Algorithms in the Real World

Data Compression 4

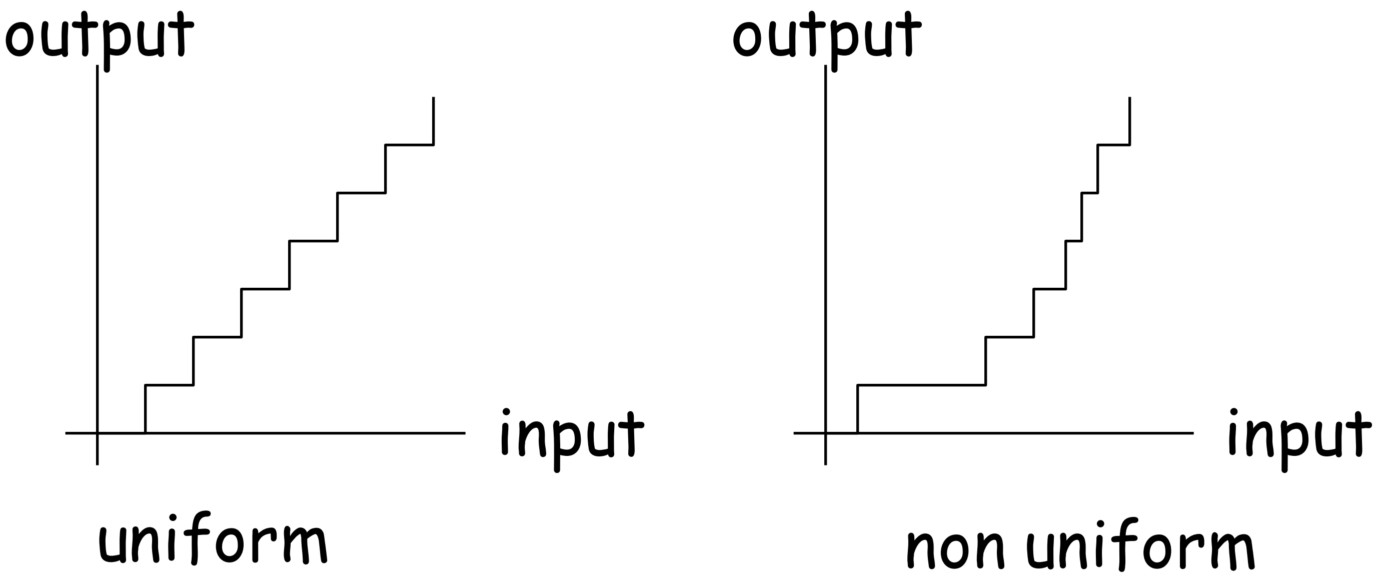
Compression Outline

Introduction: Lossy vs. Lossless, Benchmarks, …
Information Theory: Entropy, etc.
Probability Coding: Huffman + Arithmetic Coding
Applications of Probability Coding: PPM + others
Lempel-Ziv Algorithms: LZ77, gzip, compress, …
Other Lossless Algorithms: Burrows-Wheeler
Lossy algorithms for images: JPEG, MPEG, …
- Scalar and vector quantization
- JPEG and MPEG

