PACIFIC GRAPHICS 2023
Daejeon, Korea

상명대학교
SANGMYUNG UNIVERSITY

# H-ETC2: Design of a CPU-GPU Hybrid ETC2 Encoder

Hyeon-ki Lee, Sangmyung University
gusrl3204@gmail.com
Jae-Ho Nah*, Sangmyung University
jaeho.nah@smu.ac.kr

# Introduction & related work

# Introduction

- These days, high-quality textures are used to create computer graphics applications such as games and movies
- Consider the following scenario:
  - 5,000 4K X 4K-sized uncompressed textures = **83 pixels**
- Using a lot of these high-quality textures
  → Require a lot of memory and bandwidth

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

**Paul Tassi** Senior Contributor
*News and opinion about video games, television, movies and the internet.*

Follow

Feb 7, 2017, 10:58am EST

**This article is more than 6 years old.**

(Photo: Bethesda Softworks)

(Source : Forbes)

# Introduction

- These days, high-quality textures are used to create computer graphics applications such as games and movies
- Consider the following scenario:
  - 5,000 4K X 4K-sized uncompressed textures = **83 pixels**
- Using a lot of these high-quality textures
  → Require a lot of memory and bandwidth

→ How to solve this problem?

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

**Paul Tassi** Senior Contributor
*News and opinion about video games, television, movies and the internet.*

Follow

Feb 7, 2017, 10:58am EST

**This article is more than 6 years old.**

(Photo: Bethesda Softworks)

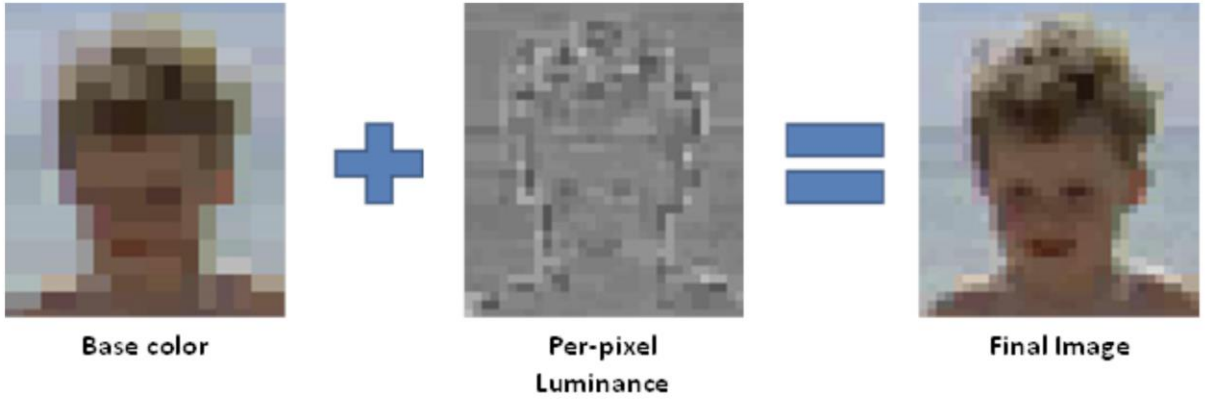
(Source : Forbes)

# Texture compression

- Widely adopted for reducing the pressure on the memory and bandwidth
  → Lossy compression
- The texture is compressed and stored in memory before being passed to the GPU
  → Unpacked on the GPU in real time
- Reducing the footprint and bandwidth of texture memory
- Standard texture compression codec
  - Microsoft BC1-7 (Desktop)
  - ETC1/ETC2/EAC (Android)
  - PVRTC (iOS)
  - ASTC (Android/iOS)

Core idea of ETC Family

Source : TEXTURE COMPRESSION TECHNIQUES



# Texture compression

- Widely adopted for reducing the pressure on the memory and bandwidth
  → Lossy compression

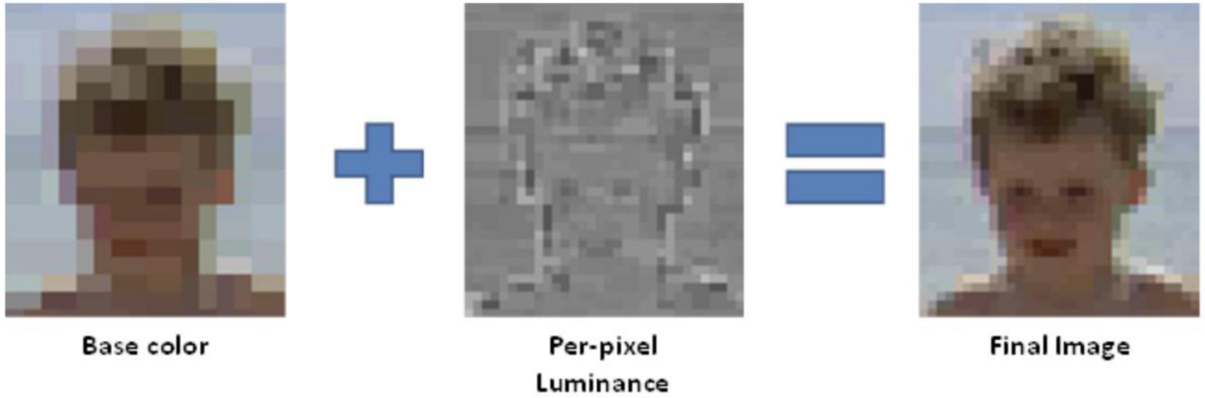
- The texture is compressed and stored in memory before being passed to the GPU
  → Unpacked on the GPU in real time

- Reducing the footprint and bandwidth of texture memory

- Standard texture compression codec
  - Microsoft BC1-7 (Desktop)
  - **ETC1/ETC2/EAC (Android)** ← **This!**
  - PVRTC (iOS)
  - ASTC (Android/iOS)

