Transform Coding of Image and Video

Dr. Mike Biggar

mjbiggar@unimelb.edu.au

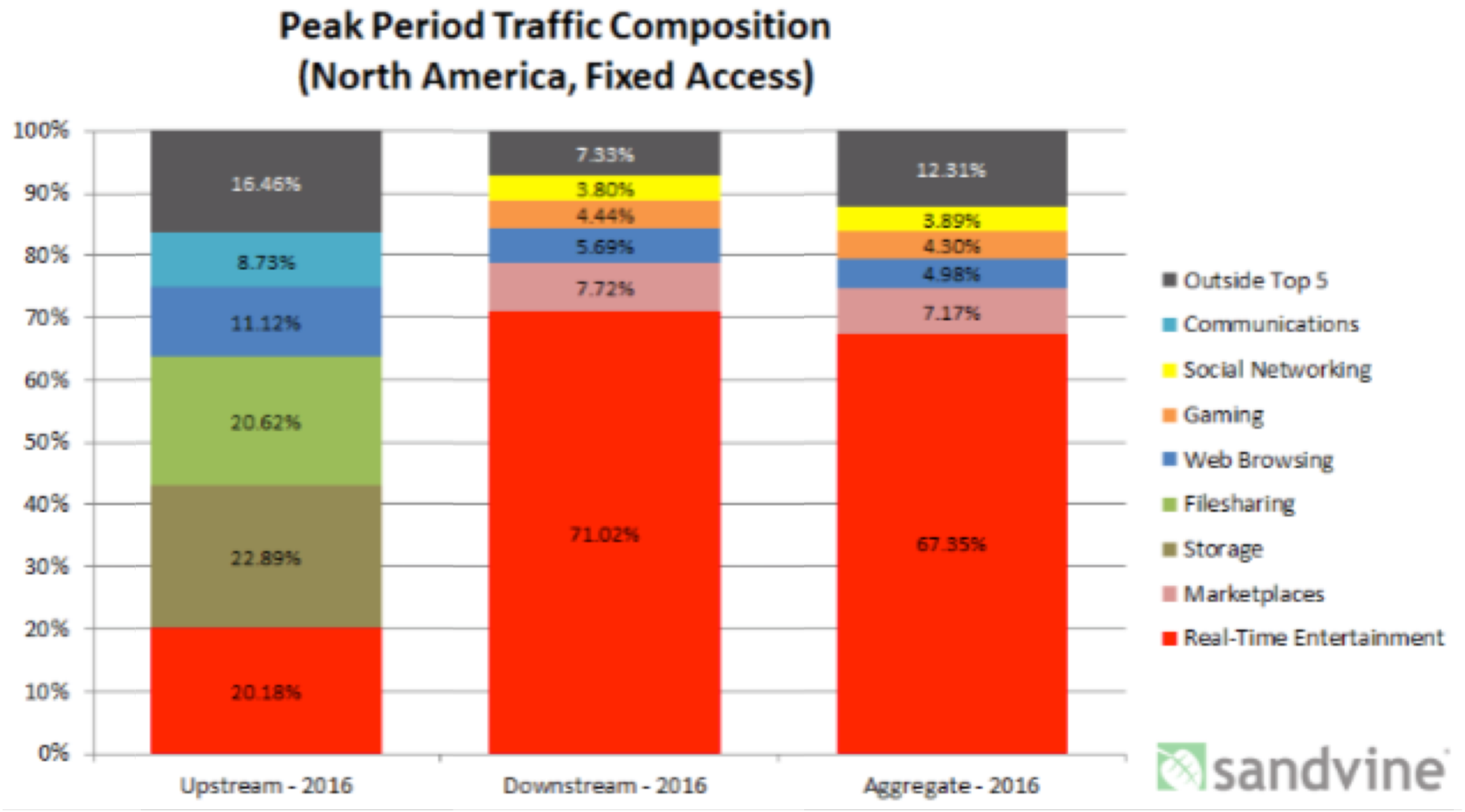
Who is this guy?

- B.E. (Elec) and M.Eng.Sc. degrees from University of Melbourne
- PhD from Imperial College, London (Advanced image and video coding methods)
- Most of career with Telecom Australia / Telstra
- Headed multimedia R&D group at Telstra Research Laboratories
- Represented Telstra, and Australia, on international standards groups (most notably MPEG)
- Specialist in "Emerging Technologies" for the Telstra CTO (until 2013)
- Teaches the *ELEN90014 Multimedia Content Delivery* unit at UniMelb (Master of Telecommunications Engineering).
- Interests in:
  - Video and image capture and reproduction
  - Media coding
  - Media streaming
  - Content management
  - Digital Rights Management
  - ….

Why include Image and video?

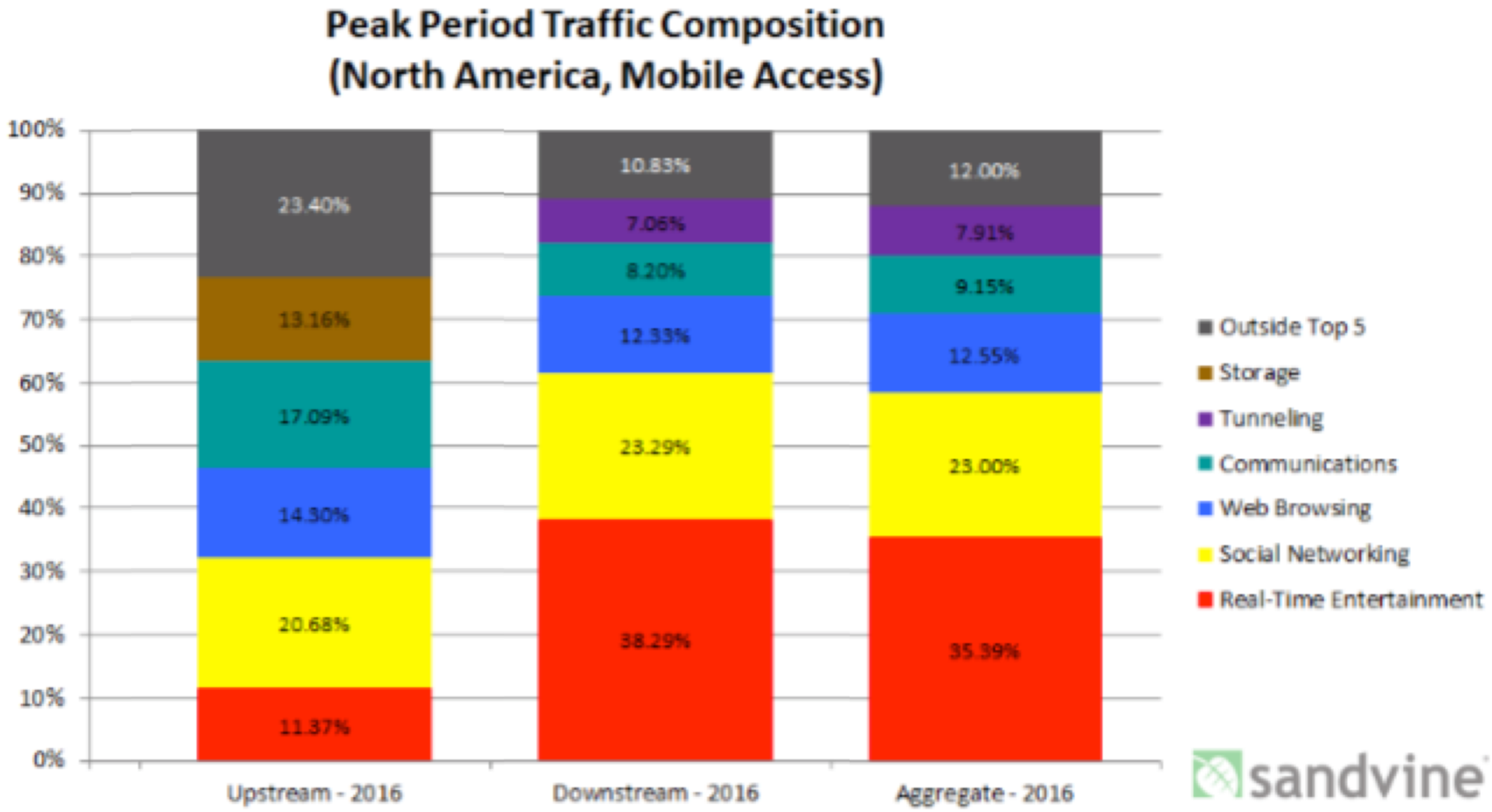
- Multimedia data dominates traffic on the Internet
- Optimum performance depends on both knowledge of network technologies AND multimedia representation and streaming
- Customer experience is an important way for telcos and vendors to differentiate themselves

How important is multimedia traffic to the network?



(Source: Sandvine Internet Phenomena Report, October 2016, https://www.sandvine.com/downloads/general/global-internet-phenomena/2016/global-internet-phenomena-report-latin-america-and-north-america.pdf)

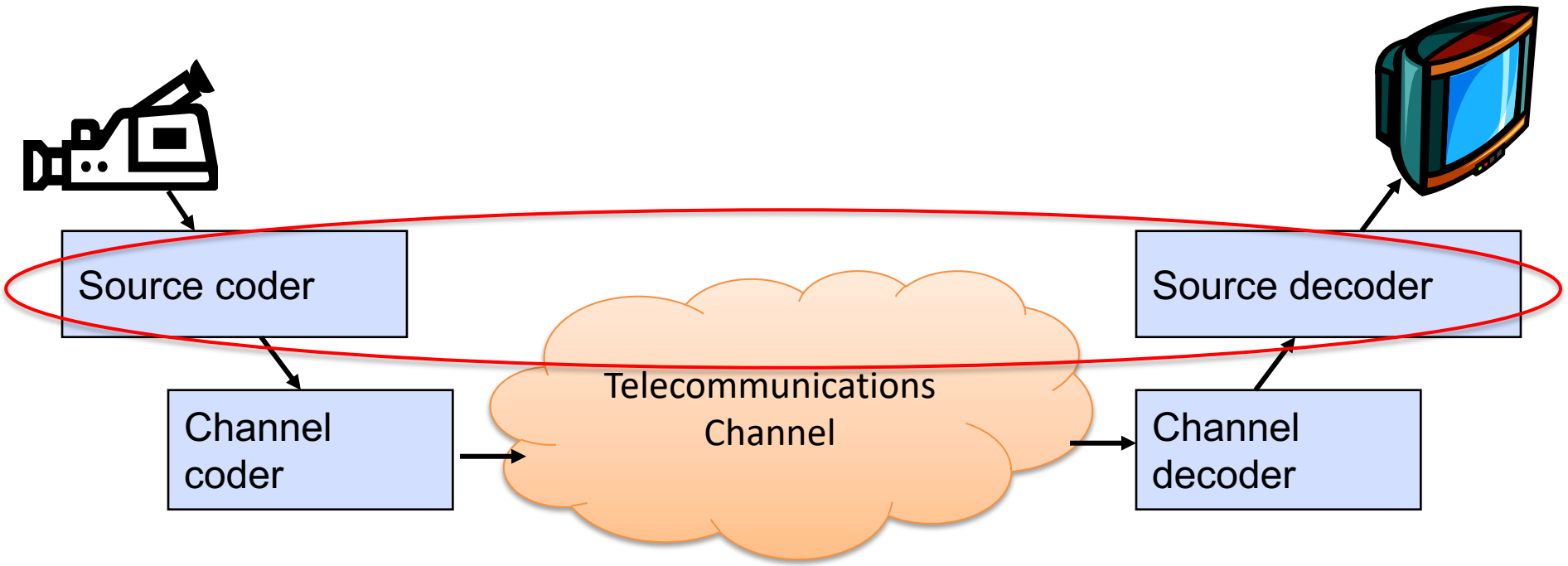
And what about mobiles?



(Source: Sandvine Internet Phenomena Report, October 2016, https://www.sandvine.com/downloads/general/global-internet-phenomena/2016/global-internet-phenomena-report-latin-america-and-north-america.pdf)

Source and channel coding

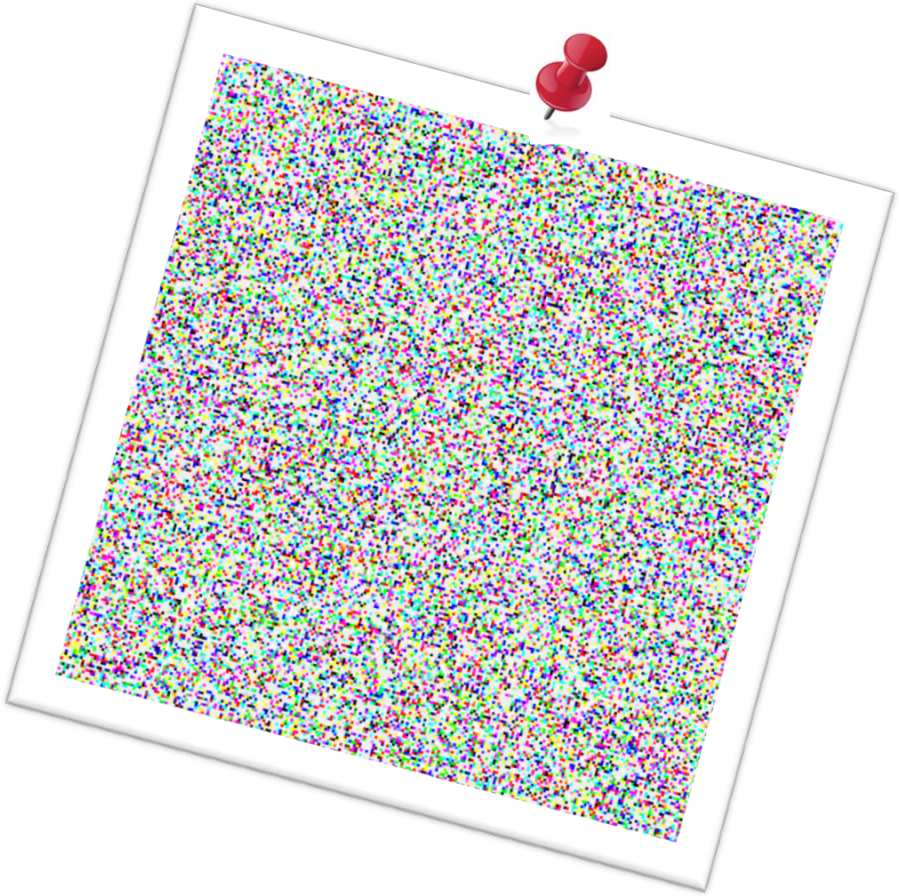
- In the channel coder
  - The channel coder structures the data according to the characteristics of the communication channel. E.g. error protection codes
- In the source coder
  - The source coder structures the data according to the characteristics of the signal to be carried – i.e. compression



Lecture contents

- Principles of image compression
- Redundancy in image data
- Predictive Coding
- Transform Coding and the DCT
- Redundant vs irrelevant data
- Removing irrelevant data; quantisation
- Colour representation of images
- Quantisation of DCT coefficients
- JPEG coder and decoder components and block diagram
- Video coding using inter-frame prediction

Do we often see pictures like this?



(Image: http://blog.reversednormal.com/?p=52)

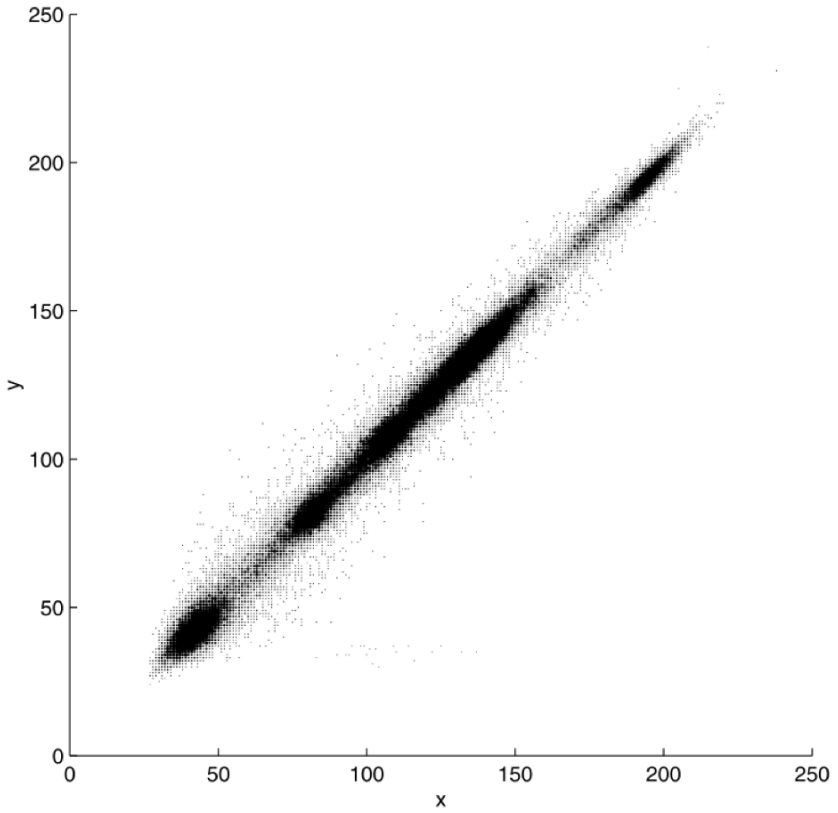
Images typically show structure.

- There is normally correlation between nearby pixels

12 Apostles photo: M. Biggar

Adjacent pixel correlation

"Lena" test image



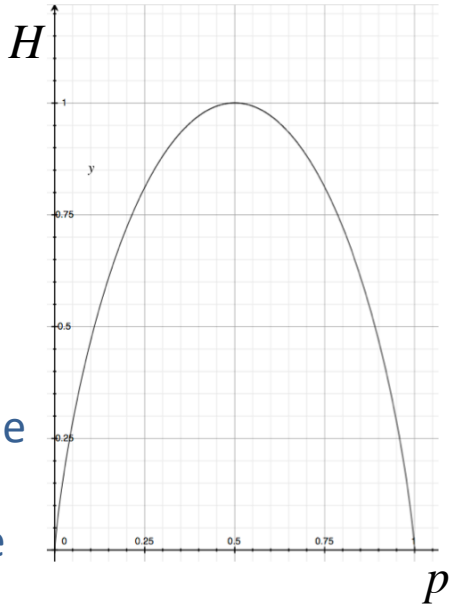
Scatter plot of adjacent pixel value pairs

(Scatter plot from *The Transform and Data Compression Handbook*, Ed. K. R. Rao and P.C. Yip., Boca Raton, CRC Press LLC, 2001)

Remember Information Theory?

- We get maximum average information when outcomes are equiprobable
- We convey maximum information if we have no idea which is the more likely of the possible data we are about to receive
- Entropy of a random binary event:

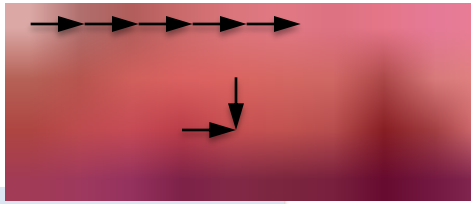
$$H = -p\log(p) - (1-p)\log(1-p)$$

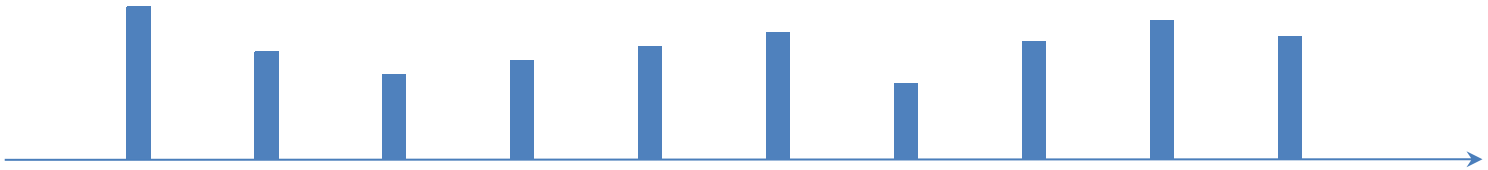


- Same principle applies to pixels, images, audio samples,….
- E.g. Image pixels:
  - If we have a good idea of the value (brightness, colour) of the next pixel that's about to be sent to us, then we have not received as much information as we could with the available bits
  - If we can decorrelate the information we send (i.e.. Transform our pixel data such that all possible outcomes are equally likely), then we have maximised the information content of every bit we sent (i.e. used the minimum number of bits to represent our image).

