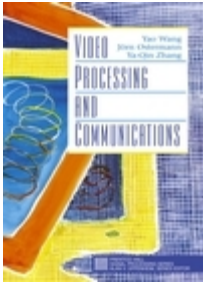


# Video Processing & Communications

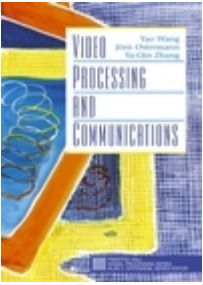
## Two-Dimensional Motion Estimation (Part I: Fundamentals & Basic Techniques)

Yao Wang  
Polytechnic University, Brooklyn, NY11201  
<http://eeweb.poly.edu/~yao>



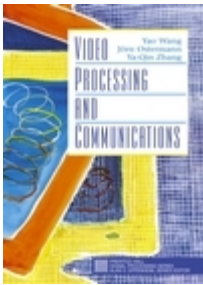
# Outline

- 3D motion model
- 2-D motion model
- 2-D motion vs. optical flow
- Optical flow equation and ambiguity in motion estimation
- General methodologies in motion estimation
  - Motion representation
  - Motion estimation criterion
  - Optimization methods
  - Gradient descent methods
- Pixel-based motion estimation
- Block-based motion estimation
  - EBMA algorithm

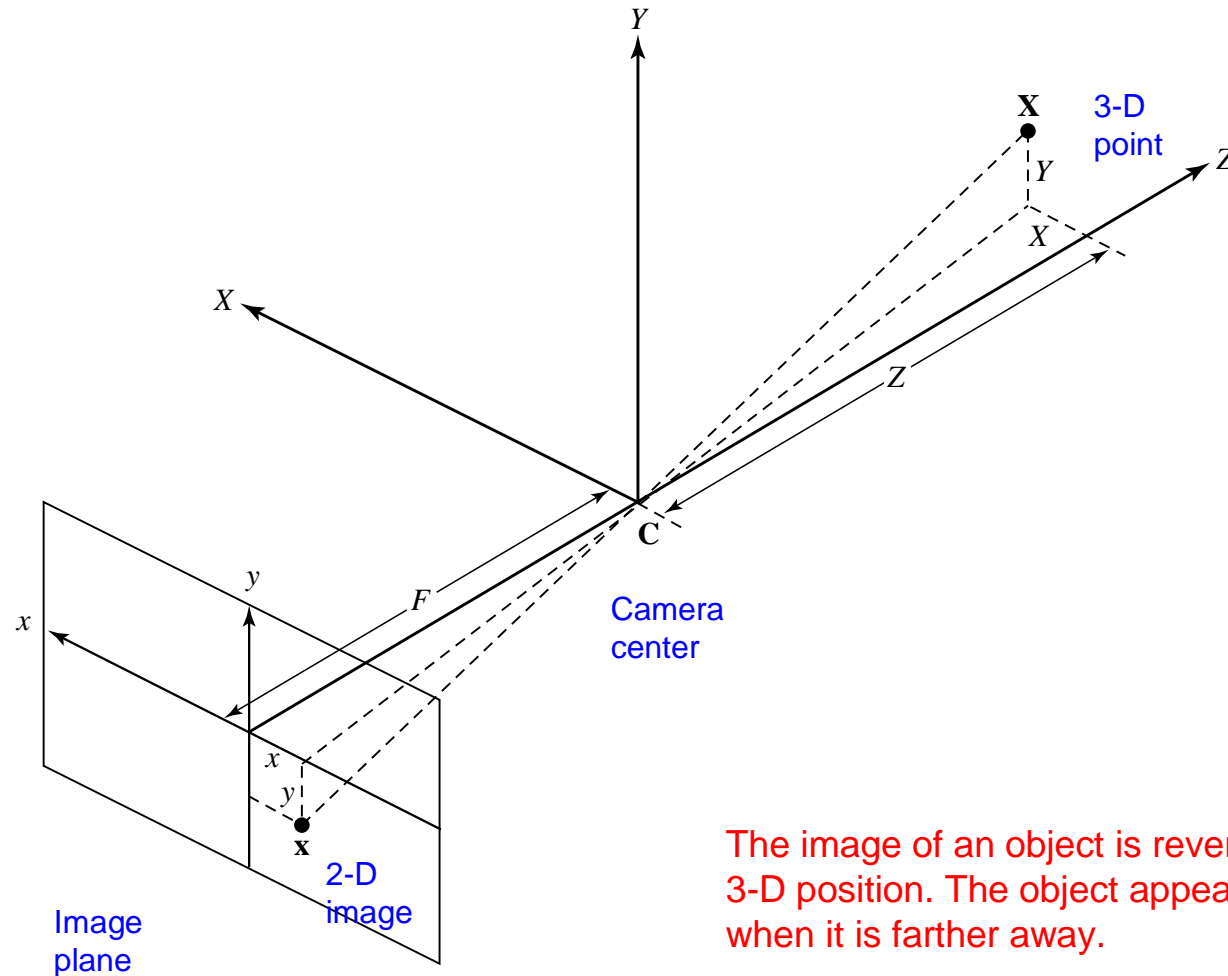


## 2-D Motion Model

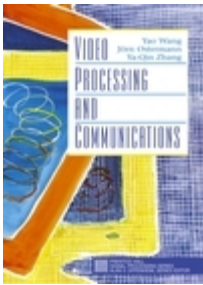
- Camera projection
- 3D motion
- Projection of 3-D motion
- 2D motion due to rigid object motion
  - Projective mapping
- Approximation of projective mapping
  - Affine model
  - Bilinear model



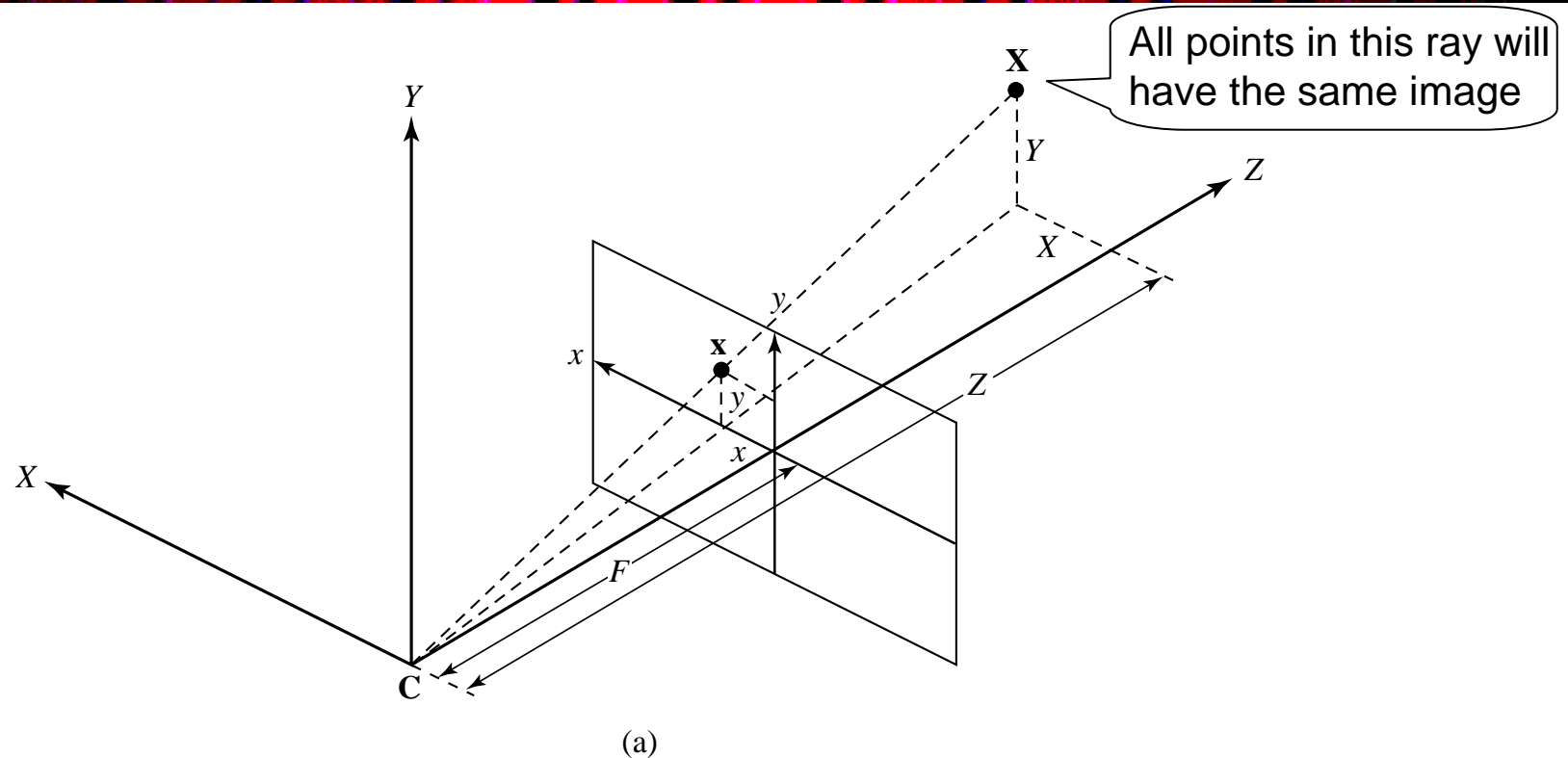
# Pinhole Camera



The image of an object is reversed from its 3-D position. The object appears smaller when it is farther away.

