Lecture 3
Time varying image analysis (cont.)

- 2-D Motion Estimation
  - Phase correlation
  - Per-recursive and Bayesian methods
  - Motion tracking
- 3-D motion
  - Optical flow and direct methods
  - Structure from motion

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

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

Phase correlation method (2/4)

- Given the motion model

  $$S_{k+1}(f_1, f_2) = S_k(f_1, f_2)\exp\{-j2\pi(f_1d_1 + f_2d_2)\}$$

  we have

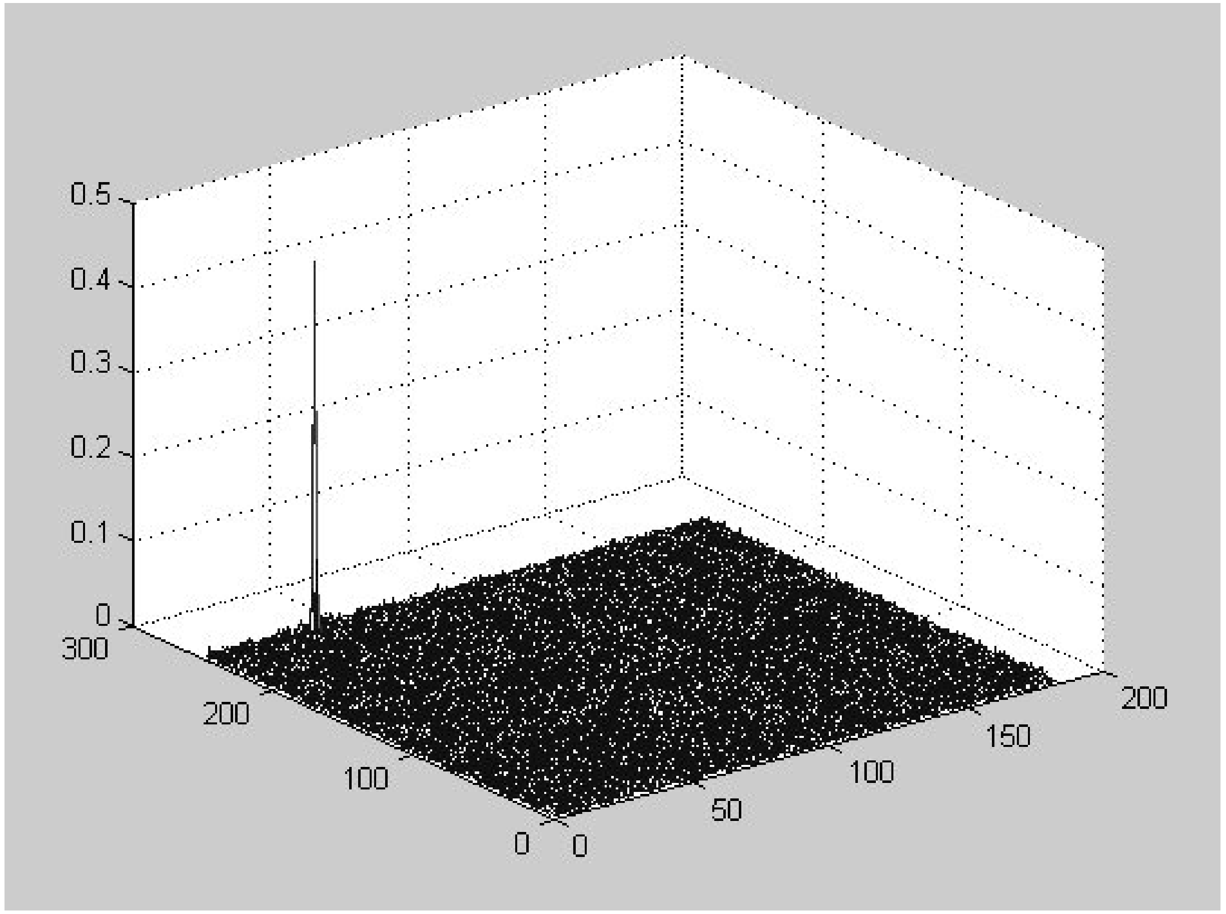
  $$\tilde{C}_{k,k+1}(f_1, f_2) = \exp\{-j2\pi(f_1d_1 + f_2d_2)\}$$

  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+Δt

$$dfd(\mathbf{x},\hat{\mathbf{d}}) = \underset{\underset{\text{Intensity}}{\uparrow}}{I}(\mathbf{x}+\underset{\underset{\text{displacement}}{\nwarrow}}{\mathbf{d(x)}}, t+\Delta t) - I(\mathbf{x},\mathbf{t})$$

$\mathbf{u}^{i}(\mathbf{x})$ is estimated by minimizing at each **x** a positive definite function **E** of the DFD wrt **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 ∇_d is the gradient operator with respect to d, the set of partial derivatives.