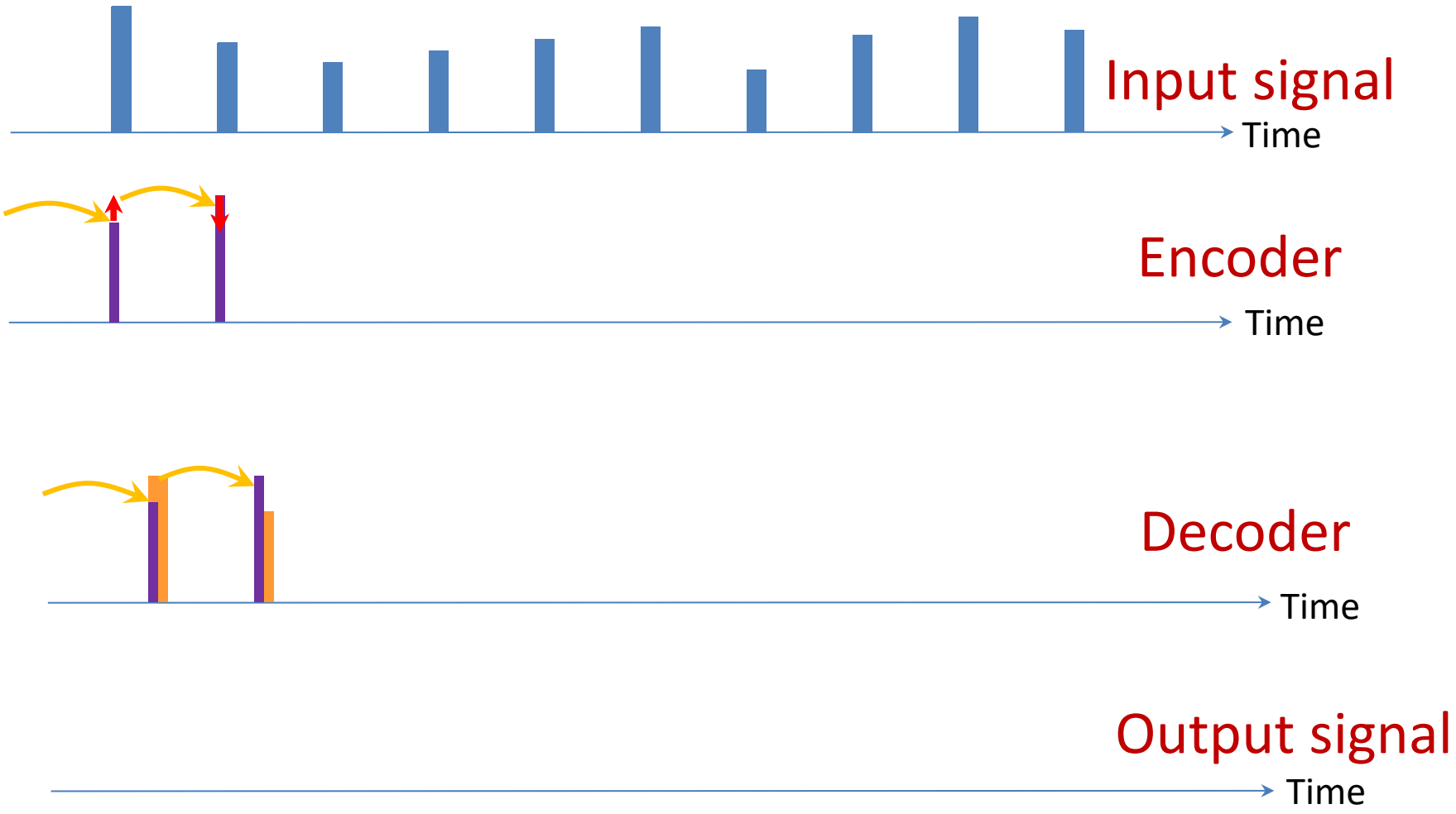
Predictive coding

- Exploit correlation between elements of the signal
- Estimate the next sample (or block, or picture,…) from previously transmitted data and encode just the prediction error
  - Prediction errors have NOTHING to do with transmission errors! They are just the differences between input signals to be communicated to a receiver, and the estimates of those signals obtained from our prediction algorithm
- DPCM (Differential Pulse Code Modulation) is a simple 1-D example of predictive coding
- 1-dimensional prediction or Multi-dimensional prediction
- Critically important: Both the encoder and the decoder must be able to perform the same prediction!



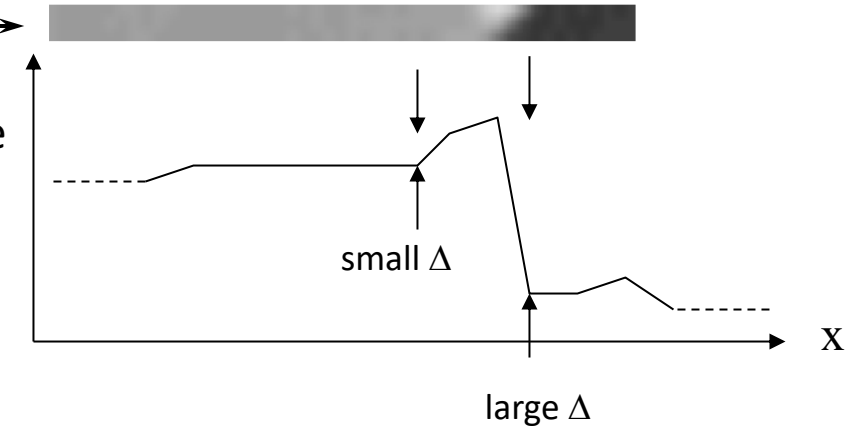


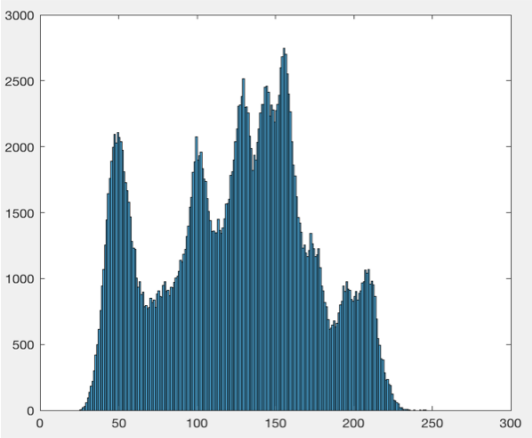
DPCM: Illustration



We're still sending 1 number per sample, so why is this a good thing to do?

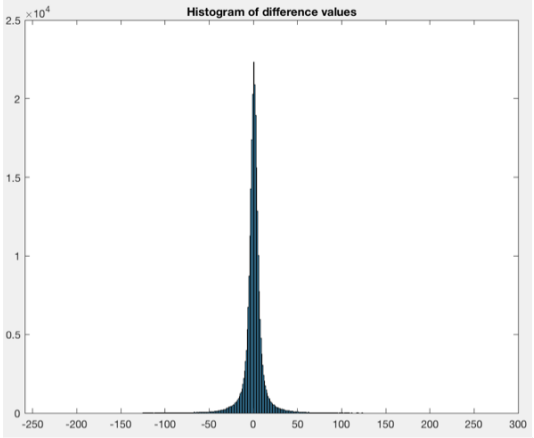
Predictive coding - DPCM





Histogram of original pixel values

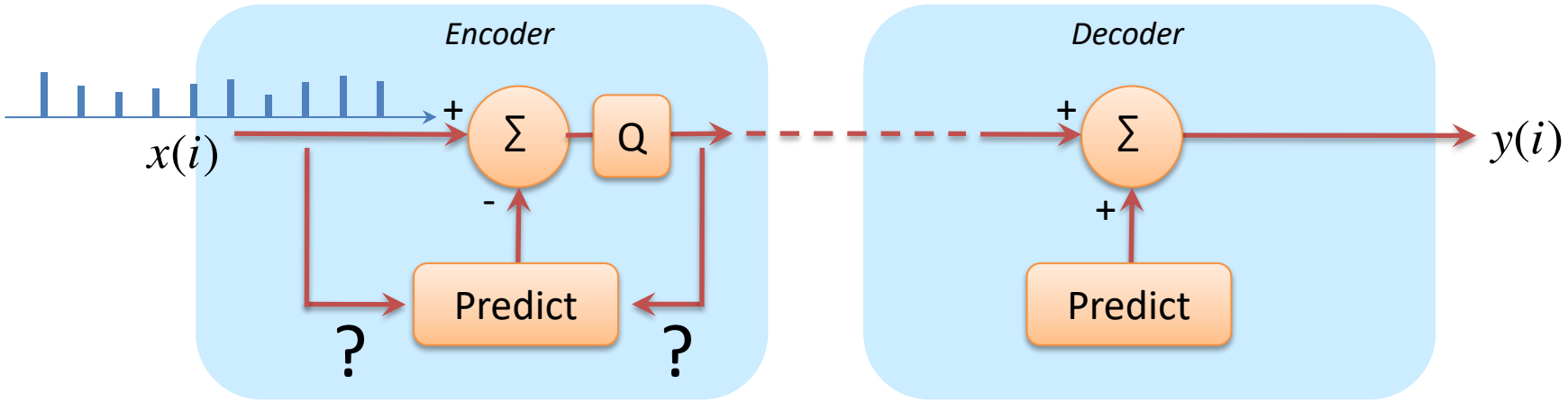
$x_i \in [0,255]$



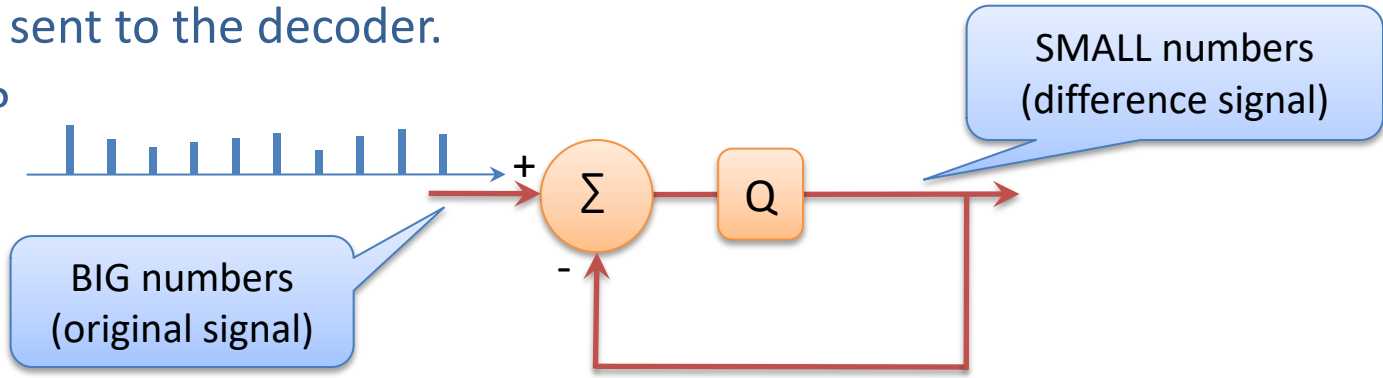
Histogram of prediction errors

$\Delta x_i \in [-255,255]$

How do we generate the prediction?

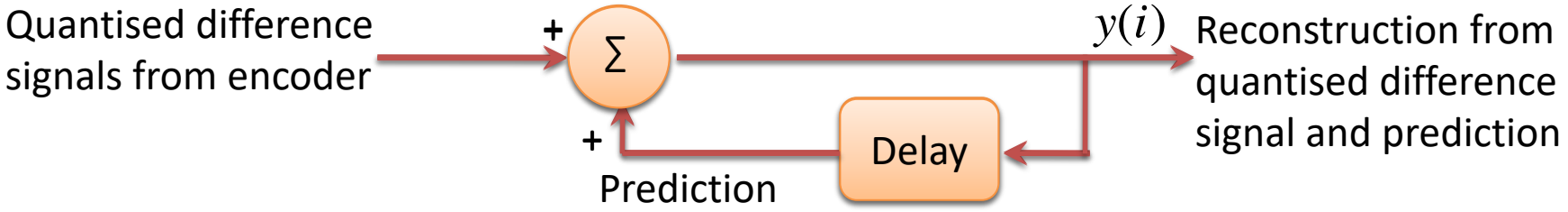


- Encoder and decoder need to be able to make the same prediction
- This wouldn't be a problem except
  - Before sending it, we will QUANTISE the difference signal to save extra bits.
  - The decoder output $y(i)$ is then NOT identical to the encoder input $x(i)$.
- So the encoder must predict incoming values $x(i)$ based on the QUANTISED difference signals sent to the decoder.
- So, will this work?

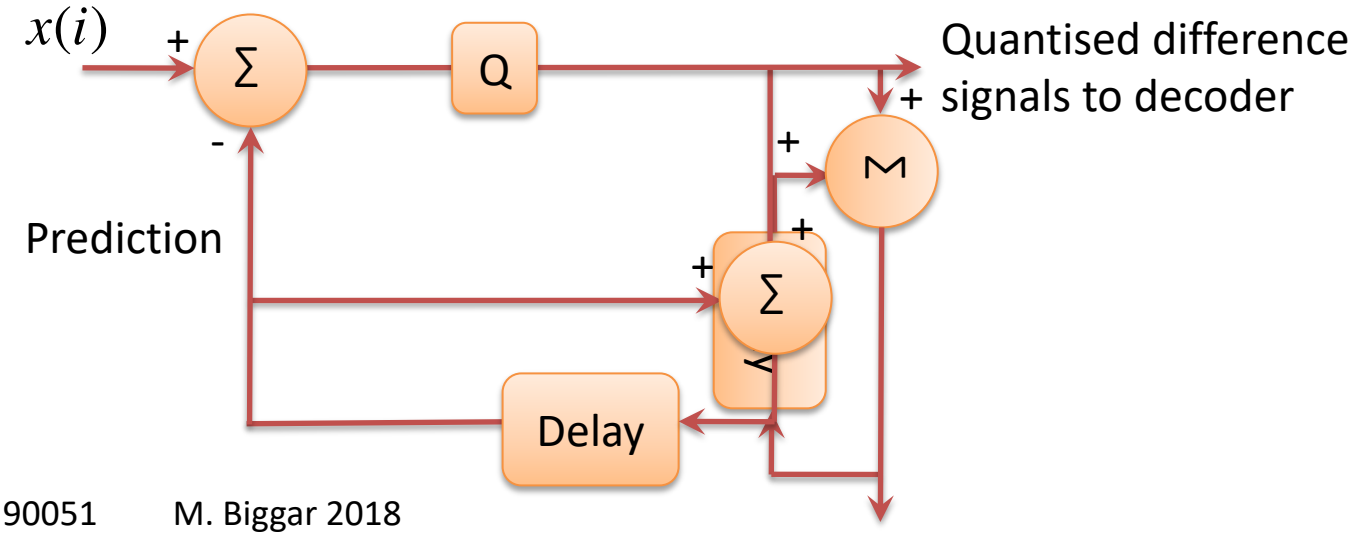


Encoder prediction loop

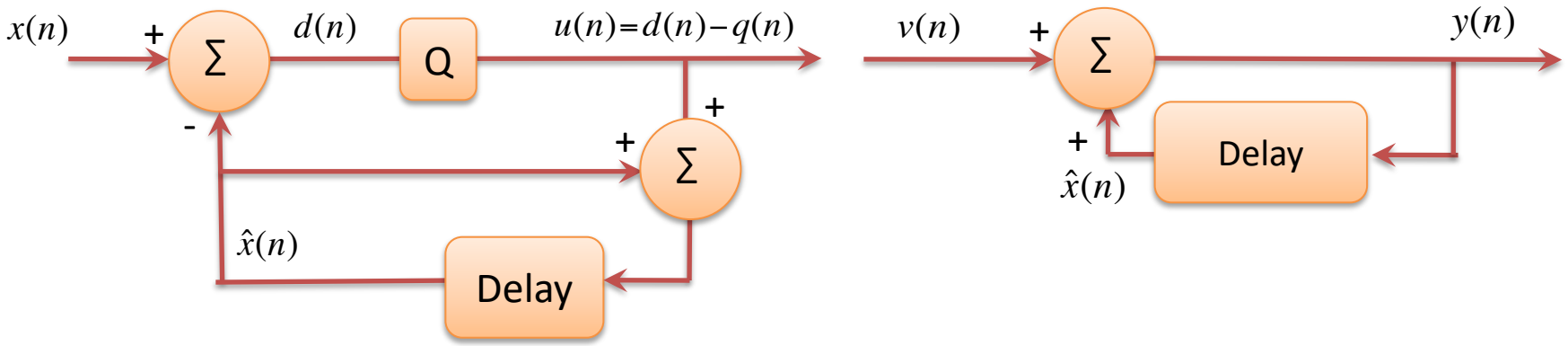
- Consider simple 1-dimensional, 1-sample prediction
- Let's work backwards from the decoder:
  - A reconstructed output value $y(i)$ will be used to predict the next value $y(i+1)$
  - A received (quantised) difference signal will be used to correct the prediction



- At the encoder, we must reproduce this prediction



Formalising some of this…



Encoder difference signal sent to quantiser: $d(n)=x(n)-\hat{x}(n)$

Quantiser output: $u(n)=d(n)-q(n)$

Reconstructed approximation to coder input $x(n)$: $y(n)=\hat{x}(n)+v(n)$

With error-free transmission, $u(n)=v(n)$ and $y(n)=\hat{x}(n)+u(n)$

$$=\hat{x}(n)+d(n)-q(n)$$

$$=x(n)-q(n)$$

Reconstruction error is defined as $r(n)=x(n)-y(n)$

$$=q(n)$$

IMPORTANT feature: Quantisation noise does not accumulate (does not contribute to future reconstruction error)

Tutorial question 1: What changes could we make to the coding architecture to better cope with errors?

SNR advantage of DPCM

With a few simplifying assumptions (quantiser behaviour), the SNR gain of DPCM over PCM is proportional to*:

Uniform distribution, 0..255

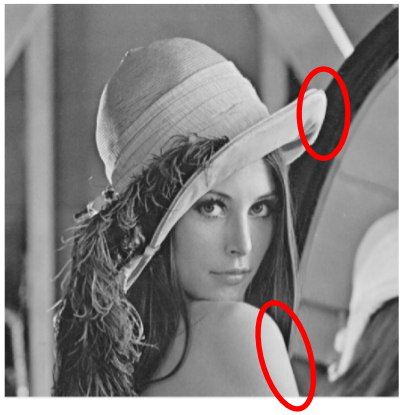
Prediction Gain $$G_P = \frac{\sigma_x^2}{\sigma_d^2}$$

Concentrated close to 0

And $$SNR_{DPCM}\,(dB) = SNR_{PCM}\,(dB) + 10log_{10}(G_P)$$

* See N.S. Jayant & P. Noll, "Digital Coding of Waveforms", Prentice-Hall, 1984, pp257-260.

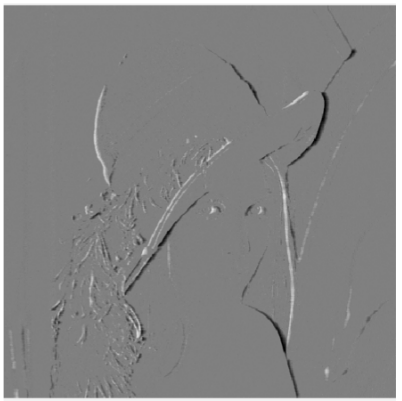
Illustration of DPCM advantage



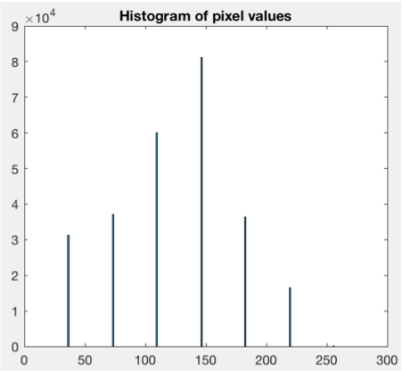
Original

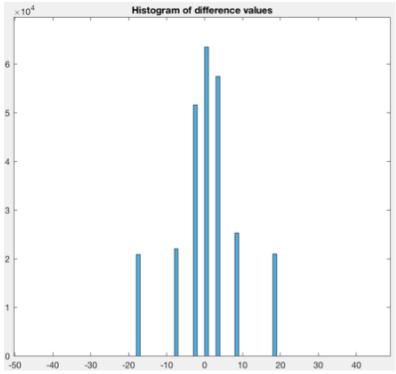
3 bit/pixel PCM

3 bit/pixel DPCM



DPCM error image (shifted by 128)

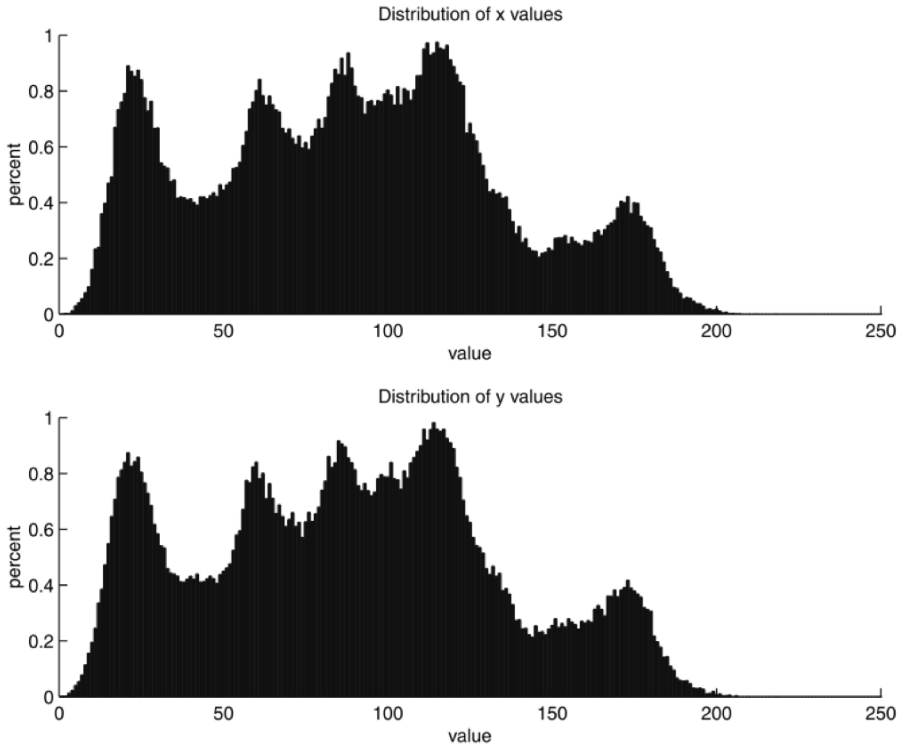




Where would you expect distortion (errors) to occur in the quantised DPCM image?

