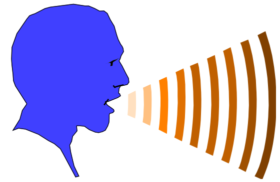


Image and Video Processing

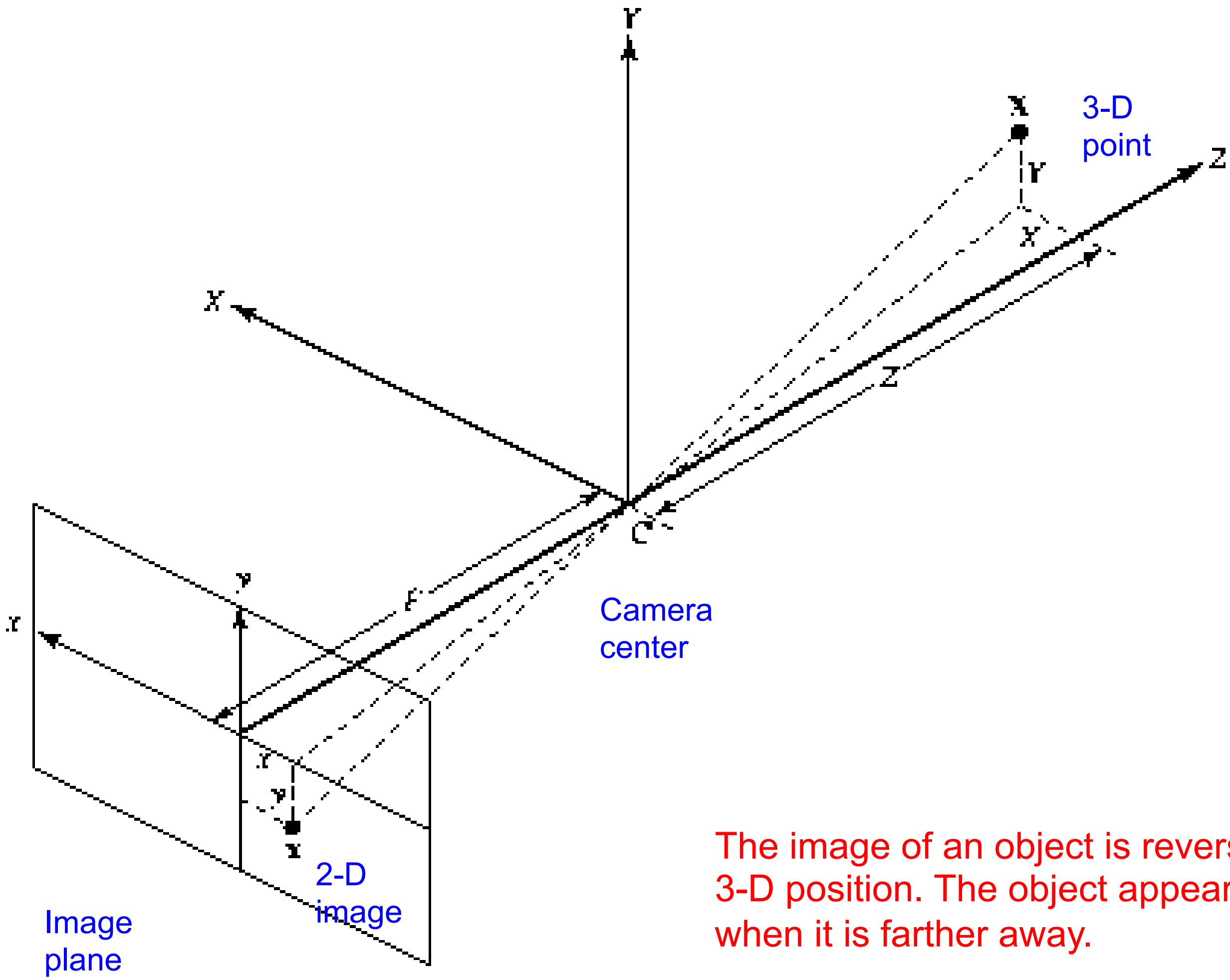
Motion Estimation

Yao Wang
Tandon School of Engineering, New York University

Outline

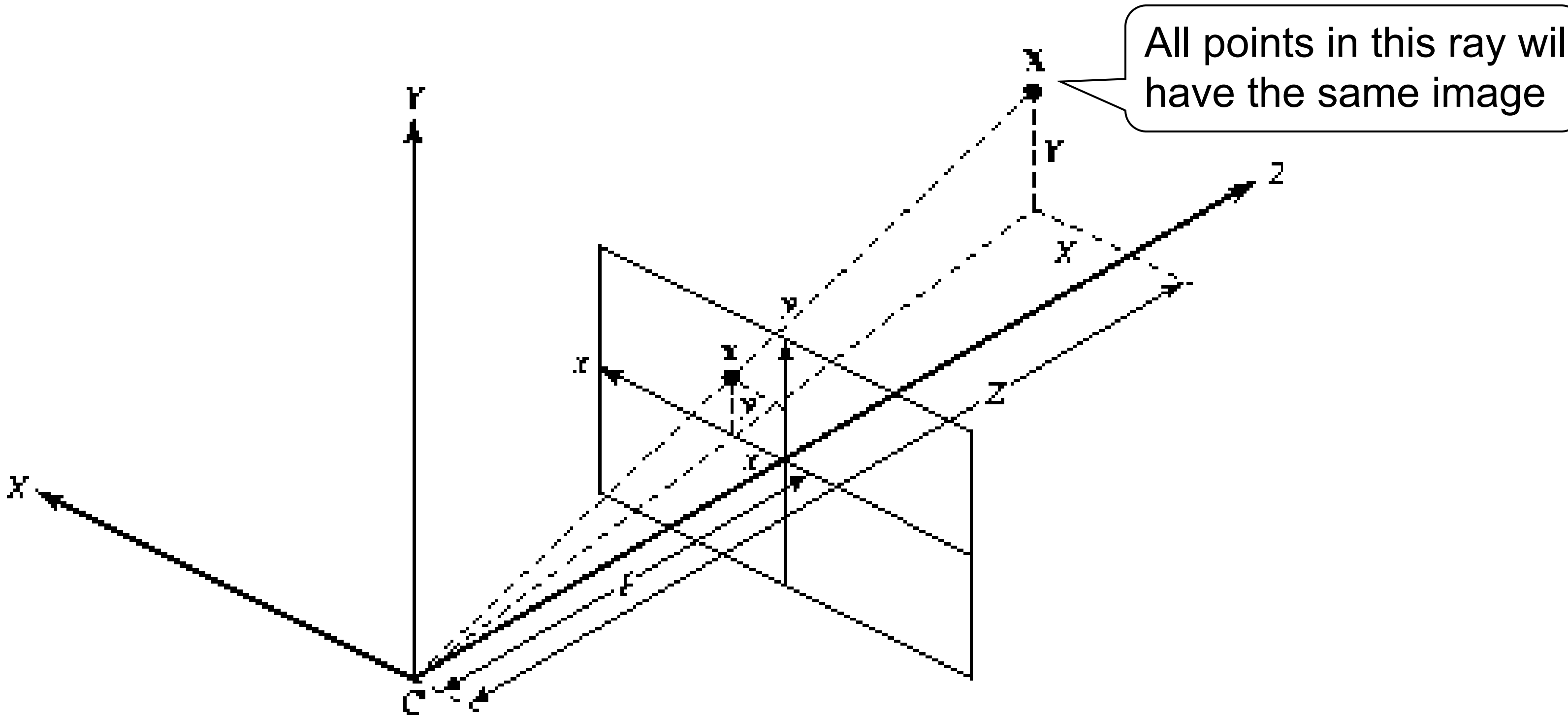
- 3D motion model
- 2-D motion model
- 2-D motion vs. optical flow
- Optical flow equation and ambiguity in motion estimation
- General methodologies in motion estimation
  – Motion representation
  – Motion estimation criterion
  – Optimization methods
  – Gradient descent methods
- Pixel-based motion estimation
- Block-based motion estimation assuming constant motion in each block
  – EBMA algorithm revisited
  – Half-pel EBMA
  – Hierarchical EBMA (HBMA)
- Deformable block matching (DBMA)
- Mesh-based motion estimation

Pinhole Camera Model



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

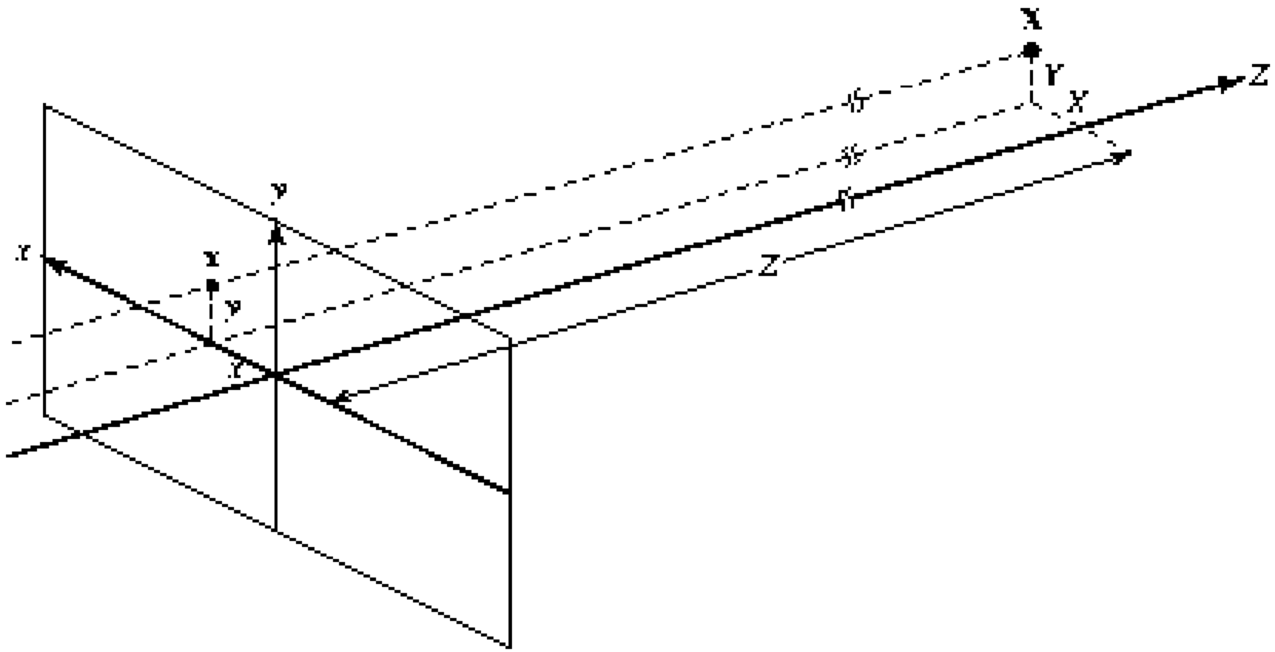


(a)