Core idea of ETC Family

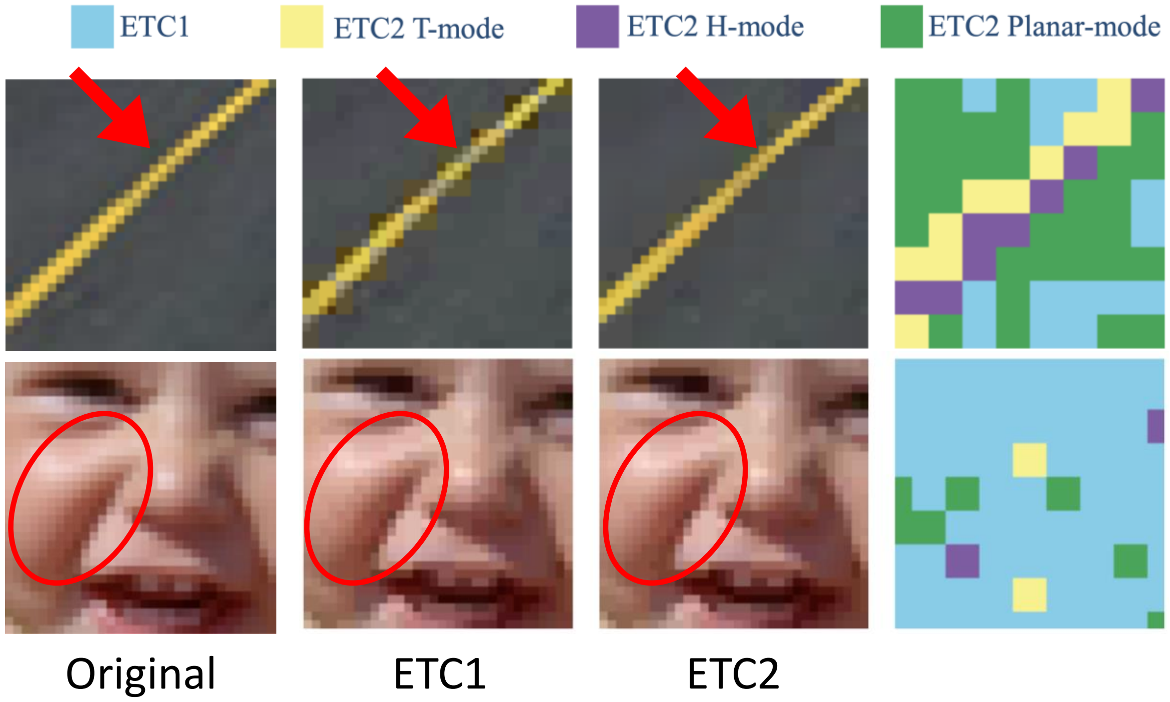
Source : TEXTURE COMPRESSION TECHNIQUES



# ETC1/ETC2/EAC

- **ETC1 (iPACKMAN)**
  - OpenGL ES 2.0 standard
  - Two base chrominance + per-pixel luminance
  - 6 : 1 compression ratio
- **ETC2**
  - OpenGL ES 3.0 standard
  - Three addition modes : T, H & Planar
  - Less block & banding artifacts
  - Alpha support (EAC)

[Ström and Petersson,GH2007]



# Our observation

- Our question

**How can we achieve fast encoding speeds**
**while preserving as much quality as possible for artist-created textures?**

- For better quality → more iterations & RGB space search
- For faster encoding speed → lightweight algorithm & optimization

- GPU = A Single Instruction Multiple Thread (SIMT) device
- We introduce a hybrid encoder using CPU-GPU, which performs fast encoding with a CPU encoder and then improves the encoding with a GPU encoder

# Core related ETC compressors

**QuickETC2**
[Nah. SA2020]

- Ultra-fast multi-threaded SIMD-optimized encoder
- Using two methods
  - Early Compression-Mode Decision
  - Luma-based T-/H-Mode Compression
- Integrated into
  - etcpak 1.0 encoder

**Betsy**
[Goldberge. 2022]

- Based on OpenGL open-source encoder
- Using improved encoding progress about each of modes
- Fine quality control
  - Q=0, 1, 2
- Integrated into
  - Godot game engine

# Core related ETC compressors

Our CPU encoder

**QuickETC2**
**[Nah. SA2020]**

- Ultra-fast multi-threaded SIMD-optimized encoder
- Using two methods
  - Early Compression-Mode Decision
  - Luma-based T-/H-Mode Compression
- Integrated into
  - etcpak 1.0 encoder

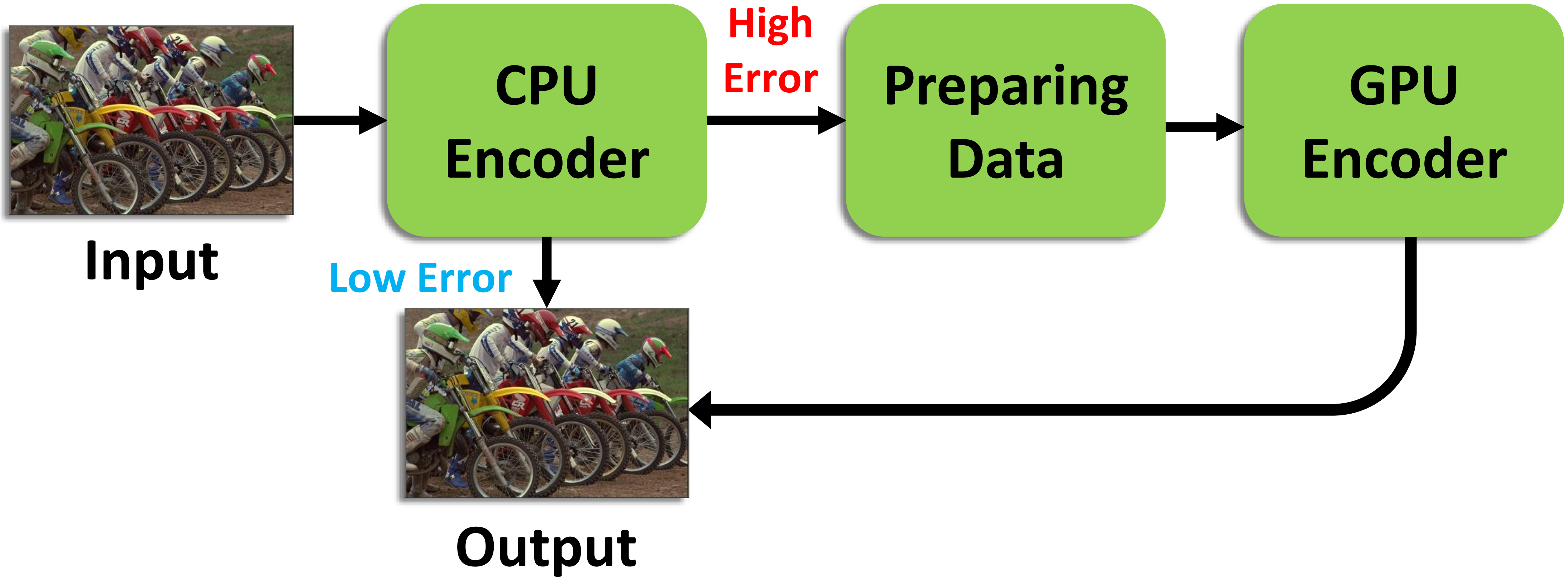
**Betsy**
**[Goldberge. 2022]**

- Based on OpenGL open-source encoder
- Using improved encoding progress about each of modes
- Fine quality control
  - Q=0, 1, 2
- Integrated into
  - Godot game engine