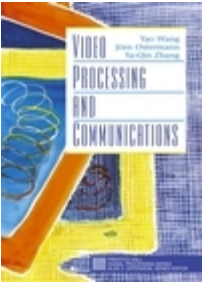


# Pinhole Camera Model: Perspective Projection

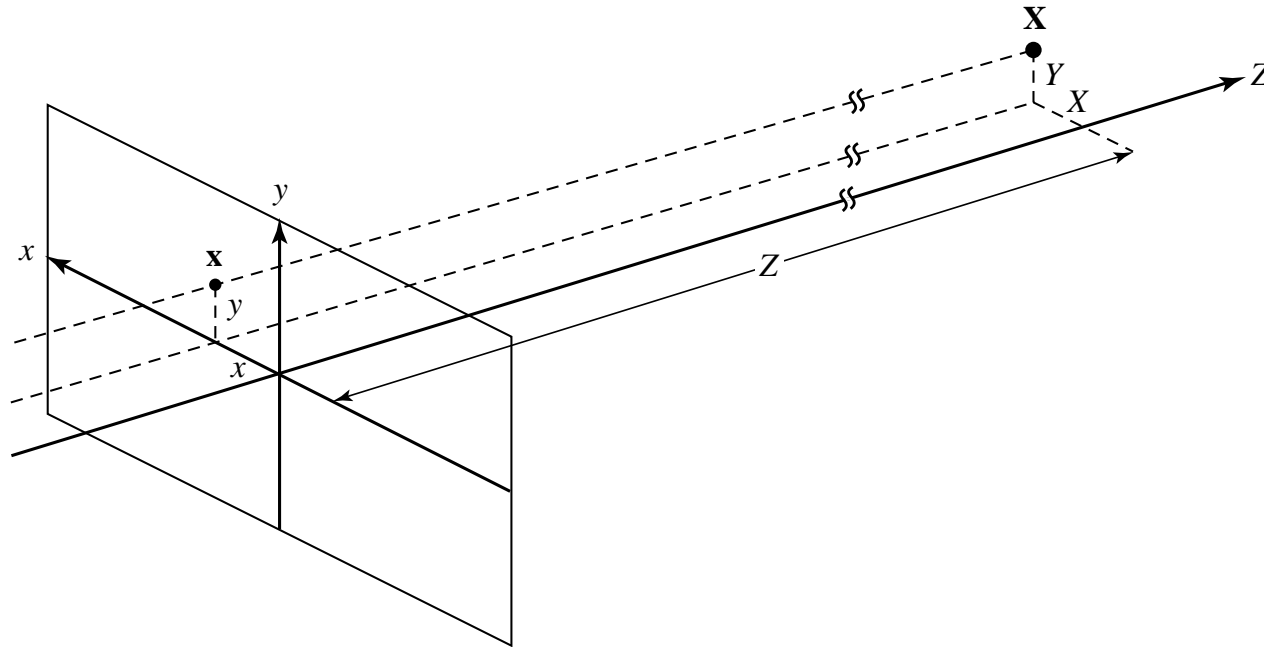


$$\frac{x}{F} = \frac{X}{Z}, \frac{y}{F} = \frac{Y}{Z} \Rightarrow x = F \frac{X}{Z}, y = F \frac{Y}{Z}$$

$x, y$  are inversely related to  $Z$



# Approximate Model: Orthographic Projection



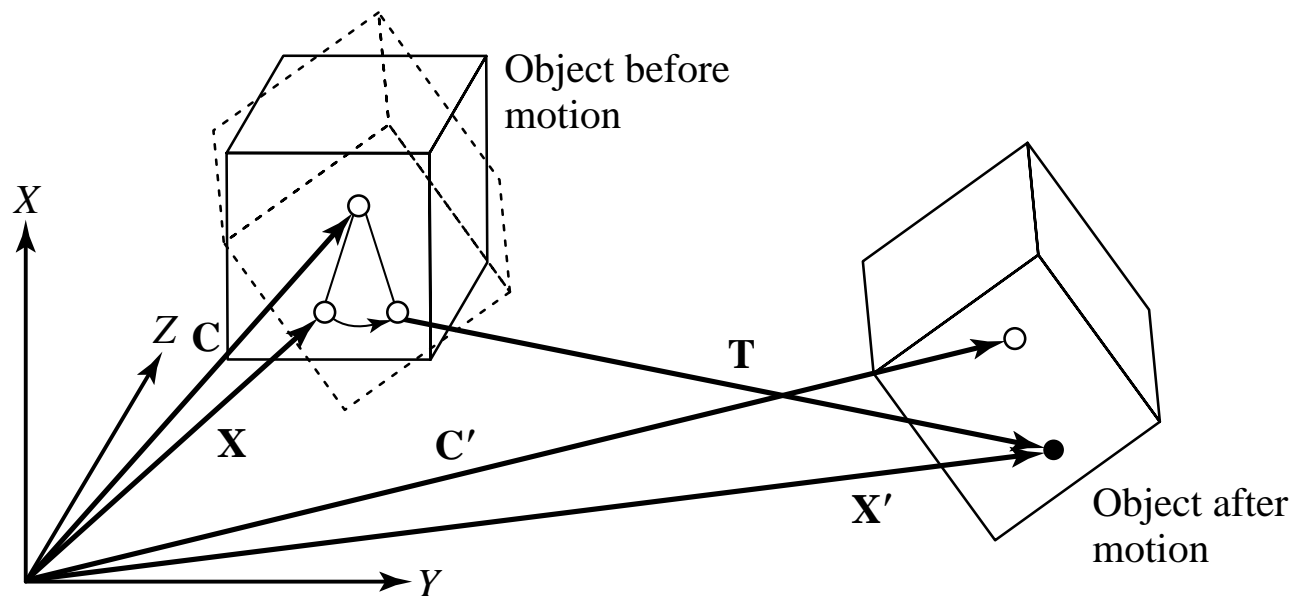
(b)

When the object is very far ( $Z \rightarrow \infty$ )

$$x = X, y = Y$$

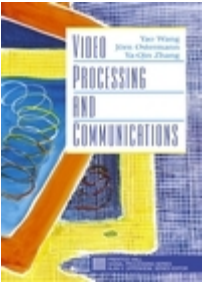
Can be used as long as the depth variation within the object is small compared to the distance of the object.

# Rigid Object Motion



Rotation and translation wrp. the object center :

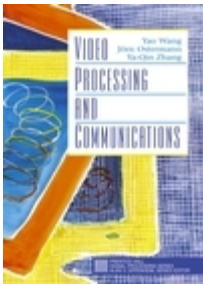
$$\mathbf{X}' = [\mathbf{R}](\mathbf{X} - \mathbf{C}) + \mathbf{T} + \mathbf{C}; \quad [\mathbf{R}] : \theta_x, \theta_y, \theta_z; \quad \mathbf{T} : T_x, T_y, T_z$$



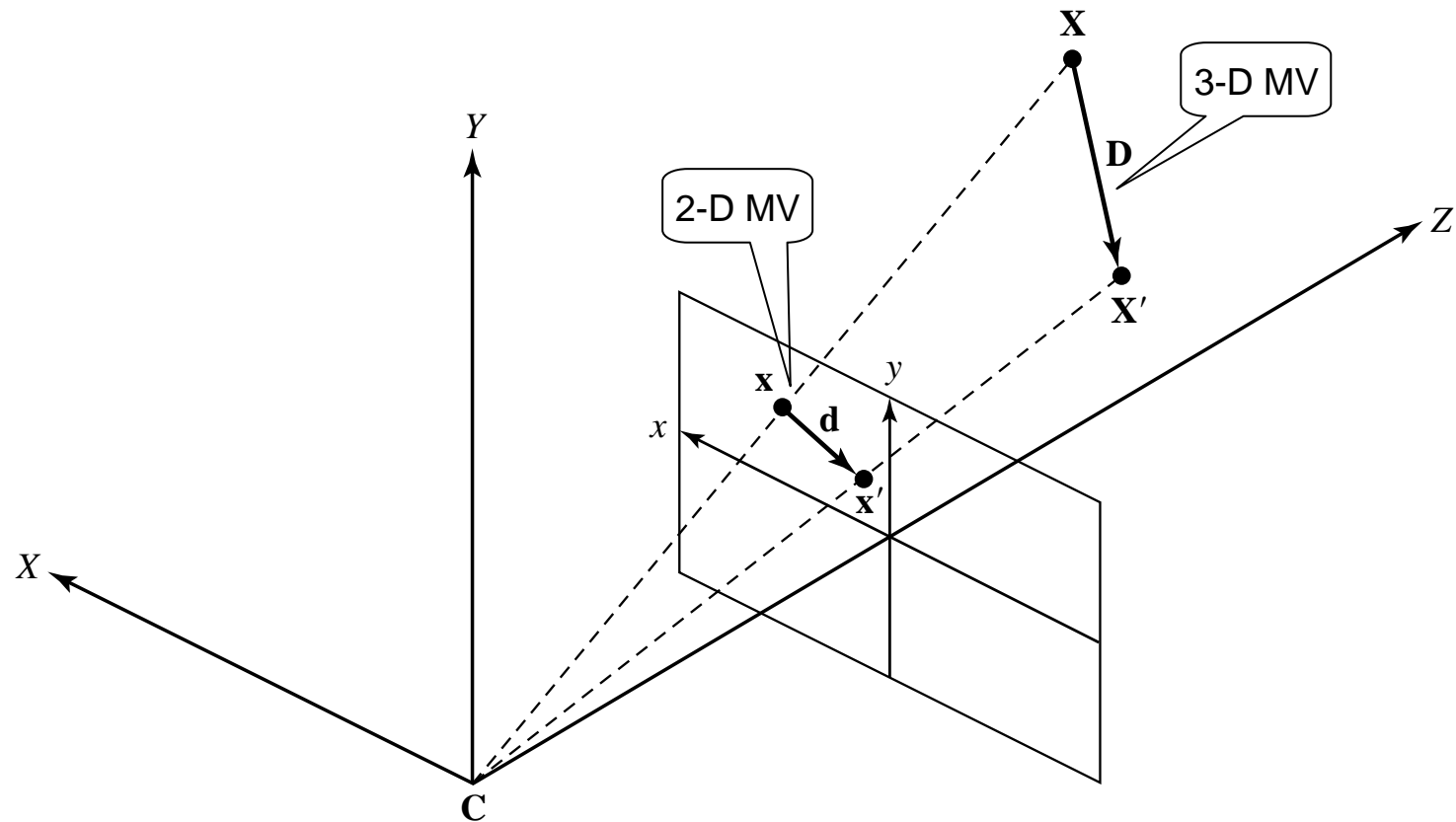
# Flexible Object Motion

- Two ways to describe
  - Decompose into multiple, but connected rigid sub-objects
  - Global motion plus local motion in sub-objects
  - Ex. Human body consists of many parts each undergo a rigid motion

