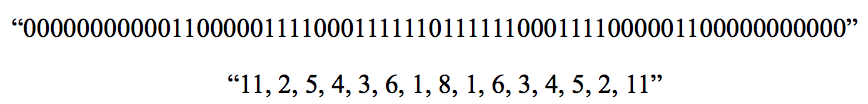
Image compression

* Redundancy: character distribution, character repetition, high usage pattern
* Trade-off in lossless: coding efficiency (compression ratio), coding complexity (memory, operation), coding delay
  + Algorithm: Lzw, Run Length Coding, Huffman
* Entropy coding: entropy is lower bound for rate in lossless coding of digital image
  + Short keyword => high probability

## Run Length Coding (RLC)

* Only use horizontal correlation



## ENTROPY ENCODER

### Huffman

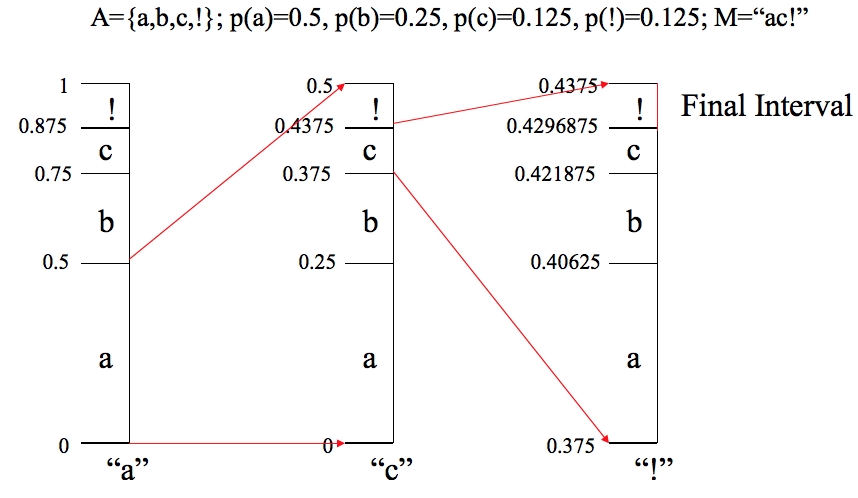
* Order by probability
* Contraction process with 2 smallest prob symbols form new composite symbol
* Other variance : Shannon Fano (split from big to smaller group)

### Universal code

* Predefined codes
* Need order of probability value

### Arithmetic coding

* More efficient than Huffman coding
* Not require integer number of bit. Just map to a float

