Motion Estimation

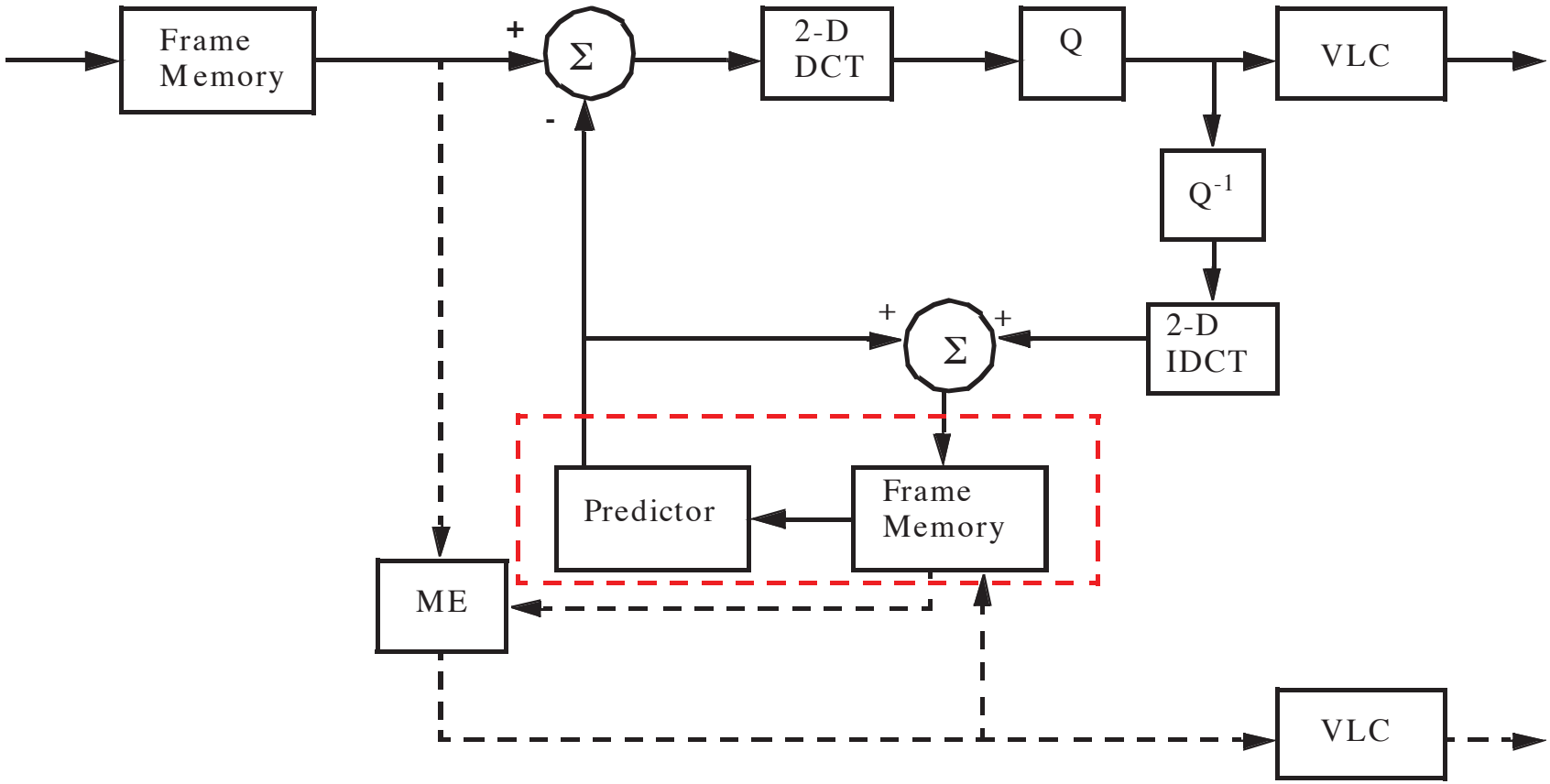
Blocking estimation
Macro Block
Pipeline

Video Process

- MPEG: http://en.wikipedia.org/wiki/Moving_Picture_Experts_Group
- H26x: http://en.wikipedia.org/wiki/H.261
- Two redundant factors: spatial and temporal

What is motion estimation

Interframe Transform/Predictive Coding



Frame 66

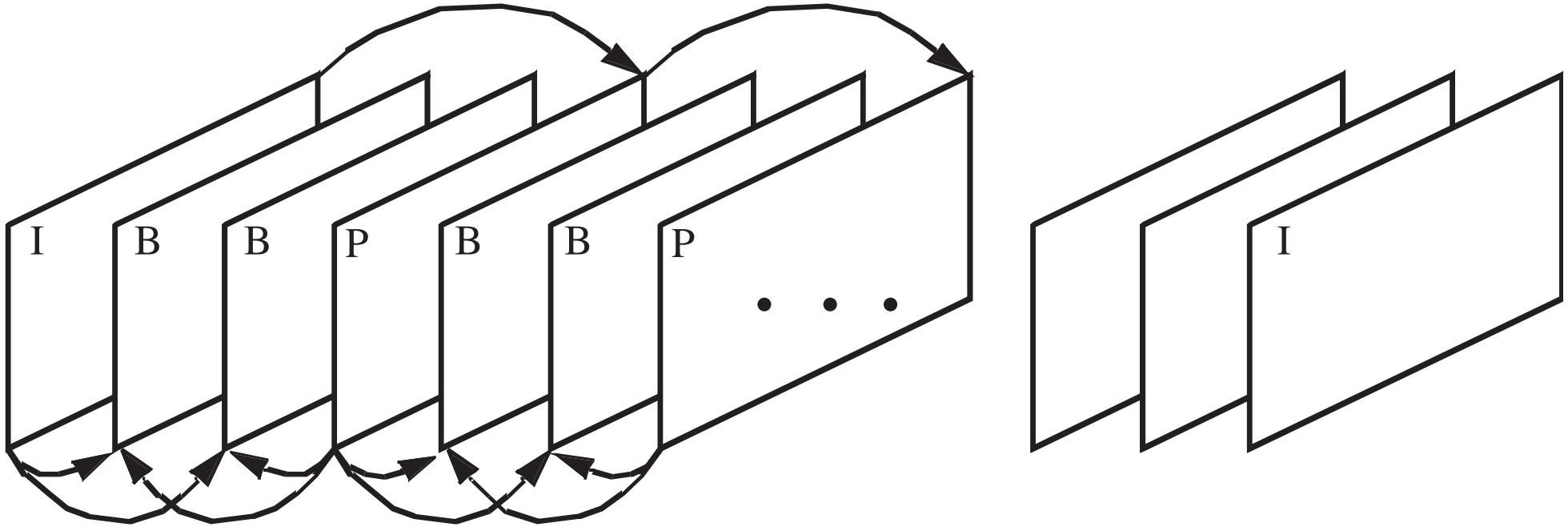
Frame 69

Absolute Difference w/o Motion Compensation

Absolute Difference with Motion Compensation

What happens in video

Interframe Coding / Motion Estimation of Video Sequence



I = Intraframe coding

P = Predictive coding

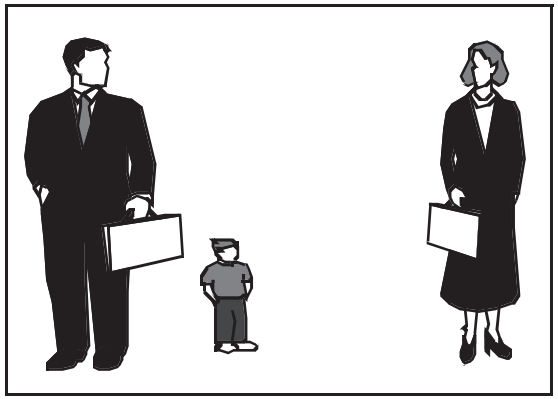
B = Bidirectional coding

Interframe operations

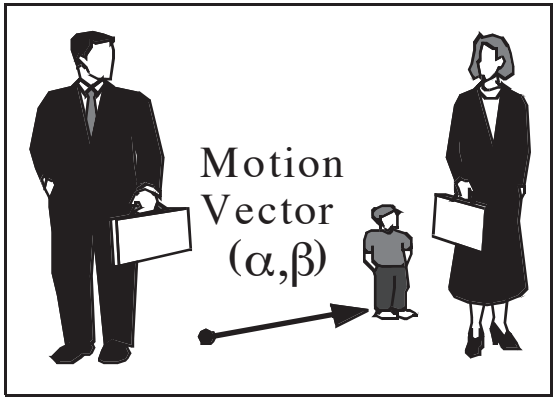
- Prediction is based on a previously processed frame
- Prediction is accomplished by motion estimation (ME)
- Motion estimation is done in spatial domain
- 2-D DCT has to be inside the coding loop and a 2-D IDCT is needed to convert the frequency domain information back to spatial domain
- ME information together with error image (if any) require much less number of bits to code than that of the original block
- ME is also the most computational intensive part of the encoder

Why Prediction?

