A Practical Encoding Approach for Texture Compression: Combining Multi-Processing and Multi-Threading

Hyeon-ki Lee[1], Jae-Ho Nah[1]

[1]Sangmyung University, Seoul, South Korea

SIGGRAPH ASIA 2025 HONG KONG 香港

Problem

The texture compression process typically consists of three stages:

(1) Image Loading
(2) Encoding
(3) File Saving

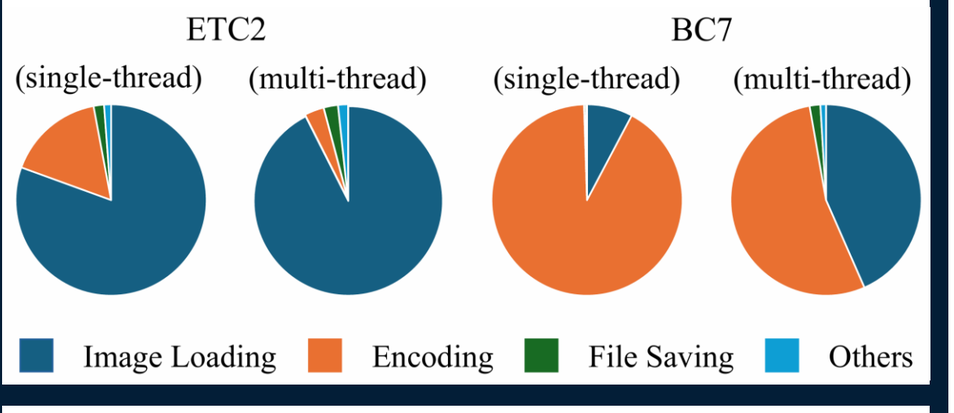
Prior Research Optimization Focus

- The encoding stage is often accelerated via multi-threading [2, 3, 4]

Remaining Bottlenecks

- The performance of encoding many textures in parallel can still be limited
  - Image Loading
  - Image Decoding (e.g., PNG, JPEG)

Time Ratio by Codec (ETC2, BC7)



Motivation

Modern CPUs have multiple cores

- CPUs enable flexible parallel execution
- Multi-processing and threading can boost encoding performance

Sorting Files by Descending File Size

- Larger files are loaded and encoded first
- Performance depends on file access strategy
- Workload is balanced and idle CPU resources are reduced

Reference

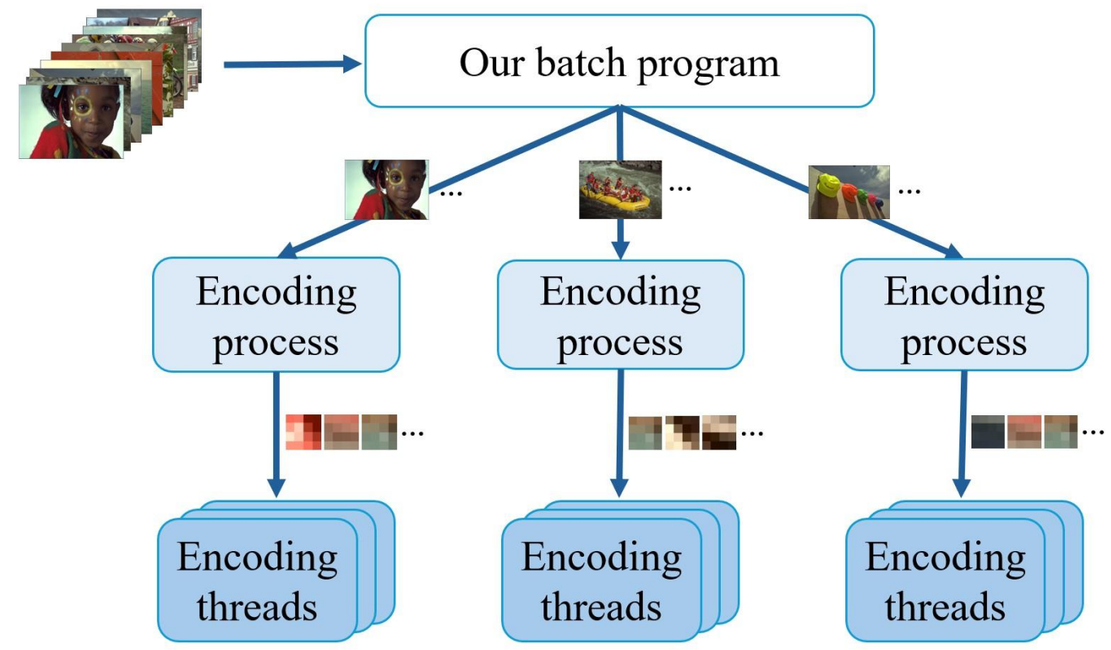
[1] Taudul, B. 2024. *etcpak 2.0*: The fastest ETC compressor on the planet. https: //github.com/wolfpld/etcpak
[2] Lee, H. and Nah, J.H. 2023. H-ETC2: Design of a CPU-GPU Hybrid ETC2 Encoder. *Computer Graphics Forum*
[3] Nah, J.H. 2020. QuickETC2: Fast ETC2 texture compression using Luma differences. *SIGGRAPH Asia (TOG).*
[4] Nah, J.H. 2023. QuickETC2-HQ: Improved ETC2 encoding techniques for real-time, high-quality texture compression. *Computer & Graphics.*