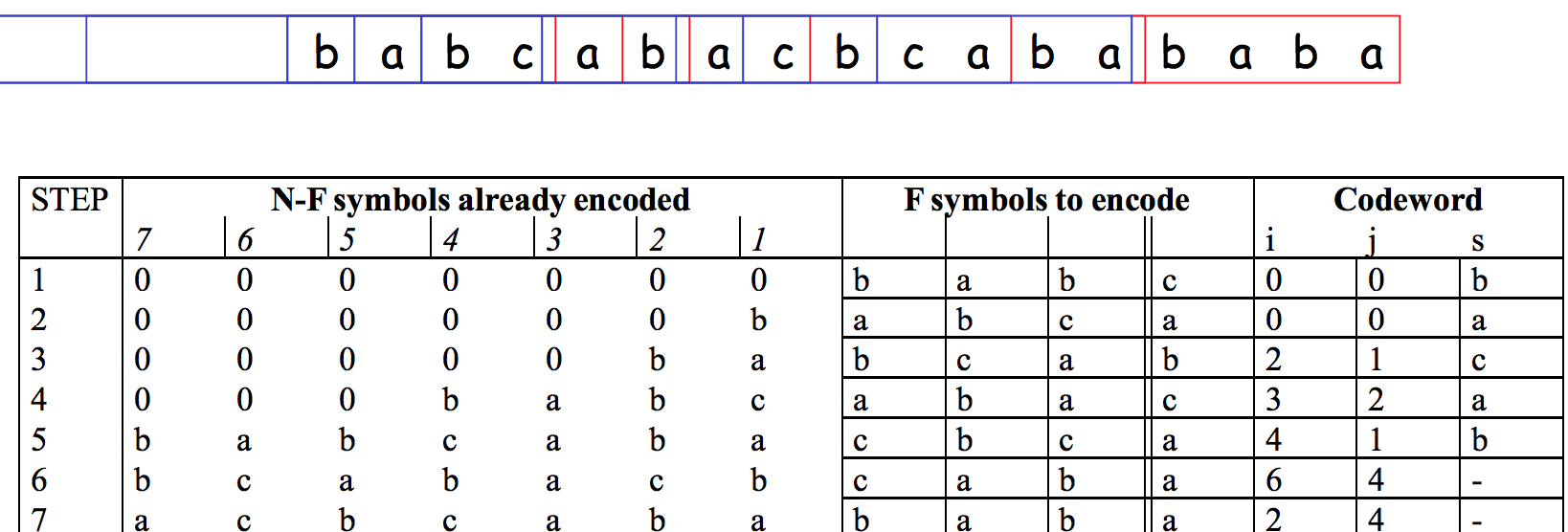


## Adaptive and non-adaptive

* Non adaptive: probability of sysbols is fixed at the beginning
  + Static 1-pass (p known)
  + Static 2-pass: read all data to compute Huffman code -> re-read use code to compress
* Adaptive: the probability of symbol depend upon the symbols decoded in that ime
  + Encoder and decoder have known p
  + Agree on procedure

## LZW

* Create table for coding
  + Read next symbol K
  + If table contain wK => w=wK
  + Else output code w => add wK into table => w = K
* LZ77



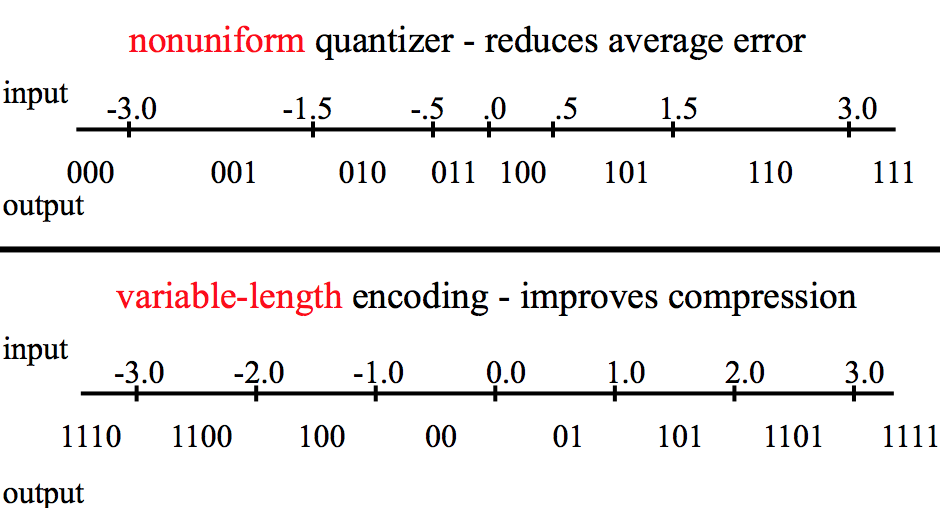
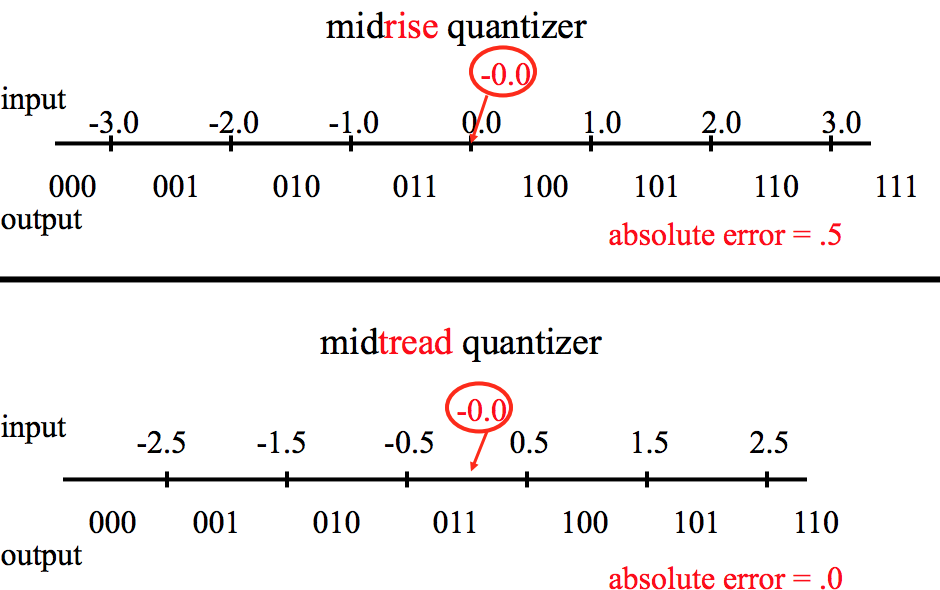
# JPEG encoding

