Image Coding

Audiovisual Processing CMP-6026A

Dr. David Greenwood

david.greenwood@uea.ac.uk

SCI 2.16a University of East Anglia

December 01 2021

Content

Lossy and lossless image compression.

- Changing colour spaces and subsampling
- DCT and quantisation
- Run-length encoding
- Entropy coding

Image Coding

How can we compress an image without *destroying* the image?

- Data and information are not the same thing.
- Goal is to identify and remove **redundancy**.

Lossless

– Image can be reconstructed **exactly**.

Lossy

- Inflated image is an **approximation** of the original.
- How much loss is *acceptable*?

Image Redundancy

Inter-pixel redundancy:

- Neighbouring pixels are related to one another

Image Redundancy

Coding redundancy:

- Not all pixel intensities are equally likely

Image Redundancy

Pycho-visual redundancy:

– We are not visually *sensitive* to everything in the image

JPEG Compression

- A framework for compressing images.
- Many algorithms can be used in the framework.
- Developed by Joint Photographic Expert Group.
- JPEG exploits the three forms of redundancy outlined.

JPEG Compression

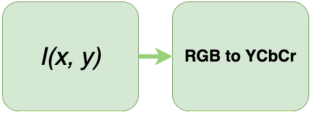


Figure 1: YC_bC_r

YC_bC_r

$$
\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B \\
C_b &= B - Y \\
C_r &= R - Y
\end{aligned}
$$

Luminance

$$Y = 0.299R + 0.587G + 0.114B$$

Humans are *more* sensitive to luminance. . .

Chrominance

$$C_b = B - Y$$
$$C_r = R - Y$$

Humans are *less* sensitive to chrominance...

YC_bC_r

We can downsample the chrominance channels without affecting the image in a *perceptible* way.

- Exploits **psycho-visual** redundancy.

JPEG Compression

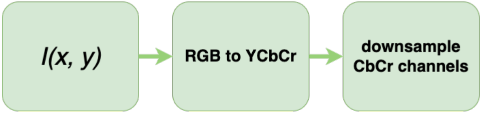


Figure 2: Chroma Subsampling

Chroma Subsampling

Subsampling scheme is expressed as a ratio **J:a:b**

- represents a conceptual window on the *chrominance* channels.

Chroma Subsampling

- **J**: horizontal sampling reference. Usually, 4.
- **a**: number of pixels in the top row that will have chroma information.
- **b**: number of *changes* of samples (Cr, Cb) between first and second row of J pixels.