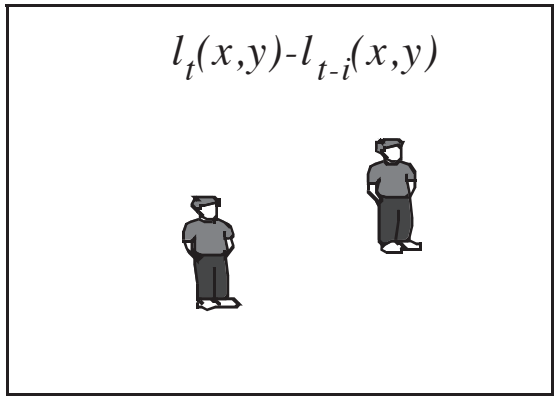
Motion Compensated Prediction



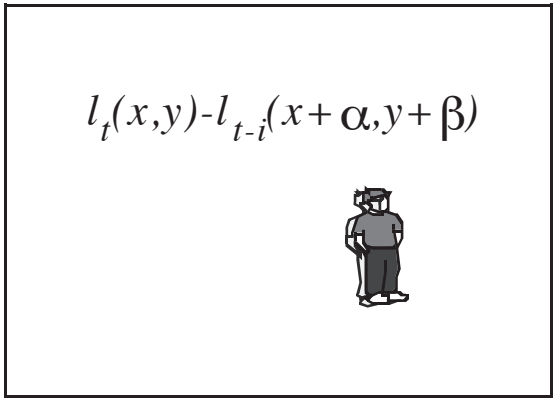
Frame at t-i



Frame at t

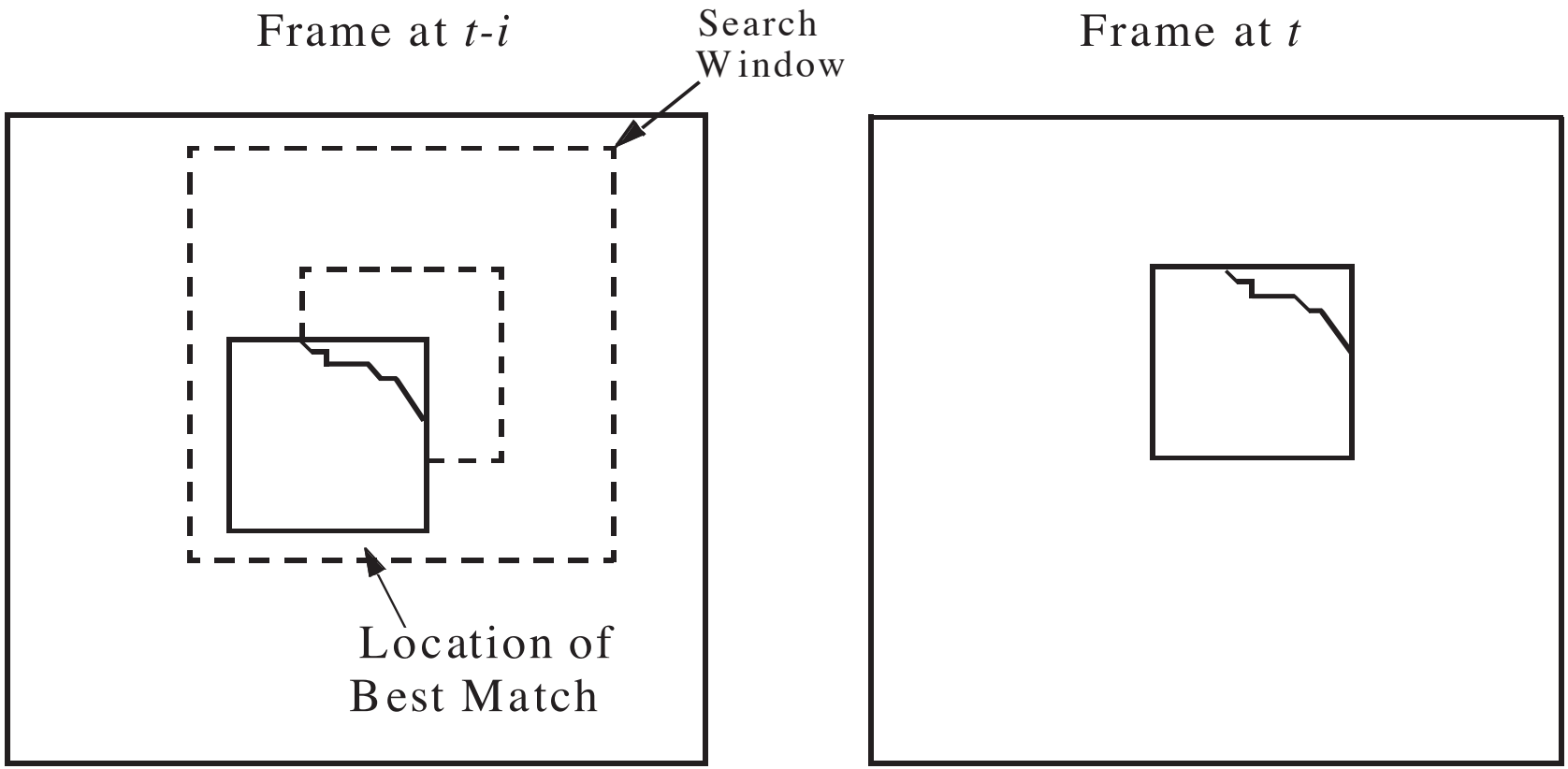


No motion compensation



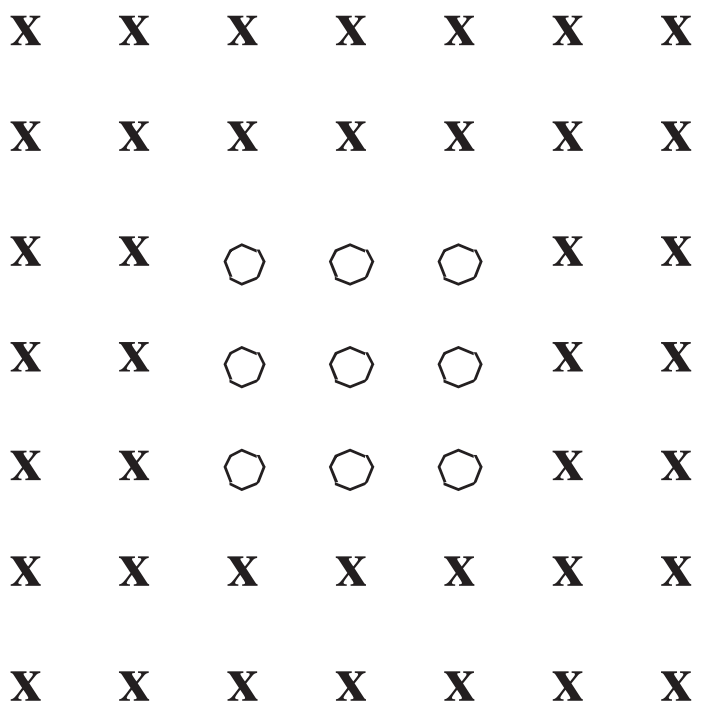
Motion Compensation

Block Matching Method



Checking Point in a Search Window

Search Window (pixels)



Search Window (block positions)

Each point above is called Checking Point

Block Matching Criteria

Mean Square Error (MSE)

$$MSE(\alpha,\beta) = \frac{1}{M^2} \sum_{x,y=1}^{M} \left[I_t(x,y) - I_{t-i}(x+\alpha, y+\beta) \right]^2$$

Mean Absolute difference (MAD)

$$MAD(\alpha,\beta) = \frac{1}{M^2} \sum_{x,y=1}^{M} \left| I_t(x,y) - I_{t-i}(x+\alpha, y+\beta) \right|$$


SIEMENS
SIEMENS
SIEMENS
SIEMENS

Important factors for block matching motion estimation

- Block size — 8×8, 16×16, variable
- Size of search window — Depends on frame differences, speed of moving objects, resolution, etc.
- Matching criterion — Accuracy *vs*. complexity, use of truncated pixels
- Search strategy — Full search, hierarchical approach, subsampling of motion field, reducing number of checking points (fast search algorithm), etc.
- Hardware consideration

Characteristics of BMME

Full Search

- Optimal, not only one but several best candidates are available
- Most time-consuming, brute force
- Due to its regularity, modular VLSI implementations have been used in practice

Fast Search Algorithms

- Suboptimal and not robust in general
- Require much less computation and also suitable for software implementation

Fast and Robust Algorithms

- A compromise between full search and fast algorithms

Naive Matlab code for BMME

```
%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;mvx=0;mvy=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;
```

Computational Complexity

Let

Block size = 16 × **16, Window size = 32** × 32,

assuming CIF frame at 30 f/s, we need

$$\left(256\frac{\text{ops}}{\text{search}}\right)\left(289\frac{\text{search}}{\text{block}}\right)\left(396\frac{\text{block}}{\text{frame}}\right)\left(30\frac{\text{frame}}{\text{sec}}\right) = 879\ \text{Mops / sec}$$

Format	Video Resolution
SQCIF	128 × 96
QCIF	176 × 144
SCIF	256 x 192
SIF(525)	352 x 240
CIF/SIF(625)	352 × 288
4SIF(525)	704 x 480
4CIF/4SIF(625)	704 × 576
16CIF	1408 × 1152
DCIF	528 × 384

For CCIR 601 or HDTV, it would require several or tens of Gops/sec.

Full search has been used in certain applications with specially designed hardware. Other search algorithms with reduced complexity have been and are still being developed.

Implementation for BMME

- Software-based
  - Single Processor is possible only for very simple fast search algorithm applied on low spatial and temporal resolution video coding
  - Parallel processors are needed for higher end applications
  - Very flexible
- ASIC
  - Not flexible in general
  - Most cost effective if carefully designed
- Hybrid
  - Programmable device for coding control
  - Hardware-assisted engine for signal processing

Fast search, based on reduction in the number of checking points

- Three-step search
- New three-step search
- Zoned-based search
- Circular/Diamond shape zonal search with half stop criteria

Please check the matlab toolbox shown on the lecture website.

Please refer to further methods:
http://www.ece.cmu.edu/~ee899/project/deepak_mid.htm

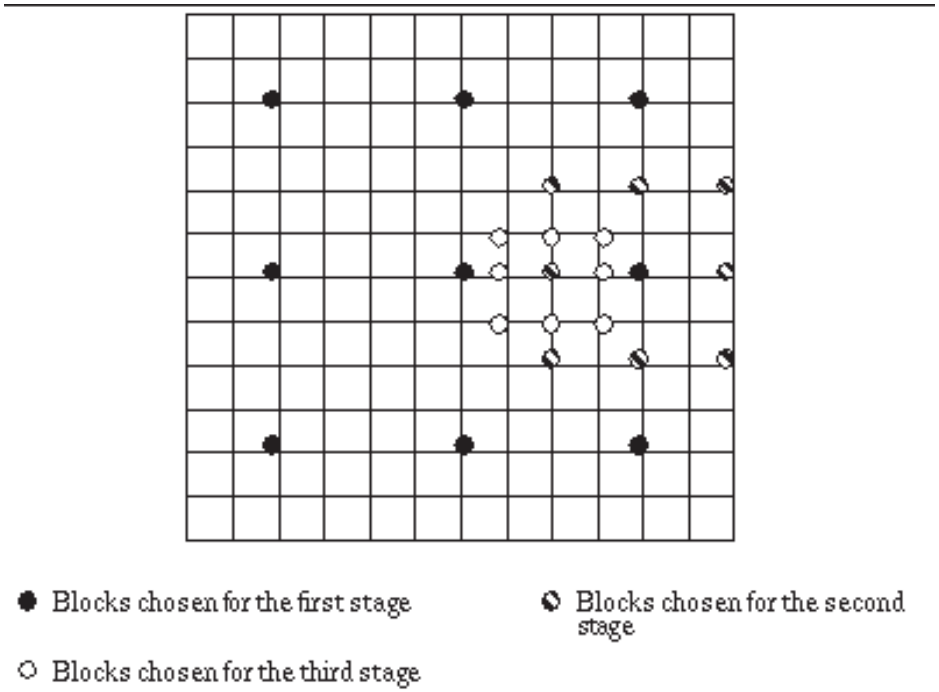
Three-step search (TSS)

This algorithm was introduced by Koga et al in 1981. It became very popular because of its simplicity and also robust and near optimal performance. It searches for the best motion vectors in a coarse to fine search pattern. The algorithm may be described as:

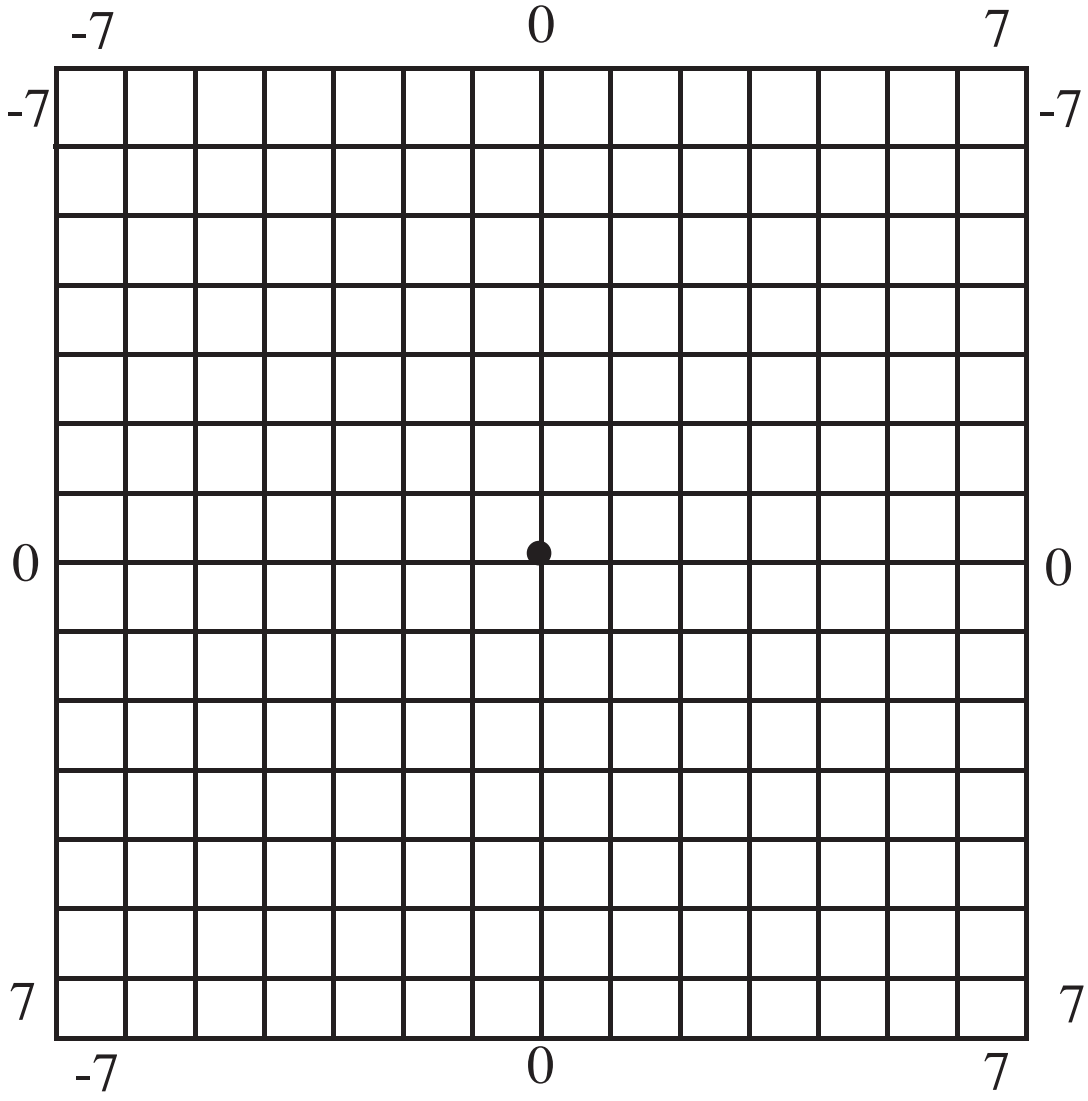
Step 1: An initial step size is picked. Eight blocks at a distance of step size from the centre (around the centre block) are picked for comparison.

Step 2: The step size is halved. The centre is moved to the point with the minimum distortion.

Steps 1 and 2 are repeated till the step size becomes smaller than 1. A particular path for the convergence of this algorithm is shown right.



Exercise



For window size $p = 7$

Example:

# of checking points =

# of sequential steps =

Worst case:

# of checking points =

# of sequential steps =

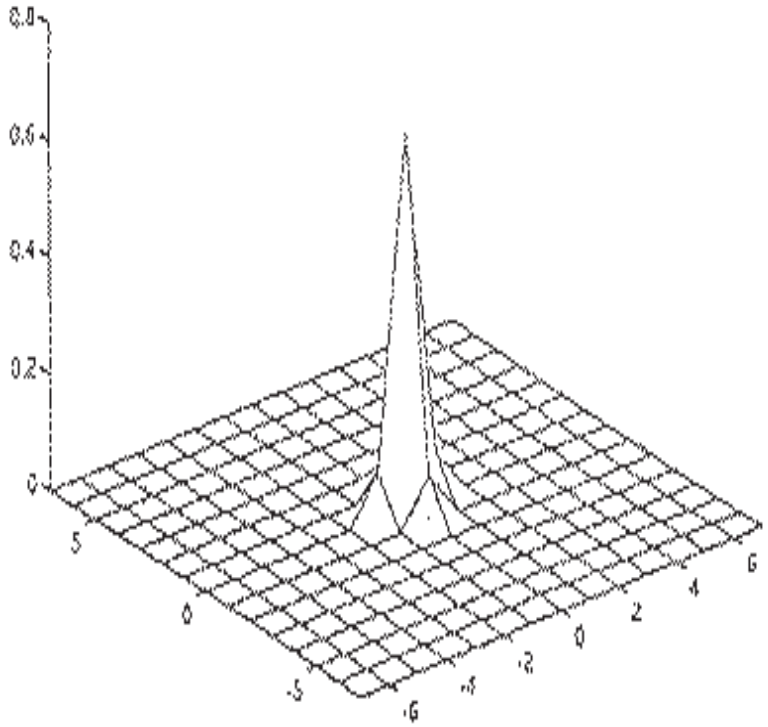
Best case:

# of checking points =

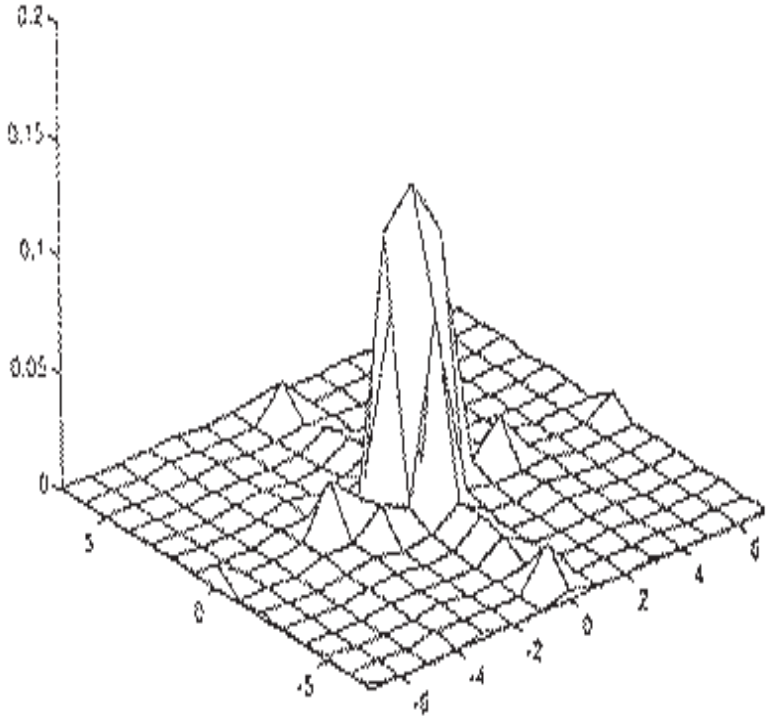
# of sequential stops =

Long-term **Motion Vector** Distribution

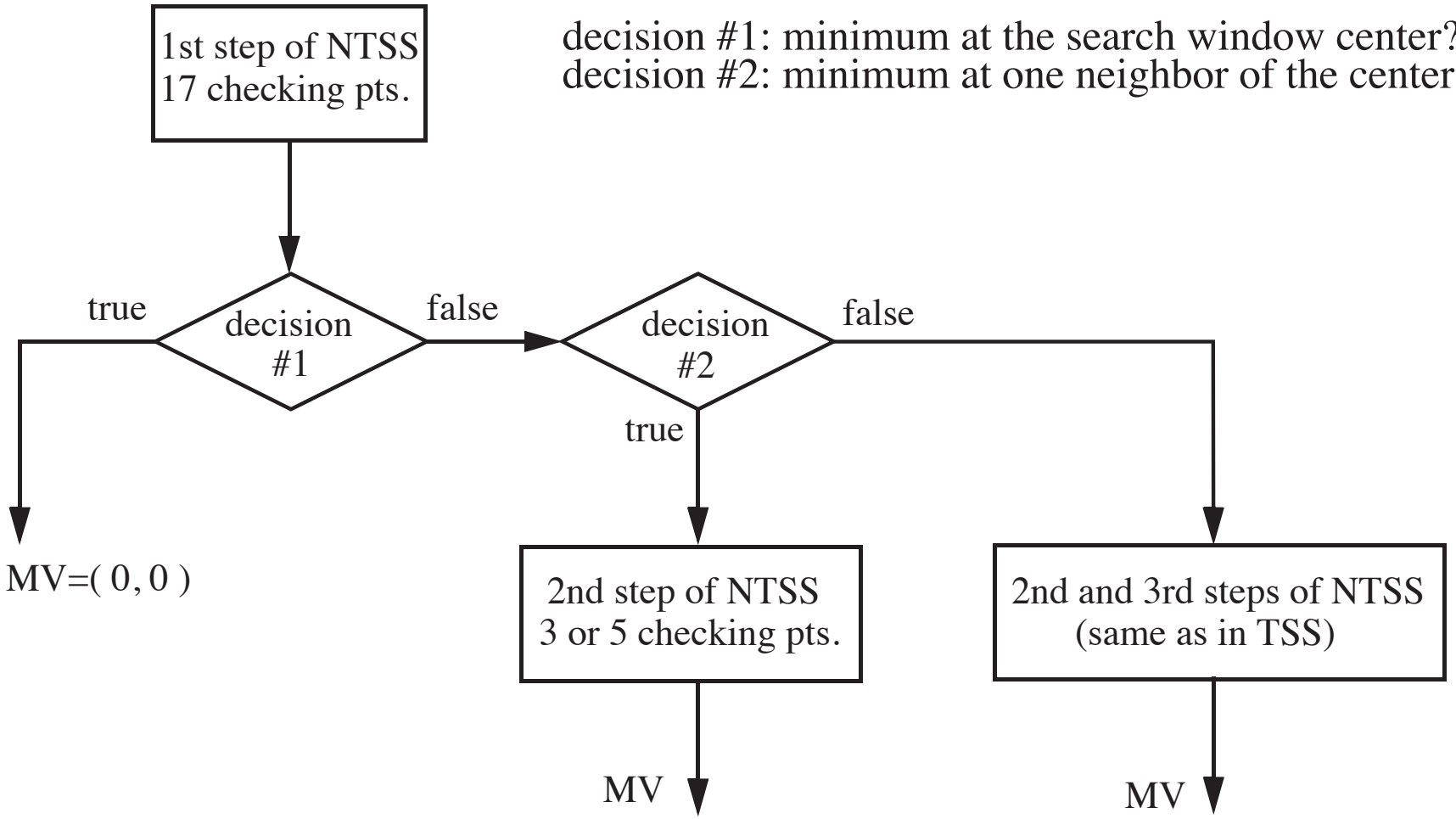
Salesman



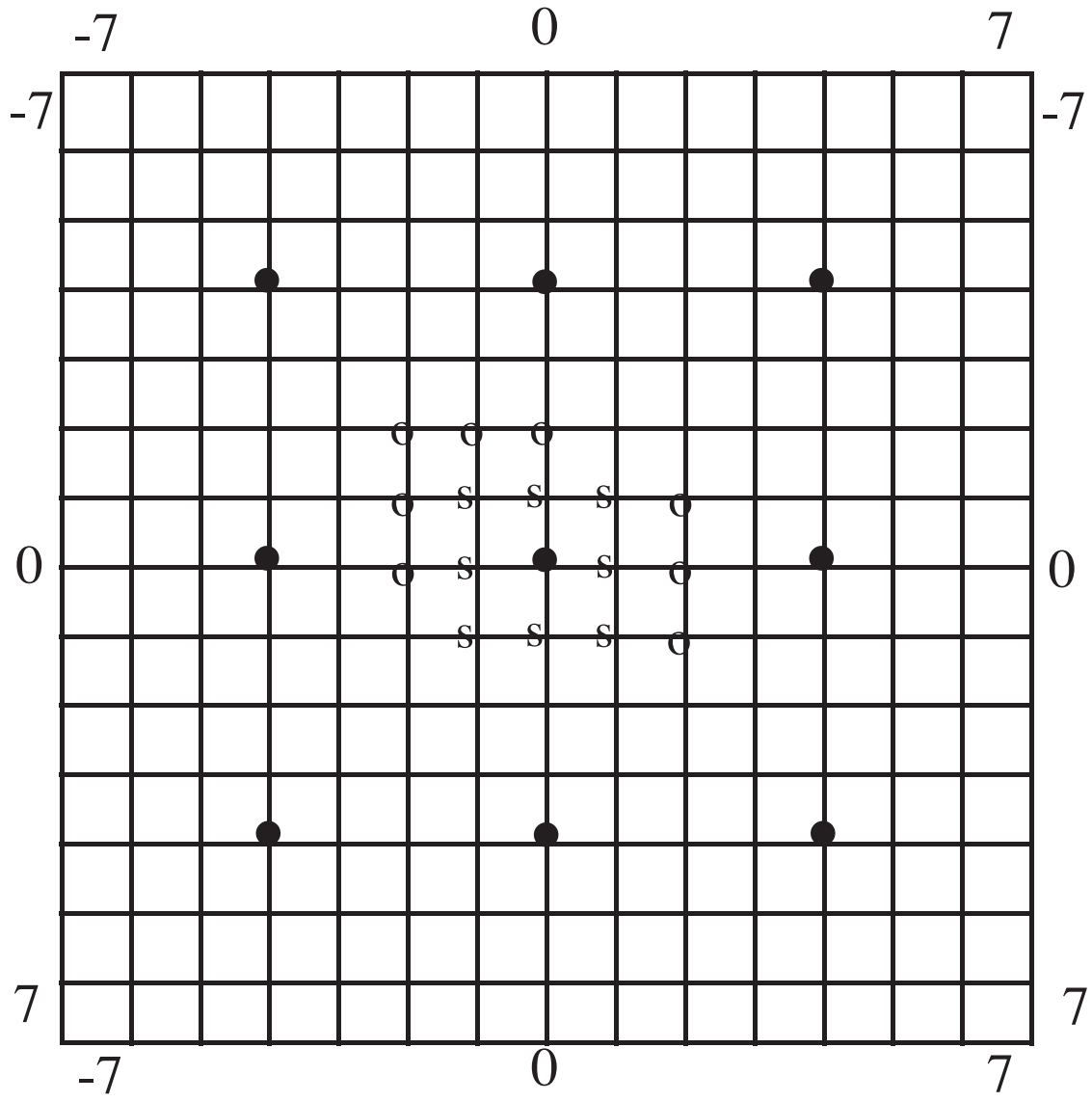
Miss America



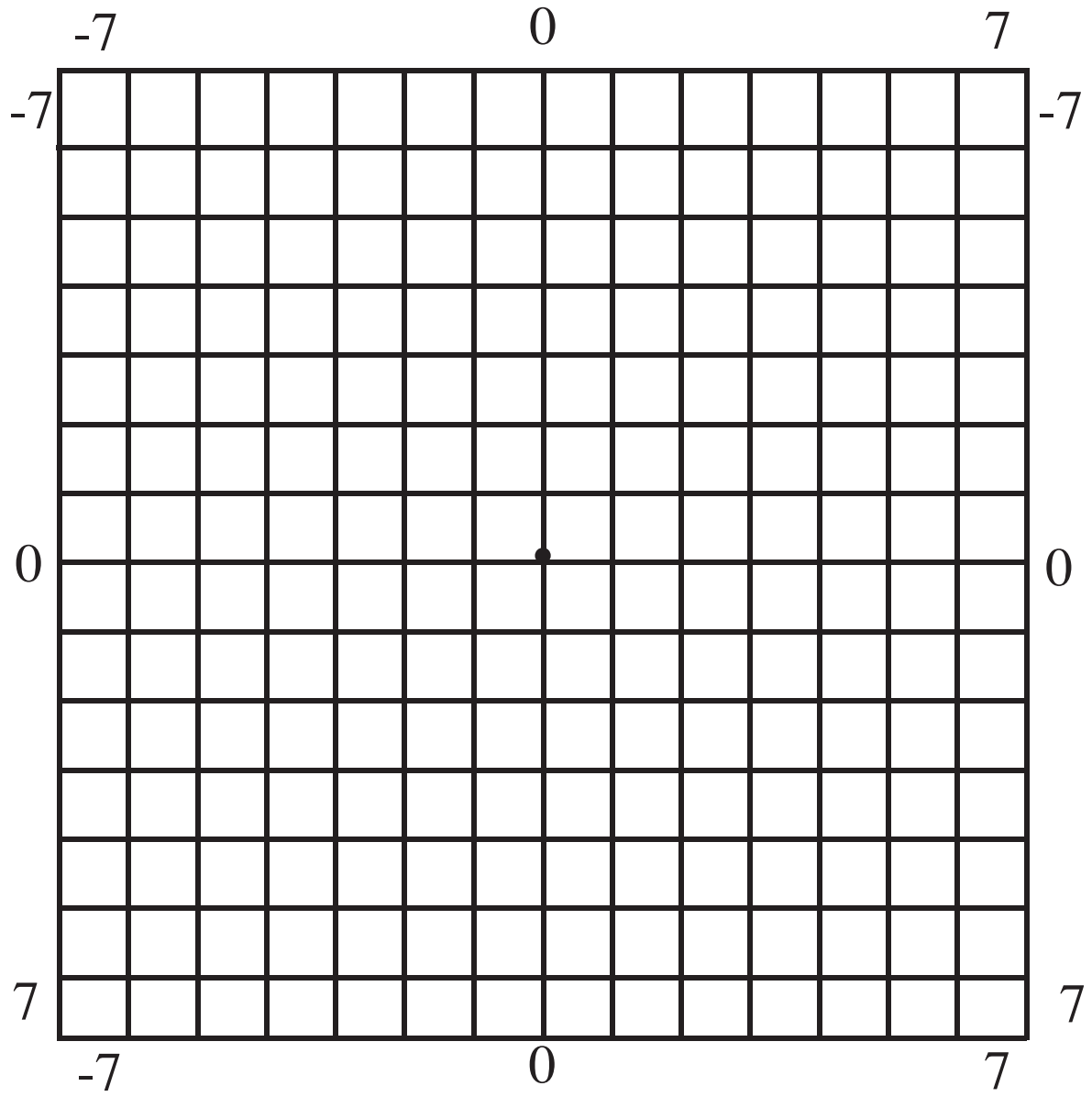
New TSS


1st step of NTSS
17 checking pts.
decision #1: minimum at the search window center?
decision #2: minimum at one neighbor of the center?
true
decision
#1
false
decision
#2
false
true
MV=(0, 0)
2nd step of NTSS
3 or 5 checking pts.
2nd and 3rd steps of NTSS
(same as in TSS)
MV
MV

New TSS Pattern



Same exercise for NTSS



For window size $p = 7$

Example:

# of checking points =

# of sequential steps =

Worst case:

# of checking points =

# of sequential steps =

Best case:

# of checking points =

# of sequential stops =

Compare TSS and NTSS

Speed-up ratio w.r.t. full search

	Salesman	Miss America
TSS	9.0	9.0
NTSS	10.94	7.94

Probability of catching true motion

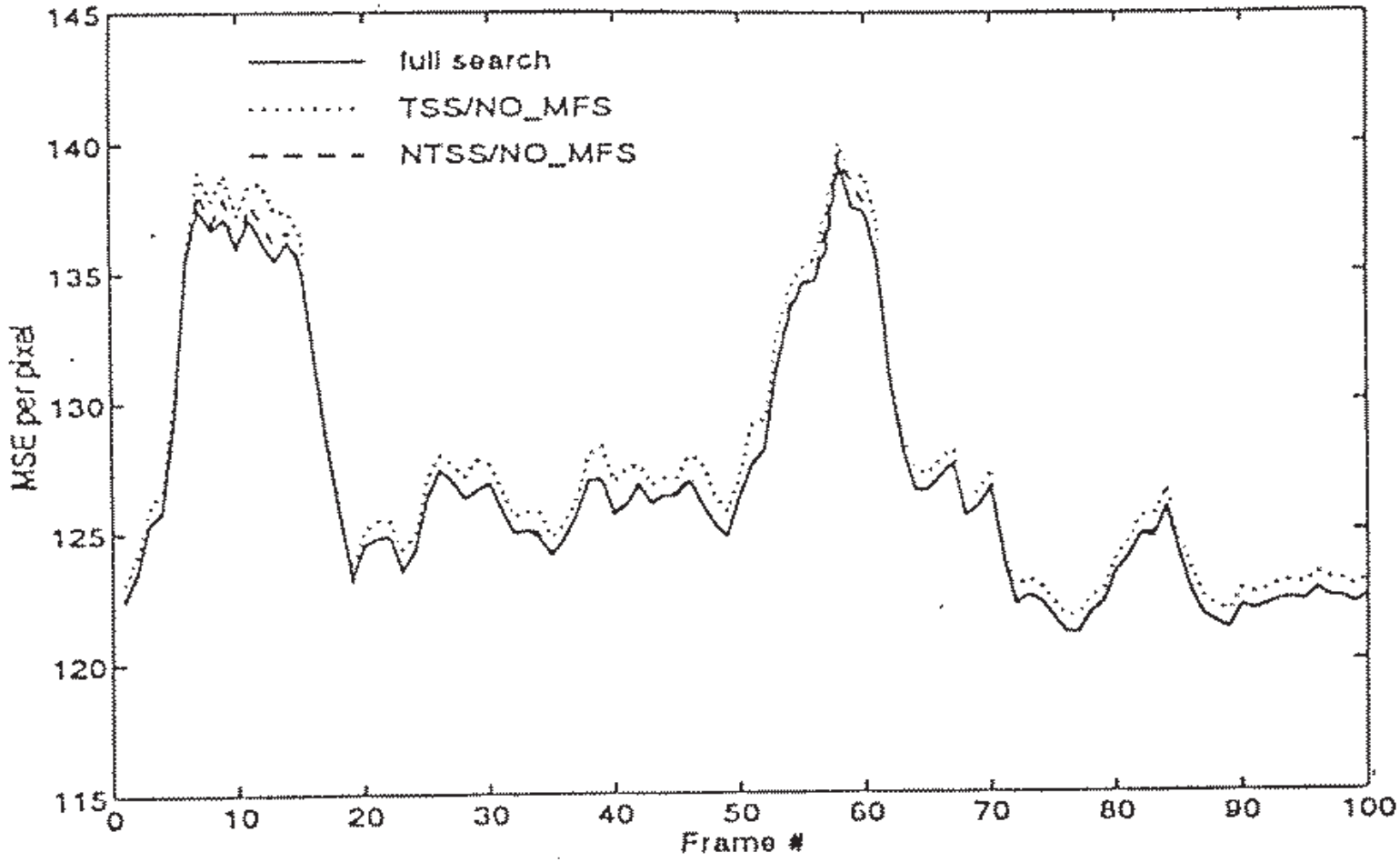
	Salesman	Miss America
TSS	0.951	0.535
NTSS	0.990	0.722

