Lossy Compression of Images

Overview of various approaches

•••

Desenho e Análise de Algoritmos

Professor: Carlos Lourenço

Miguel Falé, fc43556
Miguel Rodrigues, fc44281

Context

- Lossless compression
  - No losses
  - Important when the decompressed version needs to be identical to the original
- **Lossy compression**
  - Loss of some information compared to the original
  - Important approach for reducing data size
    - Often better reduction rates than in lossless methods
  - No significant degradation is noticed by the end-user
    - Unless if it is constantly compressed/decompressed
    - Generation loss

Context

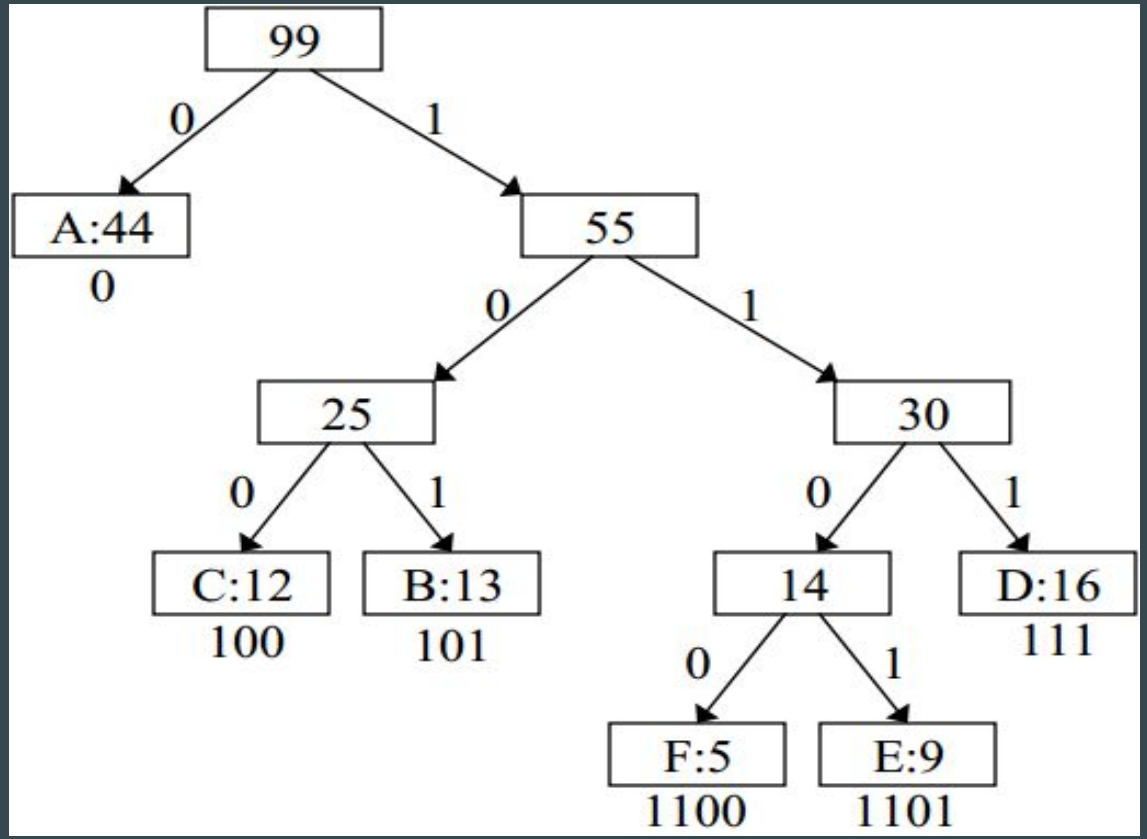
- For this project, we explored means for **lossy** compression of **images**

- Important concepts to take note of
  - **Mosaics**. Pieces of an image, with certain pixel sizes
  - **Pixel**. Atomic component with an associated color.
  - **RGB**. Popular color system formed by Red, Green and Blue components.
  - **Euclidean distance.** Distance between 2 points, useful for finding good prototypes
  - **Prototypes.** Mosaics that are the **most representative** of an image.
    - Can be obtained from other images, then applied to the input image

