ESE532: System-on-a-Chip Architecture

Day 16: March 20, 2017
MPEG Encoding

Today

MPEG Encoding

- Project
- Motion Estimation
- DCT
- Entropy Encoding

Message

- Compression is about exploiting (eliminating)
  - Redundancy (lossless)
  - Things humans don't notice much (lossy)
- Redundancy in video
  - Temporal: data repeated frame-to-frame
  - Spatial: frequency patterns
  - Quantize: high frequencies
  - Entropy encode values (e.g. frequencies)

Project

- Continue to work with MPEG Encoder
  - Speed up as much as possible on Zynq
  - Estimate custom design to achieve real-time for 1080p30: 1920x1080 at 30 fps
- Groups of 2 – you select partners
- Next 5 weeks: project report 4/21
- Weekly milestones

Why MPEG Encode?

- Original intent: different problem
- Experience:
  - MPEG harder than intended for first half warmup
  - Probably right complexity for project
- Avoid giving you a project too complex
- Want you to succeed at accelerating

Expect

- You will need to rewrite more of the code than you have so far
- C written for reference not organized or written correctly for acceleration

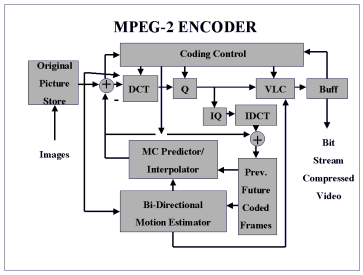
Why not do this before?

Expected

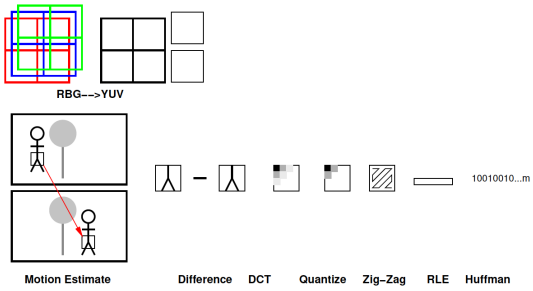
- wouldn't need to get this deep into it for the homework series.
  - Could focus on bottleneck pieces and small snippets
    - …as we did for HW3
- was familiar

MPEG Encode

MPEG Encoder



MPEG Encoder



Temporal Redundancy

Frame-to-Frame

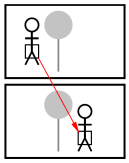
- How are two immediately adjacent video frames likely to be related?
- Common cases
  - Fixed camera (e.g. security camera)
  - Panning camera
  - Object moving left to right

Idea: Temporal Coding

- If part of frame not change
  - Avoid resending
- If something moves in frame
  - Avoid resending
    - Identify where to find in previous frame

MPEG Simplification

- Break image into 16x16 macroblocks
- Try to find macroblock in previous frame
  - (or subsequent frame)
- Ideal
  - Find perfect match
  - Only send location of match



Preclass 1

- How big is pointer for 1920x1080 frame?
- How much cheaper than sending 16x16 macroblock with 8b pixels?

Macroblock Match

Finding "identical" macroblock:

- Where is this likely to work?
- What complicates?

Challenge

- May not be exactly identical
  - Shadows change
  - Something moves over part of macroblock
  - Thing in macroblock distorts
    - Person talks, turns, rotates
    - Object moves toward/away from camera

Not Identical

- Send difference
  - Maybe less information than new macroblock
    - E.g.
      - background occluded (exposed) on side
      - Eye opens (closes)

Finding

- Find matching macroblock
  - Or best matching (hence `dist`)
- How go about finding?

Local Search

- Hypothesize that the macroblock is nearby in previous image.
  - Why might be good hypothesis?
- Simple
  - Exhaustively search within some distance
- Why expensive?