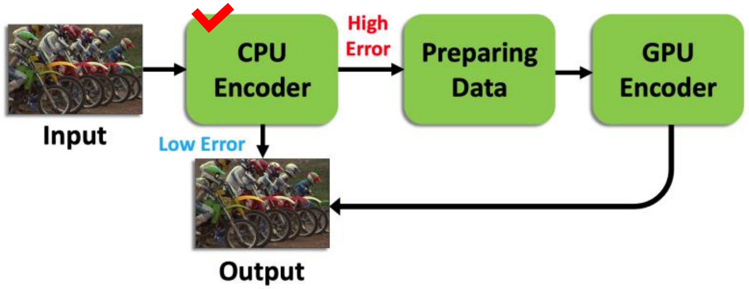
Our GPU encoder

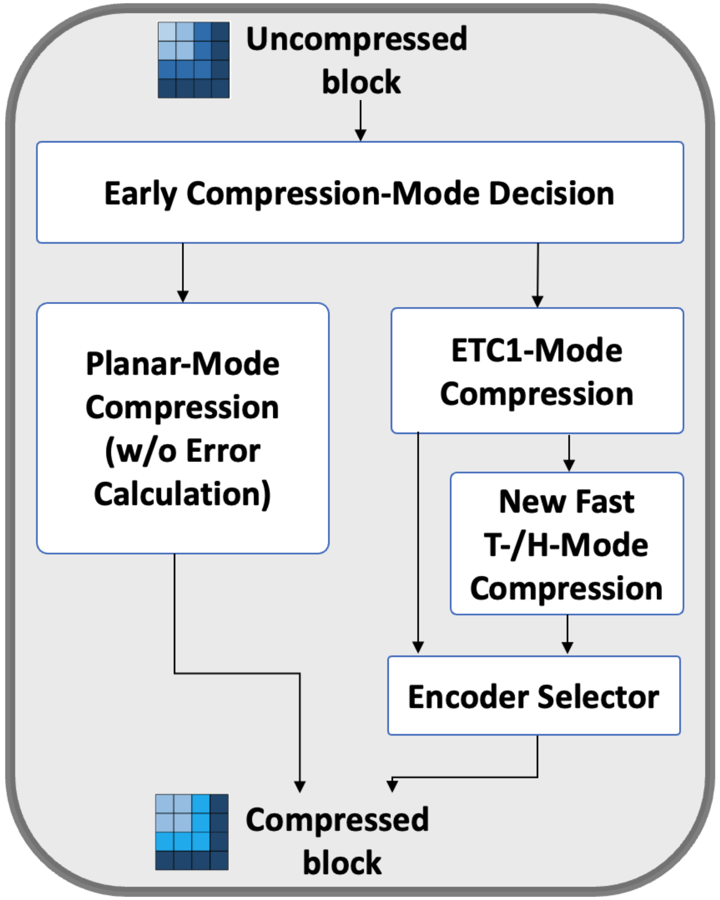
# System overview

# System overview



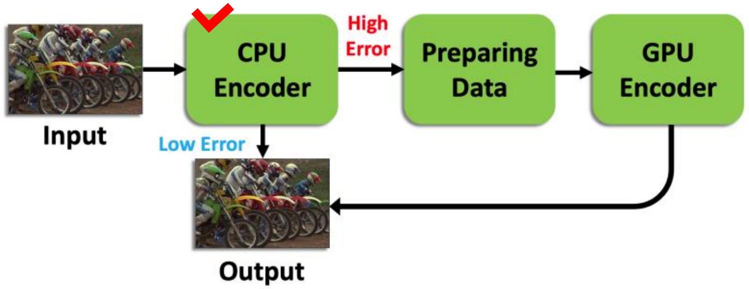
# Traditional QuickETC2

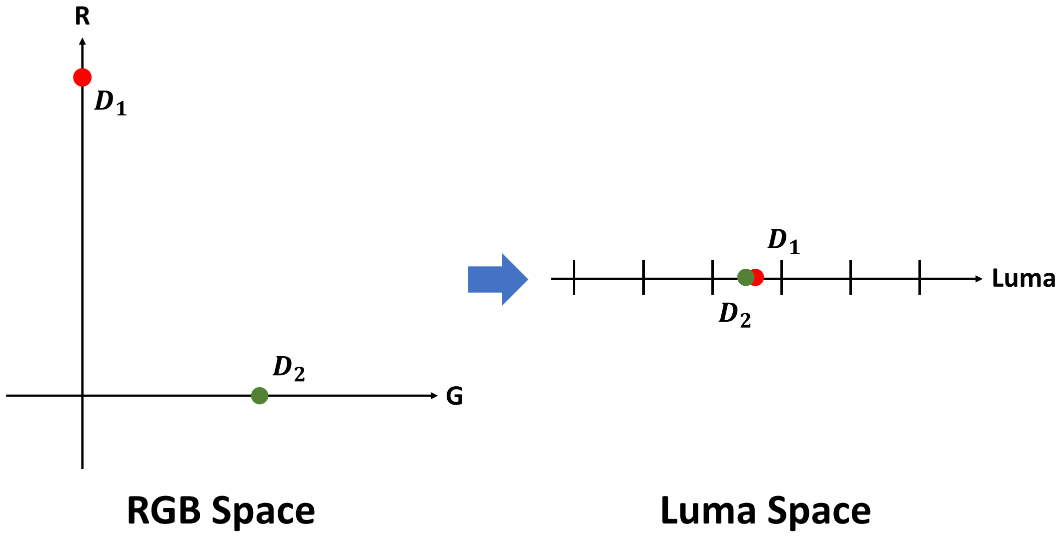



- **Early Compression-Mode Decision**
  - Converting RGB data to luma (linear luminance) for utilizing luma contrast
  - Using luma contrast to set the mode (ETC1, T-mode, H-mode, Planar-mode)
- **New Fast T-/H-Mode Compression**
  - Reduce dimensionality by converting pixel block colors to luma values
  - T-/H-mode encoding using min/max values of the luma values
- **Fastest encoding speed among ETC2 encoders**

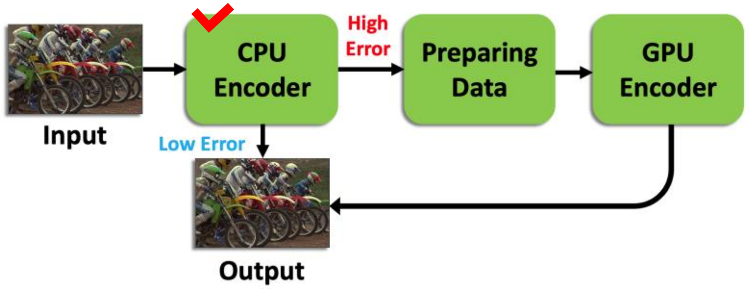
# Luma-space problem

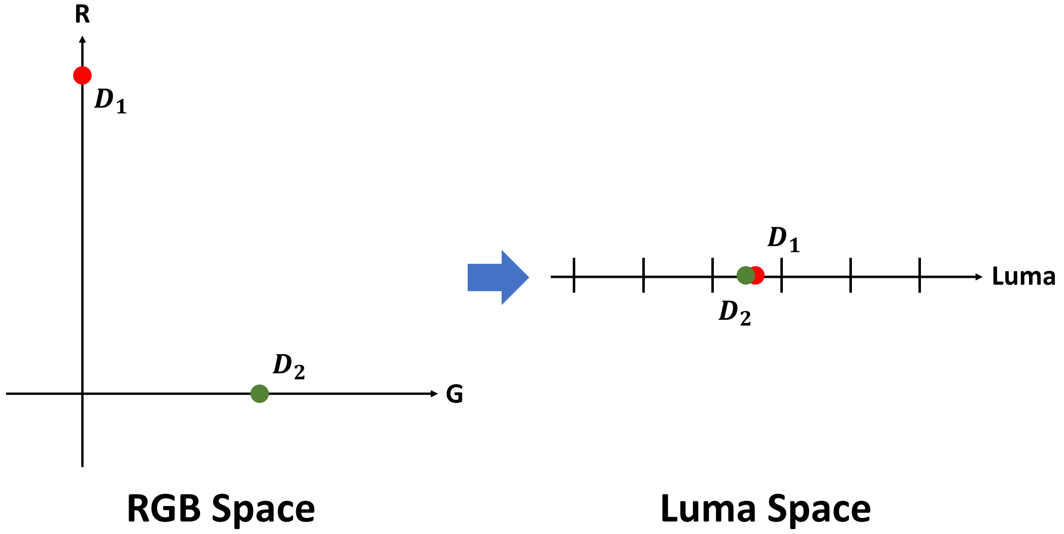



- Let's assume a situation
  - $D_1$ with RGB channel = (255, 0, 0)
  - $D_2$ with RGB channel = (0, 128, 9)
- $luma = 0.3 \times R + 0.59 \times G + 0.11 \times B$
- $D_{1(luma)} = 76.5, \;\; D_{2(luma)} = 76.509$
- They become quite similar in the luma space
  → **probability of artifacts!**