Chroma Subsampling

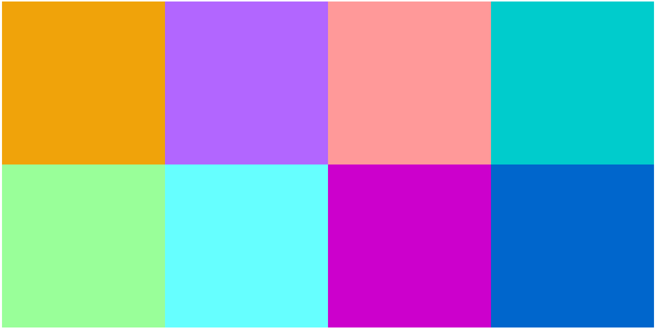


Figure 3: Chroma Subsampling

Chroma Subsampling

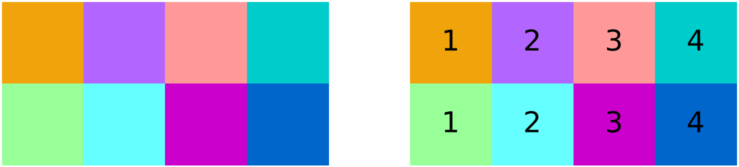


Figure 4: 4:4:4

Chroma Subsampling

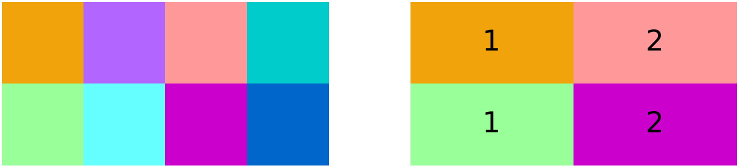


Figure 5: 4:2:2

Chroma Subsampling

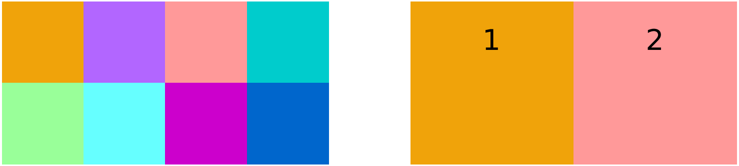


Figure 6: 4:2:0

JPEG Compression

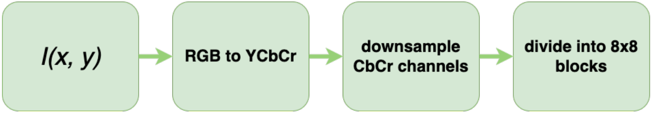


Figure 7: 8x8 Blocks

JPEG Compression

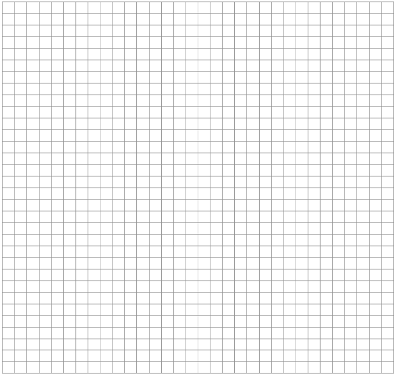


Figure 8: image matrix

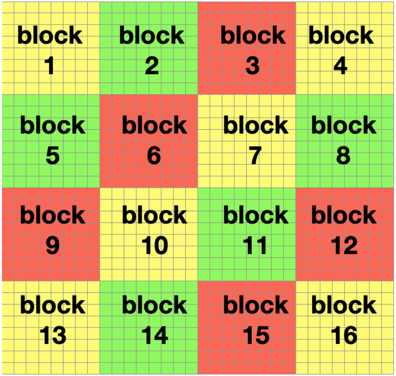


Figure 9: 8x8 blocks

JPEG Compression

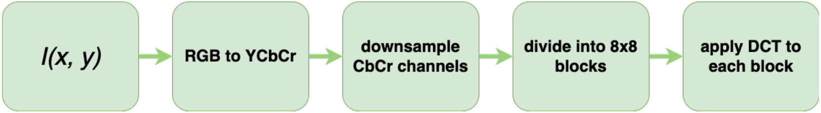
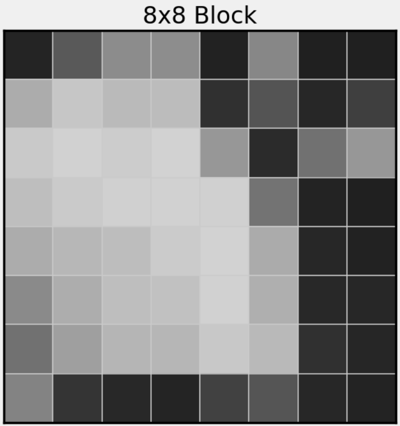


Figure 10: DCT

DCT

Transforms the image into the *frequency domain*.

DCT



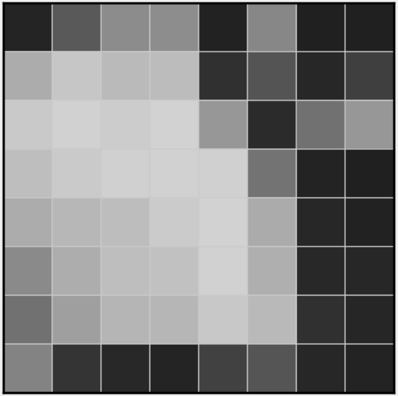
Intensities

36	90	140	141	34	135	33	32
172	198	186	188	48	84	39	63
201	209	204	210	151	43	114	151
190	202	208	209	208	115	35	33
172	183	189	203	210	171	39	34
138	173	190	193	209	175	40	39
114	159	181	182	200	185	49	38
131	53	40	37	66	85	39	35

Figure 11: image values

DCT

8x8 Block



DCT Coefficients

1041	-297	194	6	-10	34	-48	51
-20	-39	-45	121	19	-47	14	40
232	101	-75	-2	50	-41	-33	41
-5	23	129	-27	59	25	-48	44
124	29	6	-57	6	-5	-24	0
-41	22	71	10	9	19	2	-15
28	53	-42	13	-21	-9	15	-25
-36	39	-5	15	-5	-3	9	-6