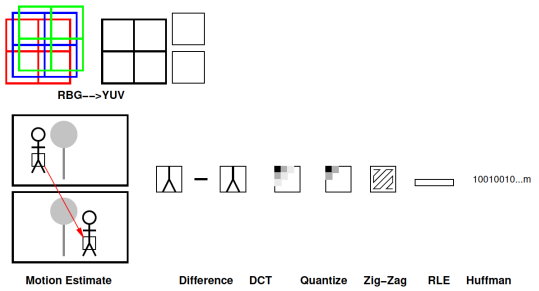
Limited Motion

- After a number of frames, less likely to find in previous frame
  - New objects, turns, rotation, scale…
- Send new, non-motion coded reference

MPEG Frames

- Interleave I, P, and B frames
  - I – not motion coded
  - P – forward prediction frames
    - Predict from previous frame
  - B – backward prediction frames
    - Predict from future frames
- E.g. (IBBPBBPBBPBB)*

MPEG Encoder



Spatial Redundancy

- Can code more efficiently in frequency domain
- Humans notice low frequency components more than high frequency
  - Can use fewer bits for high frequency components

Day 14

Fourier Transform

- Identify spectral components
- Convert between Time-domain to Frequency-domain
  - E.g. tones from data samples
  - Central to audio coding – e.g. MP3 audio

$$Y[k] = \sum_{j=0}^{n-1} \left(X[j] e^{-2i\pi \frac{k}{n}} \right)$$

Discrete Cosine Transform

- Similar to FFT
- Only uses Cosine (real part)
- (boundary condition)

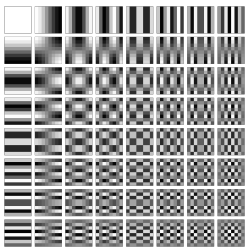
$$Y[k] = \sum_{j=0}^{n-1} x[j] \times \cos\left[\frac{\pi \times k}{N} (j + 0.5) \right]$$

$$Y[k] = \sum_{j=0}^{n-1} \left(X[j] e^{-2i\pi \frac{k}{n}} \right)$$

2D-DCT Basis

- Actually compute 2D-DCT
  - DCT in each spatial dimension
  - On 8x8 blocks
- Can be viewed as a basis transform
  - Re-expressing the 8x8 block in terms of 8x8 selection of x and y frequencies

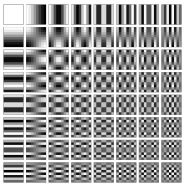
2D-DCT Basis



- Basis vectors, so all 64 components are orthogonal
- Any 8x8 spatial is a weighted sum of these 64 components

Preclass 2

- Represent all 23 pixel block?
- Non-zeros for horizontal stripes?
- Non-zeros for vertical stripes?



DCT Benefits

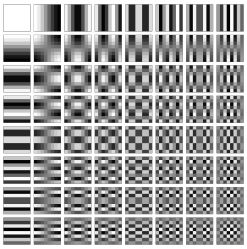
1. Concentrates weight in low frequency components (upper left)
2. High frequency components can be dropped or represented with few bits
   - Not matter so much to human perception

DCT Example

Source: http://img.tomshardware.com/us/1999/09/24/video_guide_part_3/dct.gif

2D DCT Calculation

- Brute force like this
- $O(n^4)$
  - n^2 entries
  - Each require n^2 terms



$$Y[k,m] = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} x[i,j] \times \cos\left[\frac{\pi \times m}{N}(i+0.5)\right] \times \cos\left[\frac{\pi \times k}{N}(j+0.5)\right]$$

2D DCT Calculation

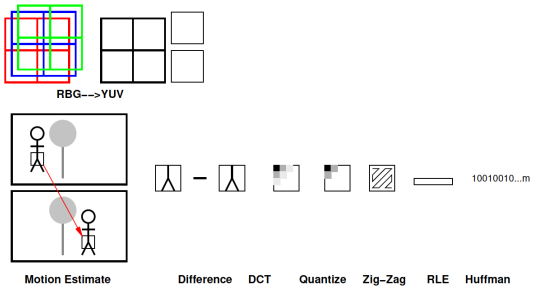
`fdct`

- Decompose, like FFT
- Perform 1D DCT across rows $O(n^3)$
  - n^2 results, each $O(n)$
- Perform 1D DCT across columns $O(n^3)$

$$Y[k] = \sum_{j=0}^{n-1} x[j] \times \cos\left[\frac{\pi \times k}{N}(j+0.5)\right]$$

