Motion estimation

Many slides from Szeliski

Why estimate visual motion?

Visual Motion can be annoying

- Camera instabilities, jitter
- Measure it; remove it (stabilize)

Visual Motion indicates dynamics in the scene

- Moving objects, behavior
- Track objects and analyze trajectories

Visual Motion reveals spatial layout

- Motion parallax

Today's lecture

Motion estimation

- image warping (skip: see previous lecture)
- patch-based motion (optic flow)
- parametric (global) motion
- application: image morphing
- advanced: layered motion models

Readings

- Szeliski, R. *CVAA*
  - Ch. 8.1, 8.2, 4.4
- Bergen *et al. Hierarchical model-based motion estimation*. ECCV' 92, pp. 237–252.
- Shi, J. and Tomasi, C. (1994). Good features to track. In CVPR' 94, pp. 593–600.
- Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. IJCV, 56(3), 221–255.

Patch-based motion estimation

Classes of Techniques

Feature-based methods

- Extract visual features (corners, textured areas) and track them
- Sparse motion fields, but possibly robust tracking
- Suitable especially when image motion is large (10s of pixels)

Direct-methods

- Directly recover image motion from spatio-temporal image brightness variations
- Global motion parameters directly recovered without an intermediate feature motion calculation
- Dense motion fields, but more sensitive to appearance variations
- Suitable for video and when image motion is small (< 10 pixels)

Patch matching (revisited)

How do we determine correspondences?

- *block matching* or *SSD* (sum squared differences)

$$E(x, y; d) = \sum_{(x',y') \in N(x,y)} [I_L(x'+d, y') - I_R(x', y')]^2$$

The Brightness Constraint

Brightness Constancy Equation:

$$J(x,y) \approx I(x+u(x,y), y+v(x,y))$$

Or, equivalently, minimize :

$$E(u,v) = (J(x,y) - I(x+u, y+v))^2$$

Linearizing (assuming small *(u,v)*)
using Taylor series expansion:

$$J(x,y) \approx I(x,y) + I_x(x,y) \cdot u(x,y) + I_y(x,y) \cdot v(x,y)$$

Gradient Constraint (or the Optical Flow Constraint)

$$E(u,v) = (I_x \cdot u + I_y \cdot v + I_t)^2$$

Minimizing:

$$\frac{\partial E}{\partial u} = \frac{\partial E}{\partial v} = 0$$

$$I_x(I_x u + I_y v + I_t) = 0$$

$$I_y(I_x u + I_y v + I_t) = 0$$

In general $I_x, I_y \neq 0$

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0$

Patch Translation [Lucas-Kanade]

Assume a single velocity for all pixels within an image patch

$$E(u,v) = \sum_{x,y\in\Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t\right)^2$$

Minimizing

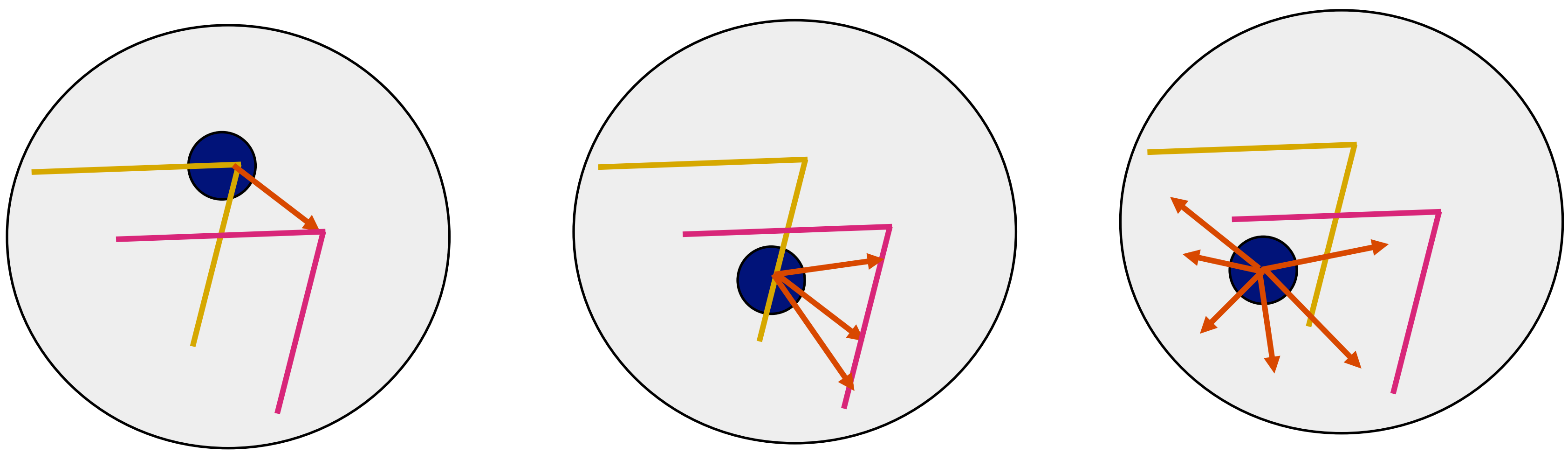
$$\begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

$$\left(\sum \nabla I (\nabla I)^T\right) \vec{U} = -\sum \nabla I I_t$$

LHS: sum of the 2x2 outer product of the gradient vector

Local Patch Analysis

How *certain* are the motion estimates?



