

# **Lecture 3**

## **Time varying image analysis**

---

- Video Source Model
- 3-D motion
- 2-D motion

# Problems

---

- Visual surveillance
  - stationary camera watches a workspace -find moving objects and alert an operator
  - moving camera navigates a workspace - find moving objects and alert an operator
- Video coding
  - use image motion to perform more efficient coding of image sequences
- Navigation
  - camera moves through the world
    - estimate its trajectory use this to remove unwanted jitter from image sequence: Image stabilization and mosaicking
    - use this to control the movement of a robot through the world
- Structure estimation

# Methods

---

- Motion detection
- Motion modeling
- Motion estimation
- Motion tracking
- Structure estimation
- Segmentation

# Source modeling of a video shot

---

- Frame-to-frame variation in images is due to
  - 3-D *camera* motion
  - *object* motion
    - rigid motion; e.g.
    - deformable motion
  - photometric effects due to scene illumination



# Motion detection (1/2)

## ■ Frame differencing

- subtract, on a pixel by pixel basis, two frames from a motion sequence. High differences between the frames due to either motion or changes in scene

## ■ Problems

- noise in images can give high differences between frames
  - compare neighborhoods rather than individual pixels
- as objects move, their homogeneous interiors don't result in changing image intensities over short time periods
  - motion detected only at boundaries
  - requires subsequent grouping of moving pixels into objects



# Motion detection (2/2)

## ■ Background subtraction

- ~~create an image of the stationary background by averaging a long sequence~~

- for any pixel, most recent frame is the background
- computing the median of a sequence of frames is a high probability assumption
- threshold on difference image
- can also compute a distribution of background pixels by fitting a mixture of Gaussians to set of intensities and assuming large population is the background - adaptive thresholding to find foreground pixels



- difference a frame from the known background frame

- even for interior points of homogeneous objects, likely to detect a difference
- this will also detect objects that are stationary but different from the background
- typical algorithm used in surveillance systems

- Algorithms such as these only work if the camera is stationary and objects are moving against a fixed bg

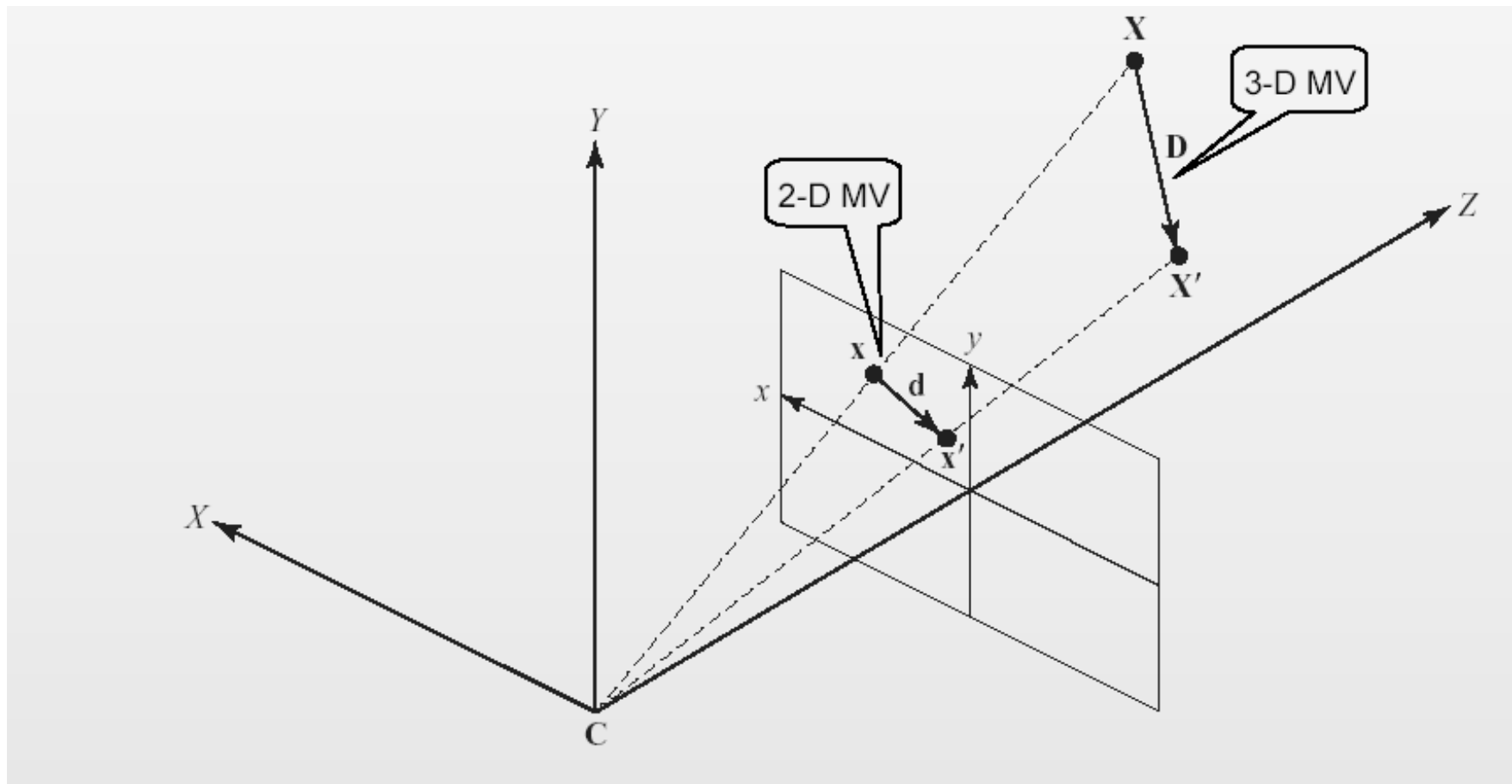
# 3-D and 2-D Motion

---

- 3-D Motion Modeling
  - Cartesian vs. Homogeneous Coordinates
  - Modeling Rigid Motion and Rotation Matrix
  - Modeling Deformable Motion
- Camera Modeling and Image Formation
  - Projective Camera (Perspective Projection)
  - Affine Camera (Weak-Perspective and Orthographic)
- 2-D motion estimation
- 3-D motion estimation
- Structure from motion
- Tracking

# 3D motion vs 2D motion

---





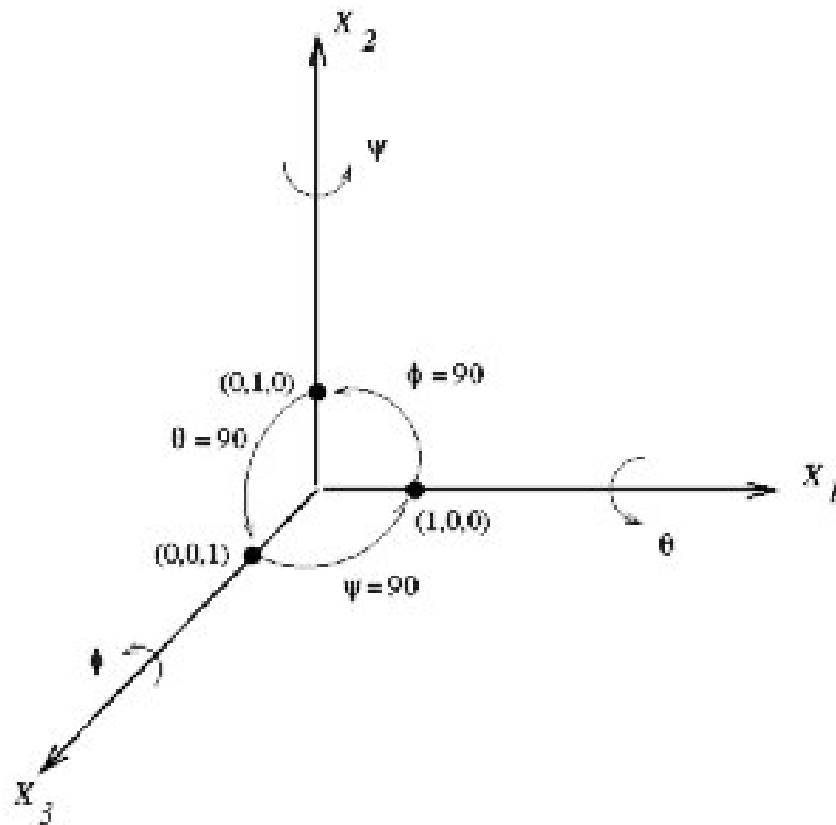
# Rigid Motion - Cartesian coordinates

---

$$\begin{bmatrix} X1' \\ X2' \\ X3' \end{bmatrix} = \begin{bmatrix} r_{11}r_{21}r_{13} \\ r_{21}r_{22}r_{23} \\ r_{31}r_{32}r_{33} \end{bmatrix} \begin{bmatrix} X1 \\ X2 \\ X3 \end{bmatrix} + \begin{bmatrix} T1 \\ T2 \\ T3 \end{bmatrix}$$

- Rotation by Euler angles about coordinate axes
- Rotation by a solid angle about an arbitrary axis

# Rigid Motion - Rotation by Euler angles



The matrices for clockwise rotation

$$\mathbf{R}_\theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

$$\mathbf{R}_\psi = \begin{bmatrix} \cos \psi & 0 & \sin \psi \\ 0 & 1 & 0 \\ -\sin \psi & 0 & \cos \psi \end{bmatrix}$$

$$\mathbf{R}_\phi = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Eulerian angles of rotation,  $\theta$ ,  $\phi$ ,  $\psi$ .

# Rigid Motion - Small angle approximation

---

- For small rotations,  $\cos \Delta\Psi \approx 1$  and  $\sin \Delta\Psi \approx \Delta\Psi$ , etc.,

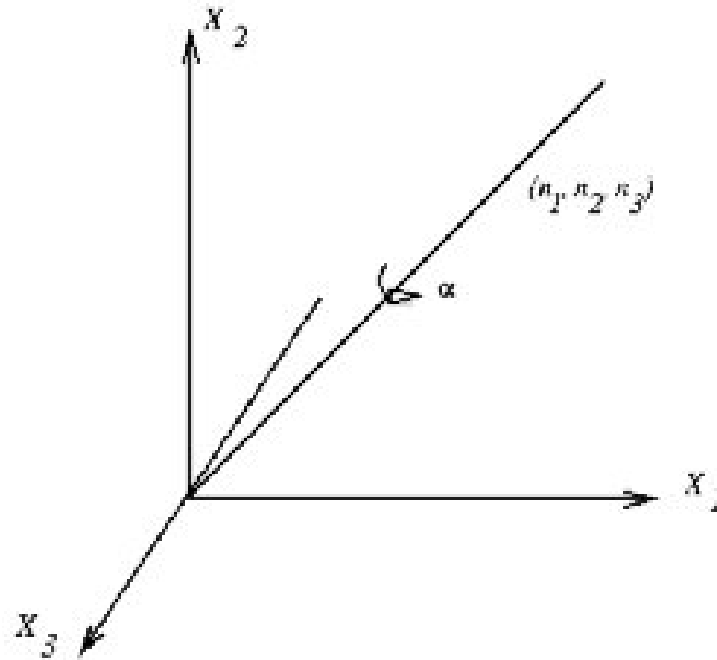
$$\mathbf{R}_\psi = \begin{bmatrix} 1 & 0 & \Delta\psi \\ 0 & 1 & 0 \\ -\Delta\psi & 0 & 1 \end{bmatrix}$$

- Then, the composite rotation matrix  $\mathbf{R}$  is given by:

$$\mathbf{R} = \mathbf{R}_\varphi \mathbf{R}_\theta \mathbf{R}_\psi = \begin{bmatrix} 1 & -\Delta\varphi & \Delta\psi \\ \Delta\varphi & 1 & -\Delta\theta \\ -\Delta\psi & \Delta\theta & 1 \end{bmatrix}$$

# Rigid Motion - Rotation by a solid angle about an arbitrary axis

---



- Rotation by the angle  $\alpha$  about an arbitrary axis  $(n_1 \ n_2 \ n_3)$

$$\mathbf{R} = \begin{bmatrix} 1 & -n_3 \sin \alpha & +n_2 \sin \alpha \\ +n_3 \sin \alpha & 1 & -n_1 \sin \alpha \\ -n_2 \sin \alpha & +n_1 \sin \alpha & 1 \end{bmatrix}$$

$\Delta\alpha$

## 3-D Velocity model

---

$$\begin{bmatrix} X_1' \\ X_2' \\ X_3' \end{bmatrix} = \begin{bmatrix} 1 & -\Delta\varphi & \Delta\psi \\ \Delta\varphi & 1 & -\Delta\theta \\ -\Delta\psi & \Delta\theta & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

$$\begin{bmatrix} X_1' - X_1 \\ X_2' - X_2 \\ X_3' - X_3 \end{bmatrix} = \begin{bmatrix} 0 & -\Delta\varphi & \Delta\psi \\ \Delta\varphi & 0 & -\Delta\theta \\ -\Delta\psi & \Delta\theta & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

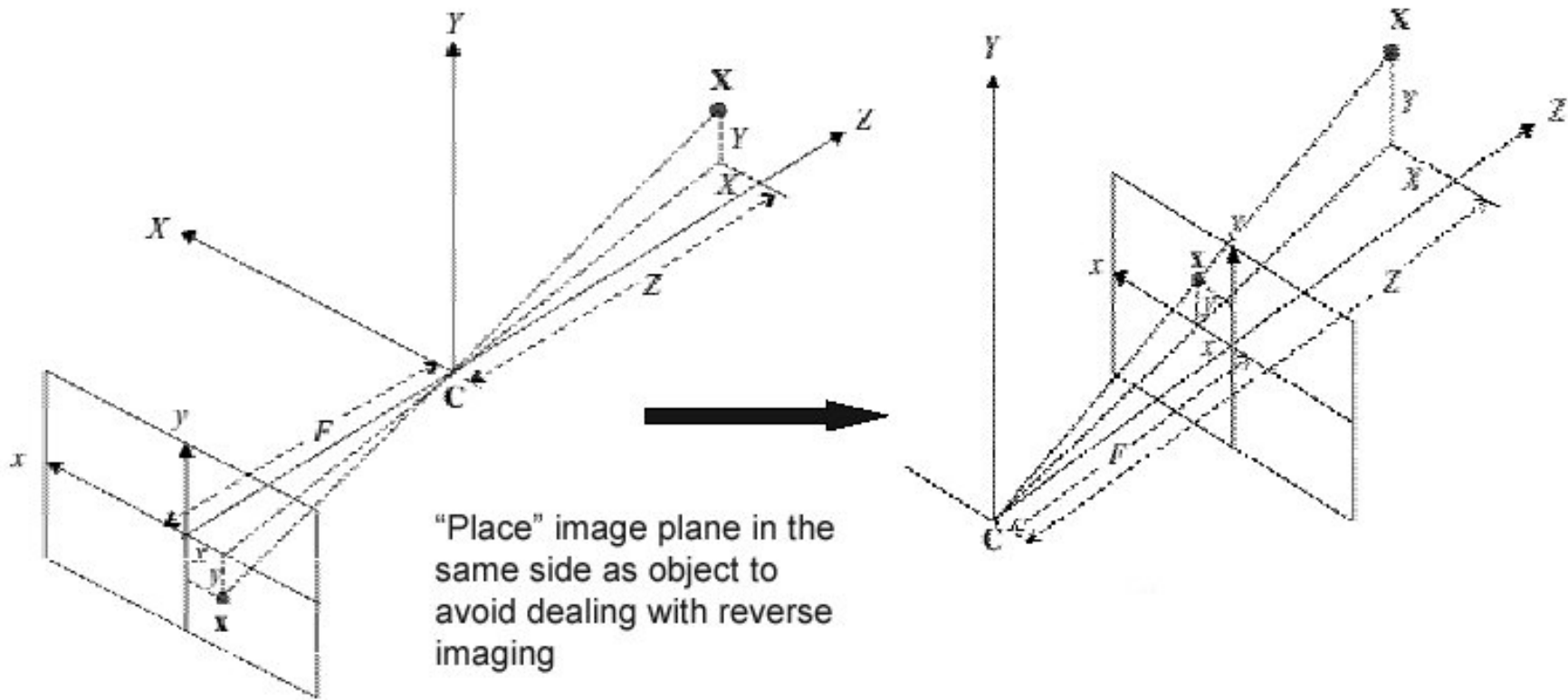
$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \end{bmatrix} = \begin{bmatrix} 0 & \Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