Back to that adjacent pixel correlation

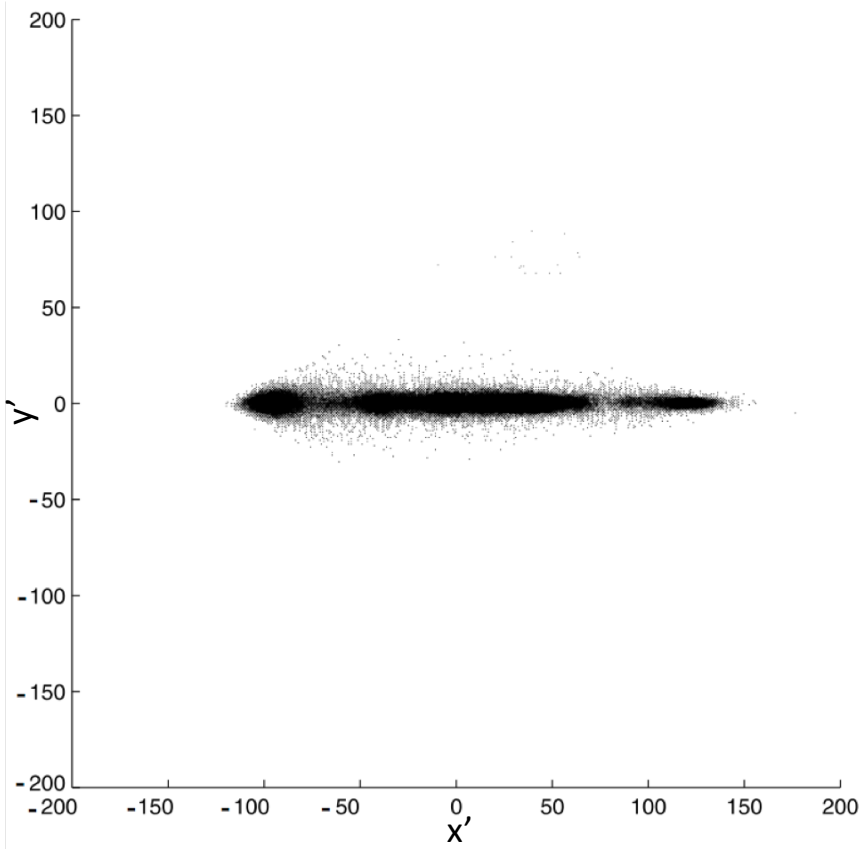
- The histograms of x and y values are similar and each is very similar to that for the set of all pixels in the image.

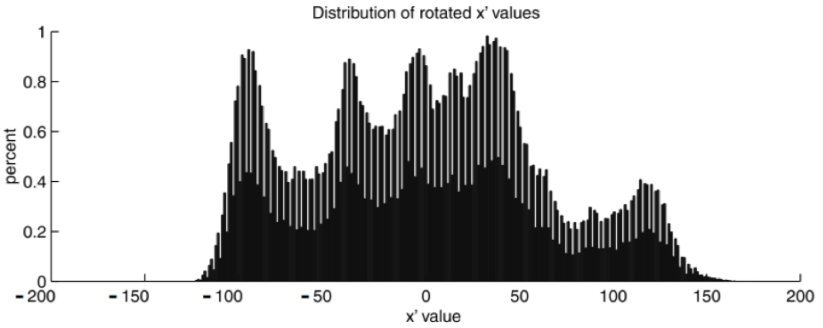


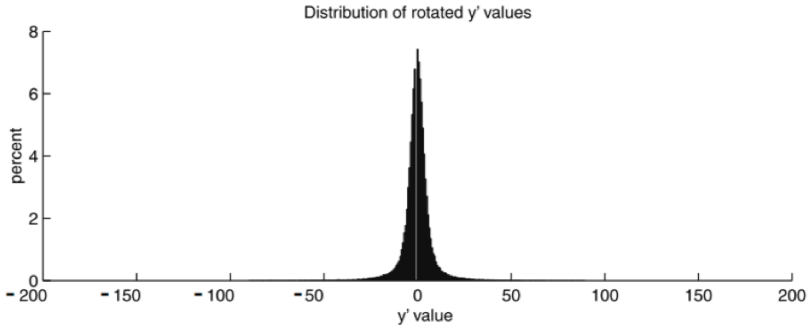
(Source: *The Transform and Data Compression Handbook*, Ed. K. R. Rao and P.C. Yip., Boca Raton, CRC Press LLC, 2001)

A data transformation

- What would happen if we transformed our (x,y) pairs (somehow), so that the scatter plot was rotated 45°?







(Source: *The Transform and Data Compression Handbook*, Ed. K. R. Rao and P.C. Yip., Boca Raton, CRC Press LLC, 2001)

Measuring correlation: Covariance

If zero mean random variables x_i, x_j are decorrelated, then the covariance is 0 if $i \neq j$:

$$E(x_i x_j) = \begin{cases} 0 & i \neq j \\ \sigma_i^2 & i = j \end{cases}$$

$E(.)$ is the Expectation operator, σ^2 is variance

We can generalise for an image block of N pixels using vector notation:

$$\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; \ldots \;\; x_N]^T$$

The covariance matrix is

$$[\mathbf{C}]_x = E\left[(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T\right]$$

Where $\mathbf{m} = E(\mathbf{x})$ is the mean. (We will now assume zero mean without loss of generality.)

Now, if the x_i are uncorrelated, then $[\mathbf{C}]$ is a diagonal matrix.

(λ is the variance)

$$[\mathbf{C}]_x = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & . & \\ & & & . \\ 0 & & & \lambda_N \end{bmatrix}$$

The ideal decorrelating transform

We would like to find the transform $\mathbf{[T]}$ to transform $\mathbf{x}$ into $\mathbf{y}$:

$$\mathbf{y} = \mathbf{T}\,\mathbf{x}$$

Such that covariance matrix $\mathbf{[C]}_y$ is diagonal.

$$[\mathbf{C}]_y = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_N \end{bmatrix}$$

The Transform which satisfies this is called the KLT (Karhunen-Loève Transform), or the Hotelling Transform.

However, its use is problematic:

- It is very complex to calculate
- There are no fast implementations
- The transform itself is signal-dependent, so should be communicated to the receiver

A simplifying model

- We can model our image data as a first-order Markov source with interelement correlation ρ $(0 \le \rho \le 1)$

$$[\mathbf{C}] = \begin{bmatrix} 1 & \rho & \rho^2 & & \rho^N \\ \rho & 1 & \rho & & \\ \rho^2 & \rho & . & \rho & \\ & & \rho & . & \rho \\ \rho^N & & & \rho & 1 \end{bmatrix}$$

- Then… I C B S

  that, as ρ -> 1, the transform that decorrelates the data has basis functions:

$$K(0,t) = \frac{1}{\sqrt{N}} \qquad 0 \le t \le N-1$$

$$K(n,t) = \sqrt{\frac{2}{N}} \cos\left(\frac{n\pi(2t+1)}{2N}\right) \qquad 0 \le t \le N-1$$

- This is the **Discrete Cosine Transform** (DCT)

The DCT

- The 1-D DCT of a function f in x is defined as

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos(((2x+1)u\pi)/2N) \qquad for\ u = 0,1,2,...,N-1$$

- And the inverse DCT is

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) \cos(((2x+1)u\pi)/2N) \qquad for\ x = 0,1,2,...,N-1$$

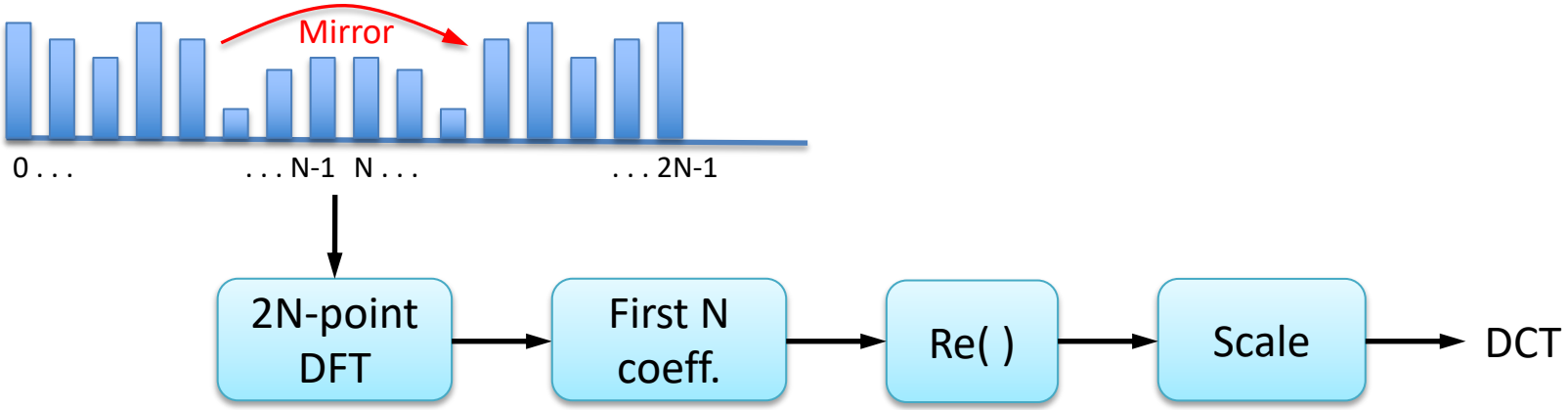
- Where:

$$\alpha(u) = \begin{cases} \sqrt{1/N} & for\ u = 0 \\ \sqrt{(2/N)} & for\ u = 1,2,....,N-1 \end{cases}$$

- Fast algorithms exist (similar to the FFT)
- All calculations are REAL (no complex arithmetic)

DCT relationship with DFT

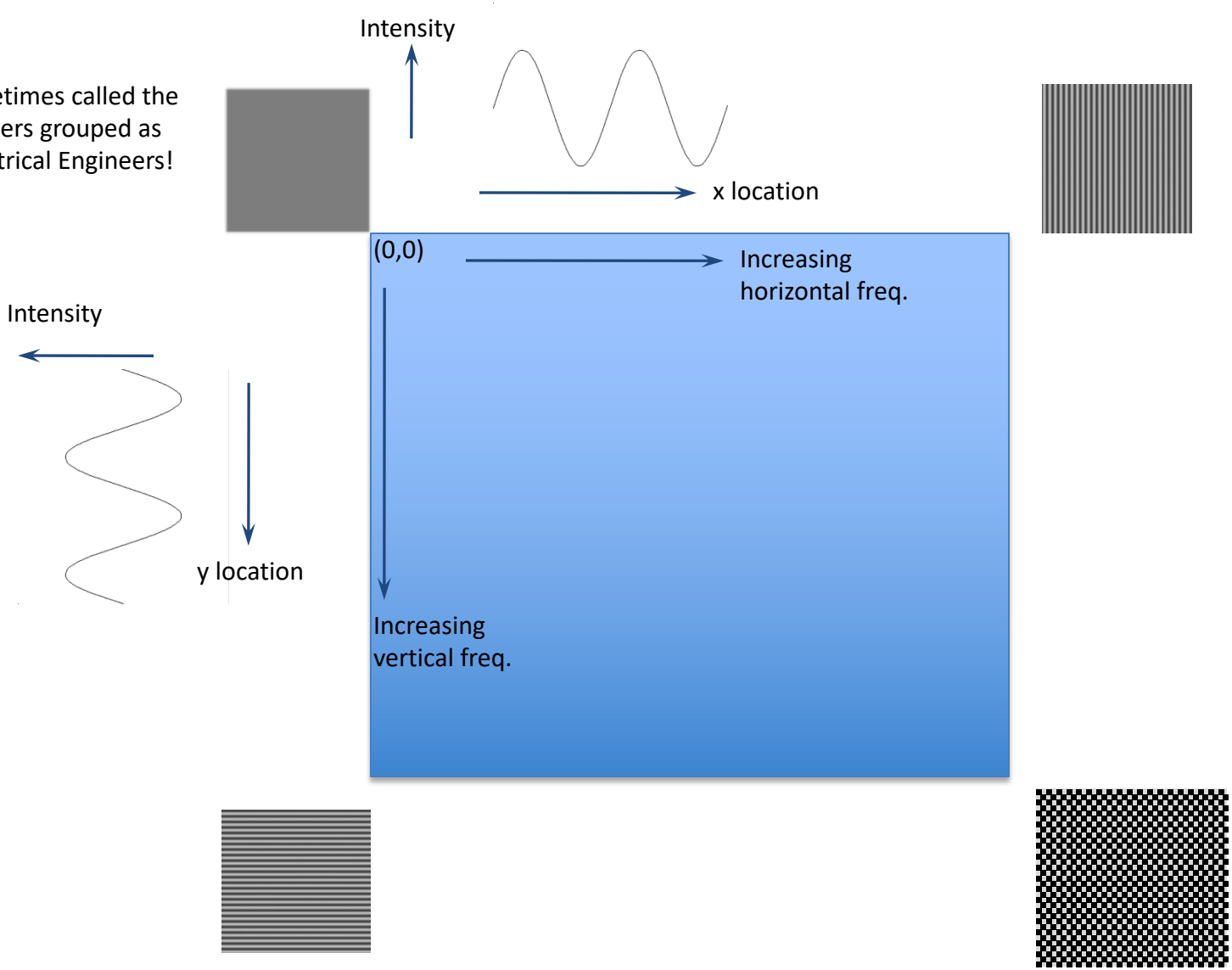
- Recall that:
  - if a DFT (Discrete Fourier Transform) has only real coefficients (cosine terms only),
  - then the function must be symmetrical in the time (or space) domain
- This gives a clue about a relationship between the DCT and the DFT



(For details, see Pennebaker & Mitchell, "JPEG Still Image Compression Standard", Kluwer Academic Publishers, 1993, pp49-50,

2D spatial frequency representations

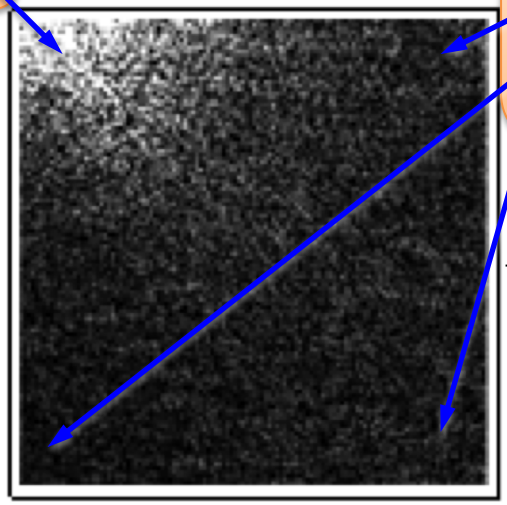
(0,0) term is sometimes called the "DC term" (all others grouped as "AC"). Blame Electrical Engineers!



Transform coding

- Now we think about actually encoding the signal in the frequency/transform domain

Transform



Low frequencies:
Large features

High frequencies:
Fine details -
- Horizontal
- Vertical
- Diagonal

Encode

Note: To visualise 2D transforms, we represent them as intensity arrays; white means large magnitude, black means near zero.

(Transform image: http://reference.wolfram.com/legacy/applications/digitalimage/UsersGuide/ImageTransforms/8.4.html)

Formulation of DCT

- 1-D case

The 1-D discrete cosine transform (DCT) is defined as

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos(((2x+1)u\pi)/2N)$$

for *u*=0,1,2, ..., *N*-1. The inverse DCT (IDCT) is given by:

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) cos(((2x+1)u\pi)/2N)$$

for *x*=0,1,2, ..., *N*-1. Where

$$\alpha(u) = \begin{cases} \sqrt{(1/N)} & \text{for } u = 0 \\ \sqrt{(2/N)} & \text{for } u = 1, 2, ..., N-1 \end{cases}$$

2-D DCT

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

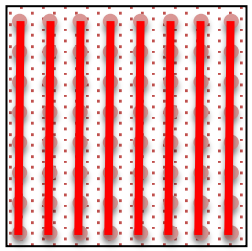
$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

- Looks like a lot of calculations! (N^2-point weighted sum for every output value; number of op's proportional to N^4)
- But (fortunately!) the 2D DCT is separable

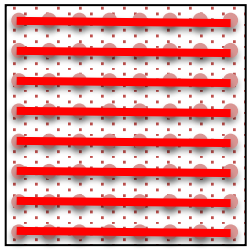
Separability

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

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



1-D transforms on the columns



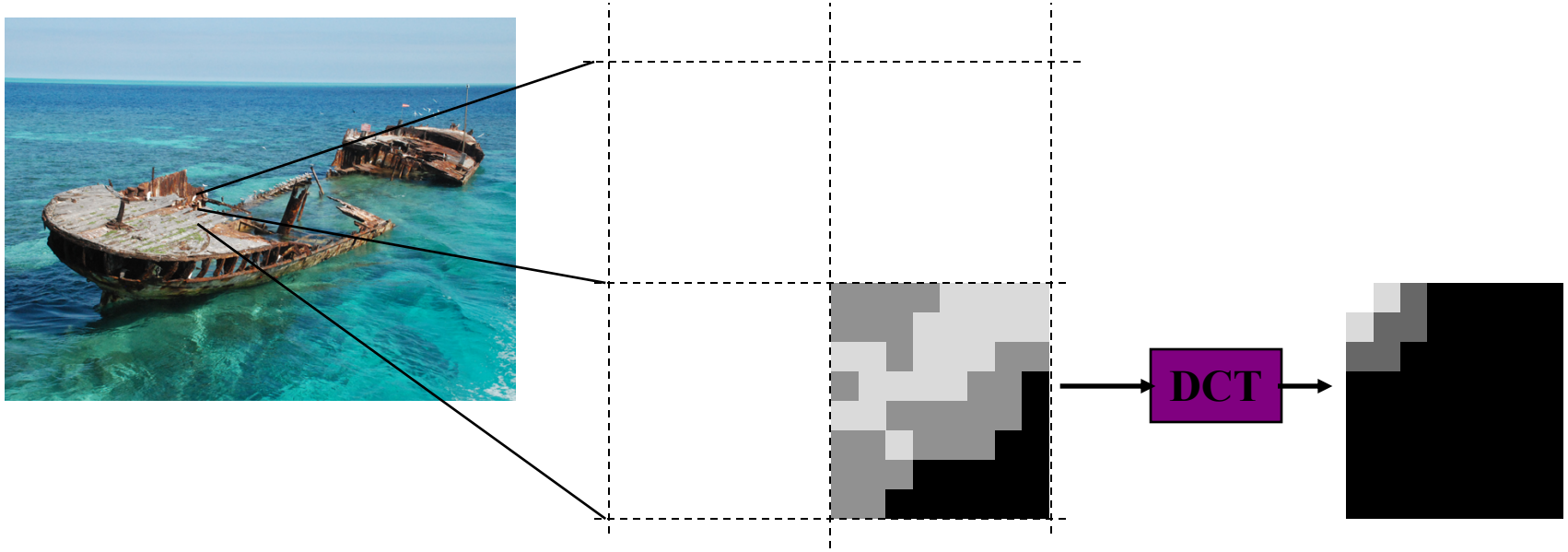
1-D transforms on the rows

i.e. apply 1-D transform to all the columns, then all the rows

(Number of op's proportional to N^3. Fast algorithms bring this down to $N^2\log(N)$)

Block based 2-D transform coding

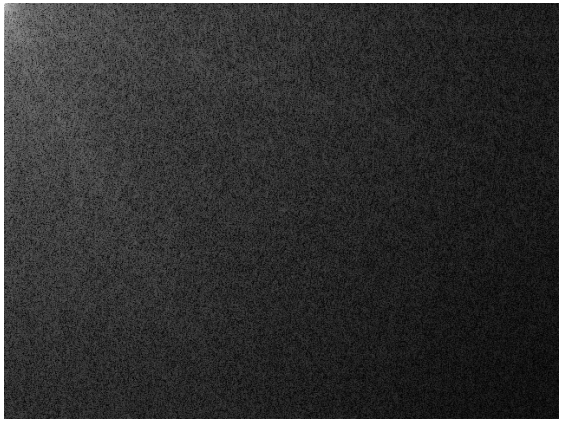
- Transform blocks of the image (instead of the whole image)
  - Computational savings
  - Local adaptivity
- Typical block sizes: 4x4, 8x8 and 16x16 pixels



