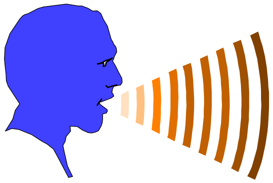


Multimedia Communication Systems II

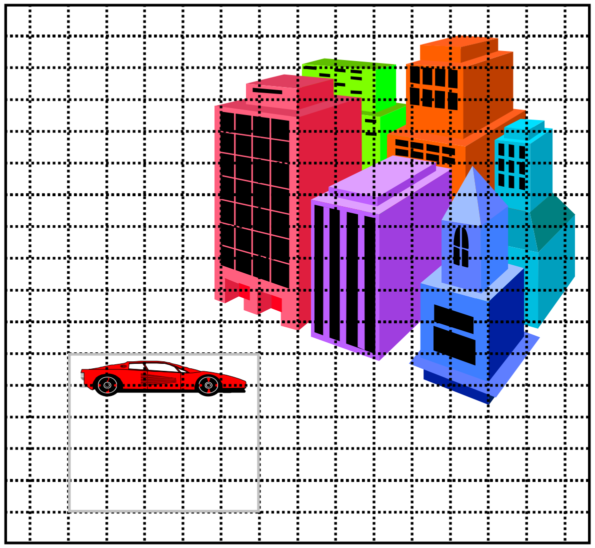
Motion Estimation for Video Coding

Yao Wang
Polytechnic University, Brooklyn, NY11201
yao@vision.poly.edu

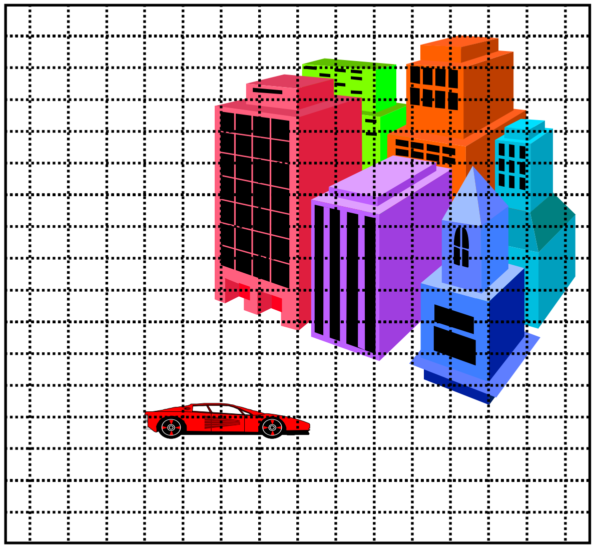
Outline

- Motion estimation for video coding
- Block-based motion estimation:
  - EBMA algorithm, Integer accuracy
  - EBMA algorithm, Half-pel accuracy
  - Complexity analysis
- Demonstration
- Use of motion estimation for video coding