# Geometric image formation(1/3)

---

- Imaging systems capture 2-D projections of a time-varying 3-D scene at certain intervals of time.
- **$P$** :  $R^4 \rightarrow R^2 \times Z$
- $(X_1, X_2, X_3, t) \rightarrow (x_1, x_2, k)$

# Geometric image formation(2/3)

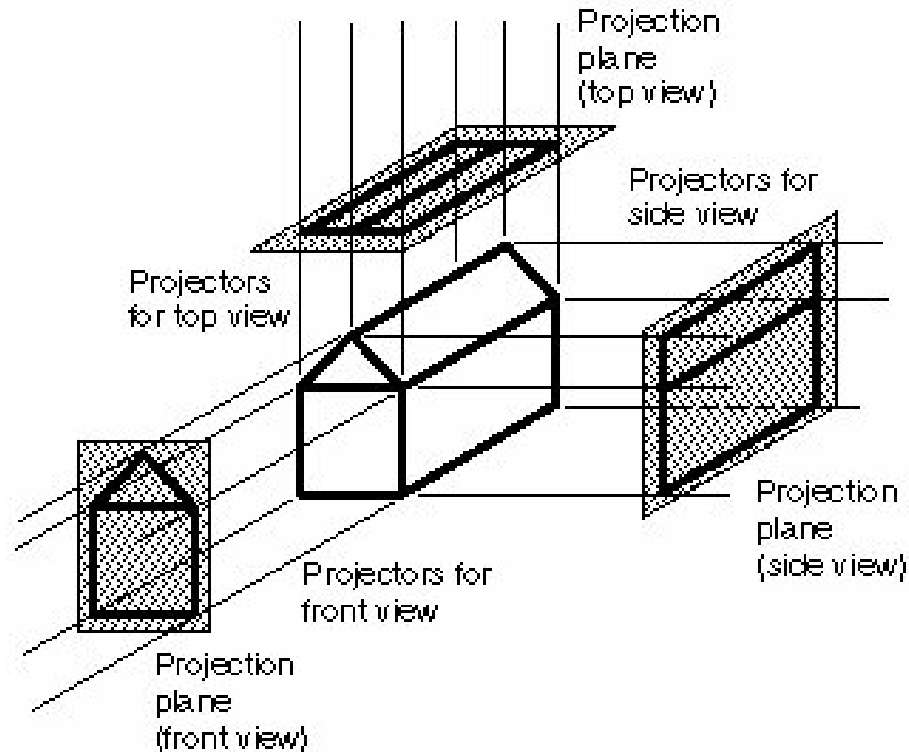


$$x = F \frac{X}{Z}$$

$$y = F \frac{Y}{Z}$$

# Geometric image formation(3/3)

## ■ Orthographic projection



$$x = X$$

$$y = Y$$



## Optical flow/correspondence

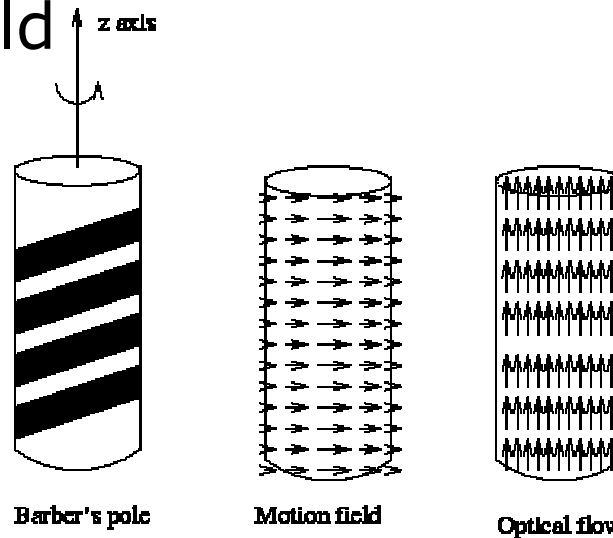
---

- Observable variations in the 2-D image intensity pattern (also called the apparent 2-D motion).
- *Optical flow* refers to a dense vector field indicating rate of change of intensity variations in  $x_1$  and  $x_2$  directions.
- *Dense correspondence field* refers to a set of frame-to-frame motion vectors indicating matching pixel pairs.

# Optical Flow Vs. Motion Field (1/2)

---

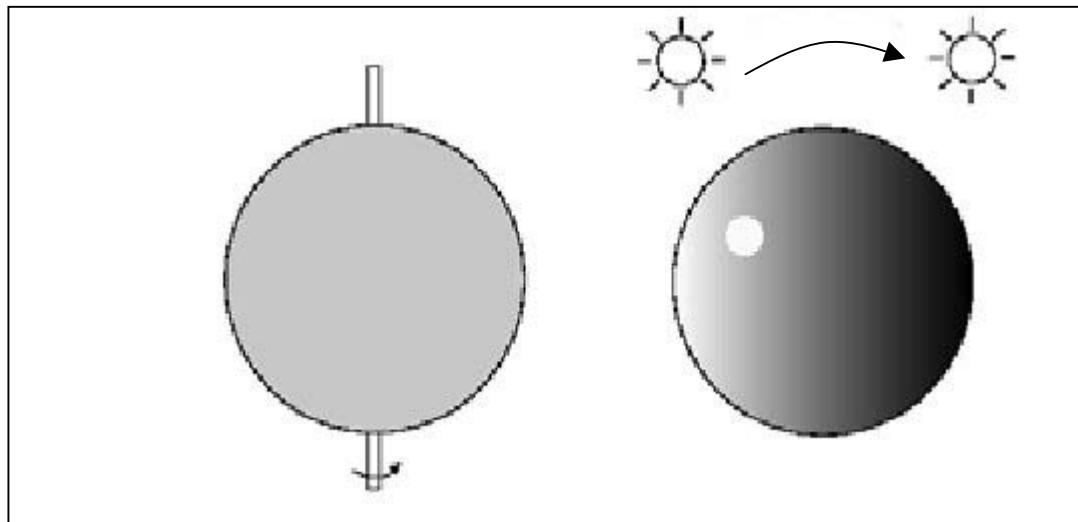
- Optical flow does not always correspond to motion field



- Optical flow is an approximation of the motion field. The error is small at points with high spatial gradient under some simplifying assumptions

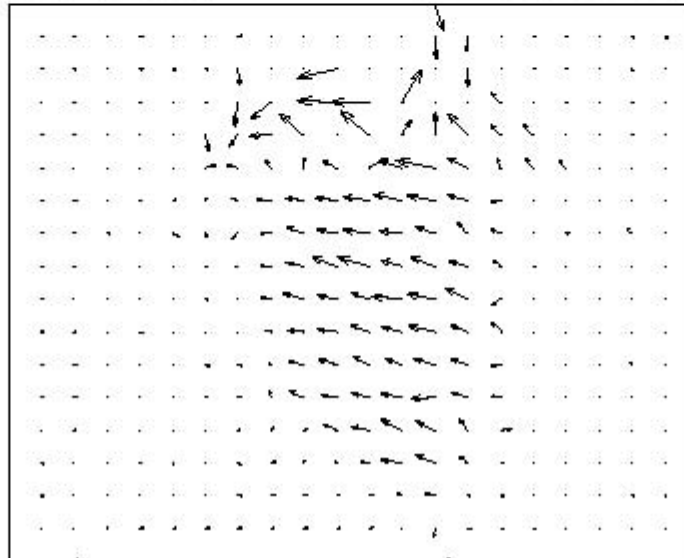
## Optical Flow Vs. Motion Field (2/2)

---



Optical flow (observed/apparent 2-D motion) may not be the same as the actual projected 2-D motion.

# 2-D motion field



# Optical flow equation

---

$$f(x + d_x, y + d_y, t + d_t) = f(x, y, t) \quad \text{Const luminance assumpt.}$$

$$LHS \approx f(x, y, t) + \frac{\partial f}{\partial x} d_x + \frac{\partial f}{\partial y} d_y + \frac{\partial f}{\partial t} d_t \quad \text{Taylor's expansion}$$

$$\Rightarrow \frac{\partial f}{\partial x} d_x + \frac{\partial f}{\partial y} d_y + \frac{\partial f}{\partial t} d_t = 0$$

$$\Rightarrow \frac{\partial f}{\partial x} v_x + \frac{\partial f}{\partial y} v_y + \frac{\partial f}{\partial t} = 0, \text{ or } (\nabla f)^T \mathbf{v} + \frac{\partial f}{\partial t} = 0$$

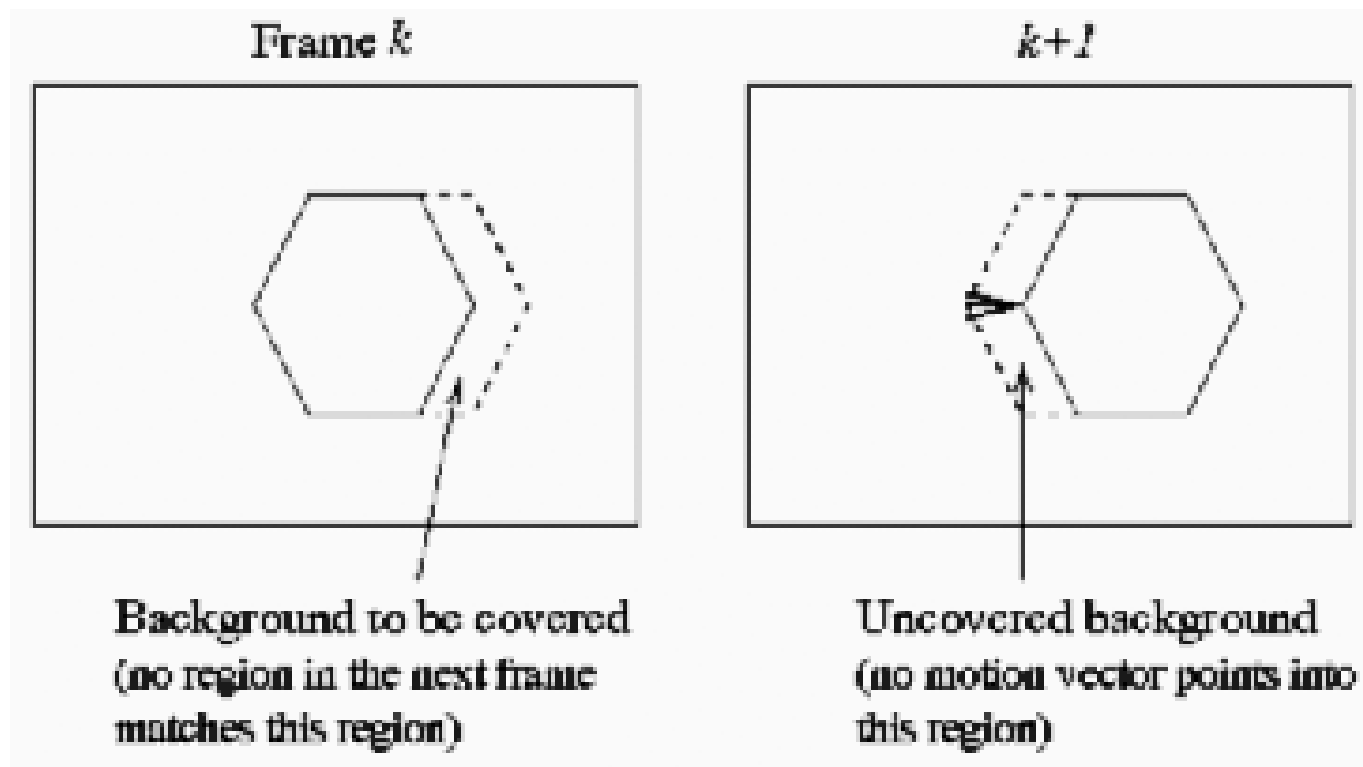
with  $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]^T$  being spatial gradient vector of  $f(x, y, t)$

# Ambiguity in motion estimation

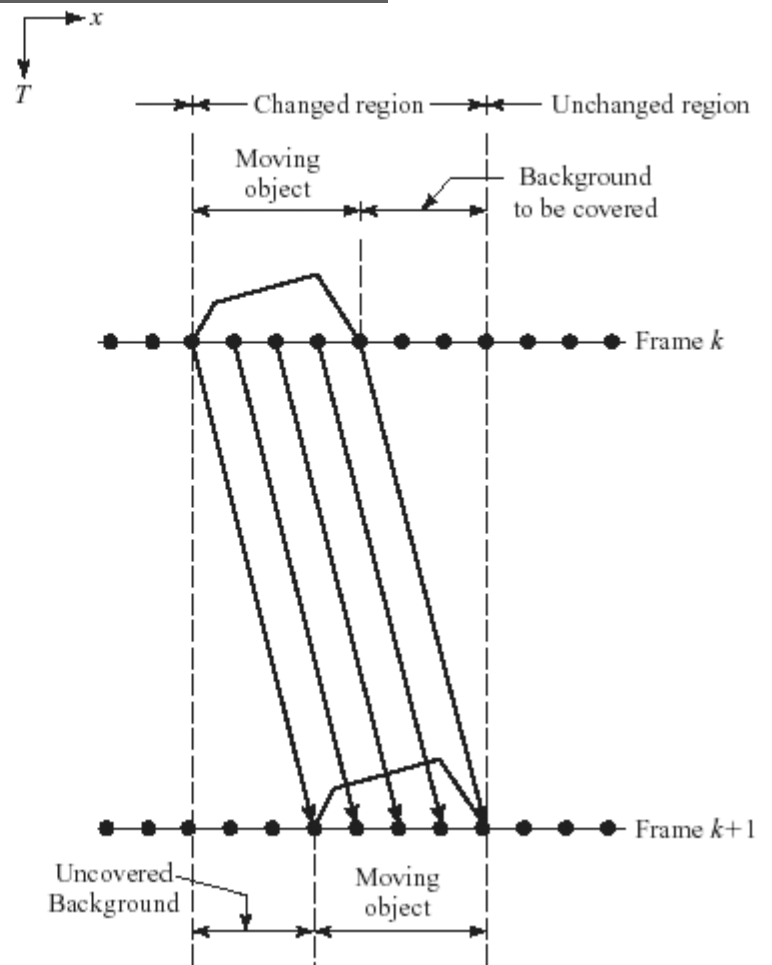
## (1/2)

