H.264 Luma Predictor

Maxine Lee, Alex Moore

May 17, 2006

Integrated Systems Group
Massachusetts Institute of Technology

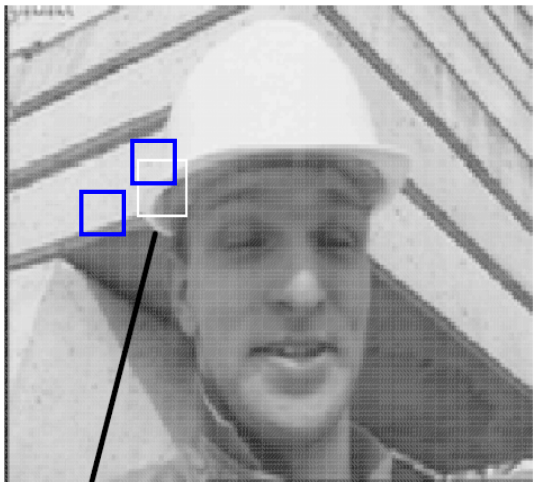
Why H.264?

- End-to-end protocol
- Better compression
- Designed for efficient encoding
- ITU standard
- It's on your iPod

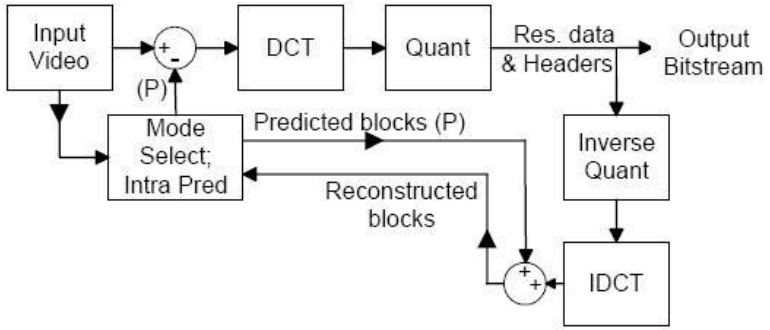
Project Scope

- Prediction module of H.264 Encoder
  - Intraframe Prediction
  - Interframe Prediction
  - Transforms
  - Luma only (no color information!)
- Why?
  - 85%+ of encoder computation time
  - Rich problem with lots of exploration