Characteristics of Typical Videos



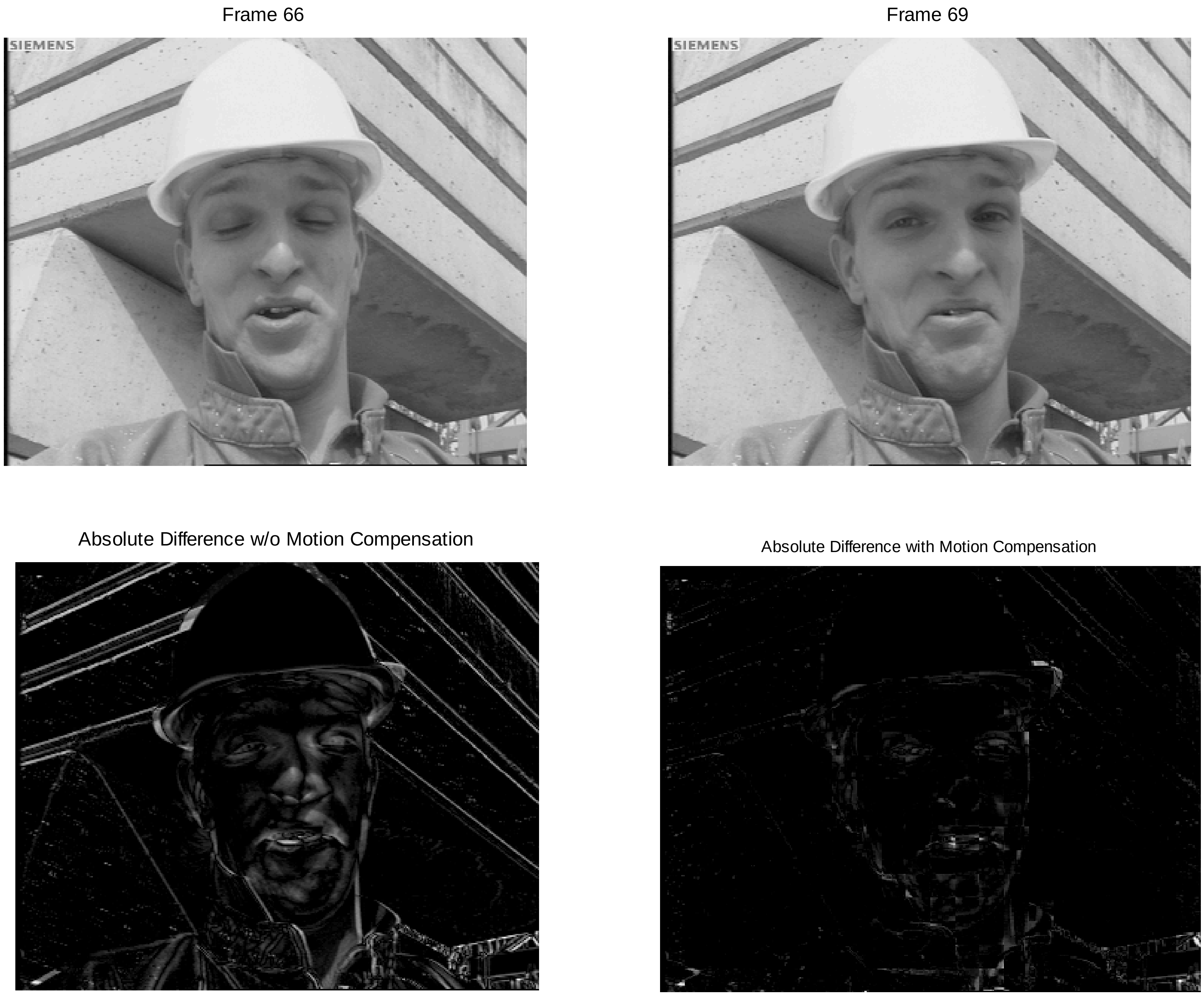
Frame *t-1*



Frame *t*

Adjacent frames are similar and changes are due to object or camera motion

--- Temporal correlation



Frame 66

Frame 69

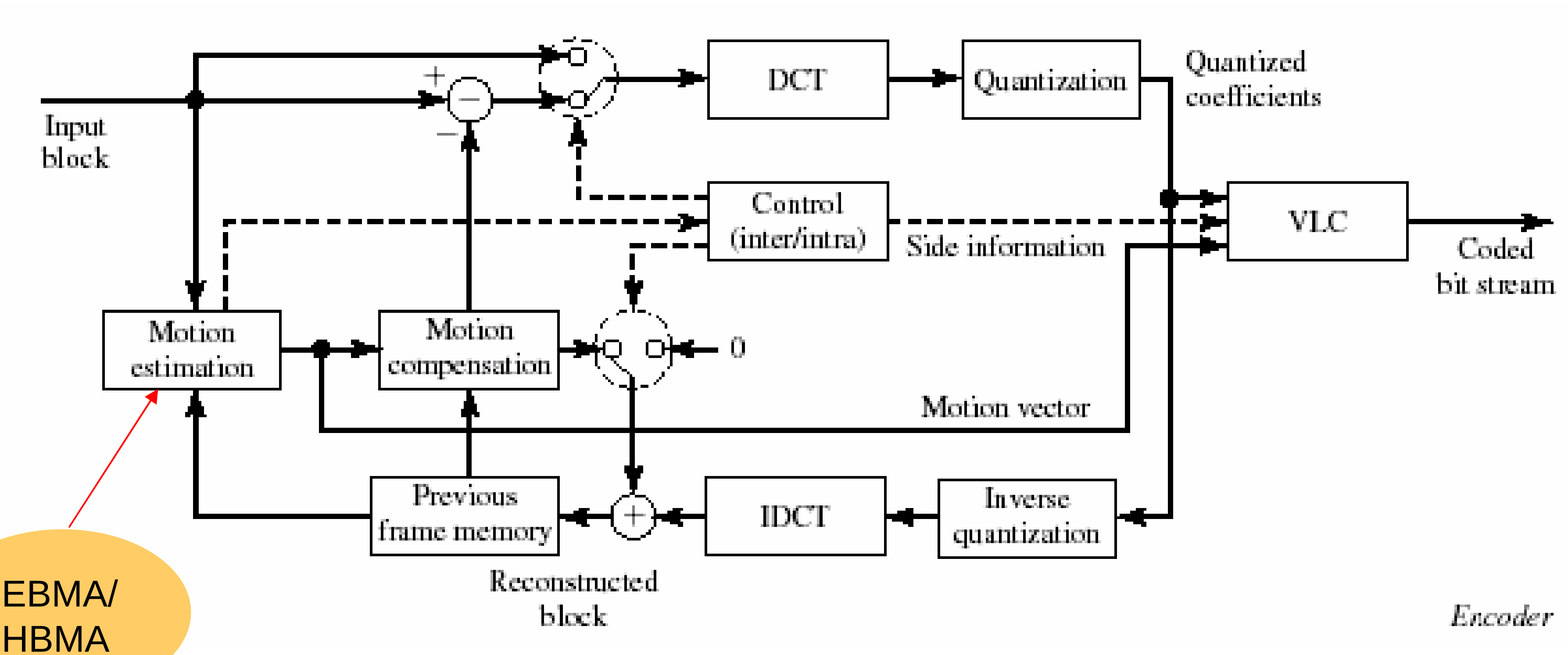
Absolute Difference w/o Motion Compensation

Absolute Difference with Motion Compensation

Key Idea in Video Compression

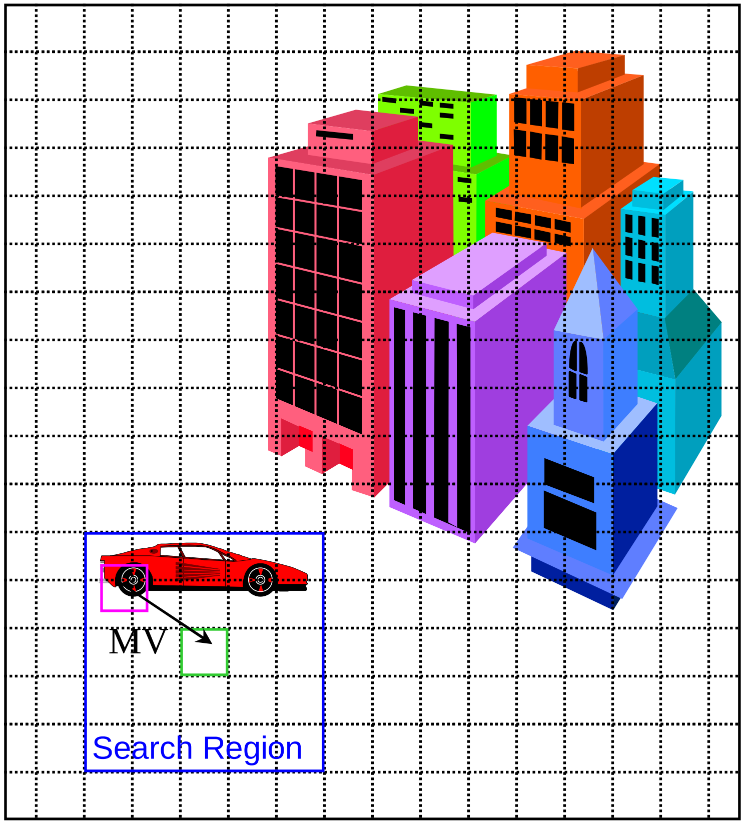
- Predict a new frame from a previous frame and only code the prediction error
- Prediction error will be coded using an image coding method (e.g., DCT-based as in JPEG)
- Prediction errors have smaller energy than the original pixel values and can be coded with fewer bits
- Those regions that cannot be predicted well will be coded directly using DCT-based method
- Use motion-compensated prediction to account for object motion
- Work on each macroblock (MB) (16x16 pixels) independently for reduced complexity
  - Motion compensation done at the MB level
  - DCT coding of error at the block level (8x8 pixels)

