Image Processing

Hardware Software CoDesign

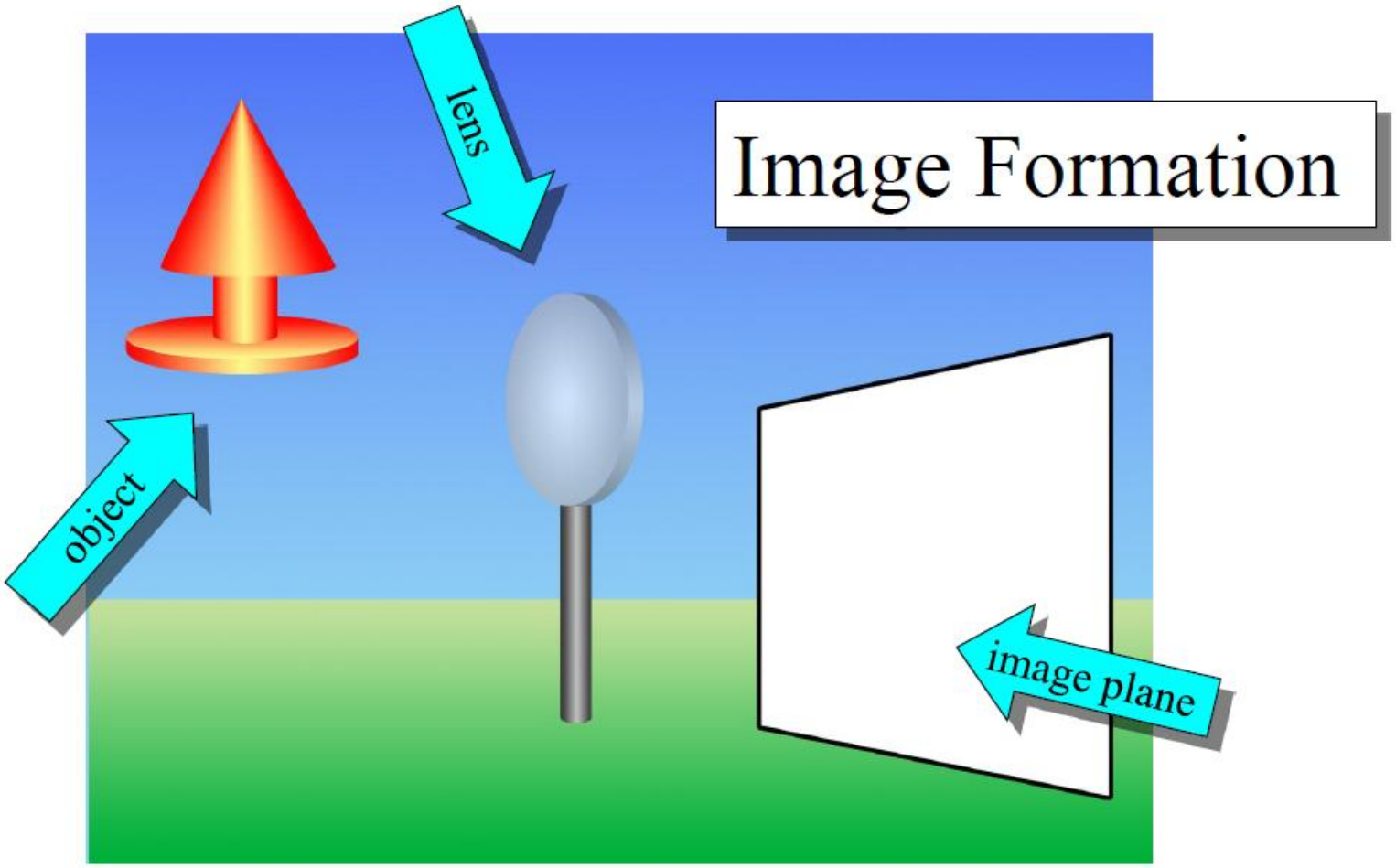
December 2011

Agenda

Image Processing

1. Introduction to Image Processing – Pixelization, Quantization (repeat)
2. The Discrete Cosine Transform
3. Why Compress or Encode Images ?
4. The Steps in Image Compression
5. Digital Camera Example
6. Discussion on the Answering Machine Assignment (Group wise completion)
7. Mid Semester Paper distribution (left-overs and any doubts)