2D DCT of a whole image

Extract Y component

Single 2D DCT



640x480 image with 640x480 DCT

"Heron Island Wreck" photo: M. Biggar

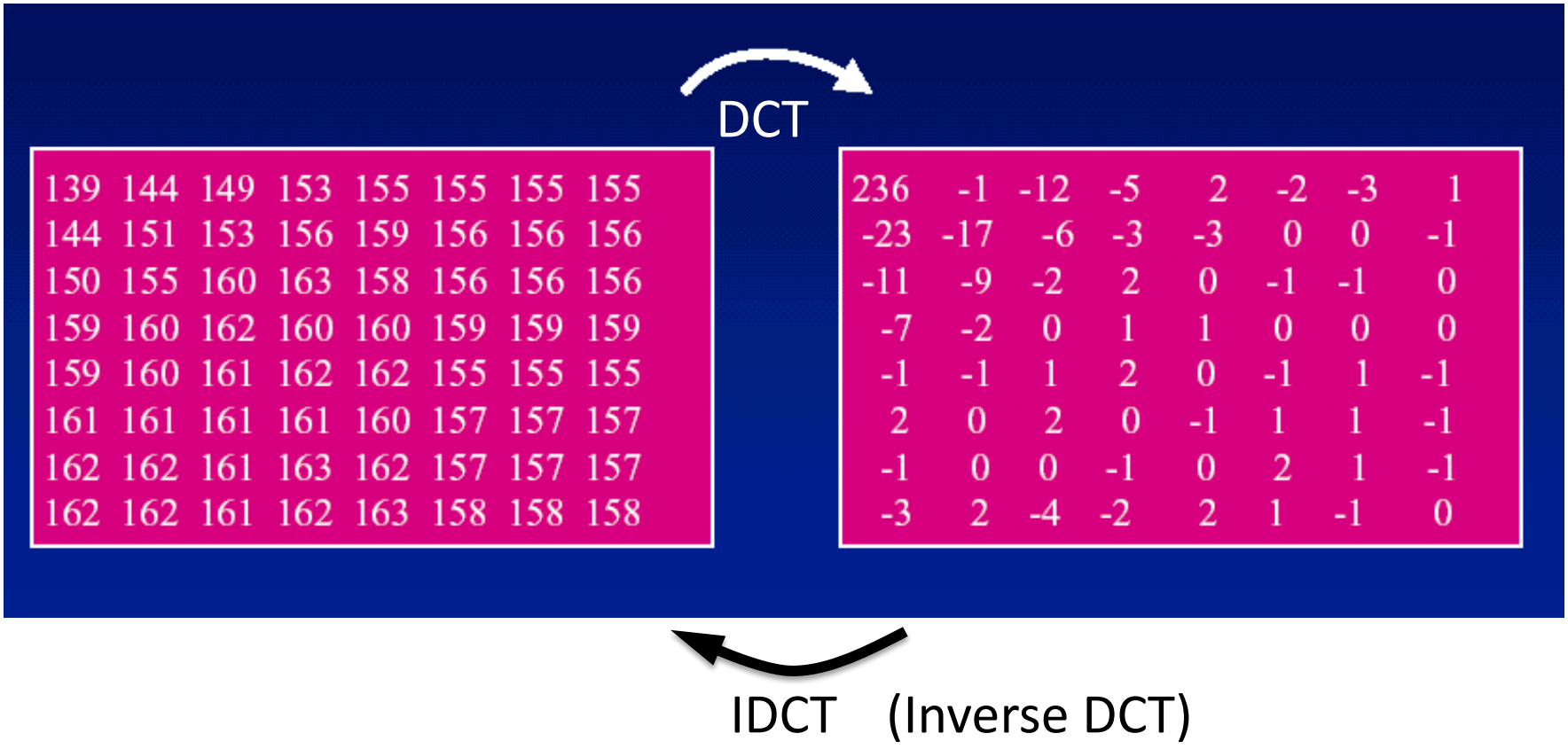
Adaptation to local characteristics

4 x 4

640x480 image segmented with smaller DCT blocks

"Heron Island Wreck" photo: M. Biggar

2-D DCT example

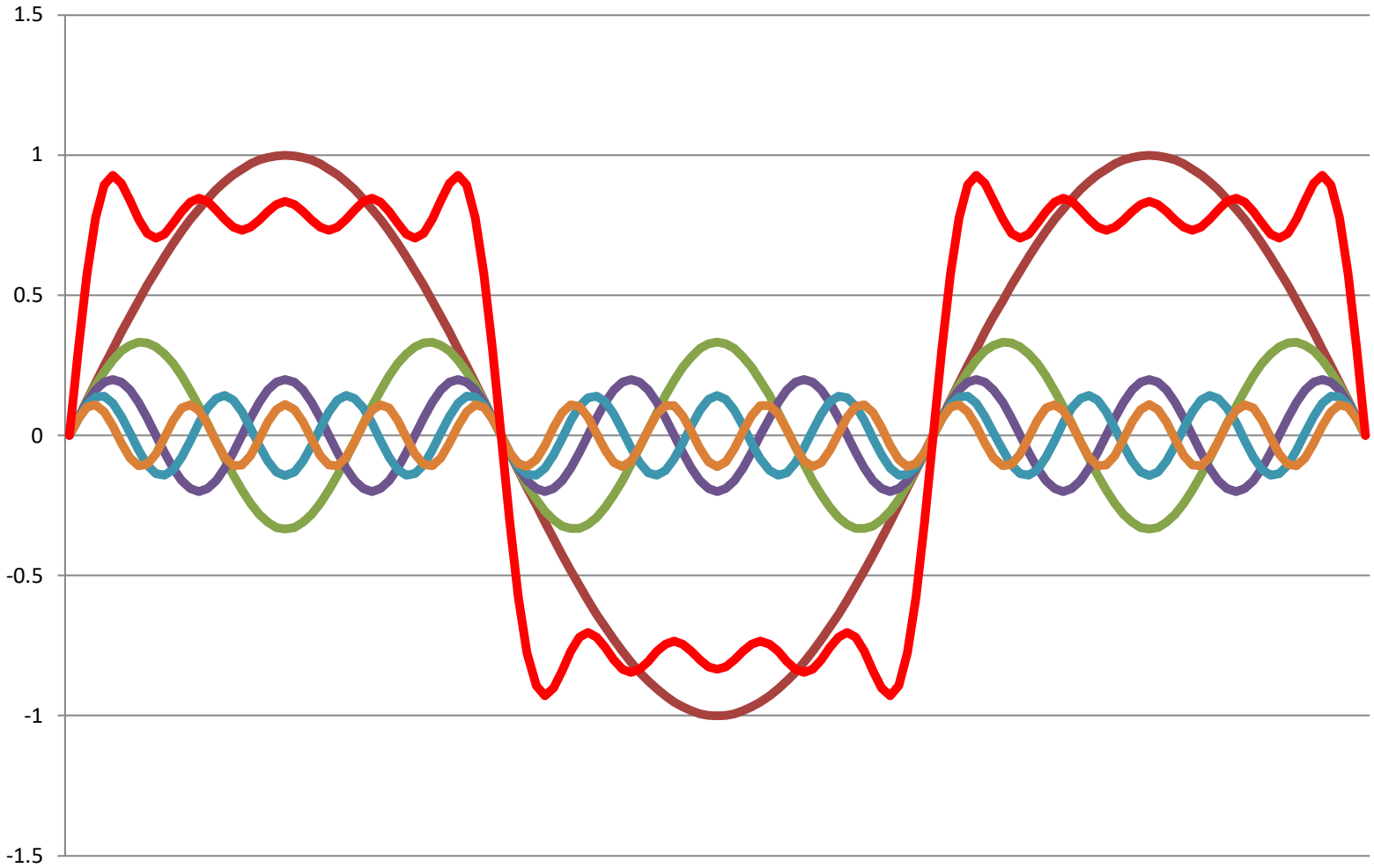


(Courtesy Prof. Fernando Pereira, Istituto Superior Tecnico, Lisbon)

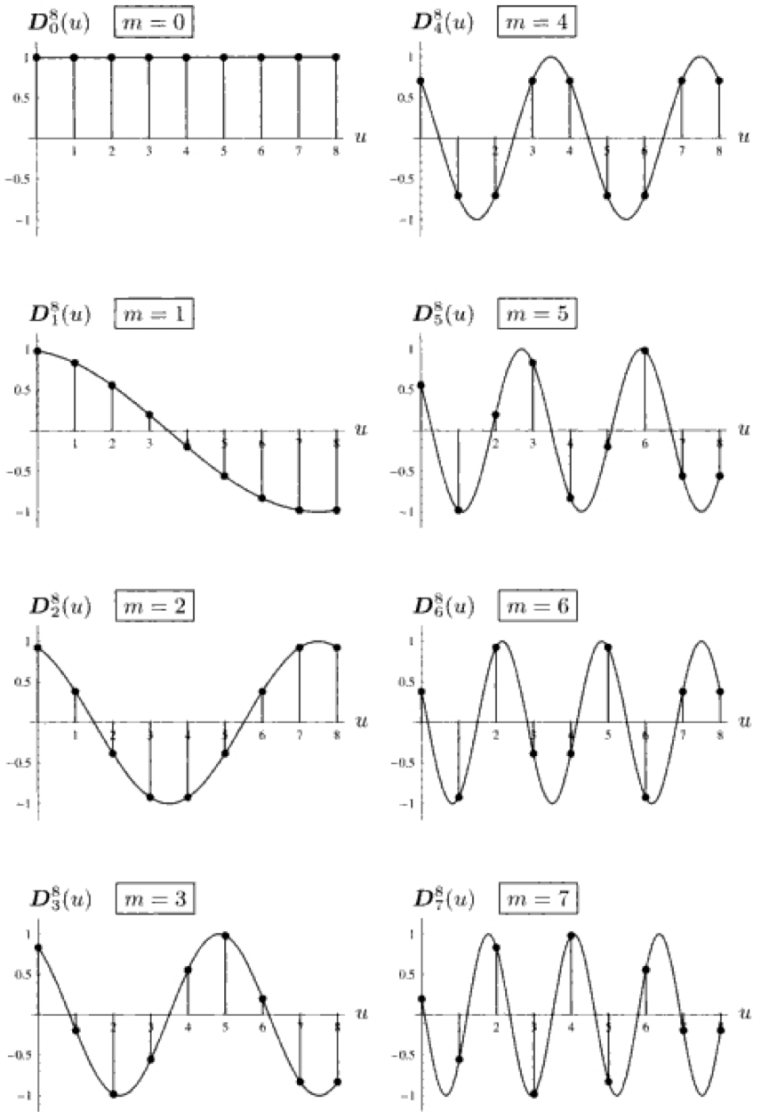
DCT tool “DCT.xls” (available via LMS)

- Play around with the numbers to see what it does.
- Put single numbers in the transform domain array to look at the basis functions
- Investigate how you can quantise or even eliminate coefficients without drastic effects in the image domain

1-D Fourier transform basis functions

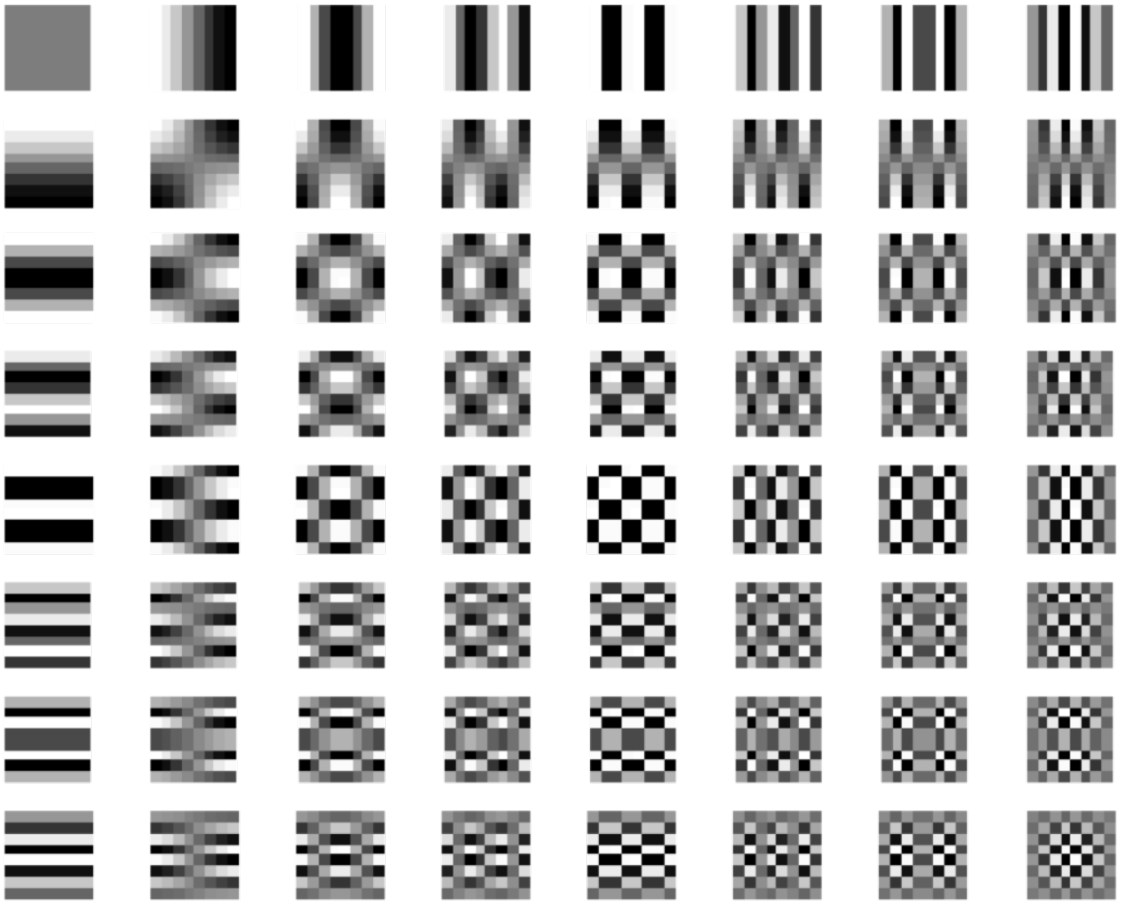


1-D 8-point DCT basis functions



(Source: http://bjl2009.blogspot.com.au/2009/07/discrete-cosine-transformdct.html)

2-D DCT 8x8 basis functions



(From http://www-mtl.mit.edu/Courses/6.095/notes/dct.html)

DCT representation

3 ways to represent pixel values

DCT

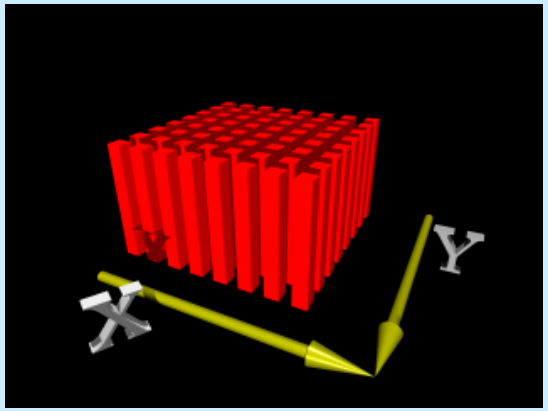
700	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

PIXELS

X

Y

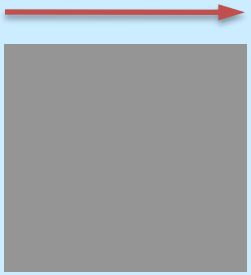
87	87	87	87	87	87	87	87
87	87	87	87	87	87	87	87
87	87	87	87	87	87	87	87
87	87	87	87	87	87	87	87
87	87	87	87	87	87	87	87
87	87	87	87	87	87	87	87
87	87	87	87	87	87	87	87
87	87	87	87	87	87	87	87



700 x

X

Y



(Adapted from Emblaze Corp. mobile video slide pack)

DCT representation (cont.)

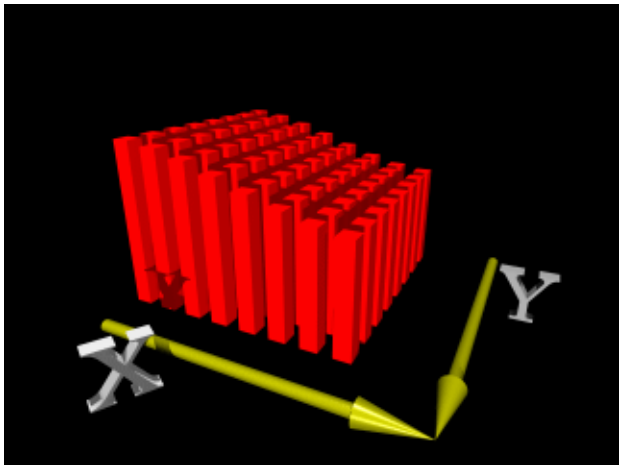
Now let's add a non-zero frequency value of 100

700	100	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	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

The result of the IDCT is

105	102	97	91	84	78	73	70
105	102	97	91	84	78	73	70
105	102	97	91	84	78	73	70
105	102	97	91	84	78	73	70
105	102	97	91	84	78	73	70
105	102	97	91	84	78	73	70
105	102	97	91	84	78	73	70
105	102	97	91	84	78	73	70

And the bar diagram looks like



700 x  + 100 x 

(Adapted from Emblaze Corp. mobile video slide pack)

DCT representation (cont.)

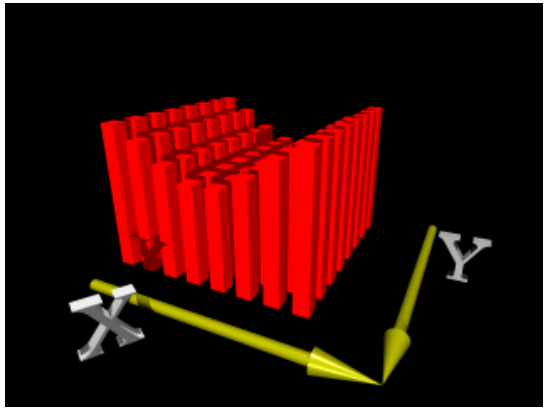
Now let's consider what happens if we place the coefficient value of 100 at the next position

700	0	100	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	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

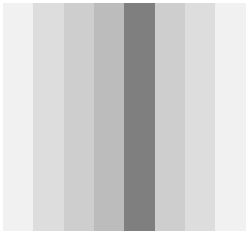
The result of the IDCT is

104	94	81	71	71	81	94	104
104	94	81	71	71	81	94	104
104	94	81	71	71	81	94	104
104	94	81	71	71	81	94	104
104	94	81	71	71	81	94	104
104	94	81	71	71	81	94	104
104	94	81	71	71	81	94	104
104	94	81	71	71	81	94	104

And the bar diagram looks like



700 x  + 100 x 



(Adapted from Emblaze Corp. mobile video slide pack)

Removing “irrelevance”

The transformations we have spoken of

- Predictive coding (e.g. DPCM)
- Transform coding

Have the effect of

- Changing the distribution of numbers representing the input signal (the histogram)
- Reorganising them such that all the big ones are together, all the small ones are together, etc.

But they don’t change the number of data points to be communicated!

How can we reduce the number of bits to be transmitted?

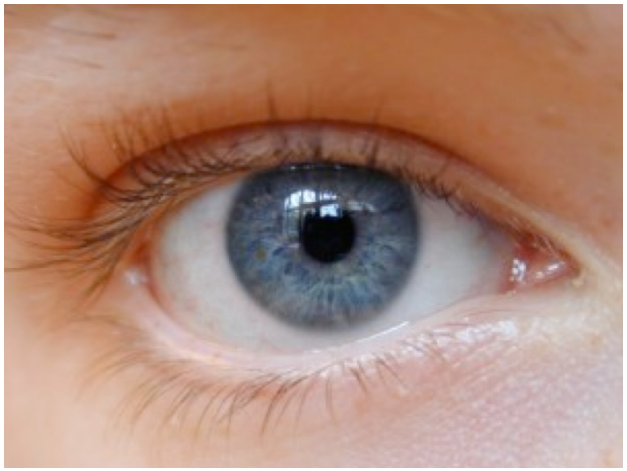
“Irrelevant” content

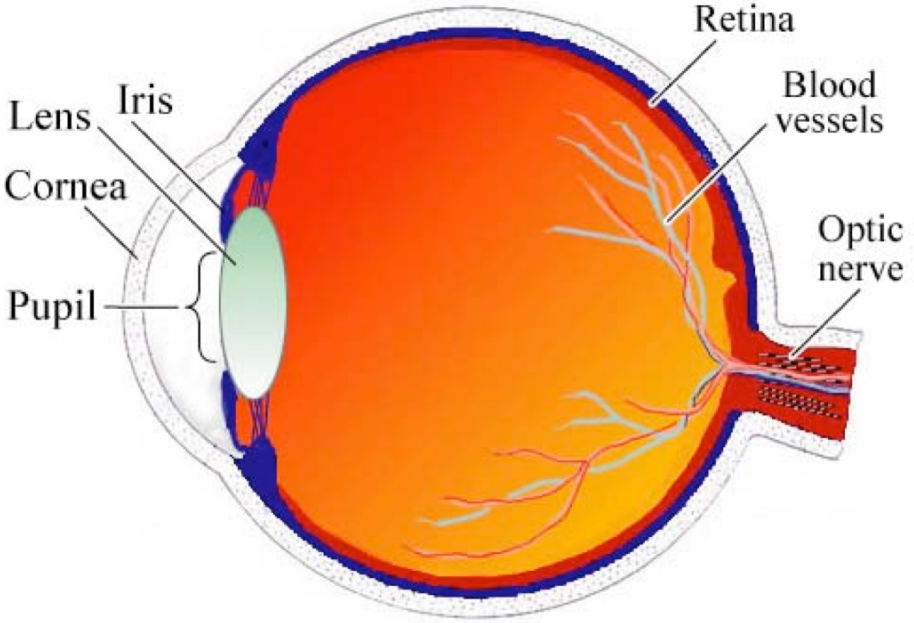
- The transformations we have spoken of
  - Predictive coding (e.g. DPCM)
  - Transform coding
- Have the effect of
  - Changing the distribution of numbers representing the input signal (the histogram)
  - Reorganising them such that all the big ones are together, all the small ones are together, etc.
- But they don’t change the number of data points to be communicated!
  - We get some statistical advantage when we assign bit patterns to symbols (e.g. Huffman Coding)
  - But we’ve still got one data point per pixel to transmit

Is all of that data really important?

How do we decide what is “irrelevant”?