Compressing graphs and meshes: BBK

Scalar Quantization

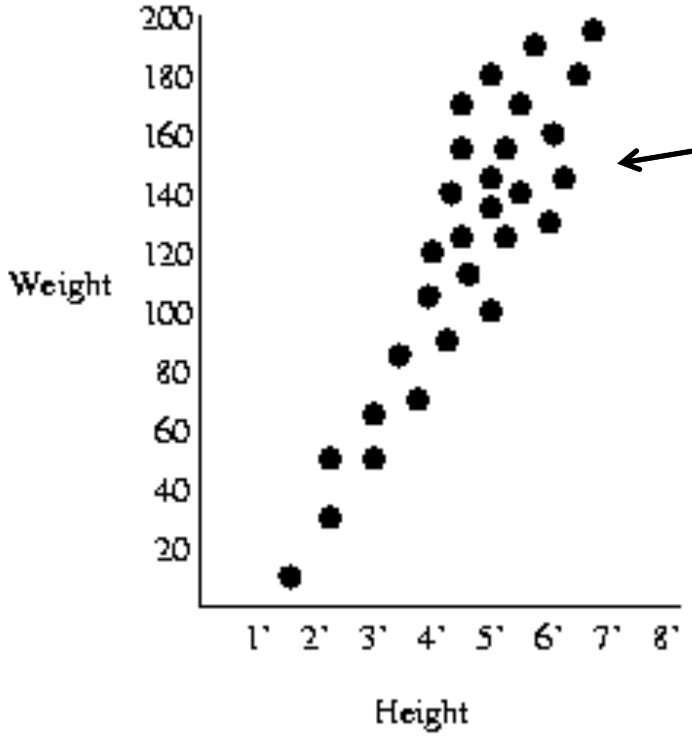
Quantize regions of values into a single value:



Quantization is lossy

Can be used, e.g., to reduce # of bits for a pixel

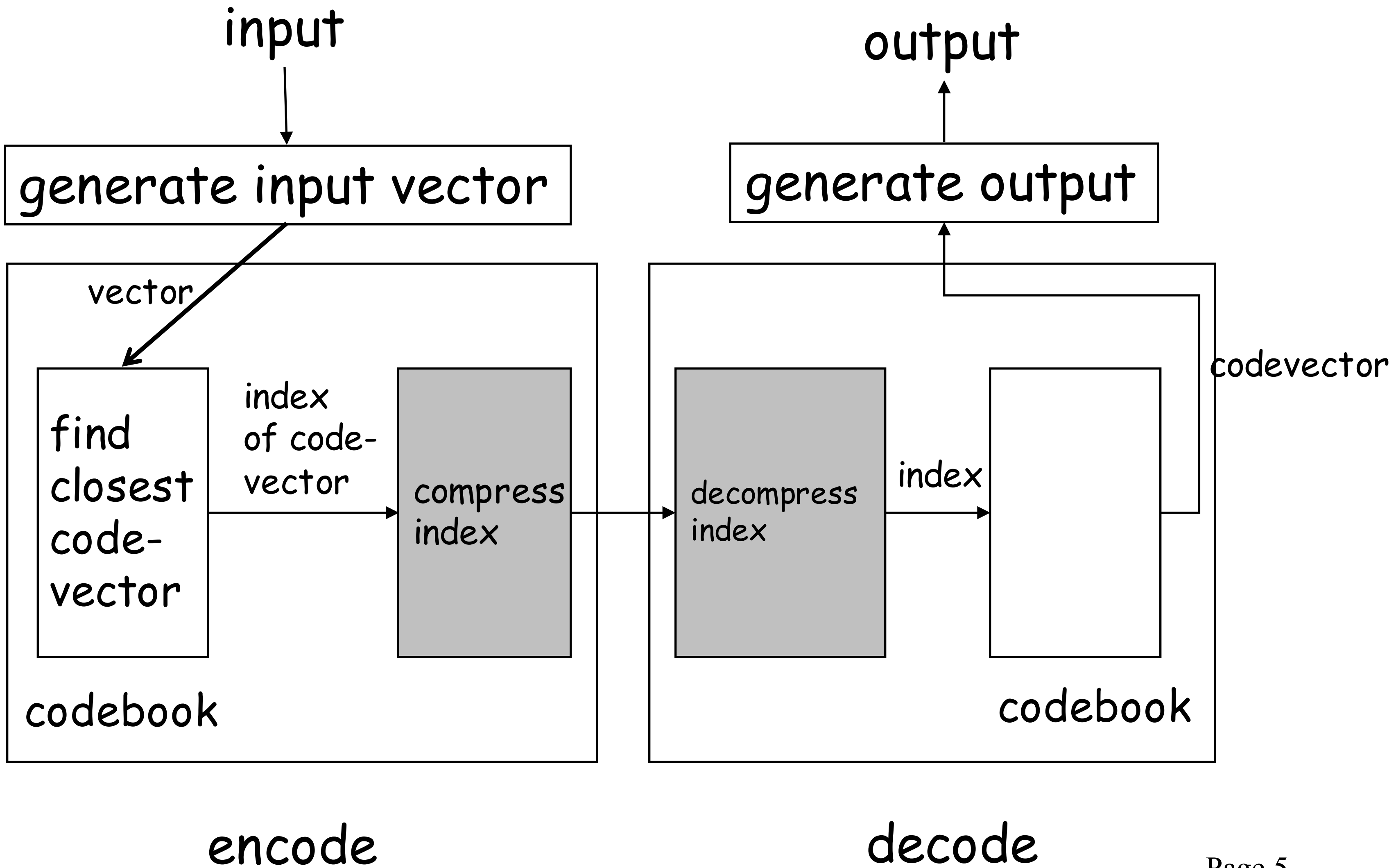
Vector Quantization: Example



representative codevectors

Input vectors are (Height, Weight) pairs.
Map each input vector to a representative "codevector".
Codevectors are stored in a codebook.

Vector Quantization



Vector Quantization

What do we use as vectors?

- Color (Red, Green, Blue)
  - Can be used, for example to reduce 24bits/pixel to 12bits/pixel
  - Used in some terminals to reduce data rate from the CPU (colormaps)
- k consecutive samples in audio
- Block of k x k pixels in an image

How do we decide on a codebook

- Typically done with **clustering**

Linear Transform Coding

Want to encode values over a region of time or space

- typically used for images or audio
- represented as a vector $[x_1, x_2, \ldots]$

Select a set of linear basis functions φ_i that span the space

- sin, cos, spherical harmonics, wavelets, …
- defined at discrete points

Linear Transform Coding

Coefficients: $\Theta_i = \sum_j x_j \phi_i(j) = \sum_j x_j a_{ij}$

Θ_i = i^{th} resulting coefficient

x_j = j^{th} input value

a_{ij} = ij^{th} transform coefficient $= \phi_i(j)$

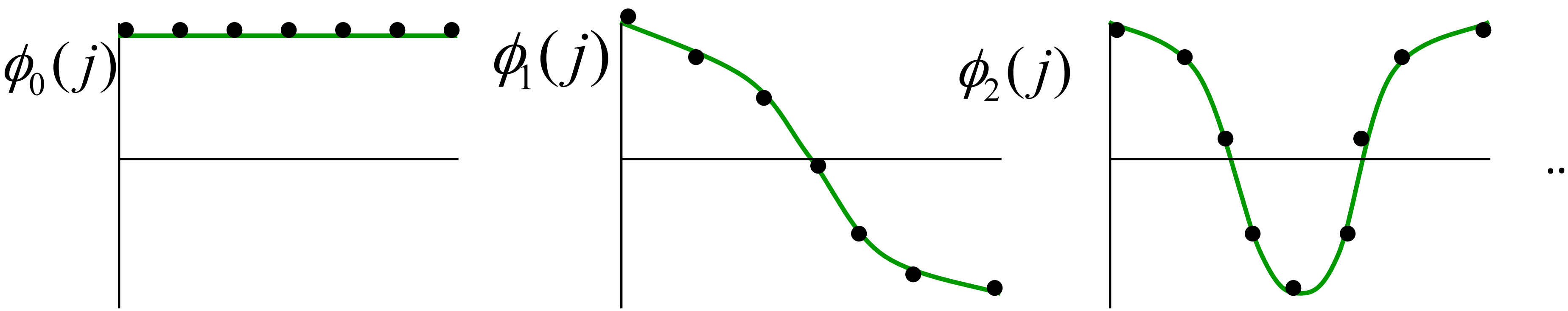
In matrix notation:

$$\Theta = Ax$$

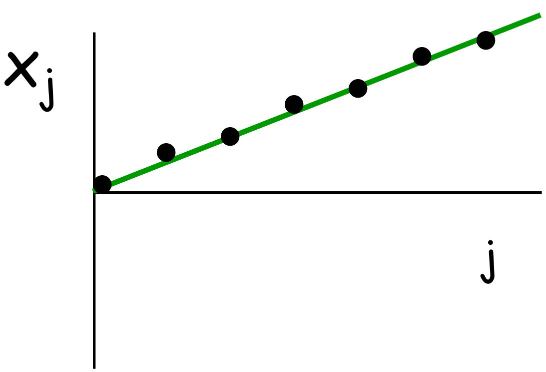
$$x = A^{-1}\Theta$$

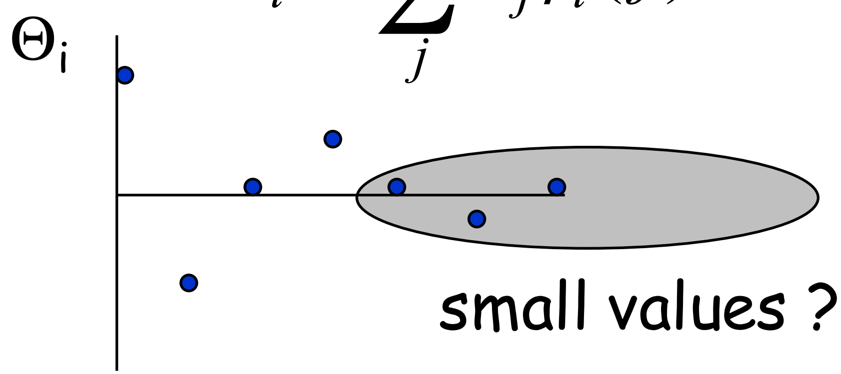
Where A is an n x n matrix, and each row defines a basis function