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

x, y are inversely related to Z

Approximate Model: Orthographic Projection



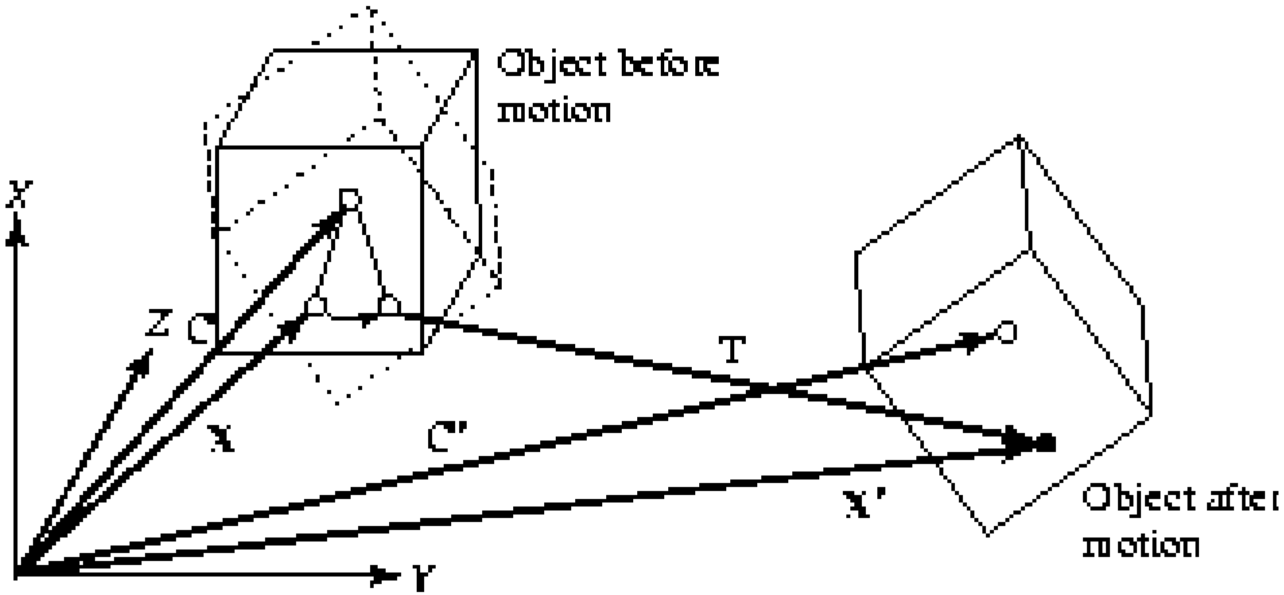
(b)

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:

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

Rotation Matrix

$$[\mathbf{R}] = [\mathbf{R}_z] \cdot [\mathbf{R}_y] \cdot [\mathbf{R}_x]$$

$$[\mathbf{R}_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \quad [\mathbf{R}_y] = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \quad [\mathbf{R}_z] = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[\mathbf{R}] = \begin{bmatrix} \cos\theta_y \cos\theta_z & \sin\theta_x \sin\theta_y \cos\theta_z - \cos\theta_x \sin\theta_z & \cos\theta_x \sin\theta_y \cos\theta_z + \sin\theta_x \sin\theta_z \\ \cos\theta_y \sin\theta_z & \sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z & \cos\theta_x \sin\theta_y \sin\theta_z - \sin\theta_x \cos\theta_z \\ -\sin\theta_y & \sin\theta_x \cos\theta_y & \cos\theta_x \cos\theta_y \end{bmatrix}$$

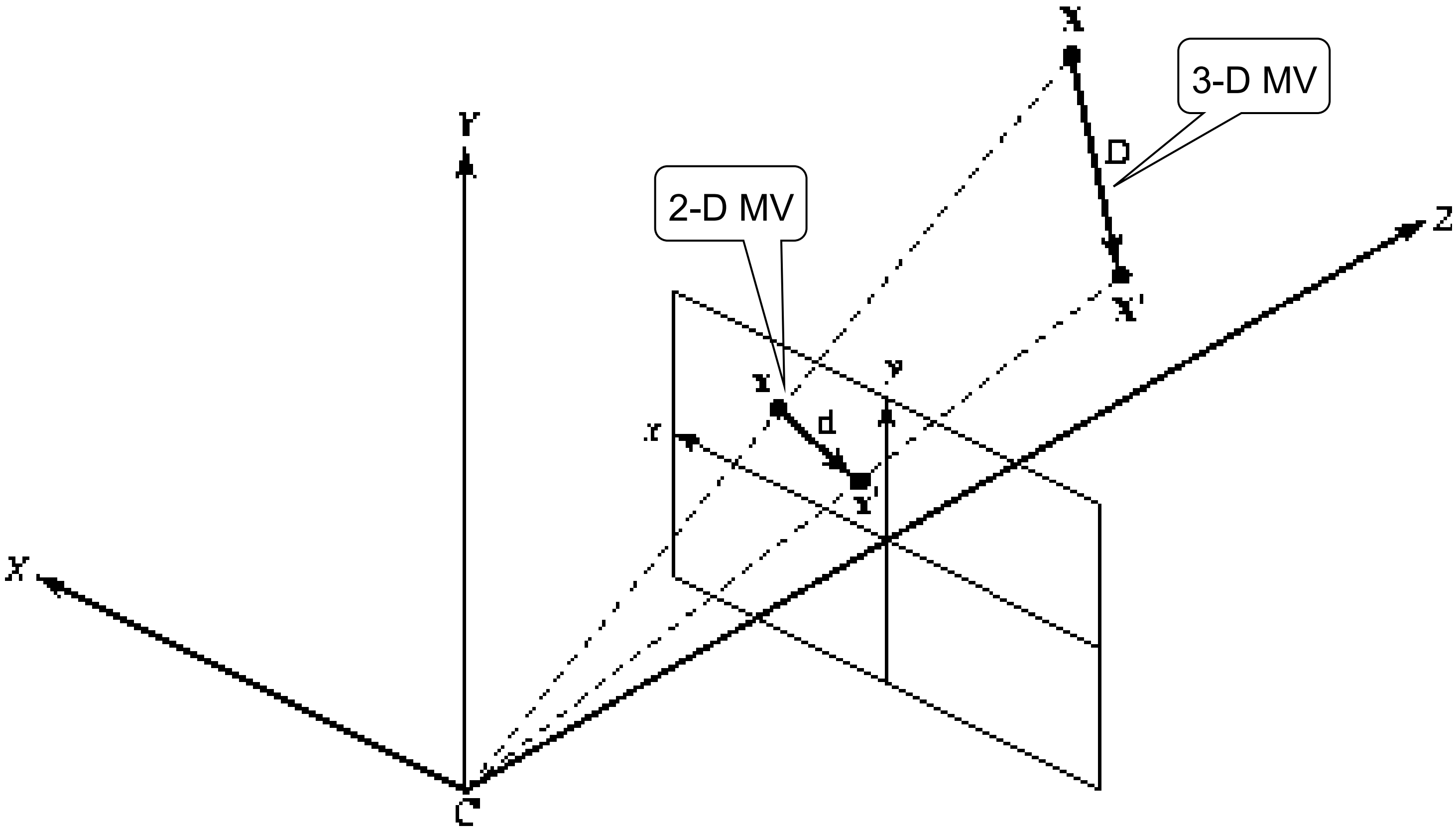
When all rotation angles are small:

$$[\mathbf{R}] \approx [\mathbf{R}'] = \begin{bmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{bmatrix}$$