The Aperture Problem

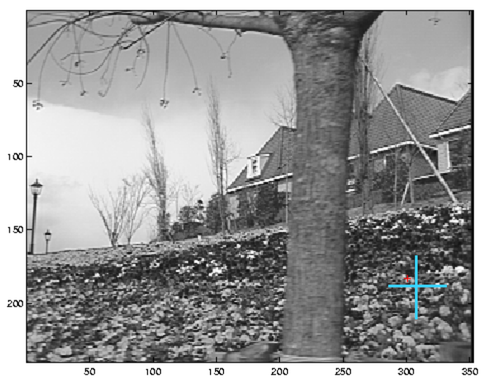
Let $M = \sum \nabla I (\nabla I)^T$ and $\vec{b} = -\sum \nabla I I_t$

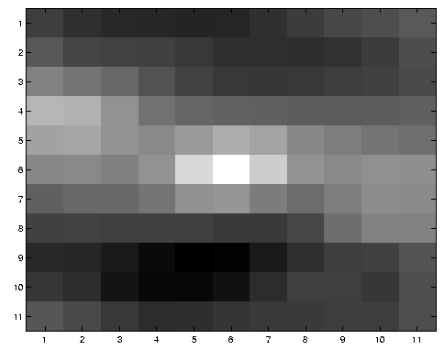
- Algorithm: At each pixel compute $\vec{U}$ by solving $M\vec{U} = \vec{b}$

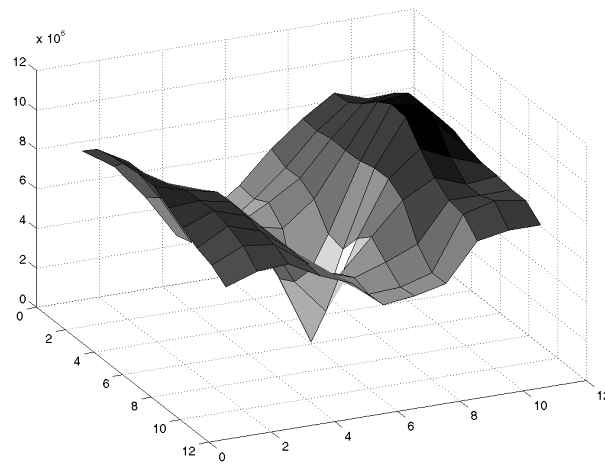
- M is singular if all gradient vectors point in the same direction
  - e.g., along an edge
  - of course, trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only *normal flow* is available (aperture problem)