Example: Cosine Transform



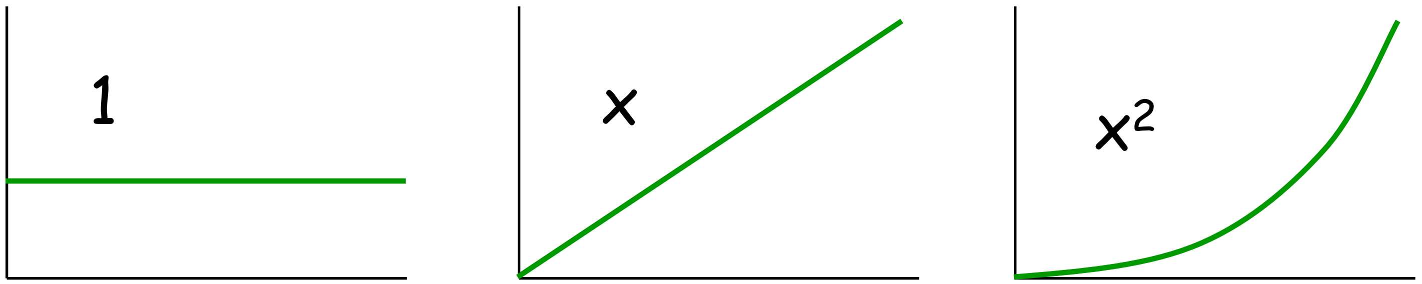
$$\Theta_i = \sum_j x_j \phi_i(j)$$



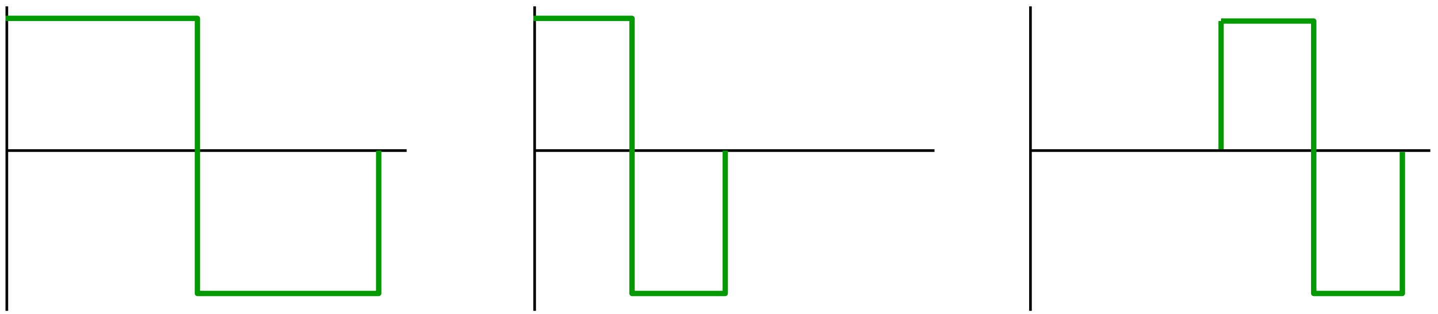


Other Transforms

Polynomial:



Wavelet (Haar):



How to Pick a Transform

Goals:

- Decorrelate (remove repeated patterns in data)
- Low coefficients for many terms
- Some terms affect perception more than others

Why is using a Cosine or Fourier transform across a whole image bad?

-- If there is no periodicity in the image, large coefficients for high-frequency terms

How might we fix this?

-- use basis functions that are not as smoothly periodic

Usefulness of Transform

Typically transforms A are **orthonormal**: $A^{-1} = A^T$

Properties of orthonormal transforms:

- $\sum x^2 = \sum \Theta^2$ (energy conservation)

Would like to compact energy into as few coefficients as possible

$$G_{TC} = \frac{\frac{1}{n}\sum \sigma_i^2}{\left(\prod \sigma_i^2\right)^{1/n}}$$

(the **transform coding gain**)
arithmetic mean/geometric mean

$\sigma_i = (\Theta_i - \Theta_{av})$

The higher the gain, the better the compression

Case Study: JPEG

A nice example since it uses many techniques:

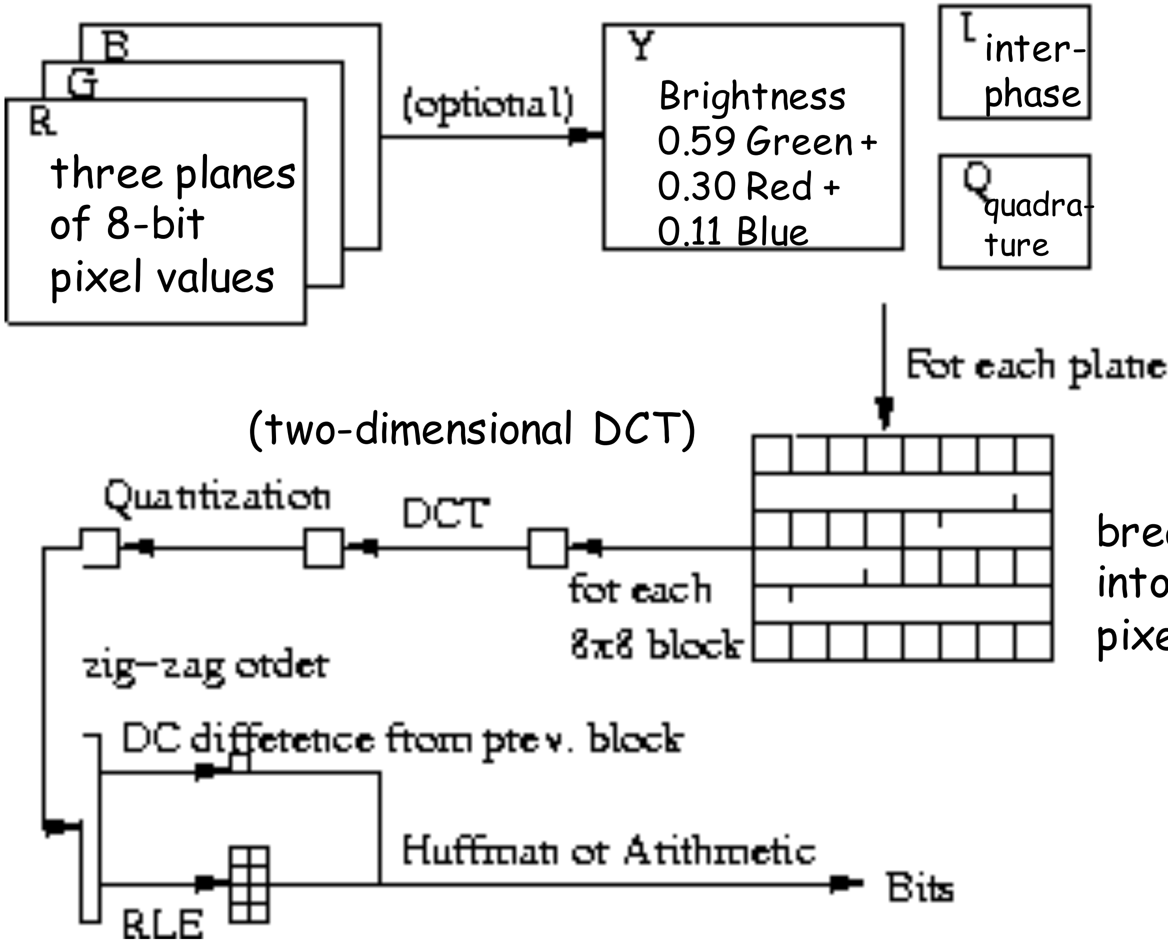
- Transform coding (Discrete Cosine Transform)
- Scalar quantization
- Difference coding
- Run-length coding
- Huffman or arithmetic coding

JPEG (Joint Photographic Experts Group) was designed in **1991** for **lossy** and **lossless** compression of **color** or **grayscale images**. The lossless version is rarely used.

Can be adjusted for compression ratio (typically 10:1)

JPEG in a Nutshell

original image



Typically down-sample I and Q planes by a factor of 2 in each dimension – lossy. Factor of 4 compression for I and Q, 2 overall.