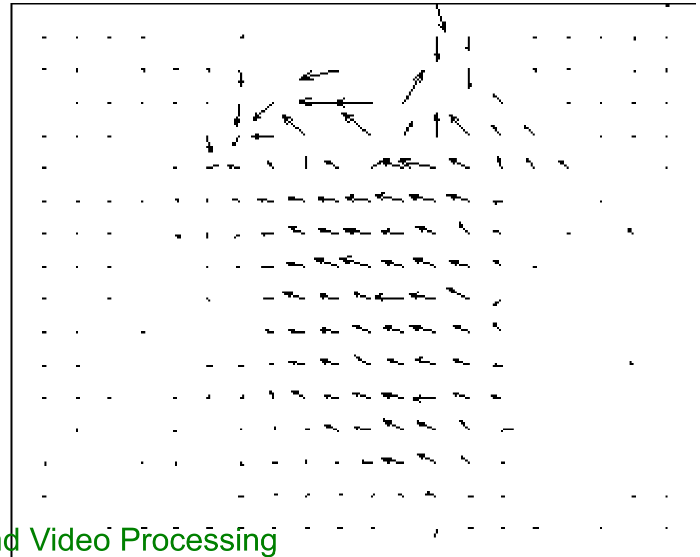
Flexible Object Motion

- 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

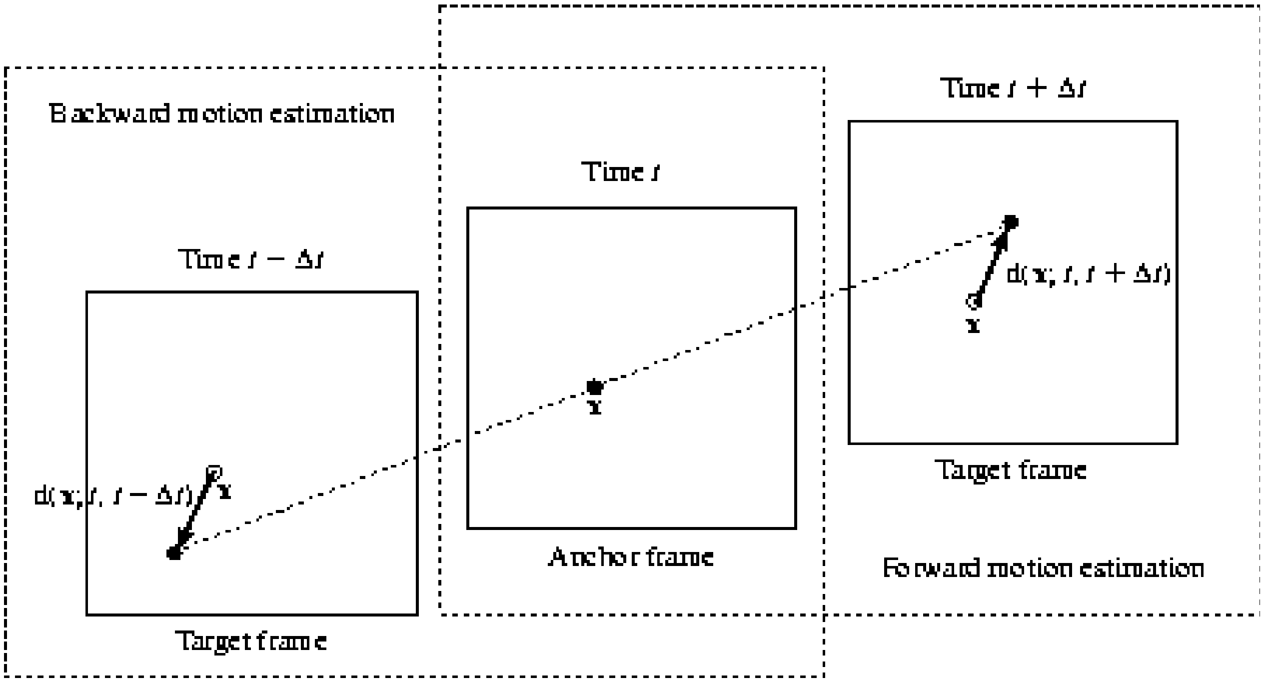


Sample 2D Motion Field



At each pixel (or center of a block) of the anchor image (right), the motion vector describes the 2D displacement between this pixel and its corresponding pixel in the other target image (left)

Motion Field Definition



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

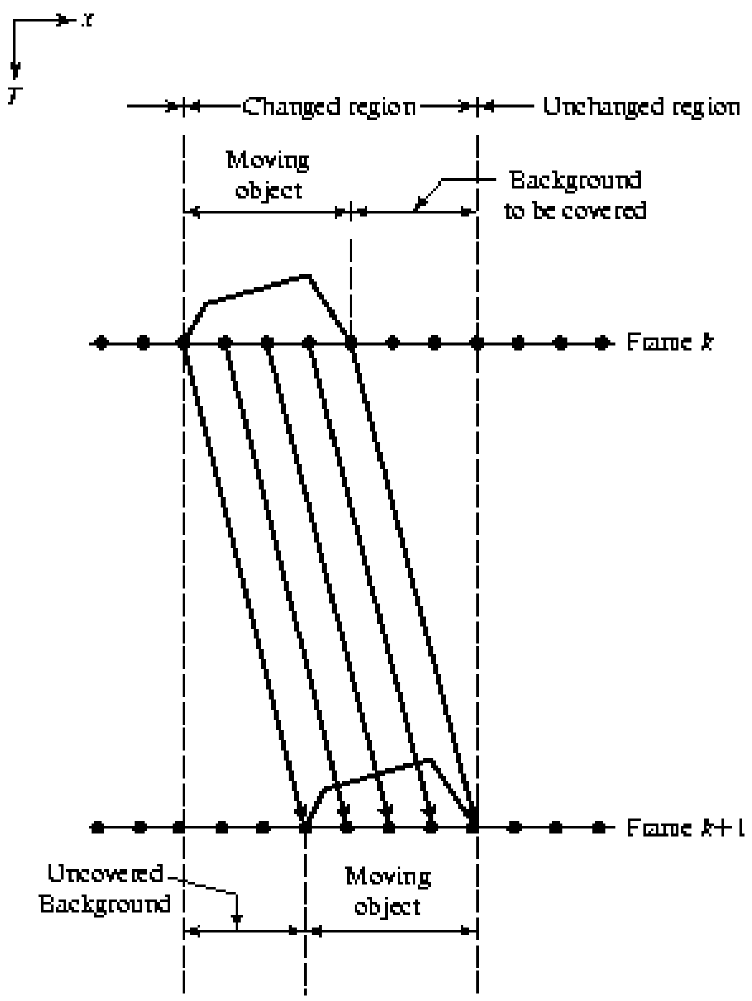
Mapping function:

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

Occlusion Effect

- Motion is undefined in occluded regions
  - uncovered region
  - Covered region

- Ideally a 2D motion field should indicate such area as uncovered (or occluded) instead of giving false MVs



2-D Motion Corresponding to Rigid Object Motion

- General case:

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

$$\xrightarrow[\text{Perspective Projection}]{}$$

$$x' = F \frac{(r_1 x + r_2 y + r_3 F) Z + T_x F}{(r_7 x + r_8 y + r_9 F) Z + T_z F}$$

$$y' = F \frac{(r_4 x + r_5 y + r_6 F) Z + T_y F}{(r_7 x + r_8 y + r_9 F) Z + T_z F}$$

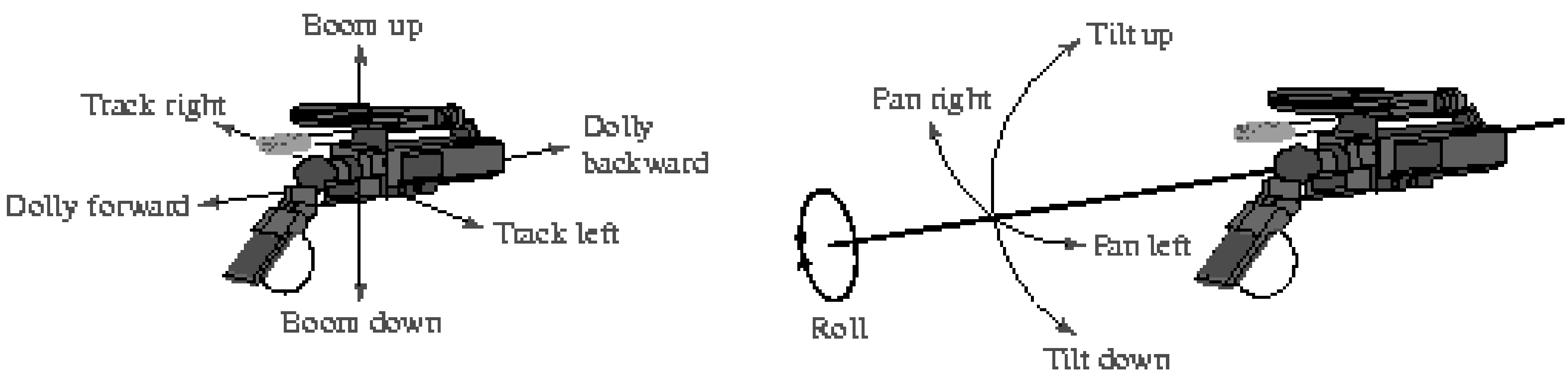
- Projective mapping:

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

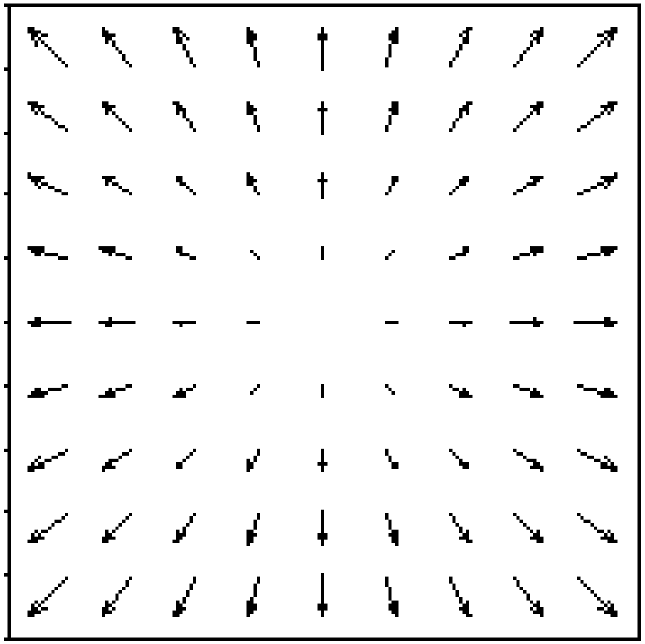
$$x' = \frac{a_0 + a_1 x + a_2 y}{1 + c_1 x + c_2 y}, \quad y' = \frac{b_0 + b_1 x + b_2 y}{1 + c_1 x + c_2 y}$$

