

# QuickETC2: How to Finish ETC2 Compression within 1 ms

Jae-Ho Nah  
nahjaeho@gmail.com  
LG Electronics













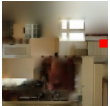


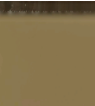
	Original	etcpak	Ours (planar only)	Ours (ETC2 full)	Etc2comp	ETCPACK
 Kodim05 (768×512)		0.23 ms		0.19 ms		105 ms
 Kodim20 (768×512)		0.31 ms		0.24 ms		392 ms
 Small-Char (512×512)		0.20 ms		0.15 ms		63 ms
 ISCV2_u2_v4 (8192×8192)		59.9 ms		48.0 ms		14.1 s
				48.3 ms	24.3 s	

Figure 1: Quality and performance comparison of our approach and other ETC2 compressors. Compared to etcpak, QuickETC2 in the planar-only mode is on average 23% faster with similar quality, and ours in the full ETC2 mode provides much better edge handling and less color distortion. Compared to Etc2comp and ETCPACK, ours is two to three orders of magnitude faster.

## CCS CONCEPTS

• Computing methodologies → Image compression.

## KEYWORDS

texture compression, ETC2

### ACM Reference Format:

Jae-Ho Nah. 2020. QuickETC2: How to Finish ETC2 Compression within 1 ms. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '20 Talks), August 17, 2020*. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3388767.3407373>

## 1 INTRODUCTION AND RELATED WORKS

With increasing graphics quality and screen resolution, the total size of textures in an app has reached up to dozens of gigabytes. It results in very long texture-compression time during software development. Additionally, some scenarios, such as compressing

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGGRAPH '20 Talks, August 17, 2020, Virtual Event, USA

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-7971-7/20/08.

<https://doi.org/10.1145/3388767.3407373>

image data for real-time 3D reconstruction [Meerits 2018], compressing on-the-fly generated textures in GIS tools [Krajcevski and Manocha 2014], resizing textures during app loading [Nah et al. 2018], improving memory utilization in web browsers [Oom 2016], and in-home streaming to mobile devices [Pohl et al. 2015], need real-time texture compression. As a result, the necessity of fast texture compression algorithms has increased.

Among several texture compression formats, ETC2 [Ström and Petterson 2007] is the OpenGL ES 3.0 standard and one of the widely-used formats on mobile devices. For ETC1/2 compression, the following tools are usually used. ETCPACK [Ericsson 2018] is a reference tool released by the first author of ETC1/2. Etc2Comp [Google Inc. and Blue Shift Inc. 2017] provides better trade-offs between quality and speed through a more targeted search, fine control of search-space exploration, and multi-threading. etcpak [Taudul and Jungmann 2019] aims at extremely fast ETC1/2 compression; this encoder consists of highly scalable parallel code and limits search spaces and the ETC2 mode (planar only).

The goal of our project is to increase the speed or quality of the existing compressors in two ways. First, we present an early compression-mode decision scheme that prevents unnecessary duplicated tests during ETC2 compression. Next, we present a new fast T-/H-mode compression algorithm to support the full ETC2 mode (T, H, and planar) without significant costs.

## 2 EARLY COMPRESSION-MODE DECISION

We observe that the three ETC2 modes assist ETC1 in different ways. The planar mode improves gradation in low-contrast regions more smoothly, but the T and H modes reduce blocky artifacts in high-contrast regions. Thus, if we can determine a proper compression mode in advance according to the luma difference (LD) ranges of a block, we can avoid duplicated compression (Figure 2).

We describe the procedure as follows. We first calculate the min/max luma values in a block, then we classify the block into one of four types: very low contrast ( $LD \leq 0.02$ ), low contrast ( $0.02 < LD \leq 0.15$ ), mid contrast ( $0.15 < LD < 0.4$ ), and high contrast ( $LD \geq 0.4$ ).

A very-low-contrast block can be compressed well in the planar mode. In the case of a low contrast block, we check whether the block can be smoothly expressed by the base colors at the corners of the block. If a pair of two corresponding corner pixels (top-left and bottom-right, or bottom-left and top-right) has the min and max luma values, we exploit a high possibility that the other pixels can be properly interpolated in the planar mode. Otherwise, we perform the ETC1 compression in etcpak as with mid-contrast blocks. Because a block compressed in the planar mode is not passed to the encode selector stage, the error calculation part in the planar mode is omitted for a speed-up.

A high-contrast block may create blocky artifacts, so we perform both the ETC1 and T-/H-mode compression. At the late encode selector stage, we compare the errors of the ETC1 and T/H modes and select the block with a lower error.

This early compression-mode decision is executed for all blocks, so its overhead should be minimized. To achieve this, all the parts described above were implemented using SSE/AVX2 operations.