Figure 12: coefficients

JPEG Compression

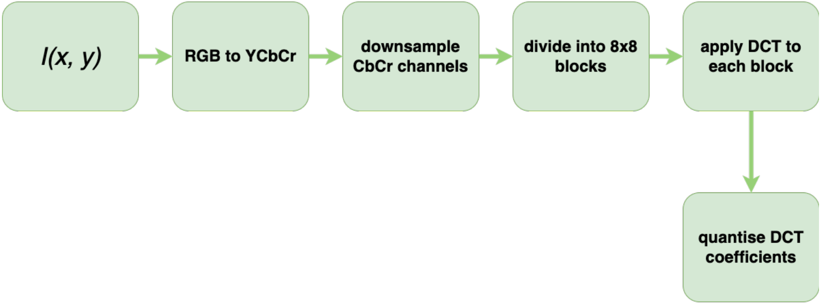


Figure 13: DCT Quantisation

DCT Quantisation

Reduce the number of bits needed to store a value by reducing precision.

- Decrease precision as we move away from the top left corner.
- High frequency details usually contribute less to the image.

DCT Quantisation

Quantisation is performed as follows:

$$DCT_q(i, j) = round\left(\frac{DCT(i, j)}{Q(i, j)}\right)$$

where Q is the quantisation matrix.

DCT Quantisation

Quantisation Matrix Q

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

Figure 14: quantisation matrix

DCT Quantisation

DCT Coefficients

1041	-297	194	6	-10	34	-48	51
-20	-39	-45	121	19	-47	14	40
232	101	-75	-2	50	-41	-33	41
-5	23	129	-27	59	25	-48	44
124	29	6	-57	6	-5	-24	0
-41	22	71	10	9	19	2	-15
28	53	-42	13	-21	-9	15	-25
-36	39	-5	15	-5	-3	9	-6

Q

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

Quantised DCT

65	-27	19	0	0	1	-1	1
-2	-3	-3	6	1	-1	0	1
17	8	-5	0	1	-1	0	1
0	1	6	-1	1	0	-1	1
7	1	0	-1	0	0	0	0
-2	1	1	0	0	0	0	0
1	1	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0

Figure 15: quantisation

DCT Quantisation

Quantised DCT

65	-27	19	0	0	1	-1	1
-2	-3	-3	6	1	-1	0	1
17	8	-5	0	1	-1	0	1
0	1	6	-1	1	0	-1	1
7	1	0	-1	0	0	0	0
-2	1	1	0	0	0	0	0
1	1	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0

Figure 16: DCT_q

JPEG Compression

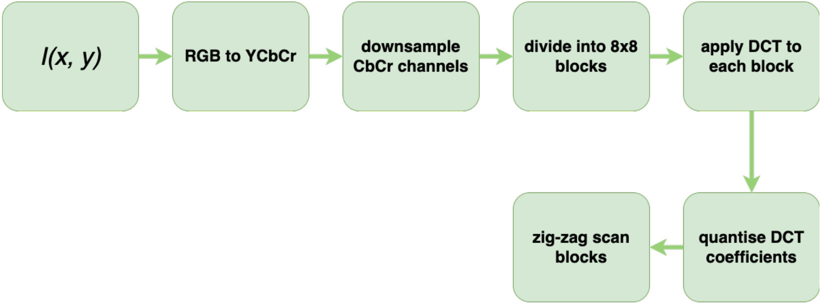


Figure 17: DCT Quantisation

ZigZag Scan

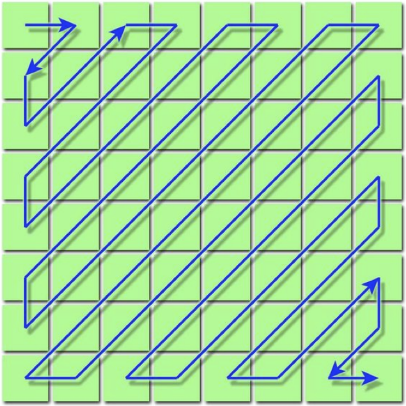


Figure 18: ZigZag Scan

ZigZag Scan

Quantised DCT

65	-27	19	0	0	1	-1	1
-2	-3	-3	6	1	-1	0	1
17	8	-5	0	1	-1	0	1
0	1	6	-1	1	0	-1	1
7	1	0	-1	0	0	0	0
-2	1	1	0	0	0	0	0
1	1	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0