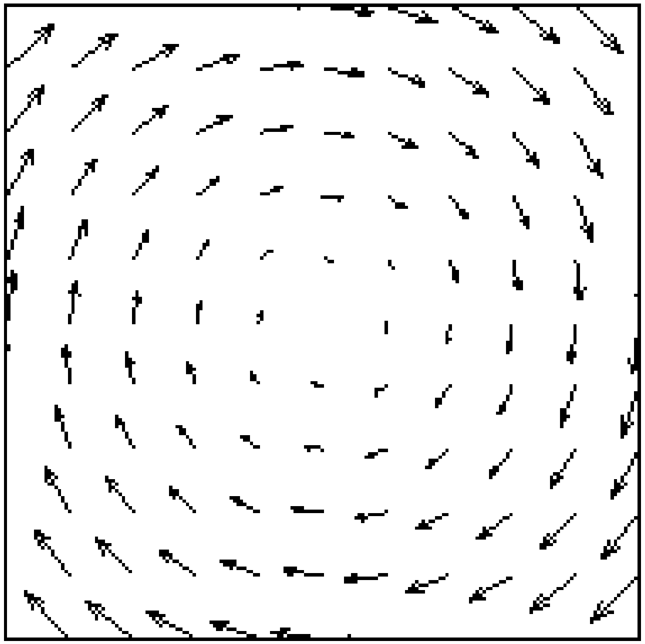
- Real object surfaces are not planar! But can be divided into small patches each approximated as planar
  - 2D motion can be modeled by piecewise projective mapping (a different projective mapping over each 2D patch)

Typical Camera Motions



2-D Motion Corresponding to Camera Motion





Camera zoom

Camera rotation around Z-axis (roll)

2-D Motion Corresponding to Camera Motion or Rigid Object Motion

- General case:

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

$$\xrightarrow[\text{Perspective Projection}]{}$$

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

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

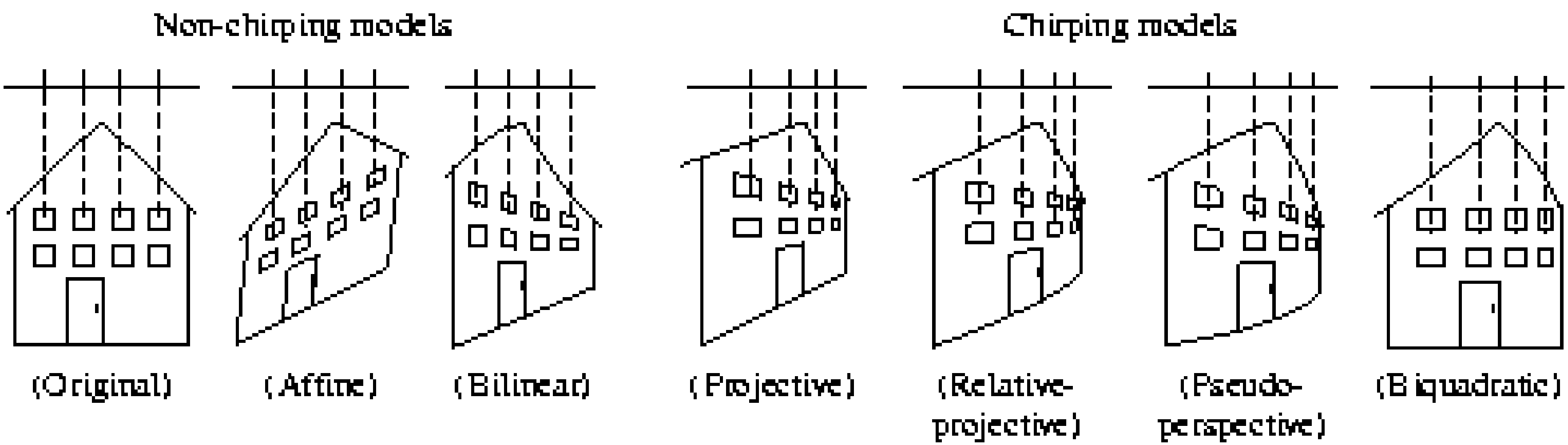
- Projective mapping:

When all the object points are far from the camera and hence can be considered on the same plane ($Z = c$):

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

The above is also true if the imaged object has a planar surface (i.e. Z=aX+bY+c) (HW!)

Projective Mapping and Its Approximations



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):

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 x + a_2 y \\ b_0 + b_1 x + b_2 y \end{bmatrix}$$

  – Good for mapping triangles to triangles

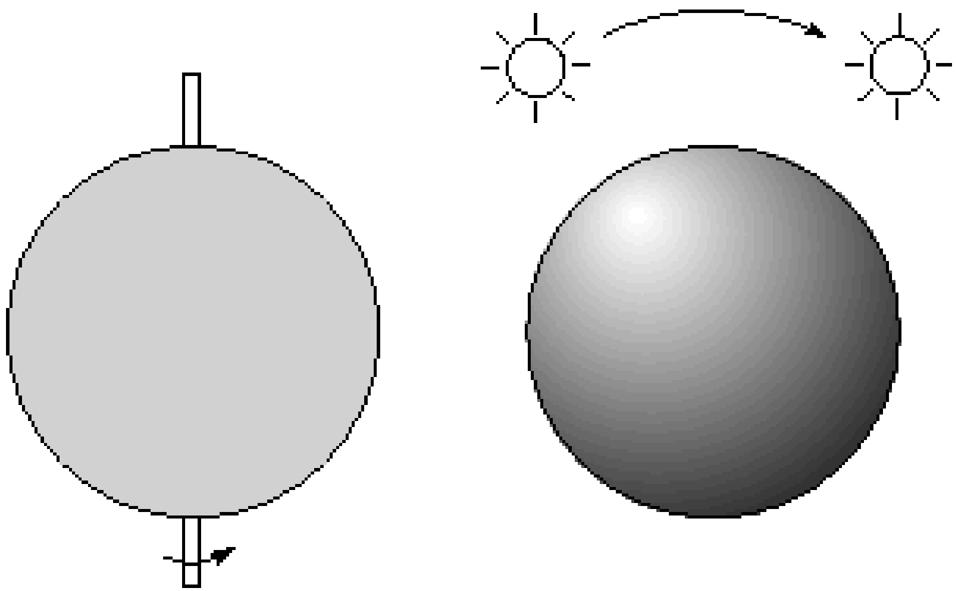
- Bilinear (8 parameters):

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 x + a_2 y + a_3 xy \\ b_0 + b_1 x + b_2 y + b_3 xy \end{bmatrix}$$

  – Good for mapping blocks to quadrangles

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:

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

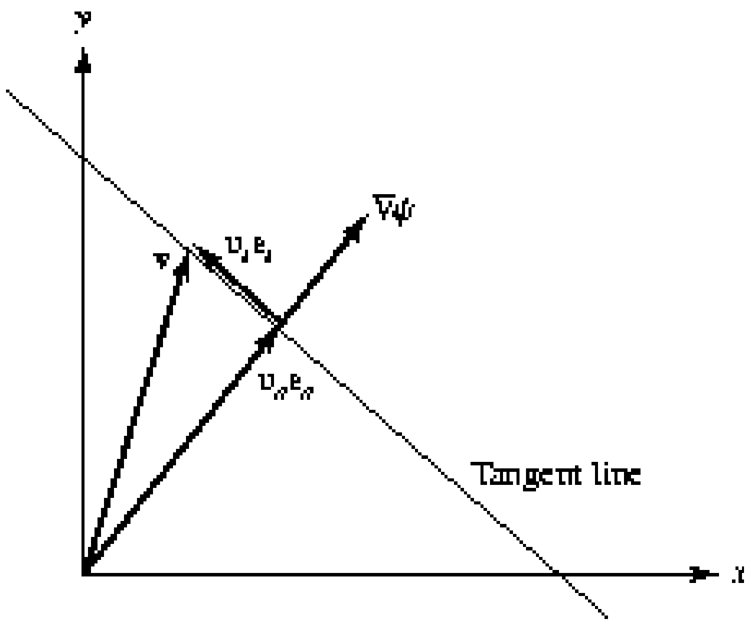
In discrete sample domain (assuming (x,y) in ψ_1 is moved to (x+dx,y+dy) in ψ_2:

$$\frac{\partial\psi_2}{\partial x} d_x + \frac{\partial\psi_2}{\partial y} d_y + \psi_2(x,y) - \psi_1(x,y) = 0$$

Note: Typo in the textbook, Eq. (6.2.3). Gradient should be wrt ψ_2

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



$$\mathbf{v} = v_n \mathbf{e}_n + v_t \mathbf{e}_t$$

$$v_n \|\nabla \psi\| + \frac{\partial \psi}{\partial t} = 0$$

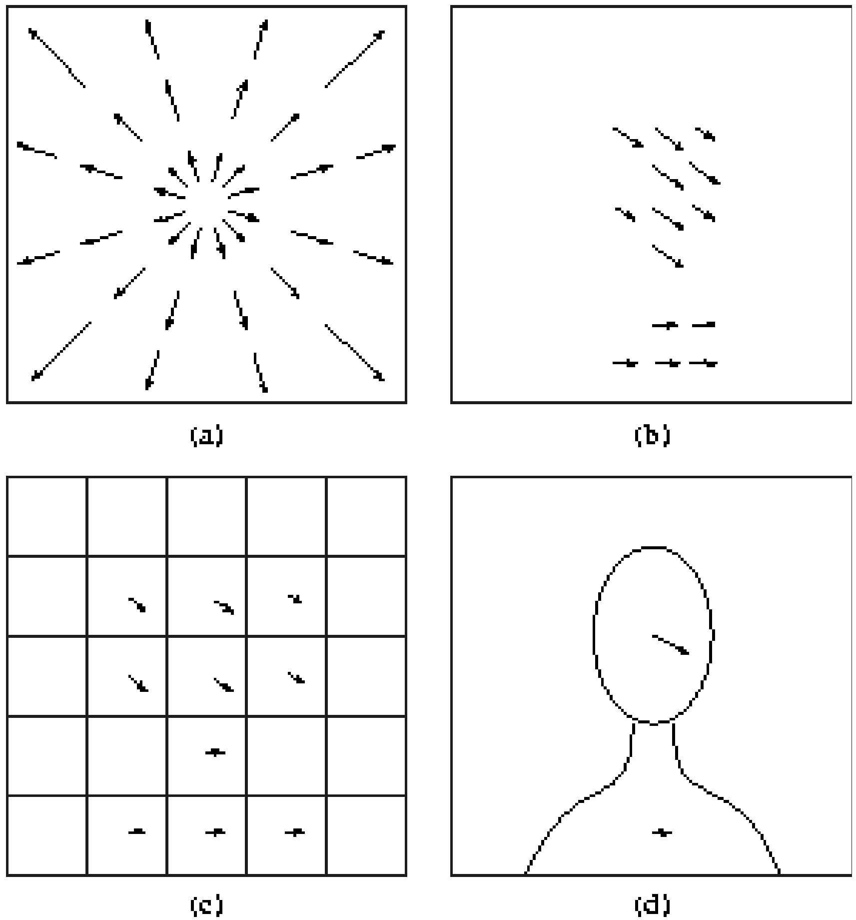
General Considerations for Motion Estimation

- Two categories of approaches:
  - Feature based: finding corresponding features in two different images and then derive the entire motion field based on the motion vectors at corresponding features.
    - more often used in object tracking, 3D reconstruction from 2D
  - Intensity based: directly finding MV at every pixel of block based on constant intensity assumption
    - more often used for motion compensated prediction and filtering, 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

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



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

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.