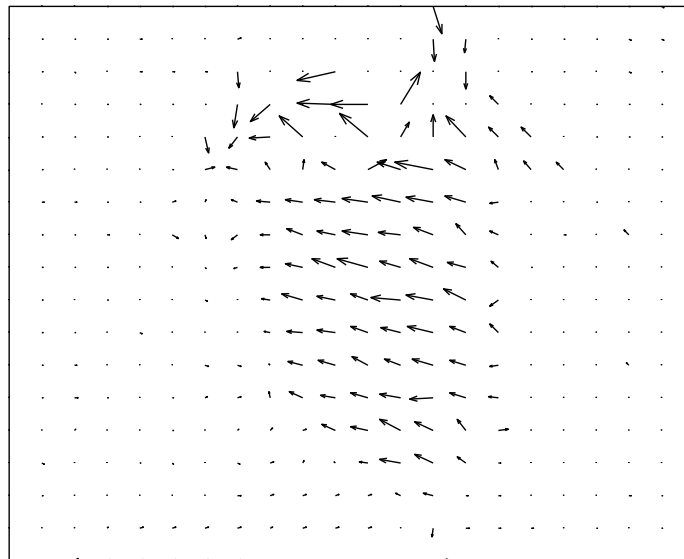




# 3-D Motion -> 2-D Motion



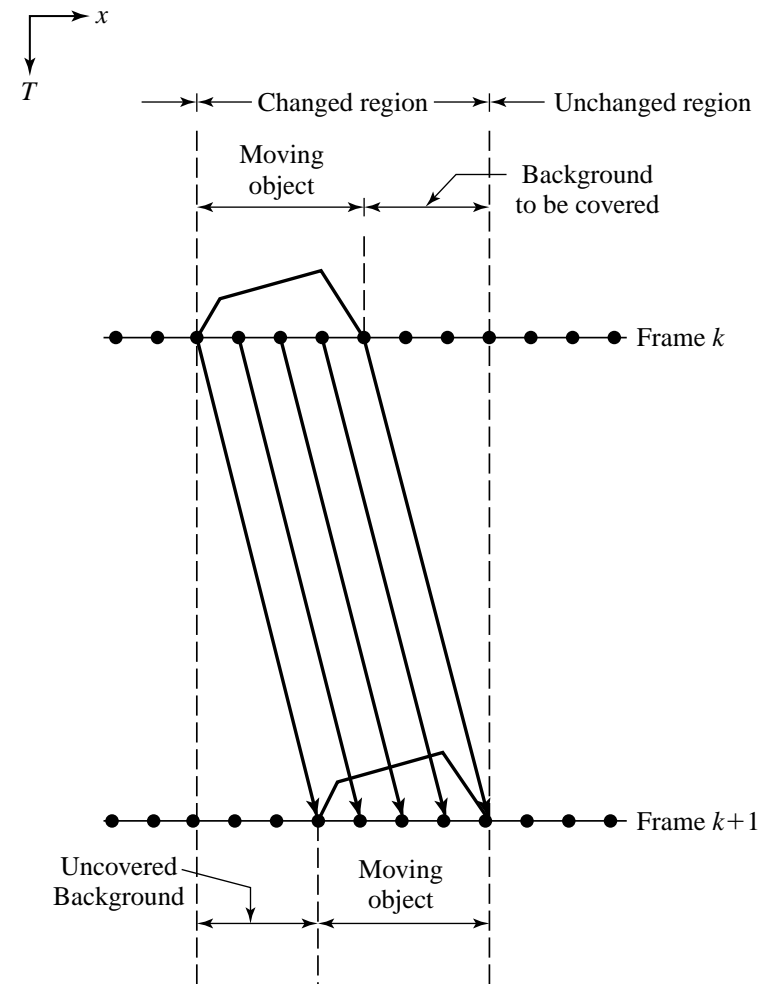
# Sample Motion Field





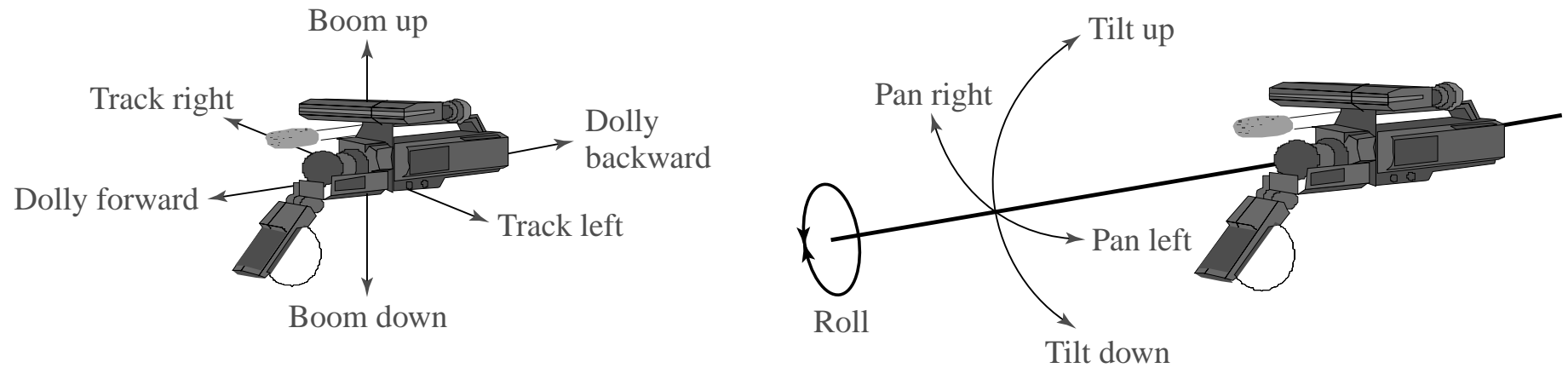
# Occlusion Effect

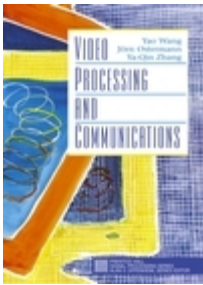
Motion is undefined in occluded regions



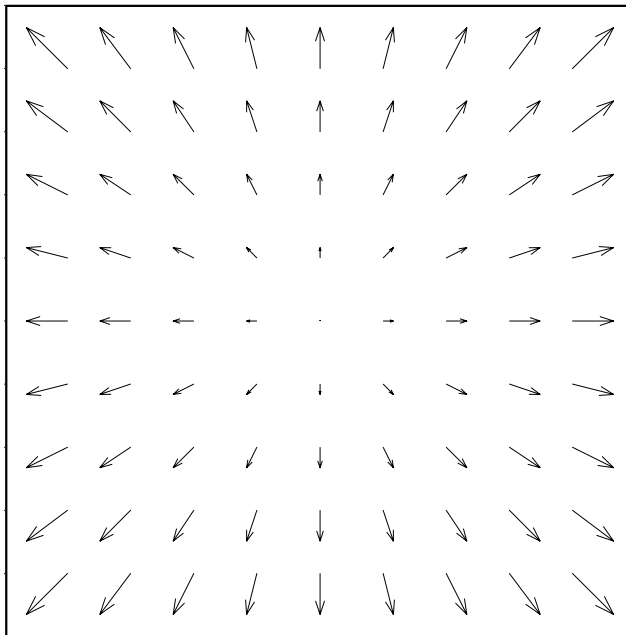


# Typical Camera Motions



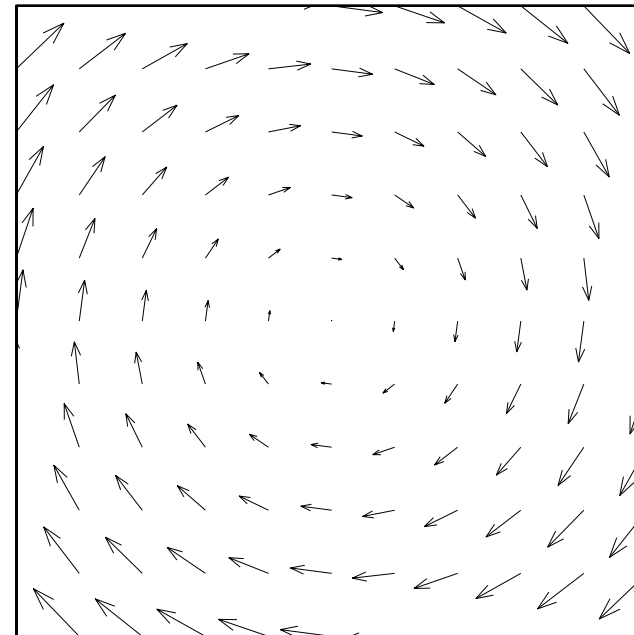


# 2-D Motion Corresponding to Camera Motion



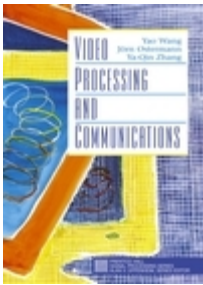
(a)

Camera zoom



(b)

Camera rotation around Z-axis (roll)



# 2-D Motion Corresponding to Rigid Object Motion

- General case:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

→  
Perspective Projection

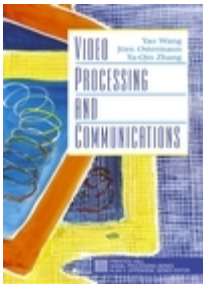
$$x' = F \frac{(r_1x + r_2y + r_3F)Z + T_xF}{(r_7x + r_8y + r_9F)Z + T_zF}$$

$$y' = F \frac{(r_4x + r_5y + r_6F)Z + T_yF}{(r_7x + r_8y + r_9F)Z + T_zF}$$

- Projective mapping:

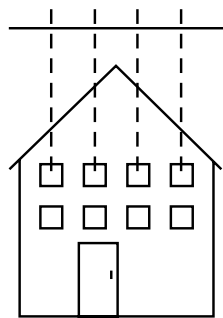
When the object surface is planar ( $Z = aX + bY + c$ ):

$$x' = \frac{a_0 + a_1x + a_2y}{1 + c_1x + c_2y}, \quad y' = \frac{b_0 + b_1x + b_2y}{1 + c_1x + c_2y}$$