# Luma-space problem




- **Re-calculation error metric**

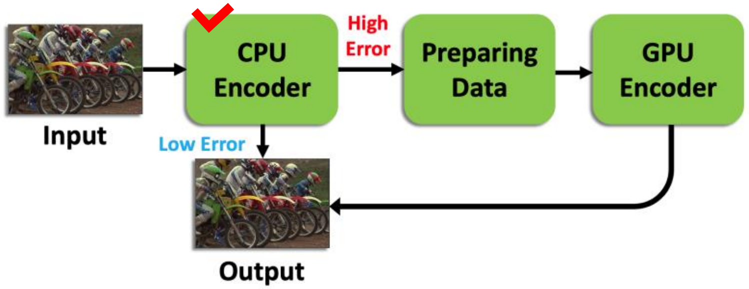
$$error = \sum_{i=0}^{N-1} \max\left(|\bar{x}_{i,r} - x_{i,r}|, |\bar{x}_{i,g} - x_{i,g}|, |\bar{x}_{i,b} - x_{i,b}|\right)^2$$

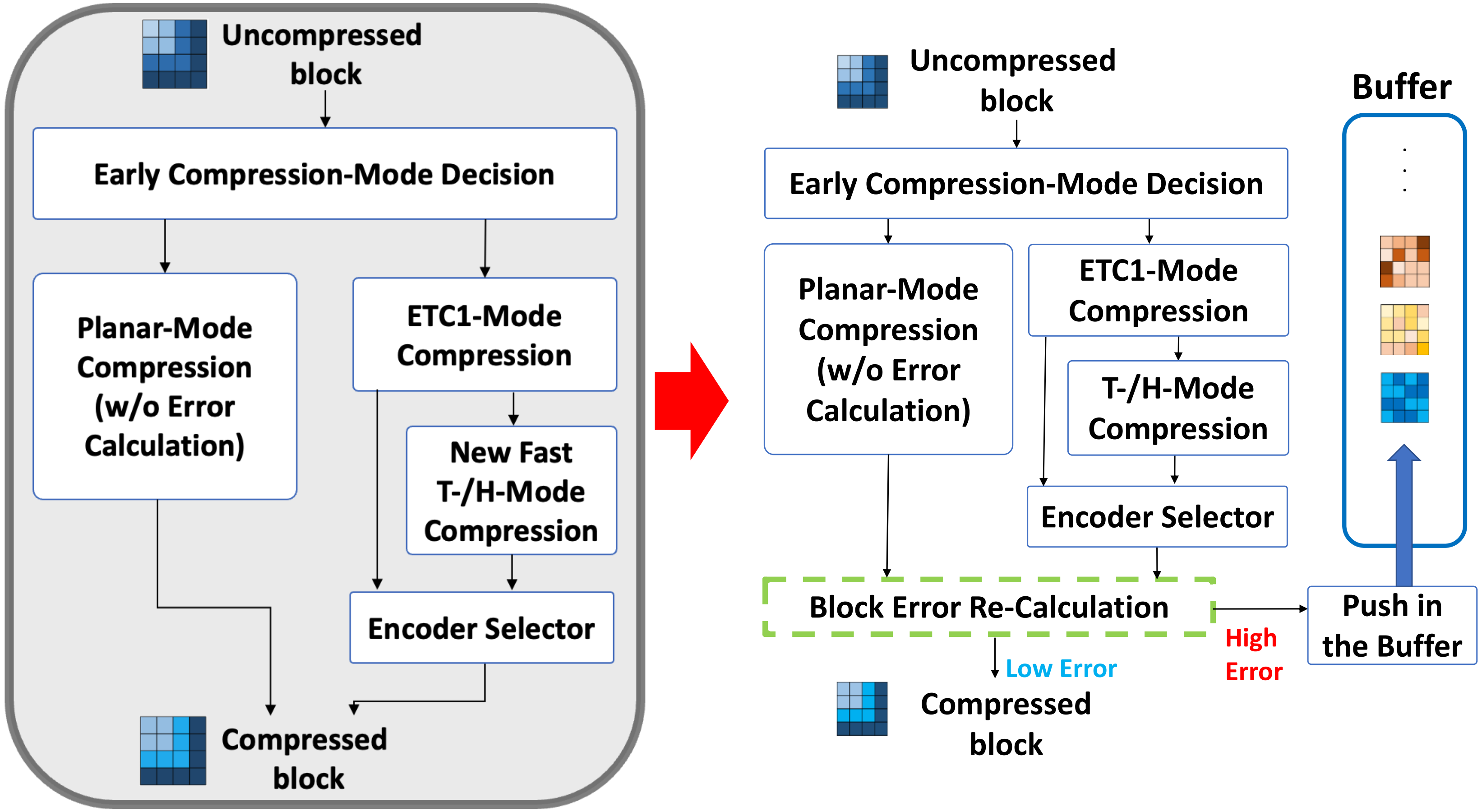
$\bar{x}$ : compressed pixel
$x$ : original pixel

- To be conservative and check the errors of each channel, we calculate the error as described above

- If the calculated error is greater than the threshold T, it is determined as a **problematic pixel block**
  - The threshold value used is based on ASTC encoder's "***dblimit***" (PSNR 35.68) [Smith. 2018]