Overview

1. Select Image
2. Select prototypes
3. Re-do image by replacing each mosaic with a prototype
4. Build dictionary
5. Compress such Dictionary
6. Decompress
7. See the results

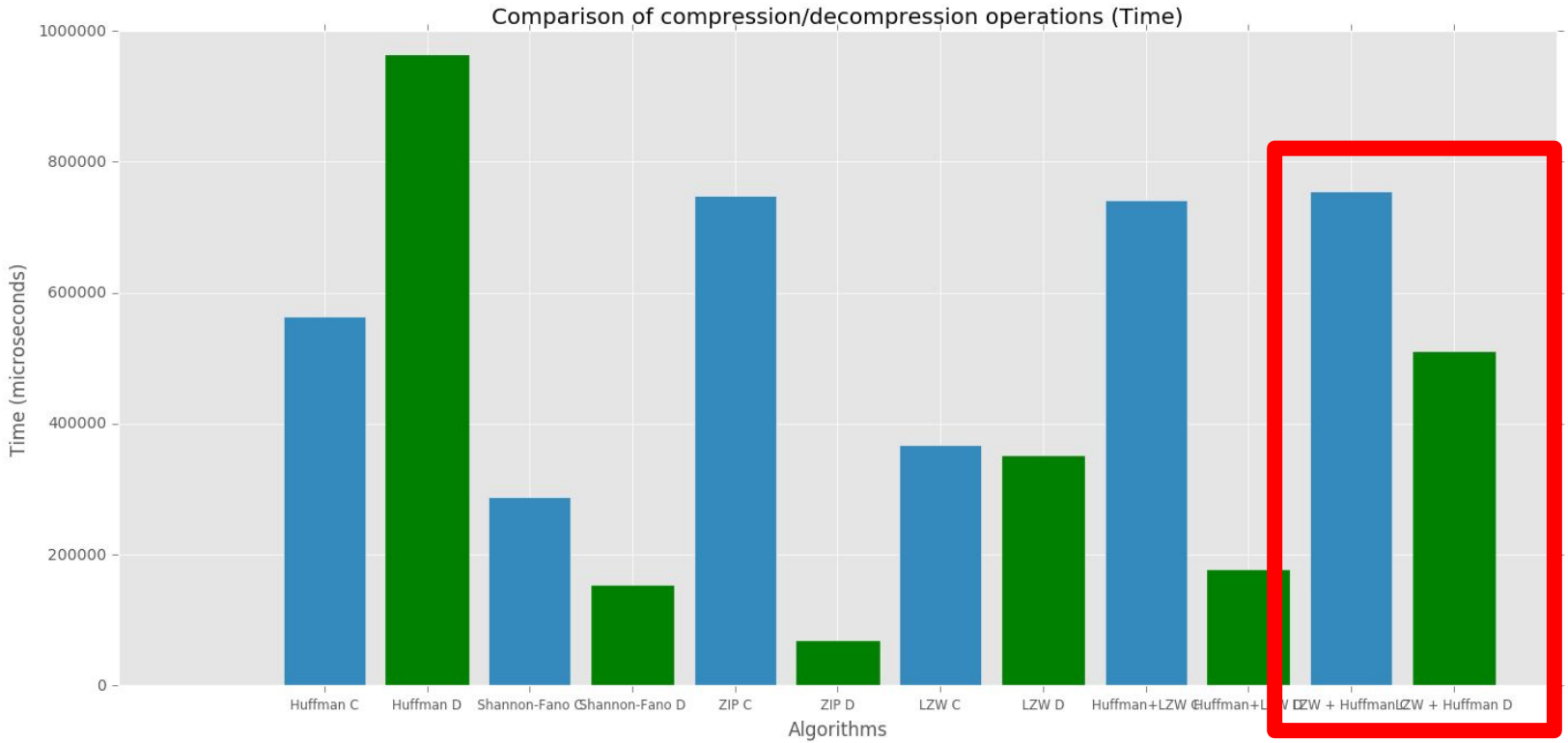
Huffman


99
0
1
A:44
0
55
0
1
25
30
0
1
0
1
C:12
100
B:13
101
14
D:16
111
0
1
F:5
1100
E:9
1101

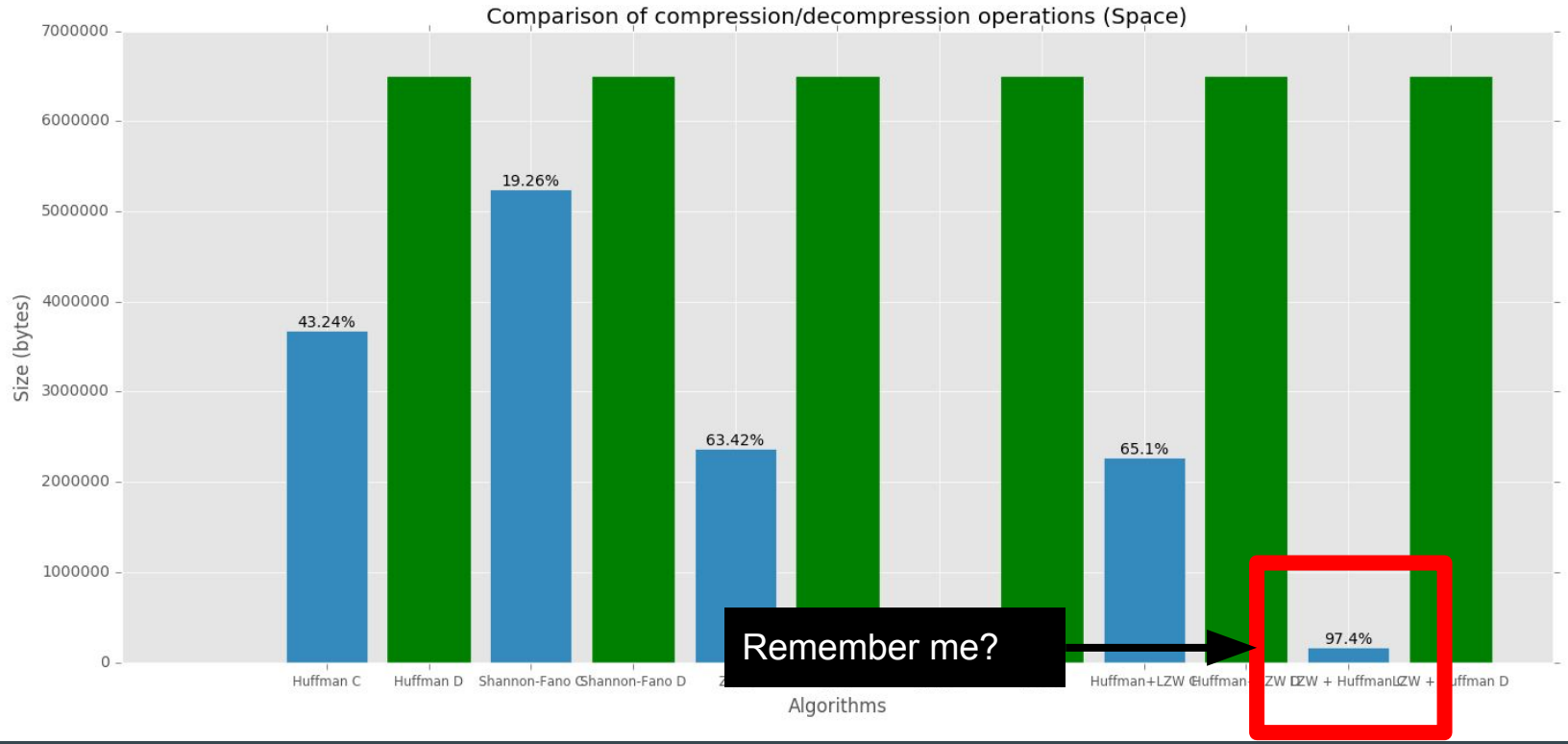
LZW

Input	Current String	Seen this Before?	Encoded Output	New Dictionary Entry/Index
b	b	yes	nothing	none
ba	ba	no	1	ba / 5
ban	an	no	1,0	an / 6
bana	na	no	1,0,3	na / 7
banan	an	yes	no change	none
banana	ana	no	1,0,3,6	ana / 8
banana_	a_	no	1,0,3,6,0	a_ / 9
banana_b	_b	no	1,0,3,6,0,4	_b / 10
banana_ba	ba	yes	no change	none
banana_ban	ban	no	1,0,3,6,0,4,5	ban / 11
banana_band	nd	no	1,0,3,6,0,4,5,3	nd / 12
banana_banda	da	no	1,0,3,6,0,4,5,3,2	da / 13
banana_bandan	an	yes	no change	none