- Corners and textured areas are OK

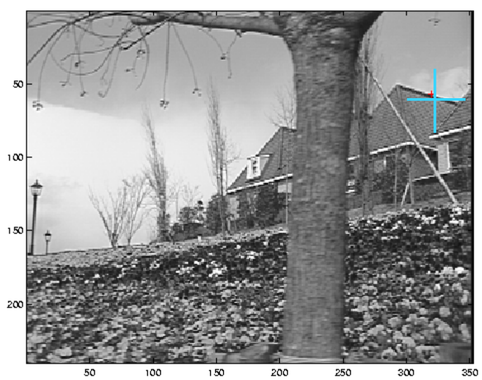
SSD Surface – Textured area

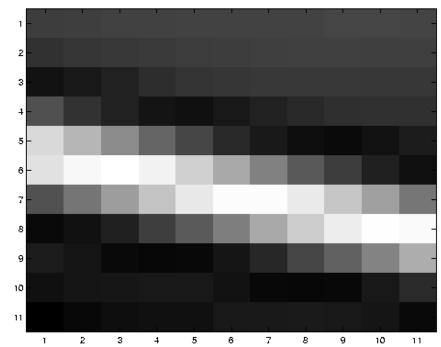


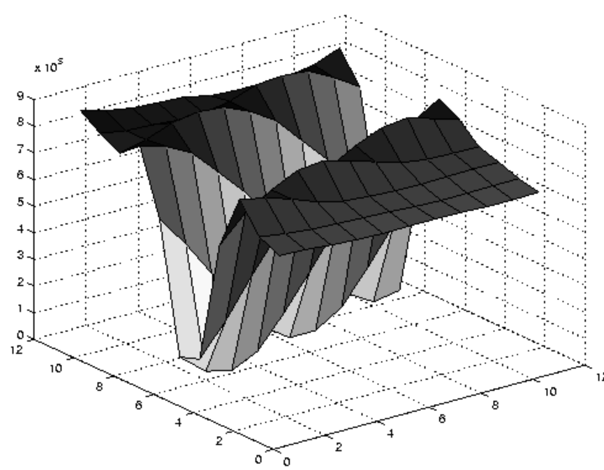




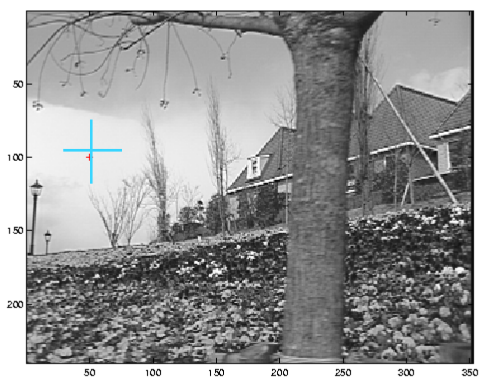
SSD Surface -- Edge

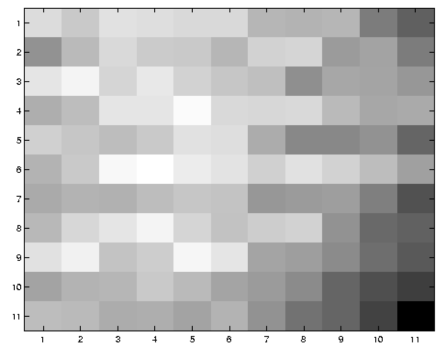


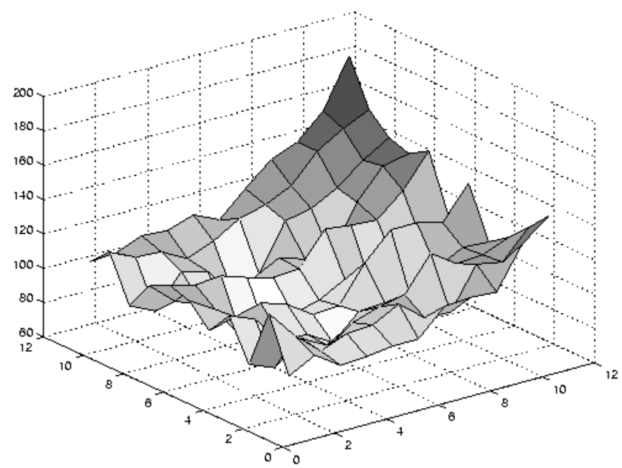




SSD – homogeneous area







Iterative Refinement

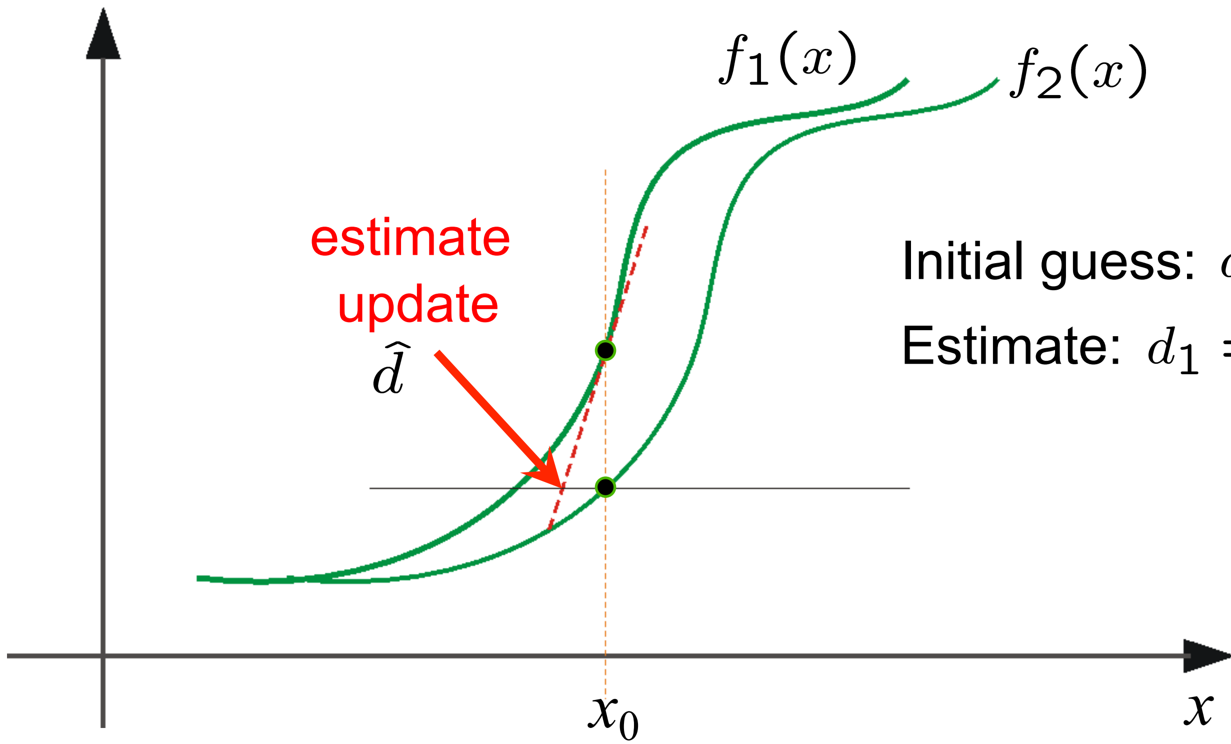
Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation

Warp one image toward the other using the estimated flow field

(easier said than done)

Refine estimate by repeating the process

Optical Flow: Iterative Estimation

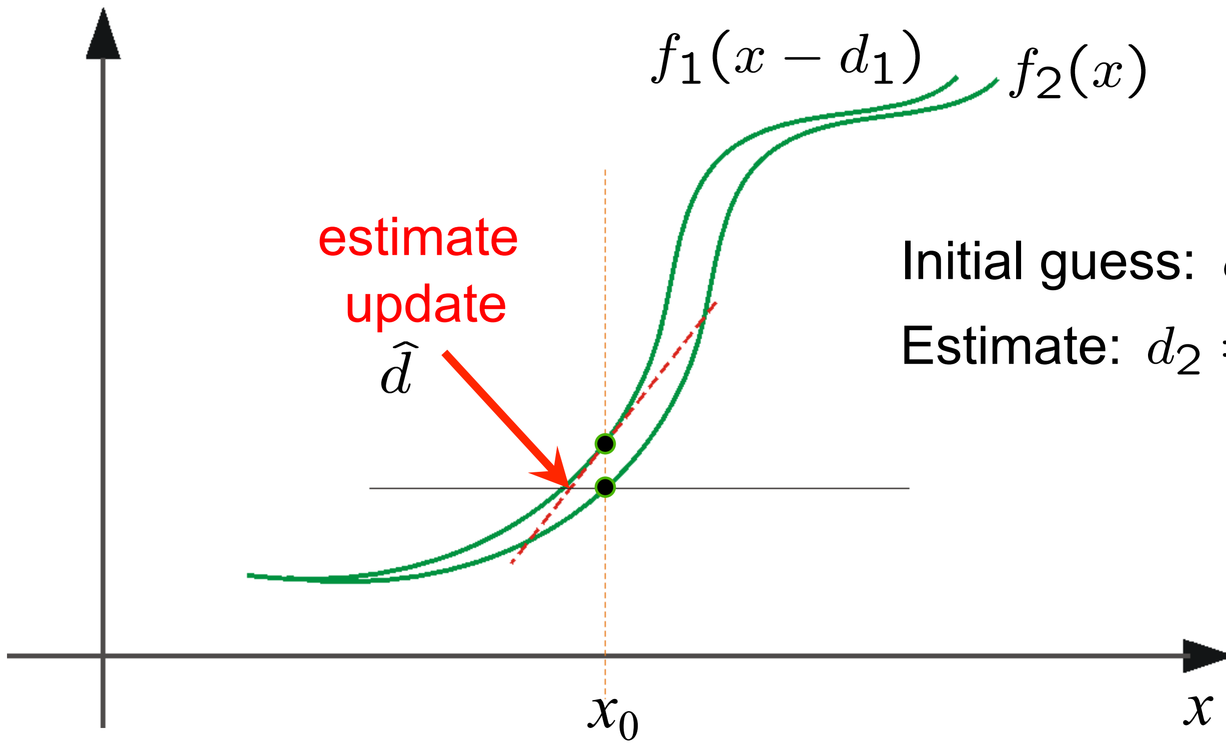


Initial guess: $d_0 = 0$

Estimate: $d_1 = d_0 + \hat{d}$

(using d for *displacement* here instead of u)