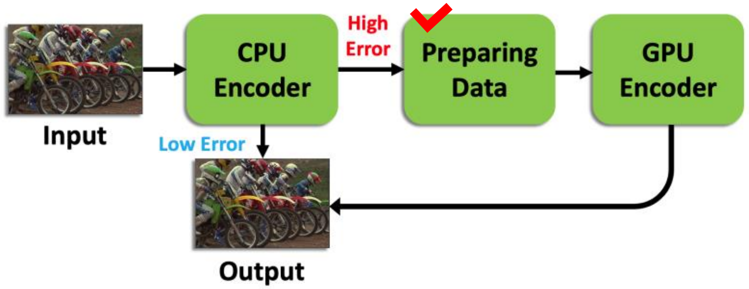
# Design of the CPU encoder

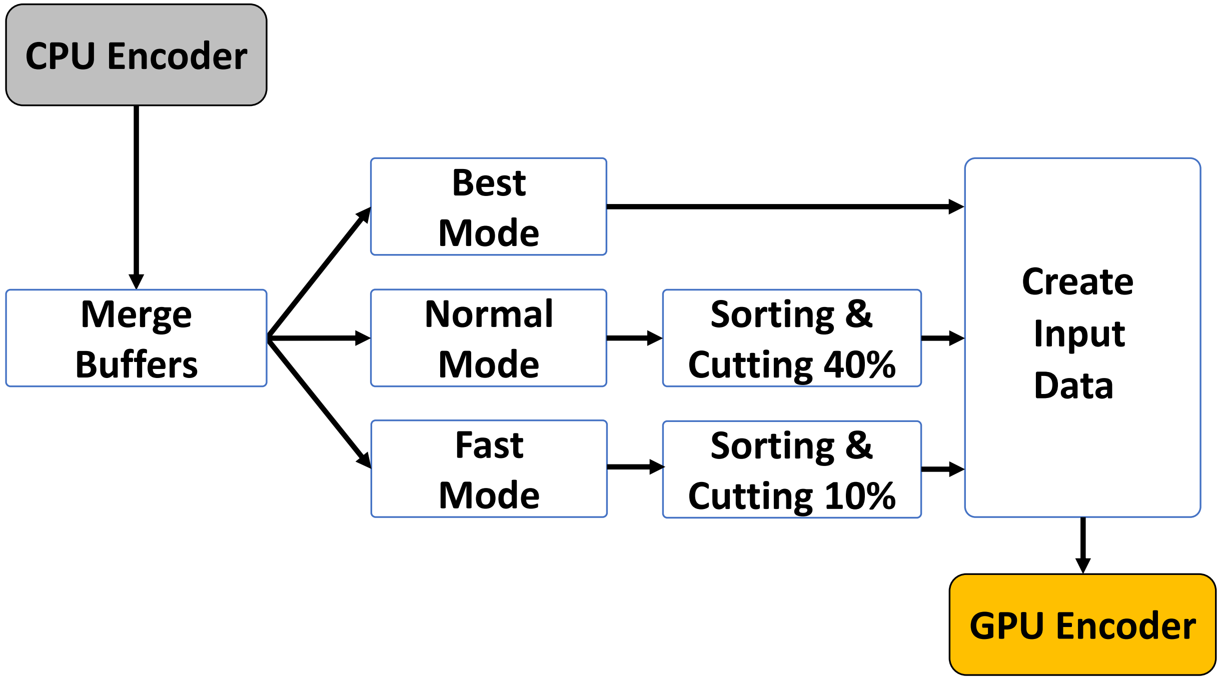


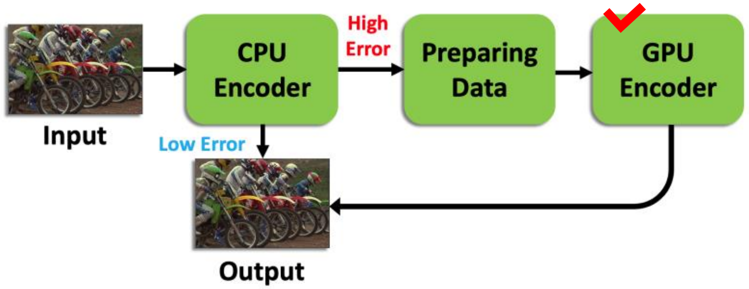

- Build upon QuickETC2 [Nah. SA2020], by adding the Block Error Re-Calculation process
- Result → high error?
  - Save in the local buffer of thread
- Result → low error?
  - Directly, save in output

# Preparing data for the GPU encoder

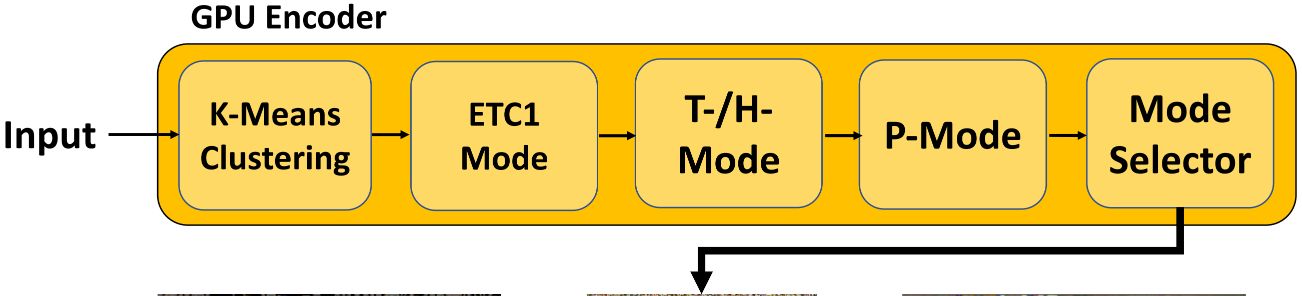



- We were inspired by Etc2comp [Google Inc. and Blue Shift Inc. 2017]
- A user can control the degree of quality
  - **Best mode**
    - No sorting,
      use **all** problematic block pixels
  - **Normal mode**
    - After sorting about errors,
      use only **40%** of all problematic pixel block
  - **Fast mode**
    - After sorting about errors,
      use only **10%** of all problematic pixel block




# Design of the GPU encoder





Result of the CPU Encoder

Result of the GPU Encoder

Output

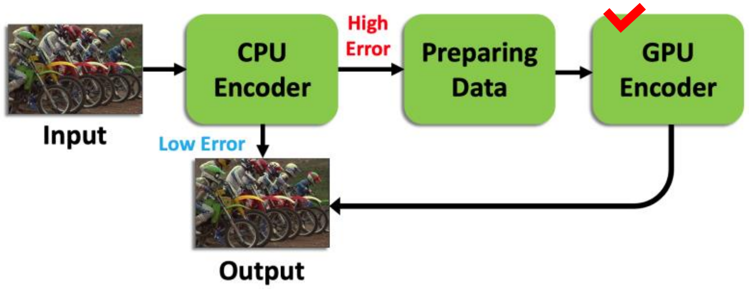
- Built upon Betsy [Goldberge. 2022]
- Two small changes that we did
  - Fixed quantization error
  - Applied perceptual error metric

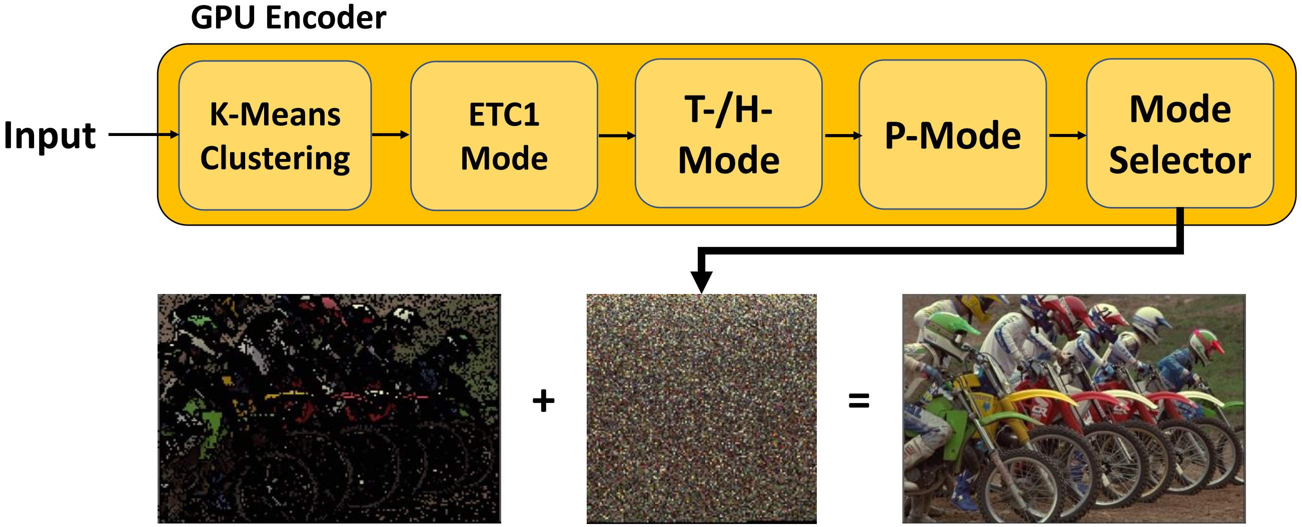
    $( error = 0.3 \times R + 0.59 \times G + 0.11 \times B )$