banana_bandana	ana	yes	1,0,3,6,0,4,5,3,2,8	none

Remember?


Comparison of compression/decompression operations (Time)
Time (microseconds)
1000000
800000
600000
400000
200000
0
Huffman C
Huffman D
Shannon-Fano C
Shannon-Fano D
ZIP C
ZIP D
LZW C
LZW D
Huffman+LZW C
LZW + Huffman
LZW + Huffman D
Algorithms

Remember?


Comparison of compression/decompression operations (Space)
Size (bytes)
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
43.24%
19.26%
63.42%
65.1%
97.4%
Huffman C
Huffman D
Shannon-Fano C
Shannon-Fano D
Huffman+LZW C
Huffman
ZW D
ZW + Huffman
ZW +
ffman D
Algorithms
Remember me?

LZW Over Huffman

- Compression
  - ‘[['mosaics/Pigeon-8_xmap_34_2.jpg', 'mosaics/Pigeon-8_xmap_13_8.jpg', ...,'mosaics/Pigeon-8_xmap_11_8.jpg','mosaics/Pigeon-8_xmap_5_22.jpg'], [0, 1, 2, 2, 3, 3, 4, 4, 2, 2, 4, ...,4, 2, 2, 4, 3, 5, 6, 1, 7, 1, 1, 1, 8, 0, 1, 3, 3, 6, 0, 0, 7]]’