* Input component’s sample are grouped into 8x8 blocks (image have many 8x8 blocks)
* Each block transform into set 64 values (corresponding to DCT coefficients) (1DC +63 AC)
* Each of 64 coefficient then **quantized (**depend on position of coefficient)
* The sequence of DC coefficient is ecoded by **DPCM**
* 63 AC quantized -> 1 dimension by zig-zag -> RLC

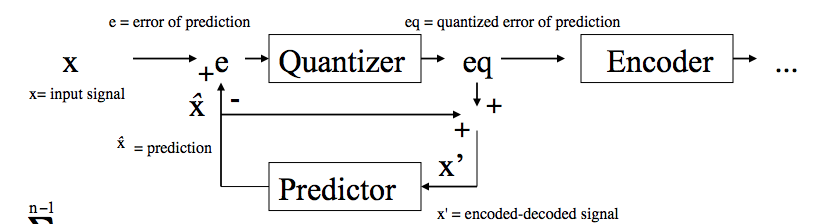
## DCT

* linear combination 64 element blocks can express any image block (8x8)

## Scalar quantization



## DPCM – Adaptive Pulse Code Mudulation



* Effected by Slope overload (function fluctuate greatly but error coding (slope) cannot be greater than threshold) and Granular Noise (output signal always increase or decrease after quantized)
* ADPCM use

## Distortion

* Compression: REDUNANCY (reversible) + IRRELEVANCY (irreversible, lossy)
* Mean square error
* PSNR, SNR, wPSNR. SNR = (db)
  + wPSNR will be higher for image have complex texture ( fur of animal vs the sky)
* BETTER RESULT WHEN PSNR > 38

## Modes of operation

* Sequential DCT-base: send sequentially all inf each block (8x8)
* Progressive (successive): send basic inf of all blocks then detail inf later
* Hierarchical:
  + Downsampling -> encode reduce size -> de-docde and upsample (interpolate)
  + Compute image different and encode it
  + Continue with next scale

## Color Images

* Transform into YUV, YCrCb, RGB
* Non-uniform Quantization: eye is sensitive with low frequency

## Interlaced GIF

* Image is displayed increasingly in several pass
* The scan line is store in unusual order

# JPEG 2000 vs JPEG

* Region of interest
* Better robust against error
* More flexibilities
* Lossy and lossless modes
* Better performance
* Can manage computer-generated images

## Encoding

* Original data -> preprocessing -> discrete wavelet transform -> uniform quantizer with deadzone -> apdaptive binary arithmetic coder -> bit stream organization -> compressed image data

