Build a Lossy Medical Image Codec with a Custom Bitstream

Chien, Y. C.
Department of Intelligent System and Application
National Yang Ming Chiao Tung University

Abstract – **Design and implement an end-to-end lossy compression system for medical images.**

Index Terms - Medical Image

I. Overview

This report is for Multi-model Image Processing class homework.

A. *Assignment Requirement*

Build a medical image compression system with two command-line tools:

- *encode: reads an input image (or a stack/volume), produces a compressed file (your format).*
- *decode: reads your compressed file, reconstructs the image (or stack/volume).*

Your codec must be lossy (i.e., reconstructed pixels may differ), but you must design it so that it preserves diagnostically relevant content as much as possible at a chosen bitrate.

B. *What I built*

I implemented a compact, reproducible image codec targeting single-slice CT images drawn from the Human_Skull_2 dataset. The codec pipeline implements (1) 8×8 block 2-D DCT, (2) scalar uniform quantization with a tunable quant step (quality), and (3) a lightweight entropy stage consisting of run-length encoding (RLE) followed by zlib compression. The bitstream contains a short header (magic/ version/ width/ height/ bitdepth/ blocksize/ quant_step/payload_len) followed by the compressed payload. Reference encoder/decoder scripts and experiment utilities are included so results can be reproduced.

C. *Why I did it — motivations and intended evaluation goals*

1) *Pedagogical baseline:* A minimal transform-quantize-entropy pipeline isolates core design choice (block size, quantization step, entropy scheme). This makes it easy for graders to verify correctness, reproduce results, and tell the reason about where gains/losses come from.
2) *Controlled RD experiments:* By varying a single quality parameter we obtain clear rate–distortion tradeoffs. This lets us quantify how compression affects pixel fidelity (RMSE/PSNR) and visualize error maps to inspect structure-dependent artifacts.
3) *Clinical relevance:* CT is high bit-depth and diagnostically sensitive. The project tests whether a simple codec can preserve diagnostically relevant signal while reducing storage/transmission cost — a practical concern in PACS, teleradiology, and mobile/edge scenarios. Emphasis is on measuring fidelity (PSNR/RMSE) and showing qualitative reconstructions/error maps for clinical plausibility.
4) *Reproducibility & simplicity:* The implementation is pure Python with small, well-documented scripts (encode/decode/run). This reduces friction for reviewers to run the experiments, modify components (e.g., replace quantizer, turn off RLE), and reproduce ablations.
5) *Ablation-friendly design:* The pipeline allows targeted ablations (e.g., with/without RLE, alternate quantizers, different block sizes) to demonstrate which components contribute most to compression gains and to artifact formation.
6) *Practical constraints:* The design intentionally favors interpretability and runtime efficiency (no heavy ML models), making it realistic to run on laptops or limited servers and suitable for course deadlines.
7) *Safety & privacy considerations:* The codec focuses on pixel data only; metadata handling is documented so patient identifiers can be preserved or redacted as required, aligning with common clinical data governance needs.

II. Design Format

A. *Bitstream specification*

Bitstream specification format

1) *Header* (big-endian/network order):
   - *Magic* (4 bytes) : ASCII "MMPC"
   - *Version* (1 byte)
   - *Width* (2 bytes, unsigned)
   - *Height* (2 bytes, unsigned)
   - *Bitdepth* (1 byte) : 8 or 16
   - *BlockSize* (1 byte) : 8
   - *QuantStep* (2 bytes, unsigned)
   - *PayloadLen* (4 bytes, unsigned)

2) *Payload*
   - *Coefficient ordering:* For each 8x8 block (blocks ordered in raster order: row by row, top-left to right and change to next row), flatten block

coefficients in row-major (i.e., row0 col0..7, row1 col0..7, ⋯). After previous, append each block's 64 quantized int16 coefficients in that order to form a coefficient array of length Ncoeff = nBlocks * 64.