---

- *Existence of a solution:* Uniform, covered/uncovered background

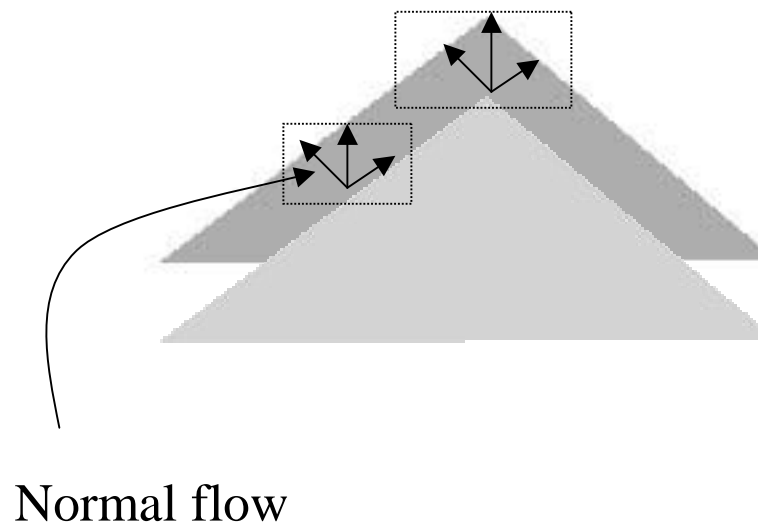


Motion is undefined in occluded regions



# Ambiguity in motion estimation (2/2)

*Uniqueness of a solution:* One equation for two unknowns; Aperture problem; Normal flow can be estimated; Additional constraints



$$v_{\perp}(\mathbf{x}, t) = \frac{-\frac{\partial s(\mathbf{x}, t)}{\partial t}}{\|\nabla s(\mathbf{x}, t)\|}$$



## **Additional constraints**

---

- OFE may be imposed on each color channel separately.
  - The displacement vector can then possibly be constrained in three different directions, if the direction of the spatial gradient at each band is different.
- Smoothness constraint


# Displaced Frame Difference

---

- The DFD between two frames at  $t$  and  $t + \Delta t$

$$dfd(\mathbf{x}, \hat{\mathbf{d}}) = I(\mathbf{x} + \mathbf{d}(\mathbf{x}), t + \Delta t) - I(\mathbf{x}, t)$$

Intensity      displacement



## Relation between DFE and OFE

---

- Use Taylor series expansion (small  $\mathbf{d}$  and  $\Delta t$ )

$$I(\mathbf{x} + \mathbf{d}(\mathbf{x}), t + \Delta t) = I(x, t) + \frac{\delta I(\mathbf{x}, t)}{\delta x_1} d_1(\mathbf{x}) + \frac{\delta I(\mathbf{x}, t)}{\delta x_2} d_2(\mathbf{x}) + \frac{\delta I(\mathbf{x}, t)}{\delta t} \Delta t$$

# OFE and DFD constraints

---

- In practice, neither the DFD nor the error in the OFE is exactly zero because
  - there is observation noise
  - scene illumination may vary
  - there are occlusion regions
  - there are interpolation errors
- Therefore one minimizes absolute or the square value of DFD or LHS of OFE

# General methods in motion estimation

---

## ■ Feature based

- Establish correspondence
- Estimate parameters

## ■ Intensity based

- Apply optical flow equation or its variants
- Good for multiple objects
- Used in video coding

# **Key issues in motion estimation**

---

- How to parameterize underlying motion field?
- What estimation criteria?
- How to search for optimal parameters?

# Motion models

---

- Parametric models
  - Translational motion
  - Affine motion
  - Perspective motion
  - Bilinear/quadratic motion
- Quasi-parametric models
- Non-parametric models

# Parametric motion models

---

- Affine motion model (6-parameters)

$$x_1' = a_1x_1 + a_2x_2 + a_3$$

$$x_2' = a_4x_1 + a_5x_2 + a_6$$

- Perspective motion model (8-parameters)

$$x_1' = \frac{a_1x_1 + a_2x_2 + a_3}{a_7x_1 + a_8x_2 + 1} \quad x_2' = \frac{a_4x_1 + a_5x_2 + a_6}{a_7x_1 + a_8x_2 + 1}$$

Bilinear motion model (8-parameters)

$$x_1' = a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4$$

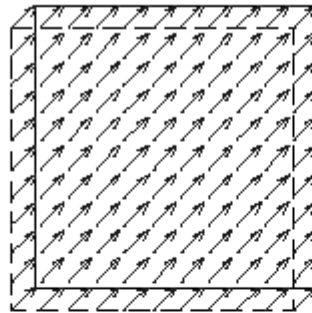
$$x_2' = a_5x_1 + a_6x_2 + a_7x_1x_2 + a_8$$



# Motion Field Corresponding to Different 2-D Motion Models

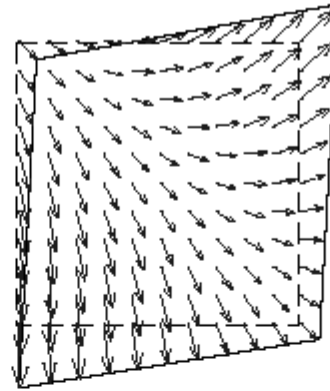
---

Translation



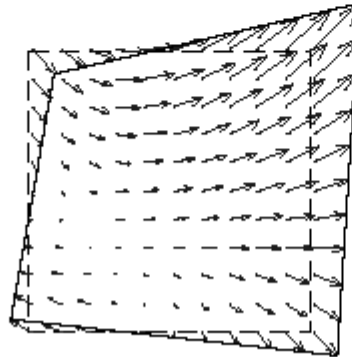
(a)

Affine



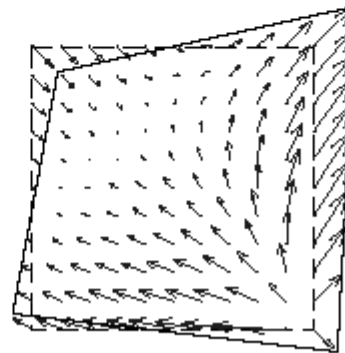
(b)

Bilinear



(c)

Perspective



(d)

# Motion representations (1/4)

---

- Pixel based representation
  - Specify MV for each pixel
  - Computationally expensive
  - Need additional constraints
- Global motionrepresentation
  - Good if major motion is camera motion
- Region based representation
  - One set of motion param for each region
  - Need iterative segmentation and estimation

# Motion representations (2/4)

---

## ■ Block based representation

- Fixed partitioning into blocks and characterize each with simple model
- Pro: Good compromise between accuracy and complexity; Con: False discontinuities btw blocks

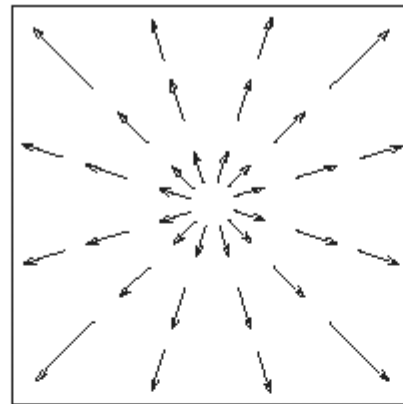
## ■ Mesh based representation

- Partition image into polygons
- Pro: Provide continuous motion, Con: Problem with object boundaries

# Motion representations (3/4)

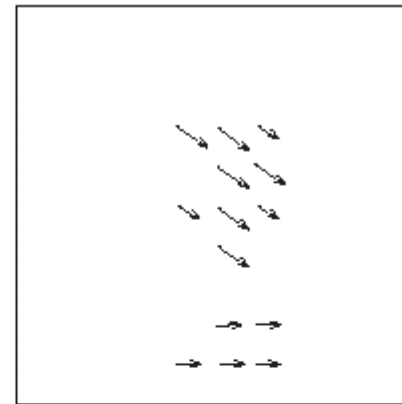
---

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



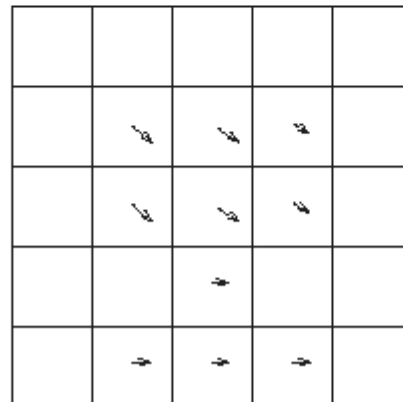
(a)

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



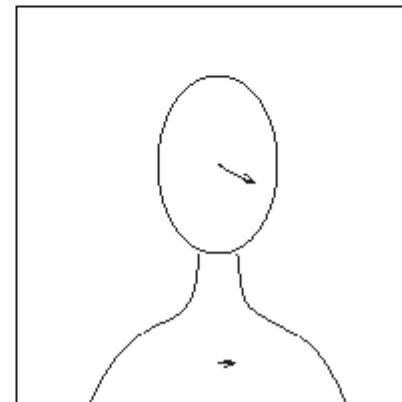
(b)

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)

# Motion representations (4/4)

---



# Nonparametric 2-D motion estimation

---

- Methods based on OFE
- Phase correlation method
- Block matching method
- Pel-recursive methods
- Bayesian methods

# Methods using OFE – Horn-Schunck Method

---

- Two criteria:
  - Optical flow is smooth,  $\mathbf{E}_s(u, v)$
  - Small error in optical flow constraint equation,  $\mathbf{E}_{\text{of}}(v_x, v_y)$
- Minimize a combined error functional
$$\mathbf{E}_c(u, v) = \mathbf{E}_{\text{of}}^2(u, v) + \lambda \mathbf{E}_s^2(u, v)$$

$\lambda$  is a weighting parameter
- Solved iteratively

# Error functional

---

## ■ OF cost

$$E_{of} = \frac{\delta}{\delta} \frac{I}{x} u + \frac{\delta}{\delta} \frac{I}{y} v + \frac{\delta}{\delta} \frac{I}{t}$$

## ■ Smoothness cost

$$E_s^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$$

$$E_s^2 = \|\nabla u\|^2 + \|\nabla v\|^2$$



## Horn-Schunck Algorithm : Discrete Case

---

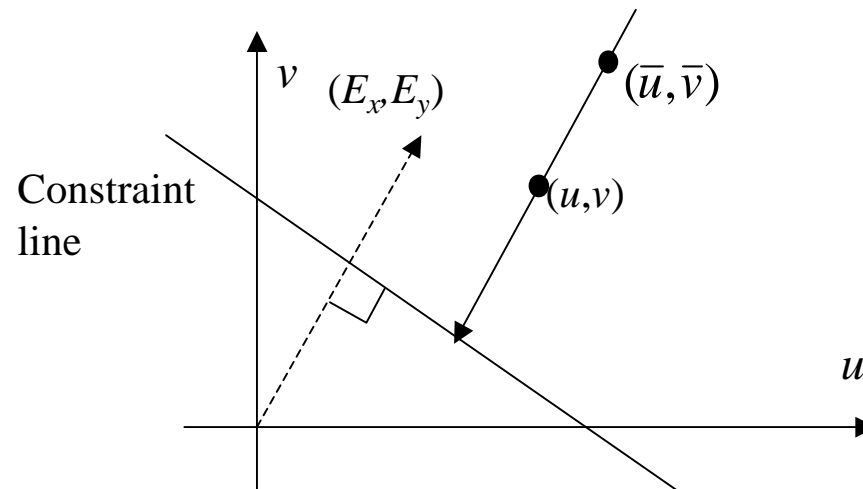
- Derivatives (and error functionals) are approximated by difference operators
- Leads to an iterative solution:

$$\begin{aligned} u_{ij}^{n+1} &= \bar{u}_{ij}^n - \alpha E_x \\ v_{ij}^{n+1} &= \bar{v}_{ij}^n - \alpha E_y \end{aligned} \quad \alpha = \frac{E_x \bar{u}_{ij}^n + E_y \bar{v}_{ij}^n + E_t}{1 + \lambda(E_x^2 + E_y^2)}$$

$\bar{u}, \bar{v}$  is the average of values of neighbors

# Intuition of the Iterative Scheme

---



The new value of  $(u, v)$  at a point is equal to the average of surrounding values minus an adjustment in the direction of the brightness gradient

# Horn - Schunck Algorithm (1/2)

```

begin
  for j:= 1 to N do for i:= 1 to M do begin
    calculate the values  $E_x(i, j, t)$ ,  $E_y(i, j, t)$ , and  $E_t(i, j, t)$  using
      a selected approximation formula;
      { special cases for image points at the image border
        have to be taken into account }
    initialize the values  $u(i, j)$  and  $v(i, j)$  with zero
  end {for};
  choose a suitable weighting value  $\lambda$ ; { e.g.  $\lambda = 10$  }
  choose a suitable number  $n_0 \geq 1$  of iterations; {  $n_0 = 8$  }
  n:= 1; { iteration counter }
  while n  $\leq$   $n_0$  do begin
    for j:= 1 to N do for i:= 1 to M do begin
       $\bar{u} := \frac{1}{4}(u(i-1, j) + u(i+1, j) + u(i, j-1) + u(i, j+1))$ ;
       $\bar{v} := \frac{1}{4}(v(i-1, j) + v(i+1, j) + v(i, j-1) + v(i, j+1))$ ;
      { treat image points at the image border separately }
       $\alpha := \frac{E_x(i, j, t)\bar{u} + E_y(i, j, t)\bar{v} + E_t(i, j, t)}{1 + \lambda(E_x^2(i, j, t) + E_y^2(i, j, t))} \cdot \lambda$ ;
       $u(i, j) := \bar{u} - \alpha \cdot E_x(i, j, t)$ ;  $v(i, j) := \bar{v} - \alpha \cdot E_y(i, j, t)$ 
    end {for};
    n:= n+1
  end {while}
end;
```

# Horn-Schunck Algorithm(2/2)

---

**begin**

**for**  $j := 1$  to  $N$  **do**      **for**  $l := 1$  to  $M$  **do**    **begin**

    calculate the values  $E_x(i,j,t)$ ,  $E_y(i,j,t)$  and  $E_t(i,j,t)$  using a selected approx formula

    initialize the values  $u(l,j)$  and  $v(i,j)$  to zero

**end** {for}

choose a suitable weighting value  $\lambda$

choose a suitable number  $n_0 \geq 1$  of iterations

$n := 1$

**while**  $n \leq n_0$  **do begin**

**for**  $j := 1$  to  $N$  **do**    **for**  $i := 1$  to  $M$  **do**    **begin**

        compute  $\underline{u}$ ,  $\underline{v}$ ,  $\alpha$

        update  $u(i,j)$ ,  $v(i,j)$

**end** {for}

$n := n + 1$

**end** {while}

**end**

# Finite difference methods

---

- Forward difference
- Backward difference
- Average difference
- Local average of the average differences
- Horn and Schunck proposed averaging four finite differences

$$\frac{\partial I}{\partial x} = \frac{1}{4} \left\{ \begin{aligned} &I(x+1, y, t) - I(x, y, t) + I(x+1, y+1, t) - I(x, y+1, t) + \\ &I(x+1, y, t+1) - I(x, y, t+1) + I(x+1, y+1, t+1) - I(x, y+1, t+1) \end{aligned} \right\}$$