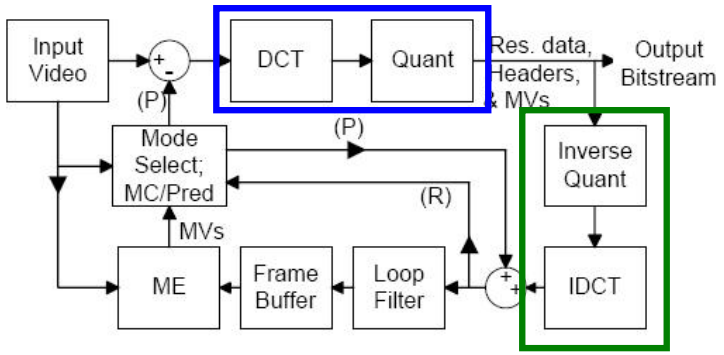
Intraframe Prediction Motivation



Intraframe Prediction Block Diagram

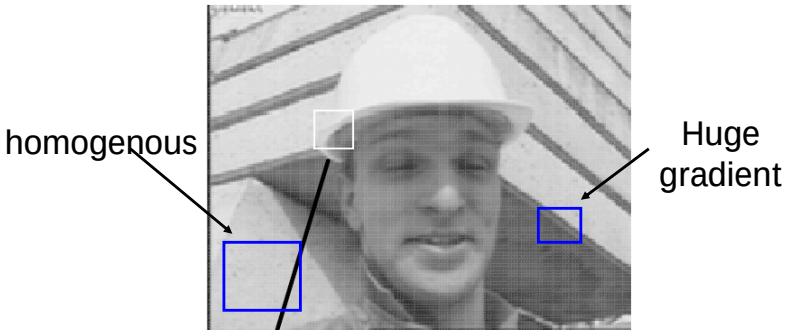


Interframe Prediction



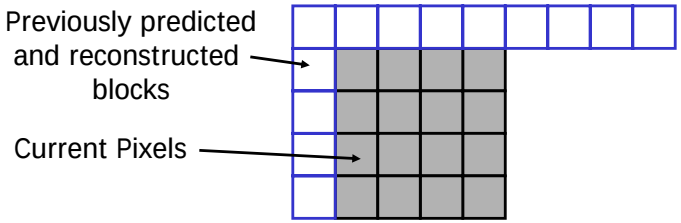
Intra-Frame Prediction

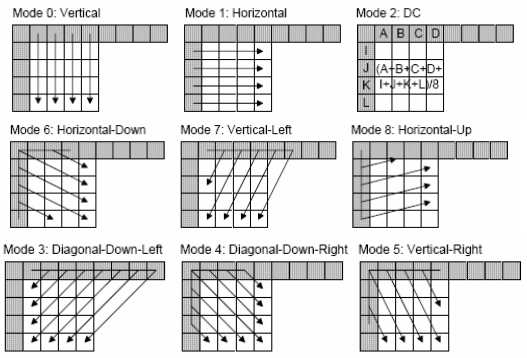
- Use spatial similarities to compress each frame
  - Use neighboring pixels to make a prediction on a block
  - Transmit the difference between actual and predicted
  - Tradeoff : prediction accuracy vs. # control bits



- H.264 Answer : 4x4 and 16x16 prediction !

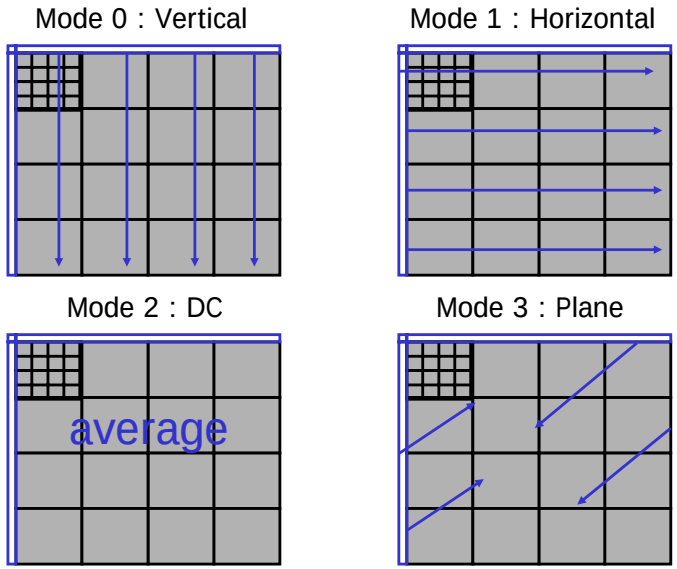
Intra – 4x4 Prediction





- 9 prediction modes
- Prediction proceeds left to right, top to bottom
- When not all boundary pixels available (i.e. we're at border of picture), can't predict with all the modes