  - Huffman encode -> 1000110011001010101011111011101101000 01
  - LZW encode -> [111,132,142,266,...]
- Decompression
  - LZW decode -> 1000110011001010101011111011101101000 01
  - Huffman decode -> ‘[['mosaics/Pigeon-8_xmap_34_2.jpg', 'mosaics/Pigeon-8_xmap_13_8.jpg', ..., 'mosaics/Pigeon-8_xmap_11_8.jpg','mosaics/Pigeon-8_xmap_5_22.jpg'], [0, 1, 2, 2, 3, 3, 4, 4, 2, 2, 4, ..., 4, 2, 2, 4, 3, 5, 6, 1, 7, 1, 1, 1, 8, 0, 1, 3, 3, 6, 0, 0, 7]]’

Seems to be the same, but surely isn’t!

Original Image

Properties

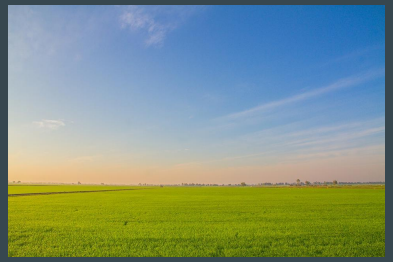
Name: Pigeon-8.jpg

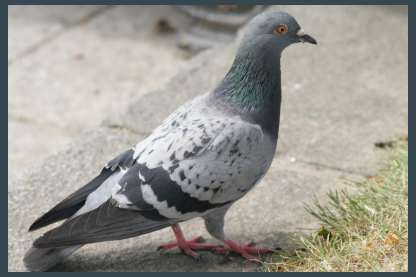
Size: 194 Kb

Dimensions: 1154*768 pixels

1st Approach

1. Take a collection of images
2. Choose the n prototypes that are most similar with each mosaic of the original image (matrixEval)
3. Build a new image from the prototype selection





