Image Compression using SVD and DCT

Math 2270-003
Spring 2012
Yizhou Ye

Image? Matrix?

Matrix?

	1	2	3	4	5	6	7	8	9
40	10	7	10	3	2	2	3	7	
41	10	5	7	3	2	2	3	7	
42	7	5	5	7	2	2	2	4	
43	10	6	5	5	3	24	2	3	
44	7	8	5	5	3	24	2	2	
45	5	5	5	5	3	2	24	2	
46	5	5	5	5	5	3	2	2	
47	5	5	7	10	5	10	2	2	
48	6	7	7	10	5	7	3	24	
49	5	6	7	10	5	5	3	10	
50	5	6	10	10	10	5	10	2	
51	5	7	10	10	10	5	5	3	
52	5	10	10	23	10	5	5	7	
53	12	7	10	10	10	10	5	7	
54	12	10	10	23	10	5	5	7	
55	5	7	10	10	10	7	5	5	
56	12	10	10	7	10	3	10	10	
57	12	10	10	10	23	10	10	10	
58	12	10	10	10	10	10	10	10	

Notes

- Image File = Header + RGB / GrayScale
- Maple / Matlab what do they do?

Matlab API

- A = imread(filename, *fmt*) reads a grayscale or color image from the file specified by the string filename.

- The return value A is an array containing the image data. If the file contains a grayscale image, A is an M-by-N array. If the file contains a true-color image, A is an M-by-N-by-3 array.

.jpeg, .jpg

- Image == matrix? No.
- Approximate way

JPEG – Joint Photographic Experts Group

imread can read any baseline JPEG image as well as JPEG images with some commonly used extensions. For information about support for JPEG 2000 files, see JPEG 2000.

Supported Bitdepths (Bits-per-sample)	Lossy Compression	Lossless Compression	Output Class	Notes
8-bit	y	y	uint8	Grayscale or RGB
12-bit	y	y	uint16	Grayscale or RGB
16-bit	-	y	uint16	Grayscale

Basically

- Read Image
- Matrix
- SVD / DCT
- done/ compressed

SVD

- SVD: singular value decomposition

Using the SVD we can write an $n \times n$ invertible matrix A as:

$$
\begin{aligned}
A &= P\Sigma Q^T = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_\mathbf{n}) \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \sigma_n \end{pmatrix} \begin{pmatrix} \mathbf{q}_1{}^T \\ \mathbf{q}_2{}^T \\ \vdots \\ \mathbf{q}_\mathbf{n}{}^T \end{pmatrix} \\
&= \mathbf{p}_1\sigma_1\mathbf{q}_1{}^T + \mathbf{p}_2\sigma_2\mathbf{q}_2{}^T + \cdots + \mathbf{p}_\mathbf{n}\sigma_n\mathbf{q}_\mathbf{n}{}^T
\end{aligned}
$$

SVD

- Note that A is m*n, U is m*m orthogonal matrix, Σ is an m*n matrix containing singular values of A, and V is an r*r orthogonal matrix. And the singular values of A are:

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_r \geq 0$$

- All these singular values are along the main diagonal of Σ.
- We can rewrite the formula in the following way:

SVD

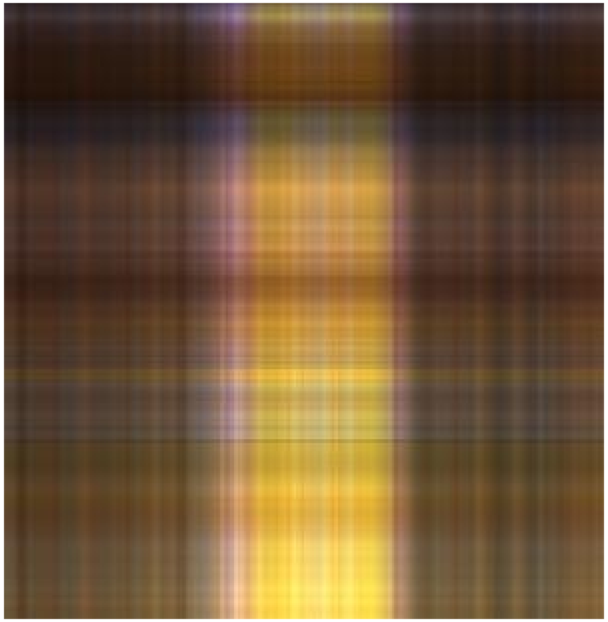
Using the SVD we can write an $n \times n$ invertible matrix A as:

$$
\begin{aligned}
A &= P\Sigma Q^T = (\mathbf{p_1}, \mathbf{p_2}, \dots, \mathbf{p_n}) \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \sigma_n \end{pmatrix} \begin{pmatrix} \mathbf{q_1}^T \\ \mathbf{q_2}^T \\ \vdots \\ \mathbf{q_n}^T \end{pmatrix} \\
&= \mathbf{p_1}\sigma_1\mathbf{q_1}^T + \mathbf{p_2}\sigma_2\mathbf{q_2}^T + \cdots + \mathbf{p_n}\sigma_n\mathbf{q_n}^T
\end{aligned}
$$

Approximation

- Approximation of SVD is the most crucial part:
- $$A = U\Sigma V^T$$
  $$A = \sum \sigma_i \vec{u_i}\, \vec{v_i}^T$$
  $$A = \sum \sigma_1 \vec{u_1}\, \vec{v_1}^T + \sigma_2 \vec{u_2}\, \vec{v_2}^T + \cdots$$
- We know that the terms {Ai} are ordered from greatest to lowest, thus we can approximate A by varying the number of items. In other words, we can change the rank of A to make the approximation (of course, larger number gives us a more accurate approximation).

Example:

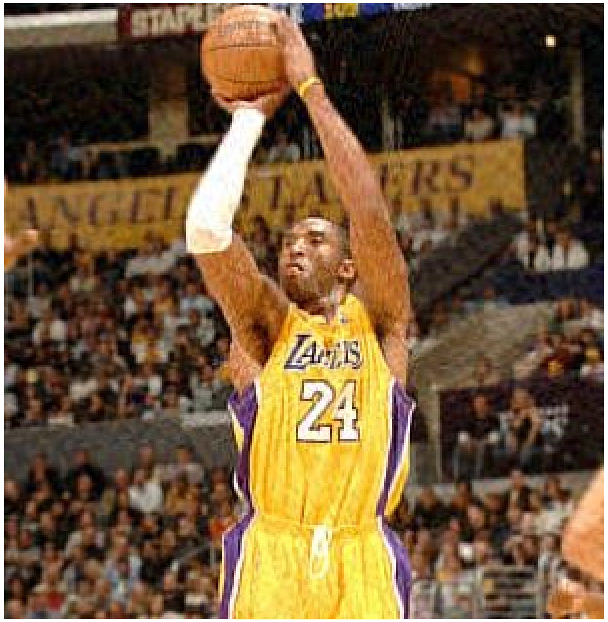


One term

Three terms

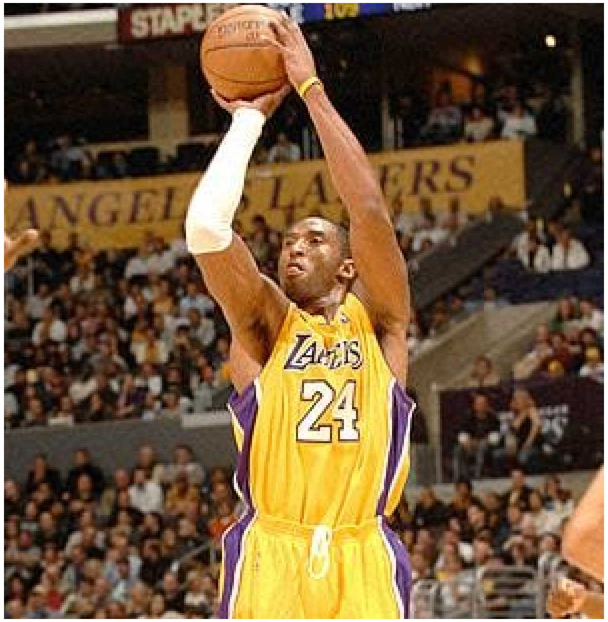
Examples:

10 terms

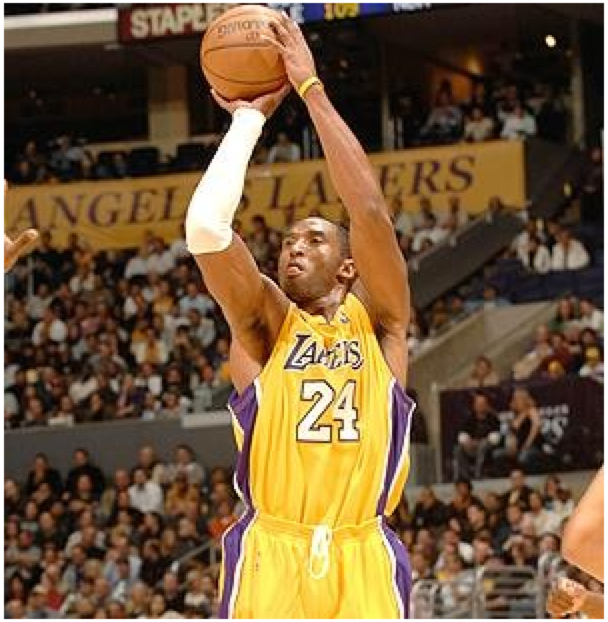


50 terms

Examples:

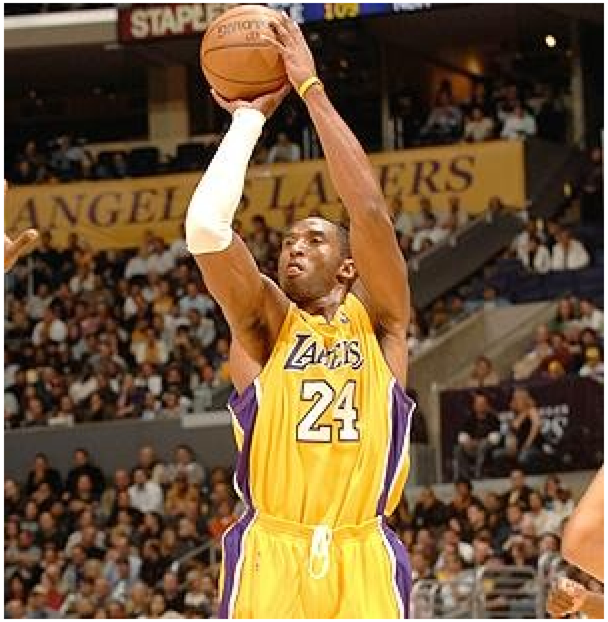


100 terms

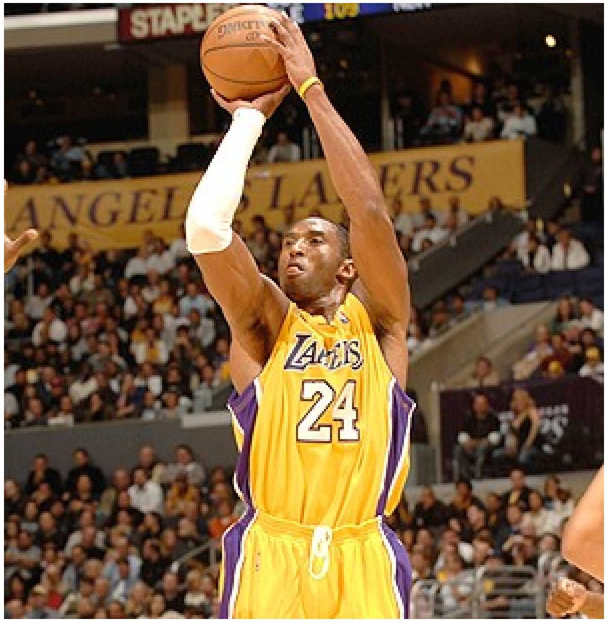


300 terms

Examples:



300 terms (rank)