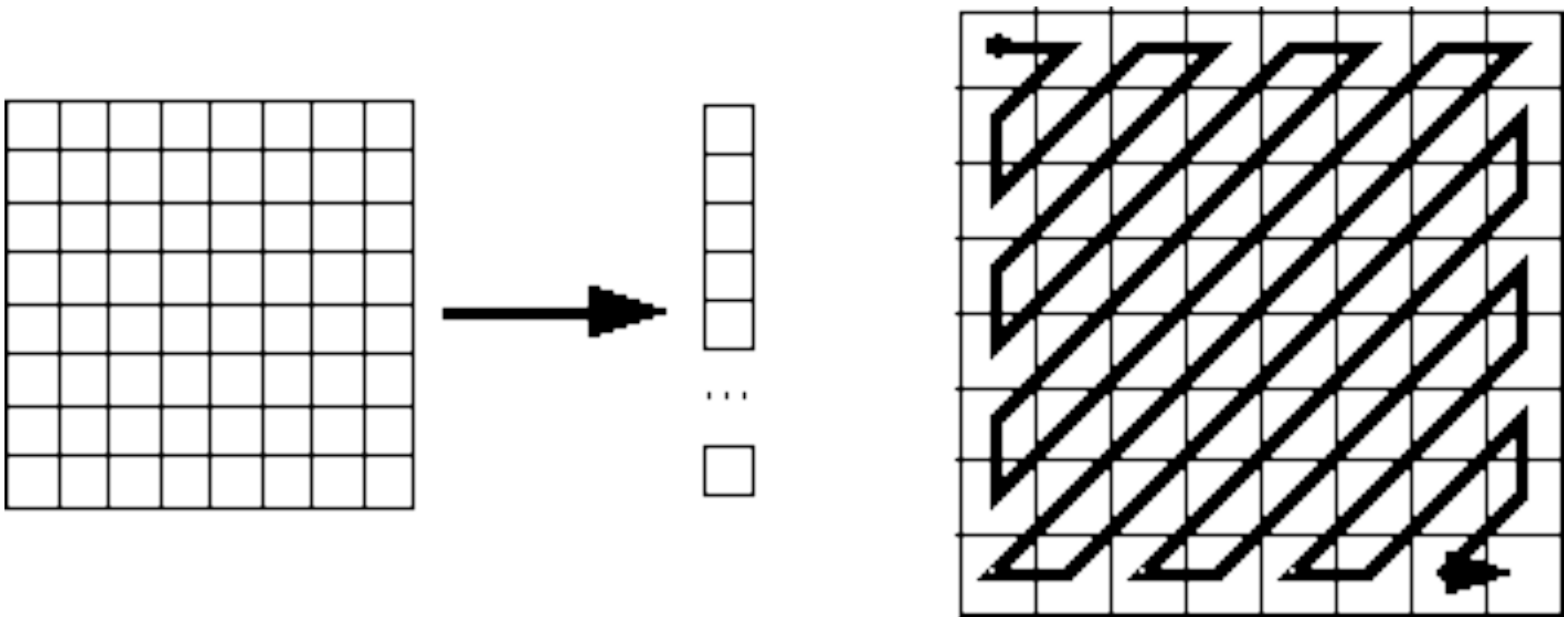
break each plane into 8x8 blocks of pixels

JPEG: Quantization Table

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

Divide each coefficient by factor shown. Also divided through uniformly by a quality factor which is under control.

JPEG: Block scanning order



- Scan block of coefficients in zig-zag order
- Use difference coding upper left (DC) coefficient between consecutive blocks
- Uses run-length coding for sequences of zeros for rest of block

JPEG: example

.125 bits/pixel (factor of 192)

Case Study: MPEG

Pretty much JPEG with **interframe coding**

Three types of frames

- **I** = intra frame (approx. JPEG) anchors
- **P** = predictive coded frames – based on previous I or P frame in output order
- **B** = bidirectionally predictive coded frames - based on next and/or previous I or P frames in output order