1st Approach

2nd Approach

1. One image
2. Choose the n prototypes that are most similar with each mosaic of the original image
3. Build a new image from the prototype selection

2nd Approach

2nd Approach

Final approach

1. One Image
2. Choose the n prototypes at random
3. **Train them**
4. Build a new image from the **trained** prototype selection

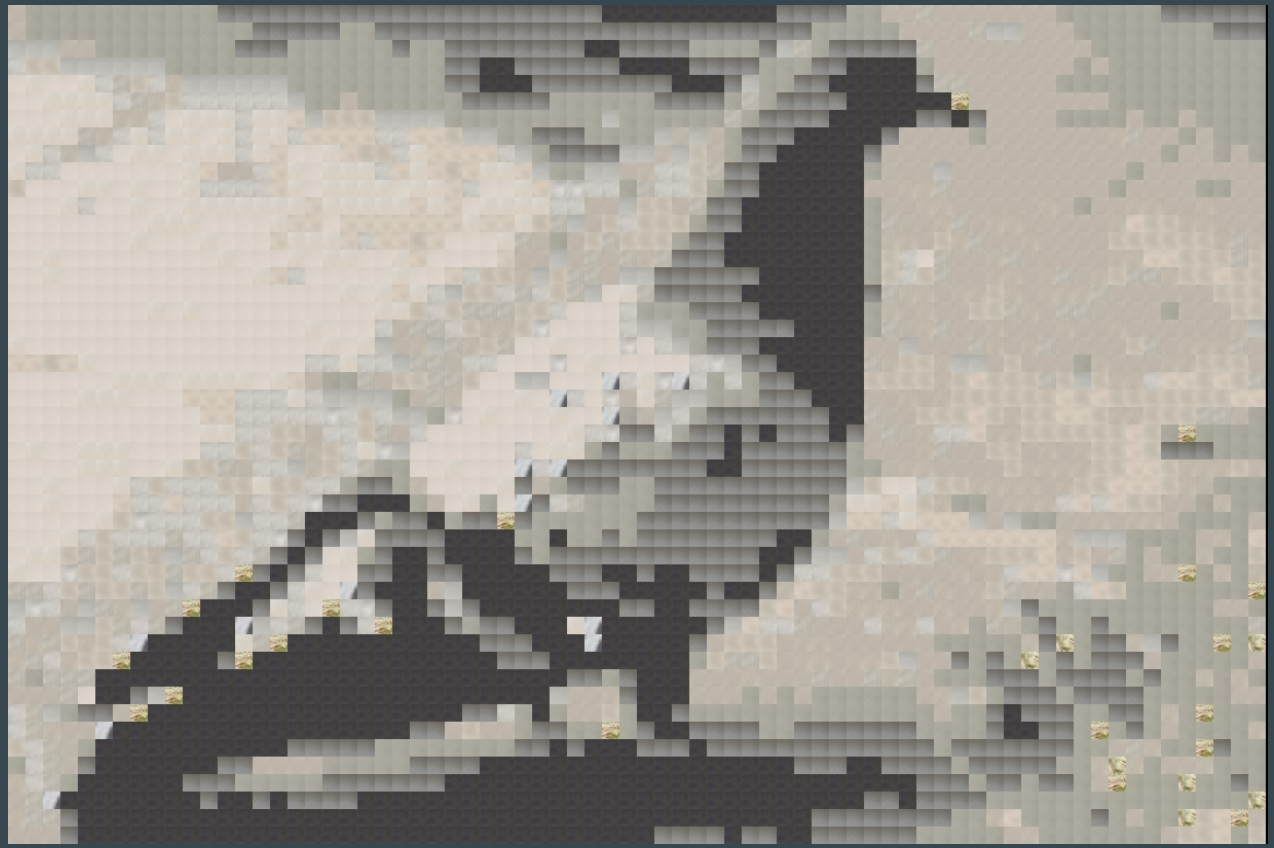
Mosaic Size: 32 Number of Prototypes: 16