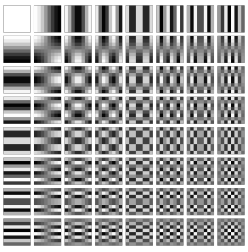
$$Y[k,m] = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} x[i,j] \times \cos\left[\frac{\pi \times m}{N}(i+0.5)\right] \times \cos\left[\frac{\pi \times k}{N}(j+0.5)\right]$$

MPEG Encoder



Quantize

```
EXTERN unsigned char
default_intra_quantizer_matrix[64]
#ifdef GLOBAL
=
{
  8, 16, 19, 22, 26, 27, 29, 34,
  16, 16, 22, 24, 27, 29, 34, 37,
  19, 22, 26, 27, 29, 34, 34, 38,
  22, 22, 26, 27, 29, 34, 37, 40,
  22, 26, 27, 29, 32, 35, 40, 48,
  26, 27, 29, 32, 35, 40, 48, 58,
  26, 27, 29, 34, 38, 46, 56, 69,
  27, 29, 35, 38, 46, 56, 69, 83
}
#endif
```



Quantize

```
EXTERN unsigned char
default_intra_quantizer_matrix[64]
#ifdef GLOBAL
=
{
  8, 16, 19, 22, 26, 27, 29, 34,
  16, 16, 22, 24, 27, 29, 34, 37,
  19, 22, 26, 27, 29, 34, 34, 38,
  22, 22, 26, 27, 29, 34, 37, 40,
  22, 26, 27, 29, 32, 35, 40, 48,
  26, 27, 29, 32, 35, 40, 48, 58,
  26, 27, 29, 34, 38, 46, 56, 69,
  27, 29, 35, 38, 46, 56, 69, 83
}
#endif
```

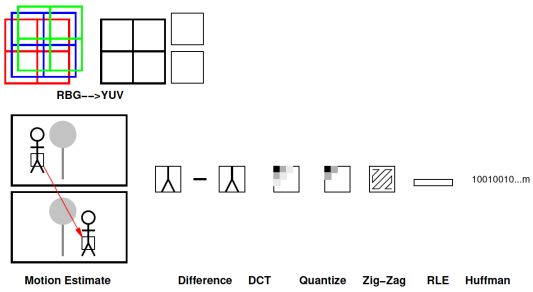
- Divide by entry in quantizer matrix before change to integer
- Most bits to upper left
- Few bits to lower right
  - More things threshold to zero

Quantize and Precision

```
EXTERN unsigned char
default_intra_quantizer_matrix[64]
#ifdef GLOBAL
=
{
  8, 16, 19, 22, 26, 27, 29, 34,
  16, 16, 22, 24, 27, 29, 34, 37,
  19, 22, 26, 27, 29, 34, 34, 38,
  22, 22, 26, 27, 29, 34, 37, 40,
  22, 26, 27, 29, 32, 35, 40, 48,
  26, 27, 29, 32, 35, 40, 48, 58,
  26, 27, 29, 34, 38, 46, 56, 69,
  27, 29, 35, 38, 46, 56, 69, 83
}
#endif
```

- What does quantization mean for DCT precisions?