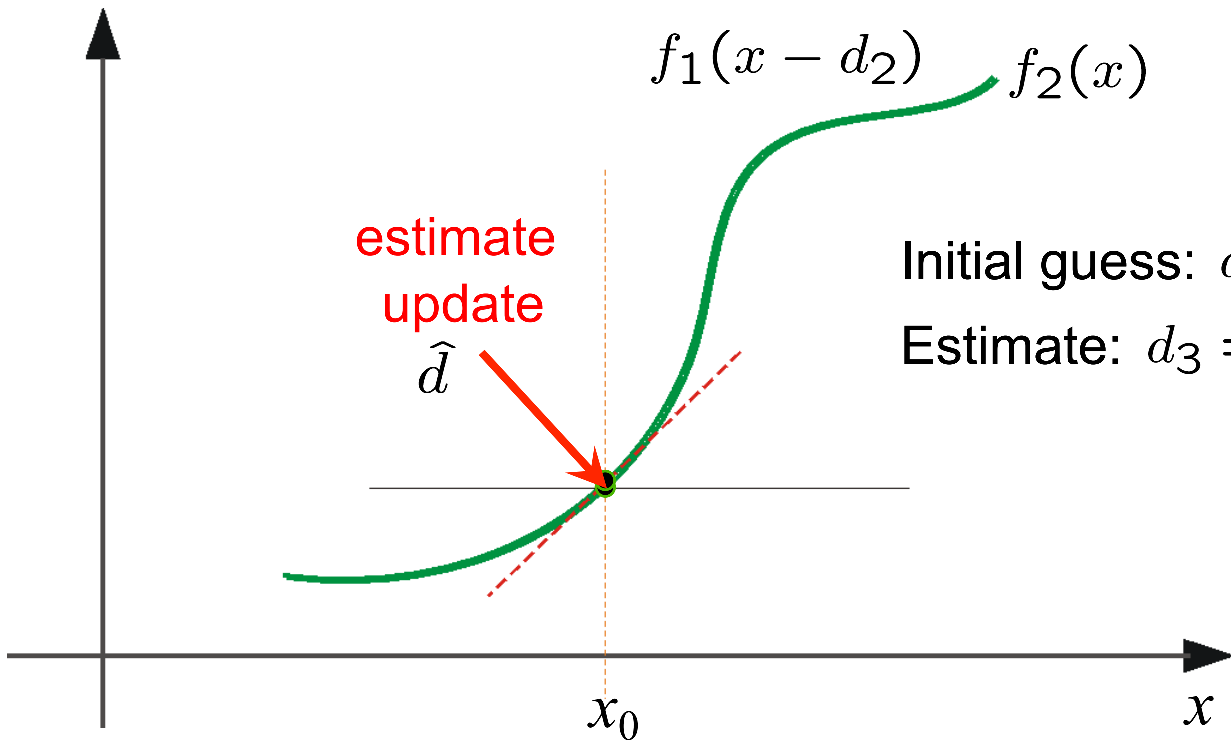
Optical Flow: Iterative Estimation



Initial guess: d_1

Estimate: $d_2 = d_1 + \hat{d}$

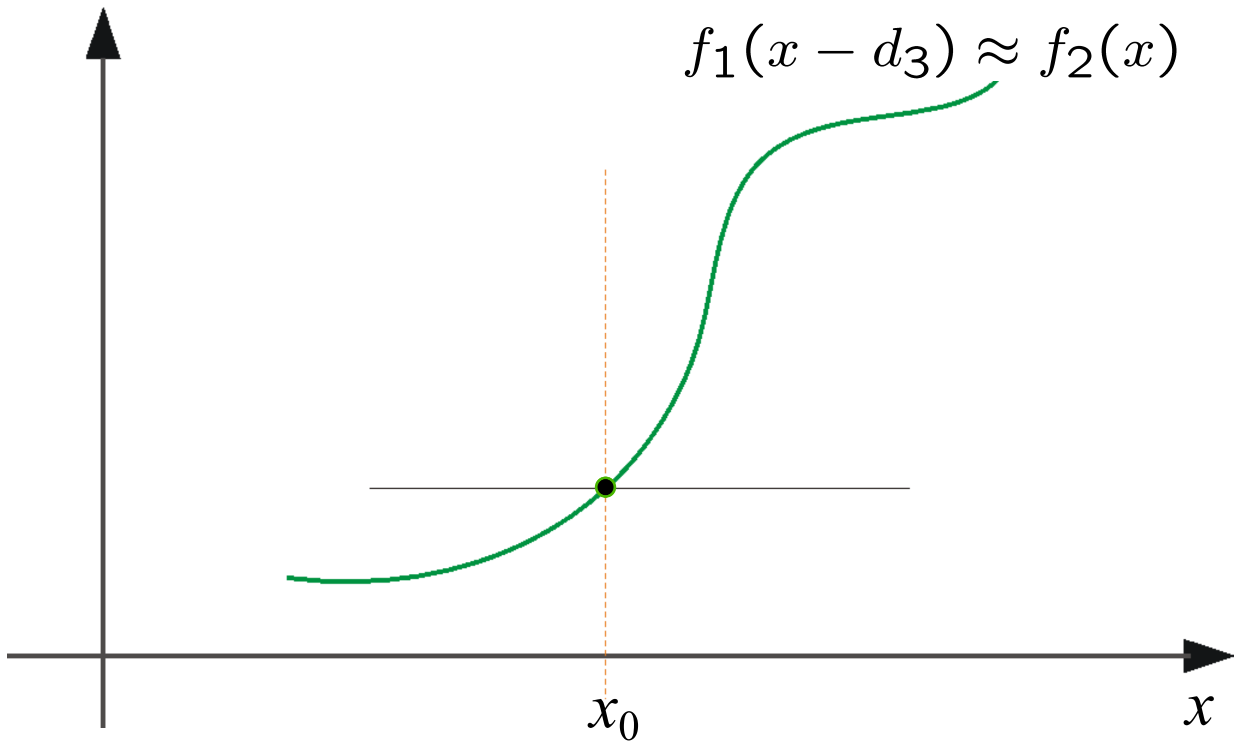
Optical Flow: Iterative Estimation



Initial guess: d_2

Estimate: $d_3 = d_2 + \hat{d}$

Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation

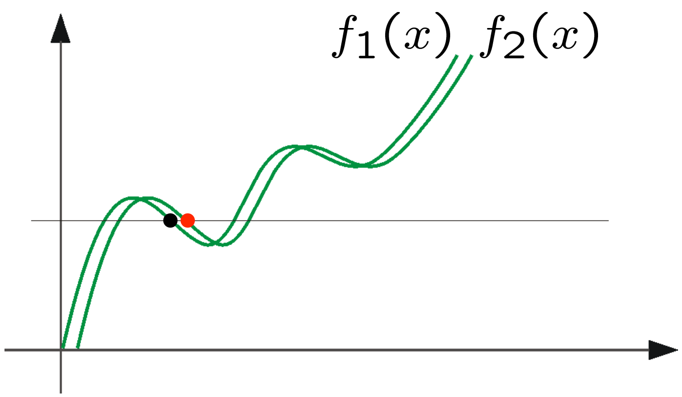
Some Implementation Issues:

- Warping is not easy (ensure that errors in warping are smaller than the estimate refinement)