## 3 NEW FAST T-/H-MODE COMPRESSION ALGORITHM

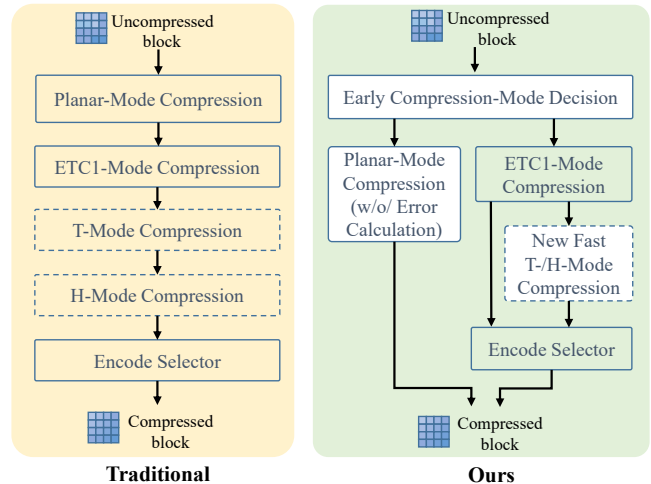
There are three reasons that the T-/H-mode compression in the previous works is time-consuming. First, clustering texels into two groups is expensive; ETCPACK uses the LBG algorithm. Second, the T and H modes are independently executed after the clustering; it doubles the time required. Third, all distances in the T and H tables are checked to find the lowest error.

To remedy the performance issues, we first replace the 3D RGB space with the 1D luma space and quickly find the two groups that have the minimum luma difference ranges. We also select a proper mode (T or H) and reduce its distance candidates, according to the luma range of each group. Finally, we implement our new algorithm using SSE/AVX2 intrinsics to exploit SIMD parallelism.

## 4 EXPERIMENTAL RESULTS AND ANALYSIS

For the comparison, we used the 64 textures which represent different types: photos, game assets, GIS map data, synthesized images, and captured images for 3D reconstruction (see the supplemental material). The experiment was performed on a desktop with an AMD Ryzen 7 3700X@3.6GHz 8-core (with hyper-threading) CPU. ETCPACK and Etc2Comp were executed with the fastest options.

Table 1 and Figure 1 show the results. Compared to etcpak, QuickETC2 in the planar-only mode achieves a 23% speed-up on average. QuickETC2 in the full ETC2 mode is two to three orders of magnitude faster than other tools that fully support the ETC2 modes.



**Figure 2: The ETC2-compression flow chart of the previous works (left) and ours (right). The dotted blocks can be excluded if speed is preferable to quality. The white blocks are newly introduced or modified in this paper.**

According to our additional experiments, our new T-/H-mode compression algorithm is 80× faster than that in ETCPACK, but the quality difference between them is negligible (PSNR -0.002 dB).

Our approach does not alter the ETC1 compression logic in etcpak, so ETC1 blocks compressed by ours and etcpak are the same. As a result, there are still quality gaps between ours and other high-quality compressors. We would like to explore how to improve the ETC1 compression logic as future work.

**Table 1: Experimental results with the 64 test textures. PSNR and SSIM values are related to quality, and million pixels per second (Mpix/s) are related to speed. Higher is better.**

Metric	etcpak	Ours (planar only)	Ours (ETC2 full)	Etc2Comp	ETCPACK
PSNR (dB)	37.17	37.27	37.40	38.35	38.52
SSIM	0.959	0.958	0.958	0.940	0.965
Mpix/s	1782	2588	2185	4.0	1.3

## REFERENCES

- Ericsson. 2018. *ETCPACK v2.74*. <https://github.com/Ericsson/ETCPACK>
- Google Inc. and Blue Shift Inc. 2017. *Etc2Comp - Texture to ETC2 compressor*. <https://github.com/google/etc2comp>
- Pavel Krajcevski and Dinesh Manocha. 2014. Fast PVRTC Texture Compression using Intensity Dilation. *Journal of Computer Graphics Techniques* 3, 4 (2014), 132–145.
- Siim Meerits. 2018. *Real-time 3D Reconstruction of Dynamic Scenes Using Moving Least Squares*. Ph.D. Dissertation, Keio University.
- Jae-Ho Nah, Byeongjun Choi, and Yeongkyu Lim. 2018. Classified texture resizing for mobile devices. In *ACM SIGGRAPH 2018 Talks*. 73.
- Daniel Oom. 2016. *Real-Time Adaptive Scalable Texture Compression for the Web*. Ph.D. Dissertation, Master's thesis, Chalmers University of Technology.
- Daniel Pohl, Bartosz Taudul, Richard Membarth, Stefan Nickels, and Oliver Grau. 2015. Advanced In-Home Streaming To Mobile Devices and Wearables. *International Journal of Computer Science and Applications* 12, 2 (2015), 20–36.
- Jacob Ström and Martin Pettersson. 2007. ETC 2: texture compression using invalid combinations. In *Proceedings of the Conference on Graphics Hardware*. 49–54.
- Bartosz Taudul and Daniel Jungmann. 2019. *etcpak 0.6.1*. <https://bitbucket.org/wolfpld/etcpak/src/master>