- The following equation must be solved simultaneously:

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

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

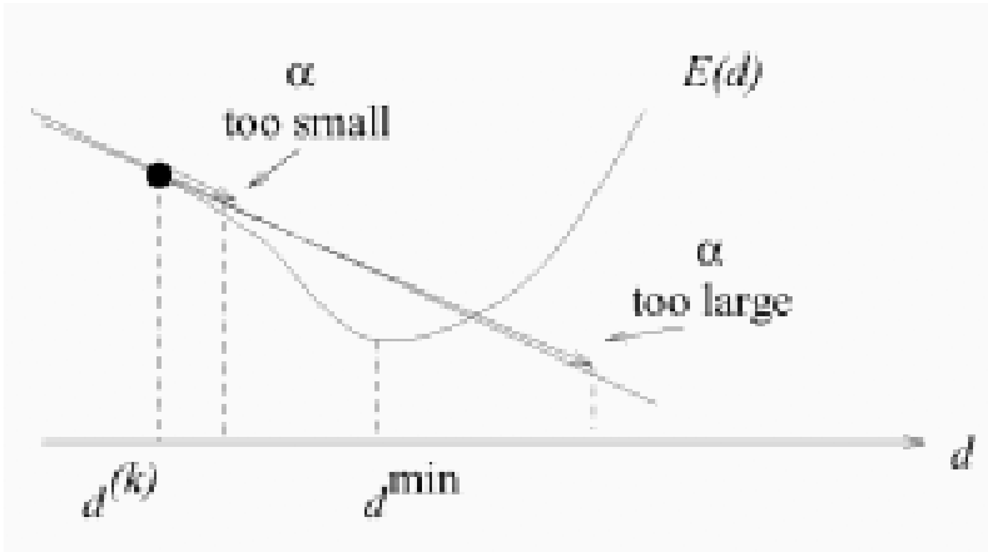
- Since an analytical solution to these equations cannot be found in general, we resort to iterative methods.

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 α is some positive scalar, known as the step size.



- If α 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 α 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 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

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

A steepest descent approach to the minimization problem yields the iteration

$$\hat{\mathbf{d}}^{i+1}(\mathbf{x}) = \hat{\mathbf{d}}^{i}(\mathbf{x}) - (1/2)\varepsilon\nabla_{\mathbf{d}}\left[dfd(\mathbf{x},\mathbf{d})\big|_{\mathbf{d}=\hat{\mathbf{d}}_i}\right]^2$$

$$= \hat{\mathbf{d}}^{i}(\mathbf{x}) - \varepsilon\, dfd(\mathbf{x},\hat{\mathbf{d}}^{i})\nabla_{\mathbf{d}} dfd(\mathbf{x},\mathbf{d})\big|_{\mathbf{d}=\hat{\mathbf{d}}_i},$$

where ∇ is the gradient with respect to 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 $\nabla_{\mathbf{d}} dfd(\mathbf{x},\mathbf{d})\big|_{\mathbf{d}=\hat{\mathbf{d}}_i} = \nabla_{\mathbf{x}} s_c(\mathbf{x}-\mathbf{d}^{i}, t-\Delta t)$

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 η^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 representation of images, for improved motion estimation.