Encoder Block Diagram of a Typical Block-Based Video Coder

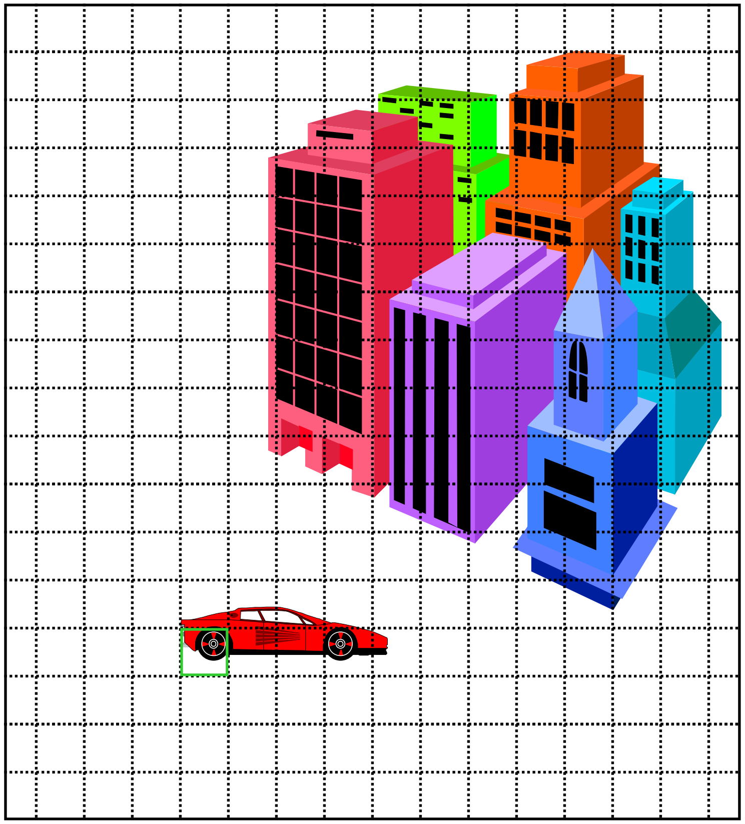


EBMA/
HBMA

Block Matching Algorithm for Motion Estimation



Frame *t-1*
(Reference Frame)



Frame *t*
(Predicted frame)

