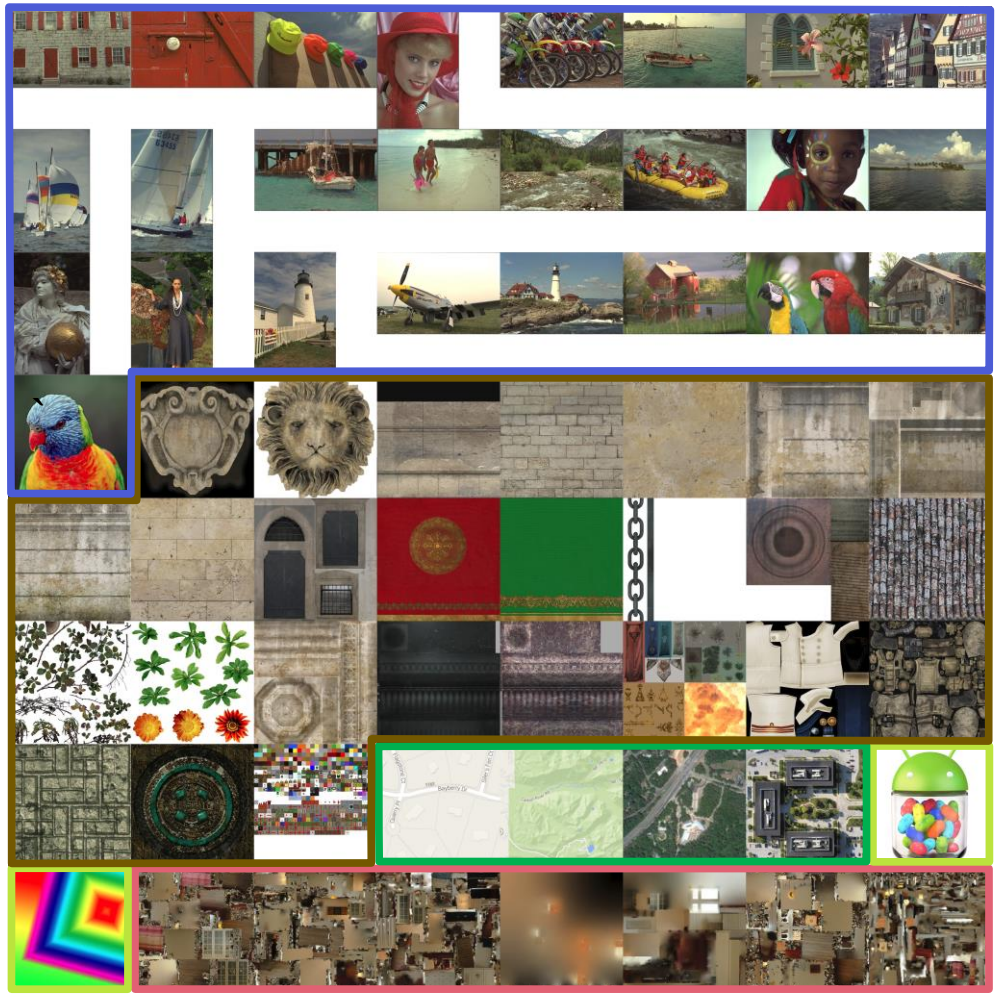
EXPERIMENTS AND RESULTS

Test setup

Results and analysis

Limitations

TEST IMAGES



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

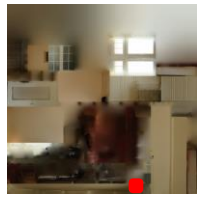
- Photos (No.1 – No. 25)
  - Kodak Lossless True Color Image Suite & Lorikeet
- Game textures (No.26 – No. 51)
  - Crytek Sponza, FasTC & Vokselia Spawn (Minecraft)
- GIS maps (No.52 – No. 55)
  - Google Maps & Cesium
- Synthesized images (No.56 – No. 57)
  - Android Jelly & Gradient
- Captured images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