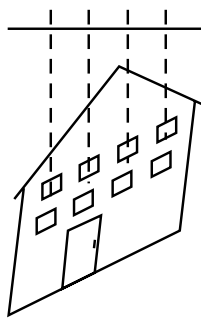


# Projective Mapping

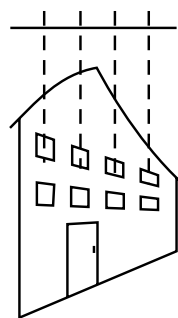
Non-chirping models



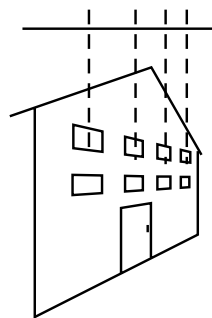
(Original)



(Affine)

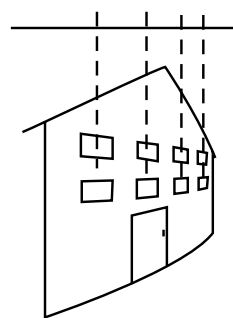


(Bilinear)

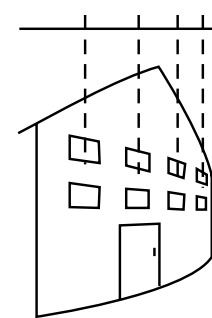


(Projective)

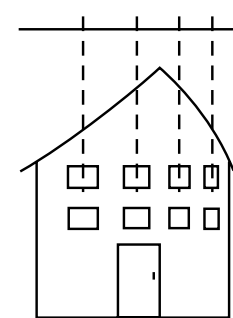
Chirping models



(Relative-projective)



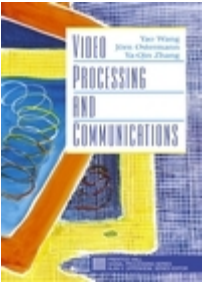
(Pseudo-perspective)



(Biquadratic)

Two features of projective mapping:

- Chirping: increasing perceived spatial frequency for far away objects
- Converging (Keystone): parallel lines converge in distance



# Affine and Bilinear Model

- Affine (6 parameters):

$$\begin{bmatrix} d_x(x, y) \\ d_y(x, y) \end{bmatrix} = \begin{bmatrix} a_0 + a_1x + a_2y \\ b_0 + b_1x + b_2y \end{bmatrix}$$

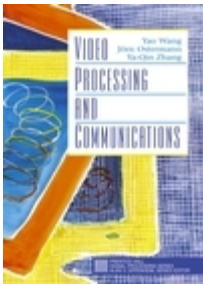
- Good for mapping triangles to triangles

- Bilinear (8 parameters):

$$\begin{bmatrix} d_x(x, y) \\ d_y(x, y) \end{bmatrix} = \begin{bmatrix} a_0 + a_1x + a_2y + a_3xy \\ b_0 + b_1x + b_2y + b_3xy \end{bmatrix}$$

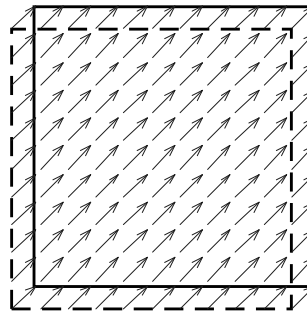
- Good for mapping blocks to quadrangles





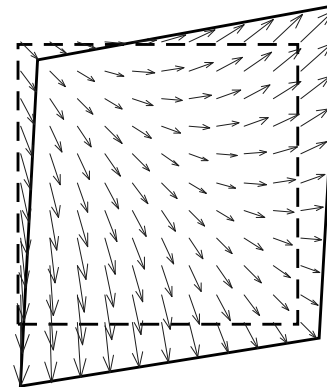
# Motion Field Corresponding to Different 2-D Motion Models

Translation



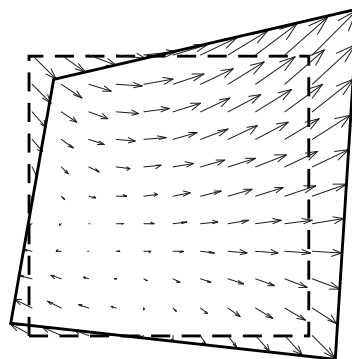
(a)

Affine



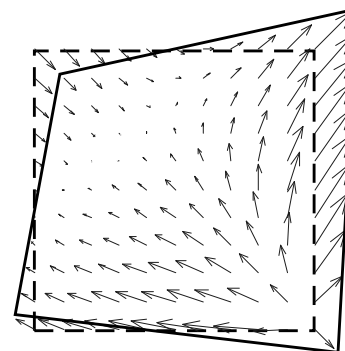
(b)

Bilinear

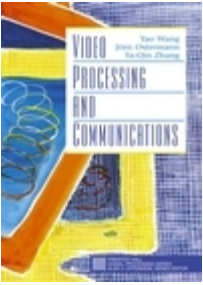


(c)

Perspective

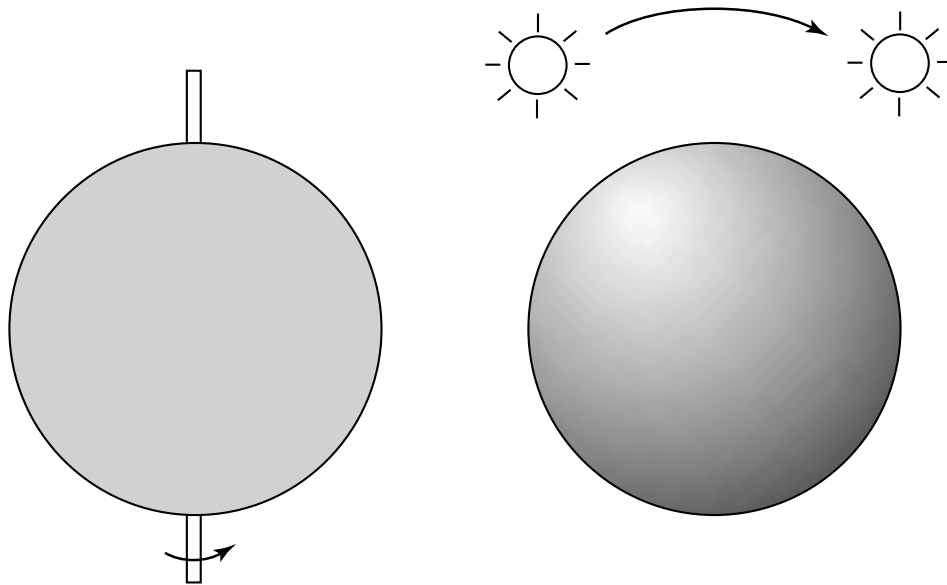


(d)



## 2-D Motion vs. Optical Flow

- 2-D Motion: Projection of 3-D motion, depending on 3D object motion and projection operator
- Optical flow: “Perceived” 2-D motion based on changes in image pattern, also depends on illumination and object surface texture



On the left, a sphere is rotating under a constant ambient illumination, but the observed image does not change.

On the right, a point light source is rotating around a stationary sphere, causing the highlight point on the sphere to rotate.



# Optical Flow Equation

- When illumination condition is unknown, the best one can do it to estimate optical flow.
- Constant intensity assumption -> Optical flow equation

Under "constant intensity assumption":

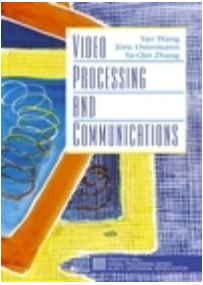
$$\psi(x + d_x, y + d_y, t + d_t) = \psi(x, y, t)$$

But, using Taylor's expansion :

$$\psi(x + d_x, y + d_y, t + d_t) = \psi(x, y, t) + \frac{\partial \psi}{\partial x} d_x + \frac{\partial \psi}{\partial y} d_y + \frac{\partial \psi}{\partial t} d_t$$

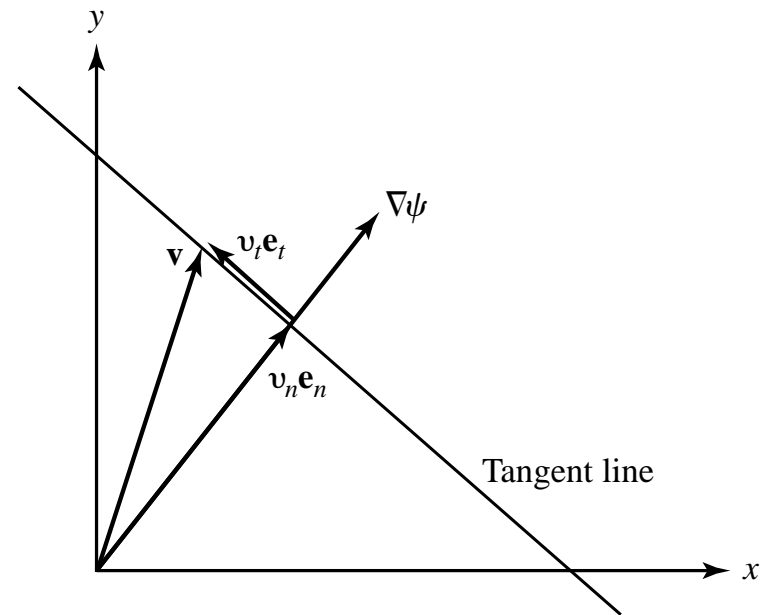
Compare the above two, we have the optical flow equation :

$$\frac{\partial \psi}{\partial x} d_x + \frac{\partial \psi}{\partial y} d_y + \frac{\partial \psi}{\partial t} d_t = 0 \quad \text{or} \quad \frac{\partial \psi}{\partial x} v_x + \frac{\partial \psi}{\partial y} v_y + \frac{\partial \psi}{\partial t} = 0 \quad \text{or} \quad \nabla \psi^T \mathbf{v} + \frac{\partial \psi}{\partial t} = 0$$

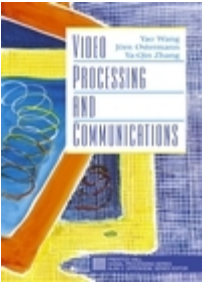


# Ambiguities in Motion Estimation

- Optical flow equation only constrains the flow vector in the gradient direction  $v_n$
- The flow vector in the tangent direction ( $v_t$ ) is under-determined
- In regions with constant brightness ( $\nabla \psi = 0$ ), the flow is indeterminate -> Motion estimation is unreliable in regions with flat texture, more reliable near edges