# Local polynomial fitting method

---

- Approximate  $I(x,y,t)$  locally by a linear combination of some low order polynomials

$$\hat{I}(x, y, t) = \sum_{i=0}^{N-1} a_i \phi_i(x, y, t)$$

- For  $N = 9 \rightarrow$  Basis functions:  
 $1, x, y, t, x^2, y^2, xy, xt, yt$

# Coefficient estimation

---

- Coefficients  $a_i$  are estimated using the least squares method

$$e^2 = \sum_{n1} \sum_{n2} \sum_{n3} (I(x, y, t) - \sum_{i=0}^{N-1} a_i \phi_i(x, y, t))^2 \Big|_{x=n1, y=n2, t=n3}$$

# Lucas and Kanade

---

- Assumes that the motion vectors remain unchanged over a particular block of pixels

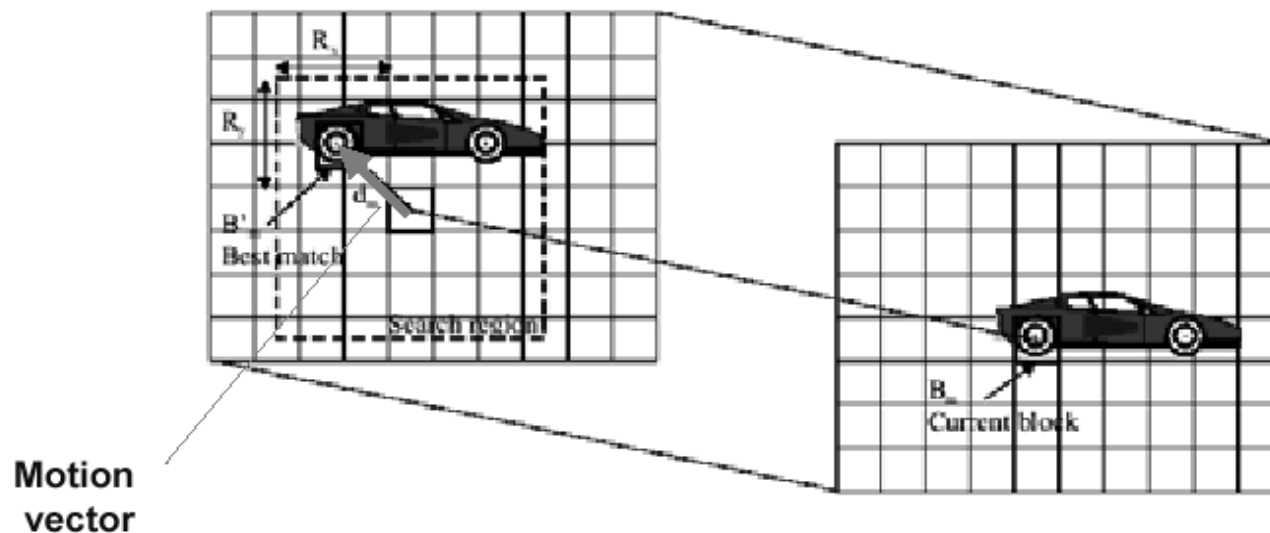
- Minimize  $E = \sum_{x \in B} \left( \frac{\delta I}{\delta x} u + \frac{\delta I}{\delta y} v + \frac{\delta I}{\delta t} \right)^2$

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \begin{bmatrix} \sum_{x \in B} \frac{\partial I}{\partial x} \frac{\partial I}{\partial x} & \sum_{x \in B} \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \\ \sum_{x \in B} \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} & \sum_{x \in B} \frac{\partial I}{\partial y} \frac{\partial I}{\partial y} \end{bmatrix}^{-1} \begin{bmatrix} - \sum_{x \in B} \frac{\partial I}{\partial x} \frac{\partial I}{\partial t} \\ - \sum_{x \in B} \frac{\partial I}{\partial y} \frac{\partial I}{\partial t} \end{bmatrix}$$



# Block matching

- Assume block-based translation motion model
- Search every possibility over a specified range for the best matching block
  - MAD (mean absolute difference) often used for simplicity



# Search procedures

---

- Usually the search area is limited to

$$-M_1 \leq d_1 \leq M_1$$

$$-M_2 \leq d_2 \leq M_2$$

- Methods

- Full(exhaustive search)
- Three-step search
- Cross search

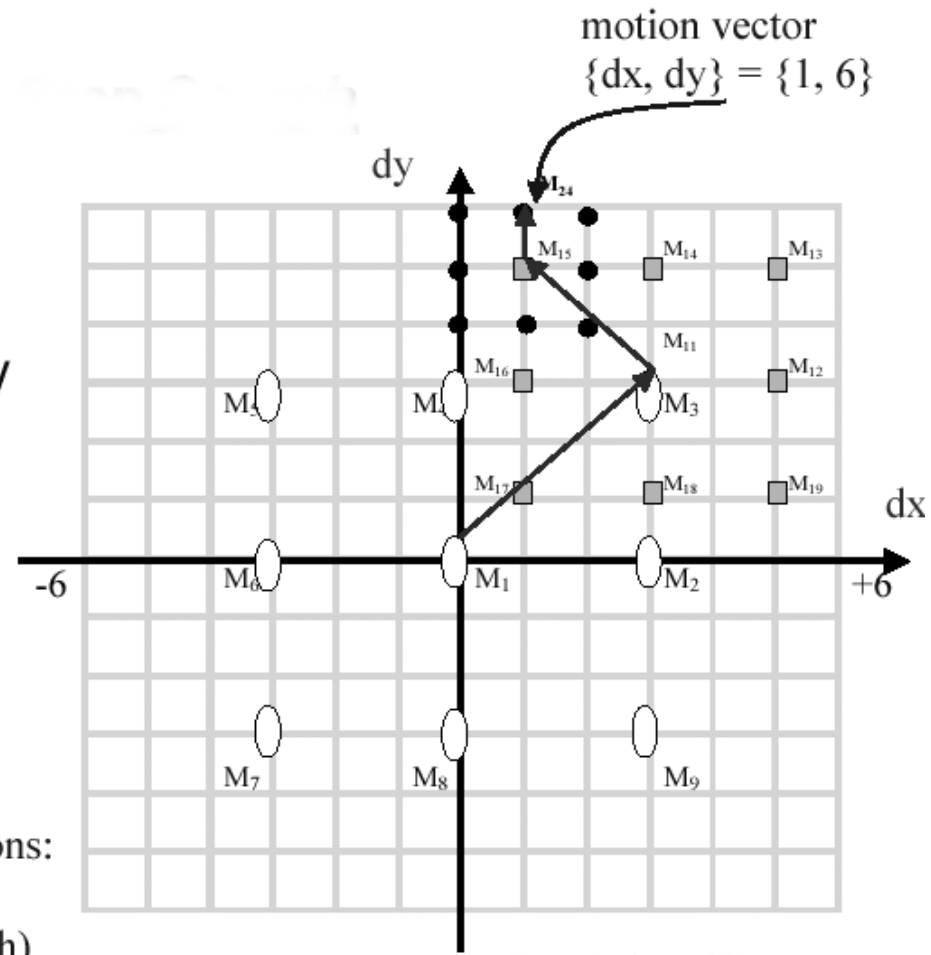
# Exhaustive search: Pros and cons

- ≡ ● Pros
  - Guaranteed optimal within search range
- Cons
  - Can only search among finitely many candidates
    - ◆ *What if the motion is “fractional”?*
  - High computation complexity
    - ◆ *On the order of [search-range-size \* image-size] for 1-pixel step size*
- ➔ How to improve accuracy?
  - Include blocks at fractional translation as candidates ~ require interpolation
- ➔ How to improve speed?
  - Try to exclude unlikely candidates

# 3-step search

- Search candidates at 8 neighbour positions
- Step-size cut down by 2 after each iteration
  - Start with step size approx. half of max. search range

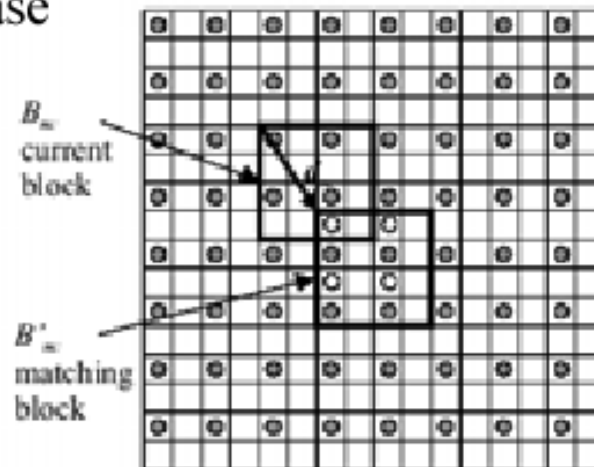
Total number of computations:  
 $9 + 8 \times 2 = 25$  (3-step)  
 $(2R+1)^2 = 169$  (full search)



# Fractional accuracy

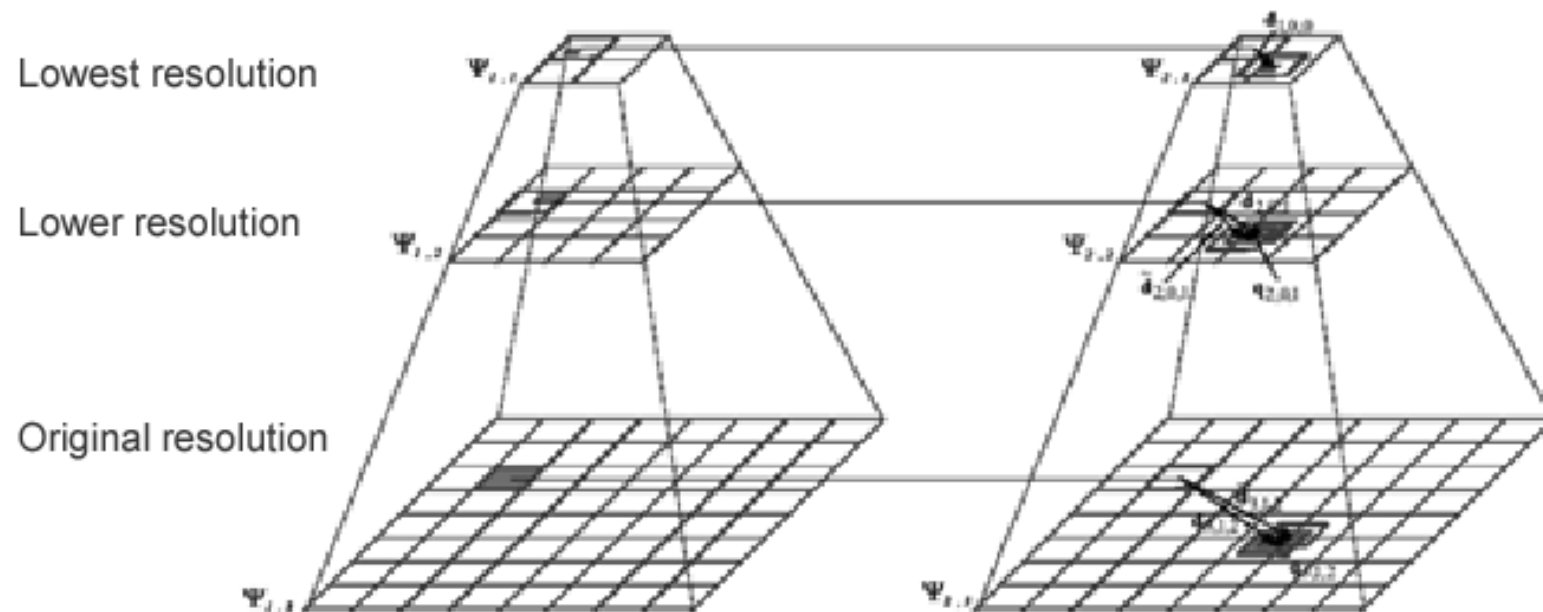
---

- For motion accuracy of  $1/K$  pixel
  - Upsample (interpolate) reference frame by a factor of  $K$
  - Search for the best matching block in the upsampled reference frame
- Half-pel accuracy  $\sim K=2$ 
  - Significant accuracy improvement over integer-pel (esp. for low-resolution)
  - Complexity increase



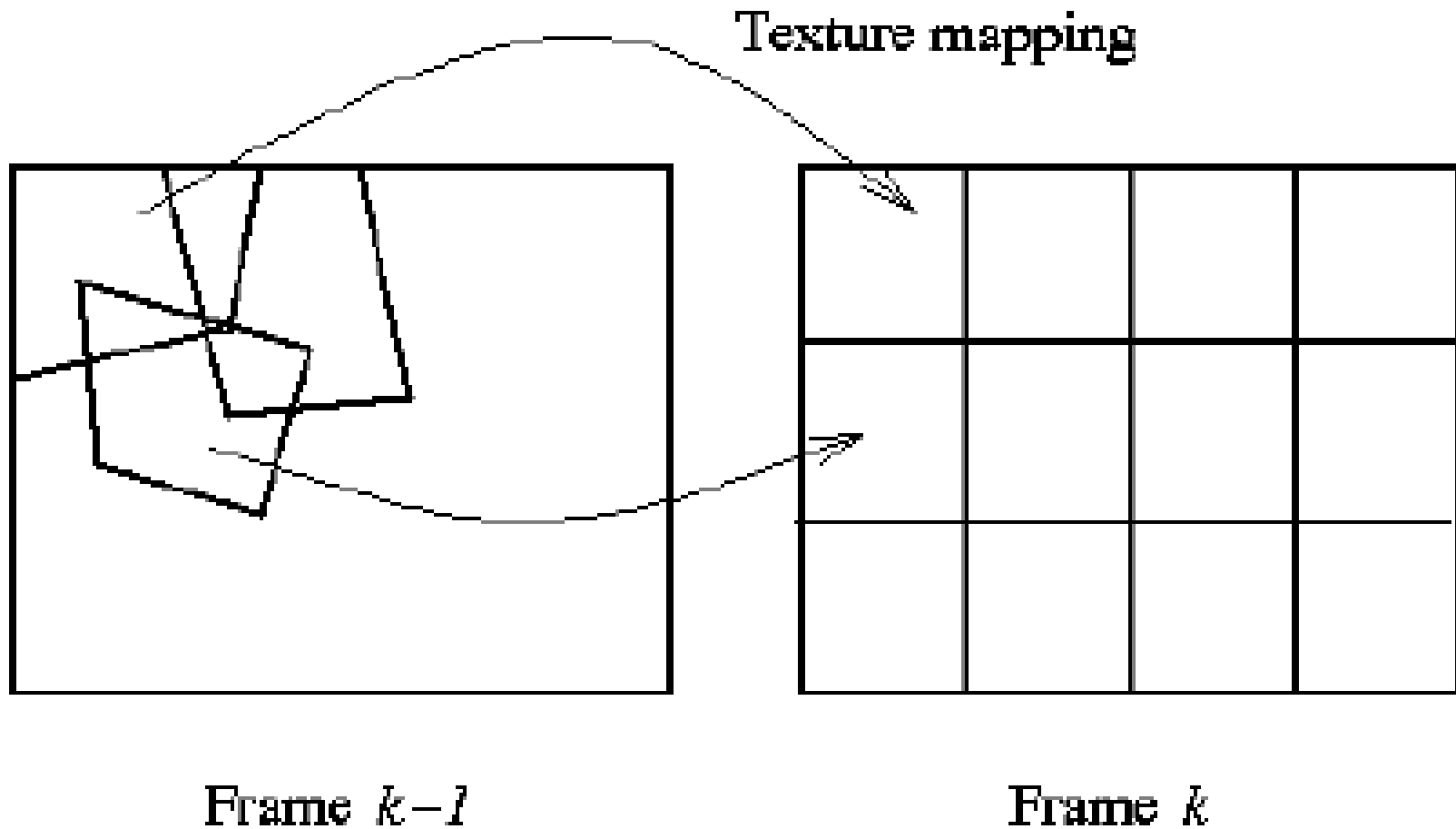
# Hierarchical block matching

- Problem with fast search at full resolution
  - Small mis-alignment may give high displacement error ( $E_{DFD}$ )
    - ♦ *esp. for texture and edge blocks*
- Hierarchical (multi-resolution) block matching
  - Match with coarse resolution to narrow down search range
  - Match with high resolution to refine motion estimation