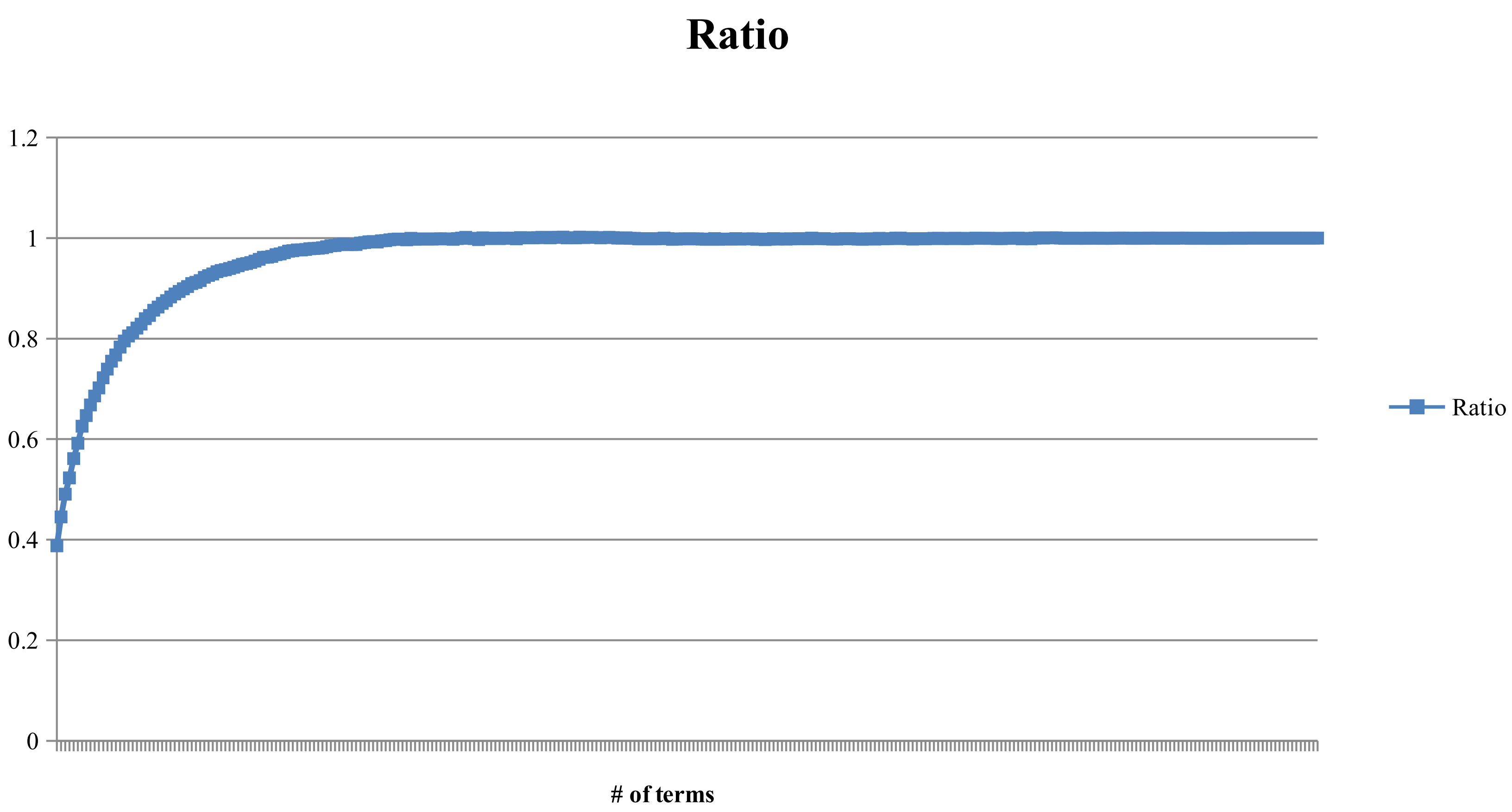


Original image

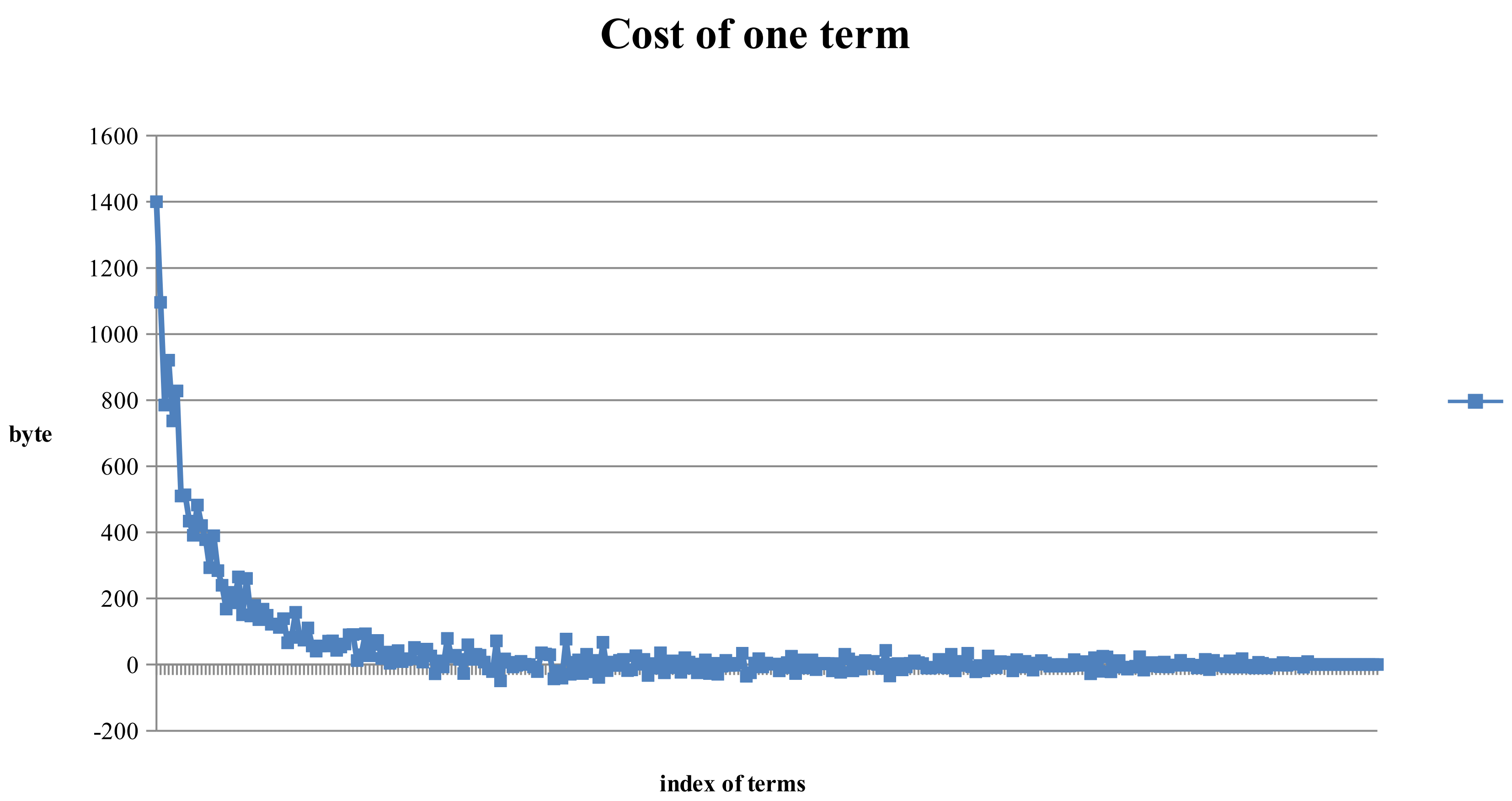
Issues

- Compression Ratio:
  - Not exactly (1+m+n) / (m*n) for a m*n A
  - This plot is draw by matlab: Image is more complex than we thought
- MatLab Read original size: 24206
  - Just the RGB / GrayScaler

Ratio



Cost of items



DCT

- “Discrete Cosine Transformation”, which works by separate image into parts of different frequencies.