Figure 19: quantised block

ZigZag Scan

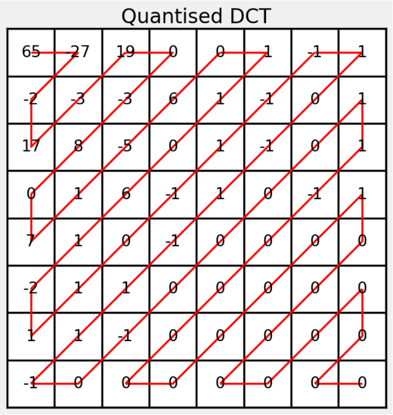


Figure 20: ZigZag Scan

$65, -27, -2, 17, -3,$
$19, 0, -3, 8, 0, \ldots$

ZigZag Scan

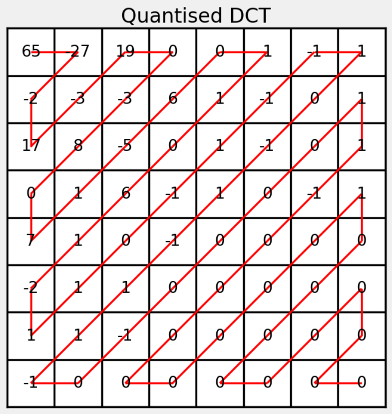


Figure 21: ZigZag Scan

Reads from low frequency coefficients to high frequency coefficients. . .

ZigZag Scan

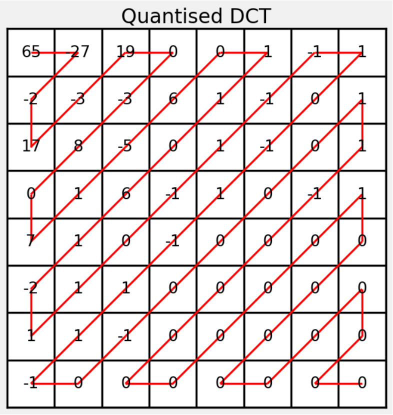


Figure 22: ZigZag Scan

More likely to encode all non-zeros and all zeros together...

- beneficial for the next step...

JPEG Compression

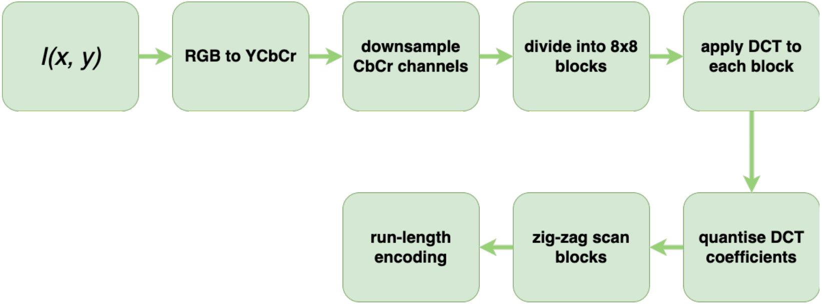


Figure 23: run-length encoding

Run Length Encoding

Extracts series of value and length of runs from sequence of values.

Exploits **inter-pixel** redundancy.

Run Length Encoding

65 -27 -2 17 -3 -3 1 1 1 -2 1 1 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Run Length Encoding

65 -27 -2 17 -3 -3 1 1 1 -2 1 1 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

65 1 -27 1 -2 1 17 1 -3 2 1 3 -2 1 1 2 0 1 -1 1 1 1 0 19

Run Length Encoding

65 -27 -2 17 -3 -3 1 1 1 -2 1 1 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

65 1 -27 1 -2 1 17 1 -3 2 1 3 -2 1 1 2 0 1 -1 1 1 1 0 19

Run Length Encoding

65 -27 -2 17 -**3** -**3** 1 1 1 -2 1 1 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

65 1 -27 1 -2 1 17 1 -**3** **2** 1 3 -2 1 1 2 0 1 -1 1 1 1 0 19

Run Length Encoding

65 -27 -2 17 -3 -3 1 1 1 -2 1 1 0 -1 1 **0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**

65 1 -27 1 -2 1 17 1 -3 2 1 3 -2 1 1 2 0 1 -1 1 1 1 **0 19**

Run Length Encoding

Exploits inter-pixel redundancy

- the relationship between neighbouring “pixels” in the zigzag scan of the DCT coefficient matrix