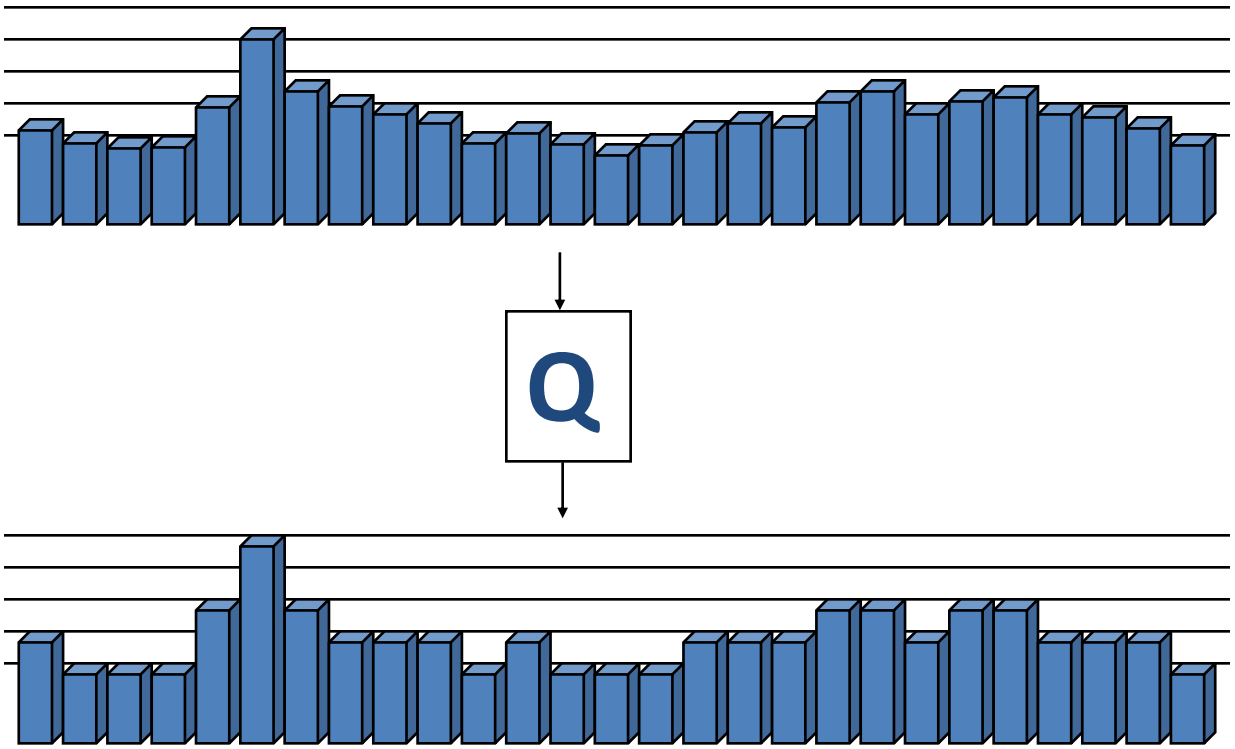
- Need to take account of the ultimate receiver of the image
- Characteristics of the human eye and eye/brain system





(Eye image from http://topouest.com/human-eye-the-one-that-shows-us-the-world/
Graphic from http://www.ecse.rpi.edu/~schubert/Light-Emitting-Diodes-dot-org/chap16/chap16.htm)

Quantisation



Only now is the coding lossy

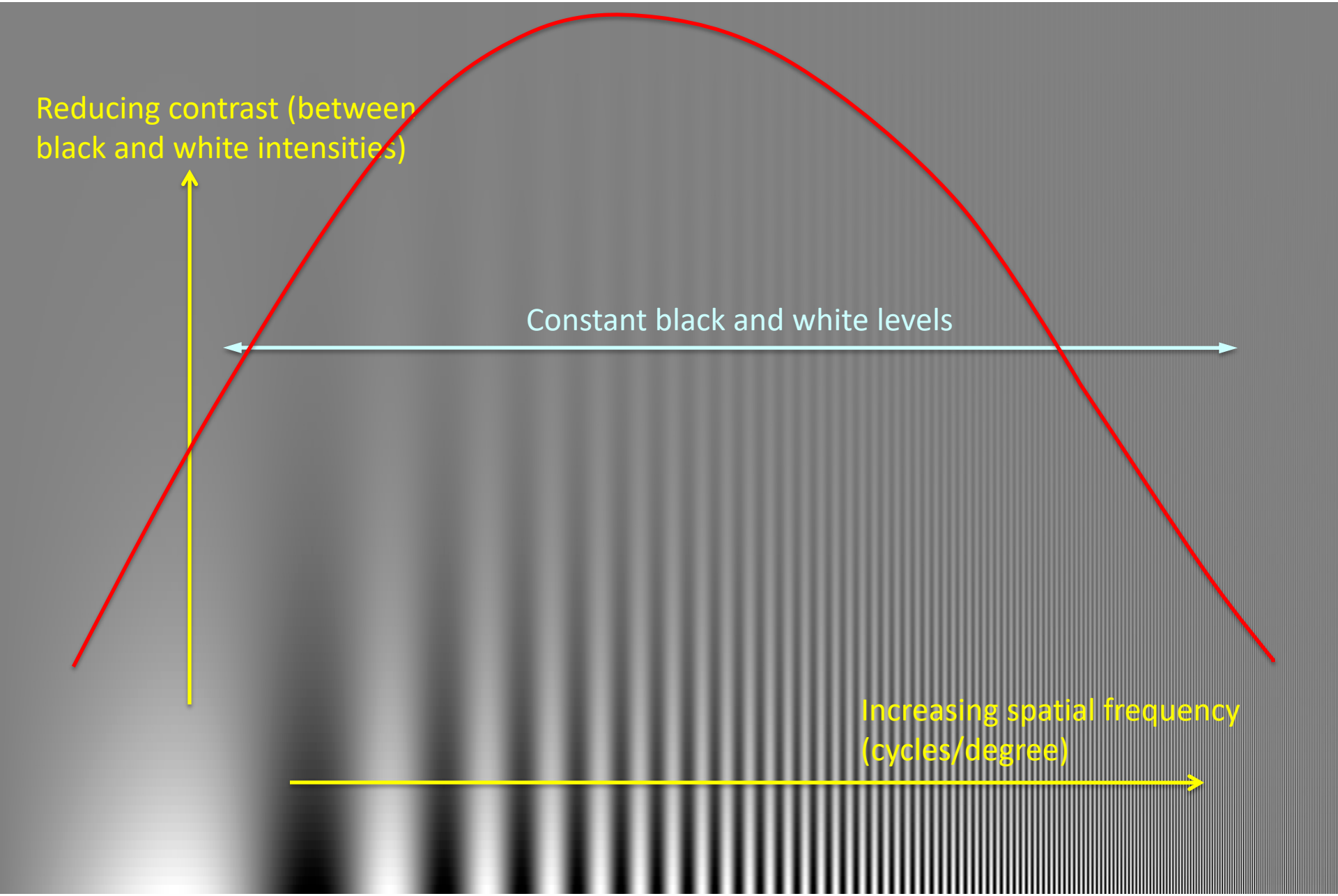
Lossless and Lossy compression

- If there is no quantisation, the compression MUST BE lossless
  - A lossless coder is completely reversible
  - You could encode and decode a million times, and still have EXACTLY the same signal
  - Examples:
    - Image: bmp
    - Data: zip
    - Audio: “FLAC” (Free Lossless Audio Coder)
- A Lossless coder can remove REDUNDANT information, but NOT “irrelevant” information.
- When would you choose Lossless coding?
- And when would you choose Lossy?

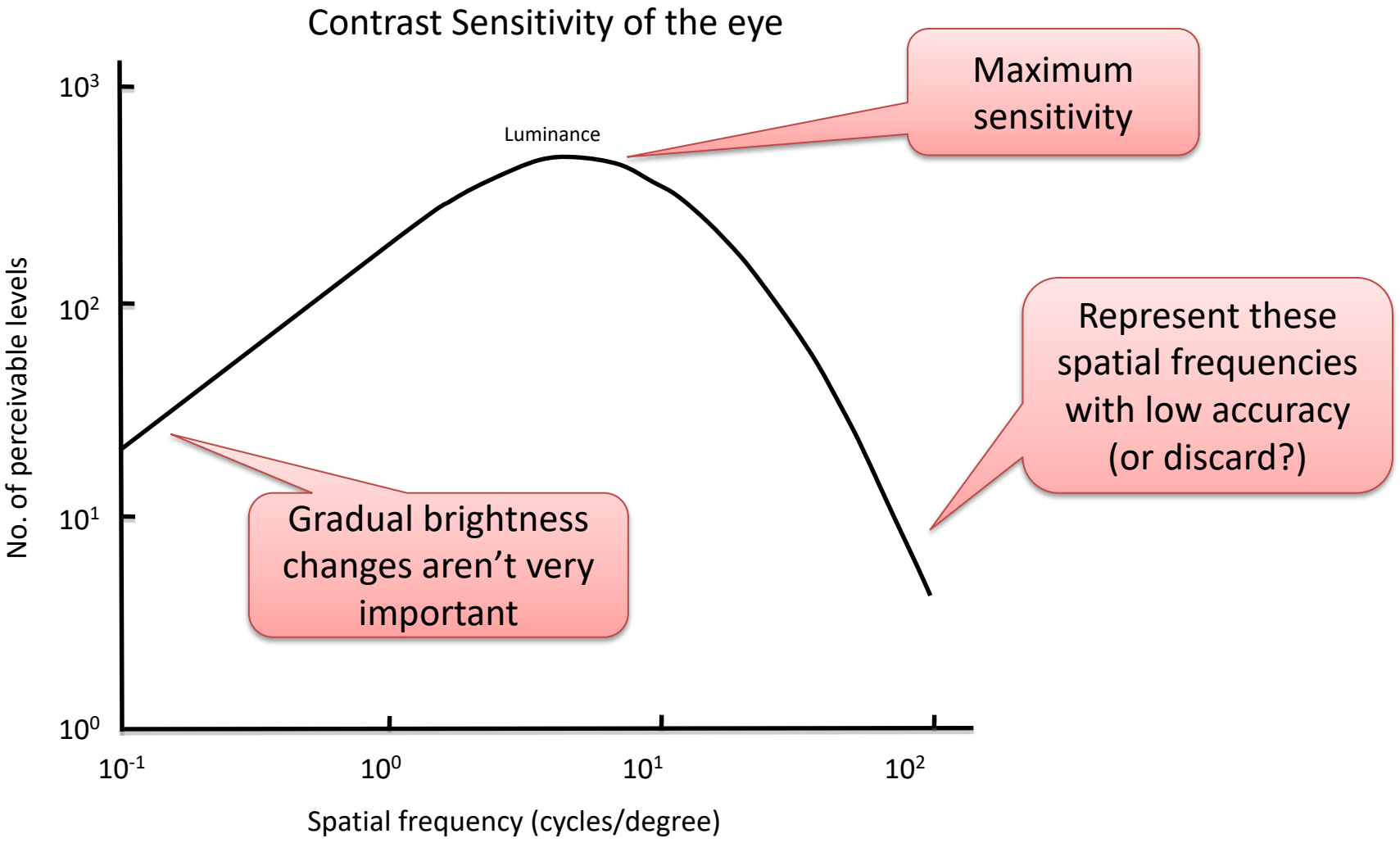
Quantising transform-domain coefficients

- Our DCT gives us a representation of the image in terms of spatial frequencies
- Should all spatial frequencies be quantised the same?

Campbell-Robson CSF (Contrast Sensitivity Function) chart



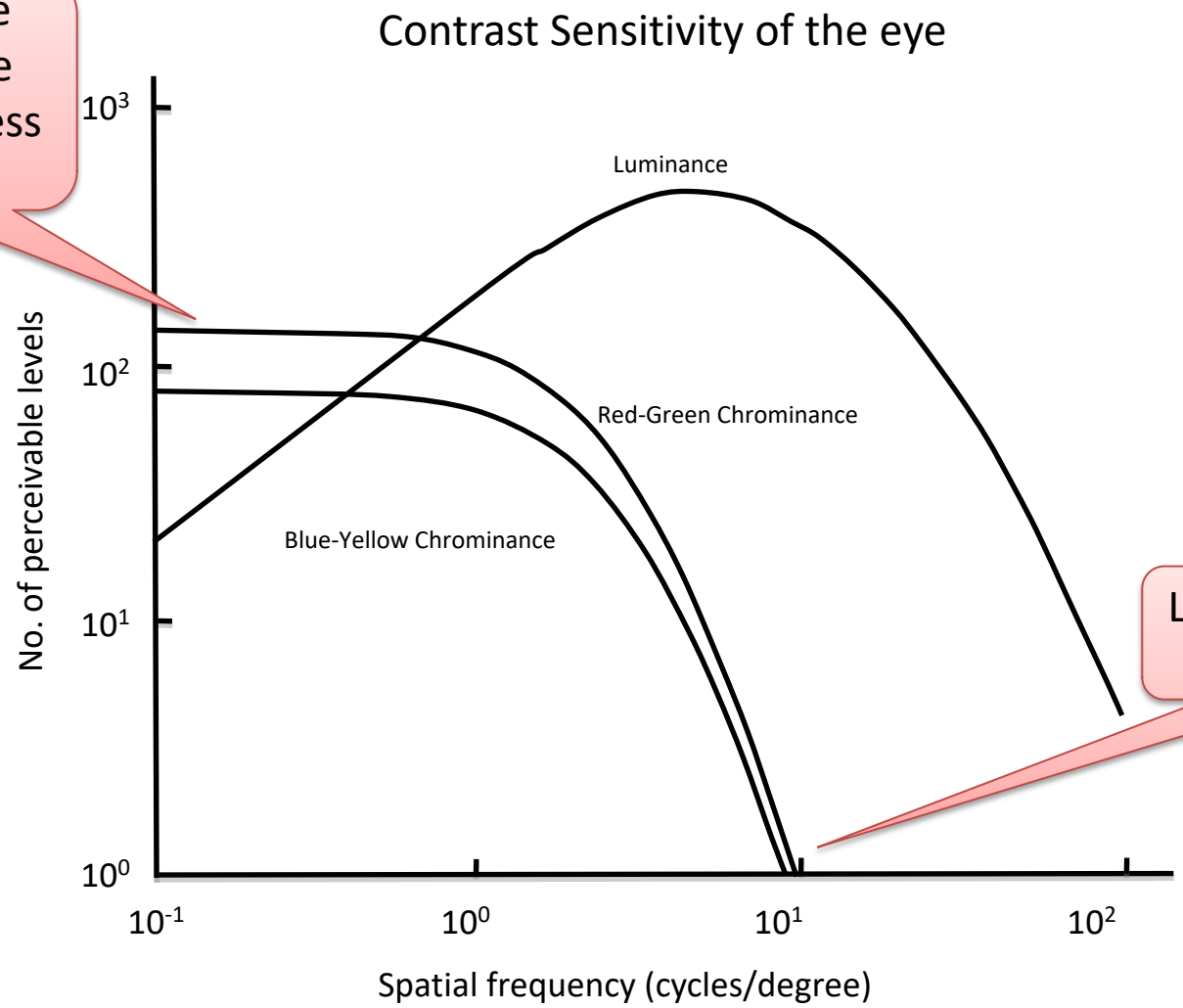
How many levels of intensity do we need?



Plot reproduced from http://cnx.org/content/m11084/latest/#sec3

Spatial frequency sensitivity of the eye

Chrominance values can be represented less accurately



Less spatial detail is needed

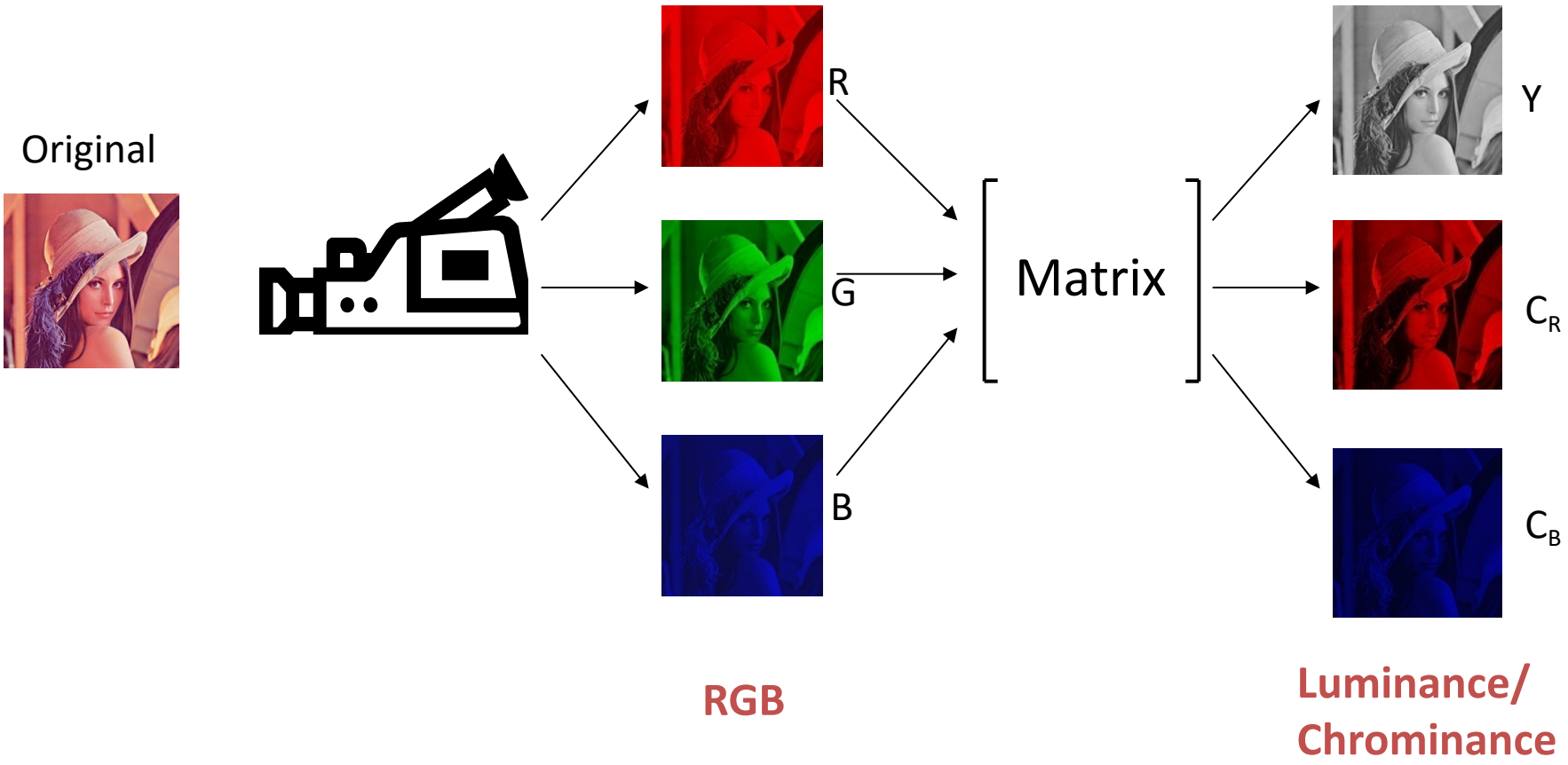
Plot reproduced from http://cnx.org/content/m11084/latest/#sec3

Illustration of colour insensitivity

Same blur applied to chrominance only

(Images: TJ Dennis, Essex University)

Colour image representations



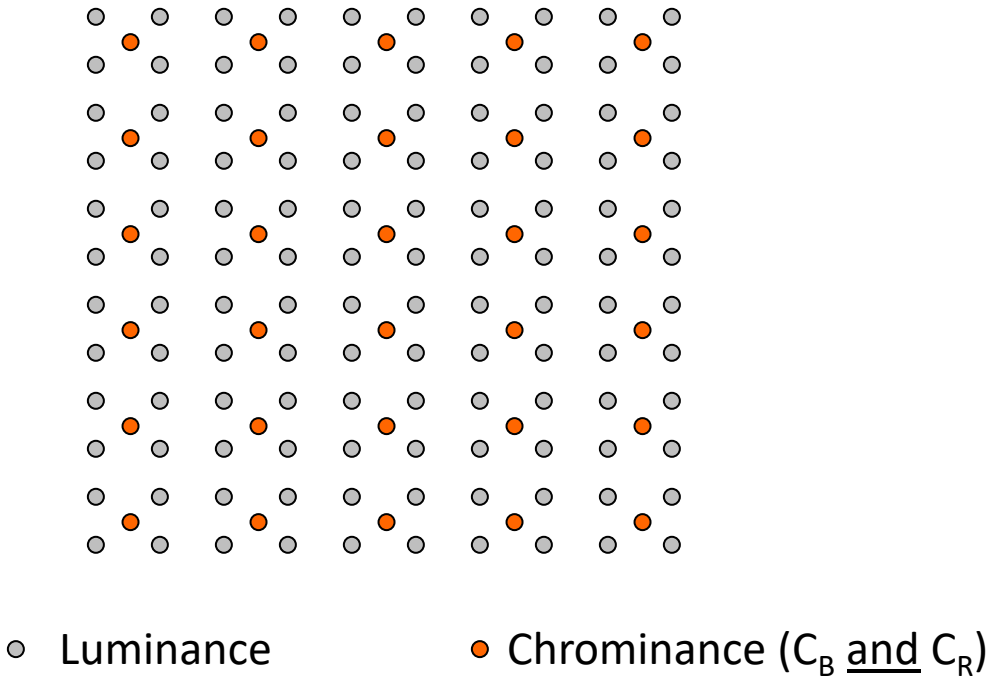
Information in each component

If we represent each component as a grey-scale image, we get this:

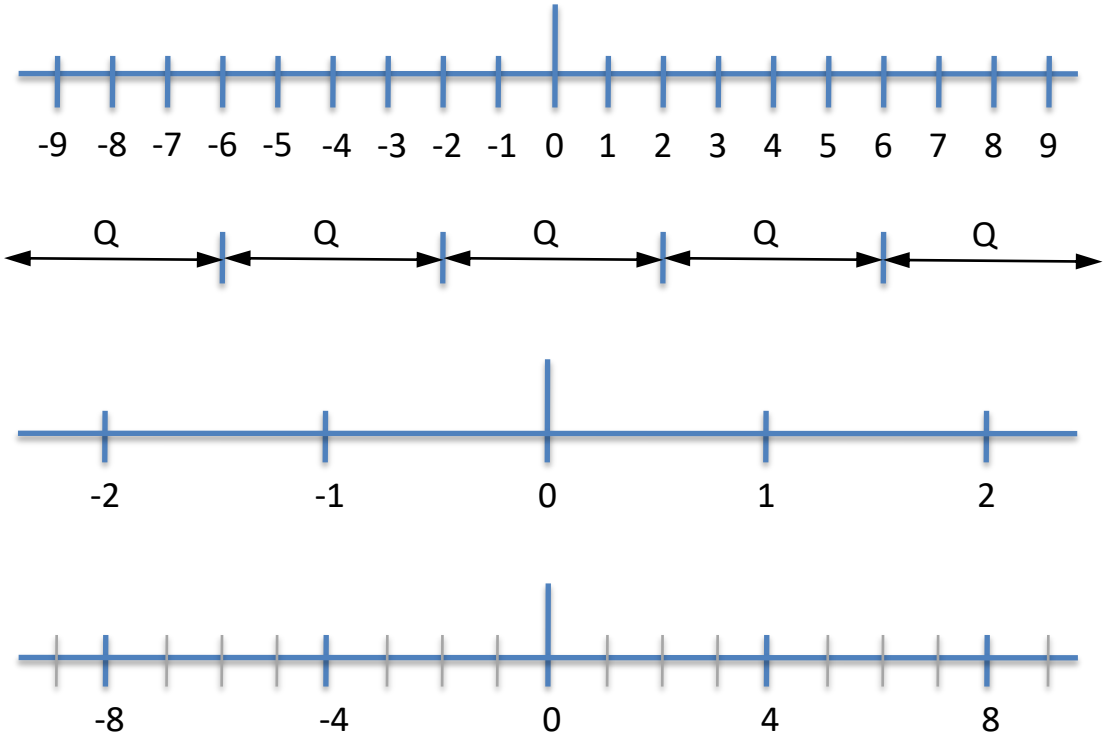
(Image from http://www.nattress.com/Chroma_Investigation/chromasampling.htm)

Sampling density – colour subsampling

- Chrominance is subsampled 2:1 in each direction



Quantisation of DCT coefficients (in image coding)



Unquantised DCT coefficients

Quantisation interval

Quantised DCT coefficients

Dequantised, or "inverse quantised" DCT coefficients

- Quantisation interval is defined per coefficient for luminance and chrominance data

JPEG quantisation tables

- Quantisation tables are sent with the coded data
- Each coder can choose which tables to use
- JPEG standard defines one set of tables for information (not mandatory)
- Based on human visual system sensitivity tests
- Threshold of sensitivity under particular resolution and viewing conditions
- In general, better (smaller quantisation level) tables should be used
- Many lazy programmers just use the standard tables
- Adobe uses their own (Photoshop, Lightroom,…)
- Cameras normally calculate them on the fly, based on the image characteristics (and manufacturers patent algorithms).