Intra - 16x16 Prediction

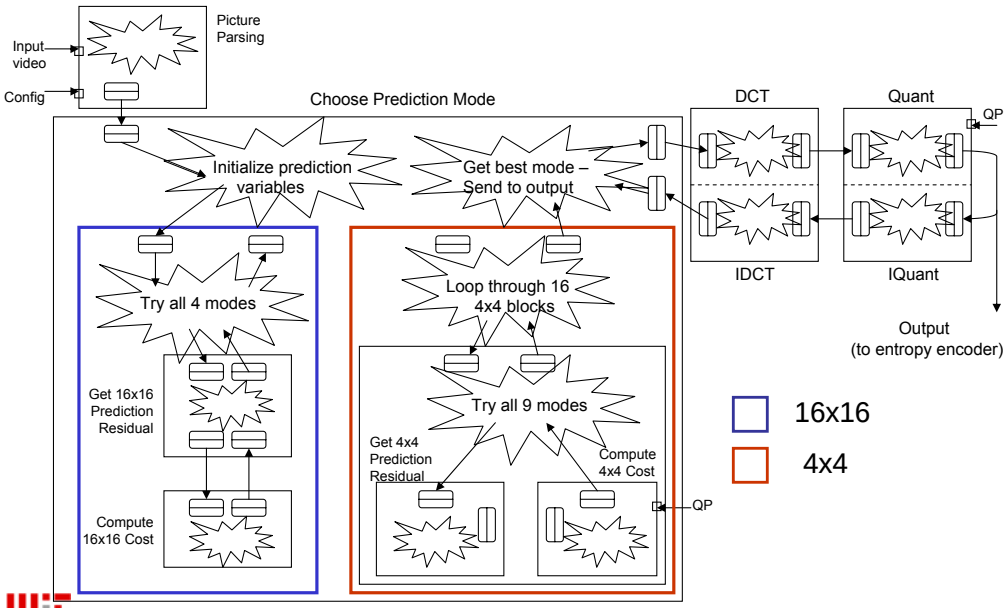


Advantages/Disadvantages

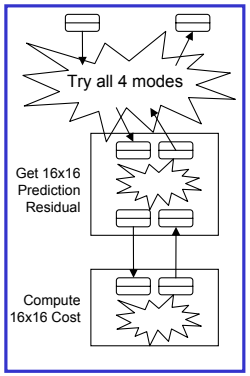
Intra 16x16	Intra 4x4
❑ Good for smooth areas	❑ Good for detailed areas ❑ Lots of options
❑ 4 modes = 2	❑ 9 modes = 4 bits for every 16 pixels (!)

- Encoder's job to compare options and pick the best
  - Exhaustive search …
  - Uses a cost function to compare different modes

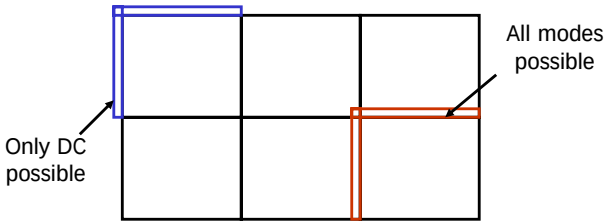
Block Diagram (Baseline)



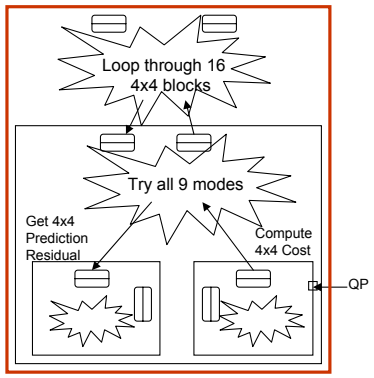
Intra – 16x16 Considerations

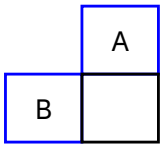


- Process
  - Loop through the available*** modes
  - Generate the prediction
  - Compute cost of residual
    - Cost ~ SAD (sum of absolute diff)
- ***What's available?
  - Depends on location in the frame!



Intra – 4x4 Considerations



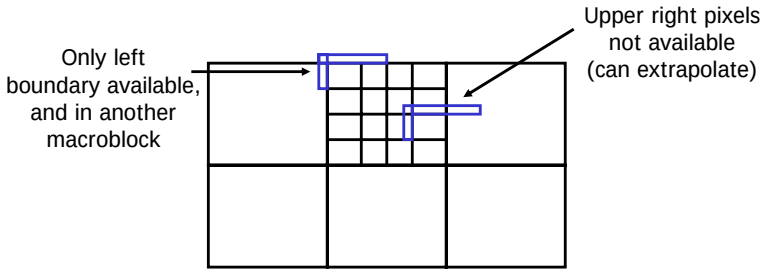


- Process:
  - Loop through all 16 blocks
  - For each block, loop through available modes
  - Get ***cost = SAD + 4*P*λ(QP)
  - Pick best mode – send to DCT
  - Save reconstructed 4x4 block, so you can use it to predict the next 4x4 block
- Cost :
  - f (QP), since overhead bits hurt more with higher compression
  - P : most probable mode

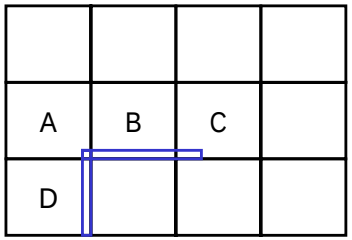
Overhead!!!

