IMAGE LOSSY COMPRESSION/DECOMPRESSION METHOD USING DCT.

Student: Nguyễn Trường Giang – 1551034.
Instructor: Associate Prof. PhD Đỗ Hồng Tuấn.

Introduction

- As our use of and reliance on computers continues to grow, so does the need for efficient ways of storing large amounts of data.
- Two kind of categories: lossless and lossy image compression.
- JPEG is an lossy image compression using Discrete Cosine Transform to separate into parts of different frequencies. Then apply a process called quantization, where the less important information are discarded → reconstructed image contain distortion but size of image are reduced.

Overview the process

1. The image broken into 8x8 blocks of pixels.
2. Working from left to right, top to bottom, the DCT is applied to each block.
3. Each block is compressed through quantization.
4. The array of compressed blocks that constitute the image is stored/ transmitted in a less amount of size.
5. When desired, the image is reconstructed through decompression, using Inverse Discrete Cosine Transform (IDCT).

DCT Equation

To compute the i, j th entry of the DCT of an image N x N [2]:

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i) C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left(\frac{(2x+1)i\pi}{2N}\right) \cos\left(\frac{(2y+1)j\pi}{2N}\right)$$

Where:

$$C(u) = \frac{1}{\sqrt{2}}\ if\ u = 0\ and\ C(u) = 1\ if\ u > 0$$

DCT Matrix

To get the matrix form from above, we will use the following equation [1]:

$$T_{i,j} = \left\{ \begin{array}{ll} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)i\pi}{2N} \right] & \text{if } i > 0 \end{array} \right\}$$

DCT Matrix

For an 8x8 block it results in this matrix:

$$T = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix}$$

Doing the DCT on an 8x8 Block

Since an image comprises hundred or even thousands of 8x8 blocks of pixels, the following description of what happen to one 8x8 block in JPEG process.

After what is done with one block we continues to do with the rest of an image.

Doing DCT on an 8x8 Block

$$Original = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix}$$

$$M = \begin{bmatrix} 26 & -5 & -5 & -5 & -5 & -5 & -5 & 8 \\ 64 & 52 & 8 & 26 & 26 & 26 & 8 & -18 \\ 126 & 70 & 26 & 26 & 52 & 26 & -5 & -5 \\ 111 & 52 & 8 & 52 & 52 & 38 & -5 & -5 \\ 52 & 26 & 8 & 39 & 38 & 21 & 8 & 8 \\ 0 & 8 & -5 & 8 & 26 & 52 & 70 & 26 \\ -5 & -23 & -18 & 21 & 8 & 8 & 52 & 38 \\ -18 & 8 & -5 & -5 & -5 & 8 & 26 & 8 \end{bmatrix}$$

$$D = TMT'$$

Doing DCT on an 8x8 Block

$$D = \begin{bmatrix} 162.3 & 40.6 & 20.0 & 72.3 & 30.3 & 12.5 & -19.7 & -11.5 \\ 30.5 & 108.4 & 10.5 & 32.3 & 27.7 & -15.5 & 18.4 & -2.0 \\ -94.1 & -60.1 & 12.3 & -43.4 & -31.3 & 6.1 & -3.3 & 7.1 \\ -38.6 & -83.4 & -5.4 & -22.2 & -13.5 & 15.5 & -1.3 & 3.5 \\ -31.3 & 17.9 & -5.5 & -12.4 & 14.3 & -6.0 & 11.5 & -6.0 \\ -0.9 & -11.8 & 12.8 & 0.2 & 28.1 & 12.6 & 8.4 & 2.9 \\ 4.6 & -2.4 & 12.2 & 6.6 & -18.7 & -12.8 & 7.7 & 12.0 \\ -10.0 & 11.2 & 7.8 & -16.3 & 21.5 & 0.0 & 5.9 & 10.7 \end{bmatrix}$$

Quantization

- Now our 8x8 block of DCT is ready for compression by quantization.
- With the level of n, the 8x8 quantization table is calculated by :

$$Q(n) = Q50 * \frac{100-n}{50} \; with\, n \geq 50$$

$$Q(n) = Q50 \; * \frac{50}{n} \; with\, n < 50$$

Q50 is basic matrix defined in [1].