(iPACKMAN [Stro¨m and Akenine-Mo¨ller. GH2005])

- At the result, we could improve block artifacts

# Design of the GPU encoder




Result of the CPU Encoder

Result of the GPU Encoder

Output

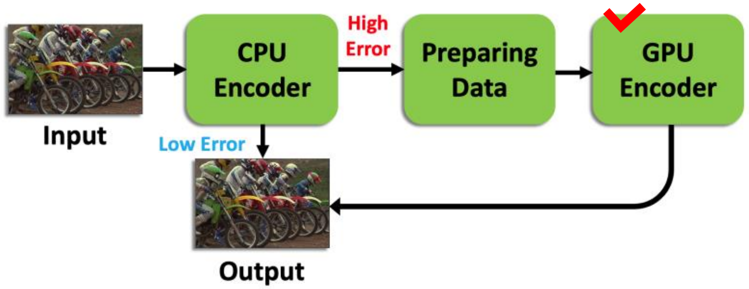
- Built upon Betsy [Goldberge. 2022]
- Two small changes that we did
  - Fixed quantization error
  - Applied perceptual error metric

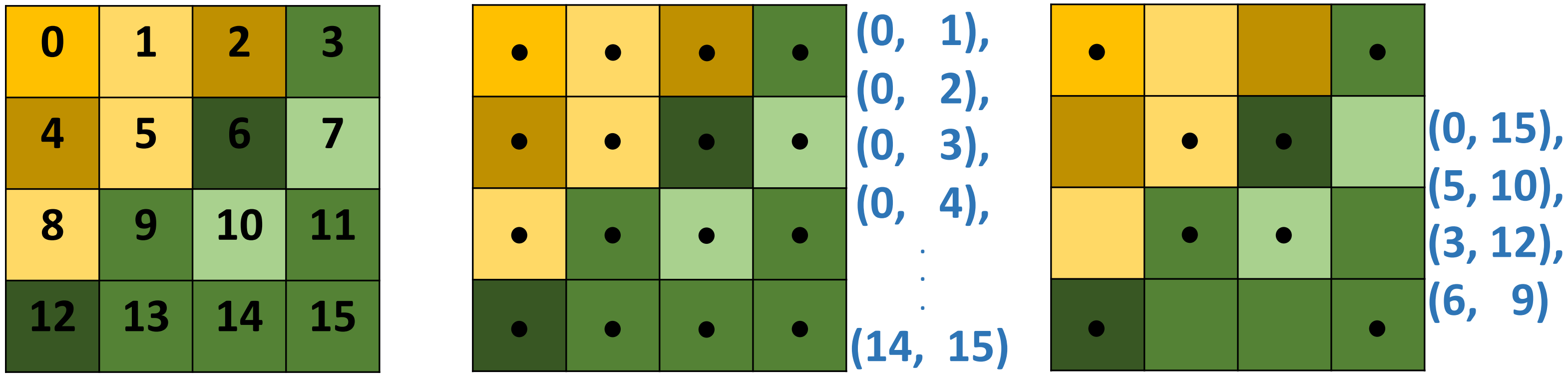
    $(error = 0.3 \times R + 0.59 \times G + 0.11 \times B)$

  (iPACKMAN [Stro¨m and Akenine-Mo¨ller. GH2005])
- At the result, we could improve block artifacts

→ **However, this GPU version is much slower than the etcpak CPU encoder!**

# Selective compression method



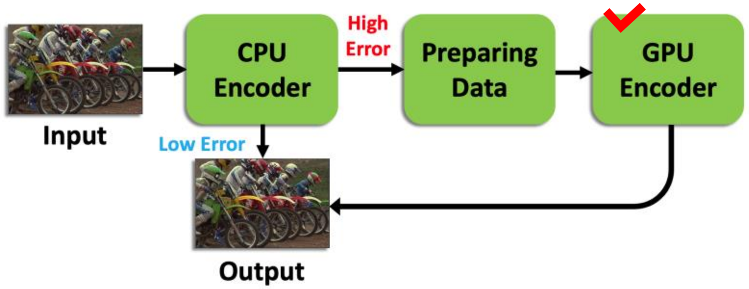




Index Table    Original Betsy GPU    Ours

- The traditional T-/H-mode was studied to improve the diagonally part (edge)
- We were inspired selective compression method of THUMB [Pettersson et al. SL2005]

- Improved encoding speed by using fewer pairs of pixel candidates $({}_{16}C_2 = 120 \rightarrow 4)$
  - T-/H-mode handles diagonally divided clusters better than ETC1 mode
  - Pixels within each individual partition represent spatial consistency