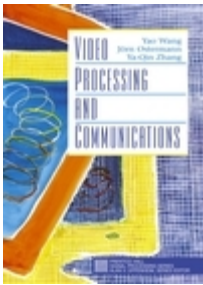


$$\mathbf{v} = v_n \mathbf{e}_n + v_t \mathbf{e}_t$$
$$v_n \|\nabla \psi\| + \frac{\partial \psi}{\partial t} = 0$$



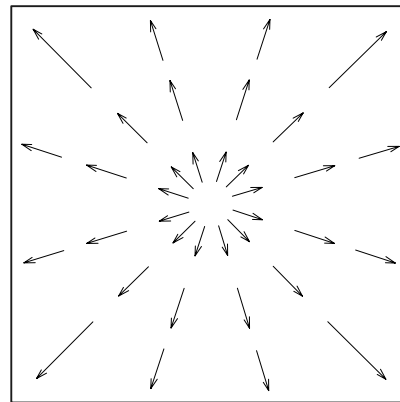
# General Considerations for Motion Estimation

- Two categories of approaches:
  - **Feature based** (more often used in object tracking, 3D reconstruction from 2D)
  - **Intensity based** (based on constant intensity assumption) (more often used for motion compensated prediction, required in video coding, frame interpolation) -> Our focus
- Three important questions
  - How to represent the motion field?
  - What criteria to use to estimate motion parameters?
  - How to search motion parameters?



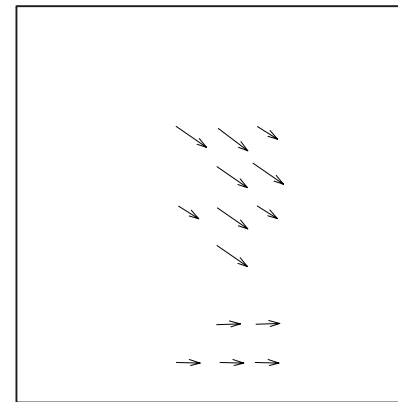
# Motion Representation

Global:  
Entire motion field is represented by a few global parameters



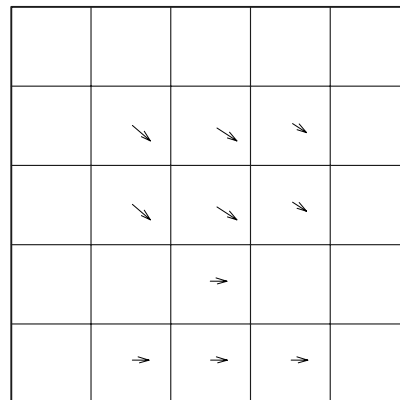
(a)

Pixel-based:  
One MV at each pixel, with some smoothness constraint between adjacent MVs.



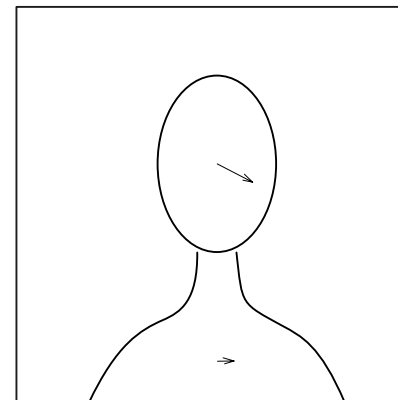
(b)

Block-based:  
Entire frame is divided into blocks, and motion in each block is characterized by a few parameters.



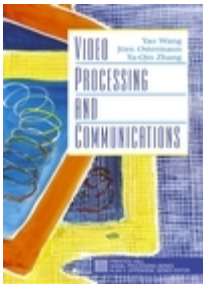
(c)

Region-based:  
Entire frame is divided into regions, each region corresponding to an object or sub-object with consistent motion, represented by a few parameters.

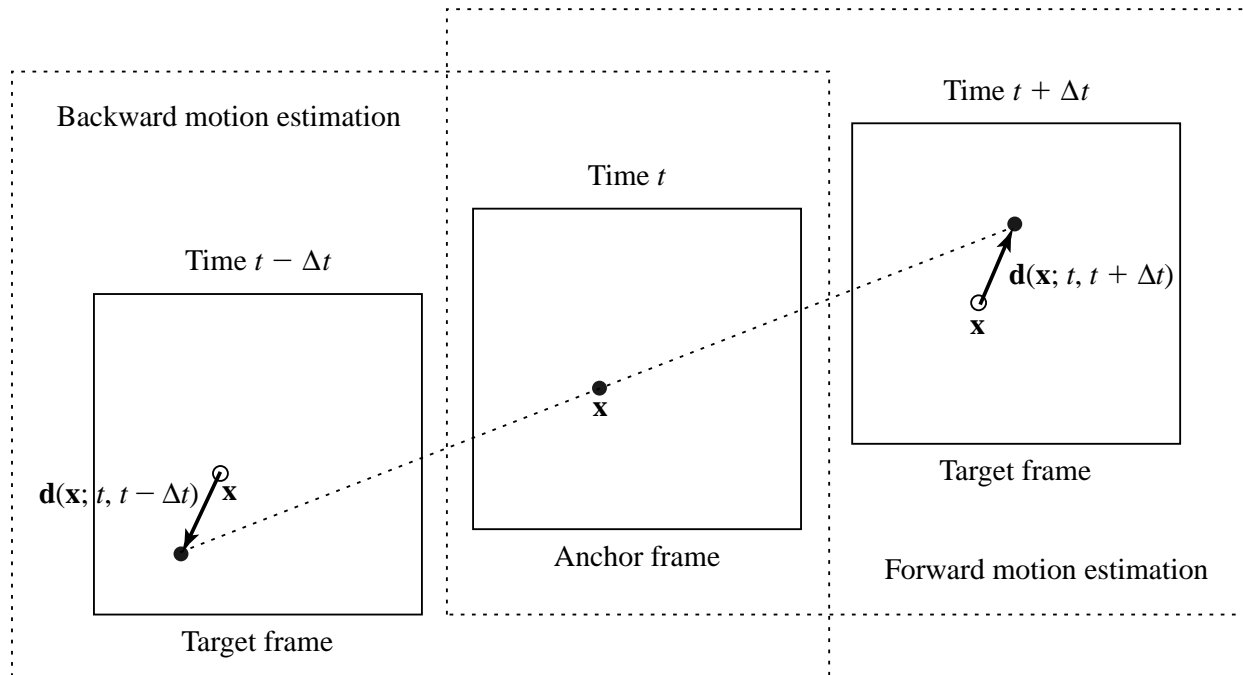


(d)

Other representation: mesh-based (control grid) (to be discussed later)



# Notations



Anchor frame:  $\psi_1(\mathbf{x})$

Target frame:  $\psi_2(\mathbf{x})$

Motion parameters:  $\mathbf{a}$

Motion vector at a pixel in the anchor frame:  $\mathbf{d}(\mathbf{x})$

Motion field:  $\mathbf{d}(\mathbf{x}; \mathbf{a}), \mathbf{x} \in \Lambda$

Mapping function:

$$\mathbf{w}(\mathbf{x}; \mathbf{a}) = \mathbf{x} + \mathbf{d}(\mathbf{x}; \mathbf{a}), \mathbf{x} \in \Lambda$$



# Motion Estimation Criterion

- To minimize the displaced frame difference (DFD)

$$E_{\text{DFD}}(\mathbf{a}) = \sum_{\mathbf{x} \in \Lambda} |\psi_2(\mathbf{x} + \mathbf{d}(\mathbf{x}; \mathbf{a})) - \psi_1(\mathbf{x})|^p \rightarrow \min$$

$$p = 1 : \text{MAD}; \quad p = 2 : \text{MSE}$$

- To satisfy the optical flow equation

$$E_{\text{OF}}(\mathbf{a}) = \sum_{\mathbf{x} \in \Lambda} |(\nabla \psi_1(\mathbf{x}))^T \mathbf{d}(\mathbf{x}; \mathbf{a}) + \psi_2(\mathbf{x}) - \psi_1(\mathbf{x})|^p \rightarrow \min$$

- To impose additional smoothness constraint using regularization technique (Important in pixel- and block-based representation)

$$E_s(\mathbf{a}) = \sum_{\mathbf{x} \in \Lambda} \sum_{\mathbf{y} \in N_x} \|\mathbf{d}(\mathbf{x}; \mathbf{a}) - \mathbf{d}(\mathbf{y}; \mathbf{a})\|^2$$

$$w_{\text{DFD}} E_{\text{DFD}}(\mathbf{a}) + w_s E_s(\mathbf{a}) \rightarrow \min$$

- Bayesian (MAP) criterion: to maximize the a posteriori probability

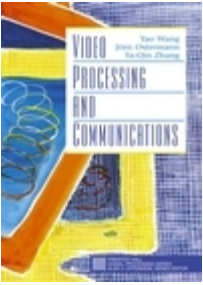
$$P(D = \mathbf{d} | \psi_2, \psi_1) \rightarrow \max$$





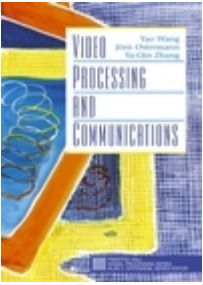
# Relation Among Different Criteria

- OF criterion is good only if motion is small.
- OF criterion can often yield closed-form solution as the objective function is quadratic in MVs.
- When the motion is not small, can iterate the solution based on the OF criterion to satisfy the DFD criterion.
- Bayesian criterion can be reduced to the DFD criterion plus motion smoothness constraint
- More in the textbook