Introduction to Image Processing



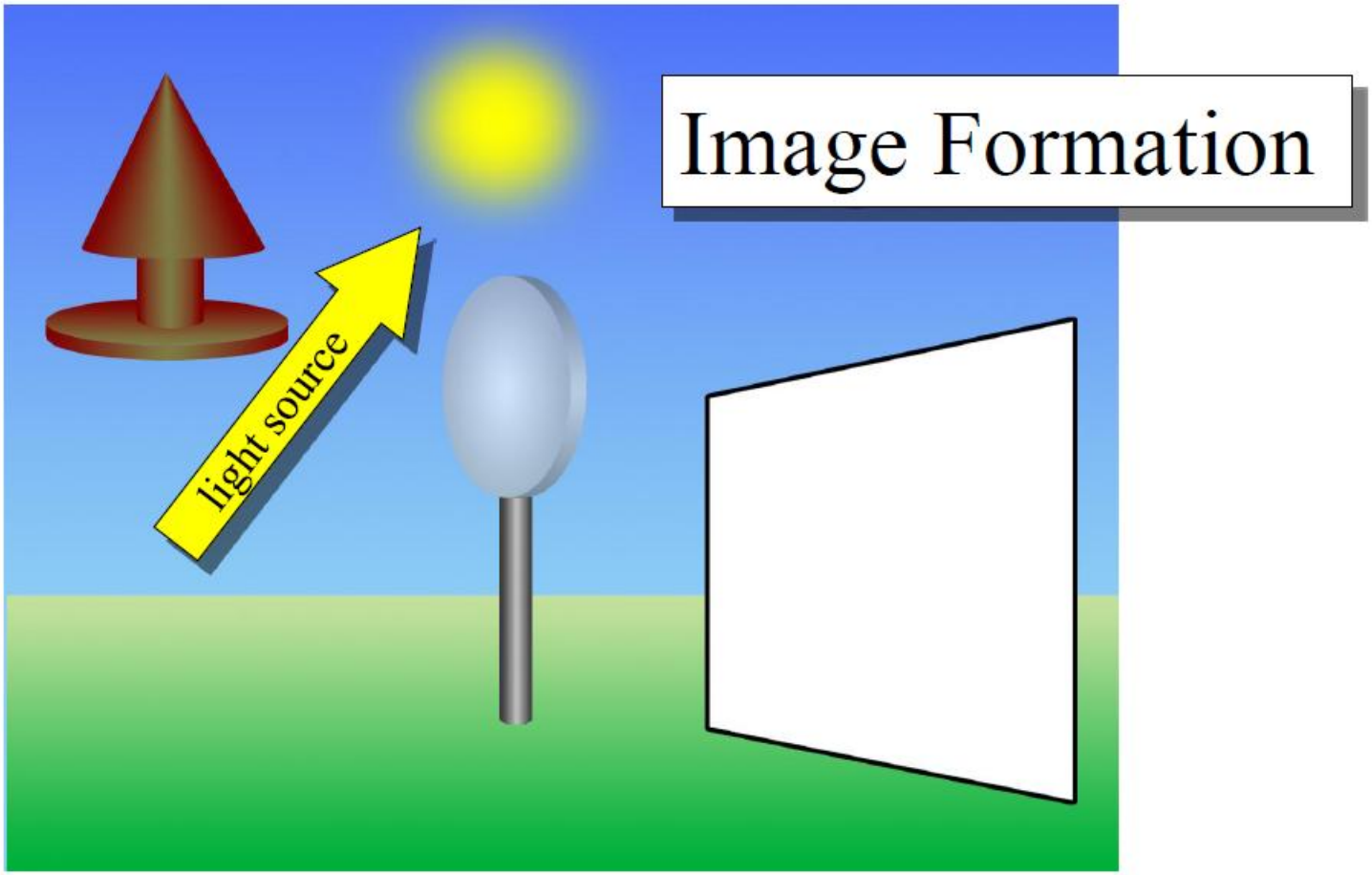


Image Formation
light source

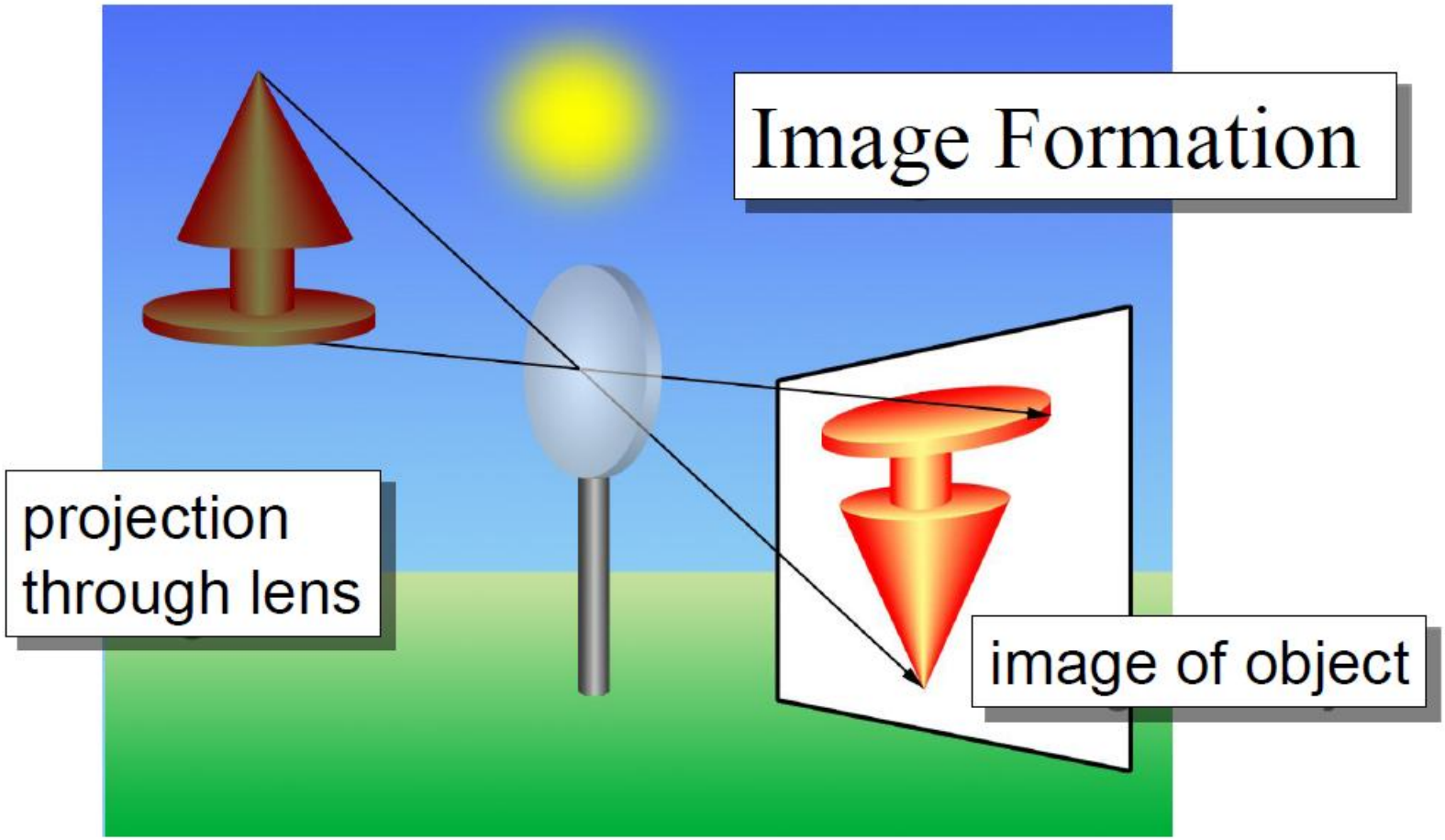


Image Formation
projection through lens
image of object

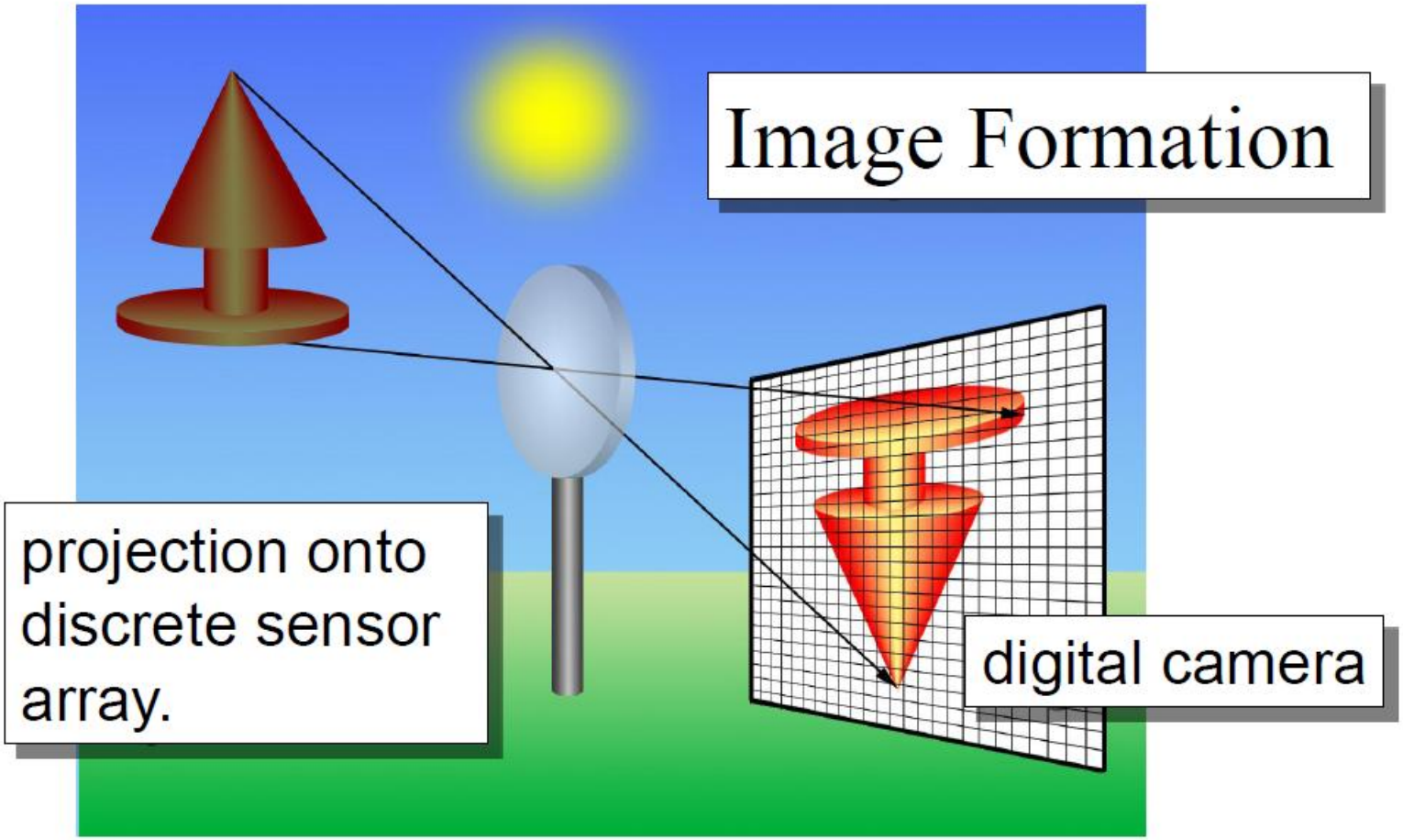


Image Formation