Average distance to true minimum

	Salesman	Miss America
TSS	0.369	1.193
NTSS	0.044	0.687

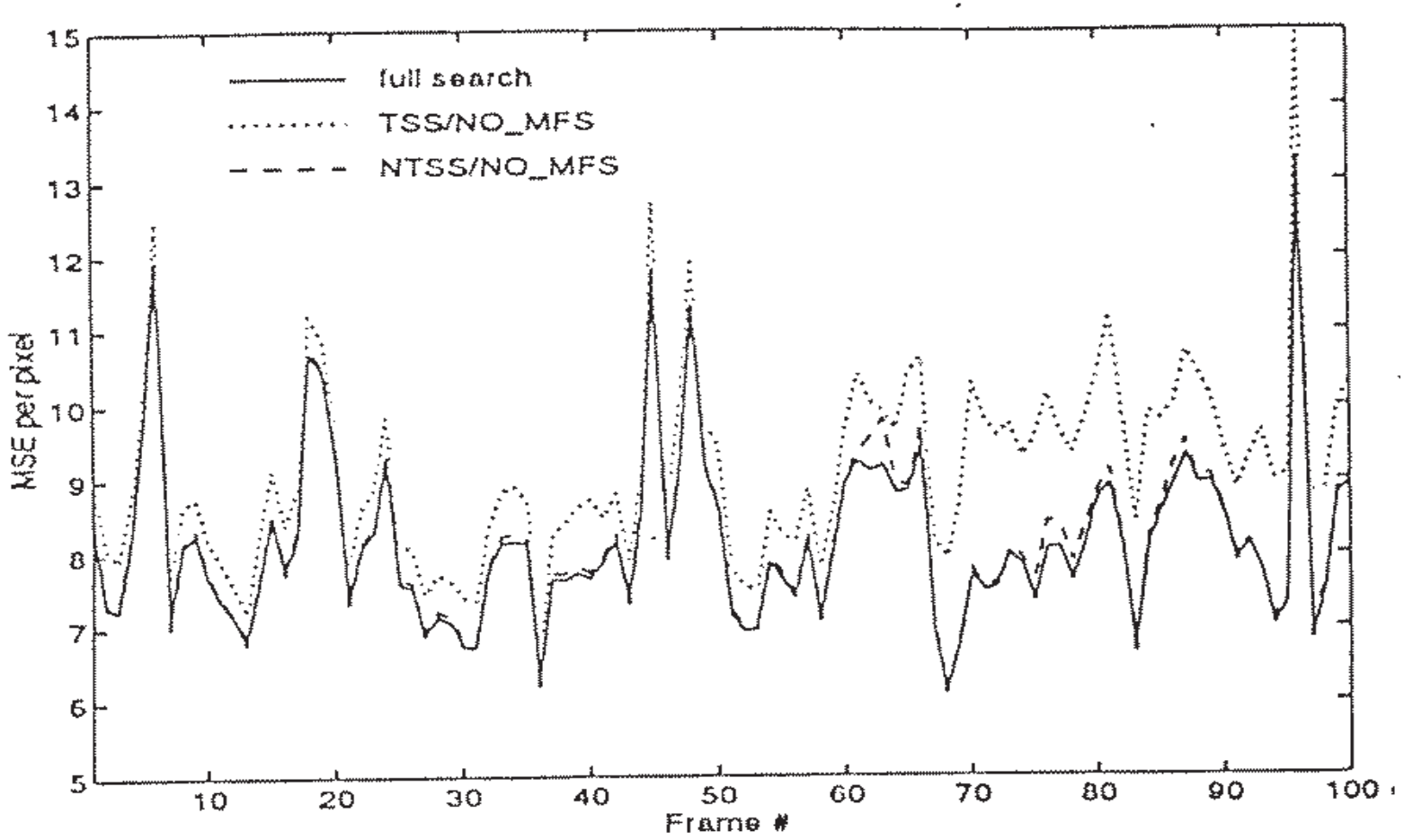
Performance in action

Performance of New Three-Step Search
Salesman

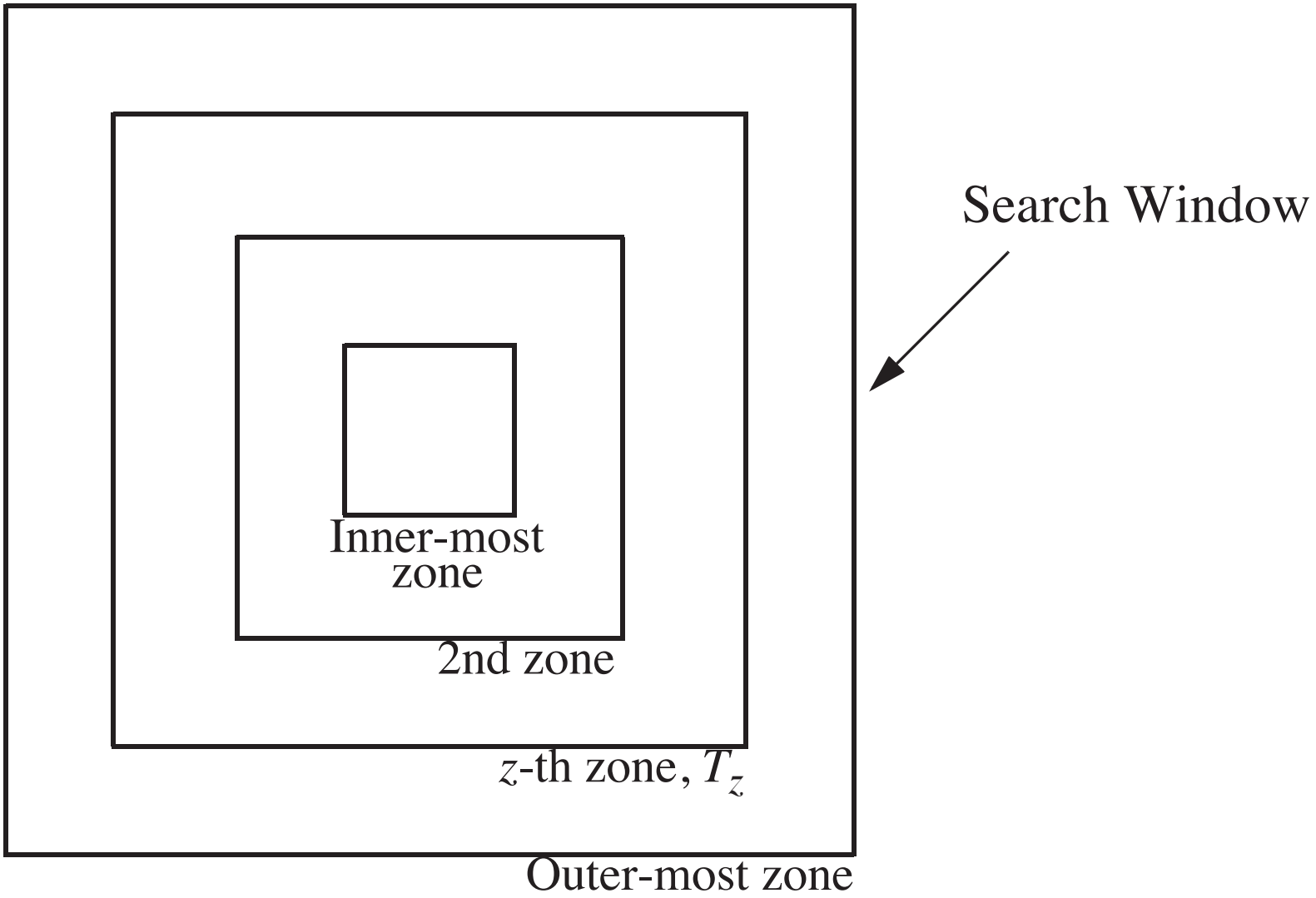


Performance in action

Performance of New Three-Step Search
Miss America



Zone-based BMME



Algorithm of Zone-based Method

Min MAD_1 = *Initial* MAD *value*;

MV(i, j) = Initial motion vector;

FOR($z = 1$; $z \le z_{max}$; $z \mathrel{+}= 1$) {

    Search the z-th zone, find *Min* MAD_z and MV_z;

    IF(*Min* $\mathrm{MAD}_z \le T_z$) {

        $MV(i, j) = MV_z$;

        stop;

    }

    ELSE {

        Min MAD_{z+1} = *Min* MAD_z;

        $MV(i, j) = MV_z$;

    }

}

Observation on Zone-based Method

- When T_1 increase, more *MV*s can be found in zone 1, *Min* MAD increases

- For stationary or quasi-stationary video sequences, very high speed-up ratio can be achieved with equal or better performance than full search because less bits is needed for *MV*s and more bits can be used to code DCT coefficients

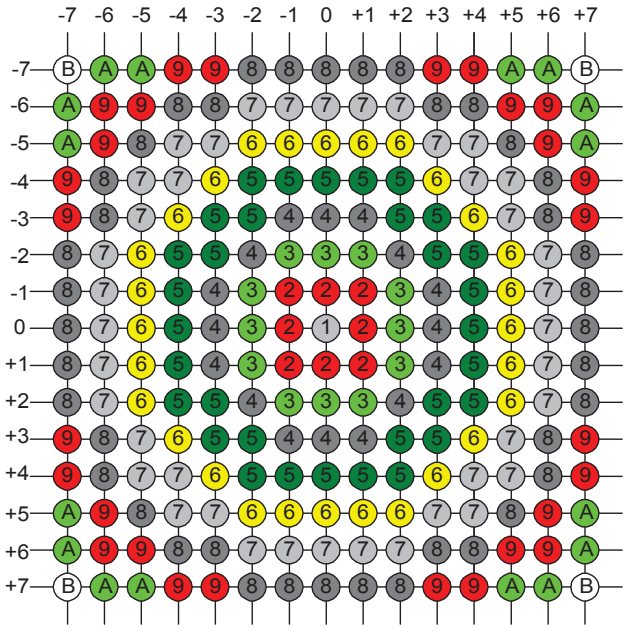
Another variation

Circular Zonal Search Algorithm (CZS)

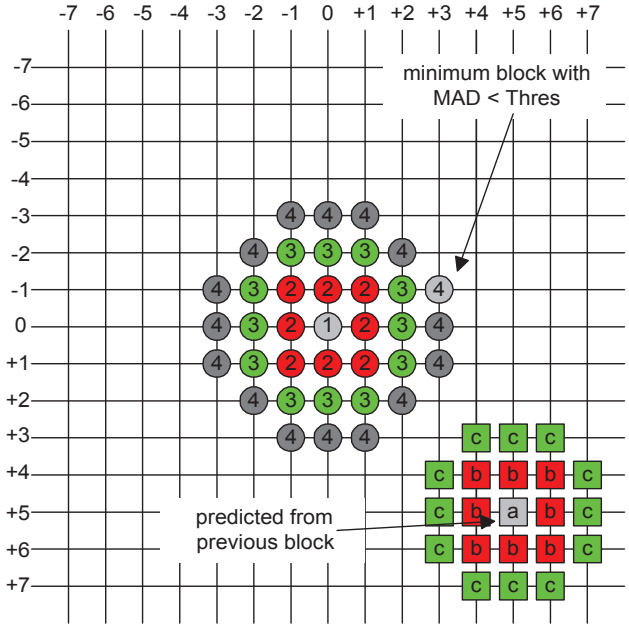
Zones are defined in a circular manner.

- Assumes motion is uniform on all directions.

Benefit from DPCM. First examine predicted MV. Requires less bits to encode motion vectors thus leaves more Bits to encode residue image.



Definition of Circular zones inside a search window



Zones defined around predicted and center. If MV not found around predicted center will be examined.