Author email: 202533021@sangmyung.kr

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025- 00521436).

Our Approach

System Overview



1. Sorting Files by Descending File Size

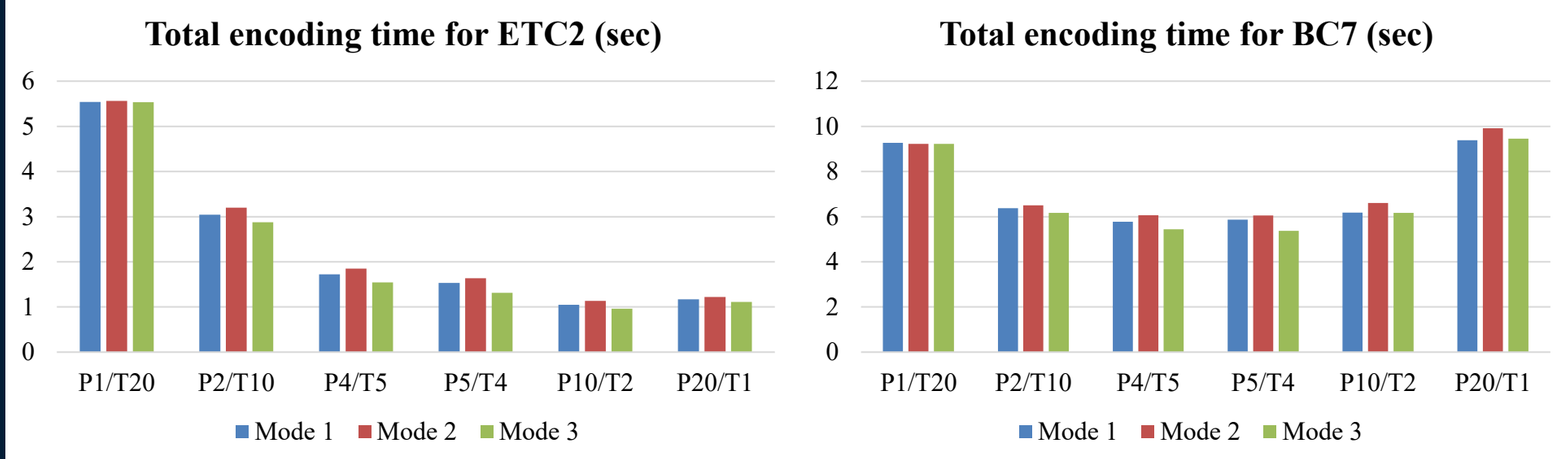
- Input files are sorted in descending order by size before encoding
- Workload is balanced by processing smaller files later

2. Combining Multi-processing and Multi-threading

- Multiple processes encode and load textures in parallel
- Multi-threading is performed at the block level within each process
- This method reduces overhead from image loading and saving while maintaining fast encoding speed

Results

We measured encoding times (sec) of our batch program (based on *etcpak 2.0* [1]) on 64 textures [3] using an Intel i7-12700 CPU (14C/20T), testing six Process/Thread (Px/Tx) configurations and three file access modes (random (M1), ascending (M2), descending by size (M3))



Our approach provides practical guidelines for processing large-scale texture datasets efficiently

- Up to 1.7x speedup for BC7 (P4/T5 or P5/T4)

Balanced parallelization between encoding and loading is essential for high-complexity codecs (e.g., BC7)

- Up to 5.76x speedup for ETC2 (P10/T2)

Multi-process configurations that parallelize image loading improve performance, as image loading dominates the total processing time