projection onto discrete sensor array.
digital camera

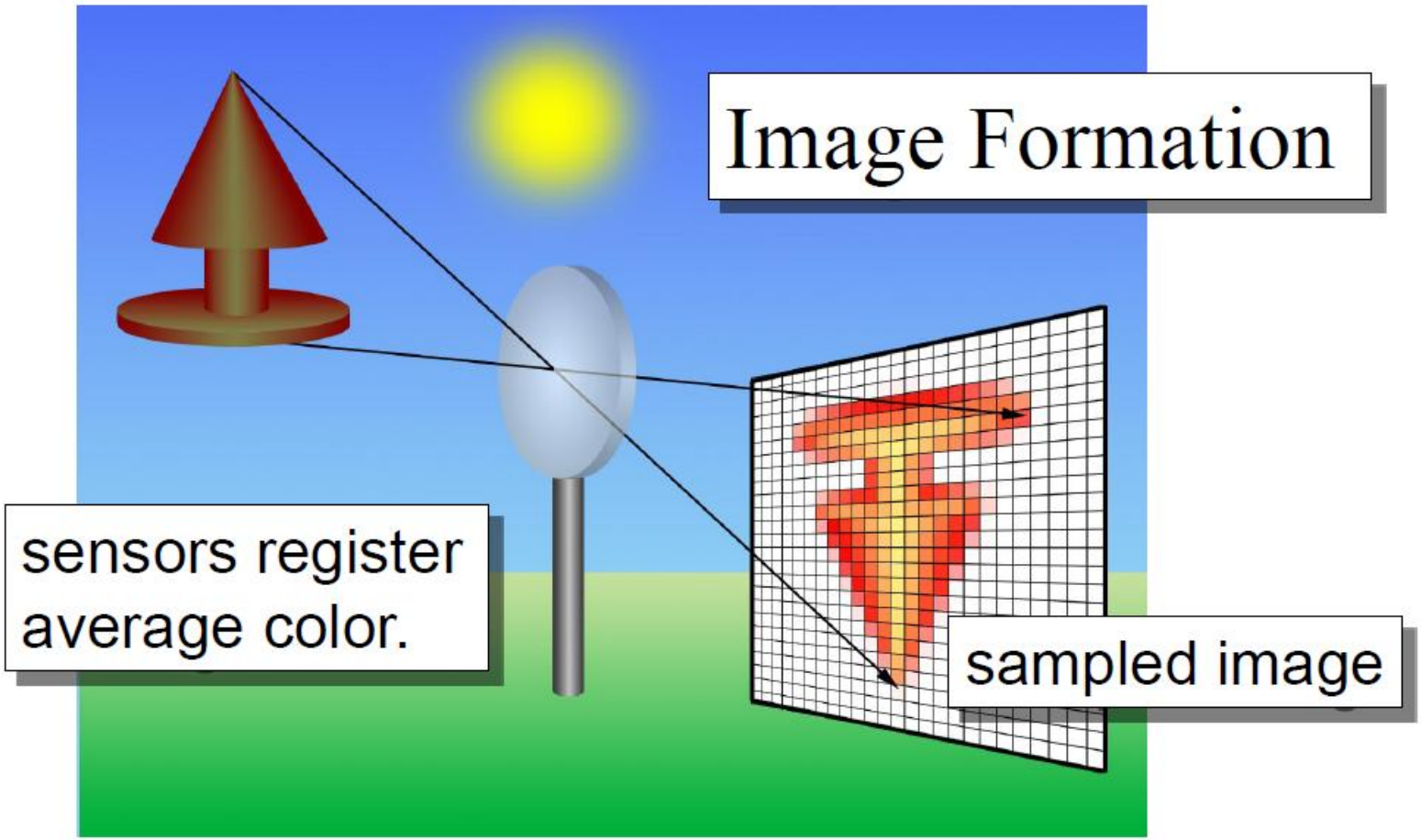


Image Formation
sensors register average color.
sampled image

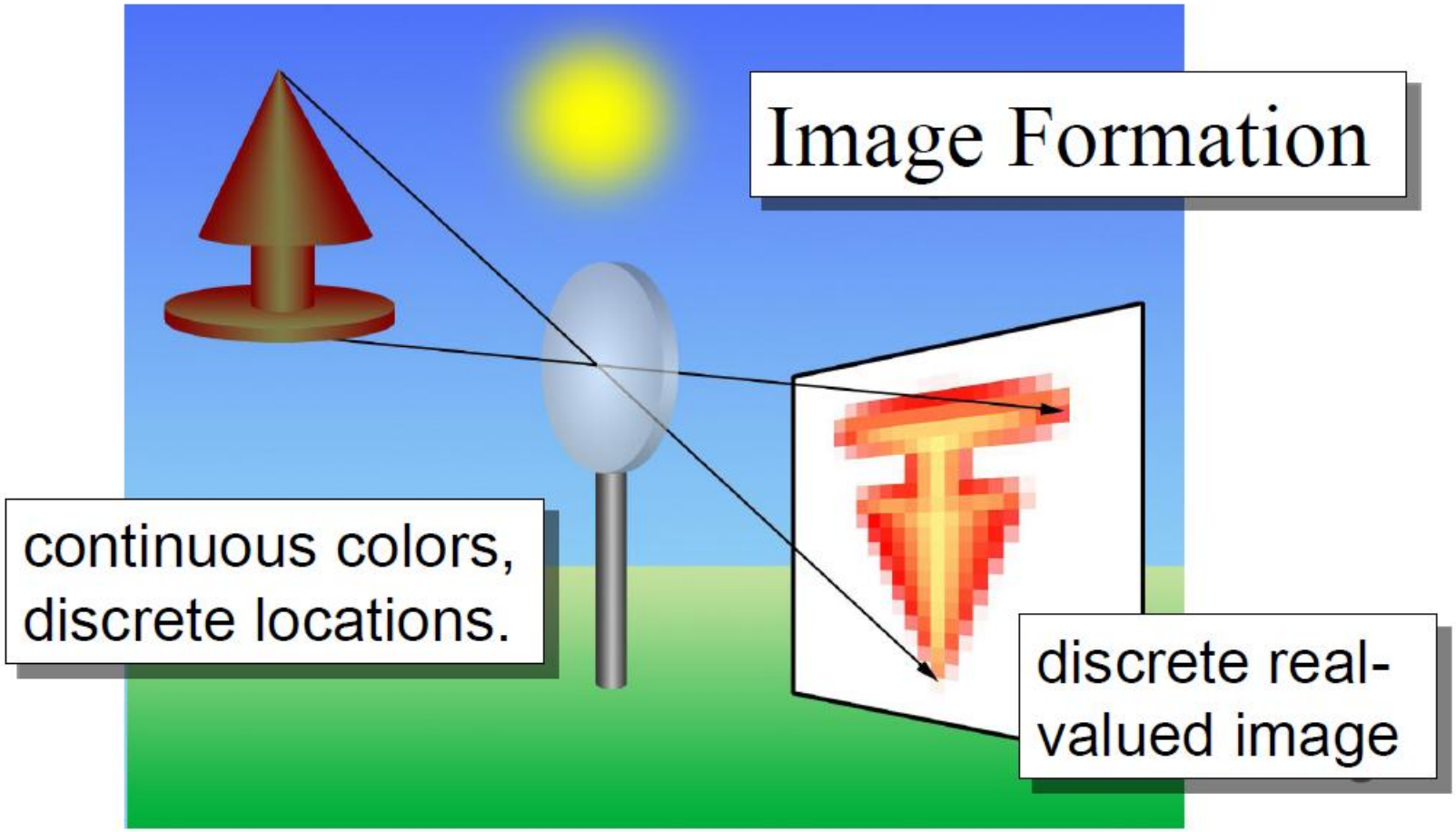
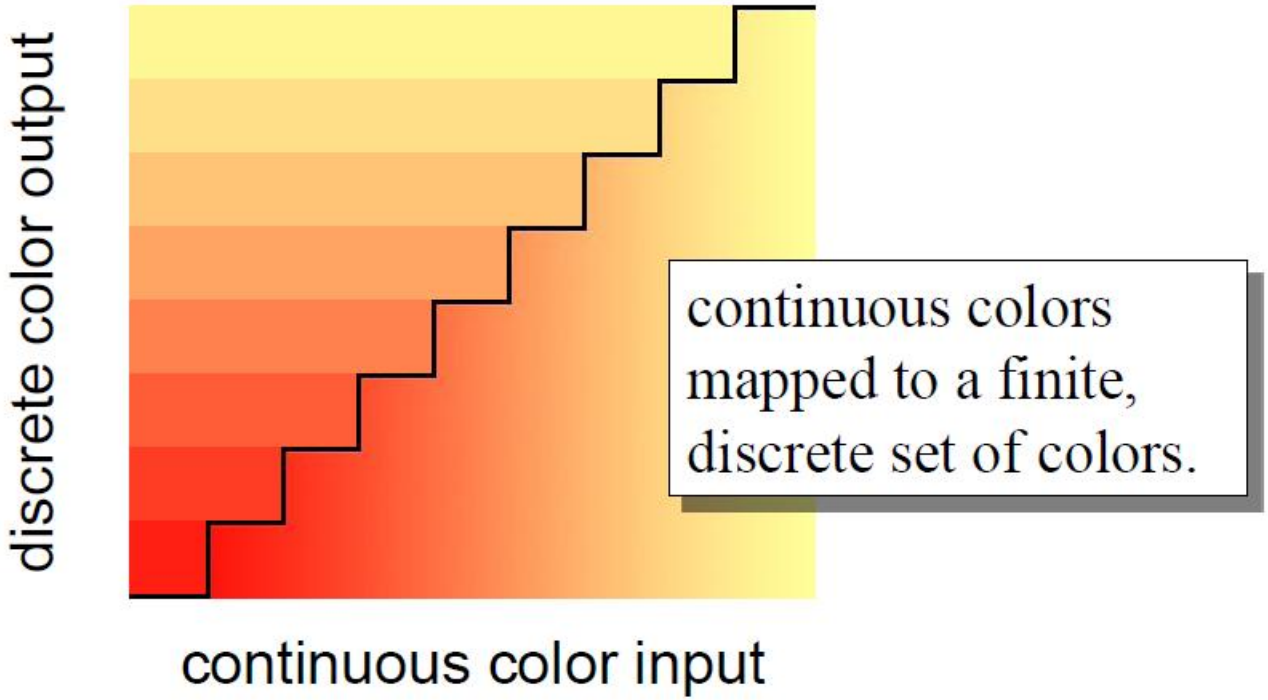
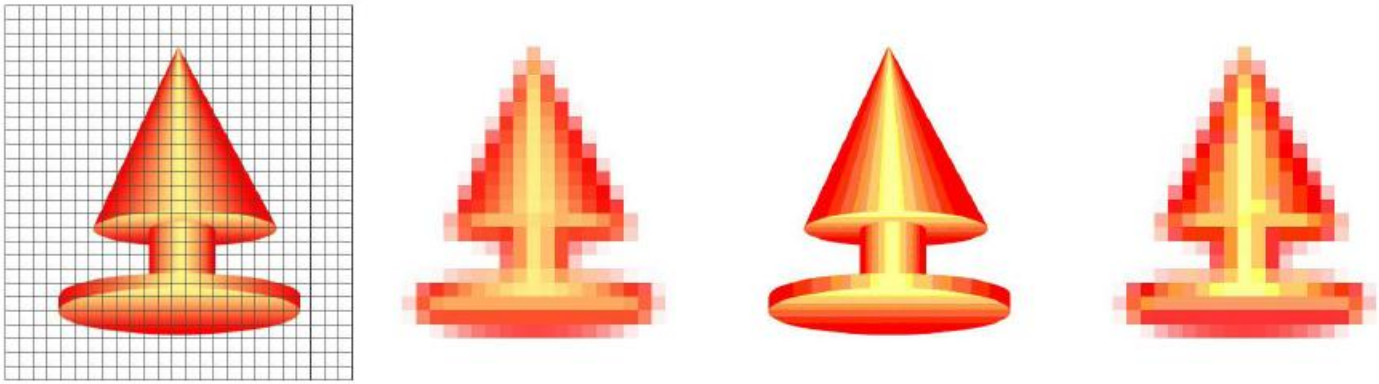