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

Motion Estimation Criterion

- To minimize the displaced frame difference (DFD) (based on constant intensity assumption)

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

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

- To satisfy the optical flow equation

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

Note typo in Eq(6.2.3)-(6.2.7). Spatial gradients should be w.r.t ψ_2

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

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

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

- 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 use coarse exhaustive search to find a good initial solution, and use this solution to deform target frame, and then apply OF criterion between original anchor frame and the deformed target frame.
- Bayesian criterion can be reduced to the DFD criterion plus motion smoothness constraint
- More in the textbook

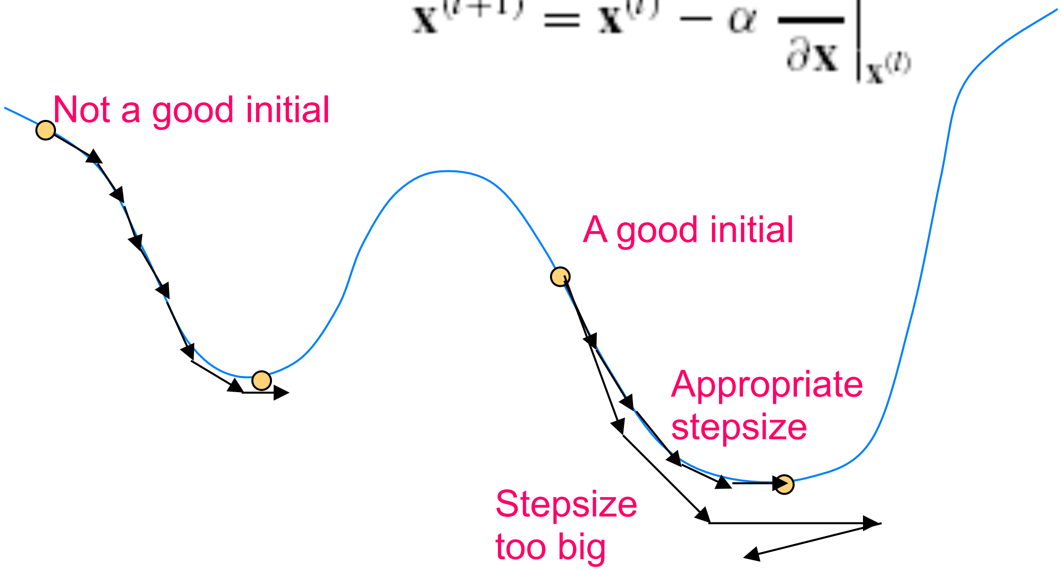
Optimization Methods

- 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 side:
  - Fixed stepsize: Stepsize must be small to avoid oscillation, requires many iterations
  - Steepest gradient descent (adjust stepsize optimally)

Newton's Method

- Newton's method

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

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

  - Converges faster than 1st 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 2nd order than with 1st order)
  - May not converge if \alpha >=1. Should choose \alpha appropriate to reach a good compromise between guaranteeing convergence and the convergence rate.

Newton-Raphson Method

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

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

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

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

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

Pixel-Based Motion Estimation

- Horn-Schunck method
  - DFD + motion smoothness criterion
- Multipoint neighborhood method
  - Assuming every pixel in a small block surrounding a pixel has the same MV
- Pel-recurrsive 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
- Recommended reading for recent advances:
  - Sun, Deqing, Stefan Roth, and Michael J. Black. "Secrets of optical flow estimation and their principles." In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2432-2439. IEEE, 2010.

Block-Based Motion Estimation

- 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.

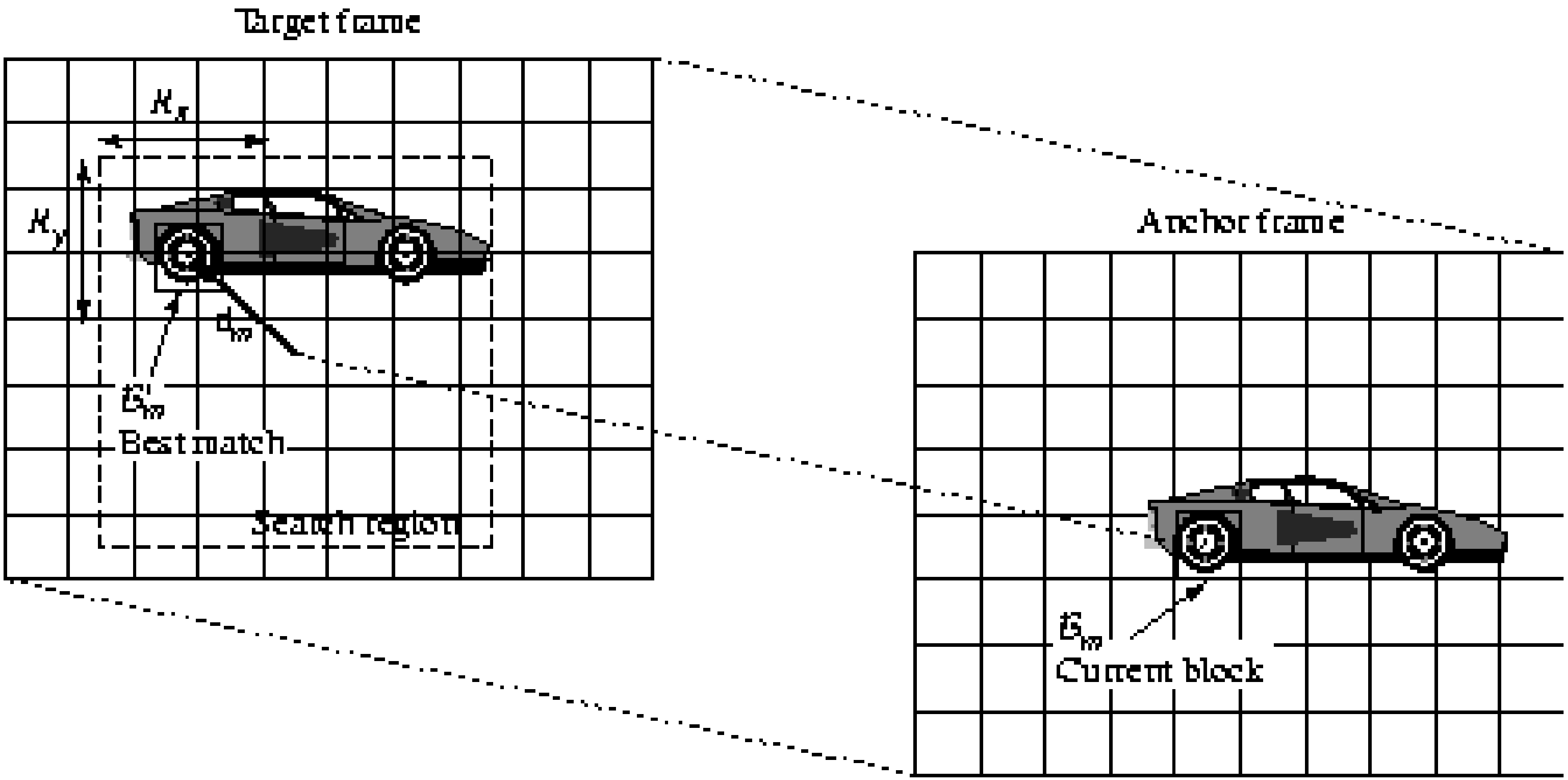
Block Matching Algorithm

- 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:

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

- 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)



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 (needs to be modified so that i+K etc are
within the image domain!)
          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 count in MAD. This program is meant to illustrate the main operations involved. Not the actual working matlab script.