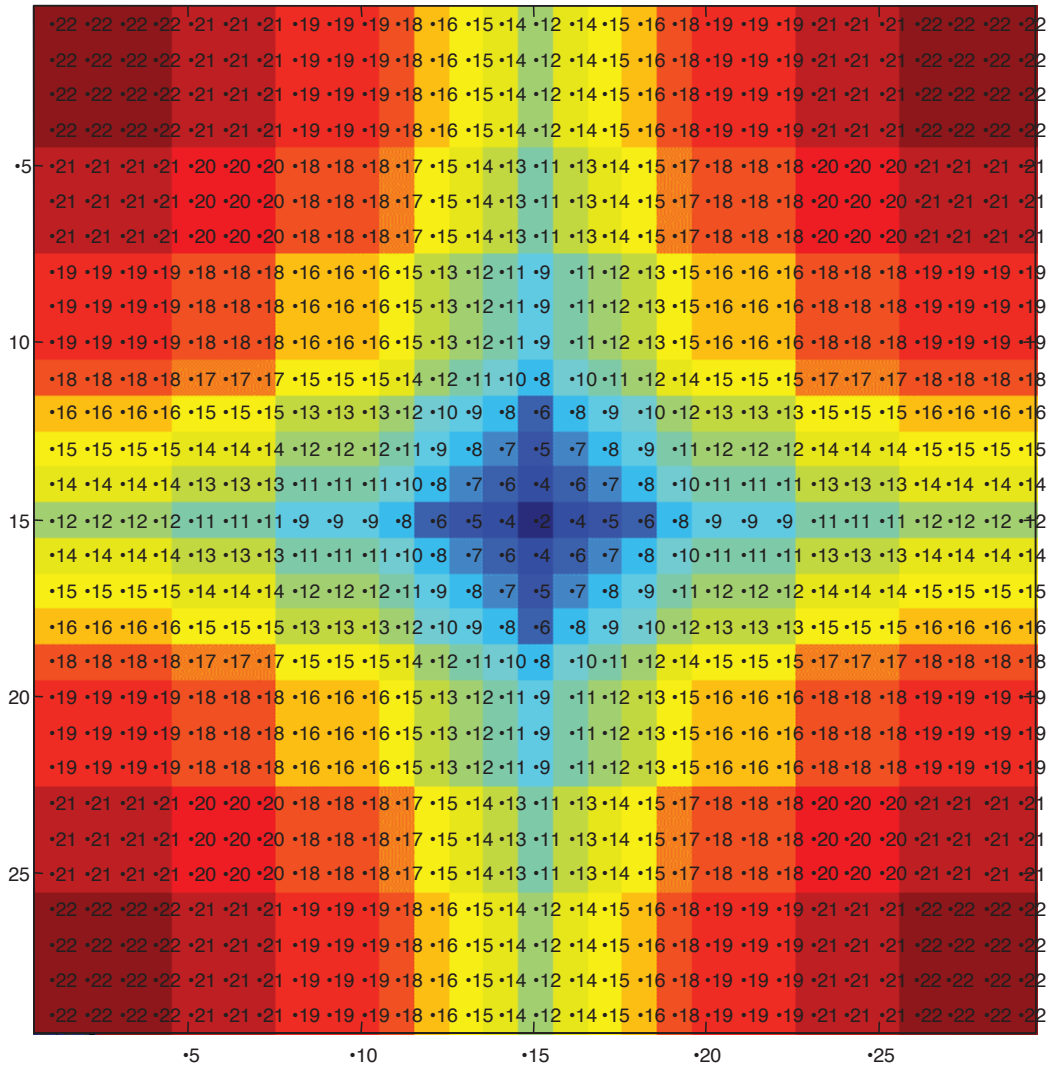
Circular Zonal Search Algorithm Cont.

- Particulars of algorithm :
  - Uses two thresholds.
    - If matching criterion satisfies threshold one, stop.
    - If not, but second threshold is satisfied, examine a few more zones (i.e. 1,2 etc) and then stop.
    - Otherwise continue to the next zones.
  - Substantial reduction of Computational Complexity (esp. for slow motion frames)
  - Slow for fast moving sequences even though small gain can be achieved.
  - Small quality gain can be observed due to less bits required from motion vectors in general.

Half-Stop Circular Zonal Search (HSCZS)

- Observations :
  - High motion sequences => ME Criterion value too high for many blocks. If threshold too high might lose some better blocks. Not a good choice.
  - If true minimum is found all other points checked (or zones) will have larger values always.
- Conclusion :
  - If the current minimum is not updated after n zones maybe it is the true minimum.
    Thus stop the search => Half-Stop Circular Zonal Search (HSCZS)
  - CZS is a subset of HSCZS ($n = \infty$).

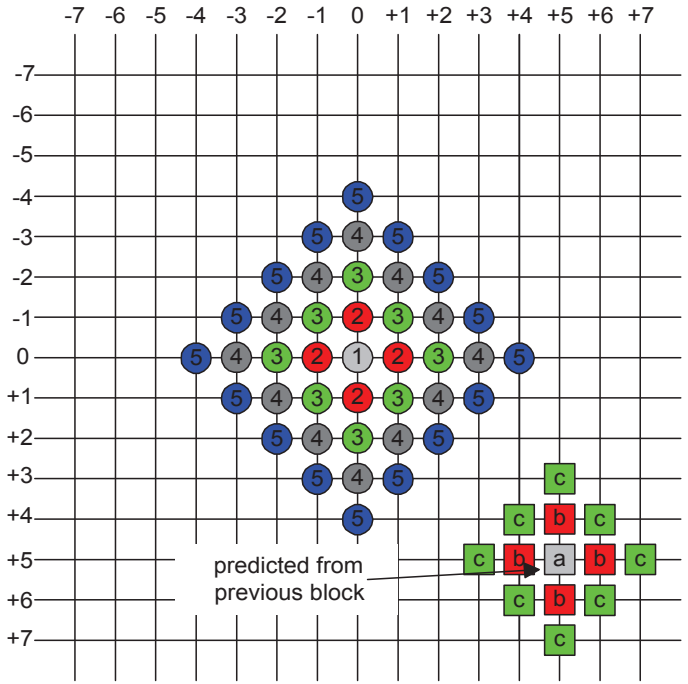
Diamond Zonal Search



Motion vectors as encoded in MPEG encoder

MVs coding scheme.

=> Diamond pattern for Zonal search seems to be more appropriate than circle. Further increase in Speed Up.



Definition of Diamond Search.

Other Techniques for Reducing Computation of Motion Estimation

- MAD computation using subsampled pixels
- Subsampling of motion field
- Subblock motion field estimation
- Combination of the above
- MAD computation using truncated pixels
- Hierarchical block matching motion estimation
- Others

Alternating pixel-decimation patterns

a	**b**	**a**	**b**	**a**	**b**	**a**	**b**
c	**d**	**c**	**d**	**c**	**d**	**c**	**d**
a	**b**	**a**	**b**	**a**	**b**	**a**	**b**
c	**d**	**c**	**d**	**c**	**d**	**c**	**d**
a	**b**	**a**	**b**	**a**	**b**	**a**	**b**
c	**d**	**c**	**d**	**c**	**d**	**c**	**d**
a	**b**	**a**	**b**	**a**	**b**	**a**	**b**
c	**d**	**c**	**d**	**c**	**d**	**c**	**d**