Image Formation
continuous colors,
discrete locations.
discrete real-
valued image

Digital Image Formation: Quantization

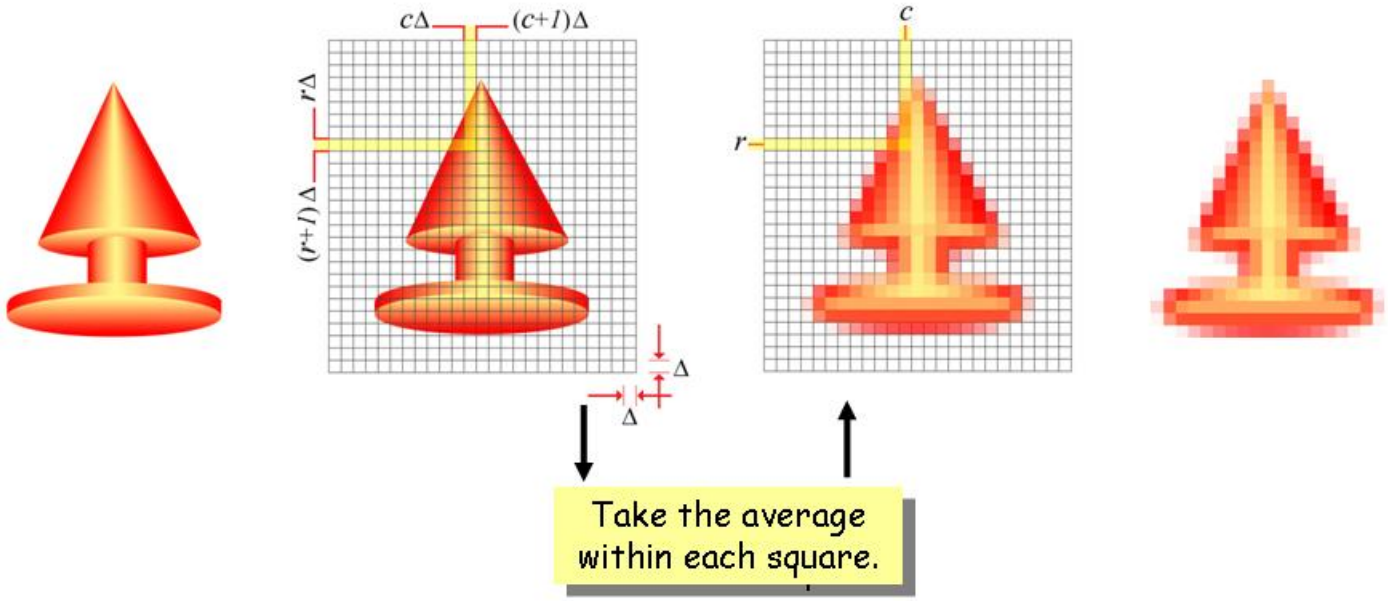


Sampling and Quantization



real image | sampled | quantized | sampled & quantized

Pixelization :-



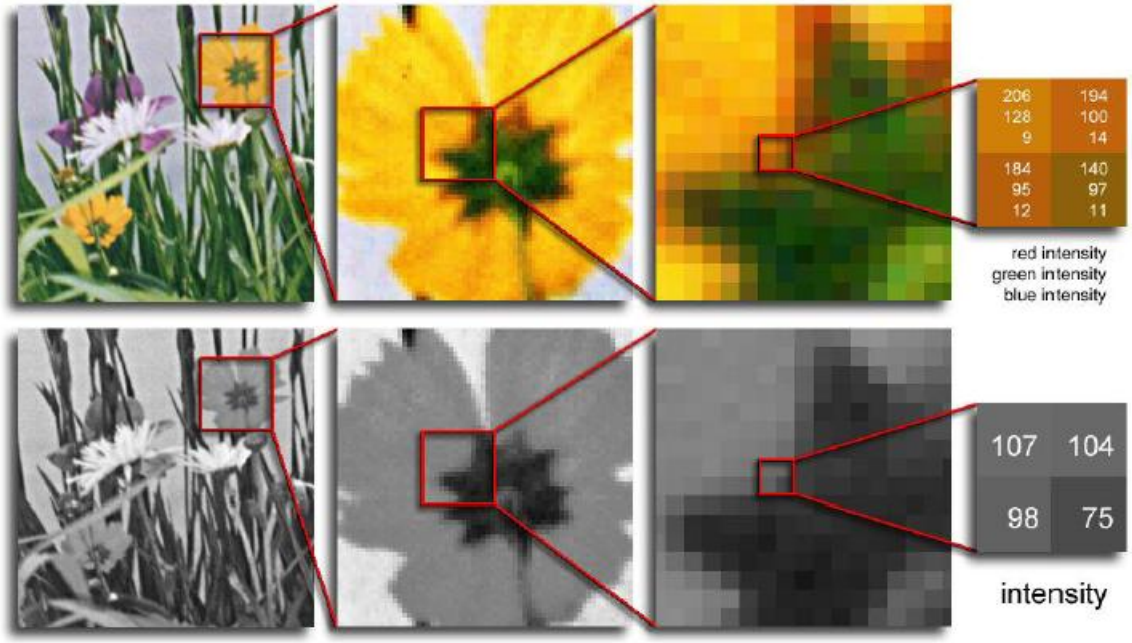
Pixelization :-

Digital Image

Color images have 3 values per pixel; monochrome images have 1 value per pixel.