# Optimization Methods

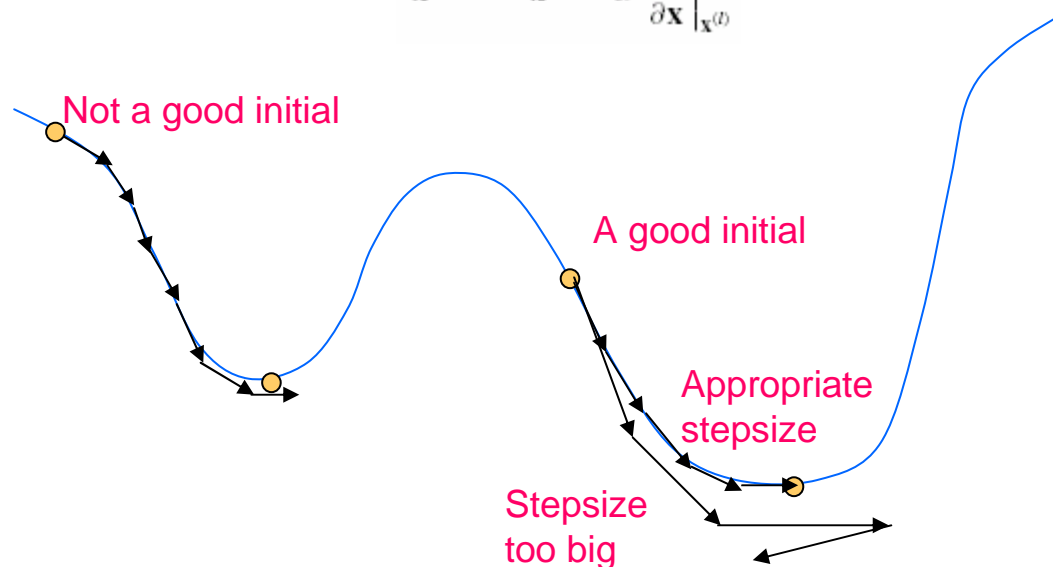
- Exhaustive search
  - Typically used for the DFD criterion with  $p=1$  (MAD)
  - Guarantees reaching the global optimal
  - Computation required may be unacceptable when number of parameters to search simultaneously is large!
  - Fast search algorithms reach sub-optimal solution in shorter time
- Gradient-based search
  - Typically used for the DFD or OF criterion with  $p=2$  (MSE)
    - the gradient can often be calculated analytically
    - When used with the OF criterion, closed-form solution may be obtained
  - Reaches the local optimal point closest to the initial solution
- Multi-resolution search
  - Search from coarse to fine resolution, faster than exhaustive search
  - Avoid being trapped into a local minimum



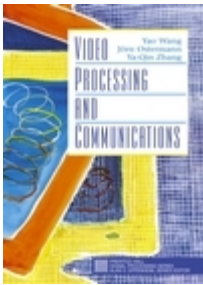
# Gradient Descent Method

- Iteratively update the current estimate in the direction opposite the gradient direction.

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} - \alpha \left. \frac{\partial J}{\partial \mathbf{x}} \right|_{\mathbf{x}^{(l)}}$$



- The solution depends on the initial condition. Reaches the local minimum closest to the initial condition
- Choice of step size:
  - Fixed stepsize: Stepsize must be small to avoid oscillation, requires many iterations
  - Steepest gradient descent (adjust stepsize optimally)



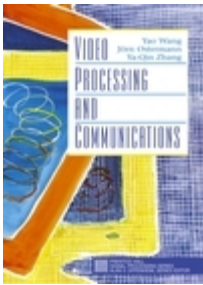
# Newton's Method

- Newton's method

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} - \alpha [\mathbf{H}(\mathbf{x}^{(l)})]^{-1} \frac{\partial J}{\partial \mathbf{x}} \bigg|_{\mathbf{x}^{(l)}}$$

$$[\mathbf{H}(\mathbf{x})] = \frac{\partial^2 J}{\partial \mathbf{x}^2} = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1 \partial x_1} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_K} \\ \frac{\partial^2 J}{\partial x_2 \partial x_1} & \frac{\partial^2 J}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_2 \partial x_K} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 J}{\partial x_K \partial x_1} & \frac{\partial^2 J}{\partial x_K \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_K \partial x_K} \end{bmatrix}$$

- Converges faster than 1<sup>st</sup> order method (i.e. requires fewer number of iterations to reach convergence)
- Requires more calculation in each iteration
- More prone to noise (gradient calculation is subject to noise, more so with 2<sup>nd</sup> order than with 1<sup>st</sup> order)
- May not converge if  $\alpha \geq 1$ . Should choose  $\alpha$  appropriate to reach a good compromise between guaranteeing convergence and the convergence rate.



# Newton-Raphson Method

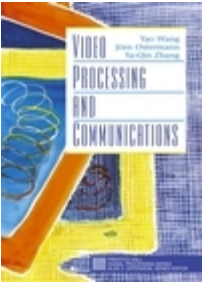
- Newton-Raphson method
  - Approximate 2<sup>nd</sup> order gradient with product of 1<sup>st</sup> order gradients
  - Applicable when the objective function is a sum of squared errors
  - Only needs to calculate 1<sup>st</sup> order gradients, yet converge at a rate similar to Newton's method.

$$J(\mathbf{x}) = \frac{1}{2} \sum_k e_k^2(\mathbf{x}),$$

$$\frac{\partial J}{\partial \mathbf{x}} = \sum \frac{\partial e_k}{\partial \mathbf{x}} e_k(\mathbf{x}).$$

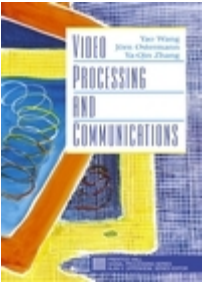
$$[\mathbf{H}] = \frac{\partial^2 J}{\partial \mathbf{x}^2} = \sum \frac{\partial e_k}{\partial \mathbf{x}} \left( \frac{\partial e_k}{\partial \mathbf{x}} \right)^T + \frac{\partial^2 e_k}{\partial \mathbf{x}^2} e_k(\mathbf{x}) \approx \sum \frac{\partial e_k}{\partial \mathbf{x}} \left( \frac{\partial e_k}{\partial \mathbf{x}} \right)^T$$

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} - \alpha [\mathbf{H}(\mathbf{x}^{(l)})]^{-1} \left. \frac{\partial J}{\partial \mathbf{x}} \right|_{\mathbf{x}^{(l)}}$$



# Pixel-Based Motion Estimation

- Horn-Schunck method
  - OF + smoothness criterion
- Multipoint neighborhood method
  - Assuming every pixel in a small block surrounding a pixel has the same MV
- Pel-recursrive method
  - MV for a current pel is updated from those of its previous pels, so that the MV does not need to be coded
  - Developed for early generation of video coder

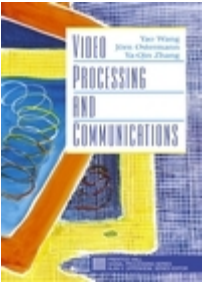


# Multipoint Neighborhood Method

- Estimate the MV at each pixel independently, by minimizing the DFD error over a neighborhood surrounding this pixel
- Every pixel in the neighborhood is assumed to have the same MV
- Minimizing function:

$$E_{\text{DFD}}(\mathbf{d}_n) = \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) |\psi_2(\mathbf{x} + \mathbf{d}_n) - \psi_1(\mathbf{x})|^2 \rightarrow \min$$

- Optimization method:
  - Exhaustive search (feasible as one only needs to search one MV at a time)
    - Need to select appropriate search range and search step-size
  - Gradient-based method



# Example: Gradient Descent Method

$$E_{\text{DFD}}(\mathbf{d}_n) = \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) |\psi_2(\mathbf{x} + \mathbf{d}_n) - \psi_1(\mathbf{x})|^2 \rightarrow \min$$

$$\mathbf{g}(\mathbf{d}_n) = \frac{\partial E}{\partial \mathbf{d}_n} = \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) e(\mathbf{x} + \mathbf{d}_n) \frac{\partial \psi_2}{\partial \mathbf{x}} \bigg|_{\mathbf{x} + \mathbf{d}_n}$$

$$\begin{aligned} [\mathbf{H}(\mathbf{d}_n)] &= \frac{\partial^2 E}{\partial \mathbf{d}_n^2} = \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) \frac{\partial \psi_2}{\partial \mathbf{x}} \left( \frac{\partial \psi_2}{\partial \mathbf{x}} \right)^T \bigg|_{\mathbf{x} + \mathbf{d}_n} + w(\mathbf{x}) e(\mathbf{x} + \mathbf{d}_n) \frac{\partial^2 \psi_2}{\partial \mathbf{x}^2} \bigg|_{\mathbf{x} + \mathbf{d}_n} \\ &\approx \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) \frac{\partial \psi_2}{\partial \mathbf{x}} \left( \frac{\partial \psi_2}{\partial \mathbf{x}} \right)^T \bigg|_{\mathbf{x} + \mathbf{d}_n} \end{aligned}$$

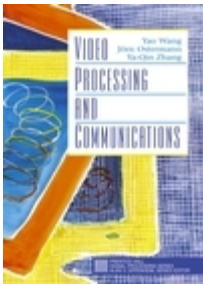
First order gradient descent :

$$\mathbf{d}_n^{(l+1)} = \mathbf{d}_n^{(l)} - \alpha \mathbf{g}(\mathbf{d}_n^{(l)})$$

Newton - Raphson method :

$$\mathbf{d}_n^{(l+1)} = \mathbf{d}_n^{(l)} - \alpha [\mathbf{H}(\mathbf{d}_n^{(l)})]^{-1} \mathbf{g}(\mathbf{d}_n^{(l)})$$





# Simplification Using OF Criterion

$$E_{\text{OF}}(\mathbf{d}_n) = \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) \left| (\nabla \psi_1(\mathbf{x}))^T \mathbf{d}_n + \psi_2(\mathbf{x}) - \psi_1(\mathbf{x}) \right|^2 \rightarrow \min$$

$$\frac{\partial E}{\partial \mathbf{d}_n} = \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) \left( (\nabla \psi_1(\mathbf{x}))^T \mathbf{d}_n + \psi_2(\mathbf{x}) - \psi_1(\mathbf{x}) \right) \nabla \psi_1(\mathbf{x}) = 0$$

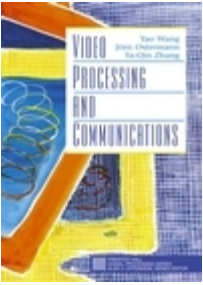
$$\mathbf{d}_{n,\text{opt}} = \left( \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) \nabla \psi_1(\mathbf{x}) (\nabla \psi_1(\mathbf{x}))^T \right)^{-1} \left( \sum_{\mathbf{x} \in B(\mathbf{x}_n)} w(\mathbf{x}) (\psi_1(\mathbf{x}) - \psi_2(\mathbf{x})) \nabla \psi_1(\mathbf{x}) \right)$$

The solution is good only if the actual MV is small. When this is not the case, one should iterate the above solution, with the following update:

$$\psi_2^{(l+1)}(\mathbf{x}) = \psi_2(\mathbf{x} + \mathbf{d}_n^{(l)})$$

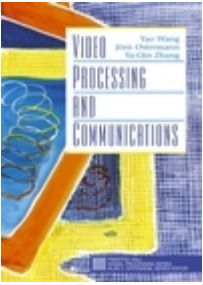
$$\mathbf{d}_n^{(l+1)} = \mathbf{d}_n^{(l)} + \Delta_n^{(l+1)}$$

**where**  $\Delta_n^{(l+1)}$  denote the MV found at that iteration



# Block-Based Motion Estimation: Overview

- Assume all pixels in a block undergo a coherent motion, and search for the motion parameters for each block independently
- Block matching algorithm (BMA): assume translational motion, 1 MV per block (2 parameter)
  - Exhaustive BMA (EBMA)
  - Fast algorithms
- Deformable block matching algorithm (DBMA): allow more complex motion (affine, bilinear), to be discussed later.