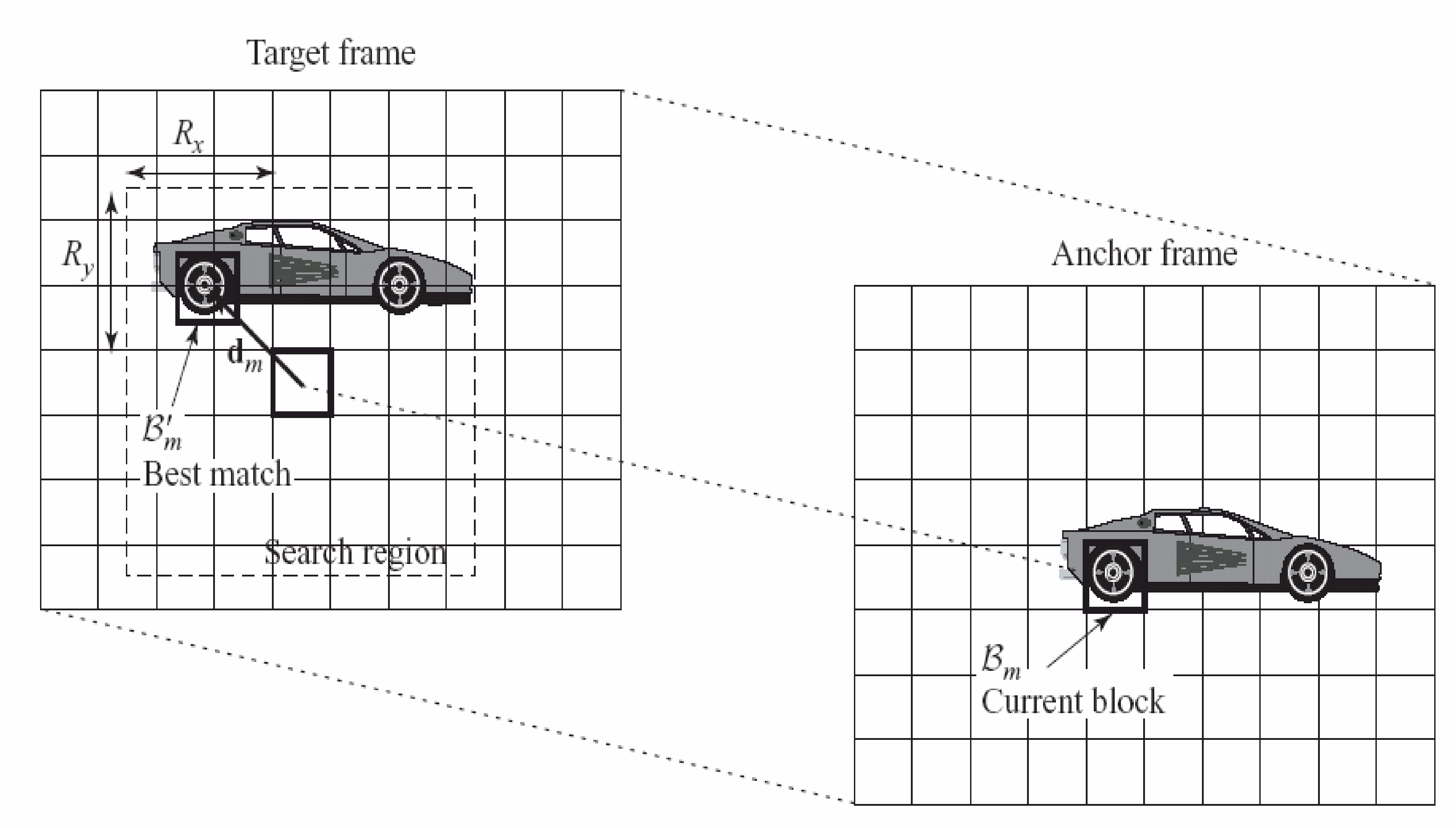
Block Matching Algorithm Overview

- For each MB in a new frame
  - Search for a block in a reference frame that has the lowest matching error
    - Using sum of absolute differences (SAD) between corresponding pels
    - Search range: depends on the anticipated motion range
    - Search step-size: integer-pel or half-pel

$$E_{\mathrm{DFD}}(\mathbf{d}_{\mathrm{m}}) = \sum_{\mathbf{x} \in B_m} \left| \psi_2(\mathbf{x} + \mathbf{d}_m) - \psi_1(\mathbf{x}) \right|^p \to \min$$

  - Displacement between the current MB and the best matching MB is the motion vector (MV)
  - Current MB is replaced by the best matching MB (motion-compensated prediction or motion compensation)

Exhaustive Block Matching Algorithm (EBMA)