a grid of squares, each of which contains a single color

each square is called a pixel (for *picture element*)



Why Compress / Encode Images

Original Image :: 352 x 288
24bits per pixel.

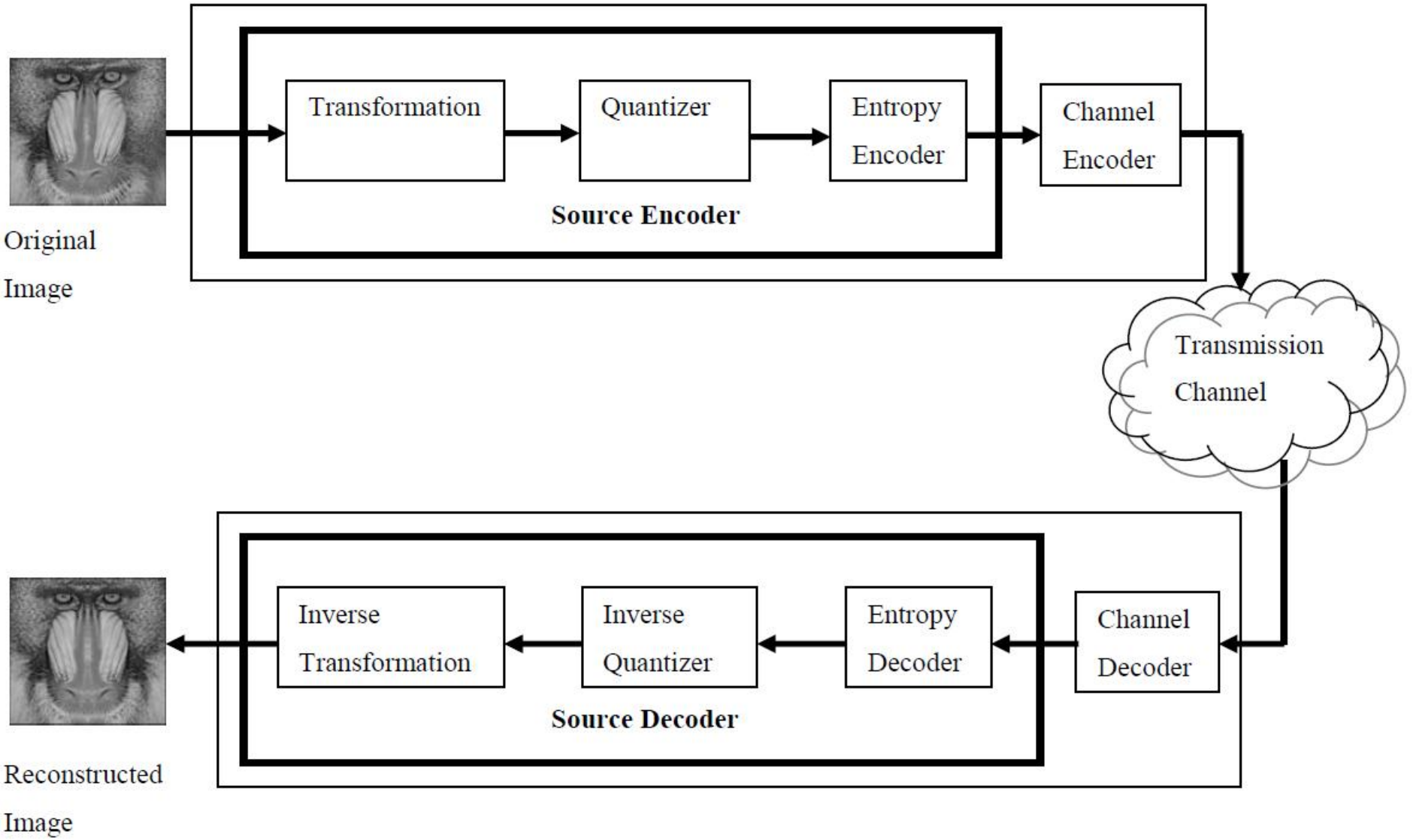
JPG FILE :: LT 20KB

Why Compress / Encode Images

Video Source	Output Data Rate[Kbits/sec]
Quarter VGA @20 frames/sec	36 864.00
CIF camera @30 frames/sec	72 990.72
VGA @30 frames/sec	221 184.00

Transmission Medium	Data Rate [Kbits/sec]
Wireline modem	56
GPRS (estimated average rate)	30
3G/WCDMA (theoretical maximum)	384

Steps in Image Compression



Components of a typical image/video transmission system

Steps in Image Compression

1. If the color is represented by RGB mode, translate it to YCrCb.

2. Divide the file into 8 x 8 blocks.

3. Transform the pixel information from the spatial domain to the frequency domain with the Discrete Cosine Transform.

4. Quantize the resulting values by dividing each coefficient by an integer value and rounding off to the nearest integer.

5. Look at the resulting coefficients in a zig-zag order. Do a run-length encoding of the coefficients ordered in this manner.

6. Follow by Huffman coding.

Steps in Image Compression

Converting RGB to YCbCr

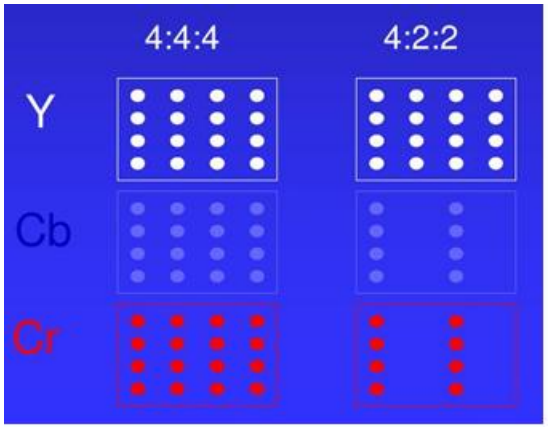
1. YCbCr color mode stores color in terms of its luminance (brightness) and chrominance (hue)

2. The human eye is less sensitive to chrominance than luminance.

3. ?? Compression then can be achieved by storing more luma details than chroma details. ?? Called chroma-sub-sampling.

4. Formats 4:4:4, 4:2:2

5. When Converted from RGB $\rightarrow$ 4:4:4 $\rightarrow$ 4:2:2 the bandwidth is reduced by 50% (achieving compression).