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

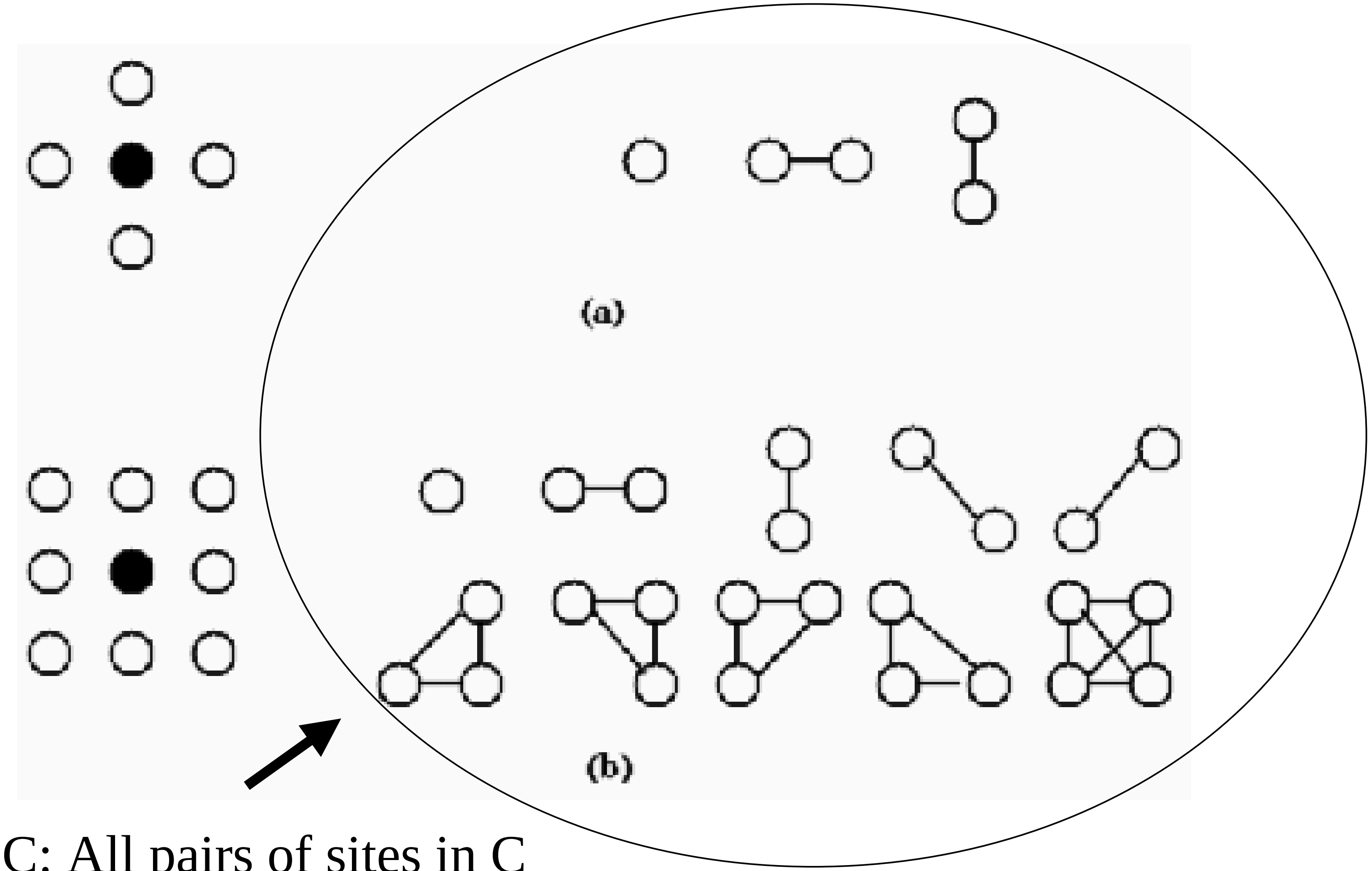
- A random field $\mathbf{z} = \{ z(\mathbf{x}), \mathbf{x} \in \mathbf{\Lambda} \}$ is defined over a lattice **L**. Let $\boldsymbol{\omega} \in \boldsymbol{\Omega}$ denote a realization of the random field **z**.
- The random field $z(\mathbf{x})$ can be continuous or discrete-valued, that is $\boldsymbol{\omega}(\mathbf{x}) \in \mathbf{R}$ or $\boldsymbol{\Gamma} = \{0,1, \ldots, L-1\}$ for $\mathbf{x} \in \mathbf{\Lambda}$.
- The neighborhood $N_{\mathbf{x}}$ of a site **x** has the properties:

(i) $\mathbf{x} \notin \boldsymbol{N}_{\mathbf{x}}$, and (ii) $\mathbf{x}_j \in \boldsymbol{N}_{\mathbf{x}\,i} \leftrightarrow \mathbf{x}_i \in \boldsymbol{N}_{\mathbf{x}\,j}$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ denote two sites of the lattice.

(In words, **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 $\mathbf{x}_j$)

- A neighborhood system over $\mathbf{\Lambda}$ is defined as $N = \{\boldsymbol{N}_{\mathbf{x}} , \mathbf{x} \in \mathbf{\Lambda} \}$

Examples of neighborhoods



Clique C: All pairs of sites in C are neighbors

Markov Random Fields (MRFs)

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

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

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

(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 $p(z(\mathbf{x}_i) \mid z(\mathbf{x}_j), \mathbf{x}_j \in \boldsymbol{N}_{\mathbf{x}\,i}), \mathbf{x}_j \in \mathbf{\Lambda}$ 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