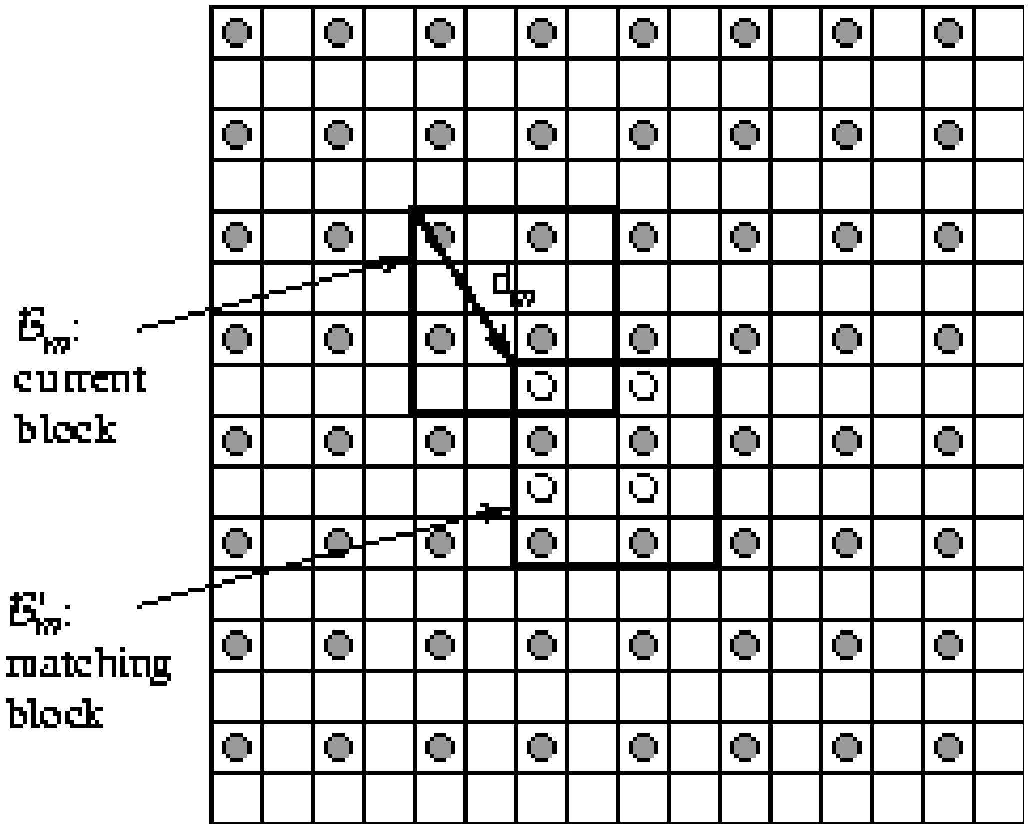
Complexity of Integer-Pel EBMA

- 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 = 2.85×10^8/frame $=8.55\times10^9$/second
- Regular structure suitable for VLSI implementation
- Challenging for software-only implementation

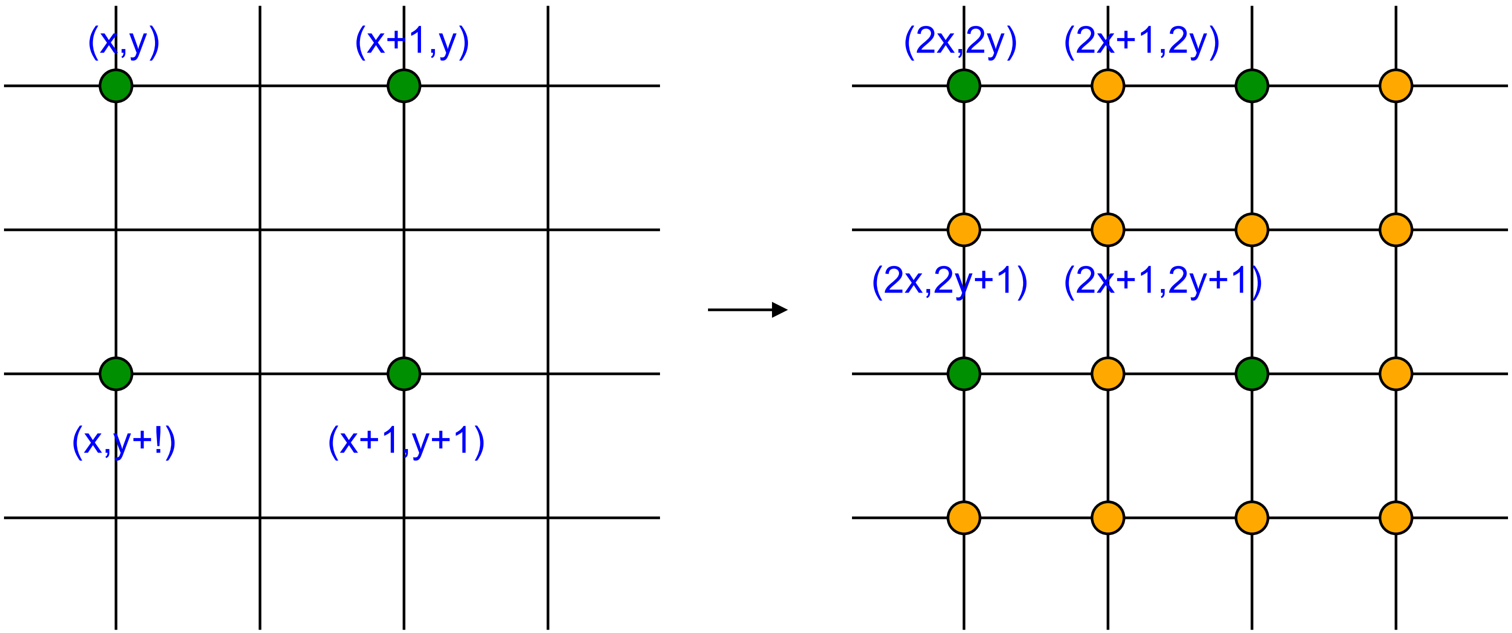
Fractional Accuracy EBMA

- 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
- 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