# Each step of improvement

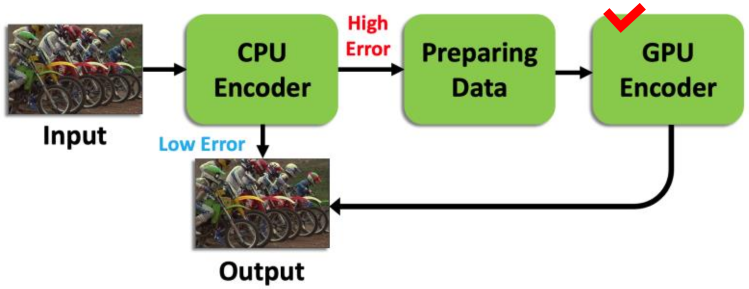



Original Betsy

+ Fix quantization errors

+ Apply the perceptual error metric

# Each step of improvement

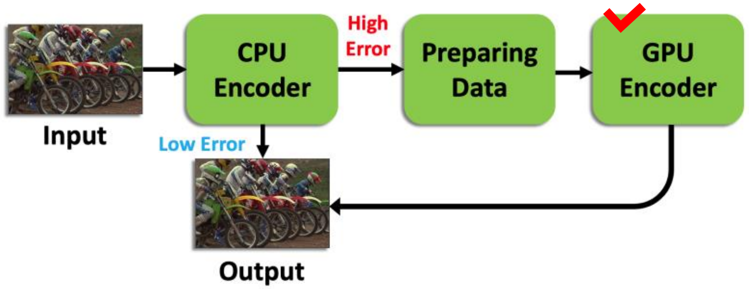


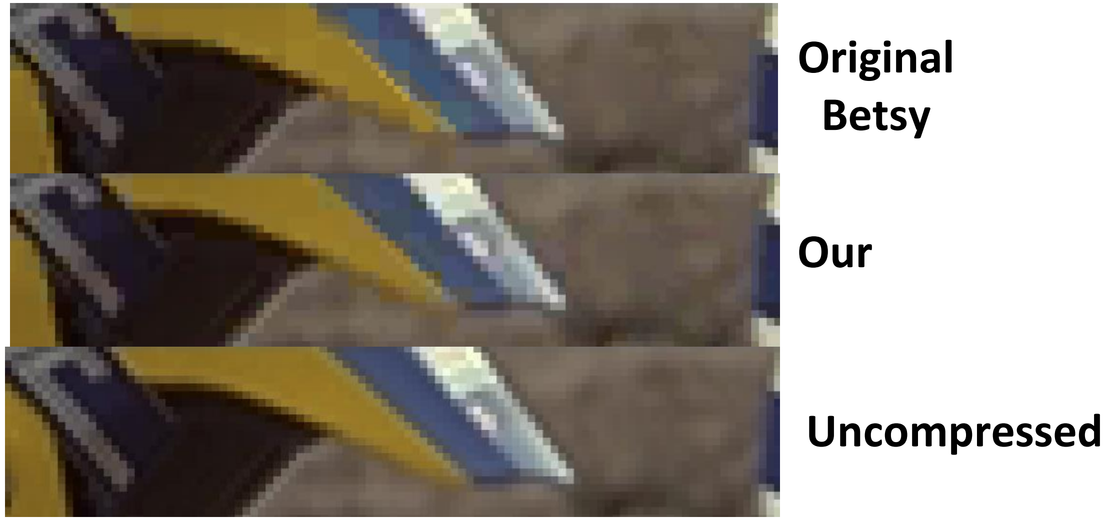

+ Selective compression method

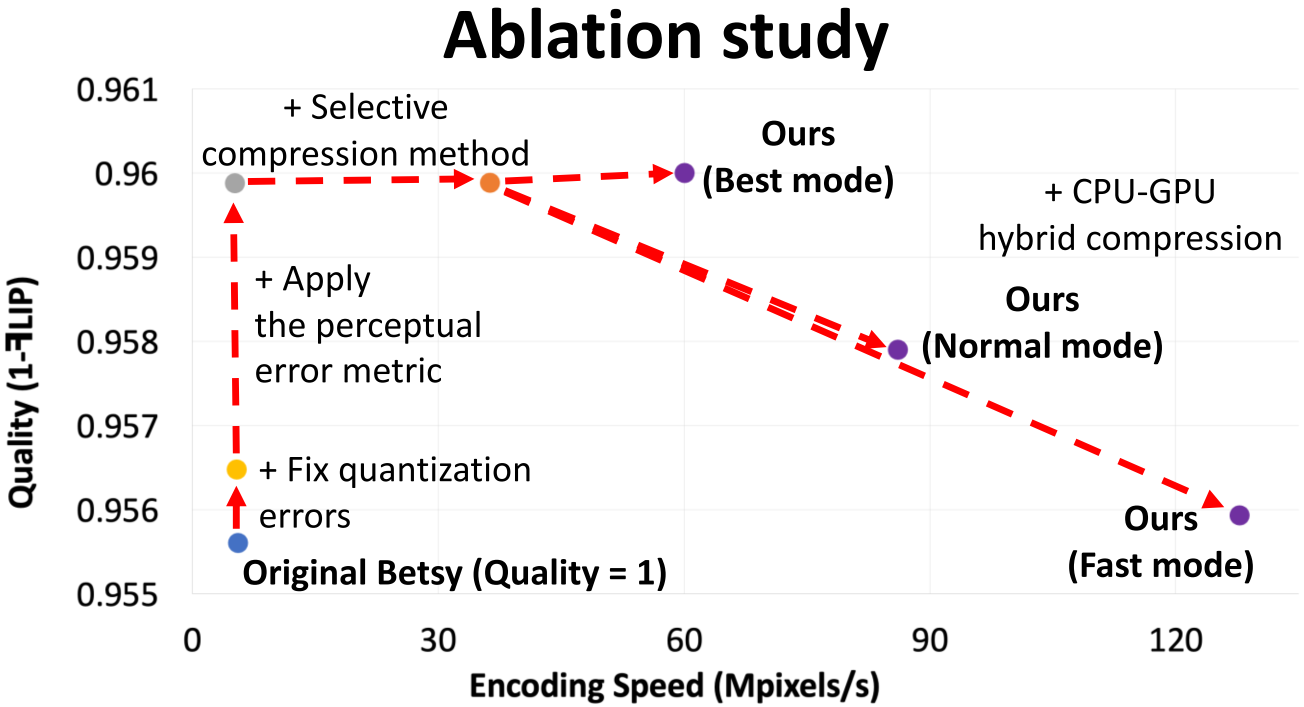
+ CPU-GPU hybrid compression (Best mode)

Uncompressed

# Each step of improvement

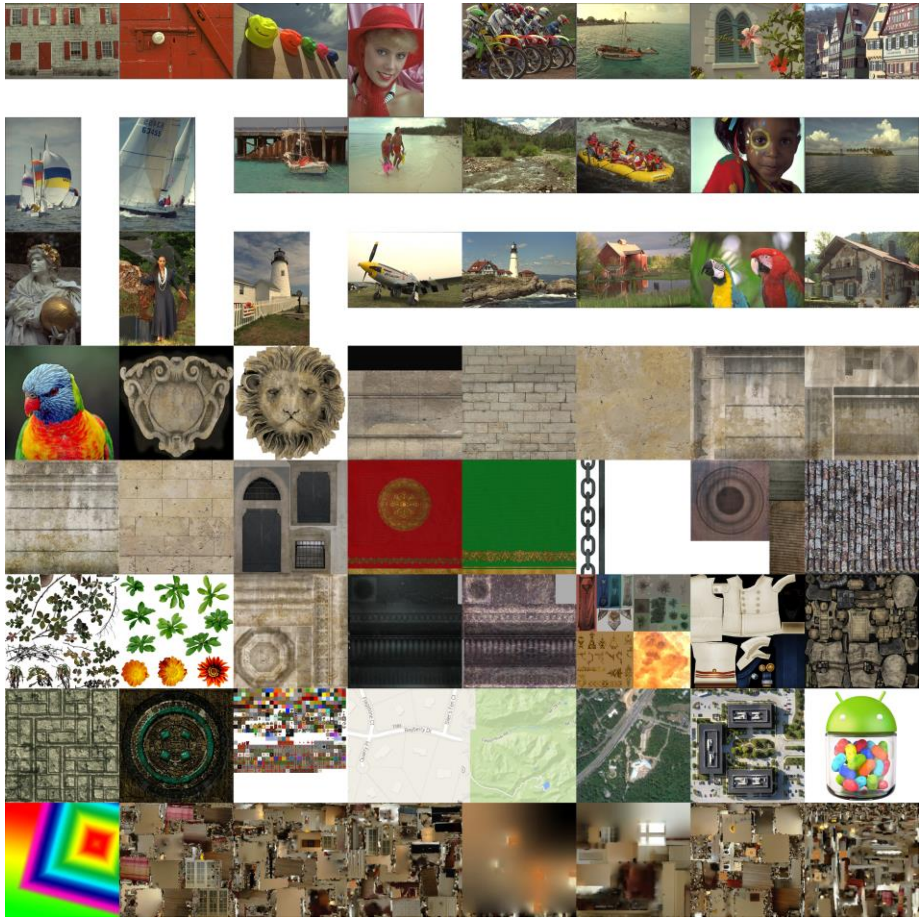


# Experiment & results

# Test images



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

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

# H/W & S/W setup

- Test hardware
  - Intel Core i5-12400 CPU, 32GB of RAM, NVIDIA GeForce RTX 3060, a 1TB SSD
- Evaluation Metric : ꟻLIP [Andersson et al. HPG2020], Mpixels/s
  - Lower ꟻLIP value indicates good quality
- Encoding settings
  - etcpak 1.0 (QuickETC2)
  - Betsy with 0, 1, and 2 as the quality parameters
  - Etc2comp with the fast and best modes
  - ETCPACK with the fast and slow modes
  - H-ETC2 (our) with the fast, normal, and best modes