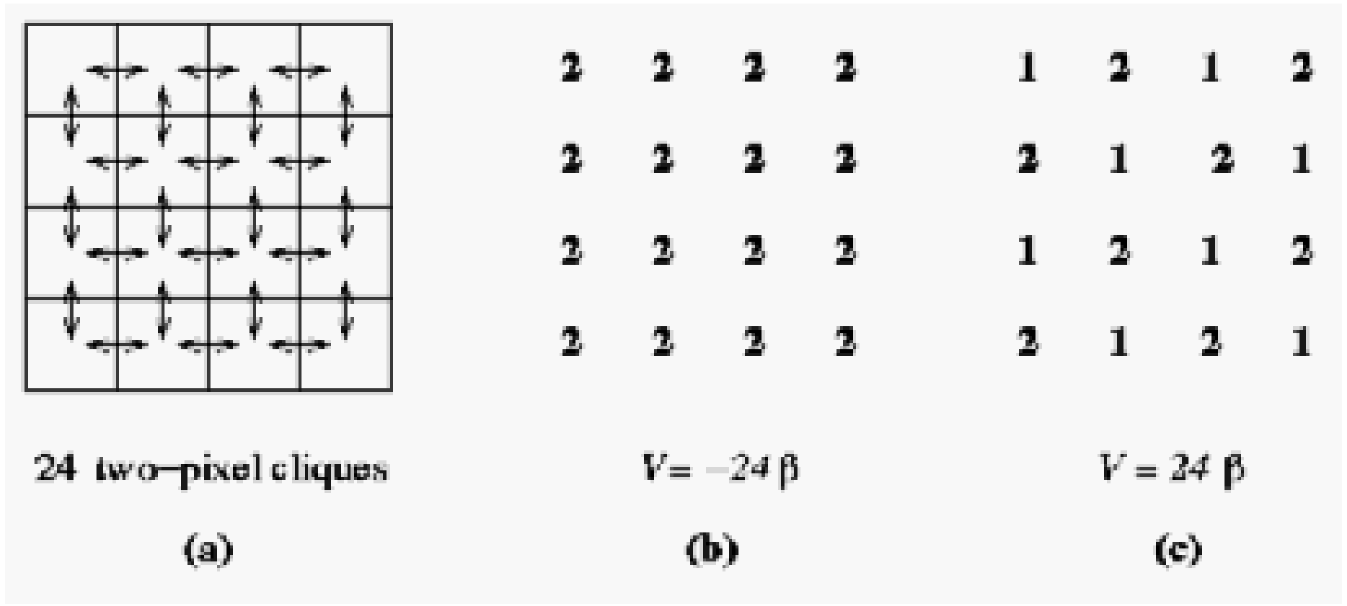
$$U(\mathbf{z})=\sum_{c\in\mathbf{C}}V_c(\mathbf{z}(\mathbf{x})\mid\mathbf{x}\in c)$$

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 β is a positive number.



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

Optimization methods

- Minimize **E(d)** wrt **d** where **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 refered 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 $\mathbf{d}$=d$^{(0)}$. Set $i=0$ and $j=1$.
- Perturb the jth 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 $$P = \begin{cases} \exp\left(-\dfrac{\Delta E}{T}\right) & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases}$$
- 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(**d**) and is always accepted.
- Compute the conditional probability of d($\mathbf{x}_i$) taking values from the set Γ given the present values of its neighbors

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

where

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

- The new value of 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

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

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

- A uniform random number, R, between 0 and 1 is generated.

If $0 \le R < 0.2$, then $\mathbf{d}(\mathbf{x}_i) = 0$ If $0.3 \le R < 0.7$, then $\mathbf{d}(\mathbf{x}_i) = 2$

If $0.2 \le R < 0.3$, then $\mathbf{d}(\mathbf{x}_i) = 1$ If $0.7 \le R < 1$, 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 ΔE 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 $\mathbf{d}_2$,

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

  or (forward motion estimation)

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

- 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

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

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**, **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 occlusion and rotation invariant line clique configuration to penalize excessive use of the ``ON'' state.

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

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

where C is some constant.

- Taking the occlusion points into account, we have

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

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

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

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

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

where

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

and $\mathbf{C}_d$, $\mathbf{C}_o$ and $\mathbf{C}_l$ denote sets of all cliques for the displacement, occlusion and line fields, respectively, $V_c(.)$ is the corresponding clique potential function, and λ_d, λ_o and λ_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