Mosaic Size: 32 Number of Prototypes: 32

Mosaic Size: 32 Number of Prototypes: 64

Mosaic Size: 32 Number of Prototypes: 128

Mosaic Size: 16 Number of Prototypes: 16



Mosaic Size: 16 Number of Prototypes: 32

Mosaic Size: 16 Number of Prototypes: 64

Mosaic Size: 16 Number of Prototypes: 128

Mosaic Size: 8 Number of Prototypes: 16

Mosaic Size: 8 Number of Prototypes: 32

Mosaic Size: 8 Number of Prototypes: 64

Mosaic Size: 8 Number of Prototypes: 128

Mosaic Size: 4 Number of Prototypes: 16

Mosaic Size: 4 Number of Prototypes: 32

Mosaic Size: 4 Number of Prototypes: 64

Mosaic Size: 4 Number of Prototypes: 128

Learning Algorithm

y -> most similar prototype

x -> mosaic

α -> alpha [0,1]

i,j -> mosaic location

$$y^n = y^n + \alpha (x_{ij} - y^n)$$

Comparison Algorithm

y -> most similar prototype

x -> mosaic

α -> alpha [0,1]

p -> number of prototypes

$$d(y^n, x) \leq d(y^m, x) \; \forall m$$

$$d(y, x) = \sqrt{\sum_{i=1}^{p} (y^i - x^i)^2}$$

Example of Lossy Compression

Original Lena Image (12KB size)

Lena Image, Compressed (85% less information, 1.8KB)

Lena Image, Highly Compressed (96% less information, 0.56KB)