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	36	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Luminance

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Chrominance

Tutorial question 2: Under what conditions are these quantisation tables optimal, and under what conditions should they be changed?

Combining the coding tools in JPEG

Putting them together, we can try to remove that which is

- Redundant
- Irrelevant

in an image.

REDUNDANT: Not <u>needed</u> to represent the picture accurately
IRRELEVANT: Not <u>perceivable</u> by the viewer/listener

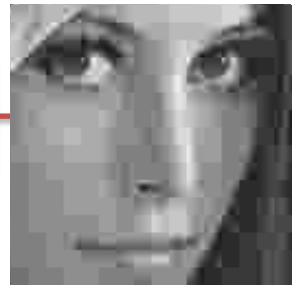
Image compression – basic process

- Encoder:

Block segmentation → DCT → Q → Codeword assignment (Huffman) → To decoder

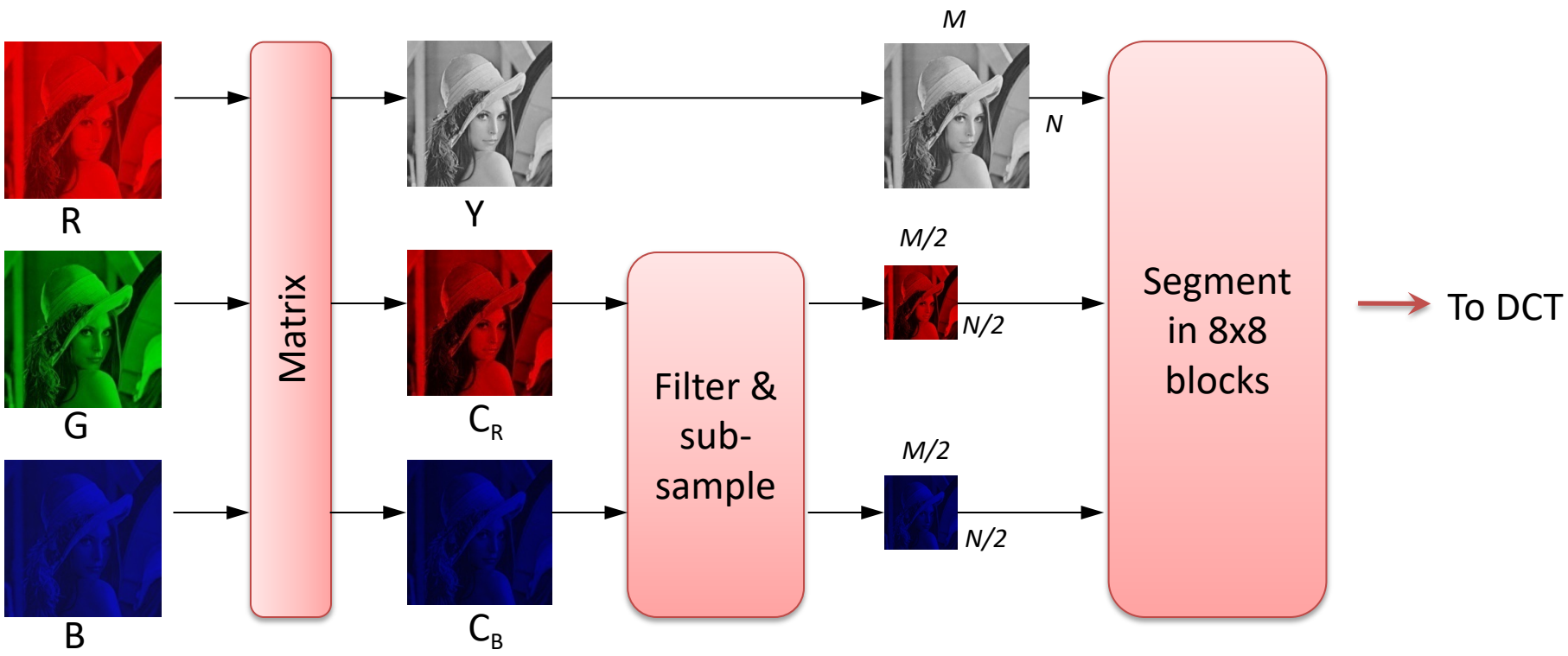
- Decoder:

From encoder → Codeword lookup → Q^{-1} → DCT^{-1} → Image block reconstruction

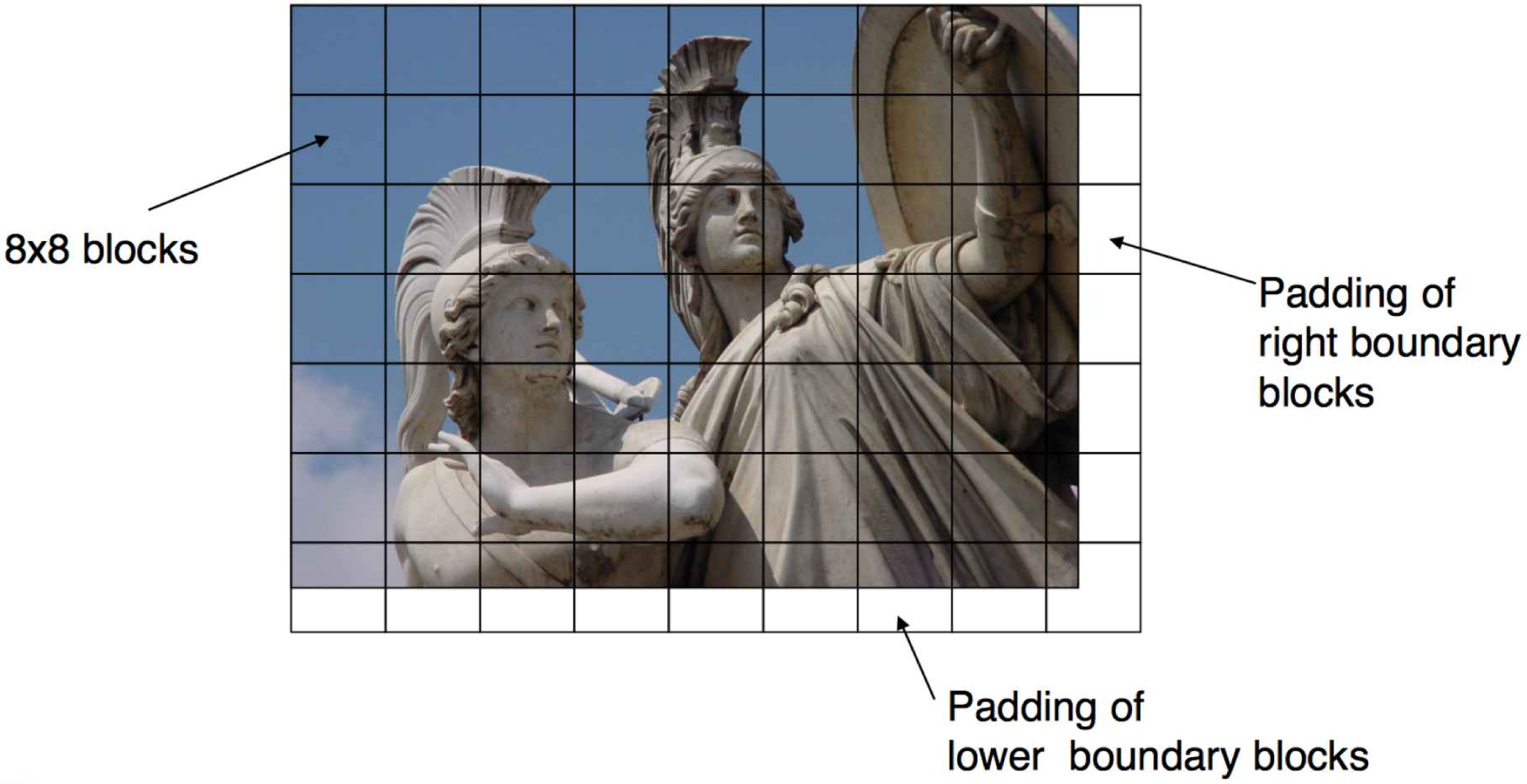


Lossy or lossless?

JPEG encoder – input stage detail

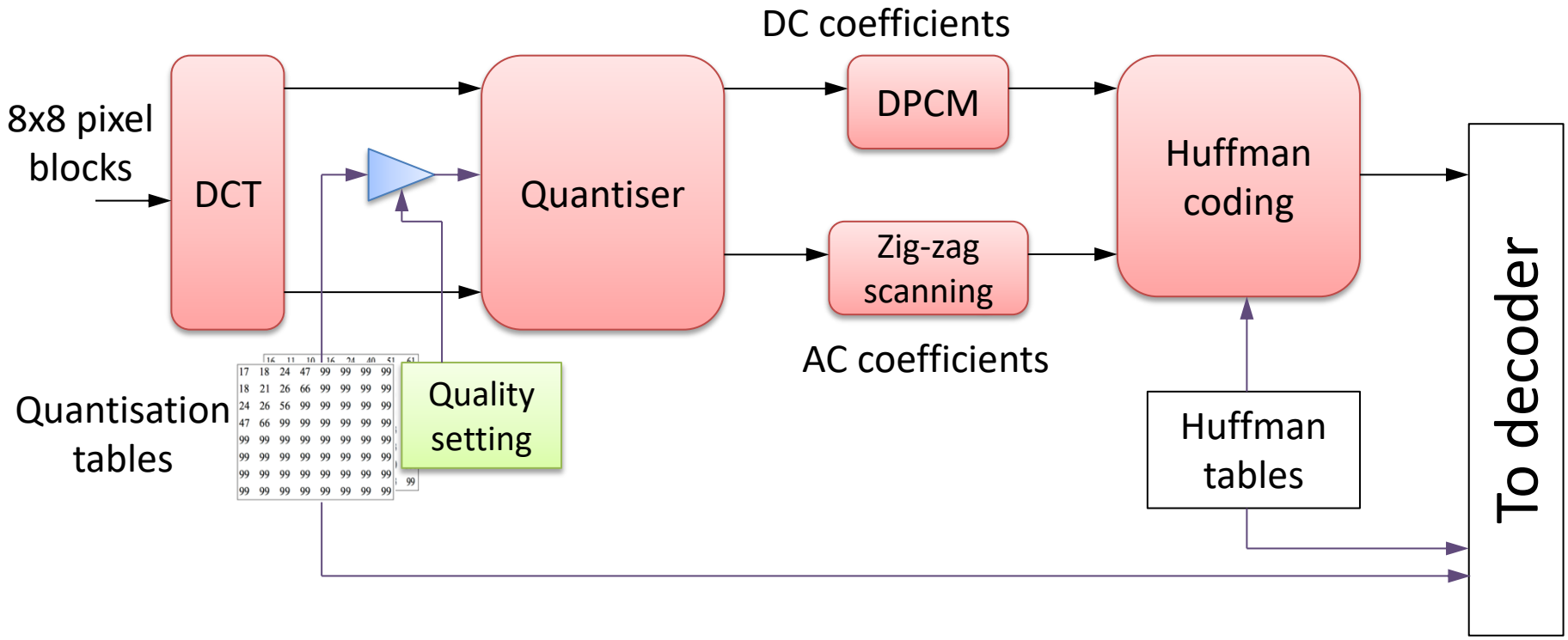


And if not a multiple of 8?

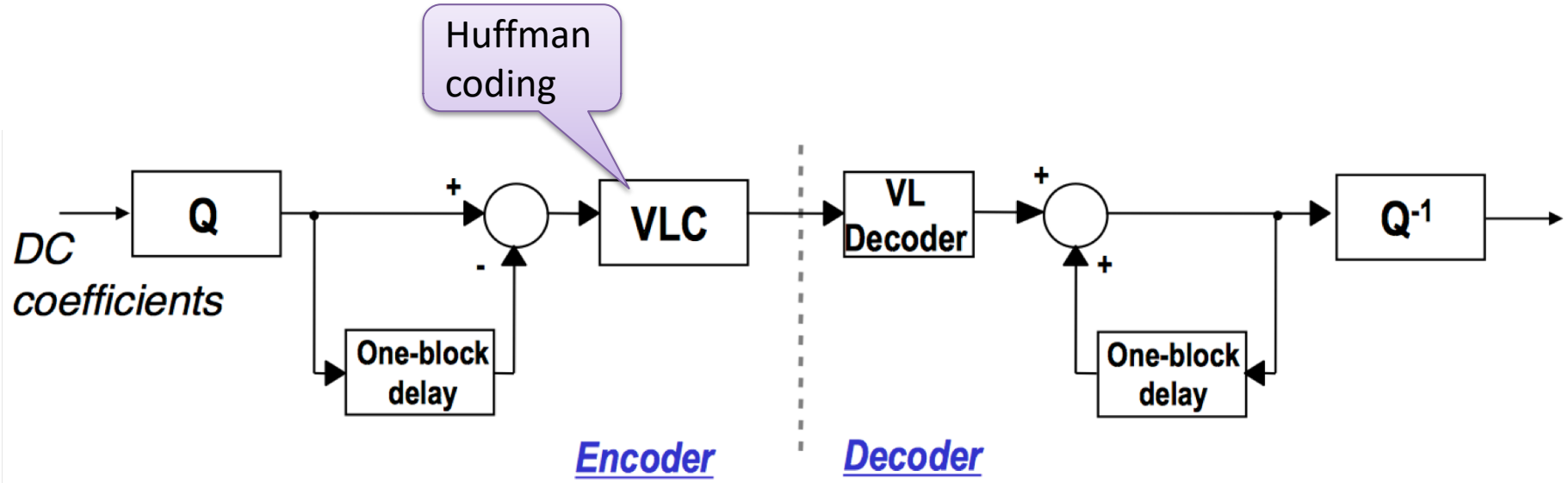


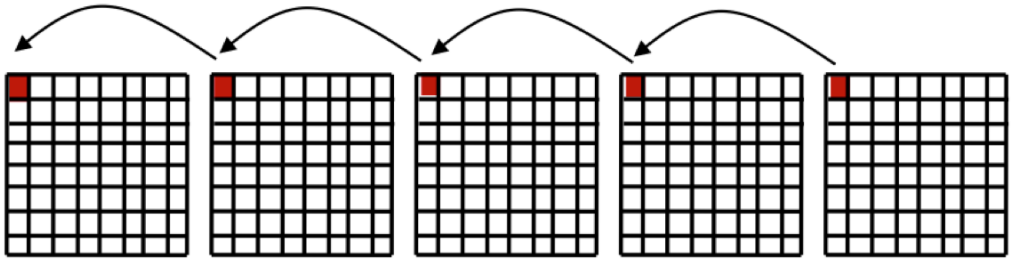
(Graphic from http://www.stanford.edu/class/ee398a/handouts/lectures/08-JPEG.pdf)

JPEG encoder – DCT, Quantisation and coding



Differential coding of DC coefficients



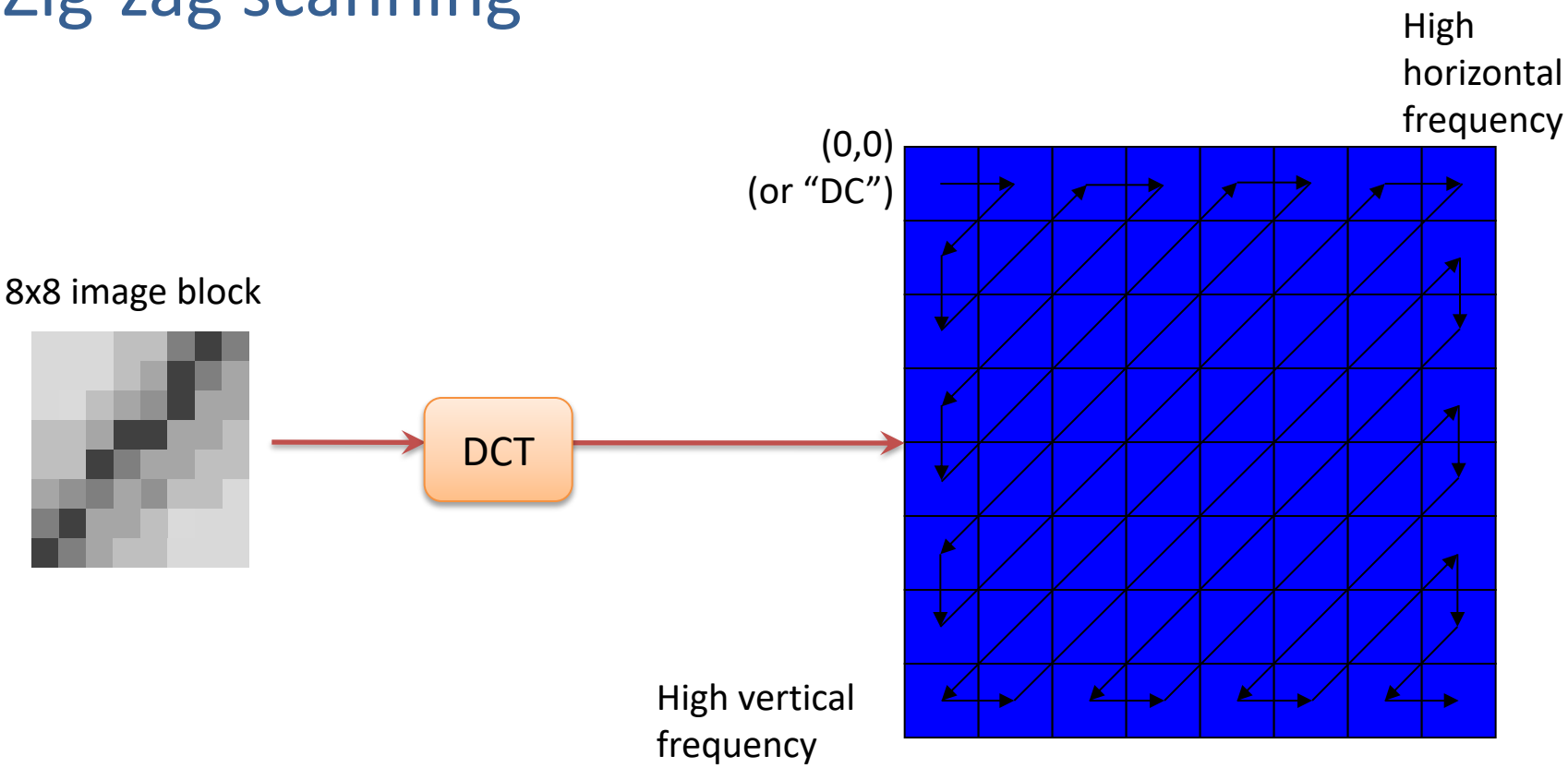


Tutorial question 3: JPEG performs predictive coding on just the "DC" (zero frequency) coefficient. Would it make sense to do it for any other coefficients?

(Graphic from http://www.stanford.edu/class/ee398a/handouts/lectures/08-JPEG.pdf)

Order of processing DCT coefficients

Zig-zag scanning



Tutorial question 4: Why does JPEG use DCT-based compression instead of DPCM?

Compressed digital video

Digital video compression exploits similarities (redundancy)

- within a picture
- and between pictures

The same conclusions from information theory apply:

- Anything we can do to decorrelate (so we can't guess what the information describing the next video frame will be) will lead to more efficient use of bit rate

And we want to exploit irrelevancy in both spatial and temporal dimensions.