JPEG Compression

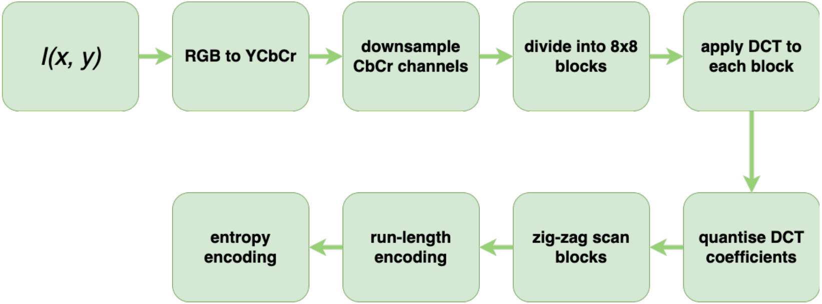


Figure 24: entropy encoding

Entropy Coding

Information and data are not the same thing.

– Claude Shannon, (1948). A Mathematical Theory of Communication.

Entropy Coding exploits **coding redundancy**

– not every value is equally likely.

Entropy Coding encodes a sequence with *variable* length code so that:

- More probable values have fewer bits, and
- less probable values have more bits.

The new alphabet requires fewer bits per pixel.

- How many bits do we need?

Recall: the *probability* of an event is:

$$p_i = \frac{N_i}{N}$$

The *information* in **bits** is:

$$I_i = -\log_2 p_i$$

The *entropy*, the smallest possible **mean** symbol length, is:

$$H = -\sum_i p_i \log_2 p_i$$

We can use these properties to develop a better coding for an image.

- The stream must be decoded *unambiguously*.
- One code cannot be the **prefix** of another.

Huffman Coding

Step 1:

- Arrange values in order of decreasing probability.
- Each forms a *leaf* in the **Huffman tree**.

Huffman Coding

Step 2:

- Merge the two leaves with the smallest probability,
  - *add* the probabilities,
  - insert the node into the sorted list.
- Assign a 1/0 to each branch being merged.

Huffman Coding

Step 3:

- Repeat until only the root node remains.
- Read codewords from the root to the leaves.

Huffman Coding

0	0	1	2	5	5	7	4	5	5
5	1	1	2	1	4	1	4	3	1
5	2	1	2	1	2	2	5	3	1
3	7	2	6	5	3	5	5	1	1
2	7	5	4	5	5	5	3	1	1
7	4	5	5	5	5	5	3	3	5
1	5	5	5	5	1	1	2	2	5
6	5	7	4	2	1	4	1	2	5
1	1	7	2	1	2	4	1	3	5
1	2	0	0	7	4	7	7	4	5

What is the Huffman code for this image?
And, what is the current bit rate?

Count the frequencies of each symbol.

Frequency	Symbol
4	0
23	1
15	2
8	3
10	4
29	5
2	6
9	7

What is the **entropy** of this image?

$p(s)$	$-\log p(s)$	$\times$
0.29	1.786	0.518
0.23	2.120	0.488
0.15	2.737	0.411
0.10	3.322	0.332
0.09	3.474	0.313
0.08	3.644	0.292
0.04	4.644	0.186
0.02	5.644	0.113
	$+$	**2.651**

Sort by the most frequent symbol.

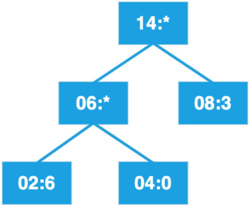
Frequency	Symbol
29	5
23	1
15	2
10	4
9	7
8	3
4	0
2	6

Merge the two leaves with the lowest frequency…

Insert the node into the sorted list.

Frequency	Symbol
29	5
23	1
15	2
10	4
9	7
8	3
6	*

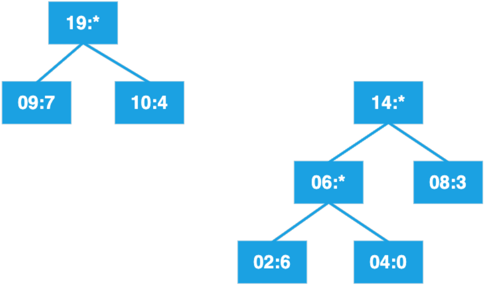
Repeat with the next two lowest frequencies.



