ZinziPEG: a Low-complexity and Error Resilient JPEG compressor for Smart Camera Network

Daniele Campana, Marco Giglio, Matteo Petracca, Claudio Salvadori
TeCIP Institute, Scuola Superiore Sant'Anna, Pisa
Consorzio Nazionale Interuniversitario per le Telecomunicazioni

Main goal and environment

To develop a video coder with two main features:

- High compression level;
- High error resilience properties on wireless channels.

The considered environment:

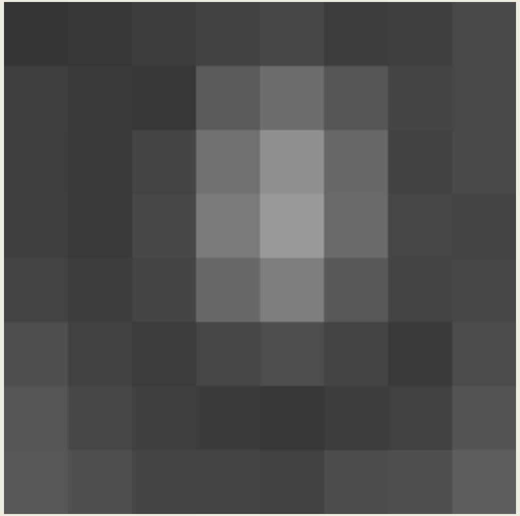
- IEEE802.15.4 compliant networks
- Low-rate networks;
- High bit error rate values.

JPEG: working principles

The JPEG standard is based on:

- Discrete Cosine Transform (DCT)
- Huffman encoding (loss-less entropy encoding)

JPEG example



8x8 pixel block

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}.$$

Discrete Cosine Transform

$$G = \begin{array}{c} u \\ \longrightarrow \\ \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \end{array} \Big\downarrow v.$$

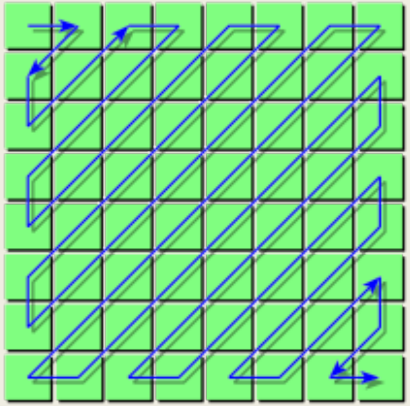
Quantization

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

$$G = \begin{array}{c} \overset{u}{\longrightarrow} \\ \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \end{array} \Big\downarrow v.$$

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Zig Zag re-ordering



$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

−26
−3 0
−3 −2 −6
2 −4 1 −3
1 1 5 1 2
−1 1 −1 2 0 0
0 0 0 -1 -1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0
0 0 0 0
0 0 0
0 0
0

Huffman Encoding

- Lossless data compression:
  - Entropy encoding
  - Can be seen as a *variable-length* code table for encoding a source symbol (such as a character in a file)
  - The more common symbols are generally represented using fewer bits than less common symbols

JPEG advantages

- Good compression level
  - Compression quality ranging form 1 to 100
  - Ratio between quality and size variable from image to image.
- JPEG is largely adopted

JPEG disadvantages

- **The image header dimension is very big**
  - It embeds quantization tables, huffman tables, etc…
  - ~500 bytes.
- **No error resiliency properties**
  - Error in the header: the decoding might converge to wrong results or it might fail.
  - Error in the data: decoding failure or wrong decoding
  - DC of blocks is correlated: the errors are propagated to following blocks

The ZinziPEG

ZinziPEG is based on an integer JPEG compressor written in C.
We performed several modifications in order to achieve the desired goals:

- markers
- header remotion
- packetization (oriented to IEEE802.15.4 standard)
- recovery and concealment on receiver side

The ZinziPEG Encoder

It implements:

- Header remotion
  - Removed about 500bytes (great impact in low quality images)
- Packetization (oriented to IEEE802.15.4 standard)
- Trailer
- Markers insertion

Header removal

- The JPEG header is very big (~500 bytes)
- The CODEC parameters (quantization tables, huffman tables, etc…) are set in the init-phase
  - Improve the compression
  - Improve the resiliency

Error in the header: it is impossible to decode the image!!