Steps in Image Compression

Converting RGB to YCbCr

The BT.601 equations are used by many video ICs to convert between digital R′G′B′ data and YCbCr are:

$$Y = (77/256)R' + (150/256)G' + (29/256)B'$$

$$Cb = -(44/256)R' - (87/256)G' + (131/256)B' + 128$$

$$Cr = (131/256)R' - (110/256)G' - (21/256)B' + 128$$

$$R' = Y + 1.371(Cr - 128)$$

$$G' = Y - 0.698(Cr - 128) - 0.336(Cb - 128)$$

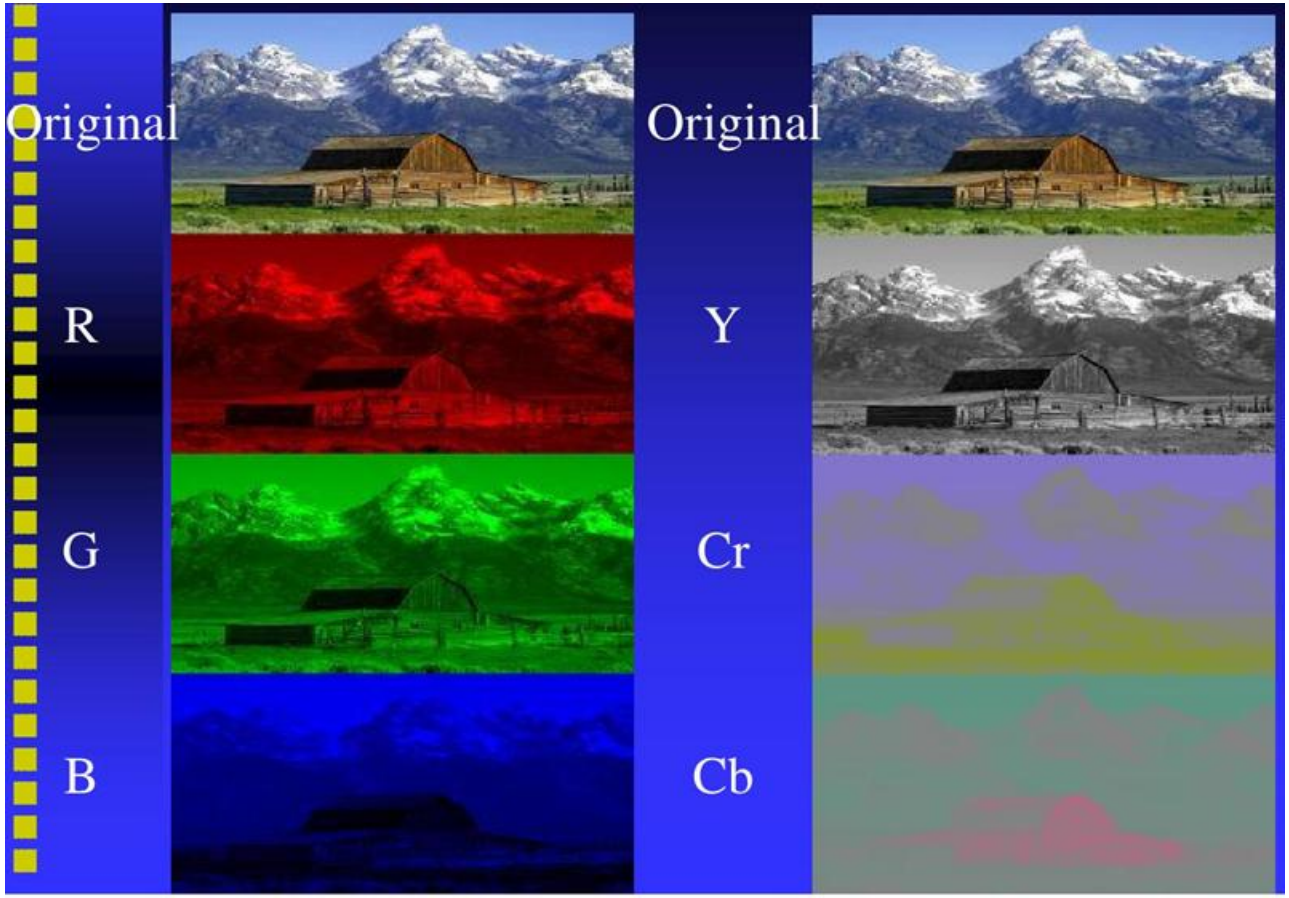
$$B' = Y + 1.732(Cb - 128)$$

1. From a Hardware - Software point of view :: This is a computation requiring bandwidth.

2. GROUP DISCUSSION :: How does one implement in HW.

Steps in Image Compression

Converting RGB to YCbCr



Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

1. The DCT transforms the data from the spatial domain to the frequency domain.

2. The spatial domain shows the amplitude of the color as you move through space.

3. The frequency domain shows how quickly the amplitude of the color is changing from one pixel to the next in an image file.

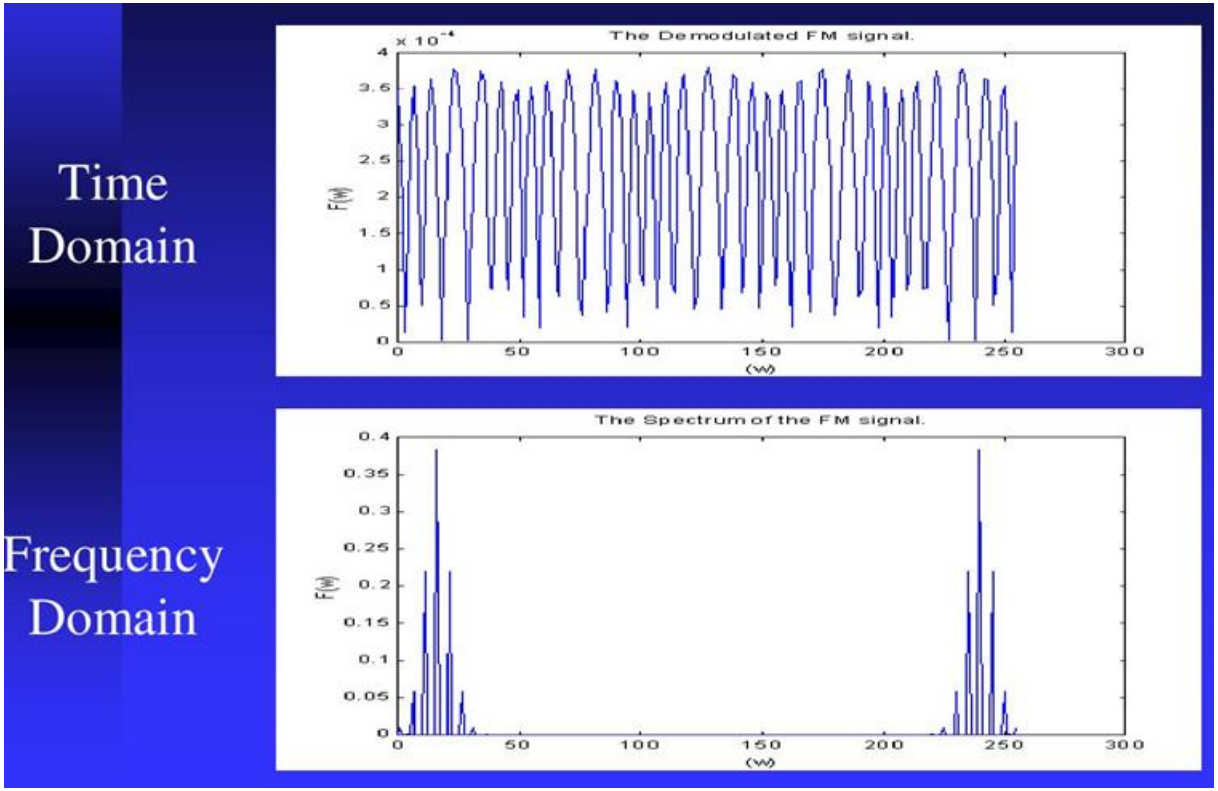
4. For the 8 x 8 matrix of color data, we'll get an 8 x 8 matrix of coefficients for the frequency components.

5. GROUP DISCUSSION :: What is being done here actually?

Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

1. GROUP DISCUSSION :: What is being done here actually?
2. "Use Time Domain –¿ Frequency Domain" transformation.
3. "Use the fact that human eye is imperfect"
4. "Achieve lossy compression, which satisfies the bandwidth / size consideration"



Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

1. Discrete Cosine Transform (DCT) is a Fourier-related transform similar to the Discrete Fourier Transform (DFT), but using only real numbers.

2. The most common variant of the DCT is the type II DCT; its inverse is called IDCT (inverse DCT)

3. The frequency domain is a better representation for the data because it makes it possible to separate out $\rightarrow$ and throw away $\rightarrow$ information that isn't very important for human perception.