Original Lena image

Properties

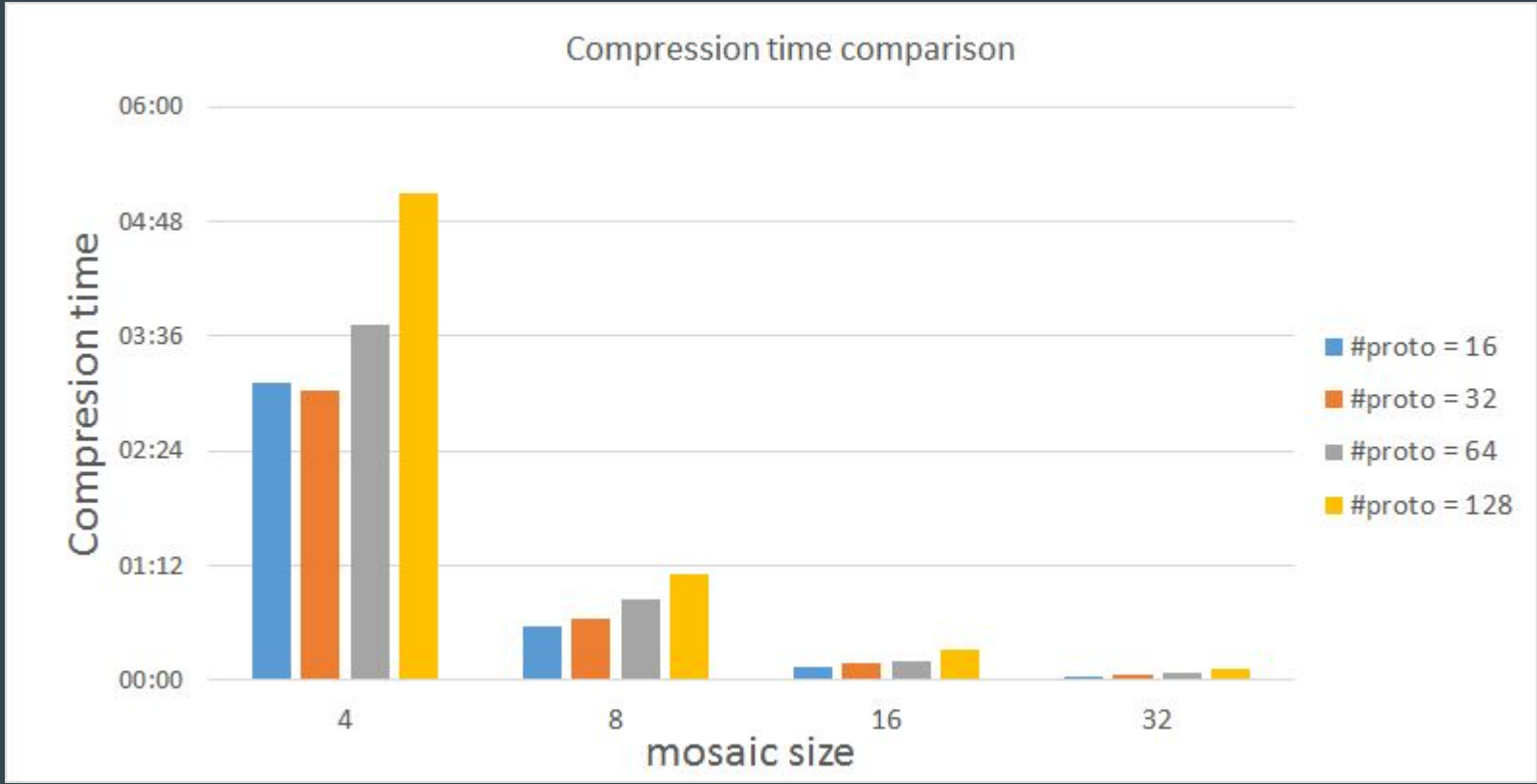
Name: lena.jpg

Size: 335 Kb

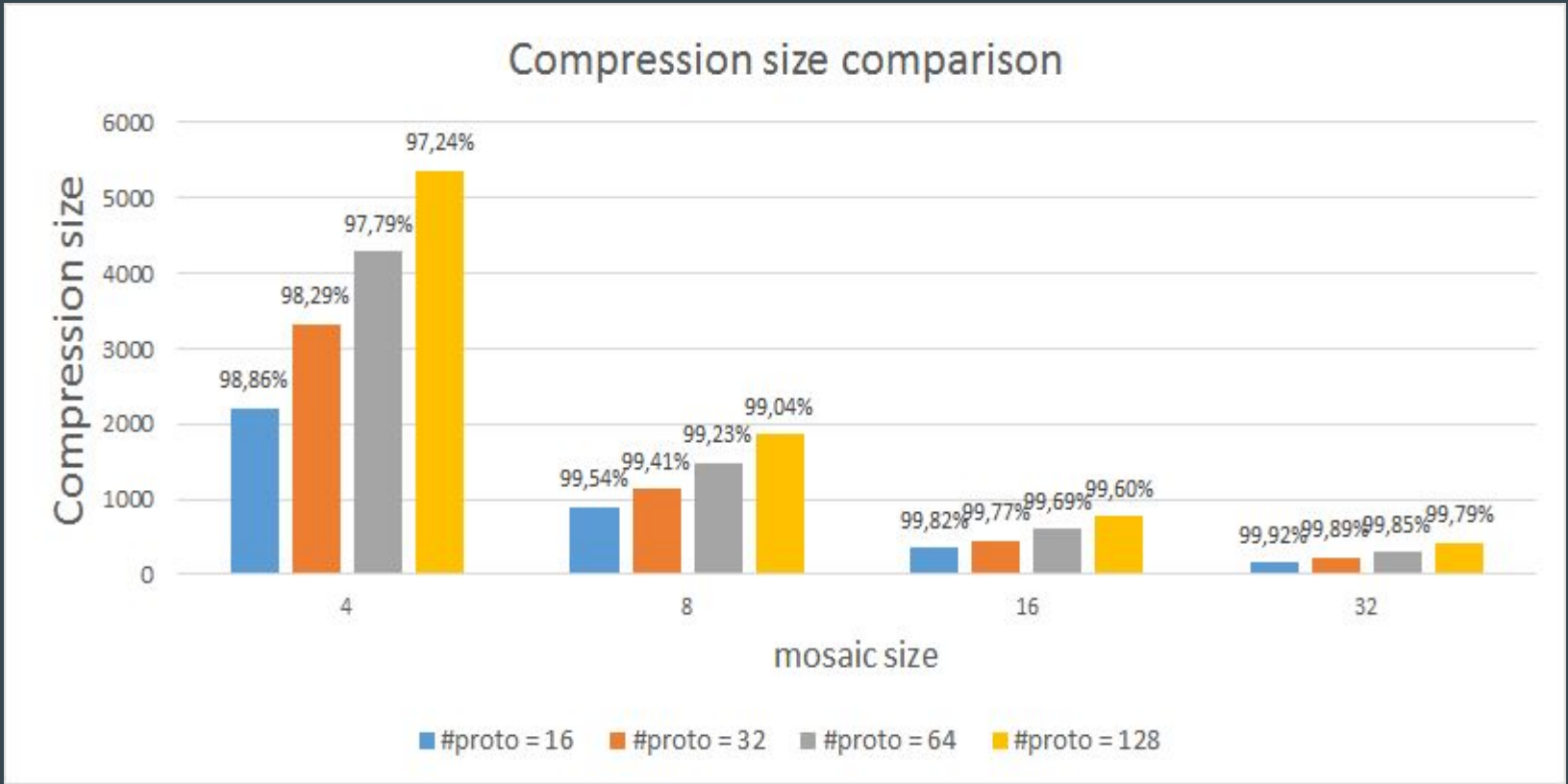
Dimensions: 512*512 pixels

Our version of Lena

Results


Compression time comparison
06:00
04:48
03:36
02:24
01:12
00:00
Compresion time
4
8
16
32
mosaic size
#proto = 16
#proto = 32
#proto = 64
#proto = 128

Results


Compression size comparison
Compression size
6000
5000
4000
3000
2000
1000
0
97,24%
97,79%
98,29%
98,86%
99,04%
99,23%
99,41%
99,54%
99,60%
99,69%
99,77%
99,82%
99,79%
99,85%
99,89%
99,92%
4
8
16
32
mosaic size
#proto = 16
#proto = 32
#proto = 64
#proto = 128

References

- Steven S. Skiena. 2008. The Algorithm Design Manual (2nd ed.). Springer Publishing Company, Incorporated.
- Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. Introduction to Algorithms (2nd ed.). McGraw-Hill Higher Education.
- Davis, Z., 2013. Huffman | An Algorithmic Lucidity. May, 2016 in: zackmdavis.net/blog/2013/06/huffman/
- Md. Rubaiyat Hasan, 2011. Data Compression using Huffman based LZW Encoding Technique. June, 2016 in: www.ijser.org/researchpaper%5CData-Compression-using-Huffman-based-LZW-Encoding-Technique.pdf

References

- Anon, 2016. Huffman coding - Rosetta Code. May, 2016 in: rosettacode.org/wiki/Huffman_coding
- Mark Nelson,2016. LZW Data Compression. June, 2016 in: marknelson.us/1989/10/01/lzw-data-compression/
- Anon, 2016. LZW compression - Rosetta Code. June, 2016 in: rosettacode.org/wiki/LZW_compression#Python
- Jia Li. Prototype Methods: K-Means. June, 2016 in: http://sites.stat.psu.edu/~jiali/course/stat597e/notes2/kmeans.pdf
- Scott Beale, 2009. Generation Loss, What JPEG Compression Looks Like After 600 Saves. June, 2016 in: http://laughingsquid.com/generation-loss-what-jpg-compression-looks-like-after-600-saves/
- https://en.wikipedia.org/wiki/RGB_color_model
- https://en.wikipedia.org/wiki/K-means_clustering

Questions?