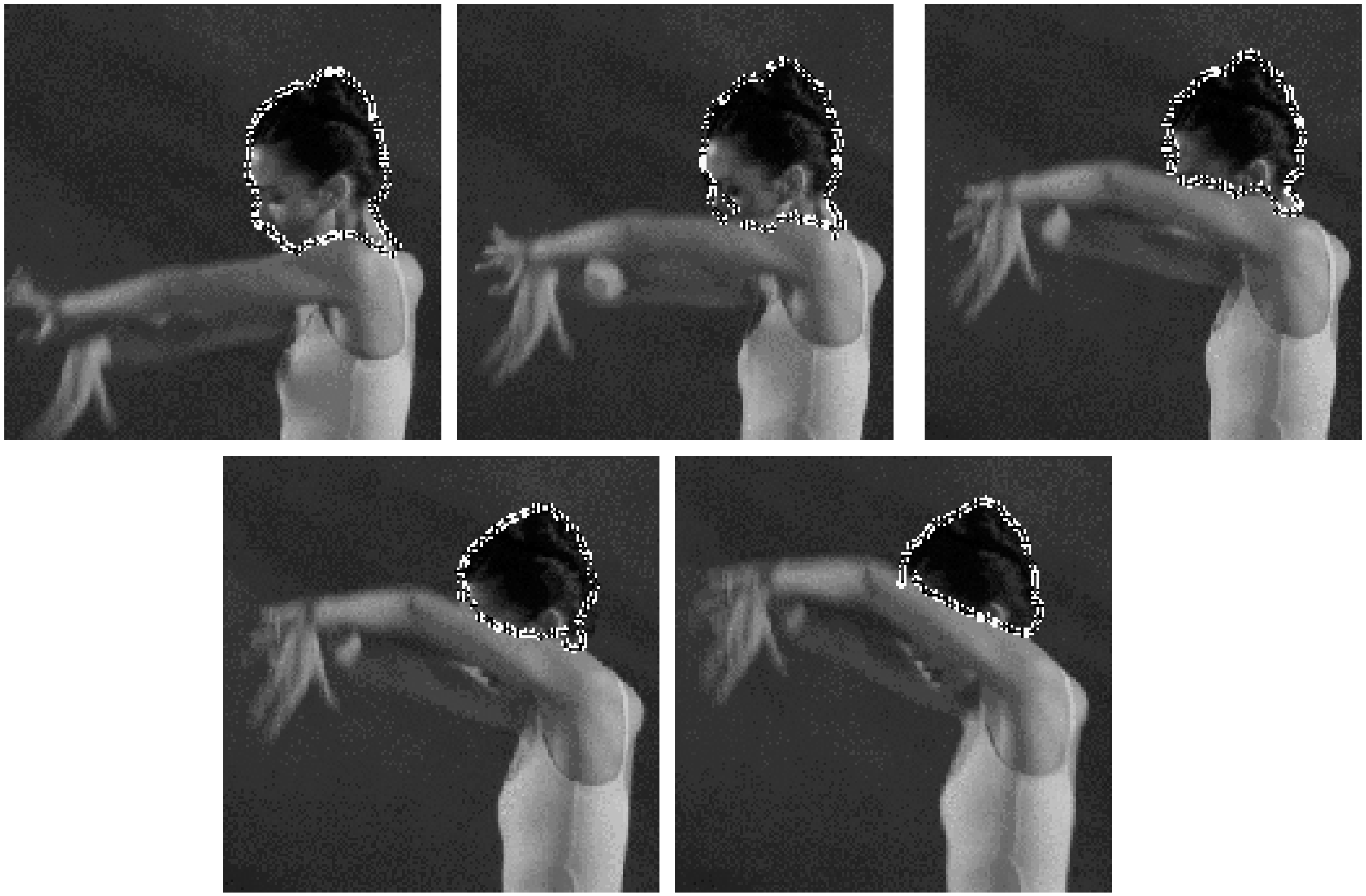
$$\begin{bmatrix} \sum I_{x_1}^2 & \sum I_{x_1 x_2} \\ \sum I_{x_1 x_2} & \sum I_{x_2}^2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} -\sum I_{x_1 t} \\ -\sum I_{x_2 t} \end{bmatrix} \qquad \min(\lambda_1, \lambda_2) > \lambda$$

- Closed-loop motion verification feedback

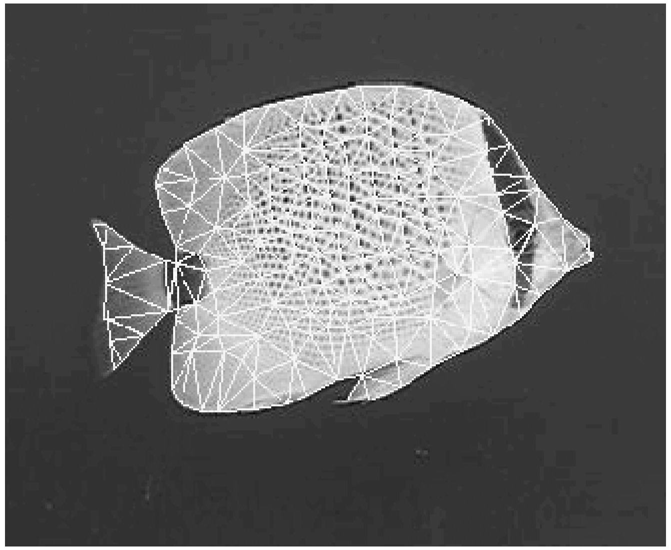
Boundary tracking (1/3)

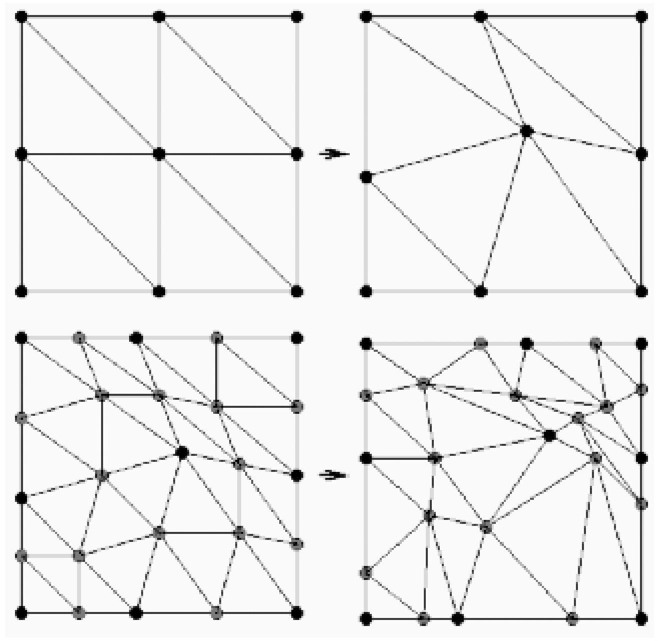
- Polygon approximation (propagate corner points)
- Spline (snake) approximation (propagate control points)