Image: Telstra

Interframe prediction

- code differences between frames

Conditional replenishment

- Simplest form of interframe prediction

Unchanged	Unchanged	Unchanged	Changed
Unchanged	Unchanged	Changed	Changed

Frame difference signals

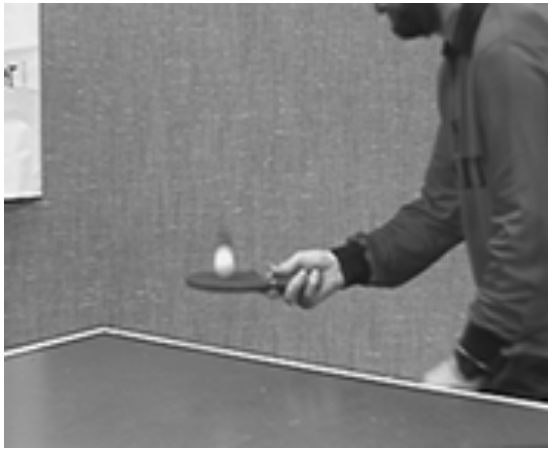


Image t

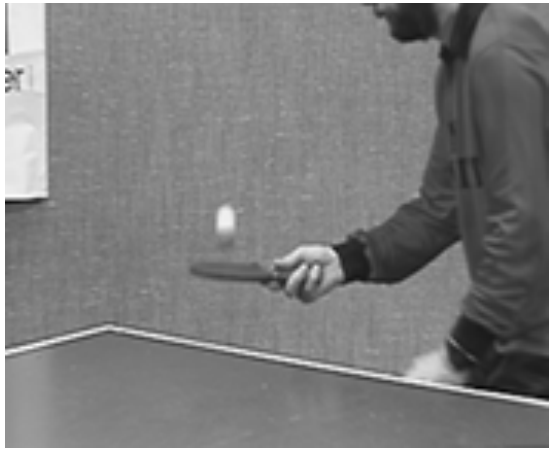
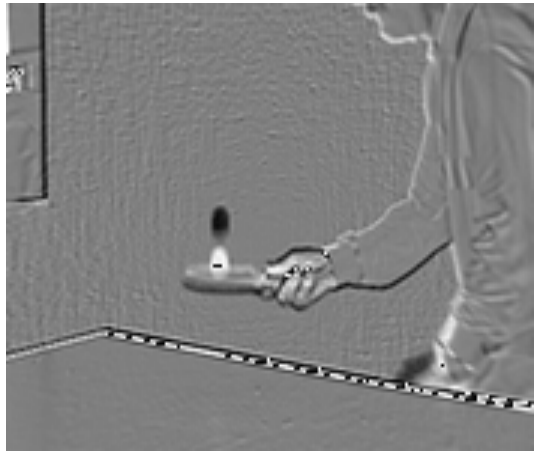


Image t+1



Difference

(Pictures courtesy Prof. Fernando Pereira, Istituto Superior Tecnico, Lisbon)

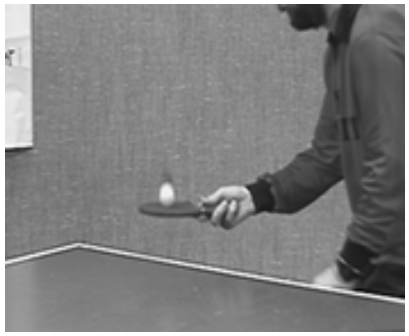
Coding and decoding with differences

Encoder

New picture

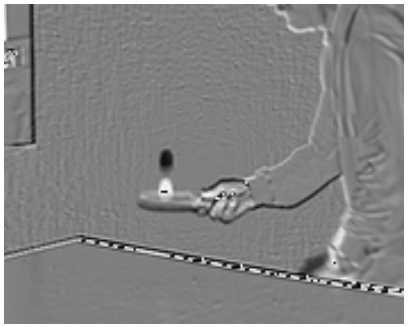
-

Previous picture



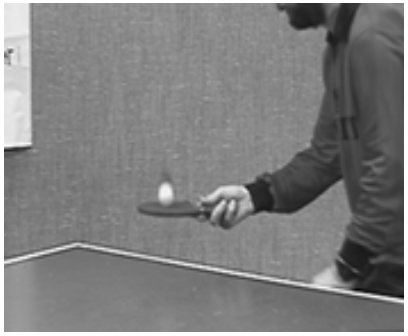
=

Difference picture



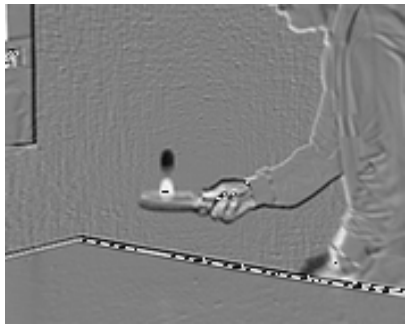
Decoder

Previous picture



+

Difference picture



=

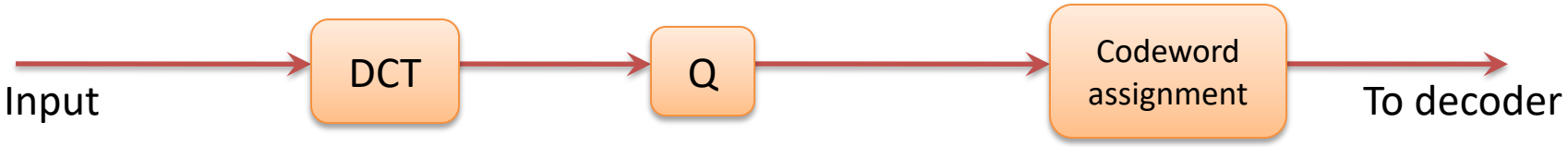
New picture

Does this remind you of something?

(Pictures courtesy Prof. Fernando Pereira, Istituto Superior Tecnico, Lisbon)

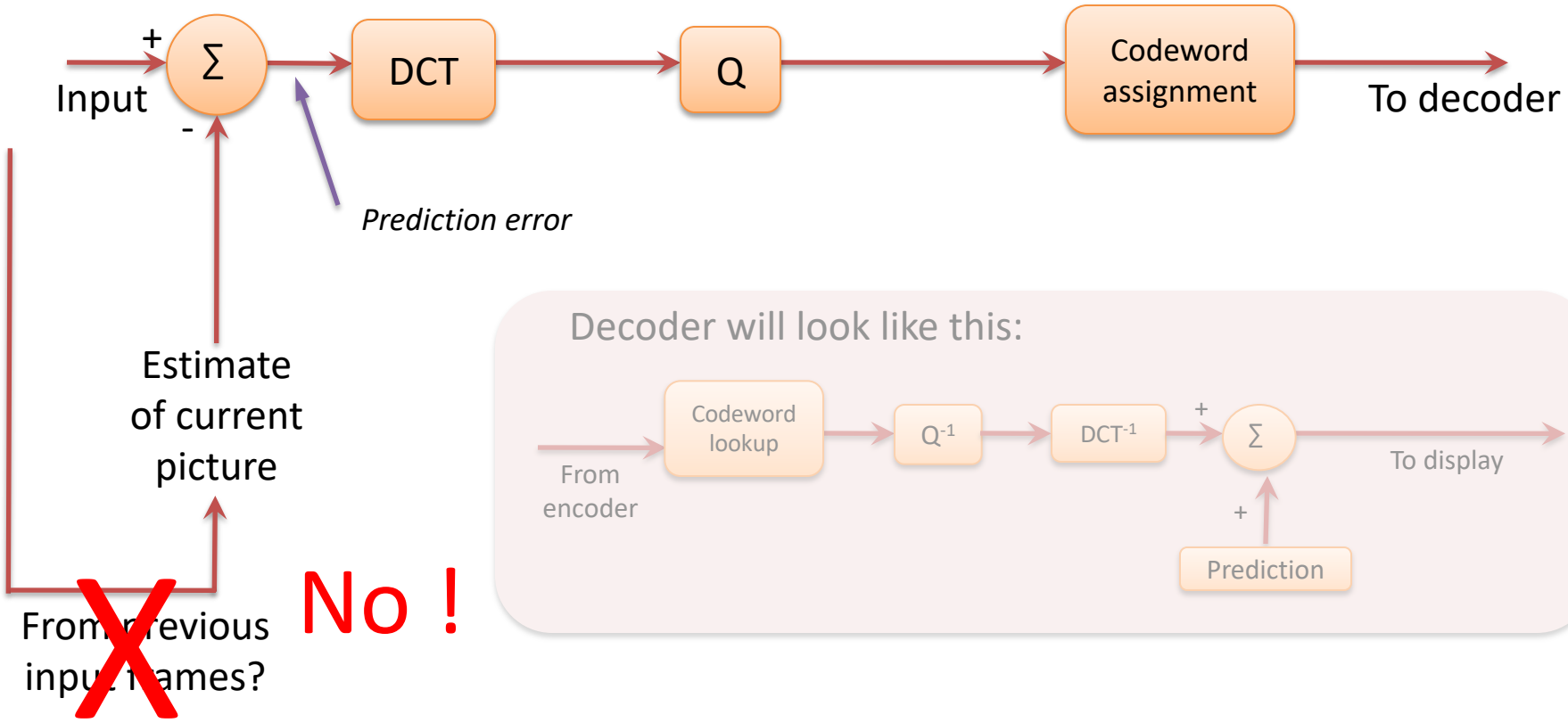
Basic video encoder

- Start with an image coder…



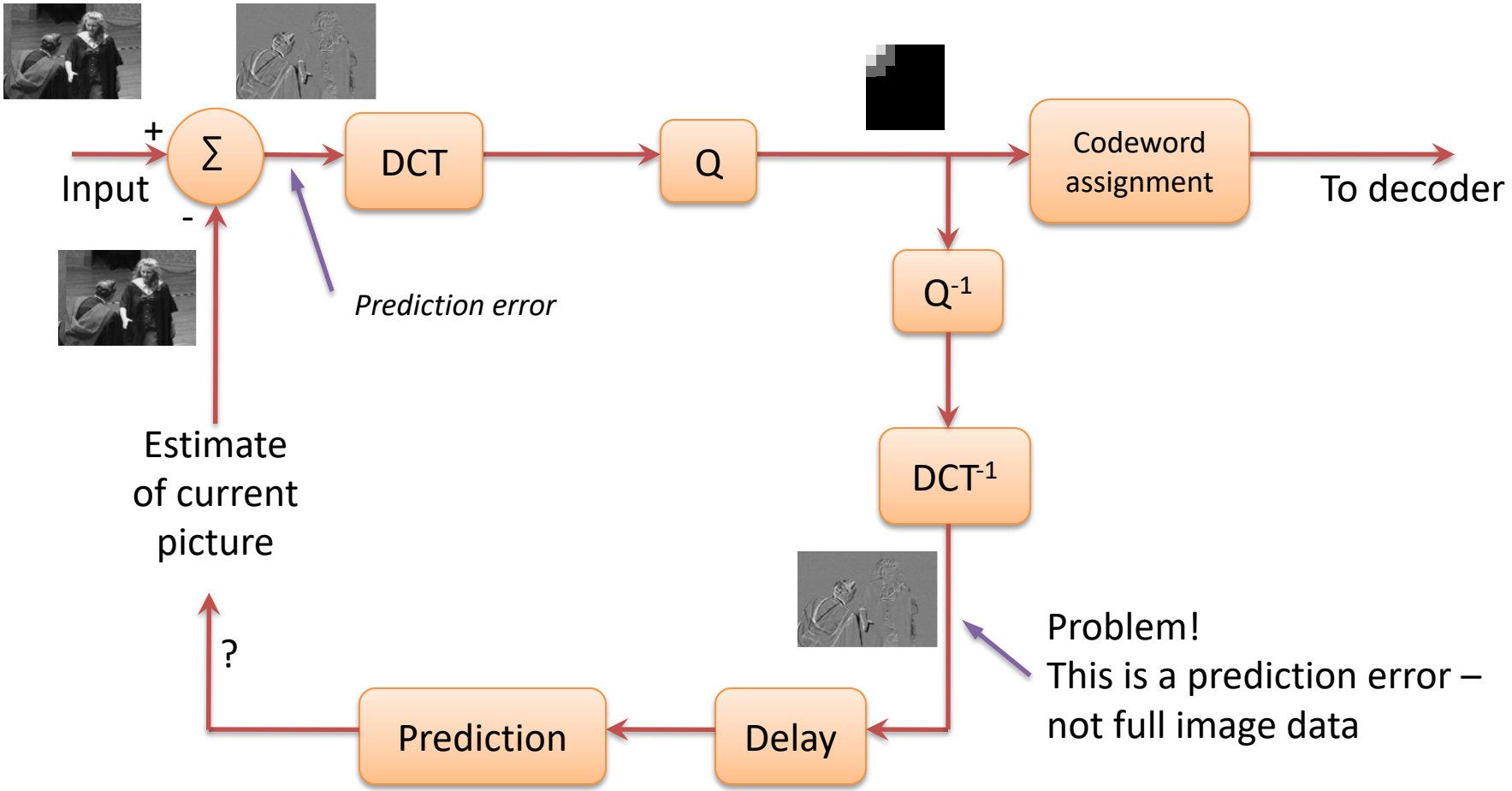
Basic video encoder

- Introduce some interframe prediction……



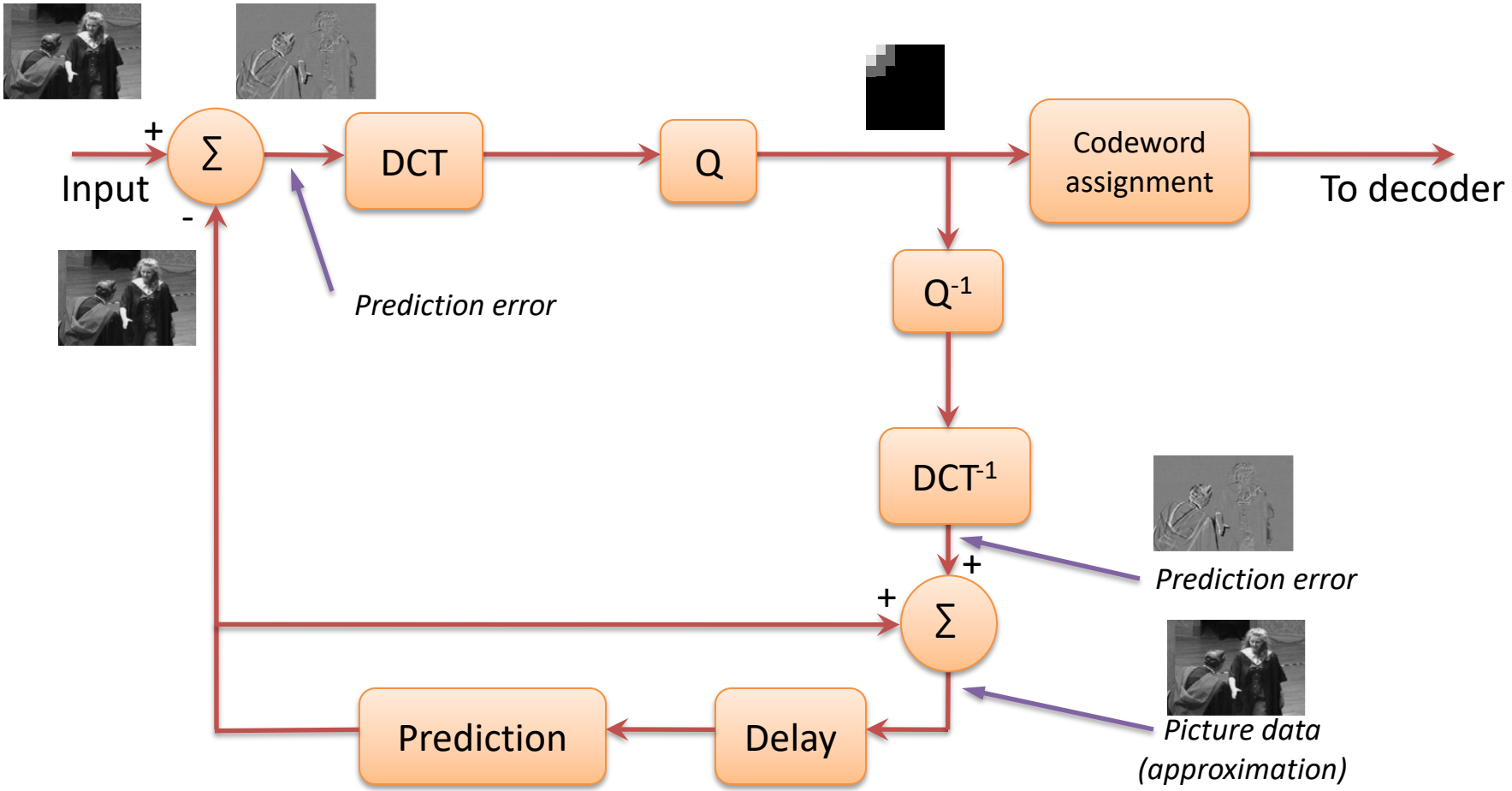
Basic video encoder

- Need to use what the decoder has……

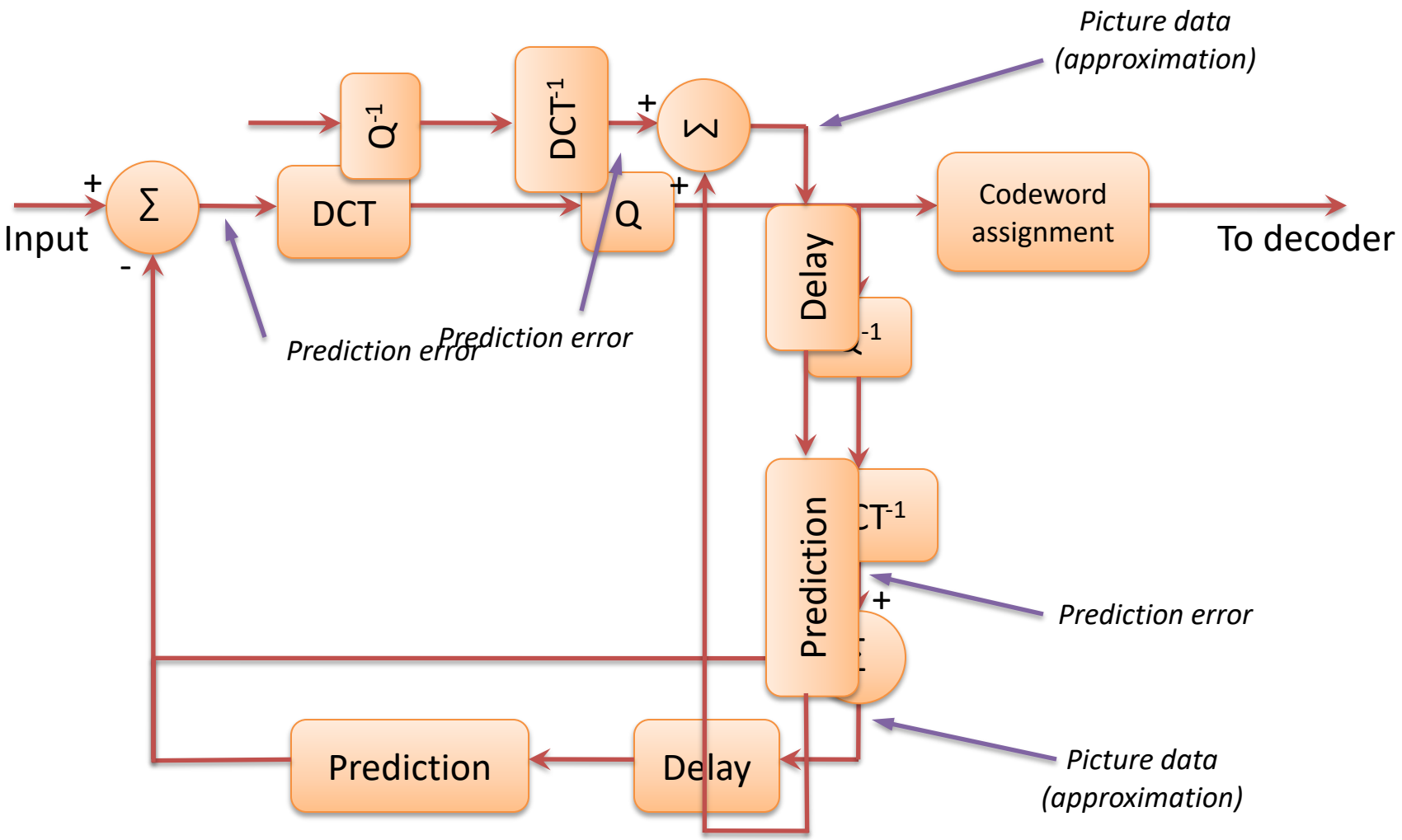


Basic video encoder

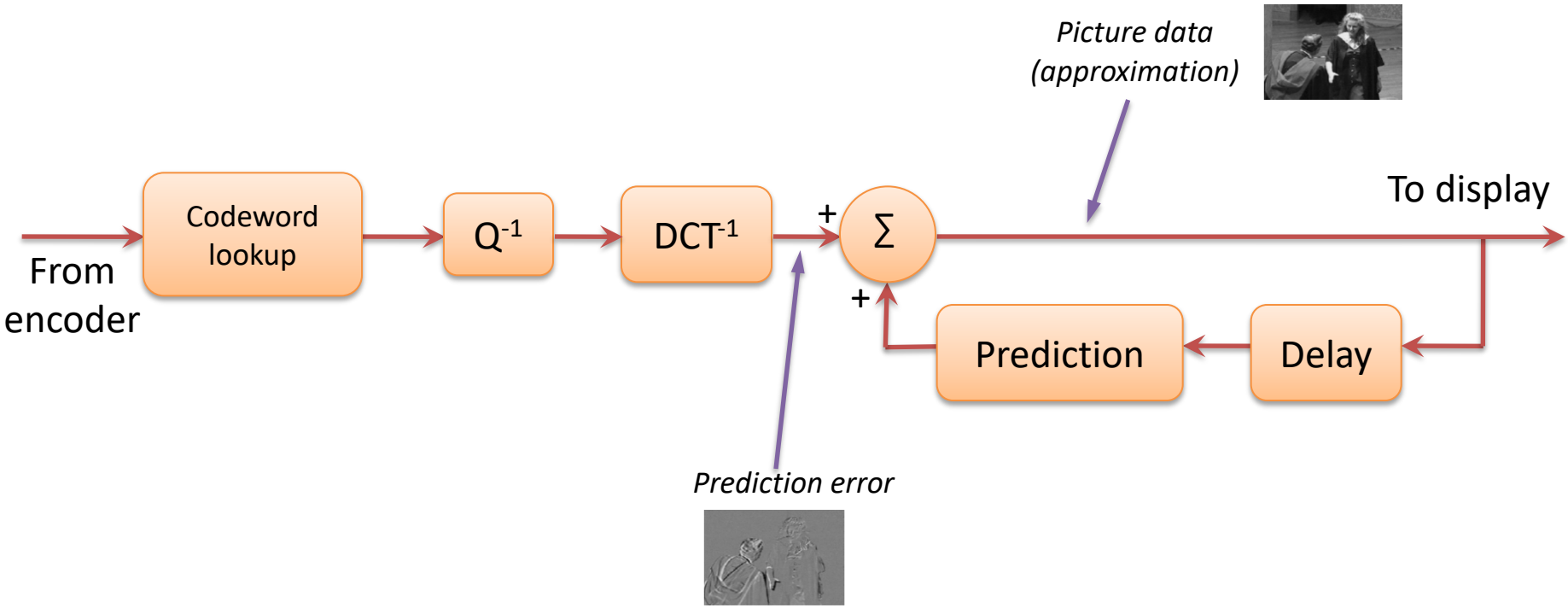
- And need to predict in the pixel, or image, domain……



The decoder is already there



Basic video decoder



Inter-frame prediction

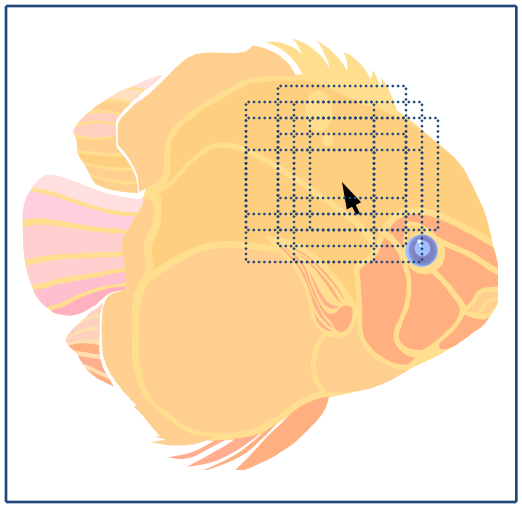
- Use previous picture as predictor for current picture
- Great when nothing moves or changes!
- But what if something does change? Can we make a more sophisticated prediction of the new picture, based on the previous picture?
- Consider two cases:

1. Scene change

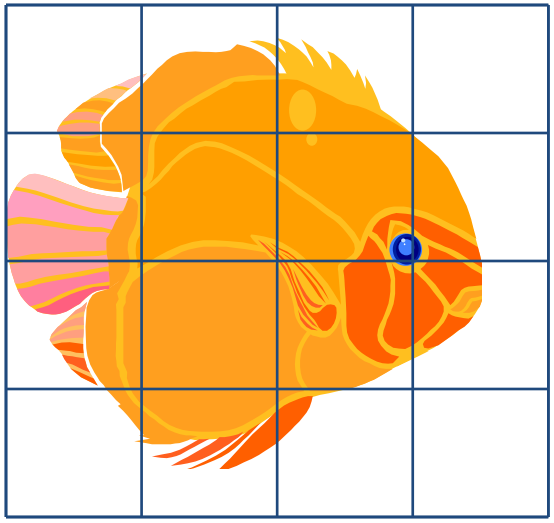
2. Moving picture

Motion compensation

Frame N-1



Frame N



- Use this motion vector to assist with inter-frame prediction

  If it is a good prediction, I can send two numbers (horizontal and vertical shift) to the decoder, but avoid sending n x n numbers (pixel data).