- Warp one image, take derivatives of the other so you don't need to re-compute the gradient after each iteration.
- Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity)

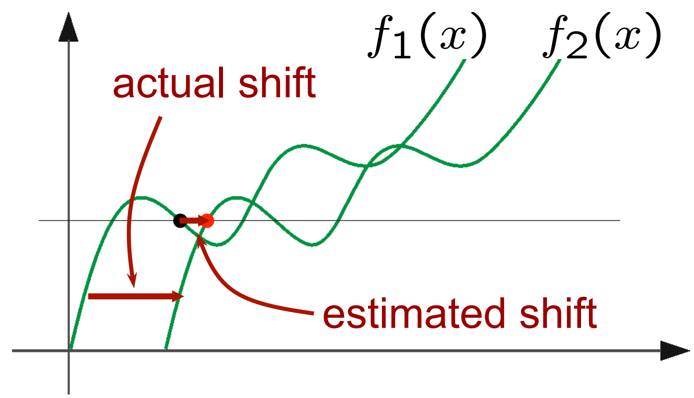
Optical Flow: Aliasing

Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity.

I.e., how do we know which 'correspondence' is correct?



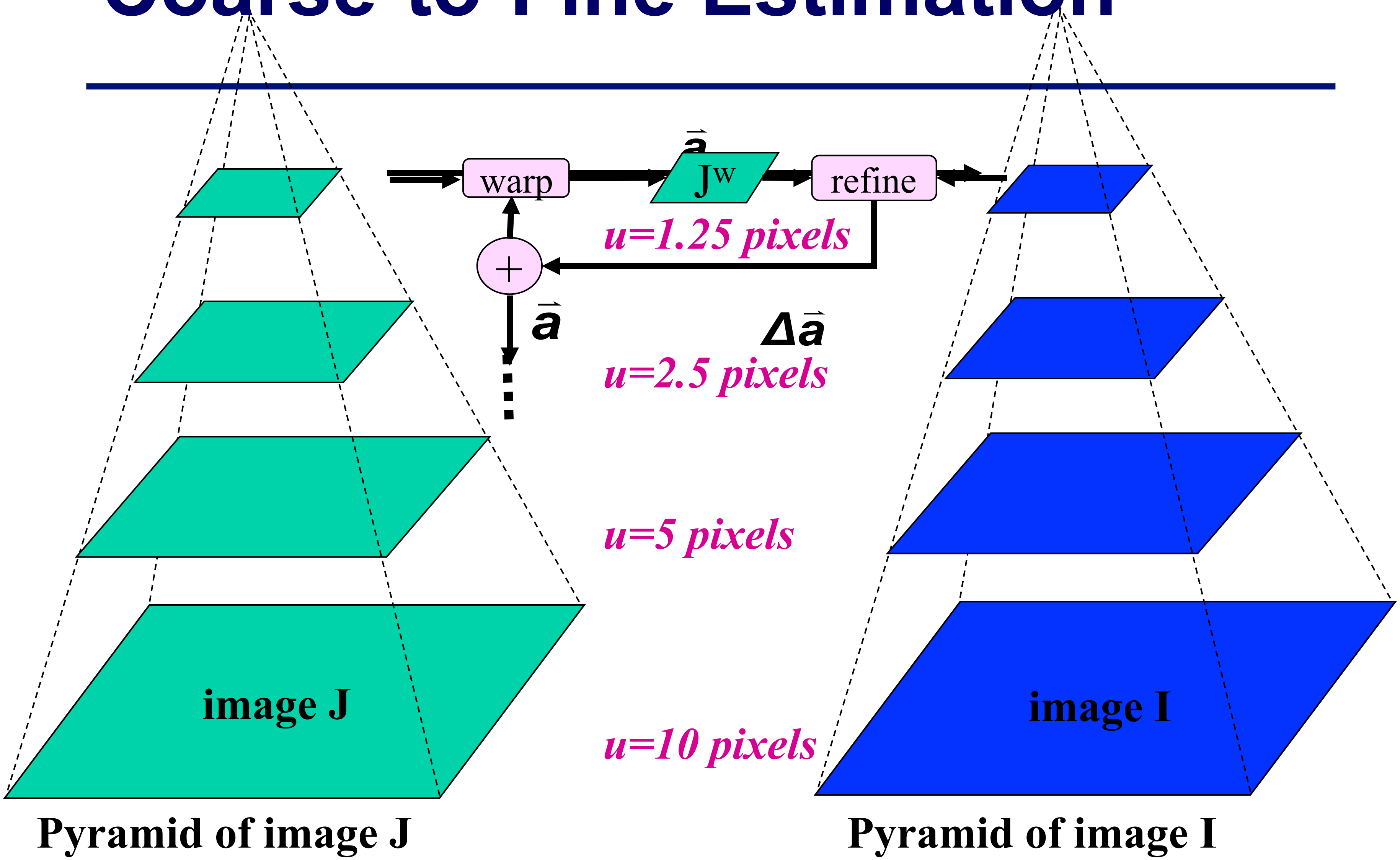
nearest match is correct (no aliasing)