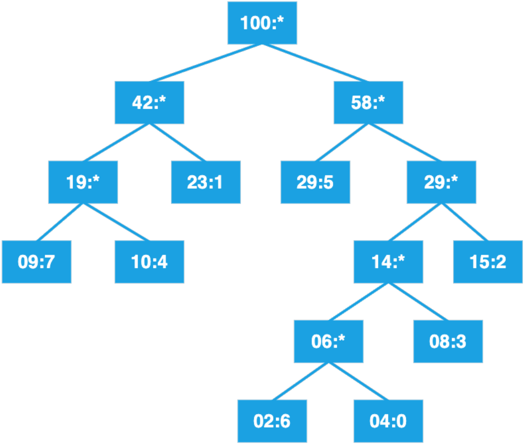
Insert the node into the sorted list.

Frequency	Symbol
29	5
23	1
15	2
14	*
10	4
9	7

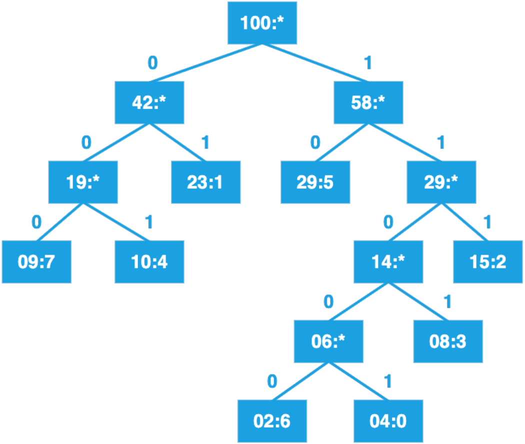
Repeat with the next two lowest frequencies.



Continue until the tree is complete.



Label left branches with **0**, right branches with **1**.



Read from the **root** to compute the new codes.

Code	Symbol
11001	0
01	1
111	2
1101	3
001	4
10	5
11000	6
000	7

Value	p(x)	code length	$\times$
5	0.29	2	0.58
1	0.23	2	0.46
2	0.15	3	0.45
4	0.10	3	0.30
7	0.09	3	0.27
3	0.08	4	0.32
0	0.04	5	0.20
6	0.02	5	0.10
		$+$	**2.68**

We can calculate the bit rate we achieved.

- Not optimal.
- optimal bit rate is 2.65
- our bit rate is 2.68
- The compression ratio is $2.68/3.0 = 0.8933$.

JPEG Compression

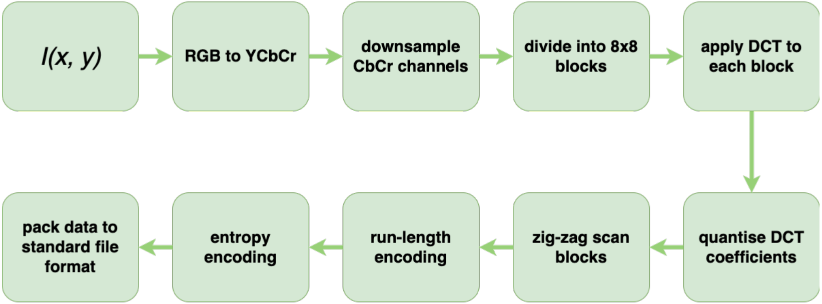


Figure 25: data packing

JPEG Compression

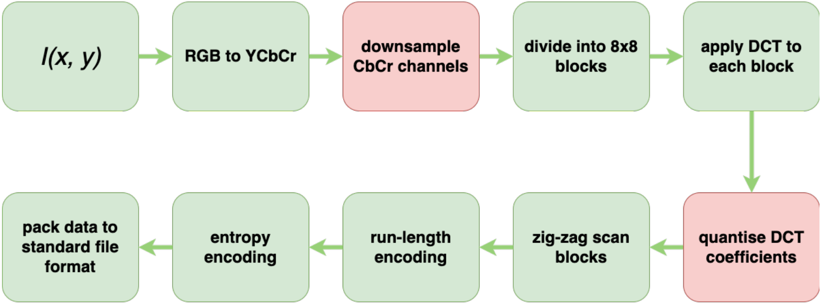


Figure 26: lossy components

JPEG Compression

Figure 27: 50% quality

JPEG Compression

Figure 28: 5% quality

Summary

Three types of redundancy are exploited in image compression.

- psycho-visual redundancy
- inter-pixel redundancy
- coding redundancy
- **JPEG** uses them all.