Motion compensation example

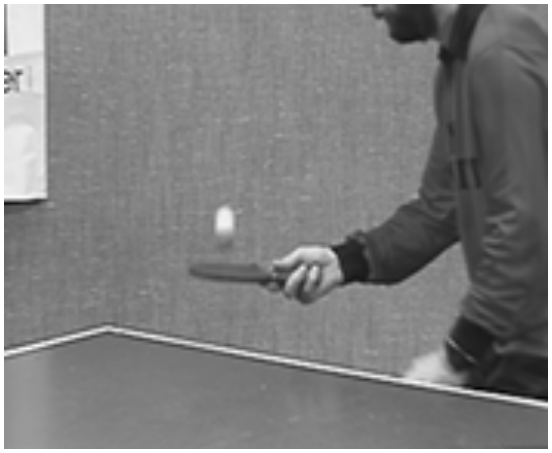


Image t

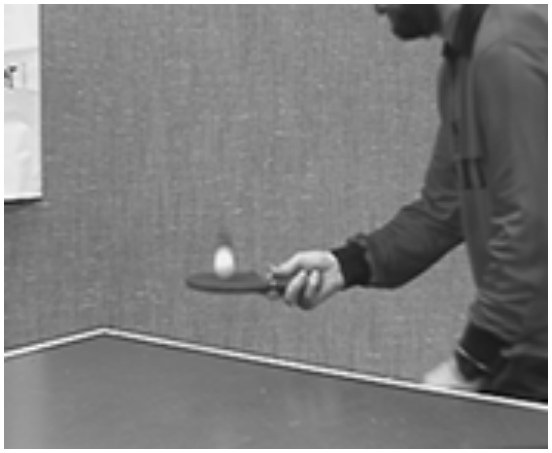
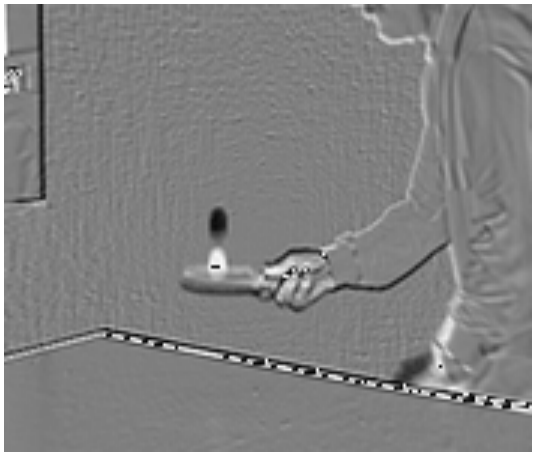
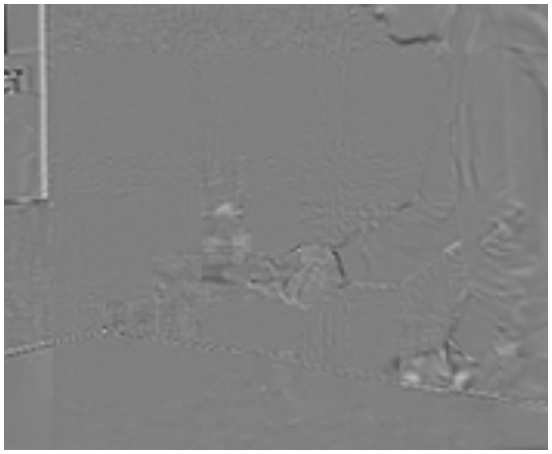


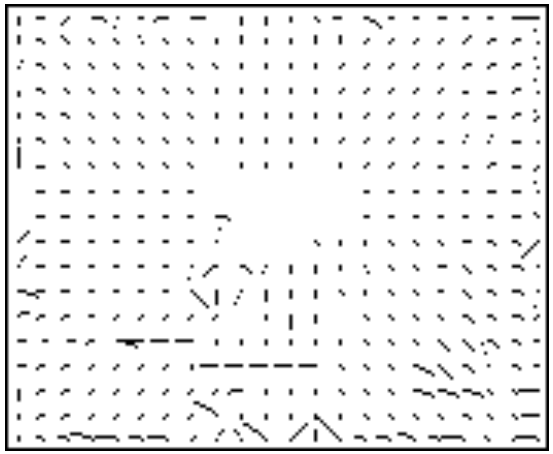
Image t-1



Difference without MC



Difference with MC

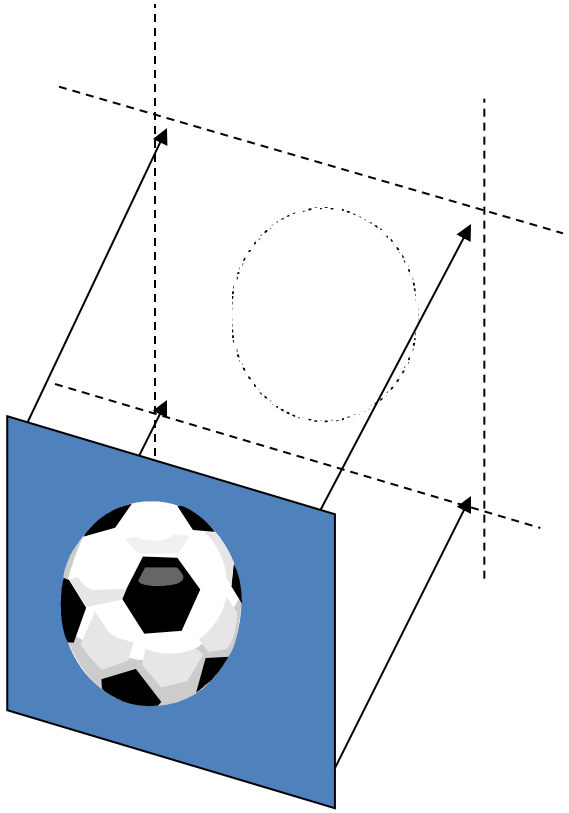


Motion vector field

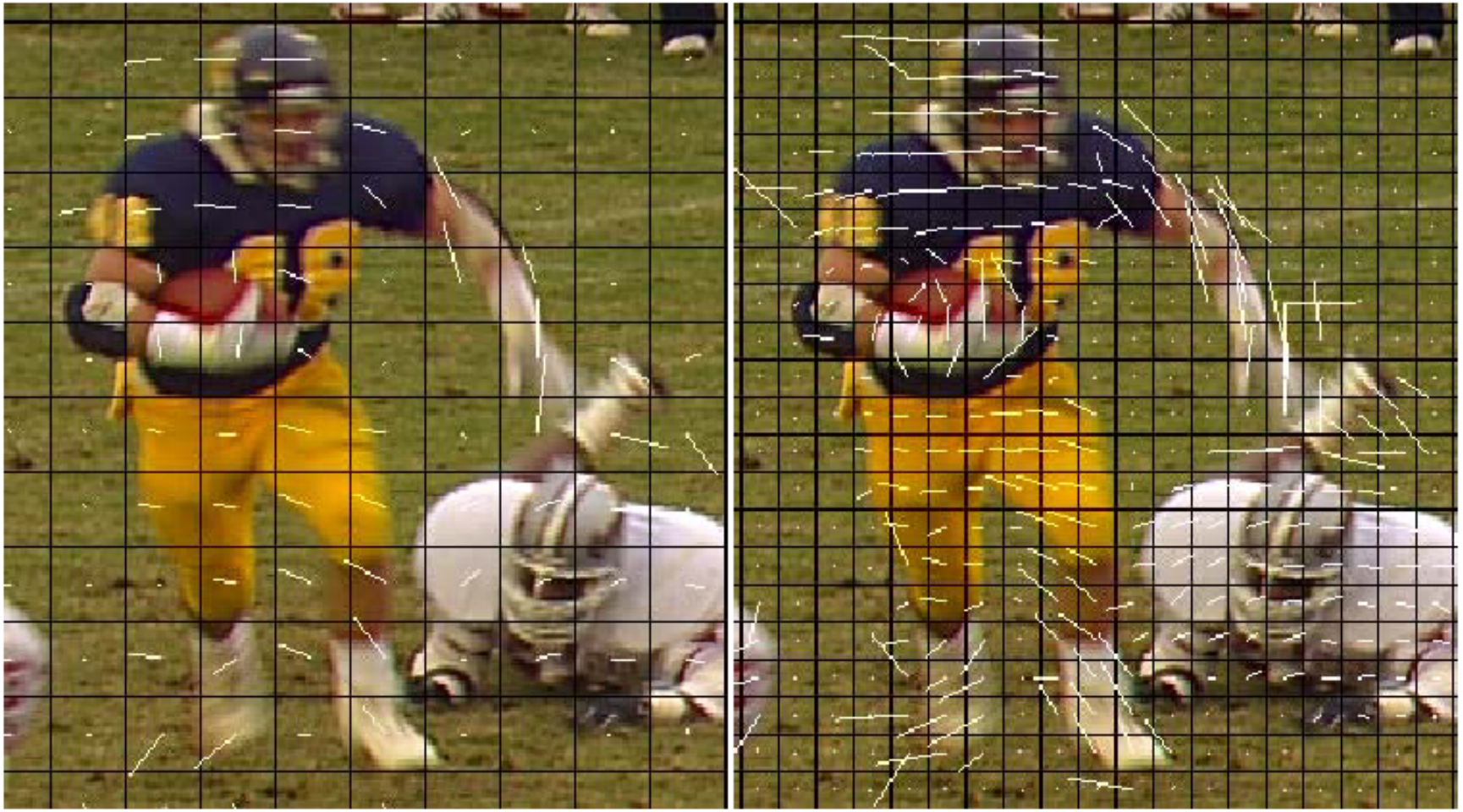
(Pictures courtesy Prof. Fernando Pereira, Istituto Superior Tecnico, Lisbon)

Motion estimation

- A pattern matching exercise
- Search strategy
  - Exhaustive, or full, search
  - “3-step” search
  - Sub-pixel motion estimation
- Minimum error criterion
  - MAD (Minimum Absolute Difference)
  - MSE (Mean Square Error)
- Will the minimum error lead to minimum bits?



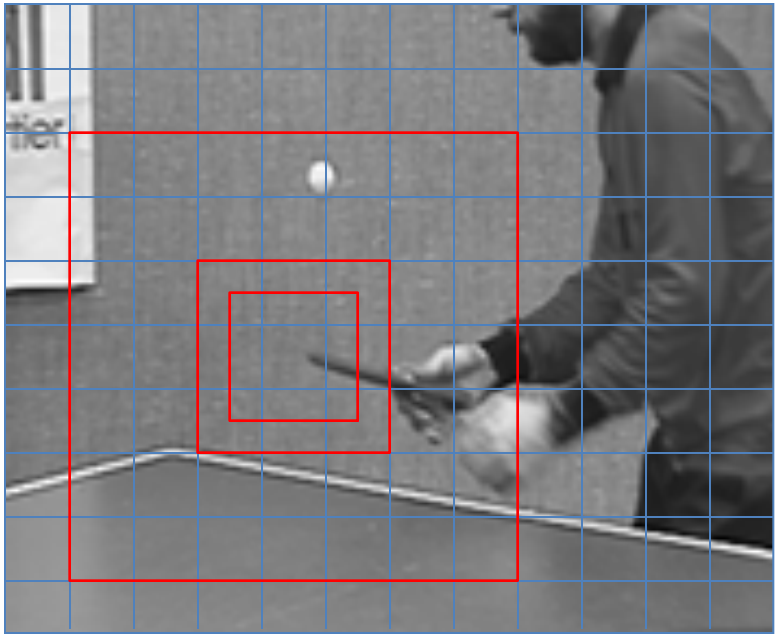
Motion vectors at different resolutions

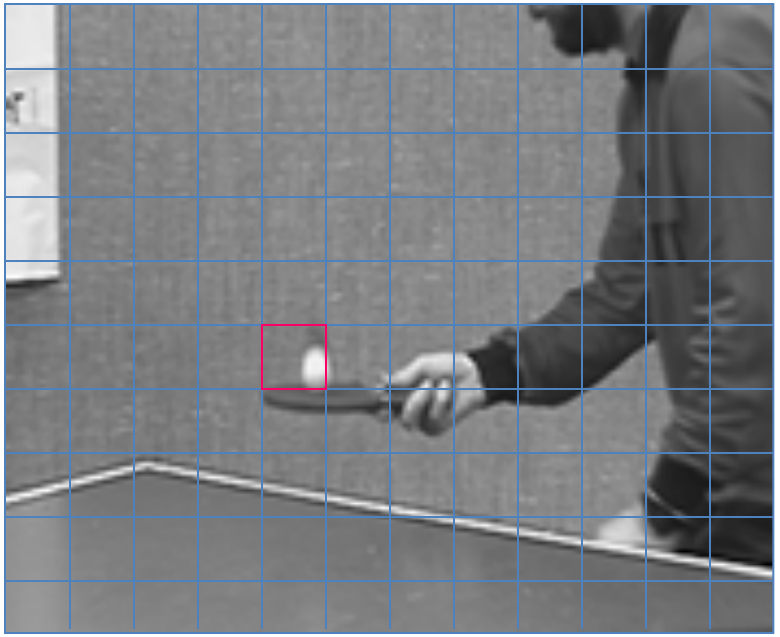


How big should the blocks be?

(Pictures courtesy Prof. Fernando Pereira, Istituto Superior Tecnico, Lisbon)

Search area - compromise





t

Over what area should we search?

(Pictures courtesy Prof. Fernando Pereira, Istituto Superior Tecnico, Lisbon)

Search area

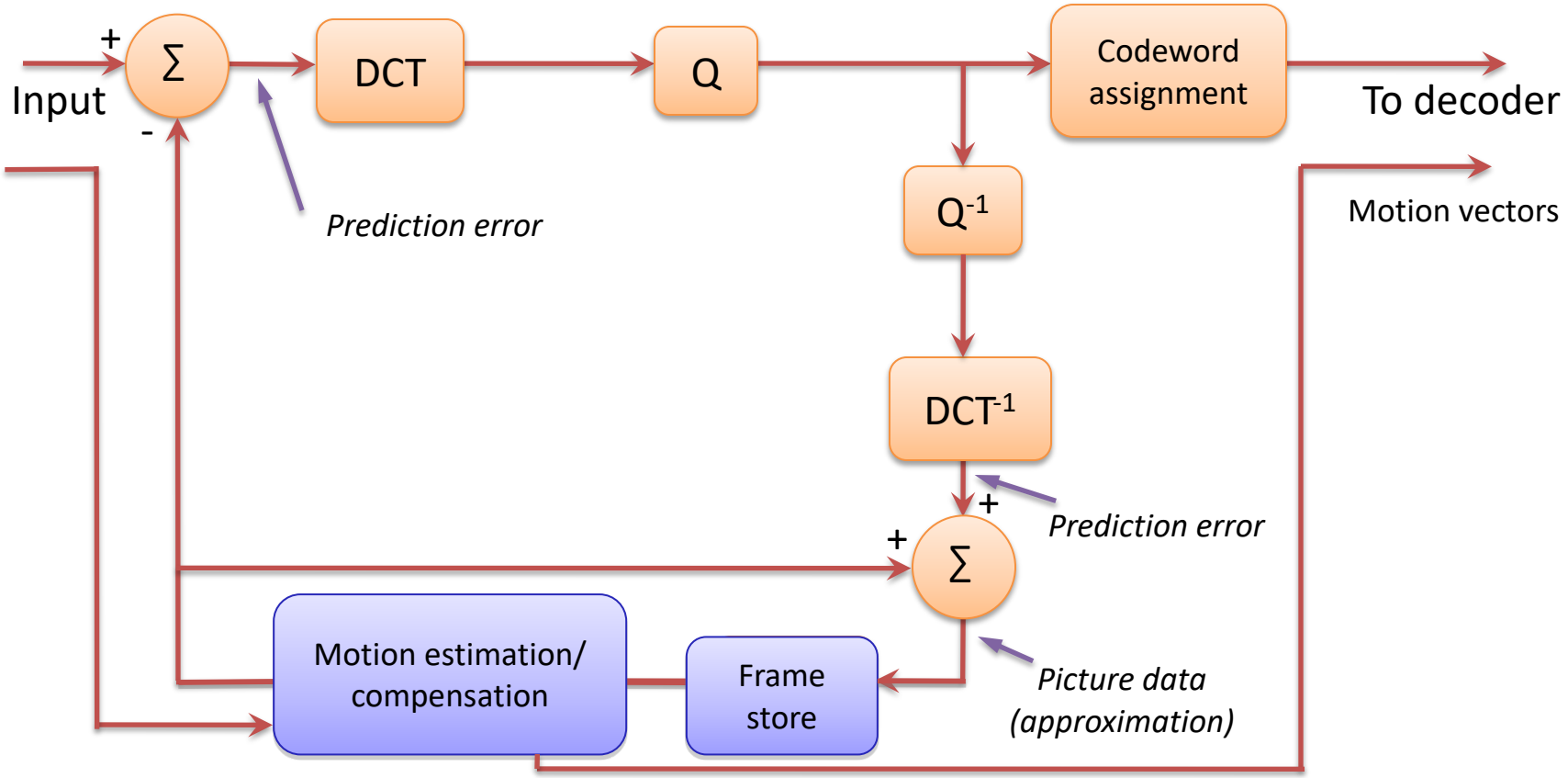
- A larger search area means:
  - ✓ More likely to find a good match

    BUT
  - × More computation
  - × Larger range of motion vector values (more bits to represent them)

Are there more motion vectors for a larger search area?

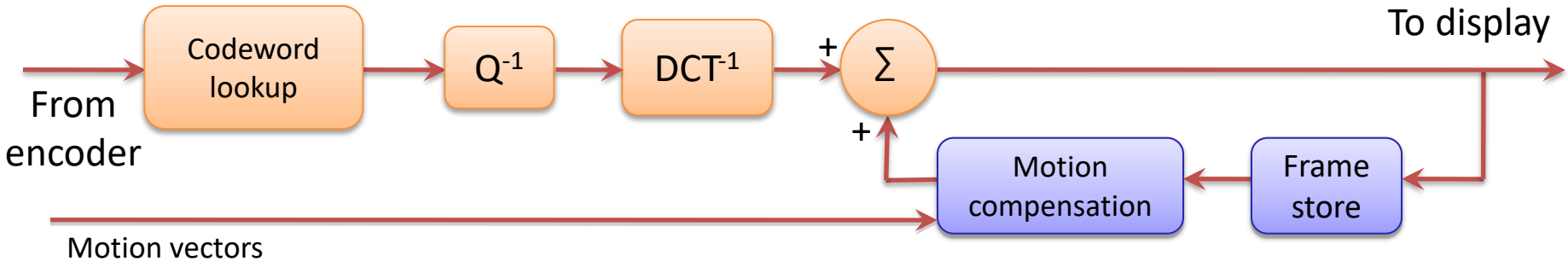
Basic video encoder

- Prediction takes the form of motion compensation



Basic video decoder

- Motion compensation in prediction loop



- Note: no motion estimation at the decoder!

Tutorial question 5: Will a video encoder such as we have described produce a constant bit rate (constant number of bytes/frame)? What if we are transmitting a constant bit rate?

You should now know:

- ✓ Principles of exploiting correlation for data reduction
- ✓ Principles of predictive coding
- ✓ Principles of transform coding
- ✓ The difference between redundant and irrelevant data
- ✓ The difference between lossy and lossless coding
- ✓ How the DCT works
- ✓ How DCT-based coding exploits characteristics of the eye
- ✓ How we exploit inter-frame predictive coding in video compression
- ✓ What motion compensation is and why it is used
- ✓ The basic form of a modern video coder