Extra Concerns with Intra 4x4

- Which boundary pixels do you use?
  - Boundary depends on where in the picture you are AND which 4x4 block you're working on



Storing Boundary Pixels



- To predict current macroblock, need pixels from FOUR neighbors (A-D)
- D can be stored in a register, since it is immediately used
- Pixels for previous row (A-C) have to be stored in a register file
- Also save A in register to limit regfile reads to 2

Synthesis Numbers

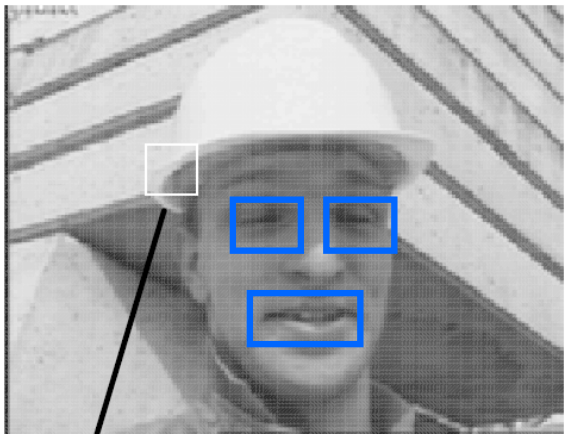
Note: not P+R – not enough RAM / hard disk
(ask us tomorrow if you're really curious about P+R numbers)

- Total Area = 609,940 um^2

Predictor	66%
DCT/IDCT	10%
Quant (with QP lookup tables)	15%
Misc.	9%

- Clock Cycle = 7.27 ns (quant multiplications)

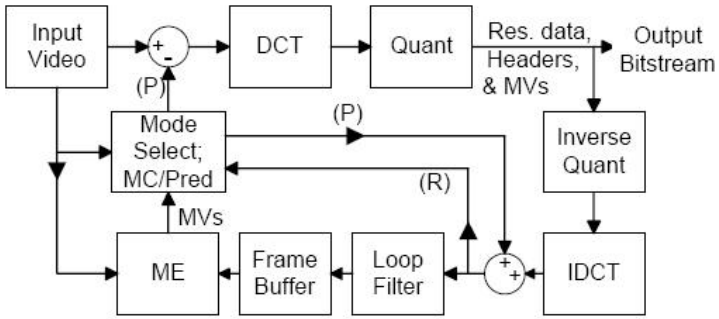
Only Three Regions of Change



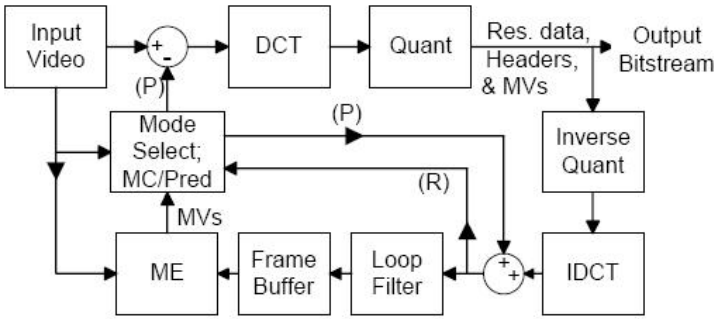
Interframe Prediction

- Use previous frame(s) to predict macroblocks of current frame
- Most of the time, majority of frame isn't moving
- If change within macroblock is sufficiently small, just reproduce it exactly!

Interframe Prediction



Interframe Prediction



Interprediction Algorithm

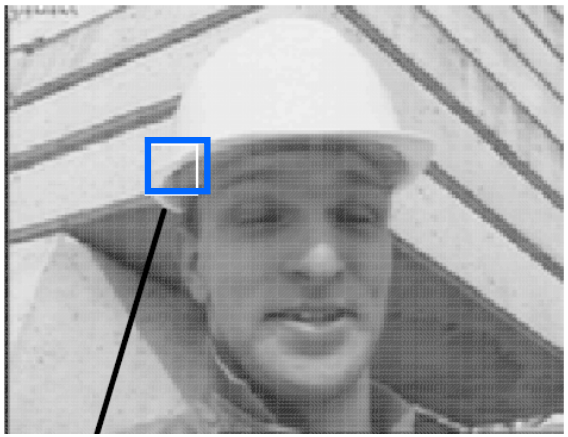
- Use a *motion vector* to predict the current macroblock.
- Start at (0,0) – same block – and calculate error for each motion vector
- Full-Search algorithm. Try all possible motion vectors within a window
- Final prediction will be block given by motion vector with minimum error