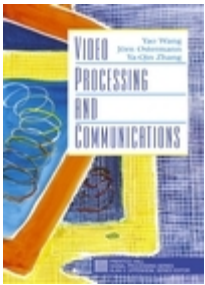
# Block Matching Algorithm

- Overview:
  - Assume all pixels in a block undergo a translation, denoted by a single MV
  - Estimate the MV for each block independently, by minimizing the DFD error over this block

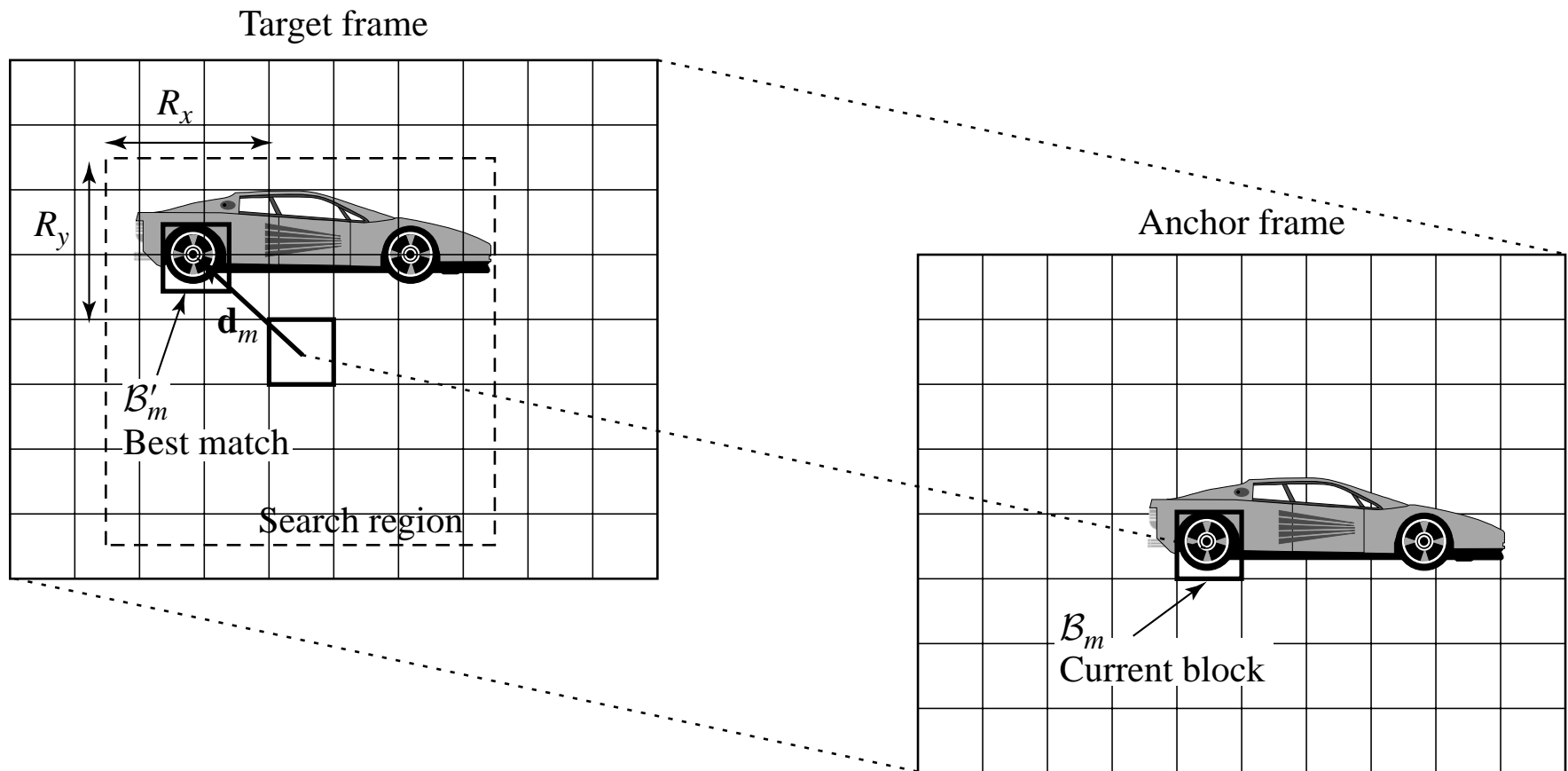
- Minimizing function:

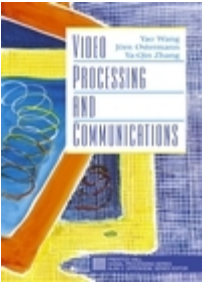
$$E_{\text{DFD}}(\mathbf{d}_m) = \sum_{\mathbf{x} \in B_m} |\psi_2(\mathbf{x} + \mathbf{d}_m) - \psi_1(\mathbf{x})|^p \rightarrow \min$$

- Optimization method:
  - Exhaustive search (feasible as one only needs to search one MV at a time), using MAD criterion (p=1)
  - Fast search algorithms
  - Integer vs. fractional pel accuracy search



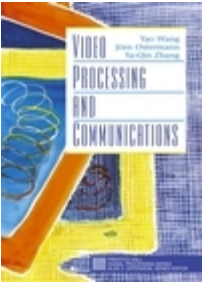
# Exhaustive Block Matching Algorithm (EBMA)





# Complexity of Integer-Pel EBMA

- Assumption
  - Image size:  $M \times M$
  - Block size:  $N \times N$
  - 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 =  $2.85 \times 10^8$ /frame =  $8.55 \times 10^9$ /second
- Regular structure suitable for VLSI implementation
- Challenging for software-only implementation