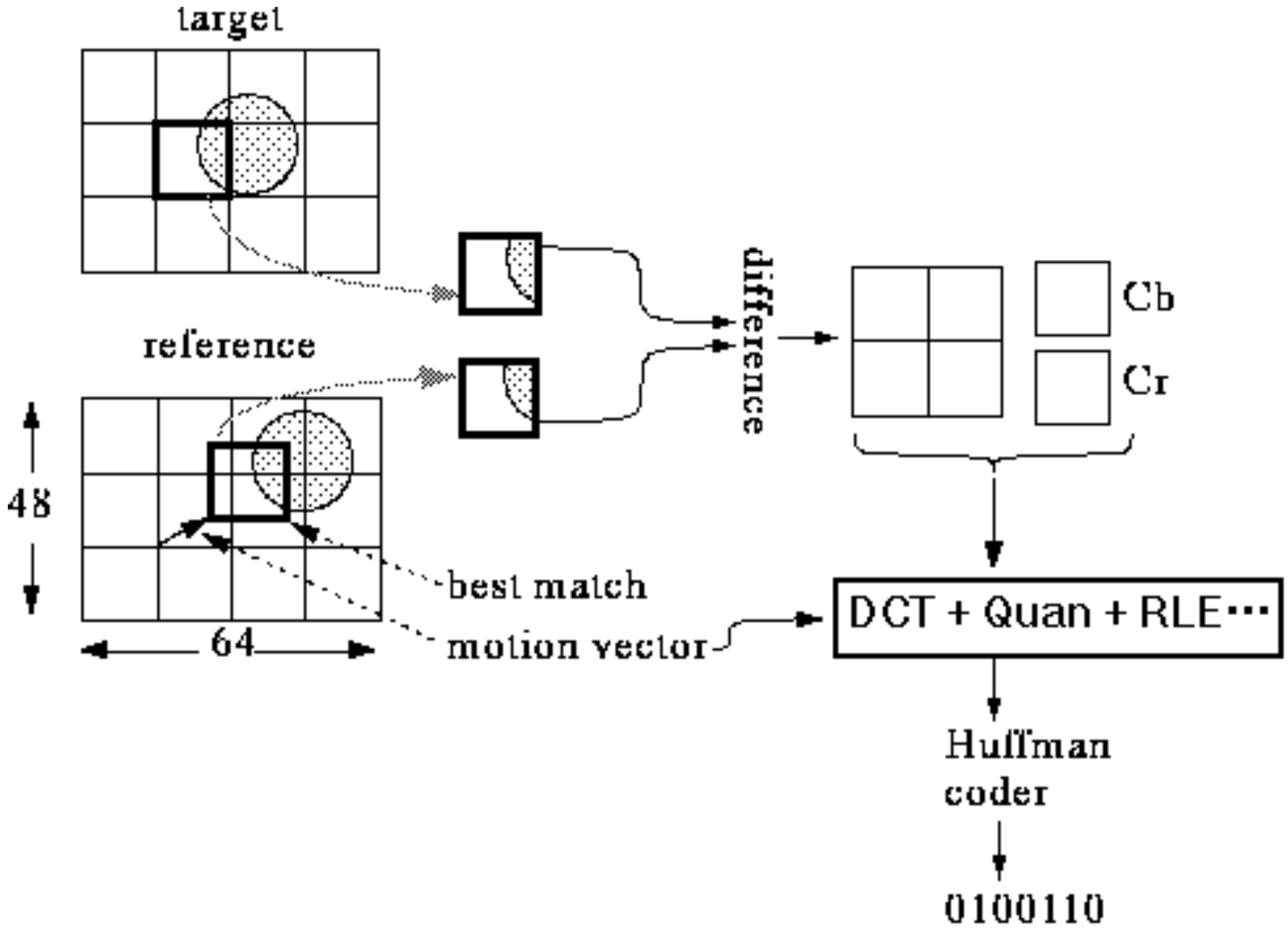
ordered chronologically in input

Example:

Type:	I	B	B	P	B	B	P	B	B	P	B	B	I
Output Order:	1	3	4	2	6	7	5	9	10	8	12	13	11

I frames are used for random access.
In the sequence, each **B** frame appears after any frame on which it depends.

MPEG matching between frames



MPEG: Compression Ratio

356 x 240 image

Type	Size	Compression
I	18KB	7/1
P	6KB	20/1
B	2.5KB	50/1
Average	4.8KB	27/1

30 frames/sec x 4.8KB/frame x 8 bits/byte
= 1.2 Mbits/sec + .25 Mbits/sec (stereo audio)

HDTV has 15x more pixels
= 18 Mbits/sec

MPEG in the "real world"

- DVDs
  - Adds "encryption" and error correcting codes
- Direct broadcast satellite
- HDTV standard
  - Adds error correcting code on top
- Storage Tech "Media Vault"
  - Stores 25,000 movies

Encoding is much more expensive than decoding.

Still requires special purpose hardware for high resolution and good compression.

Compression Summary

How do we figure out the probabilities

- Transformations that skew them
  - Guess value and code difference
  - Move to front for temporal locality
  - Run-length
  - Linear transforms (Cosine, Wavelet)
  - Renumber (graph compression)
- Conditional probabilities
  - Neighboring context

In practice one almost always uses a combination of techniques