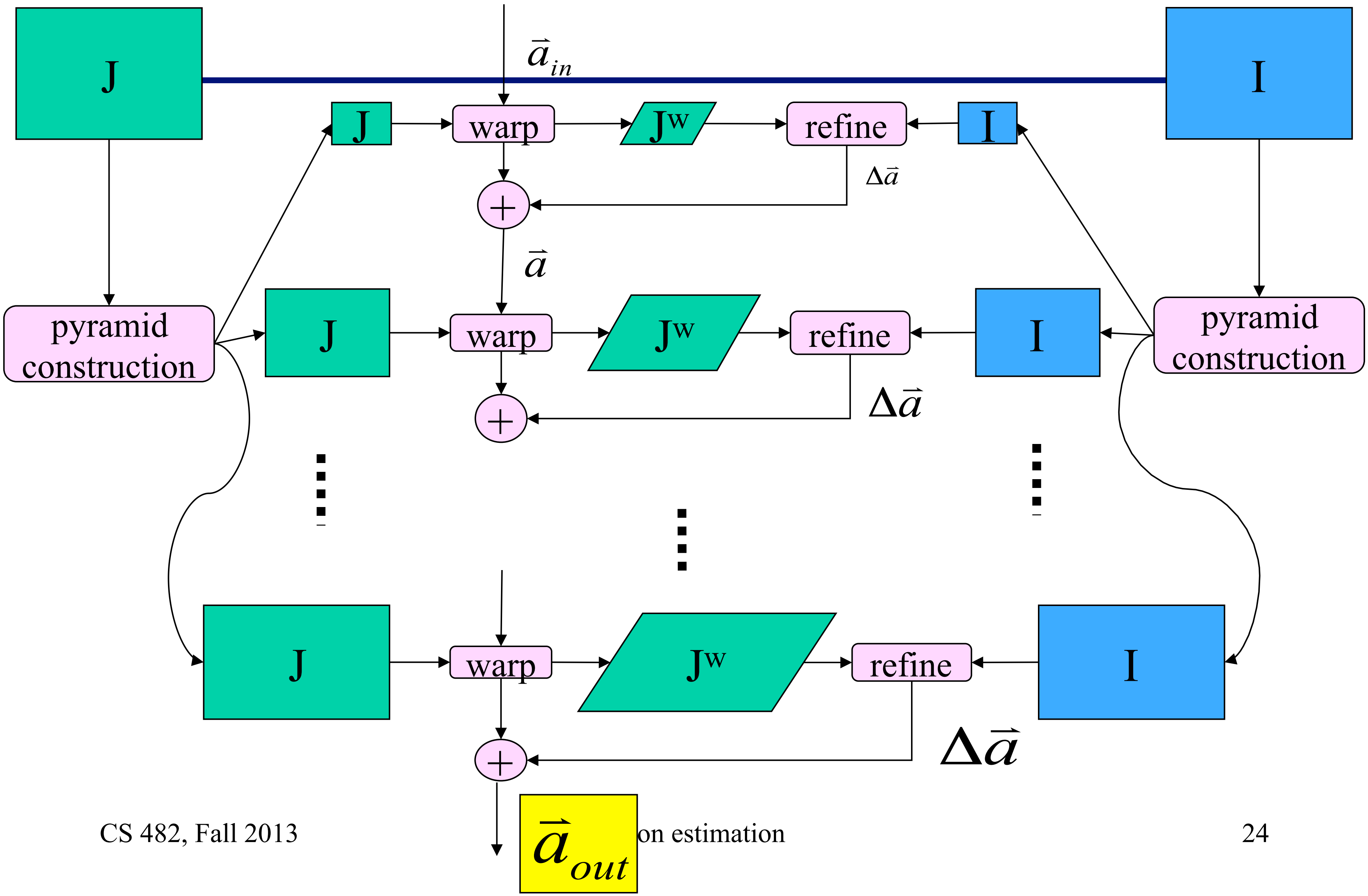
nearest match is incorrect (aliasing)

To overcome aliasing: coarse-to-fine estimation.

Coarse-to-Fine Estimation



Coarse-to-Fine Estimation



Parametric motion estimation

Global (parametric) motion models

2D Models:

Affine

Quadratic

Planar projective transform (Homography)

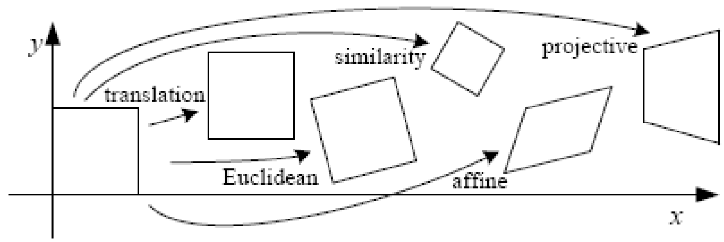
3D Models:

Instantaneous camera motion models

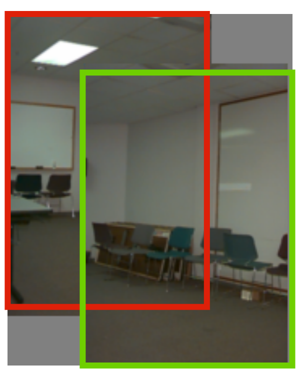
Homography+epipole

Plane+Parallax

Motion models

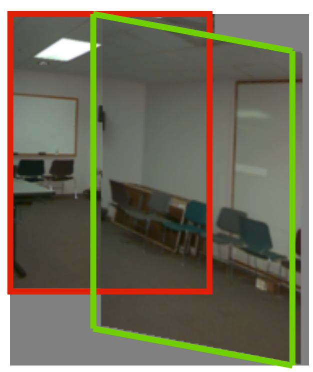


Translation



2 unknowns

Affine

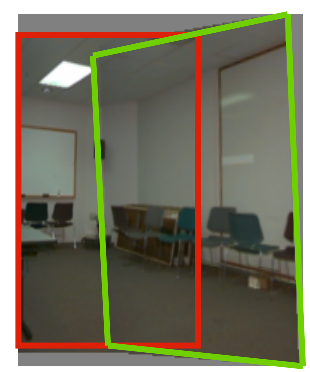


6 unknowns

Perspective

8 unknowns

3D rotation



3 unknowns

Example: Affine Motion

$$u(x,y) = a_1 + a_2 x + a_3 y$$
$$v(x,y) = a_4 + a_5 x + a_6 y$$

Substituting into the B.C. Equation:

$$I_x u + I_y v + I_t \approx 0$$
$$I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \approx 0$$

Each pixel provides 1 linear constraint in 6 *global* unknowns

Least Square Minimization (over all pixels):

$$Err(\vec{a}) = \sum \left(I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t\right)^2$$

Other 2D Motion Models

Quadratic – instantaneous approximation to planar motion

$$u = q_1 + q_2 x + q_3 y + q_7 x^2 + q_8 xy$$

$$v = q_4 + q_5 x + q_6 y + q_7 xy + q_8 y^2$$

Projective – exact planar motion

$$x' = \frac{h_1 + h_2 x + h_3 y}{h_7 + h_8 x + h_9 y}$$

$$y' = \frac{h_4 + h_5 x + h_6 y}{h_7 + h_8 x + h_9 y}$$

and

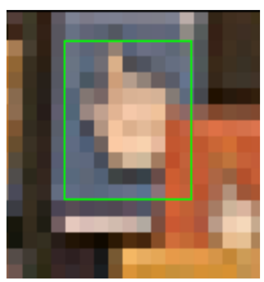
$$u = x' - x, \quad v = y' - y$$

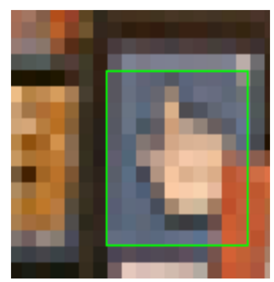
Patch matching (revisited)

How do we determine correspondences?

- *block matching* or *SSD* (sum squared differences)

$$E(x,y;d) = \sum_{(x',y')\in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$





Correlation and SSD

For larger displacements, do template matching

- Define a small area around a pixel as the template
- Match the template against each pixel within a search area in next image.
- Use a match measure such as correlation, normalized correlation, or sum-of-squares difference
- Choose the maximum (or minimum) as the match
- Sub-pixel estimate (Lucas-Kanade)

Shi-Tomasi feature tracker

1. Find good features (min eigenvalue of 2×2 Hessian)
2. Use Lucas-Kanade to track with pure translation
3. Use affine registration with first feature patch
4. Terminate tracks whose dissimilarity gets too large
5. Start new tracks when needed

Tracking results

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

Tracking - dissimilarity

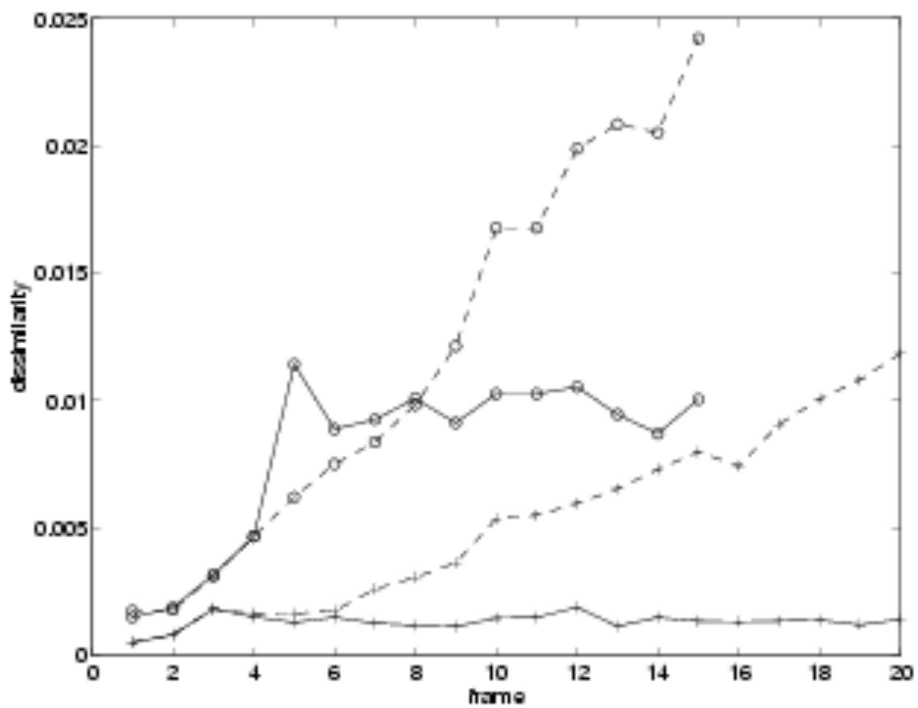


Figure 3: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 1 (plusses) and 4 (circles).

Tracking results

Figure 13: Labels of some of the features in figure 11.