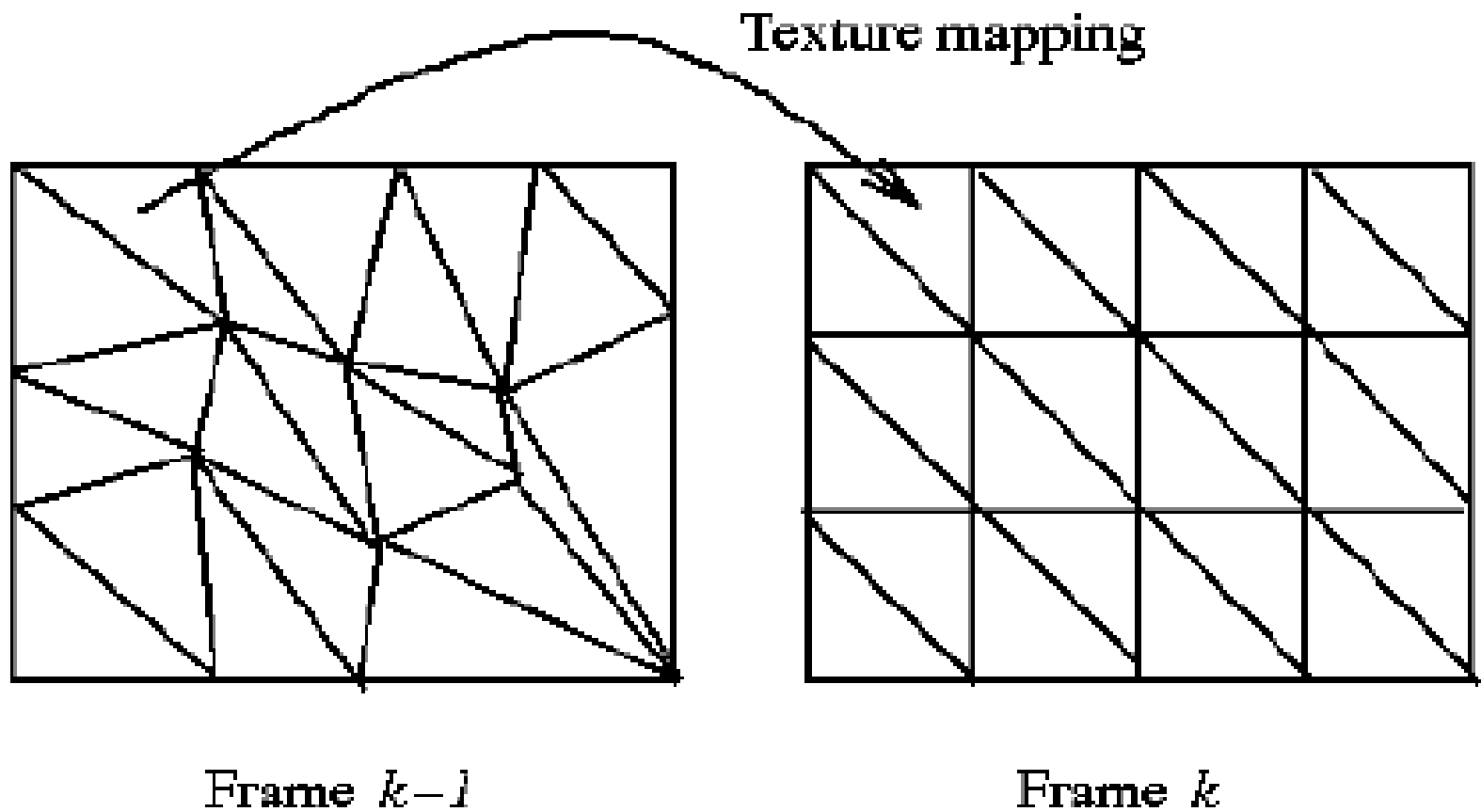
# Generalized block matching

---



# Mesh modeling

---





## Phase correlation method (1/4)

---

- The correlation between the frames  $k$  and  $k+1$  is given by

$$c_{k,k+1}(n_1, n_2) = s(n_1, n_2, k+1) ** s(-n_1, -n_2, k)$$

- Taking the Fourier transform of both sides

$$C_{k,k+1}(f_1, f_2) = S_{k+1}(f_1, f_2) S_k^*(f_1, f_2)$$

- Normalizing  $C_{k,k+1}(f_1, f_2)$  by its magnitude

$$\tilde{C}_{k,k+1}(f_1, f_2) = \frac{S_{k+1}(f_1, f_2) S_k^*(f_1, f_2)}{|S_{k+1}(f_1, f_2) S_k^*(f_1, f_2)|}$$

## Phase correlation method (2/4)

---

- Given the motion model

$$S_{k+1}(f_1, f_2) = S_k(f_1, f_2) \exp\{-j2\pi(f_1 d_1 + f_2 d_2)\}$$

we have

$$\tilde{C}_{k,k+1}(f_1, f_2) = \exp\{-j2\pi(f_1 d_1 + f_2 d_2)\}$$

and

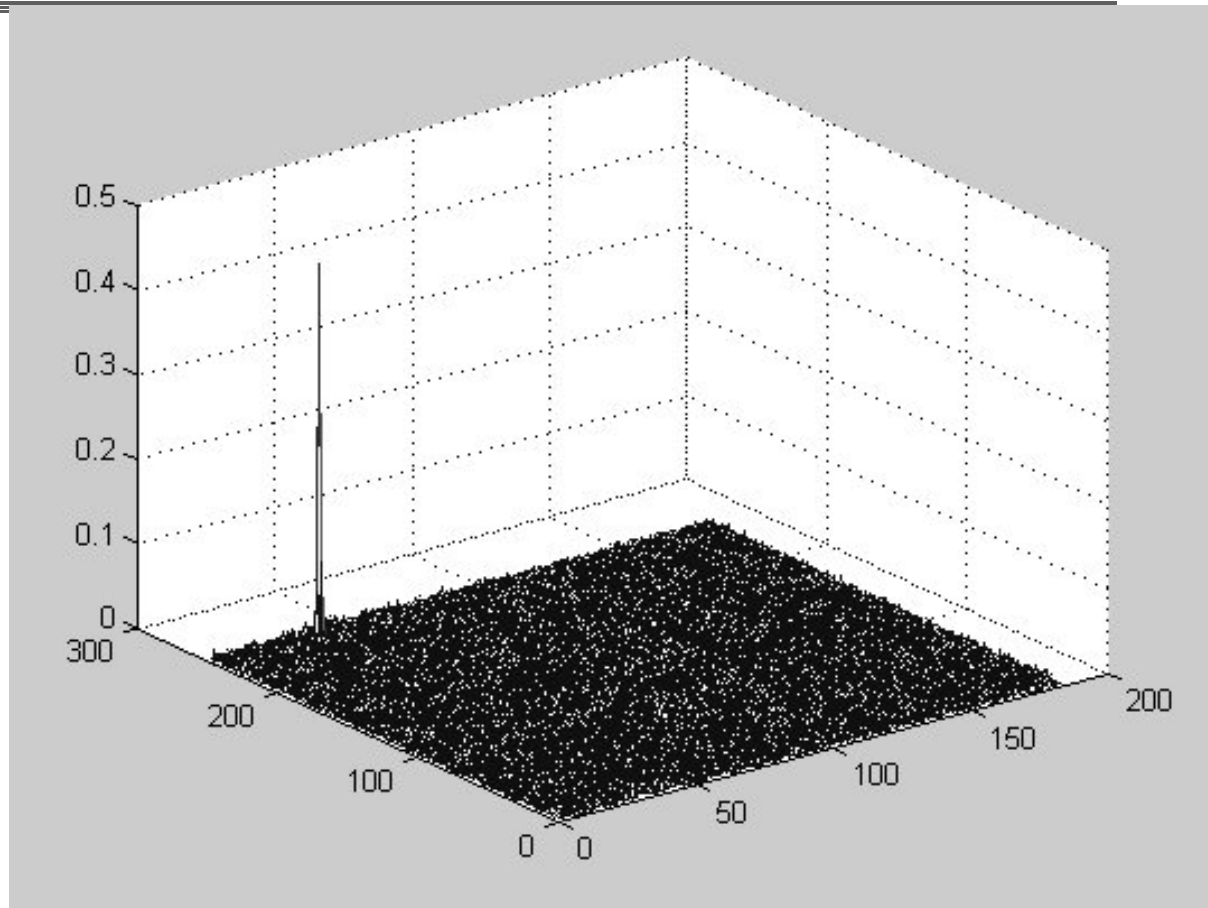
$$\tilde{c}_{k,k+1}(n_1, n_2) = \delta(n_1 - d_1, n_2 - d_2)$$

# Phase correlation method (3/4)

---



# Phase correlation method (4/4)



# **Pel-recursive and Bayesian methods**

---

- Pel-recursive methods
- Review of random fields
- Review of optimization methods
- Bayesian methods

# Pel-recursive methods

---

- Pel-recursive algorithms are of the general form

$$\mathbf{d}^{(i+1)}(\mathbf{x}) = \mathbf{d}^{(i)}(\mathbf{x}) + \mathbf{u}^{(i)}(\mathbf{x})$$

where  $\mathbf{d}^{(i)}(\mathbf{x})$  is the estimated motion vector at the pel location  $\mathbf{x}$  at the  $i$ th step,  $\mathbf{u}^{(i)}(\mathbf{x})$  is the update term at the  $i$ th step, and  $\mathbf{d}^{i+1}(\mathbf{x})$  is the new estimate.

- The update term  $\mathbf{u}^{(i)}(\mathbf{x})$  is estimated, at each pel  $\mathbf{x}$ , to minimize a positive-definite function  $E$  of the DFD with respect to  $\mathbf{d}$ .
- The motion estimate at the previous pel is taken as the initial estimate at the next pel, hence *pel-recursive*.

# Displaced Frame Difference

---

- The DFD between two frames at  $t$  and  $t + \Delta t$

$$dfd(\mathbf{x}, \mathbf{d}) = I(\mathbf{x} + \mathbf{d}(\mathbf{x}), t + \Delta t) - I(\mathbf{x}, t)$$

Intensity

displacement

$\mathbf{u}^i(\mathbf{x})$  is estimated by minimizing at each  $\mathbf{x}$  a positive definite function  $\mathbf{E}$  of the DFD wrt  $\mathbf{d}$ , so that minimum occurs when  $DFD=0$

# Minimization by Gradient Descent (1/2)

---

- ◆ A straightforward way to minimize a function is to set its derivatives to zero:

$$\nabla_d E(\mathbf{x}; d) = 0$$

where  $\nabla_d$  is the gradient operator with respect to  $d$ , the set of partial derivatives.

- ◆ The following equation must be solved simultaneously:

$$\frac{\partial E(\mathbf{x}; d)}{\partial d_1} = 0$$

$$\frac{\partial E(\mathbf{x}; d)}{\partial d_2} = 0$$

- ◆ Since an analytical solution to these equations cannot be found in general, we resort to iterative methods.

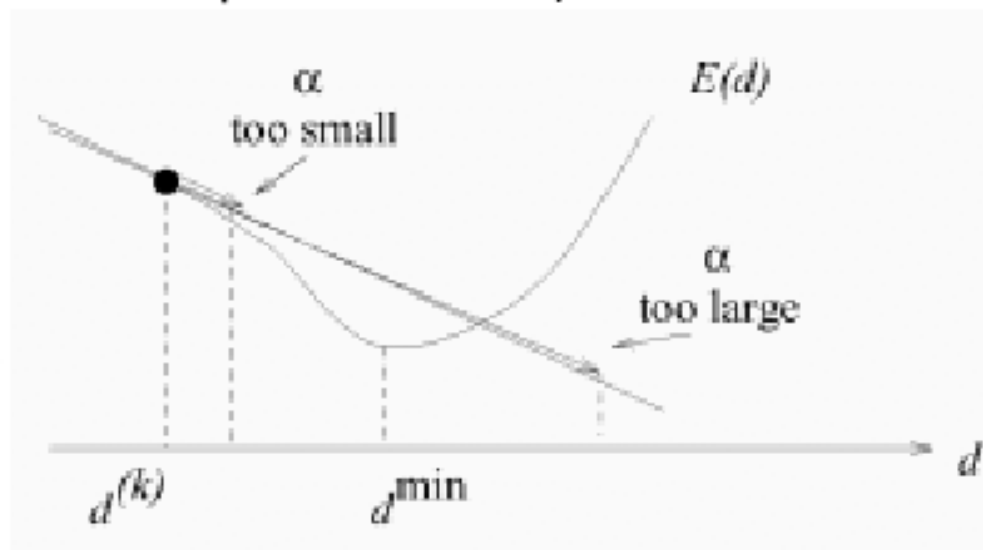


# Minimization by Gradient Descent (2/2)

- ◆ The gradient vector points AWAY from the minimum. That is, in one dimension, its sign will be positive on an “uphill” slope. Thus, to get closer to the minimum, we can update our current vector as

$$d^{(k+1)}(\mathbf{x}) = d^{(k)}(\mathbf{x}) - \alpha \nabla_d E(\mathbf{x}; d) \big|_{d^{(k)}(\mathbf{x})}$$

where  $\alpha$  is some positive scalar, known as the step size.



- ◆ If  $\alpha$  is too small, the iteration will take too long to converge, if it is too large the algorithm will become unstable and start oscillating about the minimum.