Implementation for Half-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
%first upsample f2 by a factor of 2 in each direction
f3=imresize(f2, 2,'bilinear') (or use you own implementation!)
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:0.5:R, for l=-R:0.5:R  %for every search candidate (needs to be modified!)
          %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))));
           MAD=sum(sum(abs(f1(i:i+N-1,j:j+N-1)-f3(2*(i+k):2:2*(i+k+N-1),2*(j+l):2:2*(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); wrong! need to use corresponding pixels in
f3!
                %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;
```

Example: Half-pel EBMA

target frame

anchor frame

Motion field

Predicted anchor frame (29.86dB)

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

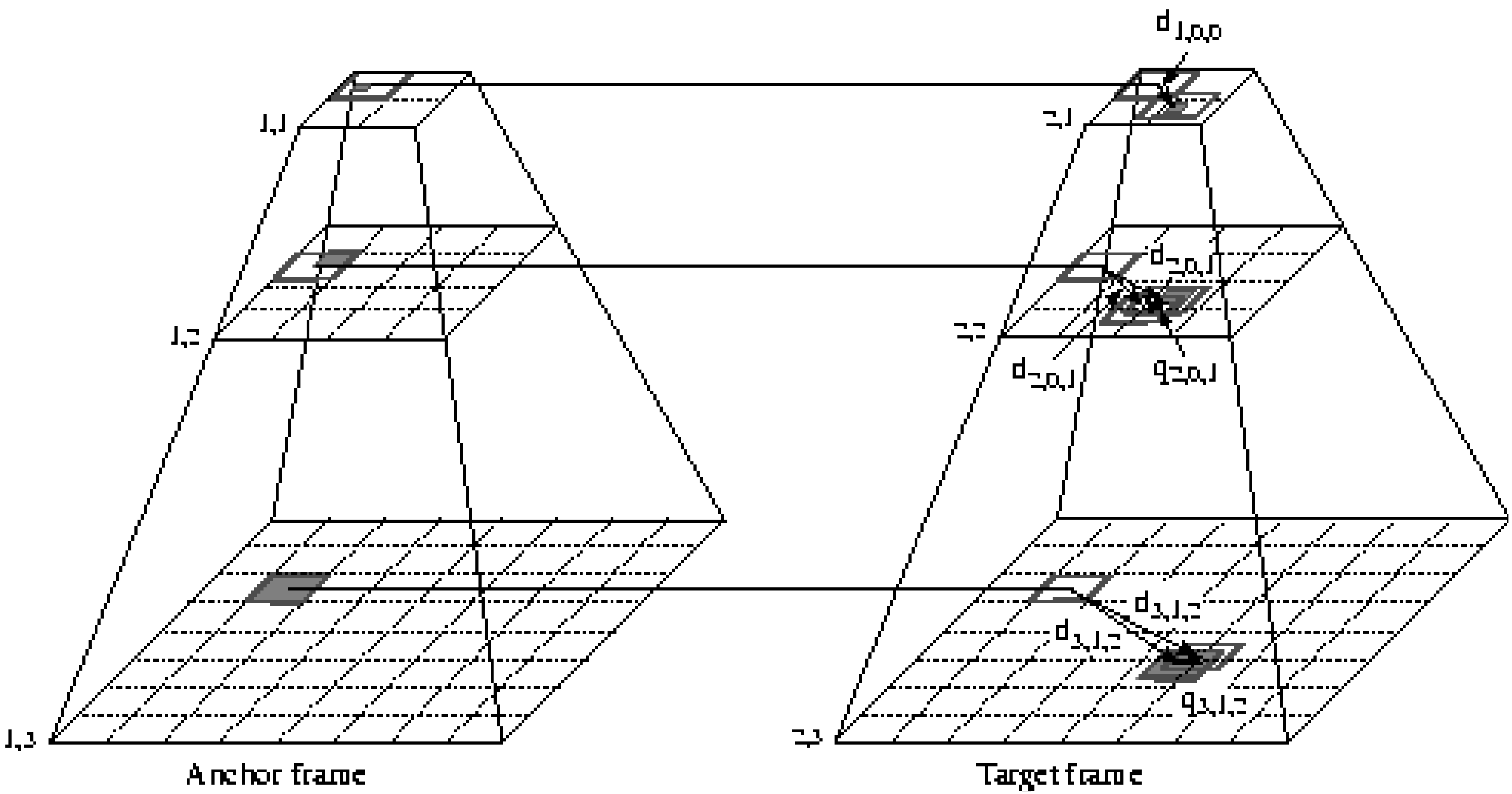
http://insy.ewi.tudelft.nl/content/image-and-video-compression-learning-tool-vcdemo

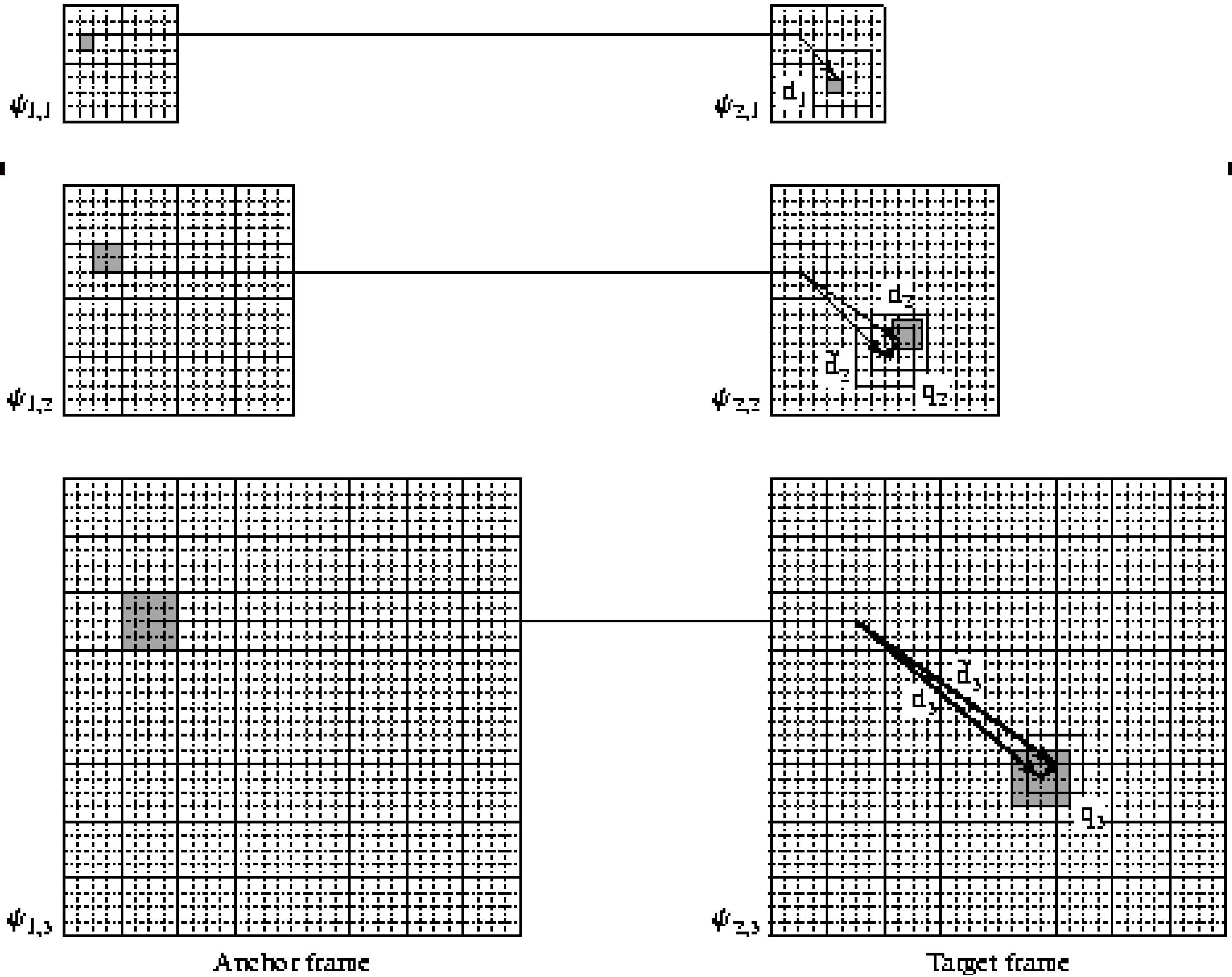
Use the ME tool to show the motion estimation results with different parameter choices

Multi-resolution Motion Estimation

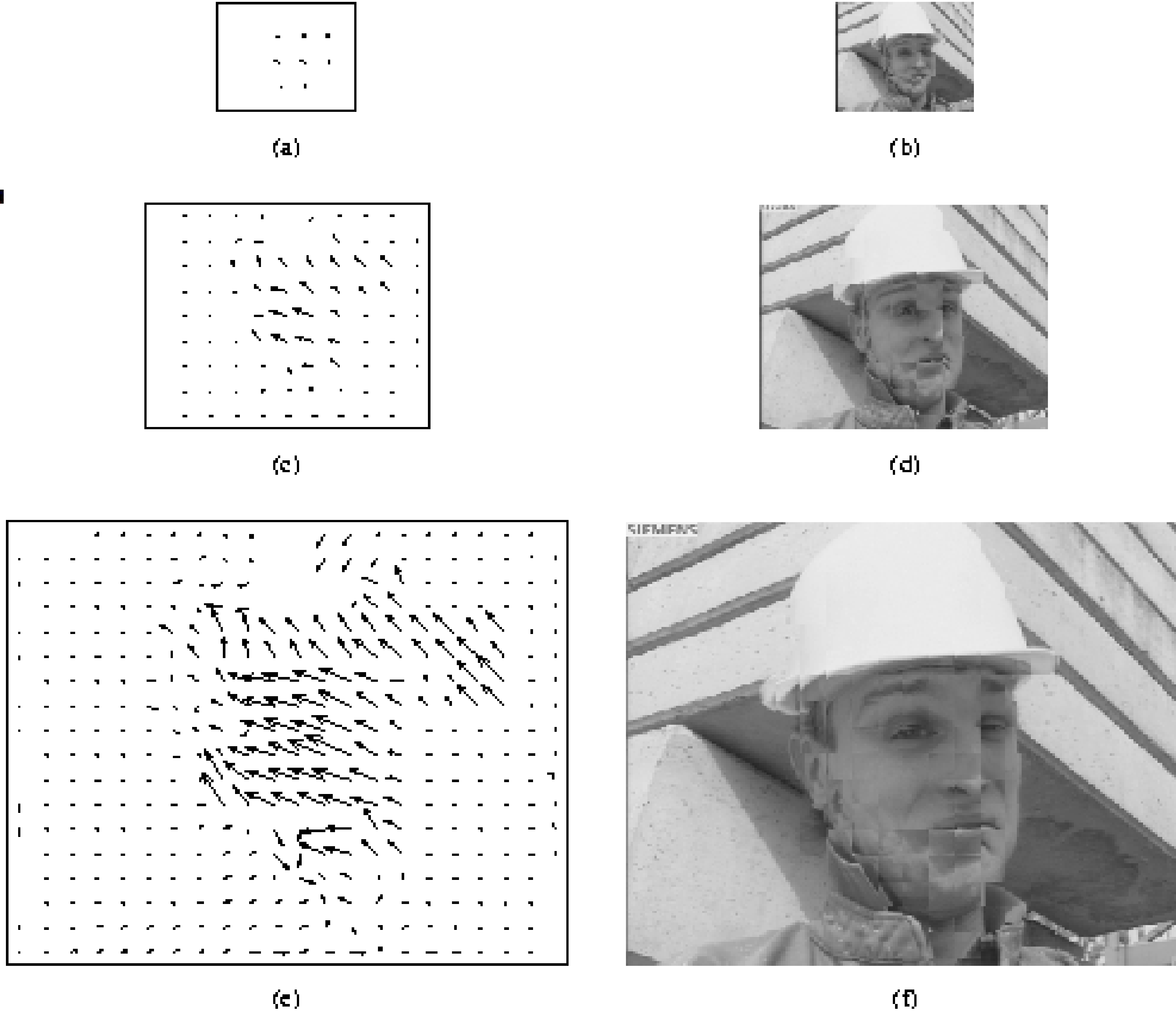
- Problems with BMA
  - Unless exhaustive search is used, the solution may not be global minimum
  - Exhaustive search requires extremely large computation
  - Block wise translation motion model is not always appropriate
- Multiresolution approach
  - Aim to solve the first two problems
  - First estimate the motion in a coarse resolution over low-pass filtered, down-sampled image pair
    - Can usually lead to a solution close to the true motion field
  - Then modify the initial solution in successively finer resolution within a small search range
    - Reduce the computation
  - Can be applied to different motion representations, but we will focus on its application to BMA

Hierarchical Block Matching Algorithm (HBMA)





ψ1,1
ψ2,1
d1
ψ1,2
ψ2,2
d2
q2
ψ1,3
ψ2,3
d3
q3
Anchor frame
Target frame



Predicted anchor frame (29.32dB)

Example: Three-level HBMA

Example: Half-pel EBMA

target frame

anchor frame

Motion field

Predicted anchor frame (29.86dB)

Computation Requirement of HBMA

- Assumption
  - Image size: MxM; Block size: NxN at every level; Levels: L
  - Search range:
    - 1st level: R/2^(L-1) (Equivalent to R in L-th level)
    - Other levels: R/2^(L-1) (can be smaller)
- Operation counts for EBMA
  - image size M, block size N, search range R
  - # operations: $M^2(2R+1)^2$
- Operation counts at l-th level (Image size: M/2^(L-l))

$$\left(M/2^{L-l}\right)^2\left(2R/2^{L-1}+1\right)^2$$

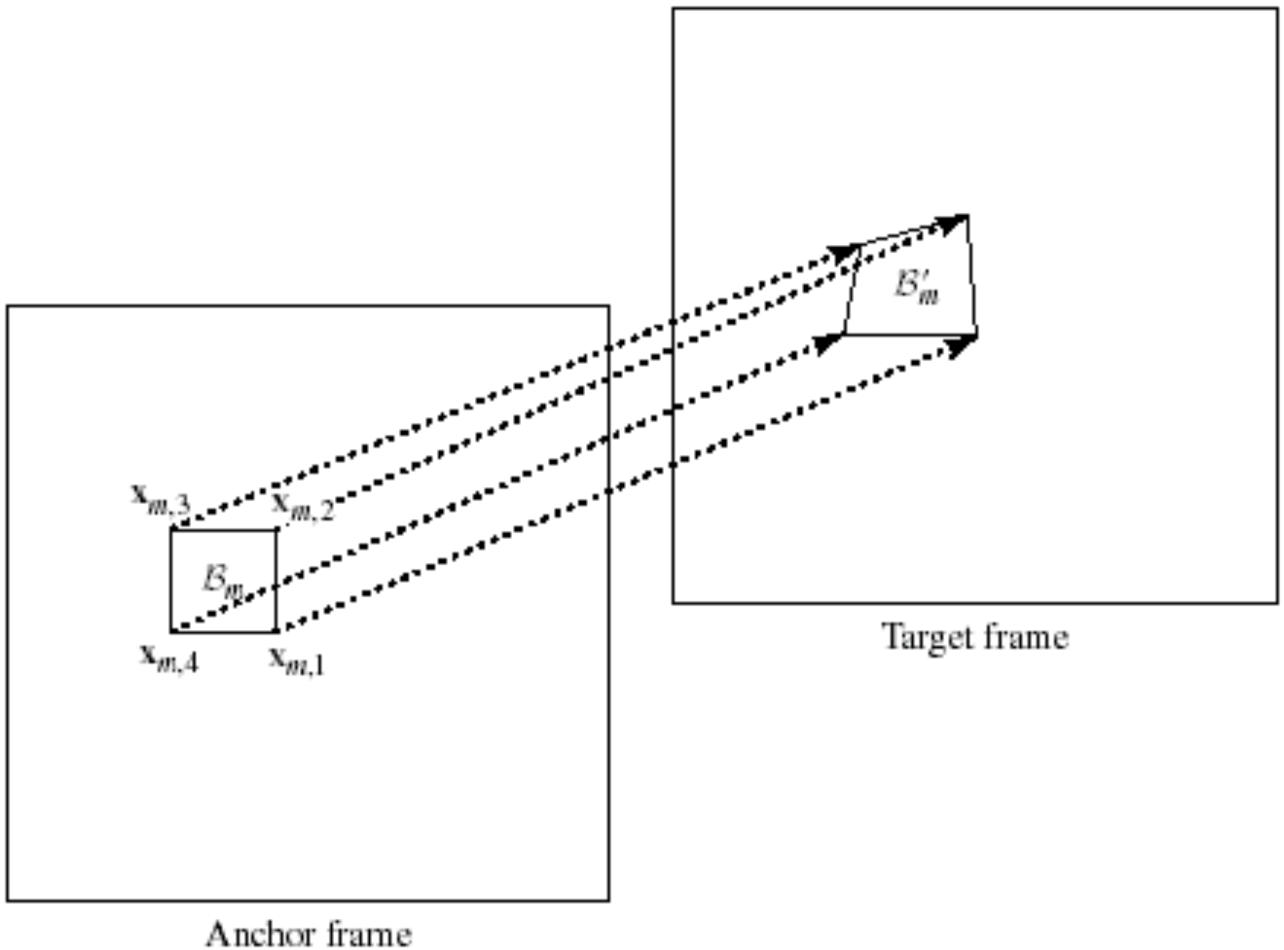
- Total operation count

$$\sum_{l=1}^{L}\left(M/2^{L-l}\right)^2\left(2R/2^{L-1}+1\right)^2 \approx \frac{1}{3}4^{-(L-2)}4M^2R^2$$

- Saving factor:

$$3\cdot 4^{(L-2)}=3(L=2);12(L=3)$$

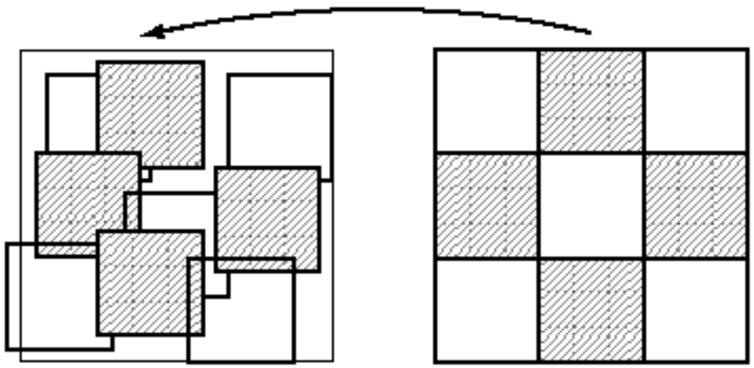
Deformable Block Matching Algorithm



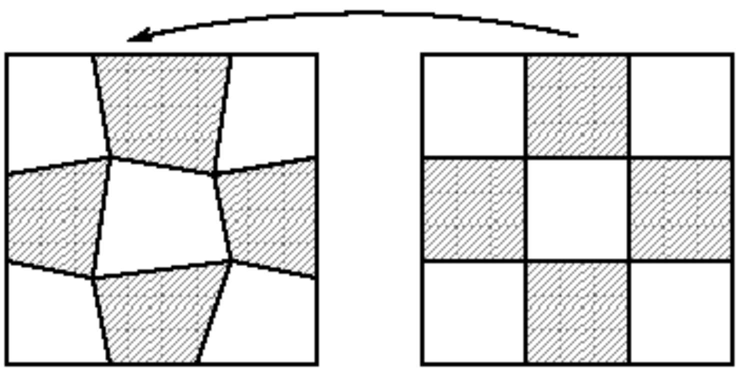
Overview of DBMA

- Partition the anchor frame into regular blocks
- Model the motion in each block by a more complex motion
  - The 2-D motion caused by a flat surface patch undergoing rigid 3-D motion can be approximated well by projective mapping
  - Projective Mapping can be approximated by affine mapping and bilinear mapping
  - Various possible mappings can be described by a node-based motion model
- Estimate the motion parameters block by block independently