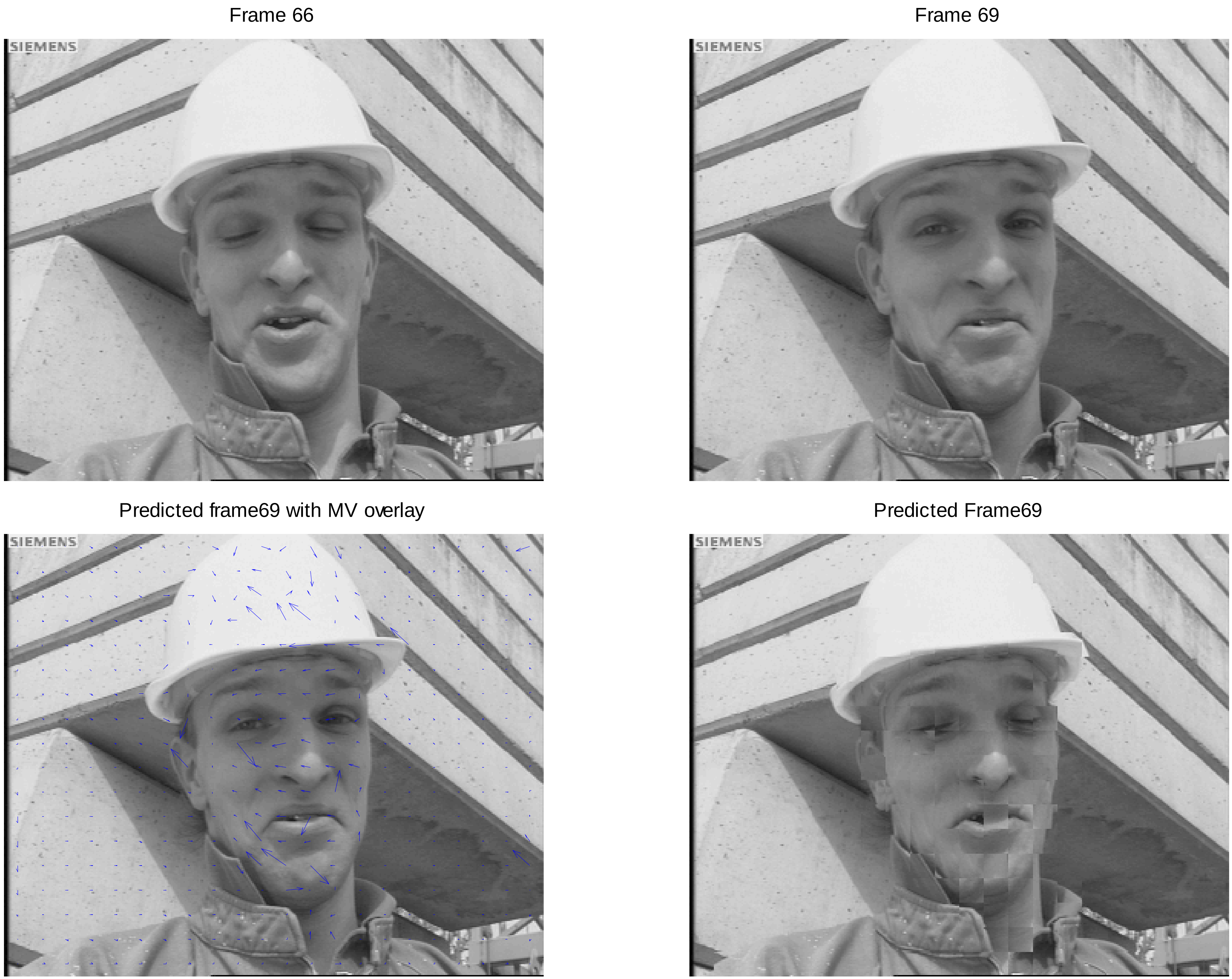


Sample Matlab Script for Integer-pel EBMA

```
%f1: anchor frame; f2: target frame, fp: predicted image;
%mvx,mvy: store the MV image
%widthxheight: image size; N: block size, R: search range

for i=1:N:height-N,
  for j=1:N:width-N  %for every block in the anchor frame
    MAD_min=256*N*N;dx=0;dy=0;
    for k=-R:1:R,
      for l=-R:1:R  %for every search candidate
        MAD=sum(sum(abs(f1(i:i+N-1,j:j+N-1)-f2(i+k:i+k+N-1,j+l:j+l+N-1))));
            % calculate MAD for this candidate
        if MAD<MAX_min
            MAD_min=MAD,dy=k,dx=l;
        end;
    end;end;
    fp(i:i+N-1,j:j+N-1)= f2(i+dy:i+dy+N-1,j+dx:j+dx+N-1);
            %put the best matching block in the predicted image
    iblk=(floor)(i-1)/N+1; jblk=(floor)(j-1)/N+1; %block index
    mvx(iblk,jblk)=dx; mvy(iblk,jblk)=dy; %record the estimated MV
end;end;
```

Note: A real working program needs to check whether a pixel in the candidate matching block falls outside the image boundary and such pixel should not be counted in MAD. This program is meant to illustrate the main operations involved. Not the actual working matlab script.