## Newton-Raphson method

- ◆ We can estimate a good value for  $\alpha$  using the well-known Newton-Raphson method for root finding

$$\mathbf{d}^{(k+1)}(\mathbf{x}) = \mathbf{d}^{(k)}(\mathbf{x}) - \mathbf{H}^{-1} \nabla_{\mathbf{d}} E(\mathbf{x}; d) \big|_{\mathbf{d}^{(k)}(\mathbf{x})}$$

where  $\mathbf{H}$  is the Hessian matrix

$$\mathbf{H}_{ij} = \left[ \frac{\partial^2 E(\mathbf{x}; d)}{\partial d_i \partial d_j} \right]$$

- ◆ In one dimension, we would like to find a root of  $E'(d)$ . Expanding  $E'(d)$  in a Taylor series about the point  $d^{(k)}$

$$E'(d^{(k+1)}) = E'(d^{(k)}) + (d^{(k+1)} - d^{(k)})E''(d^{(k)})$$

Since we want  $d^{(k+1)}$  to be zero of  $E'$ , we set

$$E'(d^{(k)}) + (d^{(k+1)} - d^{(k)})E''(d^{(k)}) = 0$$

Thus,

$$d^{(k+1)} = d^{(k)} - \frac{E'(d^{(k)})}{E''(d^{(k)})}$$

# Local vs Global minima

---

- ◆ Gradient descent suffers from a serious problem: its solution is strongly dependent on the starting point. If start in a “valley”, it will be stuck at the bottom of that valley. This may be a “local” minimum. We have no way of getting out of that local minimum to reach the “global” minimum.
- ◆ More sophisticated optimization methods, such as simulated annealing are needed to be able to reach the global minimum regardless of the starting point. However, these more sophisticated optimization methods usually require a lot more processing time.

## Netravali-Robbins Algorithm

- ◆ The Netravali-Robbins algorithm finds an estimate of the displacement vector at each pixel to minimize

$$E(\mathbf{x}; \mathbf{d}) = [dfd(\mathbf{x}, \mathbf{d})]^2$$

A steepest descent approach to the minimization problem yields the iteration

$$\begin{aligned}\hat{\mathbf{d}}^{i+1}(\mathbf{x}) &= \hat{\mathbf{d}}^i(\mathbf{x}) - (1/2)\varepsilon \nabla_{\mathbf{d}} [dfd(\mathbf{x}, \mathbf{d})]_{\mathbf{d}=\hat{\mathbf{d}}^i} \\ &= \hat{\mathbf{d}}^i(\mathbf{x}) - \varepsilon dfd(\mathbf{x}, \hat{\mathbf{d}}^i) \nabla_{\mathbf{d}} dfd(\mathbf{x}, \mathbf{d})|_{\mathbf{d}=\hat{\mathbf{d}}^i},\end{aligned}$$

where  $\nabla$  is the gradient with respect to  $\mathbf{d}$ .

$$dfd(\mathbf{x}, \mathbf{d}) = dfd(\mathbf{x}, \hat{\mathbf{d}}^i) + \nabla_{\mathbf{x}}^T S_c(\mathbf{x} - \hat{\mathbf{d}}^i, t - \Delta t)(\mathbf{d} - \hat{\mathbf{d}}^i) + o(\mathbf{x}, \hat{\mathbf{d}}^i)$$

Hence  $\nabla_{\mathbf{d}} dfd(\mathbf{x}, \mathbf{d})|_{\mathbf{d}=\hat{\mathbf{d}}^i} = \nabla_{\mathbf{x}} S_c(\mathbf{x} - \hat{\mathbf{d}}^i, t - \Delta t)$

the estimate becomes

$$\hat{\mathbf{d}}^{i+1}(\mathbf{x}) = \hat{\mathbf{d}}^i(\mathbf{x}) - \varepsilon dfd(\mathbf{x}, \hat{\mathbf{d}}^i) \nabla_{\mathbf{x}} S_c(\mathbf{x} - \hat{\mathbf{d}}^i, t - \Delta t)$$

# Walker and Rao algorithm

- ◆ Walker and Rao suggested the following step size

$$\varepsilon = \frac{1}{2 \left\| \nabla_{\mathbf{x}} s_c(\mathbf{x} - \hat{\mathbf{d}}^i, t - \Delta t) \right\|^2}.$$

This is motivated by the update term

- should be large when  $|dfd(\cdot)|$  is large and  $|\nabla s_c(\cdot)|$  is small, and
- should be small when  $|dfd(\cdot)|$  is small and  $|\nabla s_c(\cdot)|$  is large.

- ◆ Caffario and Rocca have added a bias term  $\eta^2$  to avoid division by zero in the areas of constant intensity

$$\varepsilon = \frac{1}{\left\| \nabla_{\mathbf{x}} s_c(\mathbf{x} - \hat{\mathbf{d}}^i, t - \Delta t) \right\|^2 + \eta^2}.$$

## Remarks on pel-recursive methods

- The "pel-recursive" nature of the algorithm constitutes an *implicit smoothness* constraint. The effectiveness of this constraint increases especially when a small number of iterations are performed at each pixel.
- The aperture problem can be observed in that the update term is a vector along the direction of the gradient of the image intensity. Thus, no correction is performed in the direction perpendicular to the gradient vector.
- Pel-recursive algorithms can be applied hierarchically, using multi-resolution

# Bayesian methods

---

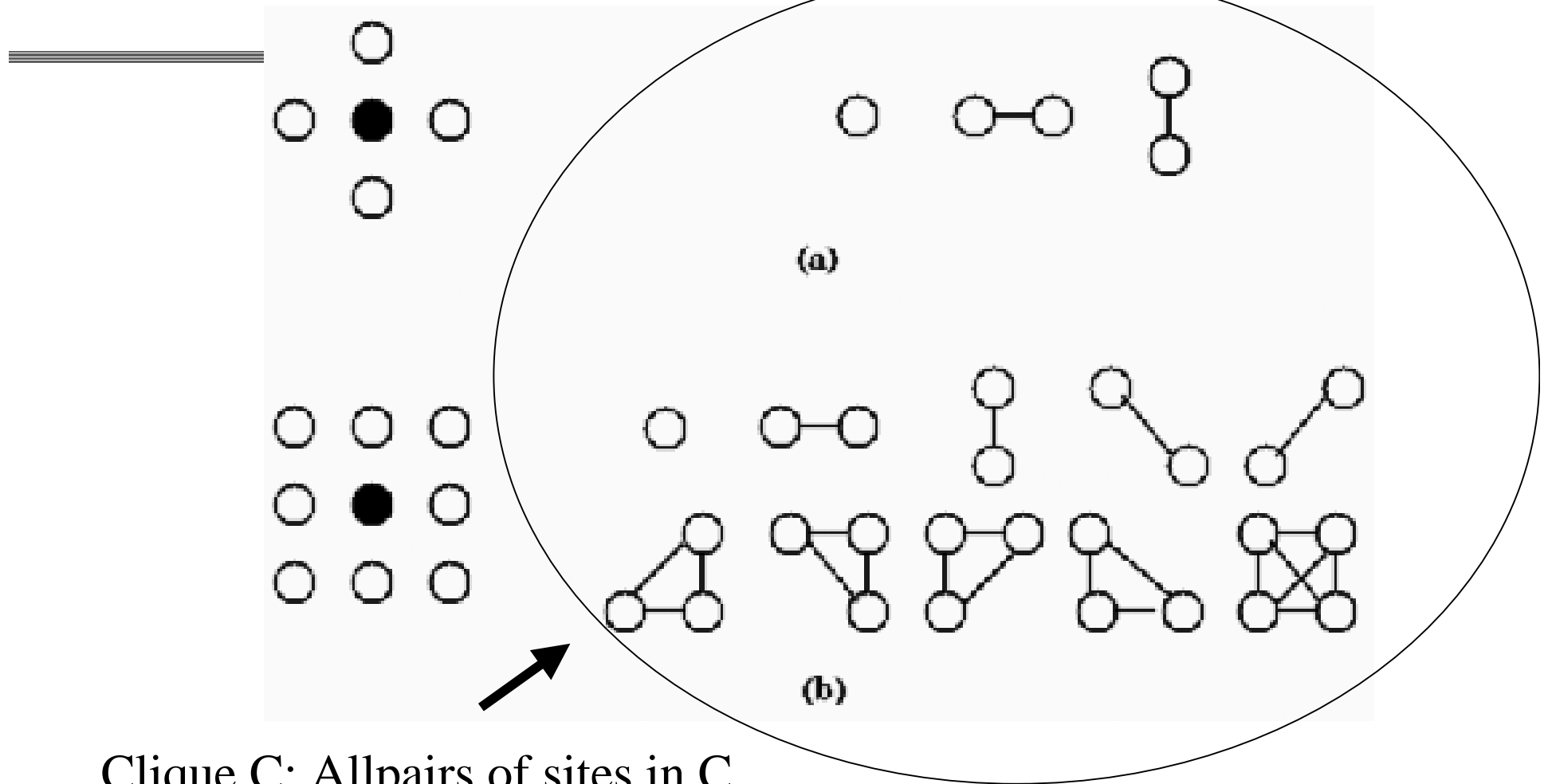
- Deviation of DFD from zero is modelled by a RP that is exponentially distributed.
- A stochastic smoothness constraint is introduced by modelling the 2-D motion vector field in terms of a Gibbs distribution

## Random fields

- A random field  $\mathbf{z} = \{ z(\mathbf{x}), \mathbf{x} \in \Lambda \}$  is defined over a lattice  $\mathbf{L}$ . Let  $\omega \in \Omega$  denote a realization of the random field  $\mathbf{z}$ .
  - The random field  $z(\mathbf{x})$  can be continuous or discrete-valued, that is  $\omega(\mathbf{x}) \in \mathbf{R}$  or  $\Gamma = \{0, 1, \dots, L-1\}$  for  $\mathbf{x} \in \Lambda$ .
  - The neighborhood  $N_{\mathbf{x}}$  of a site  $\mathbf{x}$  has the properties:
    - (i)  $\mathbf{x} \notin N_{\mathbf{x}}$ , and (ii)  $\mathbf{x}_j \in N_{\mathbf{x}_i} \leftrightarrow \mathbf{x}_i \in N_{\mathbf{x}_j}$ , where  $\mathbf{x}_i$  and  $\mathbf{x}_j$  denote two sites of the lattice.
- (In words,  $\mathbf{x}$  does not belong to its own set of neighbors, and if  $\mathbf{x}_j$  is a neighbor of  $\mathbf{x}_i$ , then  $\mathbf{x}_i$  is a neighbor of  $\mathbf{x}_j$ .)



# Examples of neighborhoods



Clique C: All pairs of sites in C are neighbors

# Markov Random Fields (MRFs)

- A random field  $\mathbf{z} = \{ z(\mathbf{x}), \mathbf{x} \in \Lambda \}$  is an MRF with respect to  $\mathbf{N}$  if

i)  $p(\mathbf{z}) > 0$  for all  $\mathbf{z}$  and

ii)  $p(z(\mathbf{x}_i) | z(\mathbf{x}_j), \text{for all } \mathbf{x}_j \neq \mathbf{x}_i) = p(z(\mathbf{x}_i) | z(\mathbf{x}_j), \mathbf{x}_j \in \mathbf{N}_{\mathbf{x}_i})$

(All realizations have non-zero pdf, and the conditional pdf at a particular site depends only on its neighborhood.)

- Difficulties with MRF models:

i) the joint pdf  $p(\mathbf{z})$  cannot be easily related to local properties, ii) it is hard to determine when a set of functions  $p(z(\mathbf{x}_i) | z(\mathbf{x}_j), \mathbf{x}_j \in \mathbf{N}_{\mathbf{x}_i}), \mathbf{x}_i \in \Lambda$  are valid conditional pdfs  
[Geman and Geman]

## Gibbs Random Fields (GRFs)

- A GRF with a neighborhood system  $N$  and the associated set of cliques  $C$  is characterized by the joint pdf

- discrete-valued 
$$p(\mathbf{z} = \omega) = \frac{1}{Q} \exp \left\{ -\frac{U(\mathbf{z} = \omega)}{T} \right\} \delta(\mathbf{z} - \omega)$$

where

$$Q = \sum_{\omega \in \Omega} \exp \left\{ -\frac{U(\mathbf{z} = \omega)}{T} \right\}$$

- continuous-valued

$$p(\mathbf{z}) = \frac{1}{Q} \exp \left\{ -\frac{U(\mathbf{z})}{T} \right\}$$

where

$$Q = \int_{\Omega} \exp \left\{ -\frac{U(\mathbf{z})}{T} \right\} d\mathbf{z}$$

and  $U(\mathbf{z})$ , the Gibbs potential (Gibbs energy) is defined by

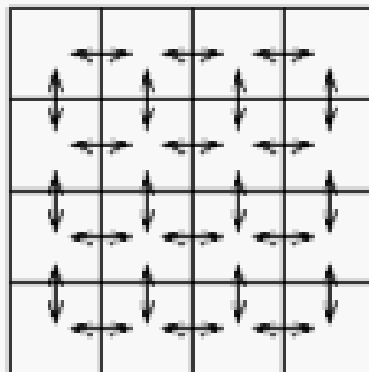
$$U(\mathbf{z}) = \sum_{c \in C} V_c(\mathbf{z}(\mathbf{x}) \mid \mathbf{x} \in c)$$

# Ex: Spatial smoothness using GRF

- Let the 2-pixel clique potential be defined as

$$V_c(z(\mathbf{x}_i), z(\mathbf{x}_j)) = \begin{cases} -\beta & \text{if } z(\mathbf{x}_i) = z(\mathbf{x}_j) \\ \beta & \text{otherwise} \end{cases}$$

where  $\beta$  is a positive number.



24 two-pixel cliques

(a)

2	2	2	2	1	2	1	2
2	2	2	2	2	1	2	1
2	2	2	2	1	2	1	2
2	2	2	2	2	1	2	1

$$V = -24\beta$$

(b)

$$V = 24\beta$$

(c)

- Over a 4 x 4 lattice, there are a total of 24 such cliques.
- A lower potential means a higher probability.

# Optimization methods

---

- Minimize  $\mathbf{E}(\mathbf{d})$  wrt  $\mathbf{d}$  where  $\mathbf{d}$  is some N-dimensional parameter vector.
- Stochastic optimization methods may find the global optimum.
  - *Simulated annealing* (stochastic relaxation)
    - Metropolis algorithm
    - Gibbs sampler (by Geman and Geman).
  - *Iterative conditional mode* (ICM) (by Besag)
  - *Mean field annealing* (MFA) (by Bilbro et al.)

# Simulated annealing

---

- Simulated annealing, also referred to as *stochastic relaxation*, belongs to *Monte Carlo* methods.
- It enables finding the global optimum of a non-convex cost function of many variables.
- Two implementations:
  - the original formulation of Metropolis
  - the Gibbs sampler proposed by Geman and Geman.
- The computational load of simulated annealing is usually significant

# Metropolis algorithm

- Choose an initial value for  $\mathbf{d}=\mathbf{d}^{(0)}$ . Set  $i=0$  and  $j=1$ .
- Perturb the  $j$ th component of  $\mathbf{d}^{(i)}$  to generate the vector  $\mathbf{d}^{(i+1)}$
- Compute  $\Delta E = E(\mathbf{d}^{(i+1)}) - E(\mathbf{d}^{(i)})$ .
- Compute  $P$  from
$$P = \begin{cases} \exp\left(-\frac{\Delta E}{T}\right) & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases}$$
- If  $P < 1$ , then draw a random number that is uniformly distributed between 0 and 1. If the number drawn is less than  $P$  accept the perturbation.
- Set  $j=j+1$ . If  $j \leq N$ , go to 2. ( $N$  = number of components of  $\mathbf{d}$ ).
- Set  $i=i+1$  and  $j=1$ . Reduce  $T$  according to a temperature schedule. If  $T > T_{\min}$ , go to 2. Otherwise terminate.