  - Discontinuity problem cross block boundaries still remain
- Still cannot solve the problem of multiple motions within a block or changes due to illumination effect!

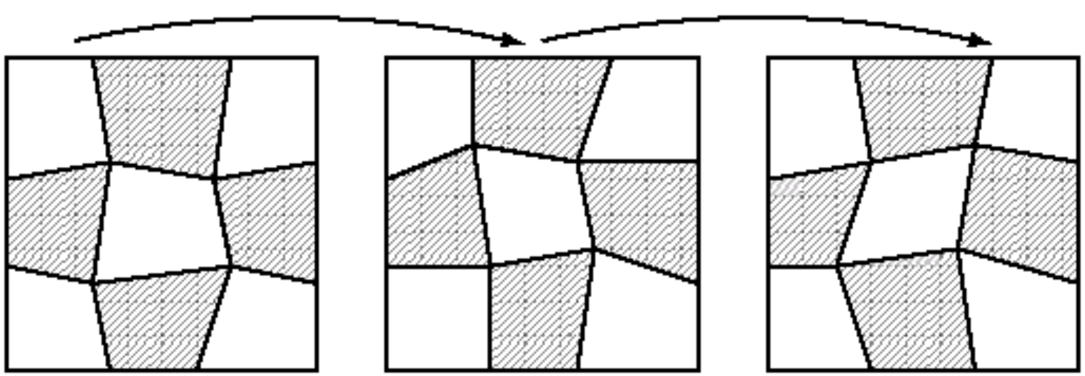
Mesh-based vs. block-based motion estimation



(a) block-based backward ME



(b) mesh-based backward ME



(c) mesh-based forward ME

Summary 1: Motion Models

- 3D Motion
  - Rigid vs. non-rigid motion
- Camera model: 3D -> 2D projection
  - Perspective projection vs. orthographic projection
- What causes 2D motion?
  - Object motion projected to 2D
  - Camera motion
  - Optical flow vs. true 2D motion
- Models corresponding to typical camera motion and object motion
  - Rigid 3D motion of a planar surface -> 2D projective mapping
  - 2D motion of each small patch can be modeled well by projective mapping (Piece-wise projective mapping)
  - Affine or bilinear functions can be used to approximate the projective mapping, but should know the caveats
  - Affine functions are often used to characterize global 2D motion due to camera motions
- Constraints for 2D motion
  - Optical flow equation
  - Derived from constant intensity and small motion assumption
  - Ambiguity in motion estimation

Summary 2: General Strategy for Motion Estimation

- 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

Summary 3: Motion Estimation Methods

- Pixel-based motion estimation (also known as optical flow estimation)
  - Most accurate representation, but also most costly to estimate
- Block-based motion estimation, assuming each block has a constant motion
  - 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.
  - HBMA can not only reduce computation but also yield physically more correct motion estimates
- Deformable block matching algorithm (DBMA)
  - To allow more complex motion within each block
- Mesh-based motion estimation
  - To enforce continuity of motion across block boundaries
- Global motion estimation (next lecture)
- Region-based motion estimation (next lecture)

Reading Assignments

- Reading assignment (Wang, et al, 2004)
  - Chap 5: Sec. 5.1, 5.5
  - Chap 6: Sec. 6.1-6.6, Apx. A, B.

- Optional reading:
  - Woods, 2012, Sec. 11.2.
  - Sun, Deqing, Stefan Roth, and Michael J. Black. "Secrets of optical flow estimation and their principles." In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2432-2439. IEEE, 2010.

Written Assignment

1. Show that the projected 2-D motion of a 3-D object planar patch undergoing rigid motion can be described by projective mapping.
2. Prob. Consider a triangular patch whose original corner positions are at $\mathbf{x}_k$, k=1,2,3. Suppose each corner is moved by $\mathbf{d}_k$, k=1,2,3. The motion field within the triangular patch can be described by an affine mapping. Express the affine parameters in terms of $\mathbf{d}_k$.
3. Prob. 6.5
4. Prob. 6.8
5. Prob. 6.9
6. (Optional) Go through and verify the gradient descent algorithm presented for estimating the nodal motions in DBMA in Eq. (6.5.2)-(6.5.6).
7. (Optional) For estimating the nodal motions in DBMA, instead of minimizing the DFD error, set up the formulation using the OF criterion (assuming nodal motions are small), and find the closed form solution of the nodal motion.

MATLAB Assignment

1. Prob. 6.12 (EBMA with integer accuracy)
2. Prob. 6.13 (EBMA with half-pel accuracy)
3. Prob. 6.15 (HBMA)

- 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.