Packet fragmentation

- An integer number of block in each data packet
  - The first block of each packet is a safe point
  - If a packet is lost, the next one can be easily decoded
- The available payload in 802.15.4 networks is only 104 bytes.
- Each packet contains an applicative trailer to describe what the system is transmitting
  - Protected by using a FEC tecnique (Hamming Code (40,7))

The packet trailer

R	FLAGS	LAST_BLOCK	NUM_BLOCKS	NUM_BITS
1	2	13	7	10

R	1 reserved bit
FLAGS	2 bits used for fragmentation purposes
LAST_BLOCK	13 bits containing the id of the last 8x8 block wich has been inserted in the packet.
NUM_BLOCKS	7 bits to represent the number of blocks contained in this packet.
NUM_BITS	10 bits representing the lenght of the zero padding

Standard JPEG markers

- Standard JPEG mechanism:
  - 2 bytes markers inserted every *n* blocks
  - used to decorrelate the DC, to repair block alignment and a safe start point to start reading whenever an error is encounter during Huffman decoding
  - If 9 consecutive markers are lost, the decompression fails

The ZinziPEG markers

- ZinziPEG mechanism:
  - provide DC decorrelation and safe start point (as JPEG);
  - markers are only 1 byte long (less memory overhead)
  - The use of both markers and trailers allows stronger resiliency
    - removed the constraints on the number of consecutive markers that can get lost

The ZinziPEG decoder

- The ZinziPEG decoder translate the encoded image in standard JPEG
  - Corrects errors due to noise on the channel (by using the FEC decoder)
  - Reconstruct the image and add the JPEG standard header
  - *"Grey concealment"* is performed whenever there is a corrupted block
- The ZinziPEG decoder ALWAYS returns a correct JPEG image, but of course some of its blocks might be corrupted (bit-flips on the DCT coefficients) or replaced with grey ones (grey concealment).
- JPEG might not converge in case of bit flips

Experimental setup

- Comparison between ZinziPEG with standard JPEG (with markers)
- Experiments on simulated channels with high BERs (i. e., BER = 5e-2)
- Metrics:
  - compressed image size (bytes);
  - quality of the received images (SSIM).

Structural SIMilarity index

S-SIM “*is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality*.”

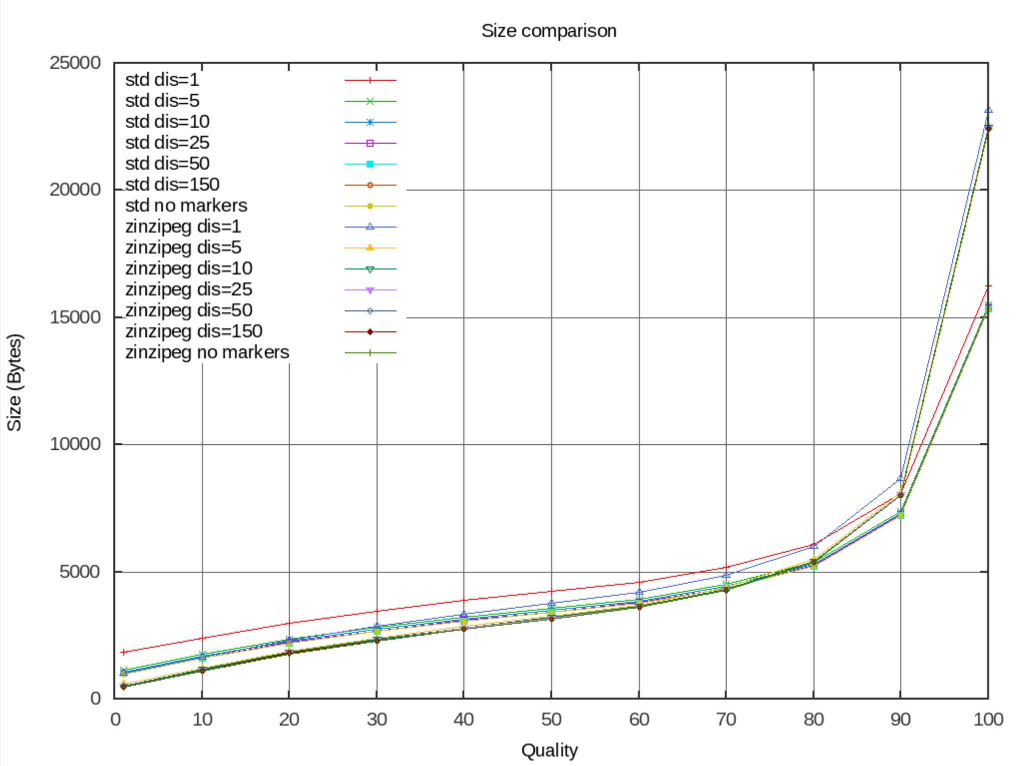
Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, *"Image quality assessment: From error visibility to structural similarity,"* IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.