H/W & S/W SETUP

- Test environments
  - AMD Ryzen 7 3700X@3.6GHz 8-core (with hyper-threading) CPU & 32 GB of RAM
  - Ubuntu 18.04 & ImageMagick 7.0.9

- Compressor settings
  - etcpak 0.6.2: (partial) ETC2
  - QuickETC2 (ours): partial ETC2, full ETC2
  - Etc2Comp: effort = 0 (fastest) & error metric = rgba
  - ETCPACK 4.0.1: fast perceptual

QUALITY & PERFORMANCE COMPARISON ON FOUR REFERENCE TEST IMAGES

	Original	etcpak ETC2(p)*	**Ours** ETC2(p)*	**Ours** ETC2	Etc2comp ETC2	ETCPACK ETC2
 *Kodim05* (768×512)		0.20 ms	0.15 ms	0.27 ms	105 ms	380 ms
 *Kodim20* (768×512)		0.21 ms	0.13 ms	0.17 ms	100 ms	392 ms
 *Small-Char* (512×512)		0.14 ms	0.09 ms	0.13 ms	63 ms	223 ms
 *ISCV2_u2_v4* (8192×8192)		35.2 ms	14.9 ms	15.3 ms	14.1 s	24.3 s

- Compared to etcpak
  - ETC2(p): Similar quality, 1.3~2.4X speed
  - Full ETC2: Better quality, 0.7~2.3X speed
- Compared to Etc2Comp & ETCPACK
  - Comparable quality
  - Two to three orders of magnitude faster

* ETC2(p): Partial ETC2 = ETC1+Planar

QUALITY & PERFORMANCE COMPARISON ON THE 64 TEST IMAGES

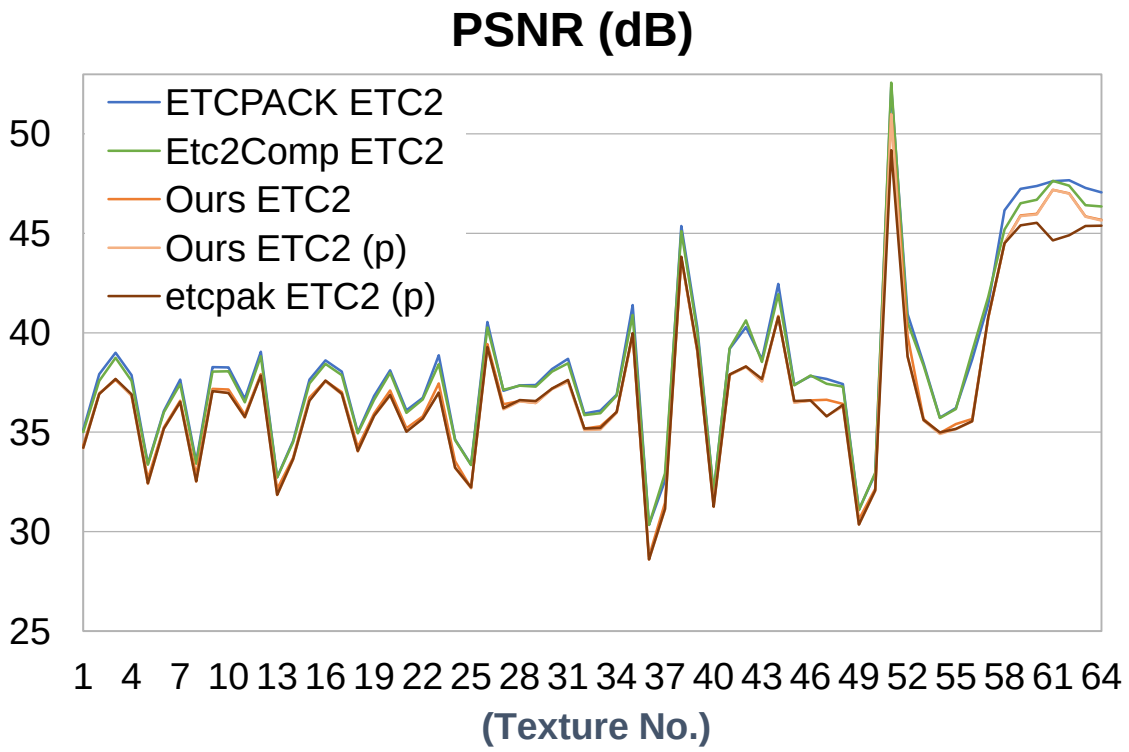
Encoder & codec		etcpak ETC2(p)	**Ours** ETC2(p)	**Ours** ETC2	Etc2Comp ETC2	ETCPACK ETC2
Quality	PSNR (dB)	37.17	37.27	37.40	38.33	38.49
	SSIM	0.959	0.958	0.958	0.940	0.967
Performance	Mpixels/s	1782	2588	2185	4.0	1.3