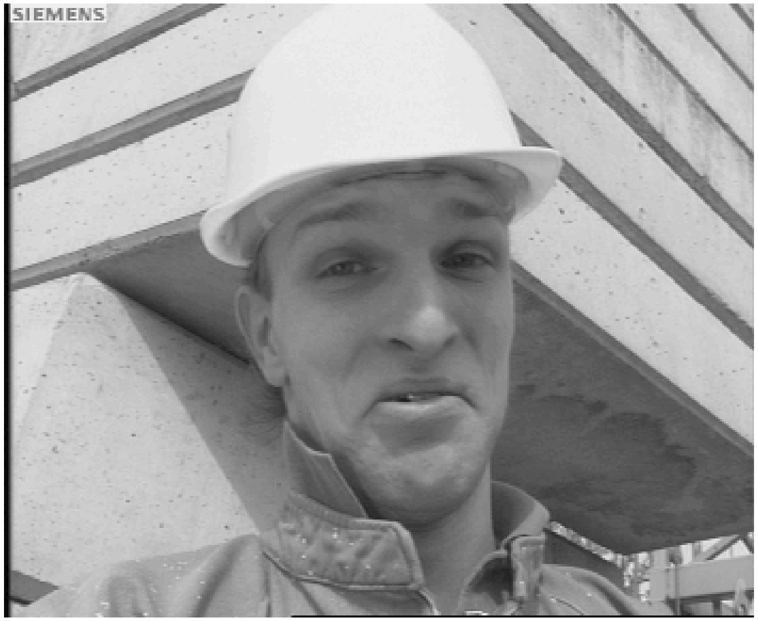
Frame 66

Frame 69

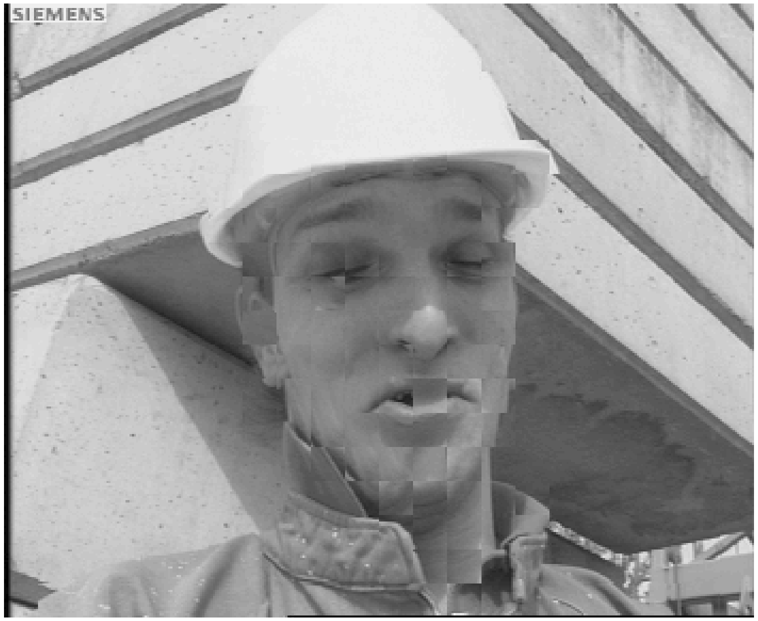
Predicted frame69 with MV overlay

Predicted Frame69

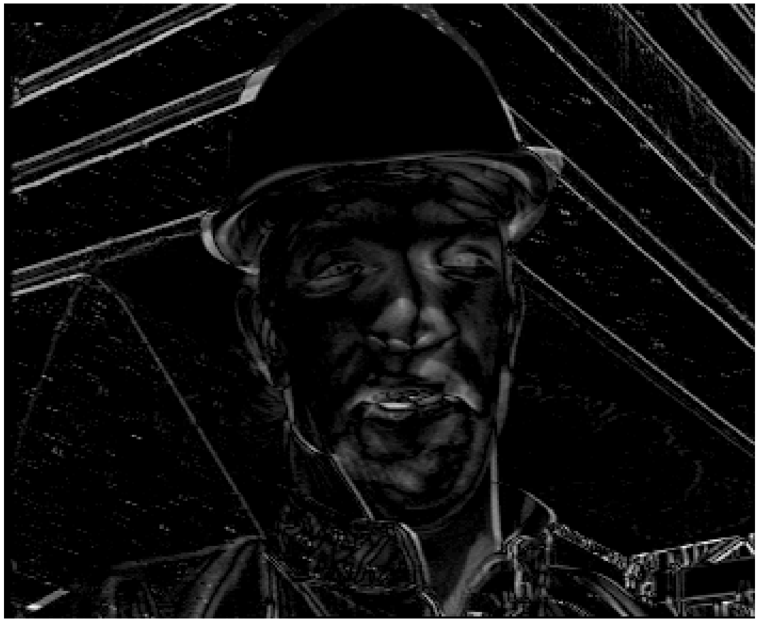
Frame 69



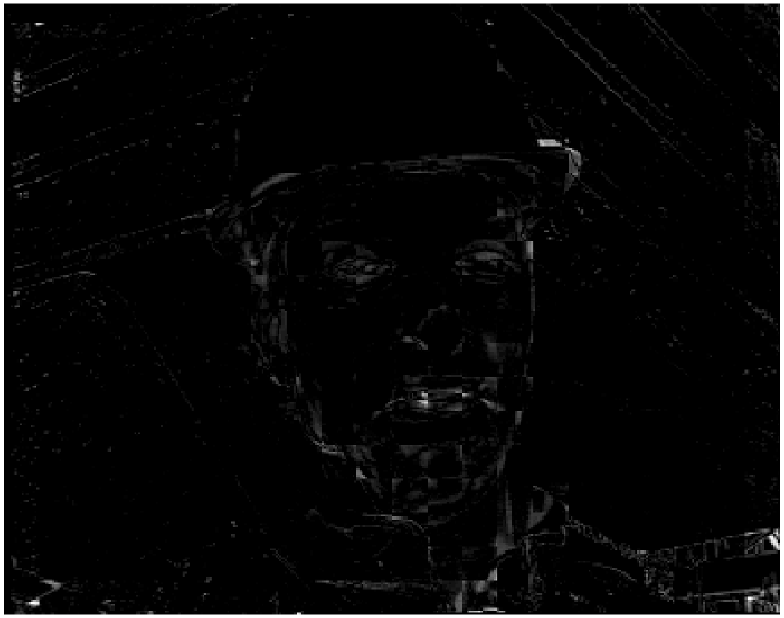
Predicted Frame69



Absolute Difference w/o Motion Compensation



Absolute Difference with Motion Compensation



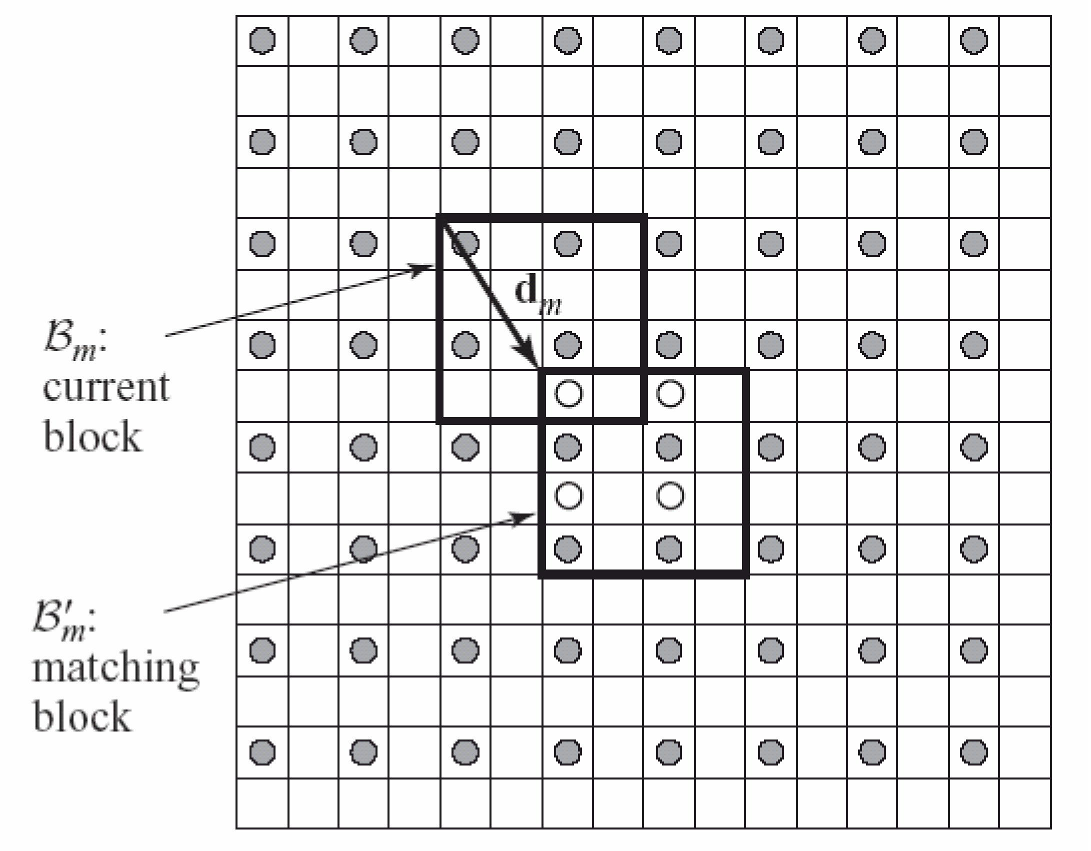
Complexity of Integer-Pel EBMA

- Assumption
  - Image size: MxM
  - Block size: NxN
  - Search range: (-R,R) in each dimension
  - Search stepsize: 1 pixel (assuming integer MV)
- Operation counts (1 operation=1 "-", 1 "+", 1 "*"):
  - Each candidate position: *N^2*
  - Each block going through all candidates: *(2R+1)^2 N^2*
  - Entire frame: *(M/N)^2 (2R+1)^2 N^2=M^2 (2R+1)^2*
    - Independent of block size!
- Example: M=512, N=16, R=16, 30 fps
  - Total operation count = 2.85x10^8/frame =8.55x10^9/second
- Regular structure suitable for VLSI implementation
- Challenging for software-only implementation