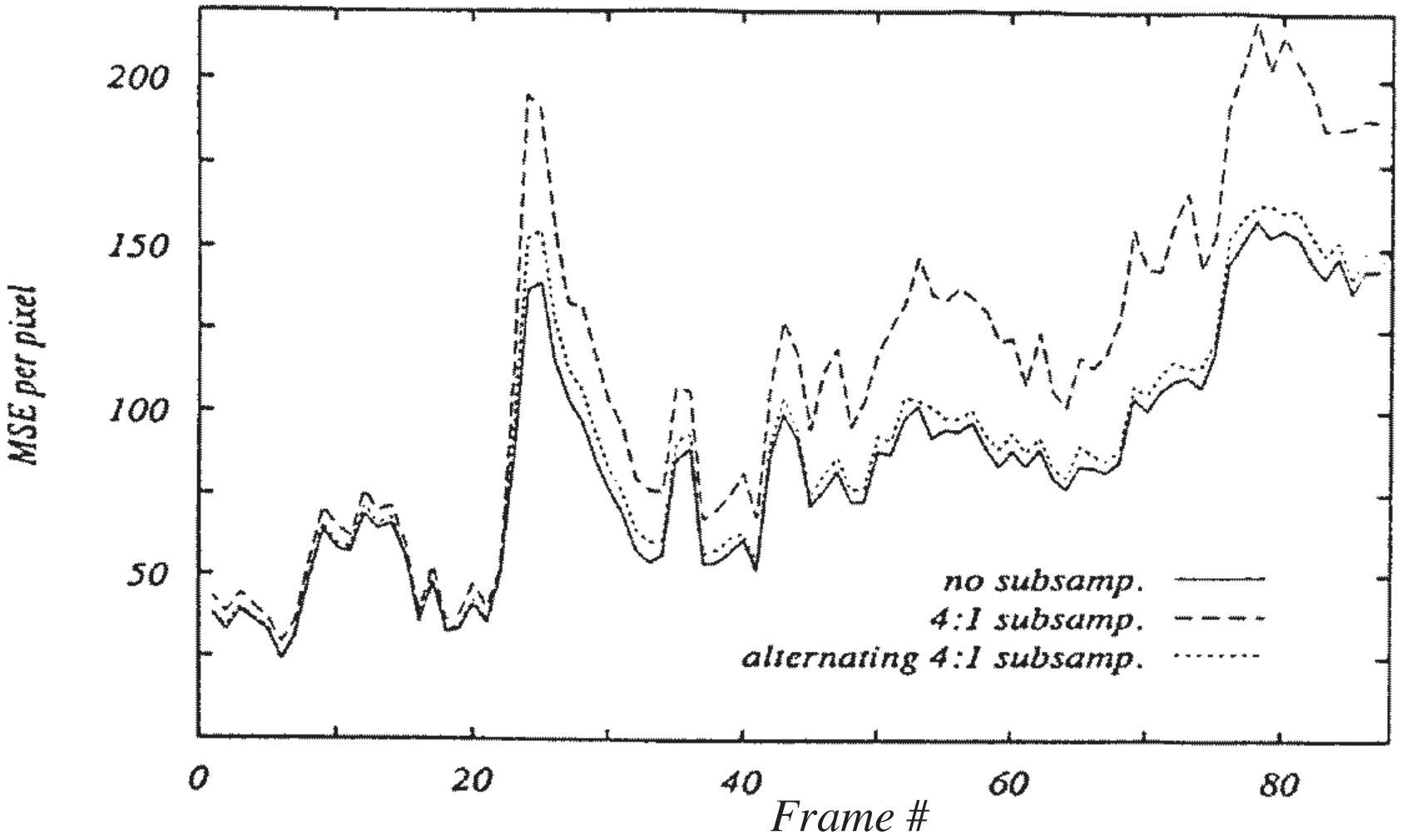
Pixel pattern in current block

3	**2**	**3**	
4	**1**	**4**	**1**
3	**2**	**3**	**2**
	1	**4**	**1**

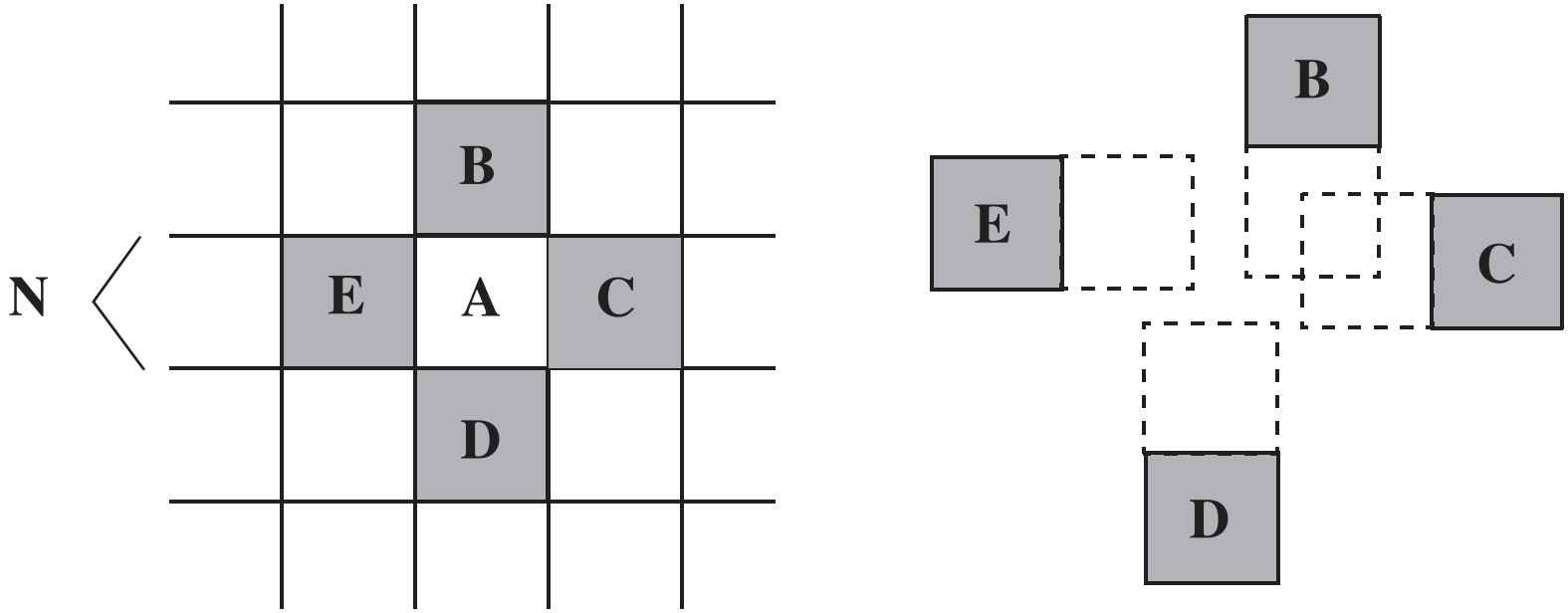
Pixel pattern in search window

- One motion vector for each subsampling pattern
- Compute match for each motion vector using all pixels
- Choose one with minimum MAD
- Can achieve approximately 4:1 reduction

Performance of Pixel subsampling



Sub-sampling and interpolation

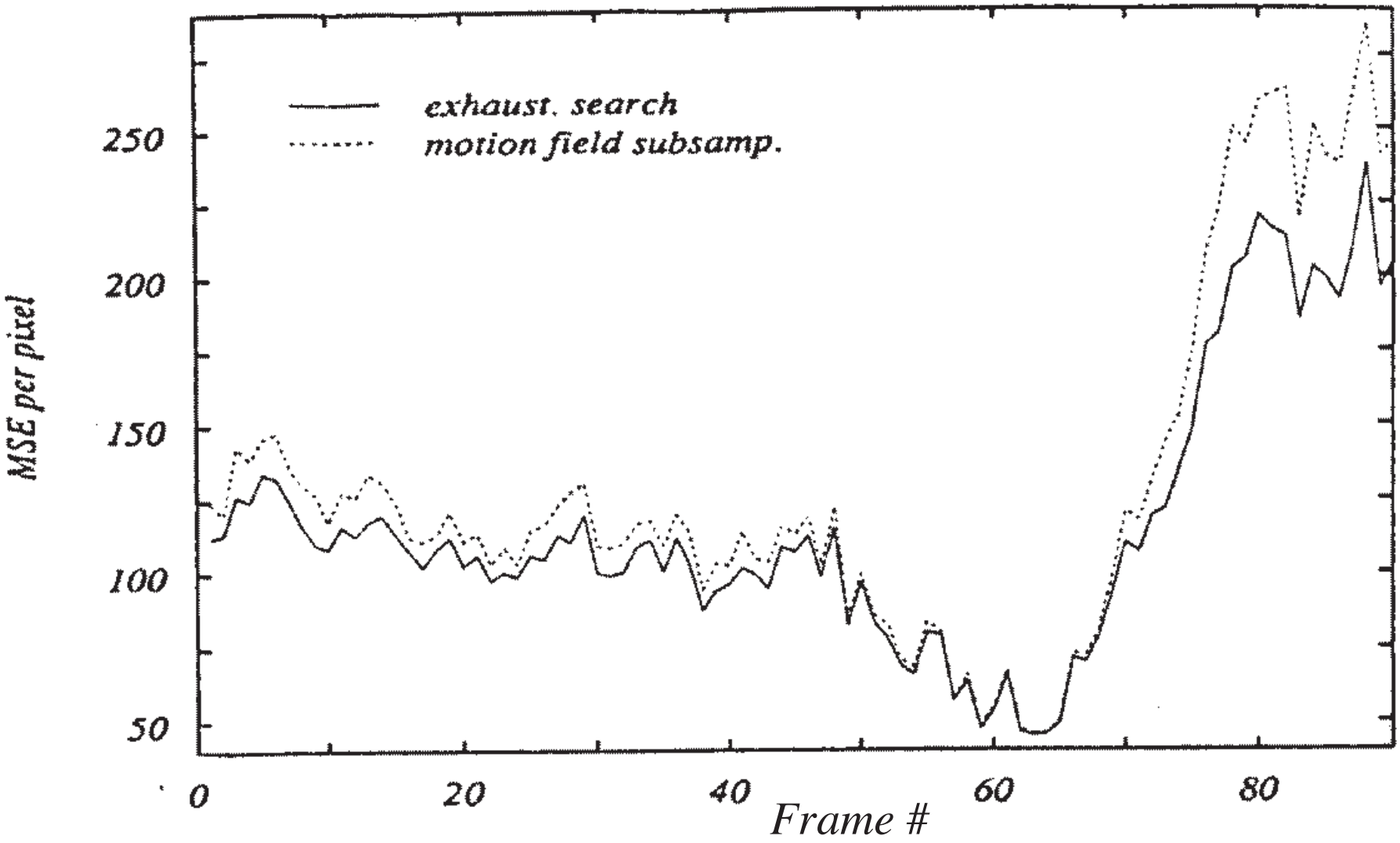


Motion-field subsampling

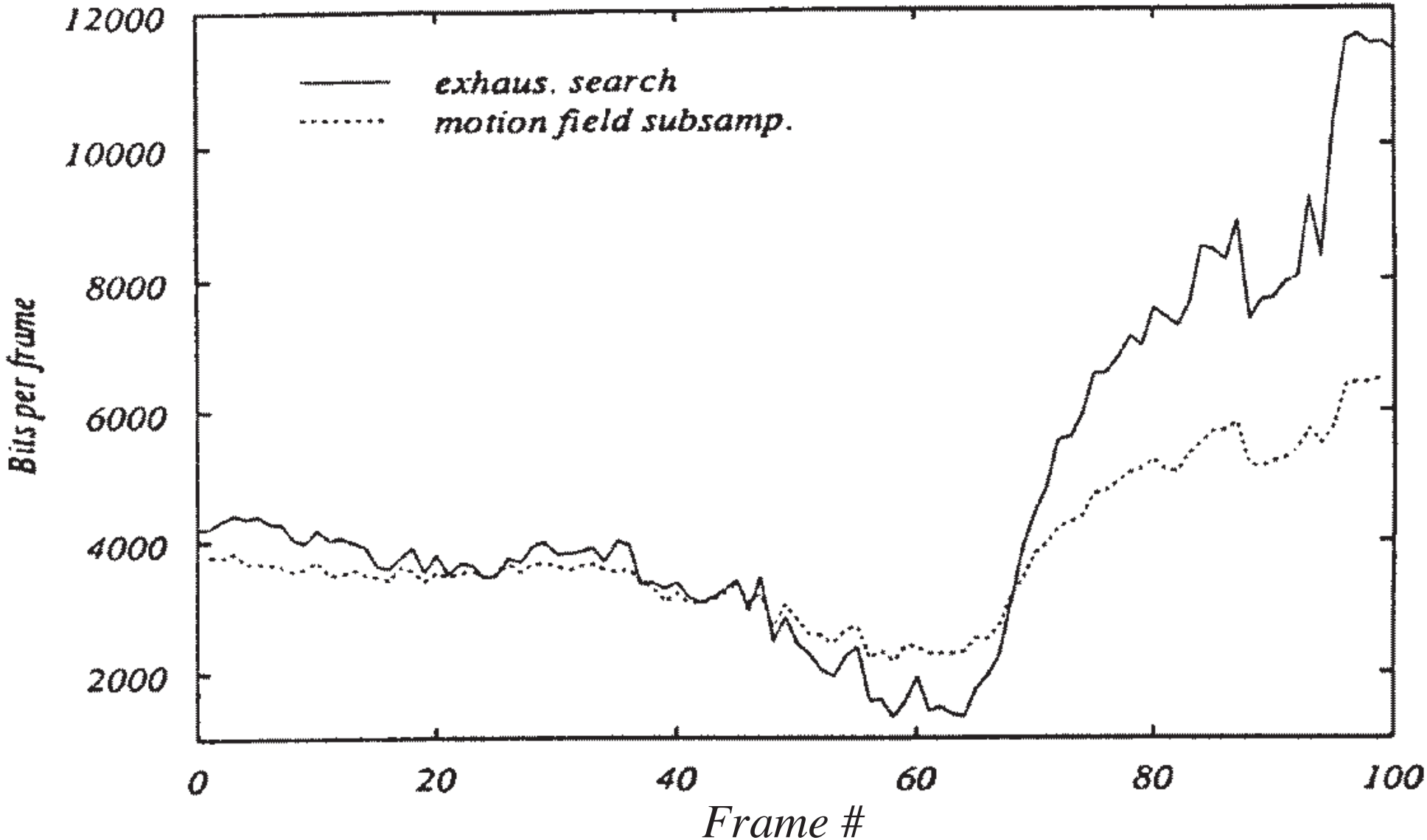
Motion-field interpolation

- Can achieve approximately 2:1 reduction in computation
- Can also achieve approximately 2:1 reduction in coding motion vectors