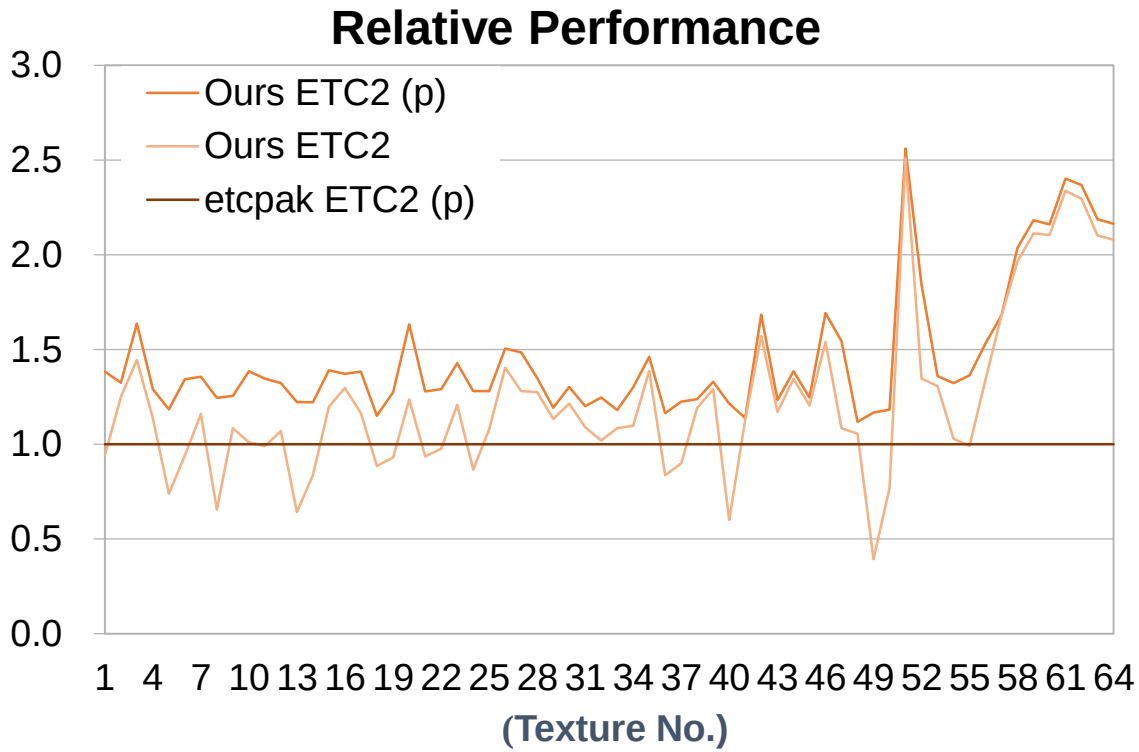
Higher is better

- Compared to etcpak
  - ETC2(p): +0.10dB PSNR w/ 1.45X performance
  - ETC2 : +0.23dB PSNR w/ 1.22X performance
- Compared to Etc2Comp
  - Comparable quality (-0.97dB PSNR & +0.018 SSIM)
  - 550X performance
- Compared to ETCPACK
  - Lower quality (-1.09dB PSNR & -0.009 SSIM)
  - 1618X performance