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantization

$$Q_{10} = \begin{bmatrix} 80 & 60 & 50 & 80 & 120 & 200 & 255 & 255 \\ 55 & 60 & 70 & 95 & 130 & 255 & 255 & 255 \\ 70 & 65 & 80 & 120 & 200 & 255 & 255 & 255 \\ 70 & 85 & 110 & 145 & 255 & 255 & 255 & 255 \\ 90 & 110 & 185 & 255 & 255 & 255 & 255 & 255 \\ 120 & 175 & 255 & 255 & 255 & 255 & 255 & 255 \\ 245 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix}$$

$$Q_{90} = \begin{bmatrix} 3 & 2 & 2 & 3 & 5 & 8 & 10 & 12 \\ 2 & 2 & 3 & 4 & 5 & 12 & 12 & 11 \\ 3 & 3 & 3 & 5 & 8 & 11 & 14 & 11 \\ 3 & 3 & 4 & 6 & 10 & 17 & 16 & 12 \\ 4 & 4 & 7 & 11 & 14 & 22 & 21 & 15 \\ 5 & 7 & 11 & 13 & 16 & 12 & 23 & 18 \\ 10 & 13 & 16 & 17 & 21 & 24 & 24 & 21 \\ 14 & 18 & 19 & 20 & 22 & 20 & 20 & 20 \end{bmatrix}$$

Quantization

$$C_{i,j} = round\left(\frac{D_{i,j}}{Q_{i,j}}\right)$$

$$C = \begin{bmatrix} 10 & 4 & 2 & 5 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -5 & 1 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & 0 & -1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Decompression

- Reconstruction of our image begin by multiply the matrix C to the quantization matrix originally used.

$$R_{i,j} = Q_{i,j}\ x\ C_{i,j}$$

$$R = \begin{bmatrix} 160 & 44 & 20 & 80 & 24 & 0 & 0 & 0 \\ 36 & 108 & 14 & 38 & 26 & 0 & 0 & 0 \\ -98 & -65 & 16 & -48 & -40 & 0 & 0 & 0 \\ -42 & -85 & 0 & -29 & 0 & 0 & 0 & 0 \\ -36 & 22 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Inverse DCT

- The IDCT is next applied to matrix R, which is rounded to the nearest integer. Finally, 128 is added to each element of that result, giving us the decompressed JPEG version N of our original 8x8 image block M

$$N = round(T' \; R \; T) + 128$$

[1]

Comparison of Matrices

$$Original = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix}$$

$$Decompressed = \begin{bmatrix} 149 & 134 & 119 & 116 & 121 & 126 & 127 & 128 \\ 204 & 168 & 140 & 144 & 155 & 150 & 135 & 125 \\ 253 & 195 & 155 & 166 & 183 & 165 & 131 & 111 \\ 245 & 185 & 148 & 166 & 184 & 160 & 124 & 107 \\ 188 & 149 & 132 & 155 & 172 & 159 & 141 & 136 \\ 132 & 123 & 125 & 143 & 160 & 166 & 168 & 171 \\ 109 & 119 & 126 & 128 & 139 & 158 & 168 & 166 \\ 111 & 127 & 127 & 114 & 118 & 141 & 147 & 135 \end{bmatrix}$$

Evaluation

- **Objective fidelity criteria**

Let $f(x,y)$ be an input image and $\hat{f}(x,y)$ be an approximation of $f(x,y)$. The images are of size $M \times N$.