- A “lossy” compression, because during a step called “quantization”, where parts of compression occur, the less important frequencies will be discarded. Later in the “recombine parts” step, which is known as decompression step, some little distortion will occur, but it will be somehow adjusted in further steps.

DCT Equations

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

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases} \qquad 2$$

i, j are indices of the ij-th entry of the image matrix, p(x, y) is the matrix element in that entry, and N is the size of the block we are working on.

8 * 8 blocks

- For a standard procedure where N=8, the equation can be also written as the following form:

$$D(i,j) = \frac{1}{4}C(i)C(j)\sum_{x=0}^{7}\sum_{y=0}^{7}p(x,y)\cos\left[\frac{(2x+1)i\pi}{16}\right]\cos\left[\frac{(2y+1)j\pi}{16}\right] \qquad 3$$

T matrix

For an 8x8 block it results in this matrix:

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

Procedure

- Break the image matrix into 8*8 pixel blocks
- Applying DCT equations to each block in level order
- Each block is compressed through quantization
- Basically done.
- When desired, it can be decompressed, by Inverse Discrete Cosine Transformation.

Example of DCT

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

-128 for each entry

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

Since pixels are valued from -128 to 127

Apply D = TMT’

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

T matrix is from the previous equations.

Human eye fact

- The human eye is fairly good at seeing small differences in brightness over a relatively large area

- But not so good at distinguishing the exact strength of a high frequency (rapidly varying) brightness variation.

We know the fact, then

- This fact allows one to reduce the amount of information required by ignoring the high frequency components. This is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer.

- This is the main lossy operation in the whole process. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers.

Quantization Matrix

Quantization

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

quantization level = 50, a common choice of Q matrix

Round Equation

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

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

Typically, upper left corner. Thus we apply zig-zag order:

Zip-Zag Ordering

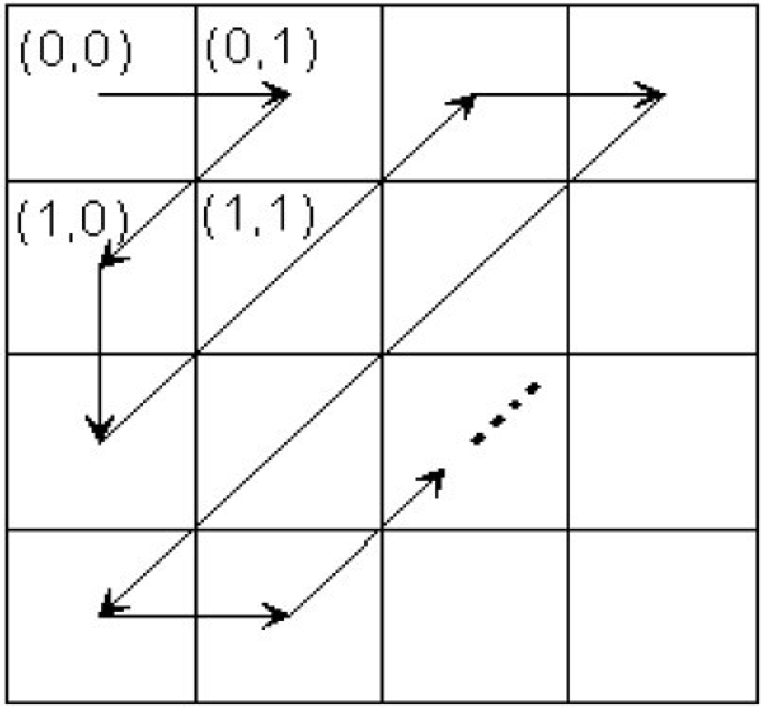


Figure 1

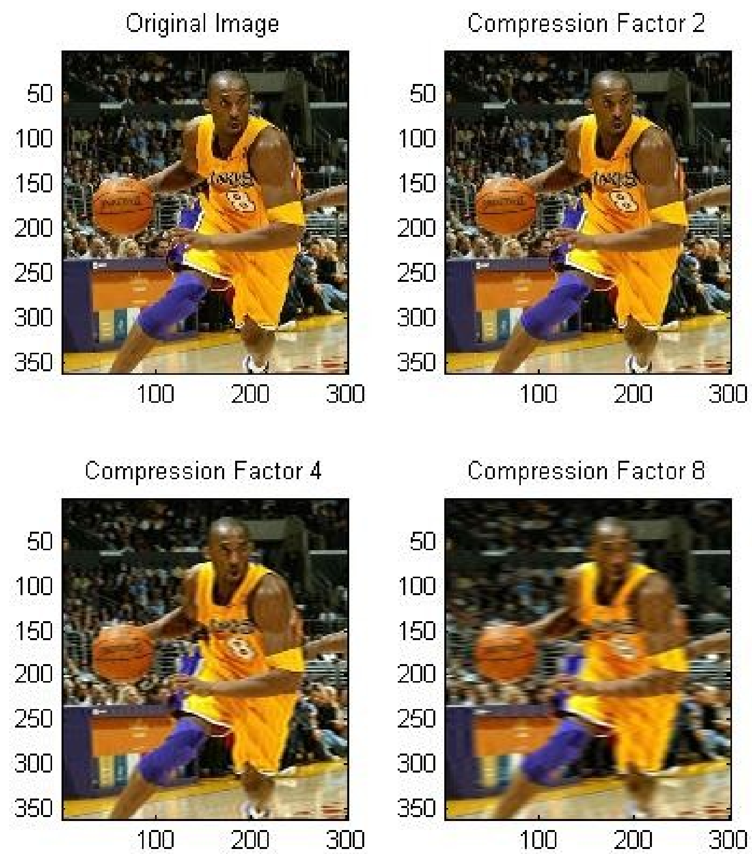
Original VS. Decompressed

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

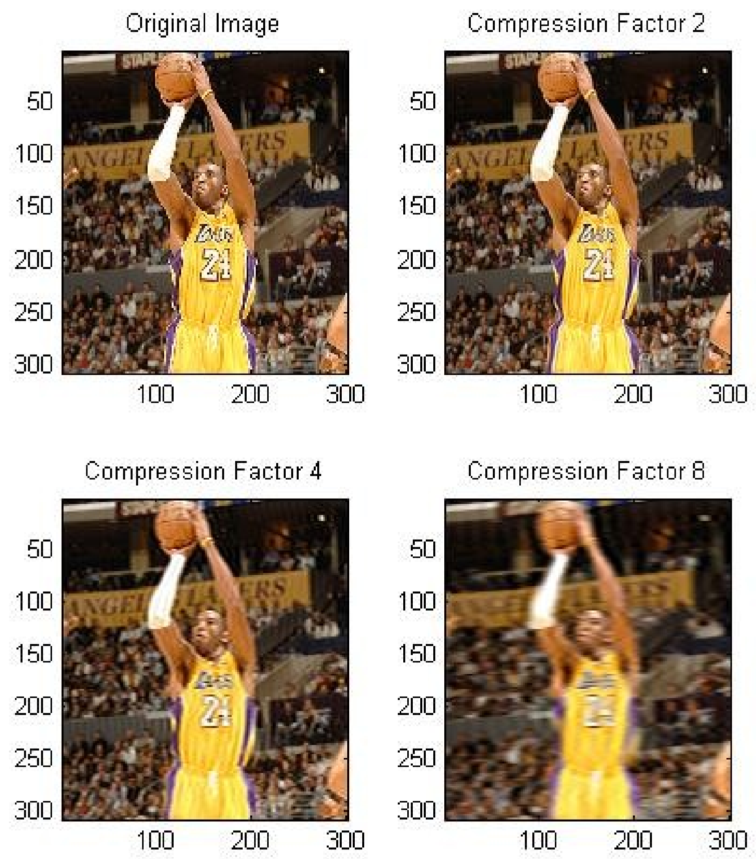
$$Original = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154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

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

Examples


Original Image
Compression Factor 2
Compression Factor 4
Compression Factor 8
50
100
150
200
250
300
350
100
200
300

More Examples



Finale

- Image can be expressed by matrix somehow, but image is much more than that.
- SVD and DCT are techniques to compress image, but both of them are "lossy".
- Still many other ways to compress:

Thank you!