4. The human eye is not very sensitive to high frequency changes $\rightarrow$ especially in photographic images, so the high frequency data, to some extent, can be discarded

Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

The most common DCT definition of a 1-D sequence of length N is

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{\pi(2x+1)u}{2N}\right], \tag{1}$$

for $u = 0, 1, 2, \ldots, N-1$. Similarly, the inverse transformation is defined as

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) \cos\left[\frac{\pi(2x+1)u}{2N}\right], \tag{2}$$

for $x = 0, 1, 2, \ldots, N-1$. In both equations **(1)** and **(2)** $\alpha(u)$ is defined as

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0. \end{cases} \tag{3}$$

It is clear from **(1)** that for $u = 0$, $C(u=0) = \sqrt{\frac{1}{N}} \sum_{x=0}^{N-1} f(x)$. Thus, the first transform coefficient is the average value of the sample sequence. In literature, this value is referred to as the *DC Coefficient*. All other transform coefficients are called the *AC Coefficients*[4].

Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right], \quad (4)$$

for $u,v = 0,1,2,\ldots,N-1$ and $\alpha(u)$ and $\alpha(v)$ are defined in **(3)**. The inverse transform is defined as

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right], \quad (5)$$

for $x,y = 0,1,2,\ldots,N-1$.

Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

- An Example 8x8 block of pixel data is shown here >
- The next step is to transform the 8x8 matrix from a positive range to one centered around zero.
- Since 8 bits are used for each value, each value is subtracted by 128.
- The resultant block is shown here ->

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

$x \longrightarrow$

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−73	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−60	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

$y \downarrow$

Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

- Original 8x8 Pixel block

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

- Corresponding DCT coefficient block

$u \longrightarrow$

−415	−30	−61	27	56	−20	−2	0
4	−22	−61	10	13	−7	−9	5
−47	7	77	−25	−29	10	5	−6
−49	12	34	−15	−10	6	2	2
12	−7	−13	−4	−2	2	−3	3
−8	3	2	−6	−2	1	4	2
−1	0	0	−2	−1	−3	4	−1
0	0	−1	−4	−1	0	1	2

$v \downarrow$

Steps in Image Compression

Divide into 8x8 blocks and Transform to Frequency Domain :: DCT

1. The DCt is lossless in the sense that the inverse DCt will give back exactly the same initial information.

2. The values from the DCT are initially floating point.

3. They are changed to integers by "quantization".

4. The roundoff errors that happen will continue to happen in a say embedded system though.

5. "GROUP DISCUSSION" : How do you want the DCT step be done HW or SW?

Steps in Image Compression

Quantization

1. Quantization involves dividing each coefficient by an integer between 1 and 255 (if 8bits are used) and rounding off.

2. The quantization table is chosen to reduce the precision of each coefficient to no more than necessary.

3. The quantization table is carried along with the compressed file.

4. $B_{j,k} = round(\frac{A_{j,k}}{Q_{j,k}})$ for $j = 0, 1, 2, ...N_1 - 1$ and $k = 0, 1, 2, ...N_2 - 1$

Steps in Image Compression

Quantization

- Quantization Table

16	11	10	16	24	40	51	61
12	12	14	16	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

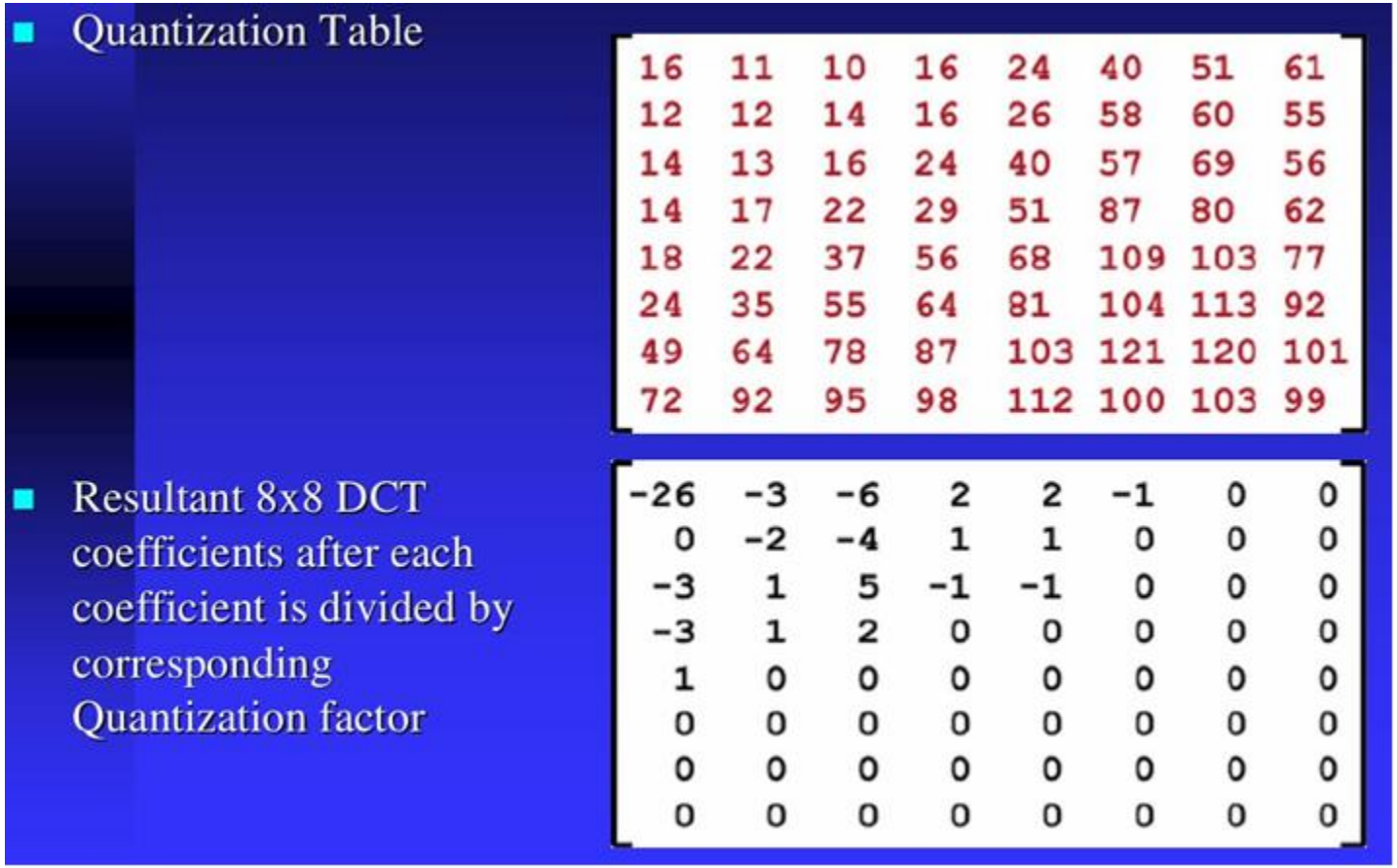
- Resultant 8x8 DCT coefficients after each coefficient is divided by corresponding Quantization factor

-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	0	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