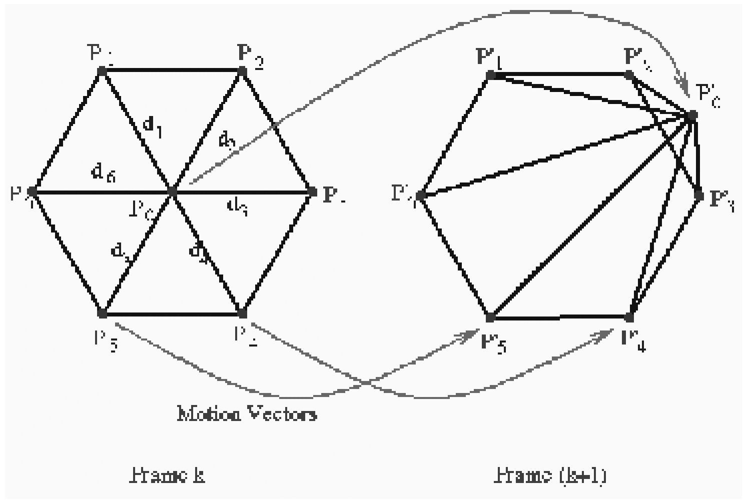
Boundary tracking (2/3)



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 $\rightarrow$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. $\longrightarrow$

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

Methods based on optical flow

$$u = V_1 + \Omega_2 X_3 - \Omega_3 x_2$$
$$v = V_2 + \Omega_3 X_1 - \Omega_1 X_3$$

$$u = f(\frac{V_1}{X_3} + \Omega_2) - \frac{V_3}{X_3} x_1 - \Omega_3 x_2 - \frac{\Omega_1}{f} x_1 x_2 + \frac{\Omega_2}{f} x_1{}^2$$
$$v = f(\frac{V2}{X_3} - \Omega_1) + \Omega_3 x_1 - \frac{V_3}{X_3} x_2 + \frac{\Omega_2}{f} x_1 x_2 - \frac{\Omega_1}{f} x_2{}^2$$

Estimation

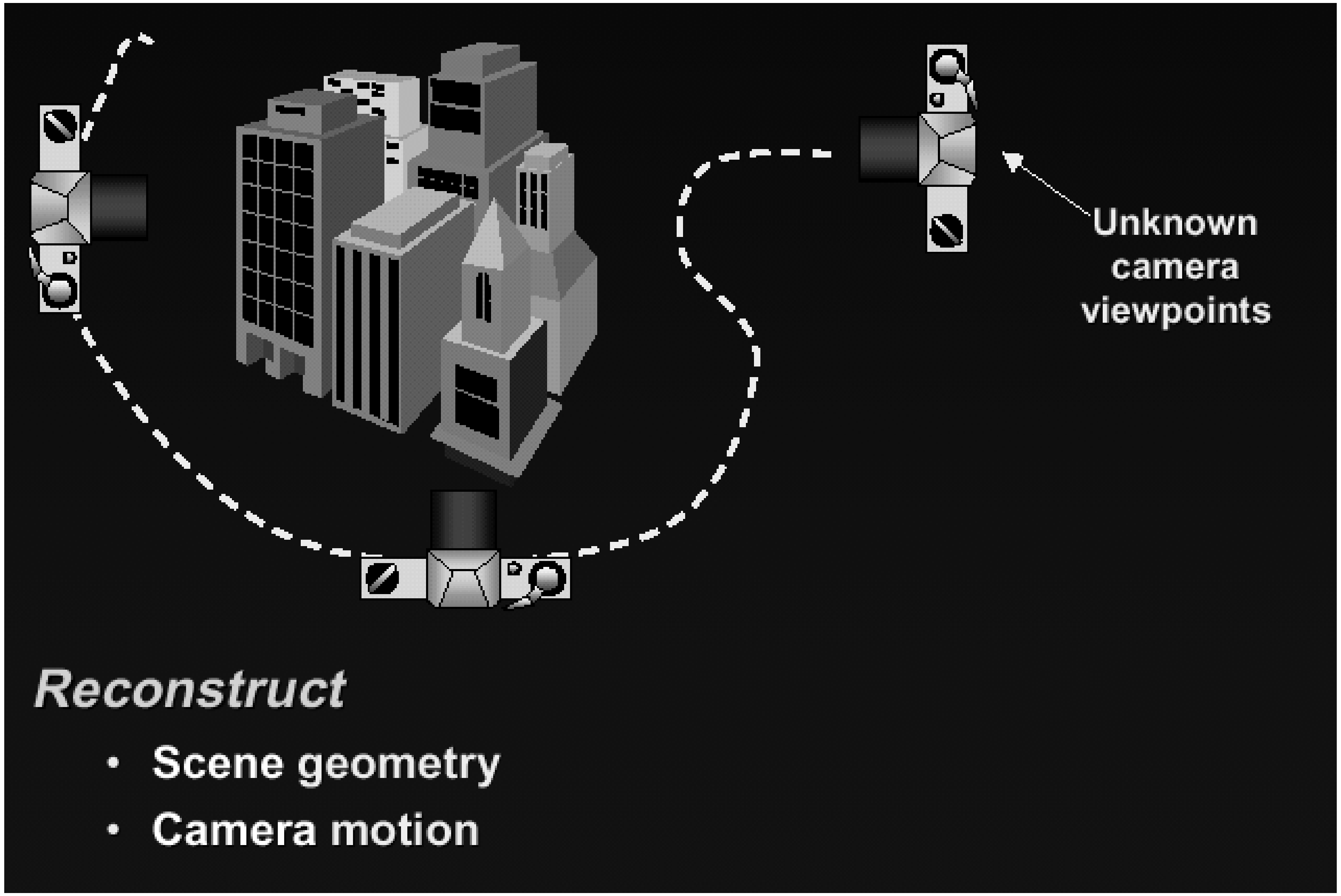
- Eliminate Z using equations for u and v
- Define $e_1 = V_1/V_3$, $e_2 = V_2/V_3$
- Obtain the linear equation