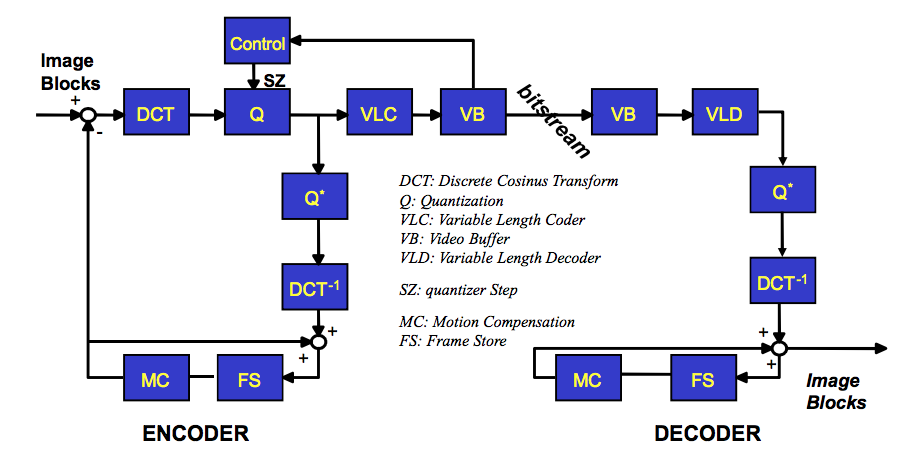
|  |  |
| --- | --- |
| FOURIER | WAVELET |
| Break up a signal into sin wave of various frequency | Into shifted and scale version of origin wavelet |
| From time-based domain to frequency-based domain | From time-based domain to time-scale-based domain |
| Loses time information | Retains time information |
|  | Provides good compression and de-noising  Performs local analysis |

* Component Transform. Reversible Component Transform (lossless) and YCbCr transform (lossy)
* Wavelet transform (5x3 filter lower complexity and lossless, 9x7 highest compression
* Quantization: tradeoff rate and distortion (by division or (lossy) by truncation (lossless)

Video coding

|  |  |
| --- | --- |
| H.261 H.263 H.264 Visioconference | MPEG-x (1,2,4,7,21) Moving Picture Expert Group |
| HEVC ( High Efficiency Video Coding) | |

* MPEG(1,2) Pixel-based -> MPEG 4 Object-based -> MPEG7 Sematic-based
* Compression efficiency gain 50% every 5 years
* Resolution ( number pixels in regions) – Definition (range of value in that pixel)
* Video sequence -> group of picture (GoP) -> Frame -> macroblock (8x8pixel 4 Illumination (Y) and Chromatic (Cr,Cb)



* Video buffer ensure constant target bit-rate output is produced by encoder
* FS: store motion-compensation pixel from previous frame (I- P- picture)

# Motion estimation compensation (block-matching BM)

* For each block B\_t current picture t => find B\_t-1 is similar to B\_t
  + Search limit in a area
  + Displacement is represented by vector
* Cost function, Mean-absolute different, Cross-correlation function (cos similiarity), Mean square, pixel difference
* Optimization version
  + Try on 8 neighbour (distance = 3). Find most similar one
  + Then do it again with distance 2 and 1 => the last similar one is optimum

# Group of Picture

* Intra picture, Predicted picture (coded using motion-compensation from previous I or P), Bidirectional ( Bidirectional coded from both past and future I and P)
* N distance between I, M distance between P
  + N=1 => MJPEG ( send fully sequence of image)
  + M=1 => MPEG1 or H.261
  + Practice M=3 N=12 (EU) or M=3 N=15 (USA) (because of electric frequency)

# Extension

## H.263 vs 261

* Sub-pixel precision (Motion Compensation)
* Options (Unrestricted Motion Vector, Advanced Prediction, Syntax-based Arithmetic coding, PB)
* Support video format (SQCIF, 4CIF, 16CIF)

## H.264

* Video is coded as one or more slide, contain macroblocks (16x16 luna, 8x8 chroma)
* Slice: skip run + macroblock (mb\_type,mb\_predict, residual)
  + Skip run: skip some macroblocks, they can be calculated by Motion compensation from neighbour -> fill to decoded frame
  + Mb\_type: intra or inter

### Intra block prediction

* Spatial prediction: from neighbour sample using mode (vertical, horizontal, DC,…)
* Residual: dif from predicted block and original block

### Inter block prediction

* Temporal prediction: from one or more previously encoded frame
* Macroblock partition (16x16) -> sub-macroblock => each partition is predict from same area in reference picture

### Deblocking filter

* Apply into Macroblock to reduce blocking distortion
* Filter strength based on intra coded, MB boundary using different reference frame or different motion vector

## MPEG-2

* Levels and profiles: stipulate conformance of algorithm between levels
* Scalable encoding: SNR (quality), spatial and temporal