Similar results to those in the previous slide

QUALITY & PERFORMANCE COMPARISON ON THE 64 TEST IMAGES

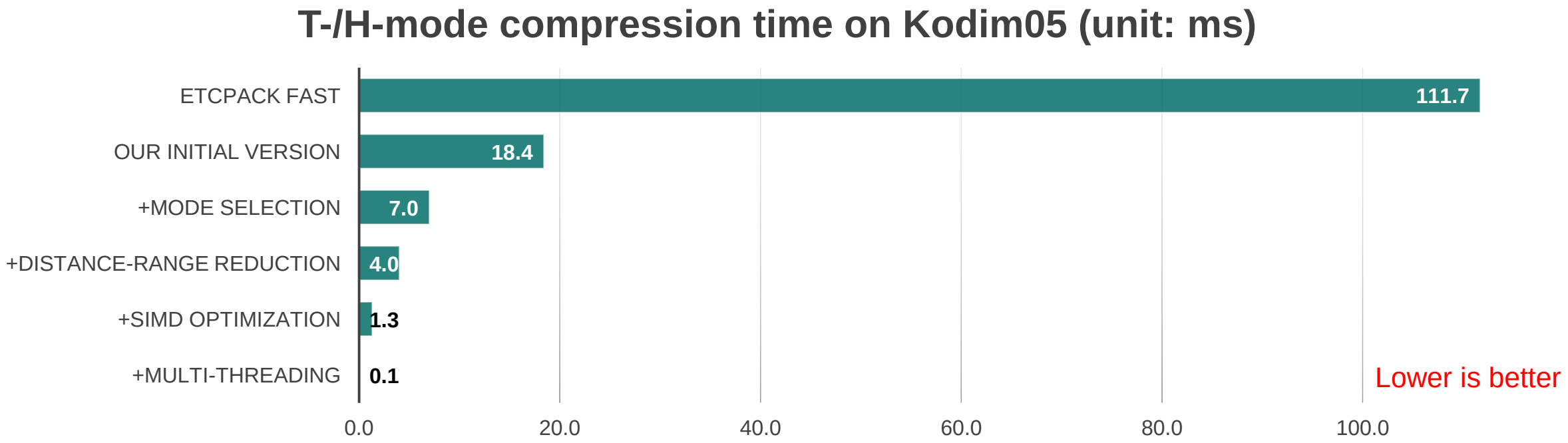




- Similar patterns between the two above graphs
- Our early compression-mode decision adaptively controls performance and quality
  - Smoothly-varied scenes : more planar compression for higher speed
  - High-contrast scenes : more additional T/H compression for higher quality