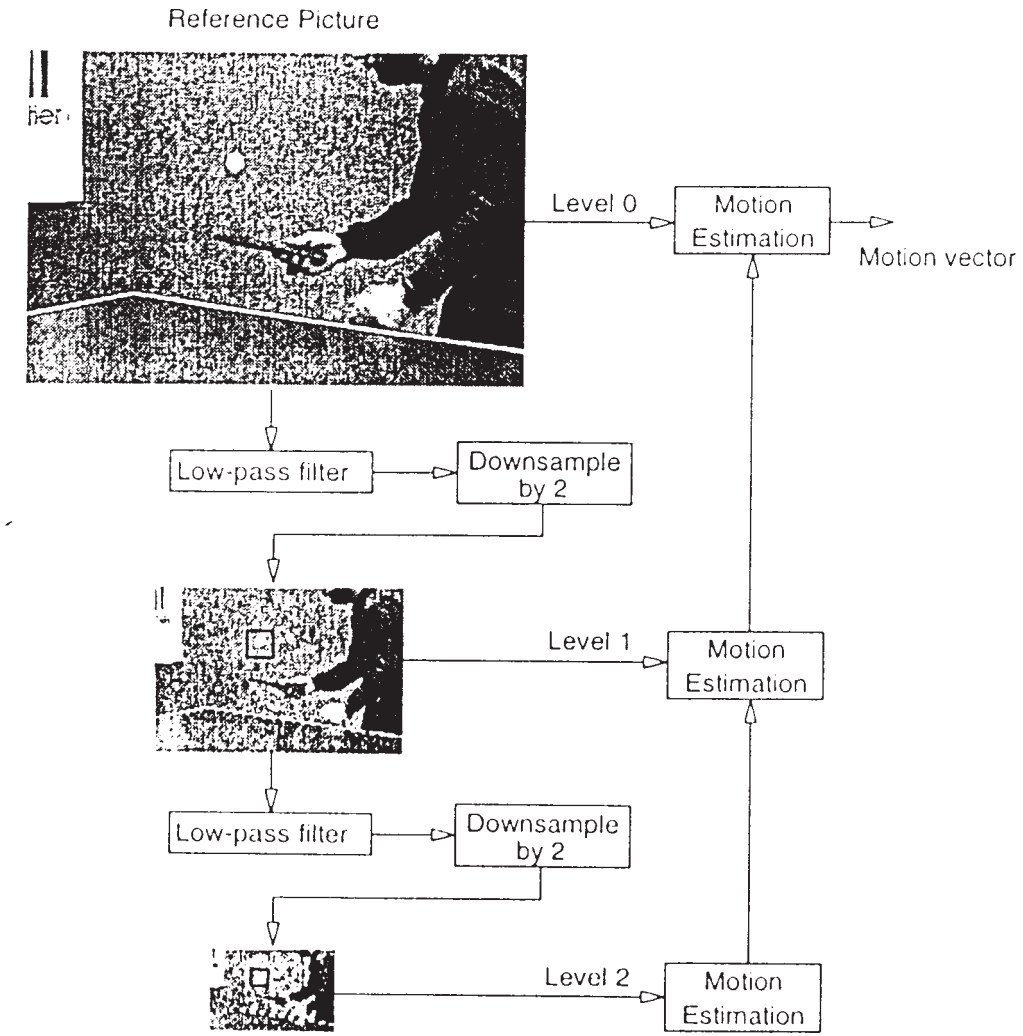
Performance



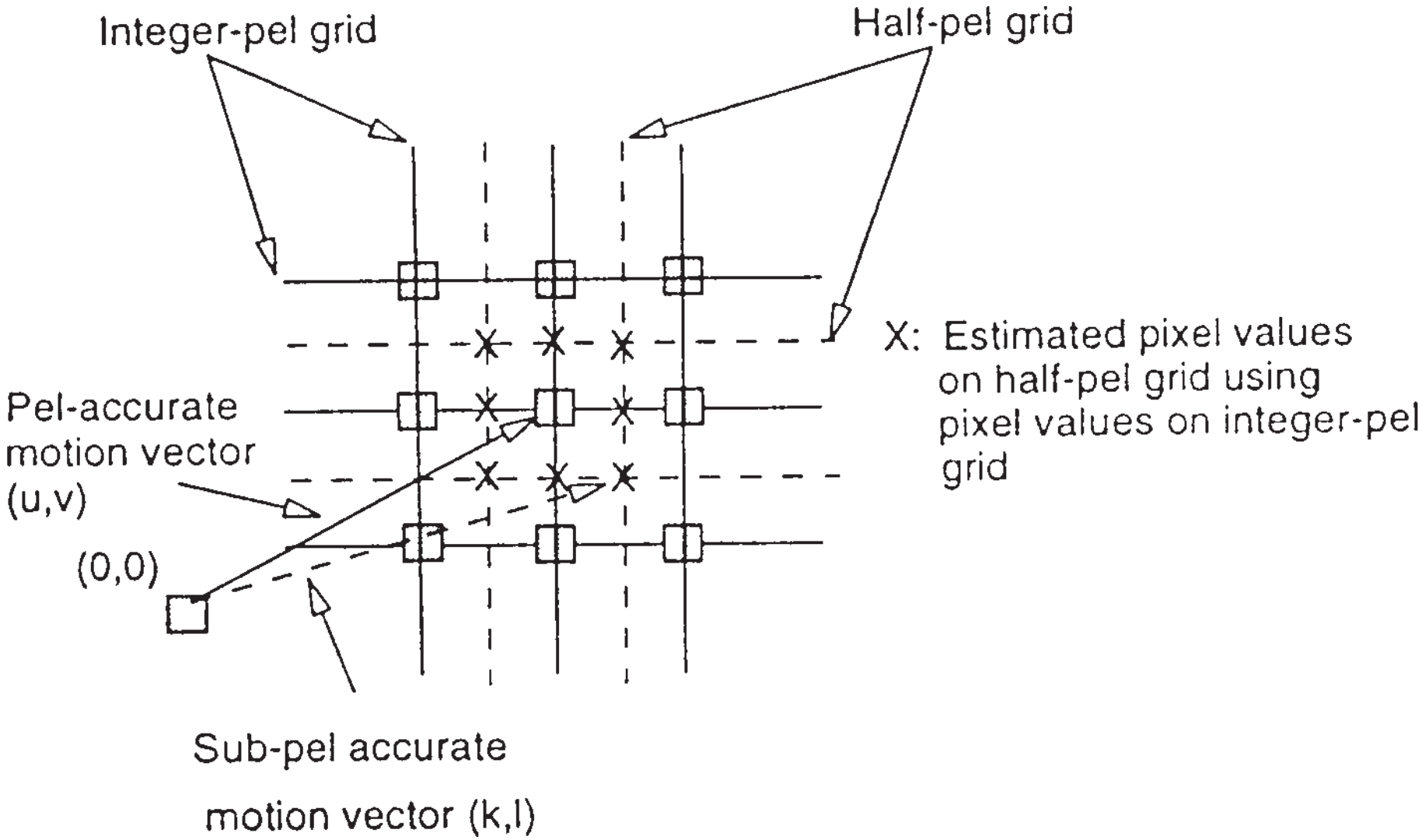
Bit-per-frame needed for subsampled Motion-Field Estimation



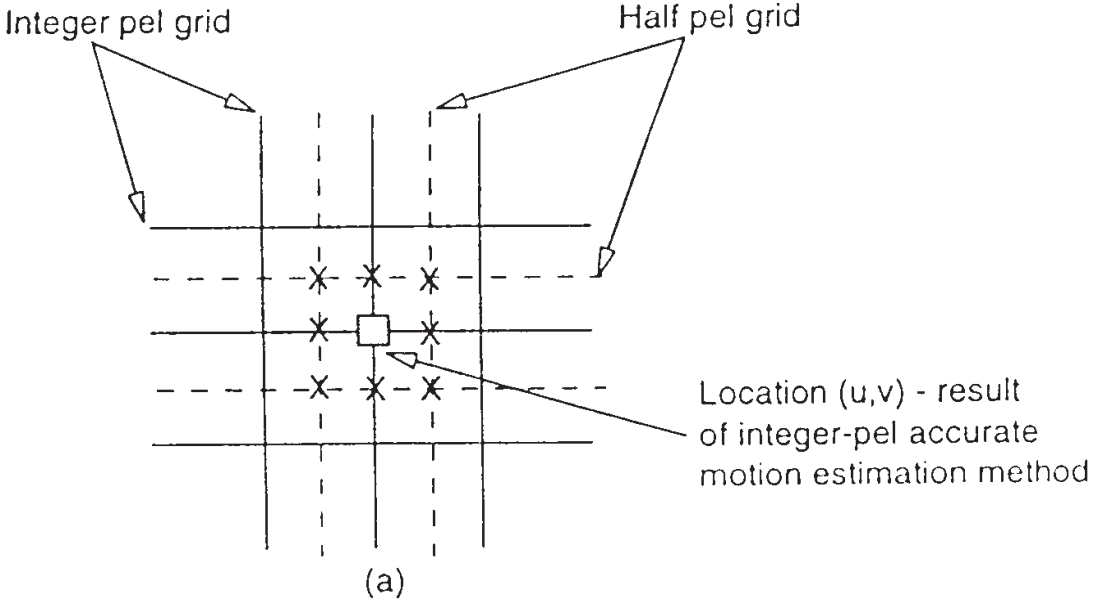
A Generic Hierarchical Motion Vector Search Strategy



Half-pixel Accurate Motion Vector Estimation



Fast method for half-pixel accurate MV estimation



Condition for locating Half-pixel MV

MV is on x_1 line if 2 (m3 - m0) < (m4 - m0)

MV is on x_2 line if (m3 - m0) > 2(M4 - m0)

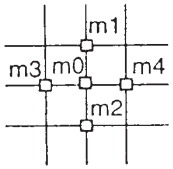
MV is on x_0 if niether of the above is true

Similarily,

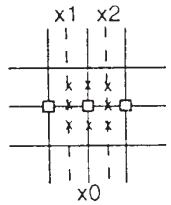
MV is on y_1 line if 2(m1 - m0) < (m2 - m0)

MV is on y_2 line if (m1 - m0) > 2(m2 - m0)

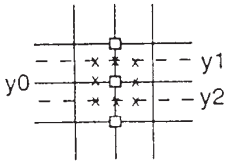
MV is on y_0 if neither of the above is true



(b)



(c)



(d)

The intersection of x_i and y_j is the location of half-pixel MV

http://scholar.harvard.edu/stanleychan/software/subpixel-motion-estimation-without-interpolation

Think back on why and what about MV?

Forward prediction; bi-directional prediction

I BBB P BBB P

Additional References

1. R. Li, B. Zeng, and M. L. Liou, "*A new three-step search algorithm for block motion estimation*", IEEE Trans. Circuit and Systems for Video Technology, Vol. 4, No. 4, pp. 438-442, aug. 1994
2. Z. L. He and M. L. Liou, "*A high performance fast search algorithm for block matching motion estimation*", IEEE Trans. Circuits and Systems for Video Technology, Vol. 7, No. 5, pp. 826-828, Oct. 1997
3. A. M. Tourapis, O. C. Au, and M. L. Liou, "*An advanced zonal blcok based algorithm for motion estimation*", Proc. IEEE Int'l Conf. On Image processing, Kobe, Japan, Oct. 1999
4. B. Liu and A. Zaccarin, "*New fast algorithms for the estimation of block motion vectors*", IEEE Trans. Circuts and Systems for Video Technology, Vol. 3, No. 2 pp.148-157, April 1993
5. Y. S. Jehng, L G. Chen, and T. D. Chuieh, "*An efficent and simple VLSI tree structure for motion estimation algorithms*", IEEE Trans. Signal Processing, Vol. 41, No. 2, pp.889-900, 1993
6. T. Komarek and P. Pirsch, "*Array architecture for block matching algorithms*", IEEE Trans. Circuits and Systems, Vol. 36, No. 10, pp. 1301-1308, 1989
7. Z. L. He, M. L. Liou, P. C. H. Chan and R. Li, "*Efficent architectures for the new three-step search algorithm*", Proc. 38th Midwest Symp. Circuits and Systems, Rio de Janeiro, Brazil, Vol. 2, pp. 1228-1231, Aug. 1995
8. Z. L. He, M. L. Liou, C. H. Chan, and C. Y. Tsui, "*Generic VLSI architecture for block-matching motion estimation algorithms*", Int'l Journal of Imaging Systems and Technology, Vol. 9, No. 4, pp. 257-273, July 1998
9. Z. L. He and M. L. Liou, "Cost effecitve VLSI architecture for full–search block-matching motion estimation algorithm", Journal of VLSI Signal Processing, Vol. 17, pp. 225-240, 1997
10. Z. L. He, C. Y. Tsui, K. K. Chan, and M. L. Liou, "Low power VLSI design for motion estimation using adaptive pixel truncation", IEEE Trans. Circuits and Systems for Video Technoogy.