$$\left[-vu \quad -x_1 \quad -x_2 \quad -x_1x_2\,(x_1^{\;2}+x_2^{\;2})(1+x_2^{\;2})(1+x_1^{\;2})\right]\mathrm{H} = ux_2 - vx_1$$

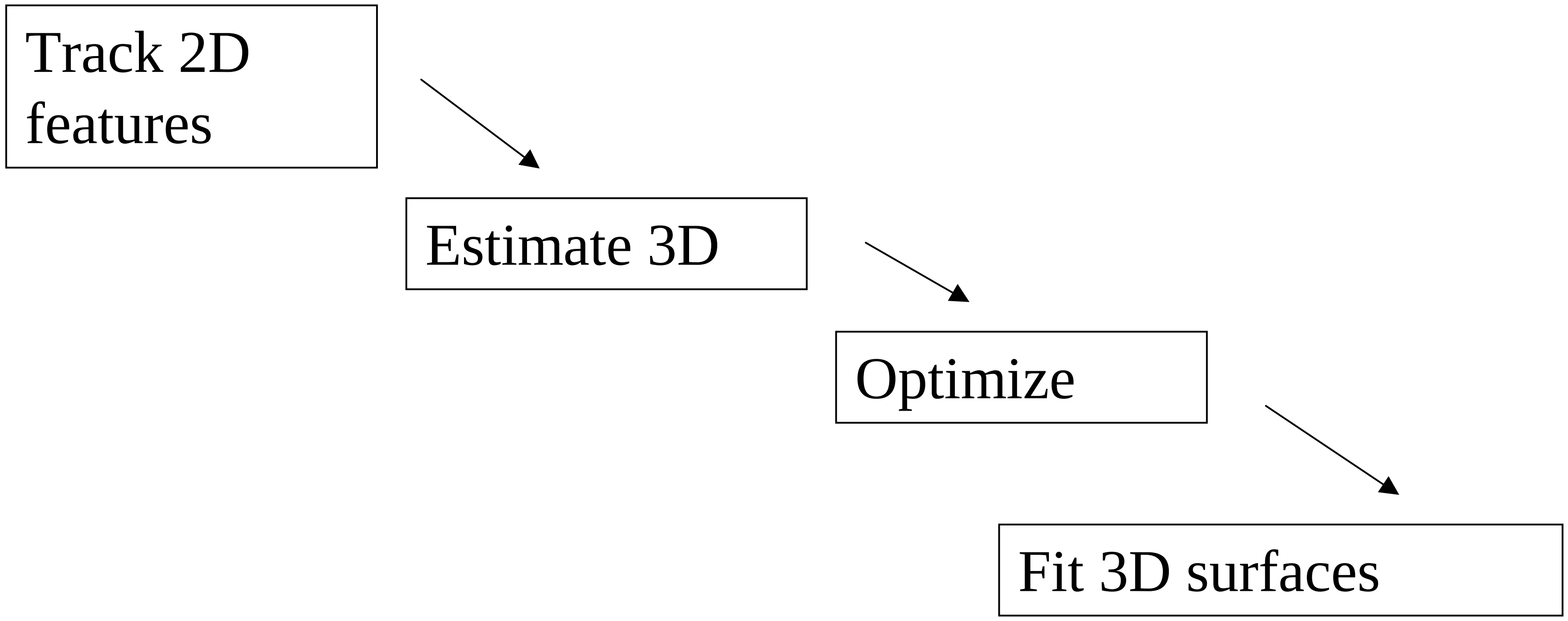
$$\mathrm{H} = \left[e_1 \quad e_2 \quad \Omega_1+\Omega_3e_1 \quad \Omega_2+\Omega_3e_2 \quad \Omega_2e_1+\Omega_1e_2 \quad \Omega_3 \quad \Omega_1e_1 \quad \Omega_2e_2\right]^T$$