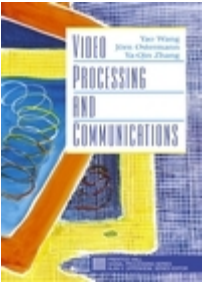


# Sample Matlab Script for Integer-pel EBMA

```
%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;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<MAD_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 count in MAD. This program is meant to illustrate the main operations involved. Not the actual working matlab script.

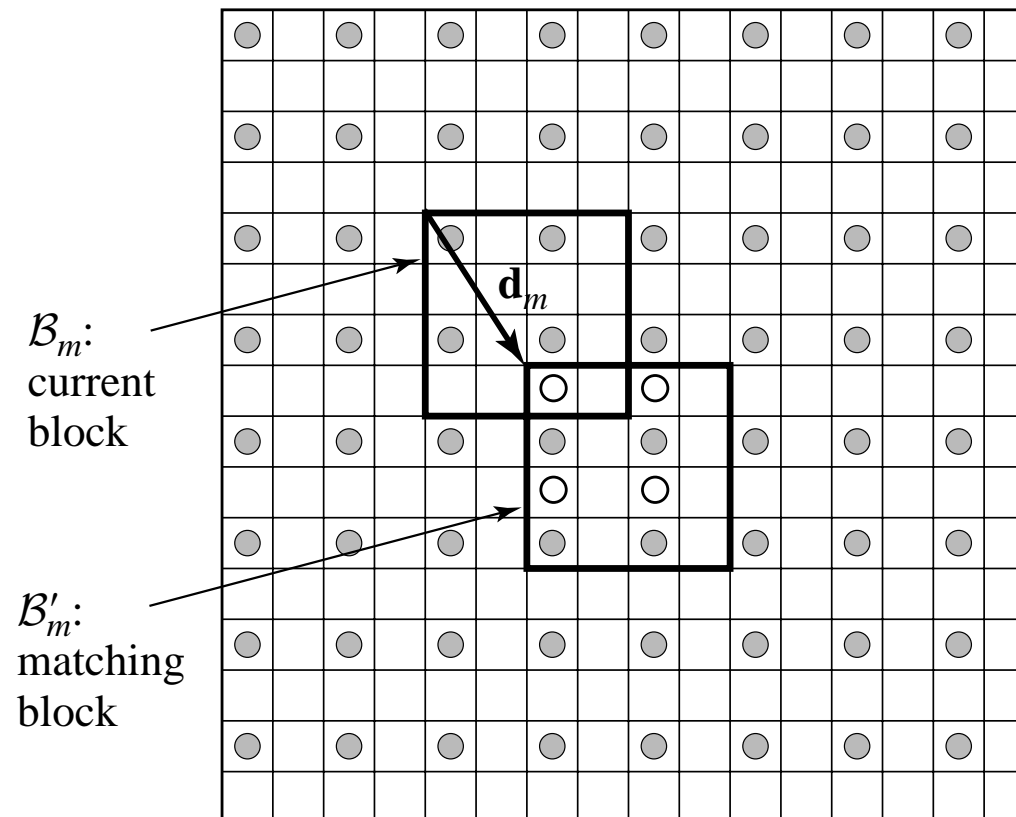


# Fractional Accuracy EBMA

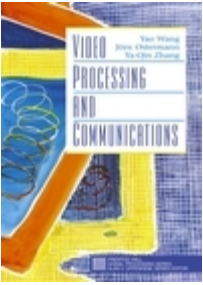
- Real MV may not always be multiples of pixels. To allow sub-pixel MV, the search stepsize must be less than 1 pixel
- **Half-pel EBMA:** stepsize=1/2 pixel in both dimension
- Difficulty:
  - Target frame only have integer pels
- Solution:
  - Interpolate the target frame by factor of two before searching
  - 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.



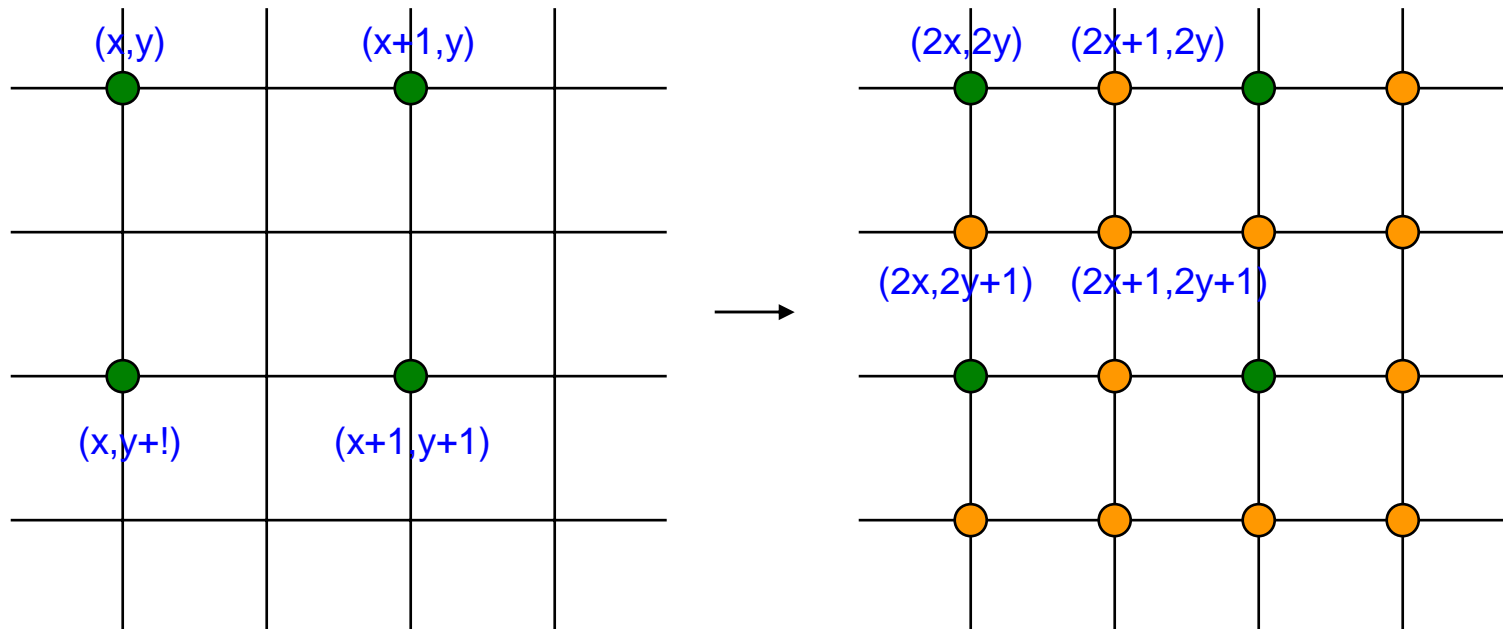
# Half-Pel Accuracy EBMA







# Bilinear Interpolation



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

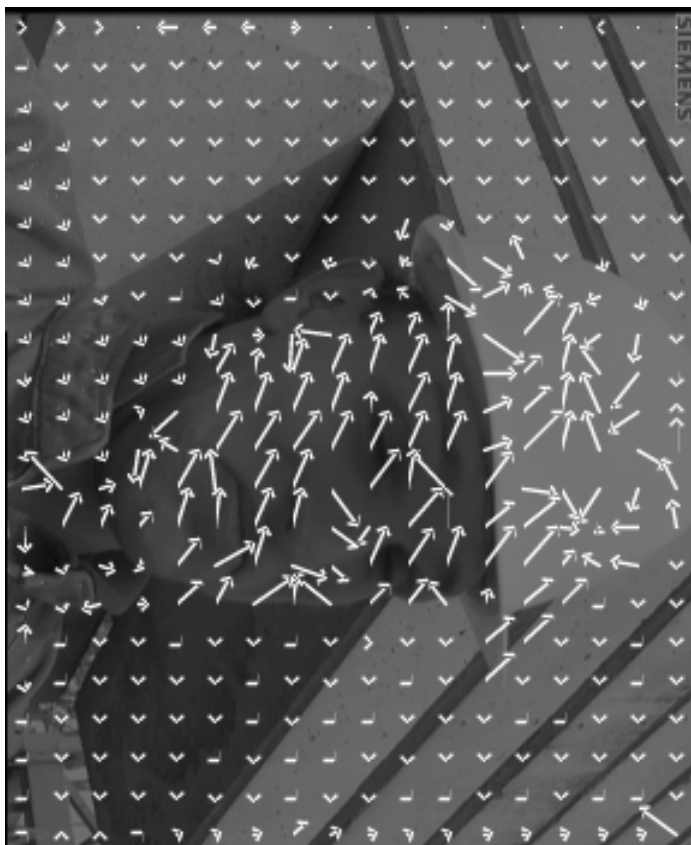
$$O[2x+1,2y]=(I[x,y]+I[x+1,y])/2$$

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

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

Motion field

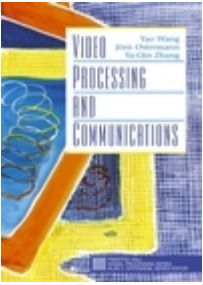
target frame



Predicted anchor frame (29.86dB)

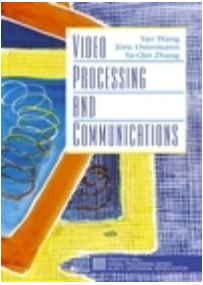
anchor frame

Example: Half-pel EBMA



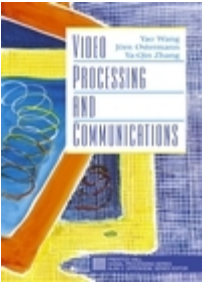
# Pros and Cons with EBMA

- Blocking effect (discontinuity across block boundary) in the predicted image
  - Because the block-wise translation model is not accurate
  - Fix: Deformable BMA (next lecture)
- Motion field somewhat chaotic
  - because MVs are estimated independently from block to block
  - Fix 1: Mesh-based motion estimation (next lecture)
  - 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



# Fast Algorithms for BMA

- Key idea to reduce the computation in EBMA:
  - Reduce # of search candidates:
    - Only search for those that are likely to produce small errors.
    - Predict possible remaining candidates, based on previous search result
  - Simplify the error measure (DFD) to reduce the computation involved for each candidate
- Classical fast algorithms
  - Three-step
  - 2D-log
  - Conjugate direction
- Many new fast algorithms have been developed since then
  - Some suitable for software implementation, others for VLSI implementation (memory access, etc)

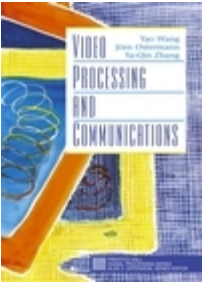


# VcDemo Example

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

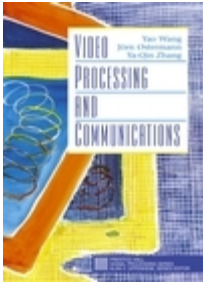
<http://www-ict.its.tudelft.nl/~inald/vcdemo/>

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



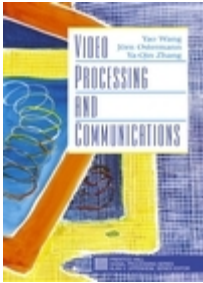
# Summary (I)

- 3D Motion
  - Rigid vs. non-rigid motion
- Camera model: 3D  $\rightarrow$  2D projection
  - Perspective projection vs. orthographic projection
- What causes 2D motion?
  - Object motion projected to 2D
  - Camera motion
- Models corresponding to typical camera motion and object motion
  - Piece-wise projective mapping is a good model for projected rigid object motion
  - Can be approximated by affine or bilinear functions
  - Affine functions can also characterize some global camera motions
- Constraints for 2D motion
  - Optical flow equation
  - Derived from **constant intensity** and **small motion** assumption
  - Ambiguity in motion estimation



## Summary (II)

- How to represent motion:
  - Pixel-based, block-based, region-based, global, etc.
- Estimation criterion:
  - DFD (constant intensity)
  - OF (constant intensity+small motion)
  - Bayesian (MAP, DFD+motion smoothness)
- Search method:
  - Exhaustive search, gradient-descent, multi-resolution (next lecture)
- Pixel-based motion estimation
  - Most accurate representation, but also most costly to estimate
- Block-based motion estimation
  - Good trade-off between accuracy and speed
  - EBMA and its fast but suboptimal variant is widely used in video coding for motion-compensated temporal prediction.



# Homework

- Reading assignment:
  - Chap 5: Sec. 5.1, 5.5
  - Chap 6: Sec. 6.1-6.4, Sec. 6.4.5, 6.4.6 not required, Apx. A, B.
- Written assignment
  - Prob. 5.3, 5.4, 5.6
  - Correction:
    - Prob. 5.3: Show that the projected 2-D motion of a 3-D object undergoing rigid motion can be described by Eq.(5.5.13)
    - 5.4: change  $aX+bY+cZ=1$  to  $Z=aX+bY+c$
  - Prob. 6.4, 6.5, 6.6
- Computer assignment
  - Prob. 6.12, 6.13
  - Note: you can download sample video frames from the course webpage. When applying your motion estimation algorithm, you should choose two frames that have sufficient motion in between so that it is easy to observe effect of motion estimation inaccuracy. If necessary, choose two frames that are several frames apart. For example, foreman: frame 100 and frame 103.