- *int16 byte order in payload*: Little-endian representation for each int16 coefficient is used (i.e., low byte first). This choice aligns with common native machine layout and with typical numpy.astype(np.int16).tobytes() behavior on little-endian hosts, So the raw coefficient-bytes stream length = Ncoeff * 2 bytes.
- *RLE encoding* (applied to the coefficient-bytes stream): RLE works on bytes (0–255). Encoder output is a sequence of pairs [count (1 byte), value (1 byte)] repeated. Each pair means "repeat value exactly count times" that count is a single unsigned byte (1 - 255). If longer runs exist, the encoder should emit multiple pairs (e.g., 300 zeros → [255,0] [45,0]).
- *Compressed payload*: The encoder use zlib.compress(RLE_bytes) to compress the RLE byte sequence and decoder reads PayloadLen bytes, calls zlib.decompress(...) to obtain RLE_bytes.
- *Fallback / detection*: If zlib.decompress(...) succeeds but the RLE decoded length does not match Ncoeff * 2, the decoder should attempt to interpret the decompressed bytes as raw int16 bytes (no RLE). Otherwise, if that still doesn't yield the expected length, raise a format error.

B. *Decoding steps*

2.1) Parse header and check magic/version: Read first 17 bytes, parse header with big-endian unpacking. Validate Magic == b"MMPC" and Version supported.

2.2) Read PayloadLen bytes, decompress with zlib: Check if PayloadLen is shorter than request then show error. Also use payload = zlib.decompress(payload_bytes) to decompress the file. If decompression fails, error (corrupted payload).

2.3) RLE-decode the byte stream to int16 coefficient array: Attempt RLE decode returns bytes (raw_coeff_bytes = RLE_decode(payload)). To try to interpret decompressed output as raw int16-bytes. And if it still mismatch, raise payload length mismatch.

2.4) Reshape coefficients: Reshape N Blocks into 8×8 blocks in row-major order and de-quantiz (coeff = q * QuantStep).

2.5) Apply inverse DCT on each block: Precompute the DCT matrix M such that C = M @ f @ M.T and f = M.T @ C @ M when M is defined using the alpha * cos(...) entries above. That ensures encoder/decoder symmetry.

$$C_{u,v} = \alpha_u \alpha_v \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f_{x,y} \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)v}{2N}\right) \quad (1)$$

where

$$\alpha_0 = \sqrt{\tfrac{1}{N}}, \quad \alpha_k = \sqrt{\tfrac{2}{N}}\ (k > 0). \quad (2)$$

And inverse DCT is:

$$f_{x,y} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha_u \alpha_v\, C_{u,v} \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)v}{2N}\right). \quad (3)$$

2.6) Place reconstructed blocks into a padded image grid of size.

2.7) Clip to [0, 2^Bitdepth − 1] and output image (or embed back into DICOM).

III. RESULT

A. *Rate–distortion table:* I evaluate rate–distortion behavior on the Human_Skull_2 (CT) dataset. The table below reports compressed size (bytes), bits-per-pixel (bpp), RMSE and PSNR for three operating points (quality = 10, 30, 60) on a representative slice. In this submission we use slice I0 from Human_Skull_2 as the worked example: the three RD points shown correspond to encoding/decoding of I0 at the listed quality settings. A fuller submission should report aggregate RD statistics across multiple slices; here I0 is provided as an illustrative single-slice example for reproducibility and visualization.

TABLE I
Rate–Distortion points for example slice I0 (3 operating points)

Quaility	Size (bytes)	bpp	RMSE	PSNR (dB)
10	67,103	1.7518	2.2373	89.33
30	27,452	0.7167	4.2758	83.71
60	16,128	0.4210	6.6731	79.84

Table I reports compressed size, bpp, RMSE and PSNR for quality settings 10, 30 and 60. As expected, increasing the quantization step (higher quality in our encoder) reduces bpp but increases RMSE and lowers PSNR — the classic rate–distortion tradeoff. For the example slice I0, quality = 30 achieves a practical tradeoff (bpp≈0.72, PSNR≈83.7 dB), substantially reducing storage while retaining high pixel-level fidelity.