# Quality & encoding speed comparison on the 64 test images

| Compressor | Mode | ꟻLIP | Mpixels/s |
|---|---|---|---|
| etcpak | | 0.0506 | 1350.82 |
| Betsy | Q=0 | 0.0474 | 6.20 |
| | Q=1 | 0.0444 | 5.63 |
| | Q=2 | 0.0438 | 2.22 |
| Etc2Comp | Fast | 0.0480 | 3.97 |
| | Best | 0.0419 | 0.15 |
| ETCPACK | Fast | 0.0419 | 0.85 |
| | Slow | 0.0375 | 0.0041 |
| H-ETC2 (ours) | Fast | 0.0440 | 127.87 |
| | Normal | 0.0421 | 86.15 |
| | Best | 0.0400 | 60.14 |

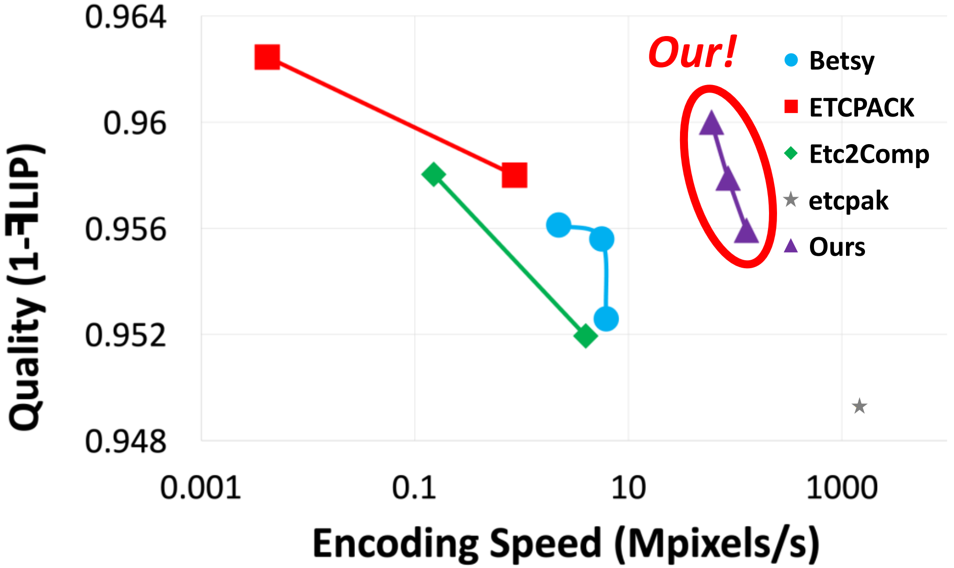
# Quality & encoding speed comparison on the 64 test images

*Jelly*

Original | etcpak | Betsy (Q=2) | Etc2comp (Best) | ETCPACK (Slow) | Our (Best)

→ Our (best) to ETCPACK (slow) show visually similar results

# Quality & encoding speed comparison on the 64 test images



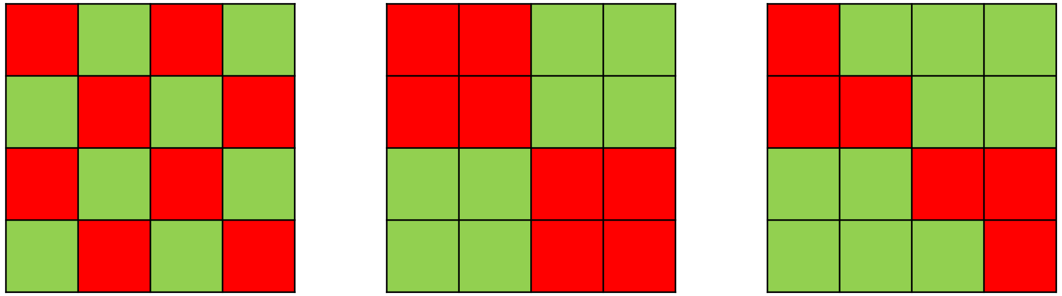

*kodim23*

Original | etcpak | Betsy (Q=2) | Etc2comp (Best) | ETCPACK (Slow) | Our (Best)

# Concluding remarks

# Concluding remarks

- We have introduced a hybrid ETC2 encoding pipeline that combines CPU and GPU processing
  - As a result, our encoder achieves a better balance between compression quality and encoding speed
- Limitations



  - limitation about extreme pixel pattern
  - Still slower encoding speed of GPU encoder than CPU encoder
- Future work
  - We aim to explore the applicability of our CPU-GPU hybrid approach to other texture formats, including BC7 and ASTC
  - Enhancing performance by refining the balance between CPU and GPU processing times

# Thank you!

# References


- [Andersson et al. HPG2020] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. **ꟻLIP: A Difference Evaluator for Alternating Images**. Proc. ACM Comput. Graph. Interact. Tech. 3, 2, Article 15 (August 2020), doi: https://doi.org/10.1145/3406183

- [Taudul. 2022] TAUDUL B.: **etcpak:the fastest ETC compressor on the planet**, 2022. URL: https://github.com/wolfpld/etcpak

- [Ericsson. 2018] ERICSSON: **ETCPACK**, 2018. URL: https://github.com/Ericsson/ETCPACK

- [Google Inc. and Blue Shift Inc. 2017] GOOGLE INC., BLUE SHIFT INC.: **Etc2Comp - texture to ETC2 compressor**, 2017. URL: https://github.com/google/etc2comp

- [Goldberge. 2022] GOLDBERG M. N.: **Betsy GPU compressor,** 2022. URL: https://github.com/darksylinc/betsy

- [Nah. SA2020] Jae-Ho Nah. 2020. **QuickETC2: Fast ETC2 texture compression using Luma differences**. ACM Trans. Graph. 39, 6, Article 270 (December 2020), doi: https://doi.org/10.1145/3414685.3417787

- [Ström and Petersson. GH2007] Jacob Ström and Martin Pettersson. 2007. **ETC2: texture compression using invalid combinations.** In Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware (GH '07). Eurographics Association, Goslar, DEU, 49–54. doi : https://dl.acm.org/doi/10.5555/1280094.1280102

- [Ström and Akenine-Möller. GH2005] Jacob Ström and Tomas Akenine-Möller. 2005. **IPACKMAN: high-quality, low-complexity texture compression for mobile phones.** In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware. Association for Computing Machinery, New York, NY, USA, 63–70. doi: https://doi.org/10.1145/1071866.1071877

- [Pettersson et al. SL2005] Pettersson, Martin, and Jacob Ström. "**Texture Compression: THUMB—Two Hues Using Modified Brightness**." Proceedings of Sigrad, Lund (2005): 7-12. URL: https://ep.liu.se/ecp/016/002/ecp01602.pdf

- [Smith. 2018] SMITH S.: **Adaptive scalable texture compression**. In GPU Pro 360 Guide to Mobile Devices. AK Peters/CRC Press, 2018, pp. 153–166.