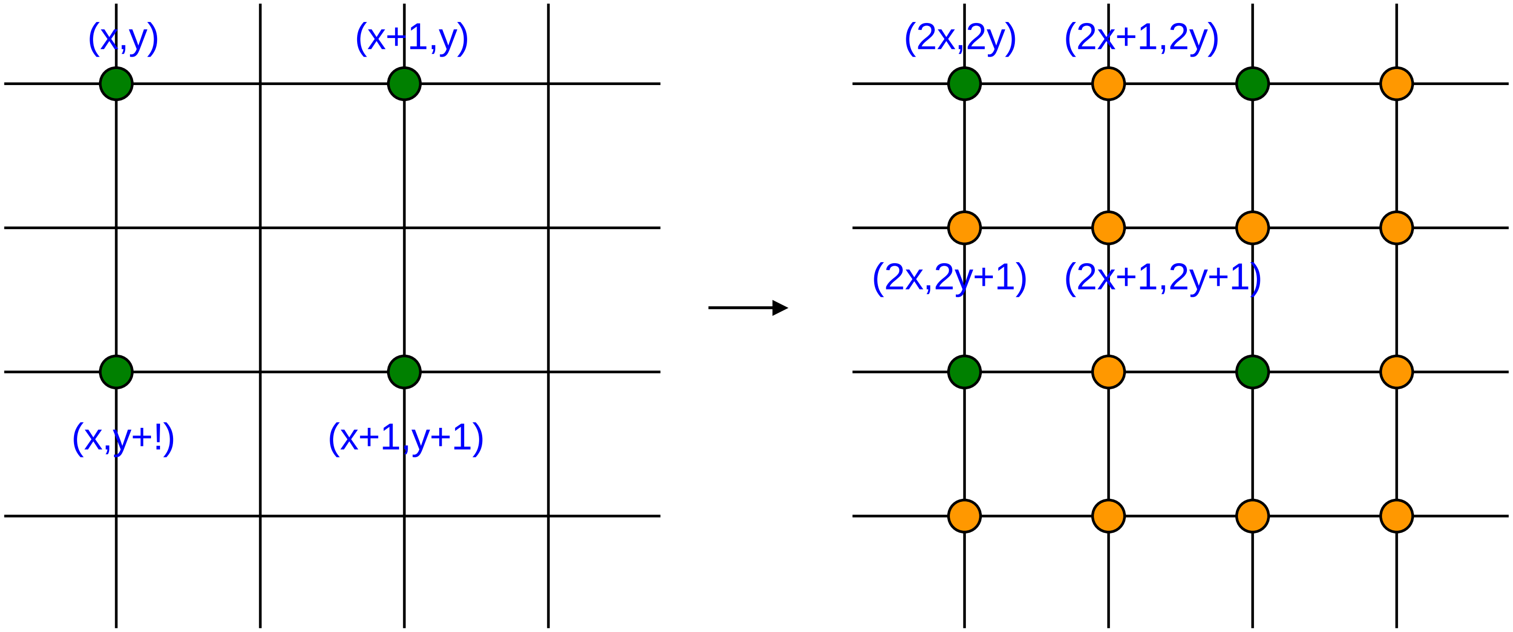
Fractional Accuracy EBMA

- Real MV may not always be multiples of pixels. To allow sub-pixel MV, the search stepsize must be less than 1 pixel
- Half-pel EBMA: stepsize=1/2 pixel in both dimension
  - Move the search block ½ pixel to left/bottom each time
- Difficulty:
  - Target frame only have integer pels
- Solution:
  - Interpolate the target frame by factor of two before searching
  - Bilinear interpolation is typically used
- Complexity:
  - 4 times of integer-pel, plus additional operations for interpolation.
- Fast algorithms:
  - Search in integer precisions first, then refine in a small search region in half-pel accuracy.

Half-Pel Accuracy EBMA



Bilinear Interpolation



$O[2x,2y]=I[x,y]$

$O[2x+1,2y]=(I[x,y]+I[x+1,y])/2$

$O[2x,2y+1]=(I[x,y]+I[x+1,y])/2$

$O[2x+1,2y+1]=(I[x,y]+I[x+1,y]+I[x,y+1]+I[x+1,y+1])/4$

Sample Matlab Script for Half-pel EBMA

```
%f1: anchor frame; f2: target frame, fp: predicted image; f22: interpolated f2;
%mvx,mvy: store the MV image, WxHt: image size; N: block size, R: search range
%interpolating f2 to f22
f22(1:2:H*2-1,1:2:W*2-1)=f2(1:H,1:W);
f22(1:2:H*2-1,2:2:W*2-2)=(f2(1:H,1:W-1)+f2(1:H,2:W))/2;f22(2:2:H*2-2,1:2:W*2-1)=(f2(1:H-1,1:W)+f2(2:H,1:W))/2;
f22(2:2:H*2-2,2:2:W*2-2)=(f2(1:H-1,1:W-1)+f2(1:H-1,2:W)+ f2(2:H,1:W-1)+f2(2:H,2:W)) /4;
for i=1:N:height-N, for j=1:N:width-N  %for every block in the anchor frame
     MAD_min=256*N*N;dx=0;dy=0;
     for k=-R:0.5:R, for l=-R:0.5:R  %for every search candidate
         MAD=sum(sum(abs(f1(i:i+N-1,j:j+N-1)-f22((i+k)*2-1:2:(i+k+N-1)*2-1,(,j+l)*2-1:2:(j+l+N-1)*2-1))));
                % calculate MAD for this candidate
         if MAD<MAX_min
                MAD_min=MAD,dy=k,dx=l;
         end;
     end;end;
     fp(i:i+N-1,j:j+N-1)= ff2((i+dy)*2-1:2:(i+dy+N-1)*2-1,,(j+dx)*2-1:2:(j+dx+N-1)*2-1);
                %put the best matching block in the predicted image
     iblk=(floor)(i-1)/N+1; jblk=(floor)(j-1)/N+1; %block index
     mvx(iblk,jblk)=dx; mvy(iblk,jblk)=dy; %record the estimated MV
end;end;
```

Note: A real working program needs to check whether a pixel in the candidate matching block falls outside the image boundary and such pixel should not be counted in MAD. This program is meant to illustrate the main operations involved. Not the actual working matlab script.