MPEG Encoder

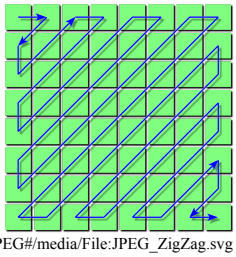


Value Diminish to Lower Right

Source: http://img.tomshardware.com/us/1999/09/24/video_guide_part_3/dct.gif

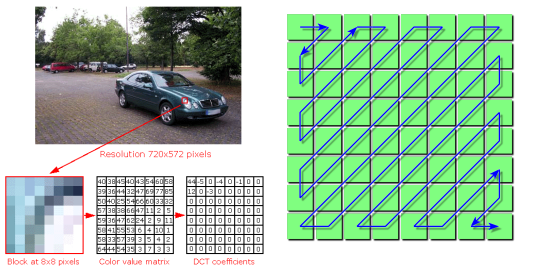
Zig-Zag Order

- Groups zeros (and small constants) together at end



Source: https://en.wikipedia.org/wiki/JPEG#/media/File:JPEG_ZigZag.svg

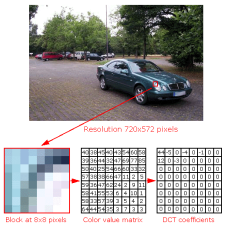
Value Diminish to Lower Right

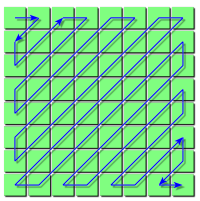


Source: http://img.tomshardware.com/us/1999/09/24/video_guide_part_3/dct.gif

Source: https://en.wikipedia.org/wiki/JPEG#/media/File:JPEG_ZigZag.svg

Value Diminish to Lower Right



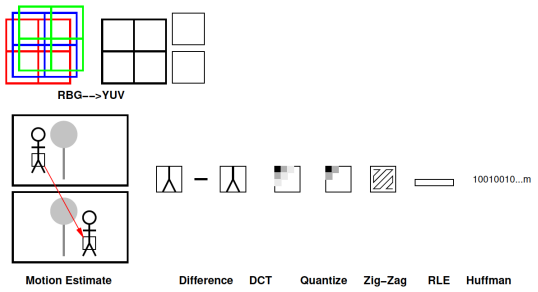


45 -5 12 0 0 0 -4 -3 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ....

Source: http://img.tomshardware.com/us/1999/09/24/video_guide_part_3/dct.gif

Source: https://en.wikipedia.org/wiki/JPEG#/media/File:JPEG_ZigZag.svg

MPEG Encoder

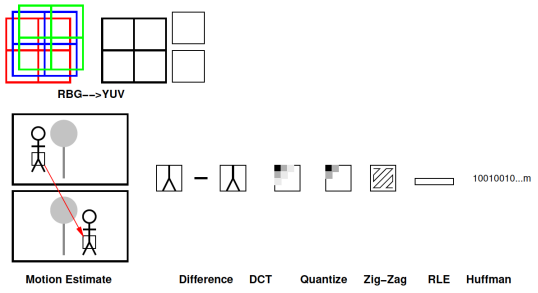


Preclass 3

- How many numbers in the Run-Length encoding?

45 -5 12 0 0 0 -4 -3 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ....

MPEG Encoder



Preclass 4

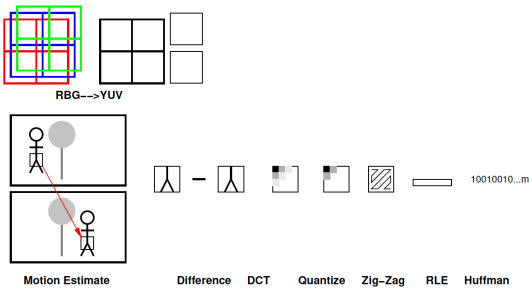
- How many bits to Huffman encode?
  - Compare to 8b encoding of RLE?

Huffman

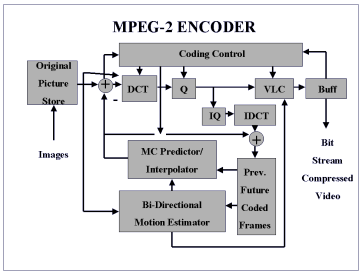
- Bit twiddling for Huffman coding inefficient on word-wide processor

```
void putbits(val,n)
{
  mask = 1 << (n-1); /* select
  for (i=0; i<n; i++)
  {
    outbfr <<= 1;
    if (val & mask)
      outbfr |= 1;
    mask >>= 1; /* select next
    outcnt--;
    if (outcnt==0) /* 8 bit bu
    {
      output_buf[bytecnt] = ou
      outcnt = 8;
      bytecnt++;
      if (bytecnt == OUTPUT_BU
        error("Output buffer t
    }
  }
}
```


MPEG Encoder



MPEG Encoder



Big Ideas

- Compression is about exploiting (eliminating)
  - Redundancy (lossless)
  - Things humans don't notice much (lossy)
- Redundancy in video
  - Temporal: data repeat frame-to-frame
  - Spatial: frequency patterns
  - Quantize: high frequencies
  - Entropy encode values (e.g. frequencies)

Admin

- Project and Project Analysis Milestone out
- Project Analysis Milestone due Friday
  - Including teaming