Steps in Image Compression

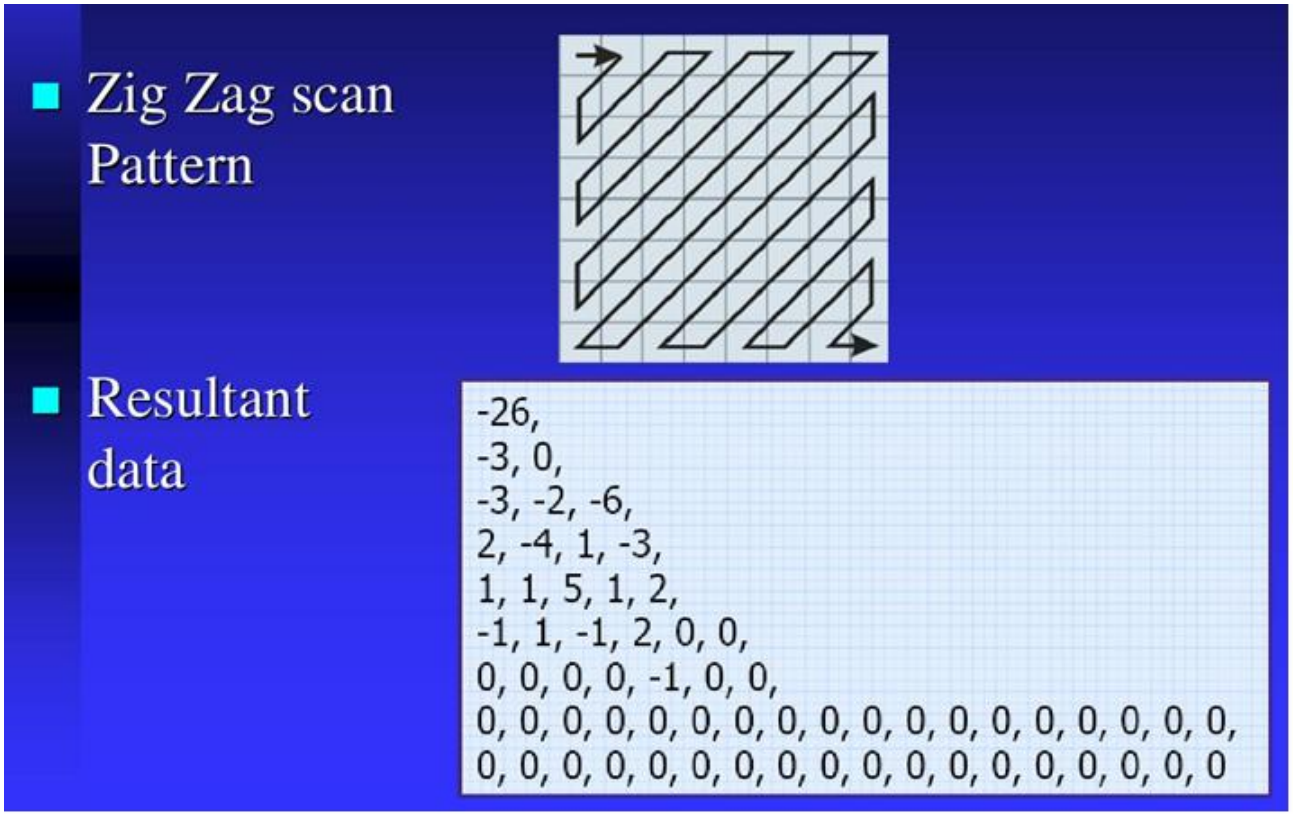
Entropy Coding

1. This is done so that the coefficients are in order of increasing frequency
2. The higher frequency coefficients are more likely to be 0 after quantization
3. This improves the compression of run-length encoding
4. Later, do run-length encoding and Huffman coding



Steps in Image Compression

Entropy Coding



1. The Output of the Zig-Zag scan is further encoded using Run Length Encoding procedure

2. In Run Length Encoding, consecutive pixels with the same value are encoded using a run-length and value pair

3. Example :: if 0x000 comes 10 times $\rightarrow$ 0x0A 0x000

Steps in Image Compression

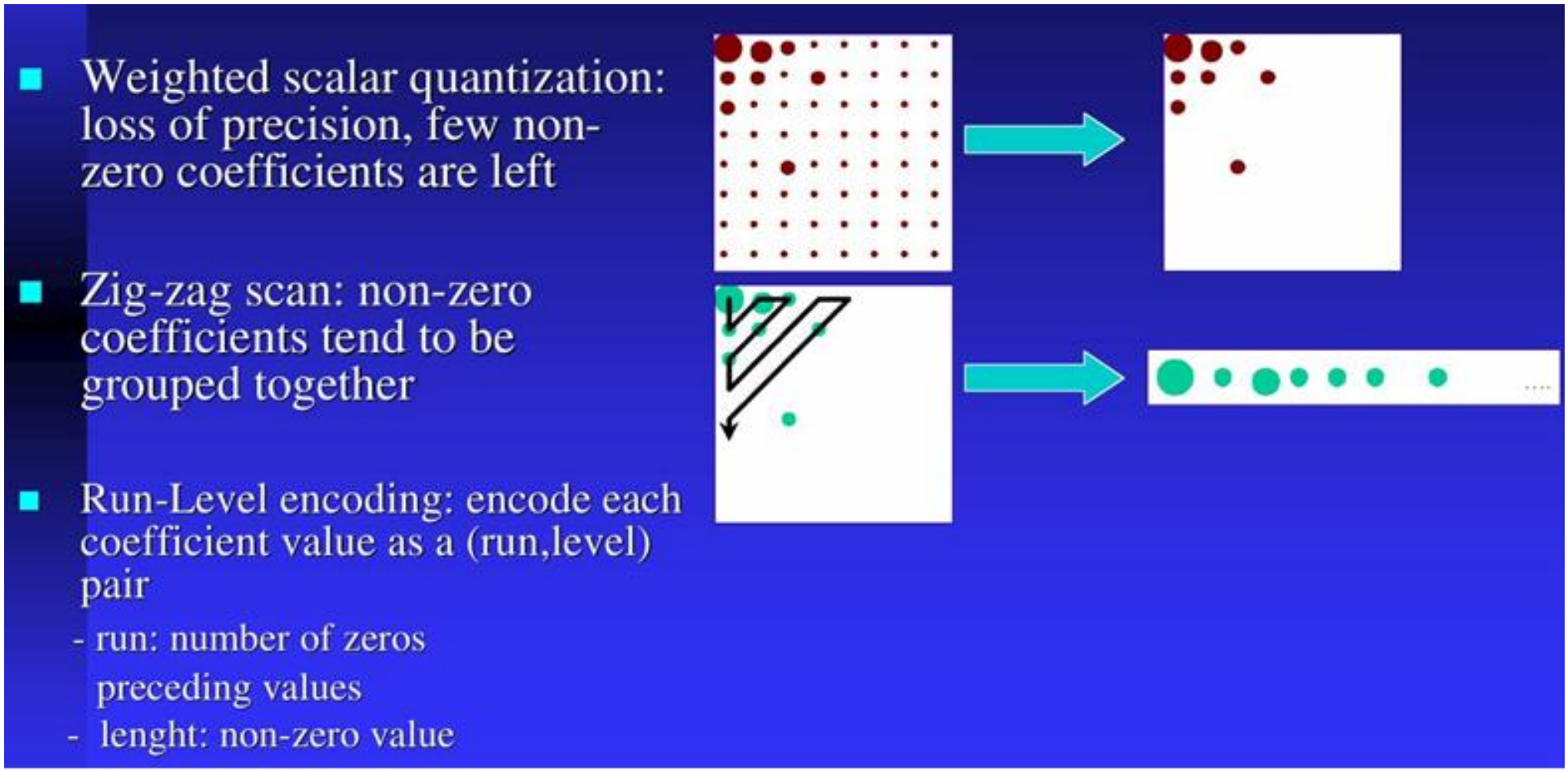
Huffman EnCoding

- The Output of the RLE data is further encoded using Huffman encoding
- Huffman coding refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.
- An EOB (End Of Block) marker is put at the end and the data is written to a file

Char	Freq	Code
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Steps in Image Compression

Small Summary of Coding Methods



Introduction to Image Processing

Acknowledgements

1. http://www.archive.org/details/Lectures_on_Image_Processing :: Richard Alan Peters II :: Dept of Electrical Engg & CS :: Vanderbilt University School of Engineering.
   (a) EECE253_01_Intro.pdf
   (b) EECE253_03_PointProcessing.pdf
   (c) EECE253_10_PixelizationQuantization.pdf
2. YCbCr to RGB Considerations :: YCbCr_Intersil_AppNote.pdf
3. http://www.slideshare.net/sanjivmalik/video-compression-basics :: Sanjiv Malik
4. Direct Use :: digicam.ppt :: http://www.facweb.iitkgp.ernet.in/ anupam/ppts.html :: Lecture Notes from Prof. Anupam Basu IIT KGP.
5. Embedded System Design - A Unified Hardware / Software Introduction :: Ch 7
6. Huffman_Coding.ppt :: Vida Movahedi