## Gibbs sampler

- Instead of making random perturbations and then deciding whether to accept or reject them, the new value is "drawn from" the distribution of  $P(\mathbf{d})$  and is always accepted.
- Compute the conditional probability of  $\mathbf{d}(\mathbf{x}_i)$  taking values from the set  $\Gamma$  given the present values of its neighbors

$$P(\mathbf{d}(\mathbf{x}_i) = \gamma \mid \mathbf{d}(\mathbf{x}_j), \mathbf{x}_i \neq \mathbf{x}_j) = Q_{\mathbf{x}_i}^{-1} \exp \left\{ -\frac{1}{T} \sum_{c \mid \mathbf{x}_i \in c} V_c(\mathbf{d}(\mathbf{x}) \mid \mathbf{x} \in c) \right\}$$

where

$$Q_{\mathbf{x}_i} = \sum_{\gamma \in \Gamma} \exp \left\{ -\frac{1}{T} \sum_{c \mid \mathbf{x}_i \in c} V_c(\mathbf{d}(\mathbf{x}) \mid \mathbf{x} \in c) \right\}$$

- The new value of  $\mathbf{d}(\mathbf{x}_i)$  is drawn from this conditional probability distribution.



## Example

- The meaning of “drawn from:” Suppose that the sample space  $\Gamma = \{ 0, 1, 2, 3 \}$ , and it was found that

$$P(\mathbf{d}(\mathbf{x}_i) = 0 \mid \mathbf{d}(\mathbf{x}_j), \mathbf{x}_i \neq \mathbf{x}_j) = 0.2 \quad P(\mathbf{d}(\mathbf{x}_i) = 2 \mid \mathbf{d}(\mathbf{x}_j), \mathbf{x}_i \neq \mathbf{x}_j) = 0.4$$

$$P(\mathbf{d}(\mathbf{x}_i) = 1 \mid \mathbf{d}(\mathbf{x}_j), \mathbf{x}_i \neq \mathbf{x}_j) = 0.1 \quad P(\mathbf{d}(\mathbf{x}_i) = 3 \mid \mathbf{d}(\mathbf{x}_j), \mathbf{x}_i \neq \mathbf{x}_j) = 0.3$$

- A uniform random number,  $R$ , between 0 and 1 is generated.
  - If  $0 \leq R < 0.2$ , then  $\mathbf{d}(\mathbf{x}_i) = 0$
  - If  $0.2 \leq R < 0.3$ , then  $\mathbf{d}(\mathbf{x}_i) = 1$
  - If  $0.3 \leq R < 0.7$ , then  $\mathbf{d}(\mathbf{x}_i) = 2$
  - If  $0.7 \leq R < 1$ , then  $\mathbf{d}(\mathbf{x}_i) = 3$
- For any initial estimate, optimization by Gibbs sampler yields an asymptotically Gibbsian distribution. This can be used to simulate a Gibbs random field with specified parameters.

# Iterated Conditional Modes (ICM)

---

- ICM aims to reduce the computational load of stochastic relaxation or Gibbs sampling.
- It is a special (deterministic) case of SA, where the temperature  $T=0$  for all iterations.
- It follows that ICM only allows perturbations with negative  $\Delta E$  which lower the cost function. Therefore, ICM is likely to get trapped in a local minimum.
- If the initial estimate is reasonable good, ICM reaches an acceptable solution in relatively few iterations.

# Bayesian motion estimation

- Let  $\mathbf{s}_k = \{ s_k(\mathbf{x}), \mathbf{x} \in \Lambda \}$  denote frame  $k$ ,  $\mathbf{d}(\mathbf{x}) = [ d_1(\mathbf{x}) \ d_2(\mathbf{x}) ]^T$  denote the displacement vector at site  $\mathbf{x}$ , and  $\mathbf{d}_1$  and  $\mathbf{d}_2$  denote the lexicographic ordering of  $x_1$  and  $x_2$  components of the displacement field, i.e.,  $s_k(\mathbf{x}) = s_{k-1}(\mathbf{x} - \mathbf{d}(\mathbf{x}))$
- The Bayesian motion estimation problem can be stated as:  
Given  $\mathbf{s}_k$  and  $\mathbf{s}_{k-1}$ , find the maximum a posteriori probability (MAP) estimates of  $\mathbf{d}_1$  and  $\mathbf{d}_2$

$$\left( \hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2 \right) = \arg \max_{\mathbf{d}_1, \mathbf{d}_2} p(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_k, \mathbf{s}_{k-1})$$

The Bayes rule states

$$p(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_k, \mathbf{s}_{k-1}) = \frac{p(\mathbf{s}_k \mid \mathbf{d}_1, \mathbf{d}_2, \mathbf{s}_{k-1}) p(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_{k-1})}{p(\mathbf{s}_k \mid \mathbf{s}_{k-1})}$$

# MAP estimation

- Since the denominator is not a function of  $\mathbf{d}_1$  and  $\mathbf{d}_2$ ,

$$\begin{pmatrix} \hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2 \end{pmatrix} = \arg \max_{\mathbf{d}_1, \mathbf{d}_2} p(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{s}_{k-1}) p(\mathbf{d}_1, \mathbf{d}_2 | \mathbf{s}_{k-1})$$

or (forward motion estimation)

$$\begin{pmatrix} \hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2 \end{pmatrix} = \arg \max_{\mathbf{d}_1, \mathbf{d}_2} p(\mathbf{s}_{k-1} | \mathbf{d}_1, \mathbf{d}_2, \mathbf{s}_k) p(\mathbf{d}_1, \mathbf{d}_2 | \mathbf{s}_k)$$

- The term  $p(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{s}_{k-1})$  is the conditional pdf, or the "likelihood (consistency) measure", that quantifies how well the estimates of  $\mathbf{d}_1, \mathbf{d}_2$  explain the observations  $\mathbf{s}_k$  given  $\mathbf{s}_{k-1}$ .
- The term  $p(\mathbf{d}_1, \mathbf{d}_2 | \mathbf{s}_{k-1})$  is the *a priori* pdf, that is modeled by a GRF, to impose *Global Spatial Smoothness* on  $\mathbf{d}_1, \mathbf{d}_2$ .

# Likelihood model

---

$$P(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{s}_k) C \exp \left\{ - \sum_{\mathbf{x} \in \Lambda} \frac{(s_k(\mathbf{x}) - s_{k-1}(\mathbf{x} - \mathbf{d}(\mathbf{x})))^2}{2\sigma^2} \right\}$$

constant



# Prior model

---

$$P(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_{k-1}) = C \exp \{ -U_d (\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_{k-1}) \}$$

$$U_d (\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{s}_{k-1}) = \lambda_d \sum_{c \in C_d} V_d^c (d_1, d_2 \mid s_{k-1})$$

# Discontinuity models

- The **occlusion field** models the occlusion/uncovered areas

$$\mathbf{o}(\mathbf{x}) = \begin{cases} 0 & \mathbf{d}(\mathbf{x}) \text{ is well defined} \\ 1 & \mathbf{x} \text{ is an occlusion pixel} \end{cases}$$

- The **line field**,  $\mathbf{l}$ , models the optical flow boundaries and has sites between every pair of pixels. The state of each line site can be ON ( $l=1$ ) or OFF ( $l=0$ ), expressing the presence and absence of a discontinuity, respectively.
- Nonnegative potentials are assigned to each

# MAP estimation with discontinuity models

---

- The MAP estimate of combined motion and discontinuity fields is given by:

$$\left( \hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2, \hat{\mathbf{o}}, \hat{\mathbf{l}} \right) = \arg \max_{\mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l}} p(\mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l} \mid \mathbf{s}_k, \mathbf{s}_{k-1})$$

- Using the Bayes rule, and the symmetry of the expression

$$\left( \hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2, \hat{\mathbf{o}}, \hat{\mathbf{l}} \right) = \arg \max_{\mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l}} p(\mathbf{s}_{k-1} \mid \mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l}, \mathbf{s}_k) p(\mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l} \mid \mathbf{s}_k)$$

- Next, we discuss the likelihood (consistency) and the *a priori* probability models.



# Likelihood model

- Assuming the change in illumination from frame to frame is insignificant and there is no occlusion, the change in intensity of a pixel along the motion trajectory is due to white Gaussian noise

$$p(\mathbf{n}_k) = C \exp \left\{ - \sum_{\mathbf{x} \in \Lambda} \frac{(s_k(\mathbf{x}) - s_{k-1}(\mathbf{x} - \mathbf{d}(\mathbf{x})))^2}{2\sigma^2} \right\}$$

where  $C$  is some constant.

- Taking the occlusion points into account, we have

$$p(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l}, \mathbf{s}_{k-1}) = C \exp \{ - U_s(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l}, \mathbf{s}_{k-1}) \}$$

which is compactly expressed in terms of the “energy function”

$$U_s(\mathbf{s}_k | \mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l}, \mathbf{s}_{k-1}) = \sum_{\mathbf{x} \in \Lambda} \frac{(1 - o(\mathbf{x}))(s_k(\mathbf{x}) - s_{k-1}(\mathbf{x} - \mathbf{d}(\mathbf{x})))^2}{2\sigma^2}$$

## The prior model

- The prior model incorporates the location of optical flow boundaries and occlusion areas while dictating that the flow vectors vary smoothly within each flow boundary.
- The *a priori* model can be expressed as

where 
$$p(\mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l} \mid \mathbf{s}_{k-1}) = C \exp\{-U(\mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l} \mid \mathbf{s}_{k-1})\}$$

$$\begin{aligned} U(\mathbf{d}_1, \mathbf{d}_2, \mathbf{o}, \mathbf{l} \mid \mathbf{s}_{k-1}) &= \lambda_d U_d(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{l}) + \lambda_o U_o(\mathbf{o} \mid \mathbf{l}) + \lambda_l U_l(\mathbf{l}) \\ &= \lambda_d \sum_{c \in C_d} V_c(\mathbf{d}_1, \mathbf{d}_2 \mid \mathbf{l}) + \lambda_o \sum_{c \in C_o} V_c(\mathbf{o} \mid \mathbf{l}) + \lambda_l \sum_{c \in C_l} V_c(\mathbf{l}) \end{aligned}$$

and  $C_d$ ,  $C_o$  and  $C_l$  denote sets of all cliques for the displacement, occlusion and line fields, respectively,  $V_c(\cdot)$  is the corresponding clique potential function, and  $\lambda_d$ ,  $\lambda_o$  and  $\lambda_l$  are positive constants.

## **2-D motion tracking**

---

- Point tracking
- Boundary (contour) tracking
- Region tracking
- Temporal motion modeling

## Feature point selection

- Cornerness

$$C(\mathbf{x}_n) = \left| I_{x_1}(\mathbf{x}_n) \right|^2 + \left| I_{x_2}(\mathbf{x}_n) \right|^2 + \Gamma(\mathbf{x}_n)$$

- Texturedness
- Eigenvalues

$$\begin{bmatrix} \sum I_{x_1}^2 & \sum I_{x_1 x_2} \\ \sum I_{x_1 x_2} & \sum I_{x_2}^2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} -\sum I_{x_1 f} \\ -\sum I_{x_2 f} \end{bmatrix} \quad \min(\lambda_1, \lambda_2) > \lambda$$

- Closed-loop motion verification feedback

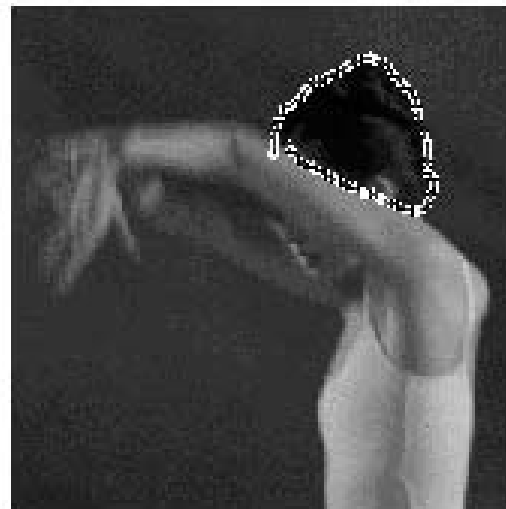
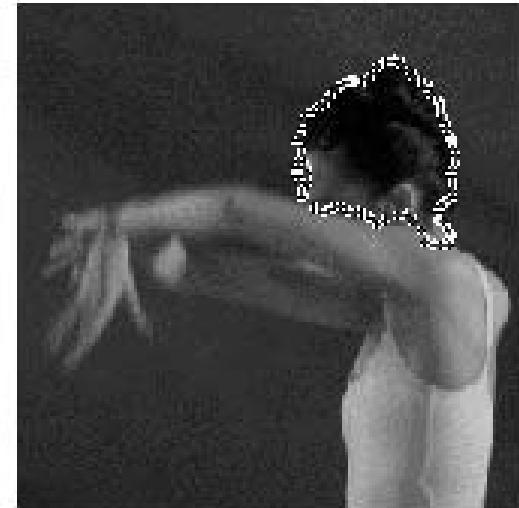
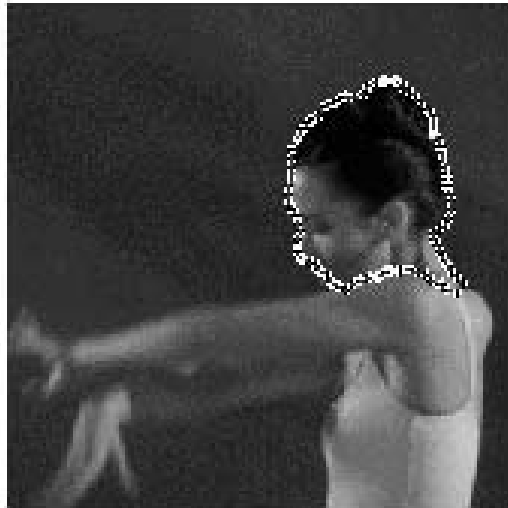
## **Boundary tracking (1/3)**