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

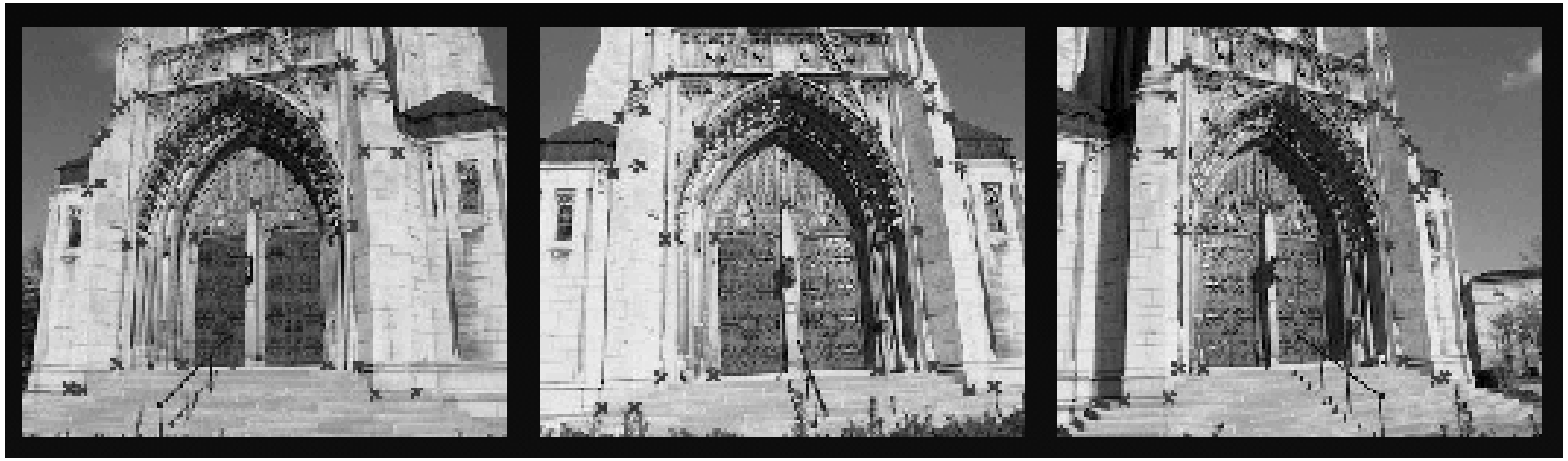
Structure from motion (1/2)



Structure from motion (2/2)



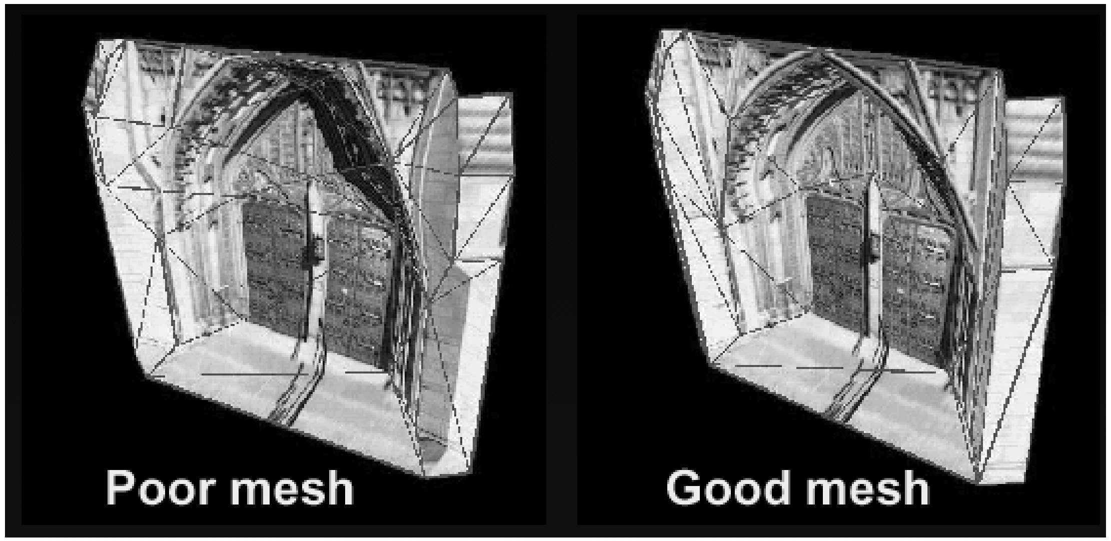
Feature tracking



Track good features

Find correspondences by correlation

Surface fitting



Example