Pros and Cons with EBMA

- Blocking effect (discontinuity across block boundary) in the predicted image
  - Because the block-wise translation model is not accurate
  - Fix: Deformable BMA
- Motion field somewhat chaotic
  - because MVs are estimated independently from block to block
  - Fix 1: Mesh-based motion estimation
  - Fix 2: Imposing smoothness constraint explicitly
- Wrong MV in the flat region
  - because motion is indeterminate when spatial gradient is near zero
- Nonetheless, widely used for motion compensated prediction in video coding
  - Because its simplicity and optimality in minimizing prediction error

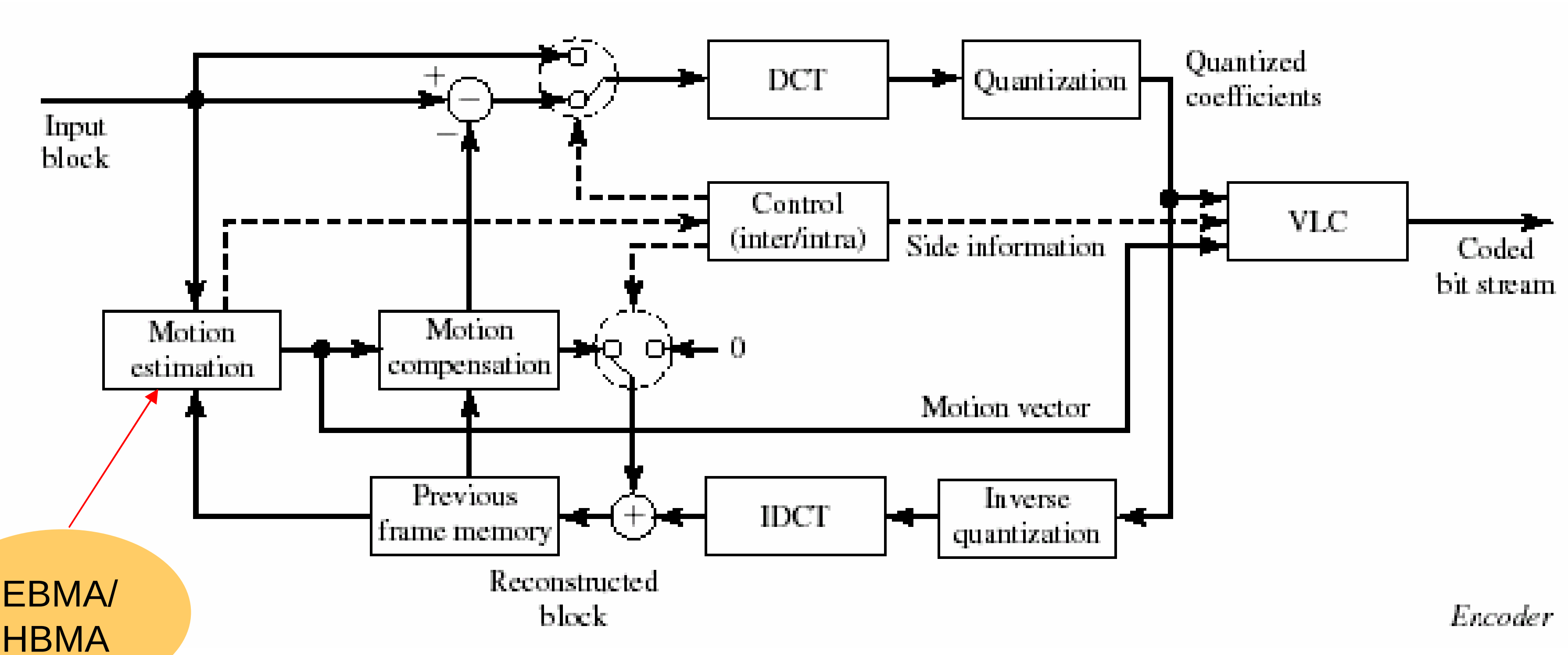
VcDemo Example

VcDemo: **Image and Video Compression Learning Tool**
Developed at Delft University of Technology

http://www-ict.its.tudelft.nl/~inald/vcdemo/

Use the ME tool to show the motion estimation results with different parameter choices

Encoder Block Diagram of a Typical Block-Based Video Coder



EBMA/
HBMA

BMA for Motion Compensated Prediction

- Uni-directional Motion Compensation:

$$\hat{f}(t,m,n) = f(t-1, m-d_x, n-d_y)$$

- Bi-directional Motion Compensation
  - Can handle better covered/uncovered regions

$$\hat{f}(t,m,n) = w_b f(t-1, m-d_{b,x}, n-d_{b,y}) \\ + w_f f(t+1, m-d_{f,x}, n-d_{f,y})$$

  - Two sets of motion vectors can be estimated independently, using BMA algorithms twice
  - For better results, the two motion vectors should be searched simultaneously, by minimizing the bi-directional prediction error

What you should know

- How does EBMA works?
- What is the effect of block size, search range, search accuracy (integer vs. half-pel) in terms of prediction accuracy and computation time?
- How is BMA used in video coding?

References

- Y. Wang, J. Ostermann, Y. Q. Zhang, *Video Processing and Communications,* Prentice Hall, 2002. chapter 6
- Y. Wang, EL514 Lab Manual on Video Coding.