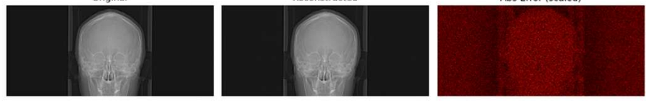

Fig. 1: I0 (quality = 10)

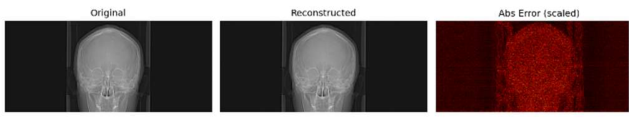

Fig 2: I0 (quality = 30)

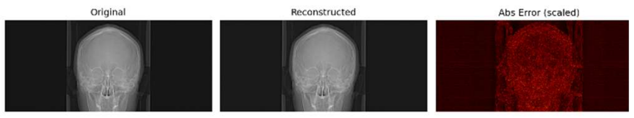

Fig. 3: I0 (quality = 60)

I visualized reconstructions at three encoder quality settings to compare differences. Figure 1 shows that, at the highest-quality setting, the reconstructed image is nearly indistinguishable from the original: bone boundaries, the nasal cavity, and fine bone texture are well preserved. The scaled absolute-error map reveals primarily small, spatially diffuse noise; bone outlines are faintly visible but with very low intensity (consistent with RMSE ≈ 2.24 and PSNR ≈ 89 dB). This observation implies a very small quantization step (high quality), so DCT coefficients are largely preserved. Consequently, the entropy stage (RLE + zlib) contributes a larger fraction of the bitstream at this operating point, which explains the relatively high bpp.

At the mid-quality operating point (Figure 2), the reconstructed image remains visually very similar to the original, but fine skeletal details—particularly minute bone textures—are noticeably smoothed and some small features appear dimmer. The overall bone outline and major edges are still well preserved. The scaled absolute-error map shows errors concentrated in the skull region (the central skull area appears brighter), whereas the background and blank side regions exhibit low error. This pattern indicates that high-frequency components associated with important anatomical structures are most affected by quantization, while the global error magnitude remains modest (PSNR ≈ 83.7 dB). In practice, this represents a common and often useful rate–distortion trade-off: the bitrate is substantially reduced (lower bpp) while preserving diagnostically relevant gross anatomy, but high-frequency detail is degraded—therefore task-specific validation (e.g., for lesion detection or fine-structure assessment) is recommended before clinical use.

Figure 3 shows the reconstructed image and scaled absolute-error map for slice I0 at the low-quality operating point (encoder quality = 60). Visually, the reconstructed image is noticeably smoother than at higher qualities: fine intracranial bone textures and other high-frequency details are substantially attenuated, and edges show mild blurring. The scaled error map highlights the skull silhouette much more strongly than at q=30; errors are both brighter and more spatially widespread inside the skull region, indicating a larger and more structured distortion field. Quantitatively, this matches the measured metrics (RMSE ≈ 6.67, PSNR ≈ 79.84 dB), which show an increased error magnitude relative to higher-quality points. The data imply that a large quantization step is being applied at q=60: many high-frequency DCT coefficients are driven toward zero or heavily rounded, producing smoothing and loss of fine features. While this operating point yields marked bitrate savings (bpp ≈ 0.42 in our example), it also reduces the visibility of diagnostically relevant micro-structure. Therefore, q=60 may be acceptable for storage- or transmission-constrained use cases where only coarse anatomical information is required, but it is not recommended for tasks that rely on subtle high-frequency cues (e.g., detecting small fractures, micro-calcifications, or fine trabecular patterns) unless validated by downstream clinical task evaluation.

IV. Limitations & future work

A. *Limitations:*

1) Entropy stage is simplistic: RLE + zlib is easy to implement but not bit-optimal compared to context-adaptive coders (Huffman, arithmetic, ANS).
2) Blocking artifacts: independent 8×8 block quantization can create blocking at low bitrates; no deblocking or in-loop filters implemented.
3) Single-slice only: no inter-slice prediction or 3-D modeling for volumetric CT.
4) DICOM metadata handling is minimal — patient / study tags are not preserved in the bitstream by default.

B. *Future work:*

1) Replace RLE+zlib with a tuned entropy coder (ANS or context model) and measure RD gains.
2) Add spatial predictors (intra prediction) to reduce DC energy prior to transform.
3) Implement deblocking or post-filtering to reduce perceptual artifacts at low bitrates.
4) Extend pipeline to multi-slice volumes with inter-slice prediction and optionally test clinical task-driven metrics (lesion detection / segmentation fidelity).

Acknowledgment

Here have the whole work here: b1029009Chien/MMIP: multiple-model image proceesing