---

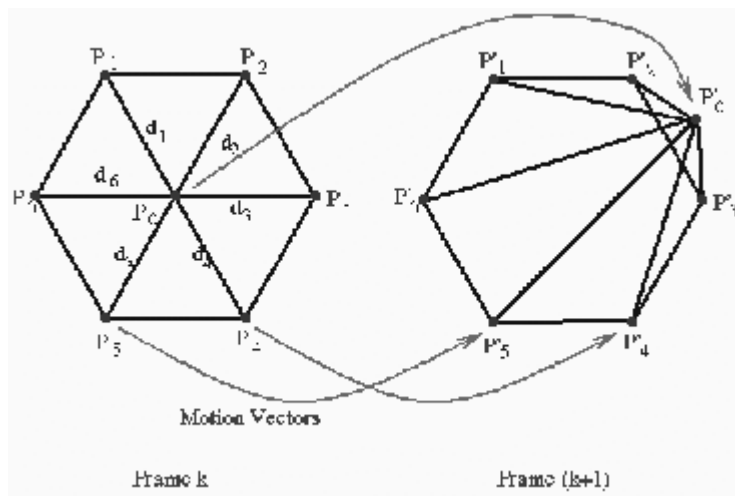
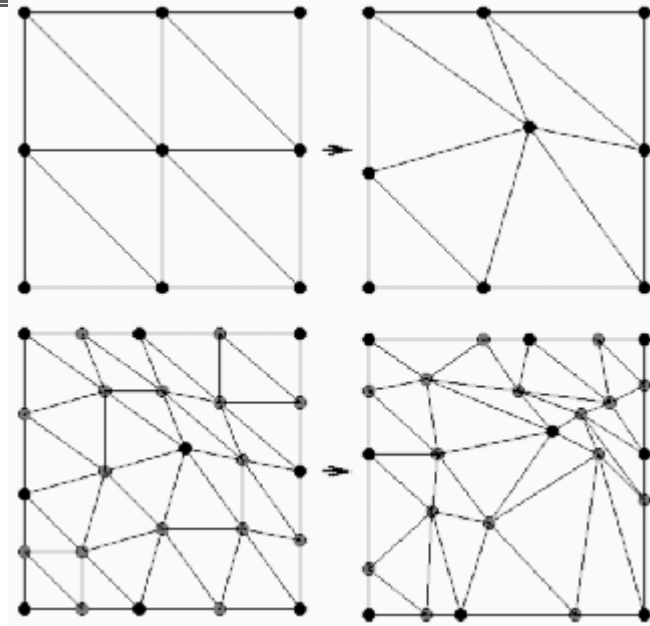
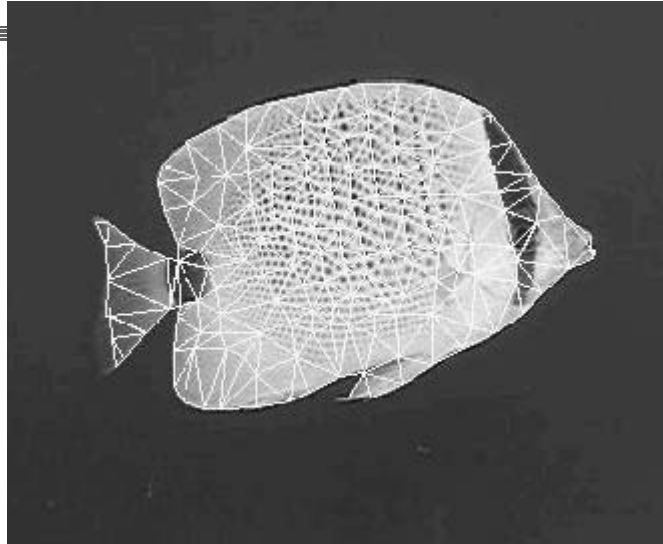
- Polygon approximation (propagate corner points)
- Spline (snake) approximation (propagate control points)

## Boundary tracking (2/3)

---



# Boundary tracking (3/3)



# Temporal modeling – Batch vs recursive estimators

---

- Batch estimators : Process the entire data record at once
- Recursive estimators: Process each observation as it becomes available to update the motion parameters  
Kalman filter



## **3-D motion estimation**

---

- Methods requiring point correspondence
  - Based on orthographic projection
  - Based on perspective projection
- Methods not requiring point correspondences
  - Optical flow based methods
  - Direct methods

# Orthographic projection

---

$$\begin{bmatrix} X_1' \\ X_2' \\ X_3' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

$$x_1' = r_{11}x_1 + r_{12}x_2 + (r_{13}X_3 + T_1)$$

$$x_2' = r_{21}x_1 + r_{22}x_2 + (r_{23}X_3 + T_2)$$

# Perspective projection

---

$$x_1' = \frac{r_{11}x_1 + r_{12}x_2 + r_{13} + \frac{T_1}{X_3}}{r_{31}x_1 + r_{32}x_2 + r_{33} + \frac{T_3}{X_3}}$$

$$x_2' = \frac{r_{21}x_1 + r_{22}x_2 + r_{23} + \frac{T_2}{X_3}}{r_{31}x_1 + r_{32}x_2 + r_{33} + \frac{T_3}{X_3}}$$

# Methods based on point correspondences: Iteration

---

$$x_1' = r_{11}x_1 + r_{12}x_2 + (r_{13}X_3 + T_1)$$

$$x_2' = r_{21}x_1 + r_{22}x_2 + (r_{23}X_3 + T_2)$$

$$x_1' = x_1 - \Delta\Phi x_2 + (\Delta\Psi X_3 + T_1)$$

$$x_2' = \Delta\Phi x_1 + x_2 + (\Delta\Theta X_3 + T_2)$$

Find  $X_3$

Find rot.  
and trans.  
param.

$$\begin{bmatrix} x_1' - x_1 \\ x_2' - x_2 \end{bmatrix} = \begin{bmatrix} -x_2 & 0 & X_3 & 1 & 0 \\ x_1 & -X_3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta\Phi \\ \Delta\Theta \\ \Delta\Psi \\ T_1 \\ T_2 \end{bmatrix}$$

# Methods based on optical flow

---

$$u = V_1 + \Omega_2 X_3 - \Omega_3 x_2$$

$$v = V_2 + \Omega_3 X_1 - \Omega_1 X_3$$

$$u = f\left(\frac{V_1}{X_3} + \Omega_2\right) - \frac{V_3}{X_3} x_1 - \Omega_3 x_2 - \frac{\Omega_1}{f} x_1 x_2 + \frac{\Omega_2}{f} x_1^2$$

$$v = f\left(\frac{V_2}{X_3} - \Omega_1\right) + \Omega_3 x_1 - \frac{V_3}{X_3} x_2 + \frac{\Omega_2}{f} x_1 x_2 - \frac{\Omega_1}{f} x_2^2$$

# Estimation

---

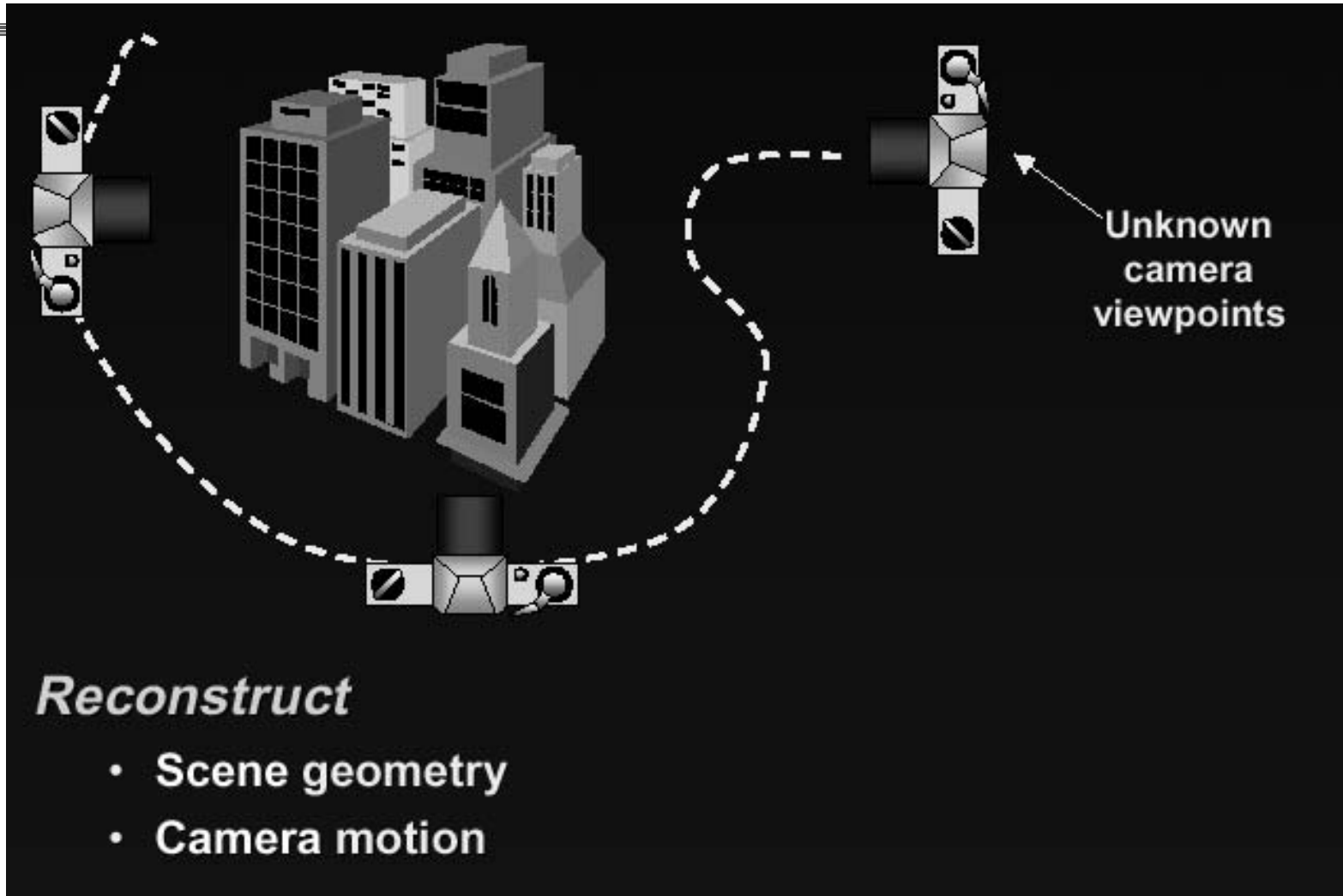
- Eliminate  $Z$  using equations for  $u$  and  $v$
- Define  $e_1 = V_1/V_3$ ,  $e_2 = V_2/V_3$
- Obtain the linear equation

$$\begin{bmatrix} -vu & -x_1 & -x_2 & -x_1x_2(x_1^2 + x_2^2)(1+x_2^2)(1+x_1^2) \end{bmatrix} H = ux_2 - vx_1$$

$$H = \begin{bmatrix} e_1 & e_2 & \Omega_1 + \Omega_3 e_1 & \Omega_2 + \Omega_3 e_2 & \Omega_2 e_1 + \Omega_1 e_2 & \Omega_3 & \Omega_1 e_1 & \Omega_2 e_2 \end{bmatrix}^T$$

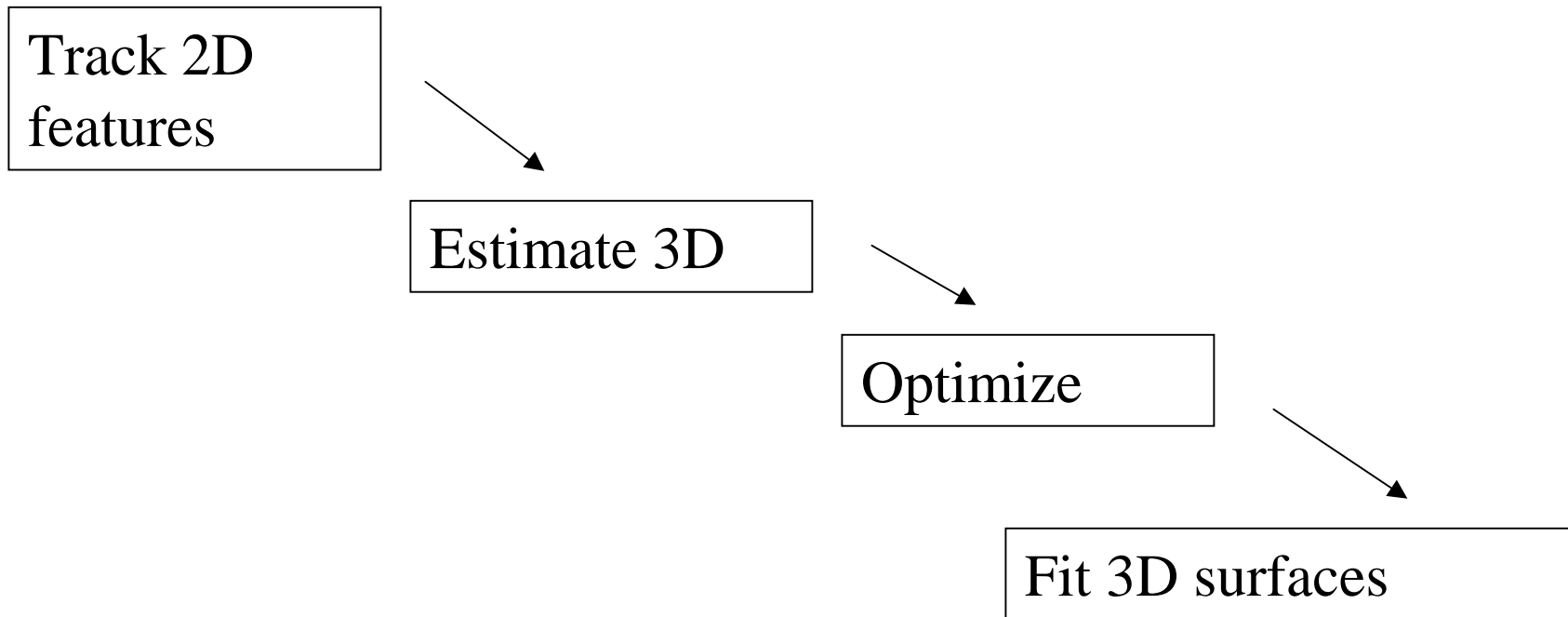
- Solve for 5 motion param. Using minimum of 8 image points
- Estimate depth param from eqns of  $u$  or  $v$ .

# Structure from motion (1/2)



# Structure from motion (2/2)

---





# Feature tracking

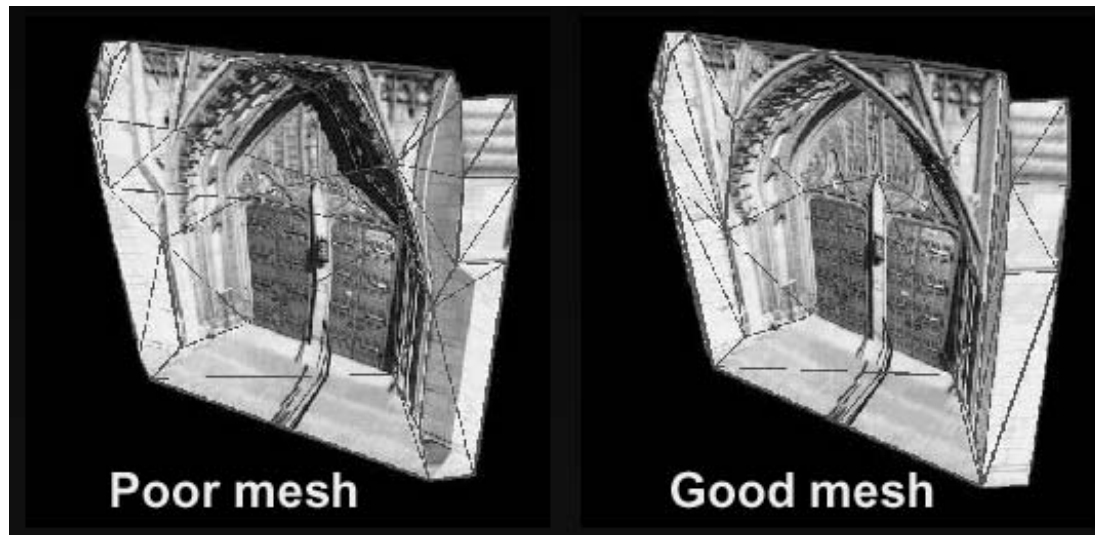


Track good features

Find correspondences by correlation

# Surface fitting

---



# Example

---