### Profile

* Simple: YUV 4:2:0, Non-scalable coding supporting functionality (NSC) for Random Access
* Main: YUV 4:2:0, Same **simple** but NSC for B-picture prediction
* SNR scalable: YUV 4:2:0, same main, SNR scalable coding
* SPATIAL scalable: YUV 4:0:0 Spatial scalable coding
* High: YUV 4:2:2 same spatial scalable, 3 layers with SNR and and spatial scalable coding

### Y:U:V

* Force chunks of pixel share chromatic
* Second number – number of chromatic values for top row
* Third number – number of chromatic value for bottom row
* 4:2:0 => bottom row share color with top row

## MPEG-4

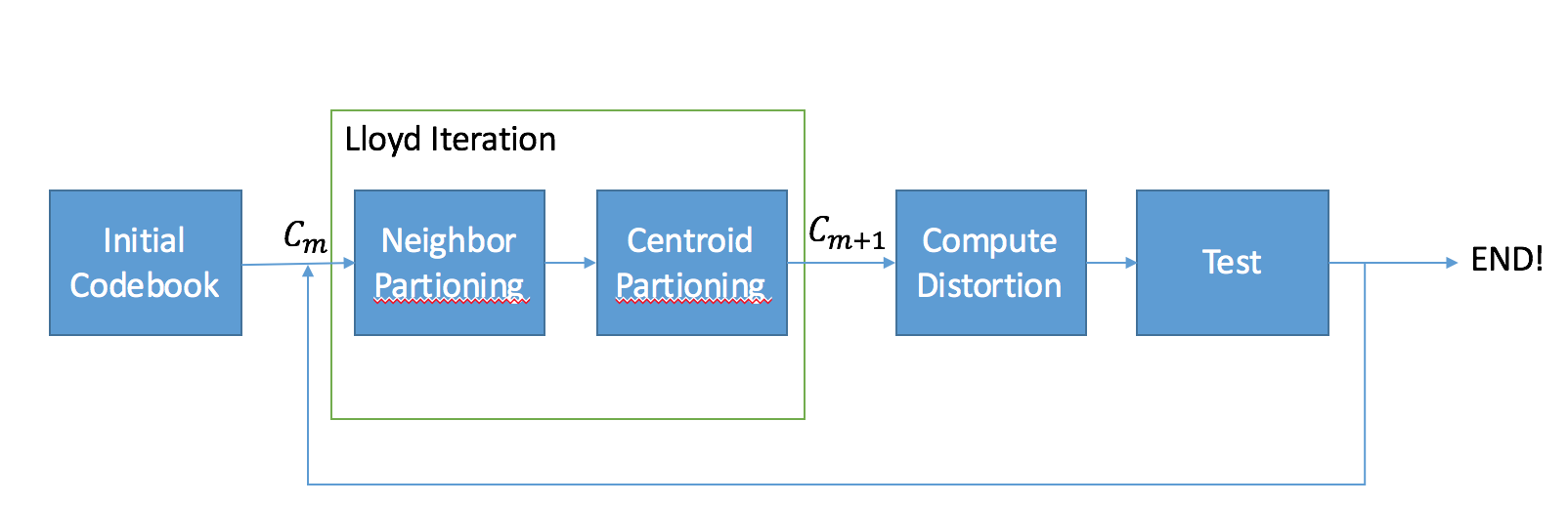
* Shape – Texture
* Video object & video object plane

Vector Quantization

* Goal: minimize global distortion
* To represent your vector by vocabulary in codebook
  + Find vector in code book that “close” to your vector
  + Get index of vector in code book
* Optimal quantization (Lloyd Max) - Joint optimization
  + Given codebook find best partition=> nearest neighbour condition
  + Given the partition find best codebook -> centroid condition
* Scalar quantization – vector quantization

# Generlized Loyd Algorithm (GLA)

* Begin with initial codebook C\_1,
* Given code book C\_m perform the Lloyd iteration to generate the improved codebook C\_(m+1)
* Compute average distortion C\_(m+1) if is good enough and stop



# Linde, Buzo, Gray (LBG)

* Goal: mean distortion decrease, local minimum, choise of the initial cookbook
* Idea: start with code book of size 2
  + Each iteration each vector split into 2 vectors (+/- a)
  + Optimal codebook with 2n vector