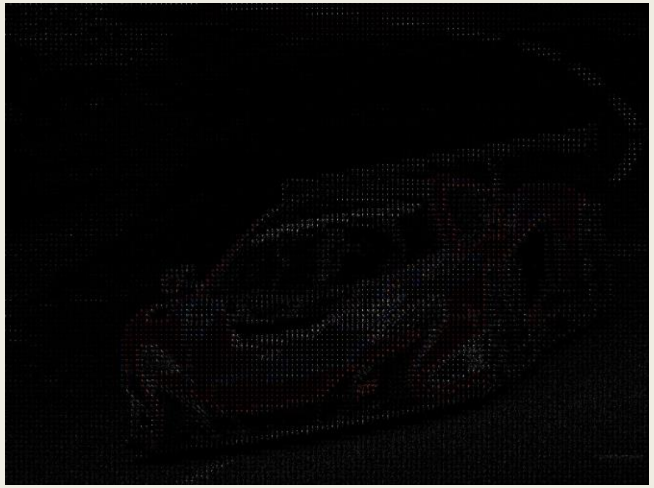
The *root-mean-square error* is

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2 \right]^{1/2}$$

The *mean-square signal-to-noise ratio* of the output image, denoted $\mathrm{SNR}_{\mathrm{ms}}$

$$\mathrm{SNR}_{\mathrm{ms}} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) \right]^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2}$$

Example:



```
Filename:  car2.jpg
Level of compression:  10
Compressing....
Image size:  (768, 1024, 3)
Compression Time:  10.2  sec
Decompressing...
Decompression Time:  2.4  sec
Total:  12.7  sec
RMS:  0.0122
SNR:  2.2446
```

Example:

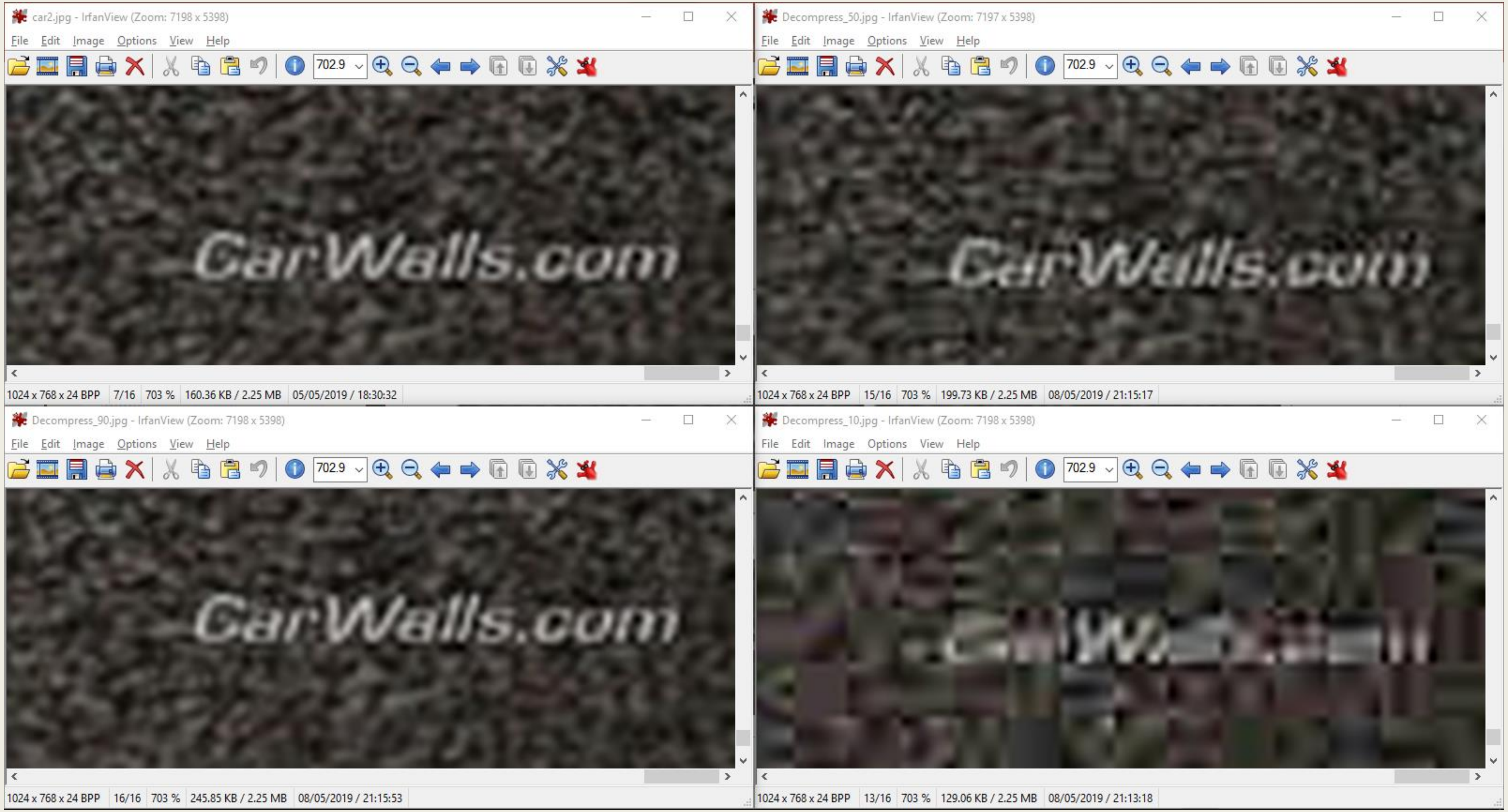
Original file

```
Filename:  car2.jpg
Level of compression:  50
Compressing....
Image size:  (768, 1024, 3)
Compression Time:  11.0  sec
Decompressing...
Decompression Time:  2.3  sec
Total:  13.3  sec
RMS:  0.0084
SNR:  5.0847
```

```
Filename:  car2.jpg
Level of compression:  90
Compressing....
Image size:  (768, 1024, 3)
Compression Time:  11.6  sec
Decompressing...
Decompression Time:  2.4  sec
Total:  14.0  sec
RMS:  0.0039
SNR:  24.2914
```

```
Filename:  car2.jpg
Level of compression:  10
Compressing....
Image size:  (768, 1024, 3)
Compression Time:  10.2  sec
Decompressing...
Decompression Time:  2.3  sec
Total:  12.5  sec
RMS:  0.0122
SNR:  2.2446
```