Interprediction Algorithm

Interprediction Algorithm

Interprediction Algorithm

Interprediction Algorithm



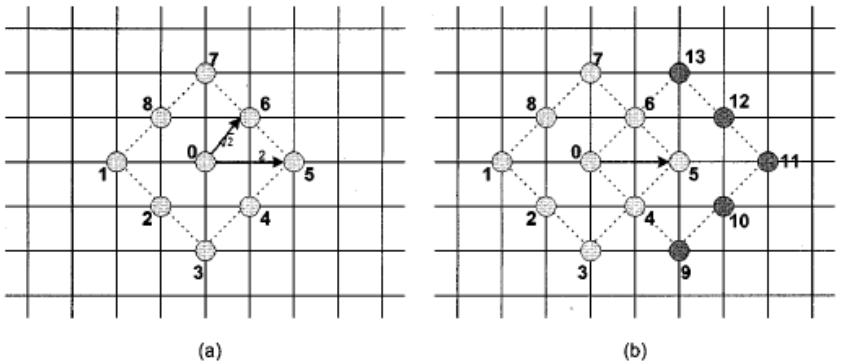
Problem…

- Assume a window size of 16 (conservative)
- 1024 possible motion vectors to check per macroblock (vs. 9 for intra)
- 307200 possible motion vectors per frame!

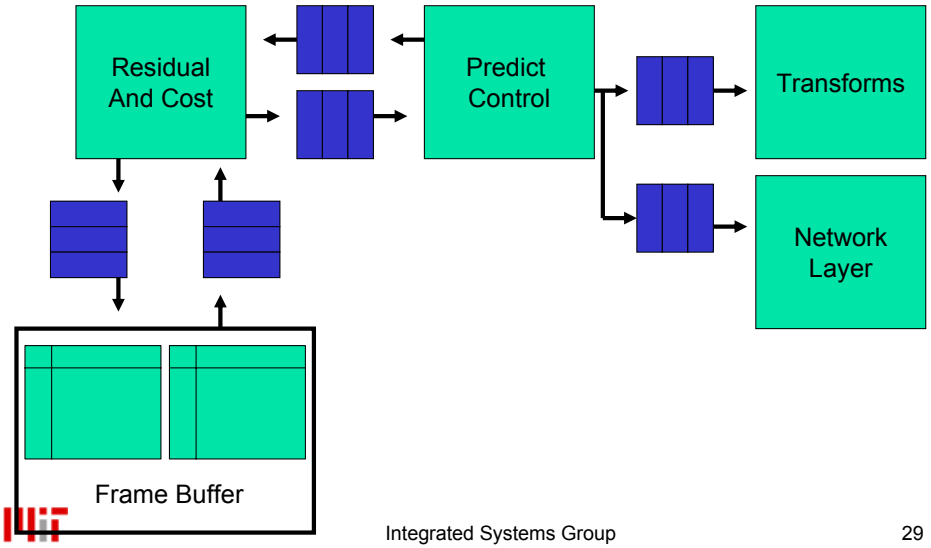
Solution

- A better algorithm! Assume motion estimation gets better as we get closer to ideal motion vector.
- Diamond-shaped algorithm reduces points checked by ~80% with mean error per pixel about 3 (vs about 2) for FS.
- Hexagonal algorithm reduces by another ~35% (3.2 mean error vs 3.0)

Hexagonal Algorithm



Circuit Implementation



Results…

- Results? What Results?
- H.264 predictor ~40x size of SMIPS processor
- Frame buffer adds ~18000 area (+4%)
  - But we're cheating (64x48 video size)
- Interprediction block adds ~35000 area (+7%)
- Performance evaluation TBA

References

[1]Ghandi, M.M., and Ghanbari, M. The H.264 Video Coding Standard for the Next Generation Multimedia Communication. IAEEE Journal.

[2]Richardson, I. H.264 / MPEG-4 Part 10 Tutorial www.vcodex.com.

[3]Malvar, H.S., Hallapuro, A., Karczewicz, M., and Kerofsky, L. Low-Complexity Transform and Quantization in H.264/AVC. IEEE Transactions on Circuits and Systems for Video Technology, July 2003.