Image size

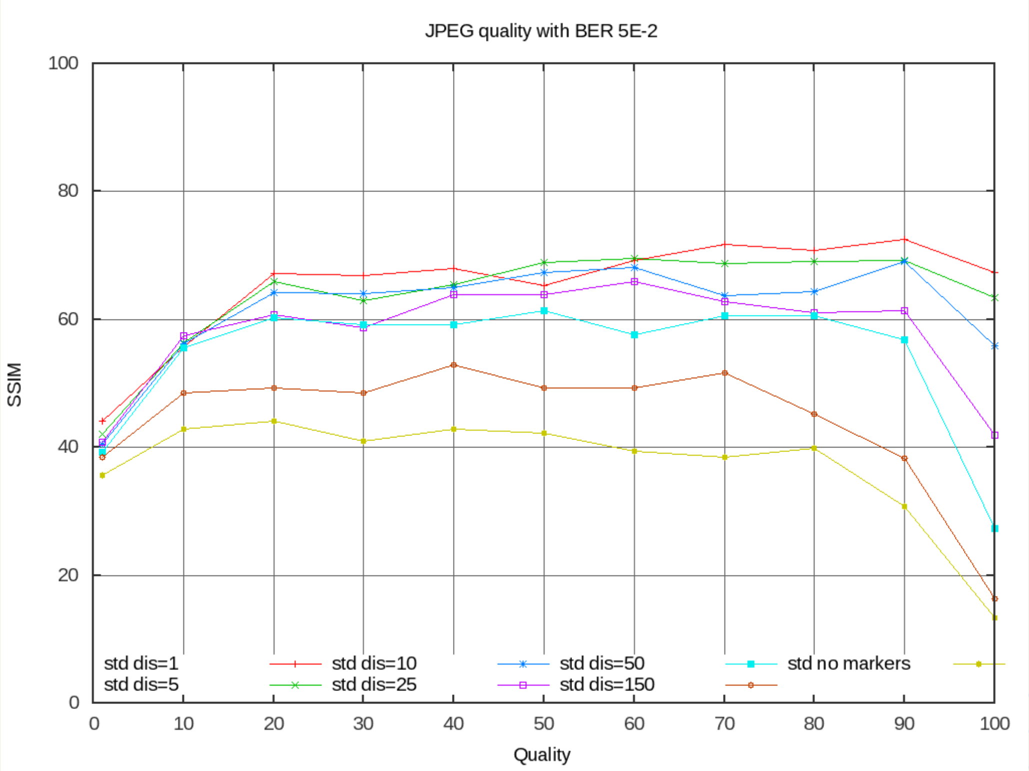
Low quality:
ZinziPEG might even outperform JPEG thanks to the reduced overhead due to header remotion.

Medium quality:
ZinziPEG files are only slightly bigger than JPEG ones.

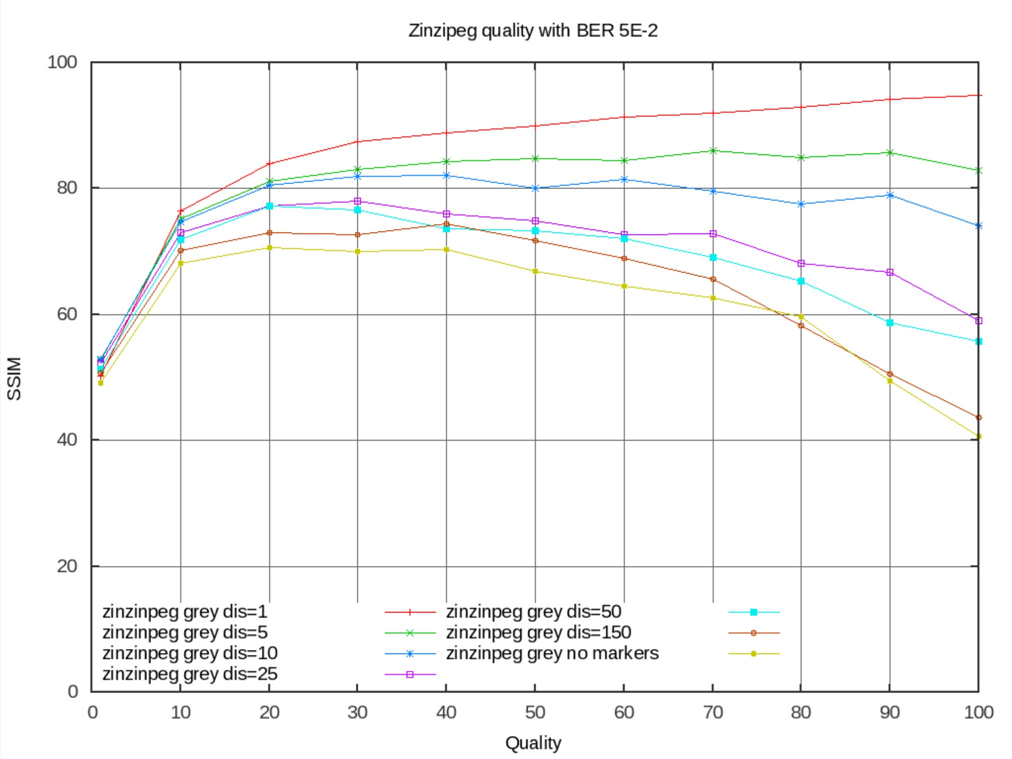
High quality:
ZinziPEG requires much more space than JPEG due to the increased **number of trailers** to be sent.



Quality comparison (JPEG)



Quality comparison (ZinziPEG)



Conclusions

- JPEG-like compressed image size
- Huge error resiliency improvement
- Simple implementation, suitable for micro-controllers based embedded systems
  - Suitable for micro-controllers W/O FPU, since the integer implementation