Example


car2.jpg - IrfanView (Zoom: 7198 x 5398)
File Edit Image Options View Help
702.9
CarWalls.com
1024 x 768 x 24 BPP 7/16 703 % 160.36 KB / 2.25 MB 05/05/2019 / 18:30:32
Decompress_50.jpg - IrfanView (Zoom: 7197 x 5398)
File Edit Image Options View Help
702.9
CarWalls.com
1024 x 768 x 24 BPP 15/16 703 % 199.73 KB / 2.25 MB 08/05/2019 / 21:15:17
Decompress_90.jpg - IrfanView (Zoom: 7198 x 5398)
File Edit Image Options View Help
702.9
CarWalls.com
1024 x 768 x 24 BPP 16/16 703 % 245.85 KB / 2.25 MB 08/05/2019 / 21:15:53
Decompress_10.jpg - IrfanView (Zoom: 7198 x 5398)
File Edit Image Options View Help
702.9
1024 x 768 x 24 BPP 13/16 703 % 129.06 KB / 2.25 MB 08/05/2019 / 21:13:18

Comparison table

<u>Device:</u> Dell Latitude E6430, Intel Core i7 – 3520M, 8GB RAM, Window 10 64bit.
<u>Software:</u> Anaconda 4.6.14 Jupyter Notebook, Python 3.6.8

	Size	Level of Compression	Time (sec)	RMS	SNR
	(256, 256, 3)	30	1.133	0.0376	3.5249
		50	1.201	0.0343	4.1059
		90	1.259	0.0177	16.4038
	(768, 1024, 3)	30	13.187	0.0095	4.0506
		50	13.789	0.0084	5.0847
		90	13.989	0.0039	24.2914

Conclusion

- Different level of compression give almost the same image for human eyes, but for computer vision, the original and the “after process” image is very different.
- As you increase the size of image, the process will take longer to compute.
- Bigger level of compression will take longer to process, RMS will decrease, SNR increase

→ the quality is almost the same with the original.

References

[1] Image Compression and the Discrete Cosine Transform

by Ken Cabeen and Peter Gent – College of the Redwoods
https://www.math.cuhk.edu.hk/~lmlui/dct.pdf

[2] Lecture Note in Digital Image Processing 2018 at HCMUT

by Associate Prof. PhD Đỗ Hồng Tuấn.

Thank you