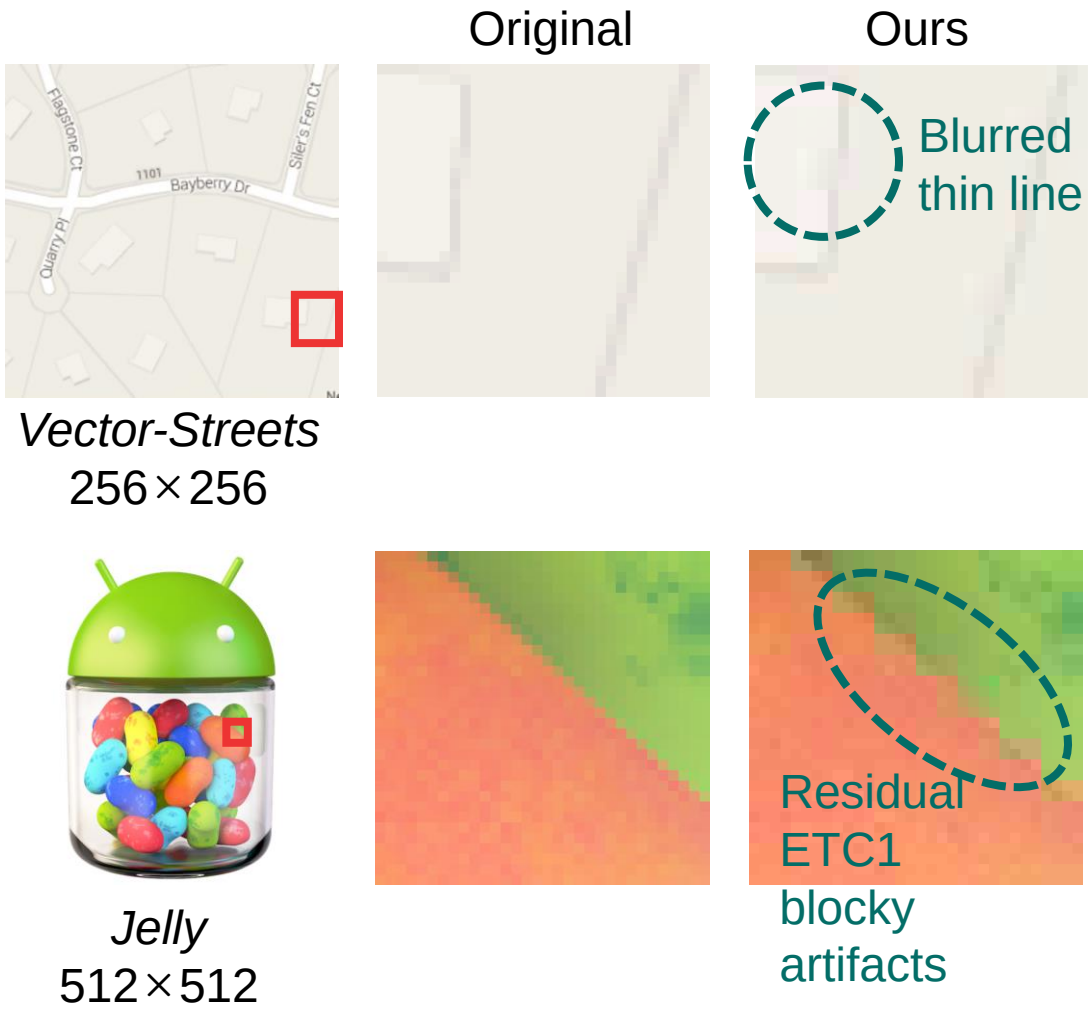
COMPARISON OF T-/H-MODE COMPRESSION ONLY

- Timings of the T-/H-mode compression part (w/ early compression-mode decision)
  - Our method is 80X faster than ETCPACK's fast T-/H-mode compression (w/ the same number of threads)
- Almost same quality (+0.005dB PSNR than ETCPACK)
  - For high-contrast regions, our luma-based compression is reasonable



LIMITATIONS

- Drawback of the early-compression decision scheme
  - Sometimes prevents further quality enhancement
- Not affect EAC compression
  - Improves neither quality nor speed of compression for one- or two-channel textures (normal maps, lightmaps, etc.)



SIGGRAPH THINK BEYOND
2020 S2020.SIGGRAPH.ORG

CONCLUDING REMARKS

Summary

Future work

References

CONCLUSIONS AND FUTURE WORK

- Two approaches for fast ETC2 compression
  - Early compression-mode decision scheme & new T-/H-mode compression algorithm
  - Exploit the luma difference in a block for faster processing
  - Two to three orders magnitude faster performance than the existing high-quality ETC2 compressors

- Future work
  - Quality & speed improvement - ETC1, EAC & early compression-mode decision
  - ARM Neon implementation for mobile devices
  - GPU implementation (which can possibly be used for streaming G-buffer compression)

- Source code will be available soon with the full-paper version

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]
- Siim Meerits. 2018. **Real-time 3D Reconstruction of Dynamic Scenes Using Moving Least Squares**. Ph.D. Dissertation. Keio University. [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]

Q&A

- Thanks for watching
- Any questions?