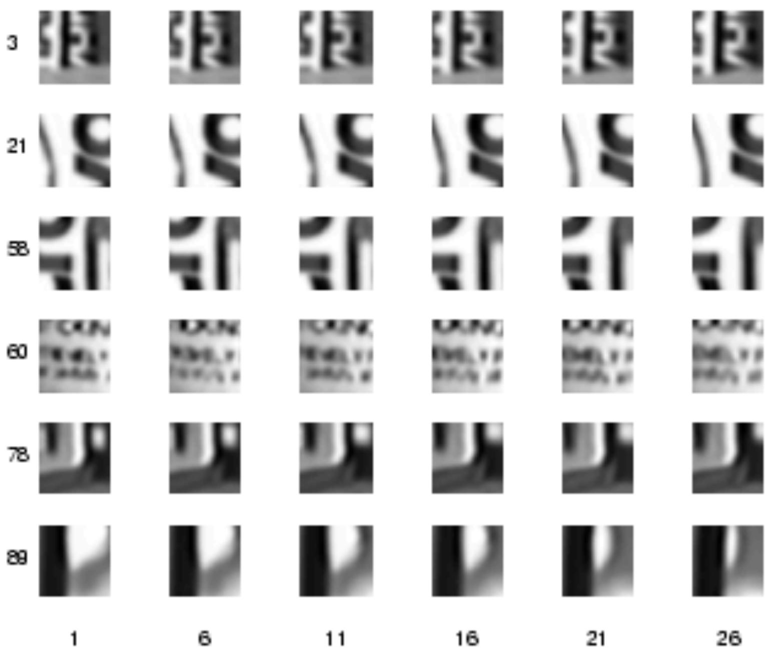


Figure 14: Six sample features through six sample frames.

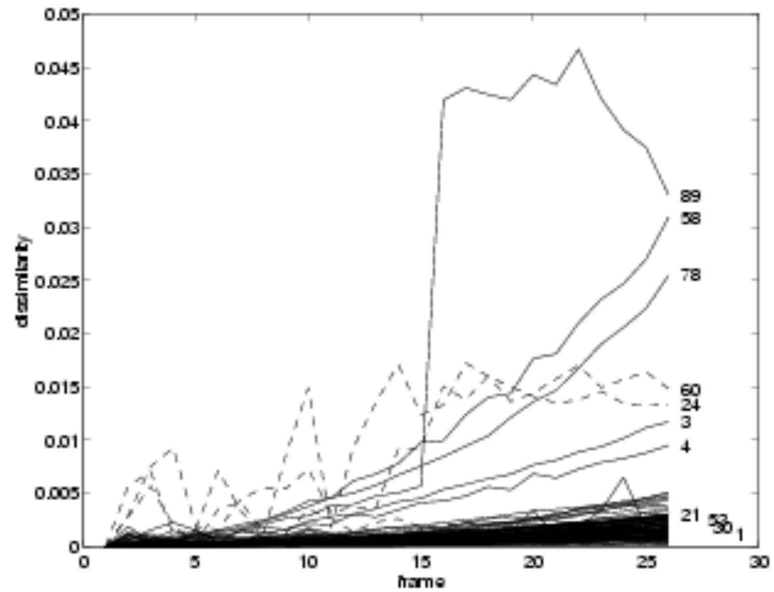


Figure 15: Affine motion dissimilarity for the features in figure 11. Notice the good discrimination between good and bad features. Dashed plots indicate aliasing (see text).

Features 24 and 60 deserve a special discussion, and

Correlation Window Size

Small windows lead to more false matches

Large windows are better this way, but…

- Neighboring flow vectors will be more correlated (since the template windows have more in common)
- Flow resolution also lower (same reason)
- More expensive to compute

Small windows are good for local search:
more detailed and less smooth (noisy?)

Large windows good for global search:
less detailed and smoother

Robust Estimation

Noise distributions are often non-Gaussian, having much heavier tails. Noise samples from the tails are called outliers.

Sources of outliers (multiple motions):

- specularities / highlights
- jpeg artifacts / interlacing / motion blur
- multiple motions (occlusion boundaries, transparency)

velocity space

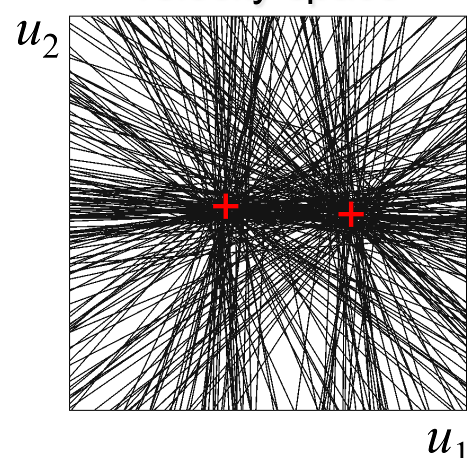
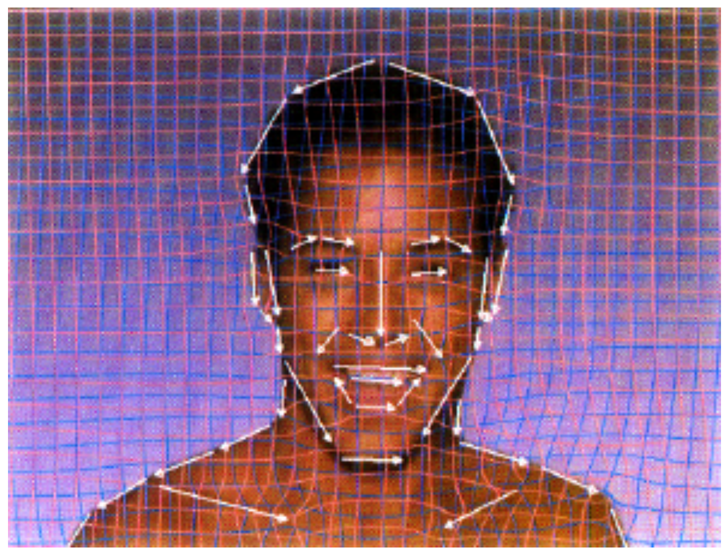


Image Morphing

Image Warping – non-parametric

Specify more detailed warp function



Examples:

- splines
- triangles
- optical flow (per-pixel motion)

Image Warping – non-parametric

Move control points to specify spline warp

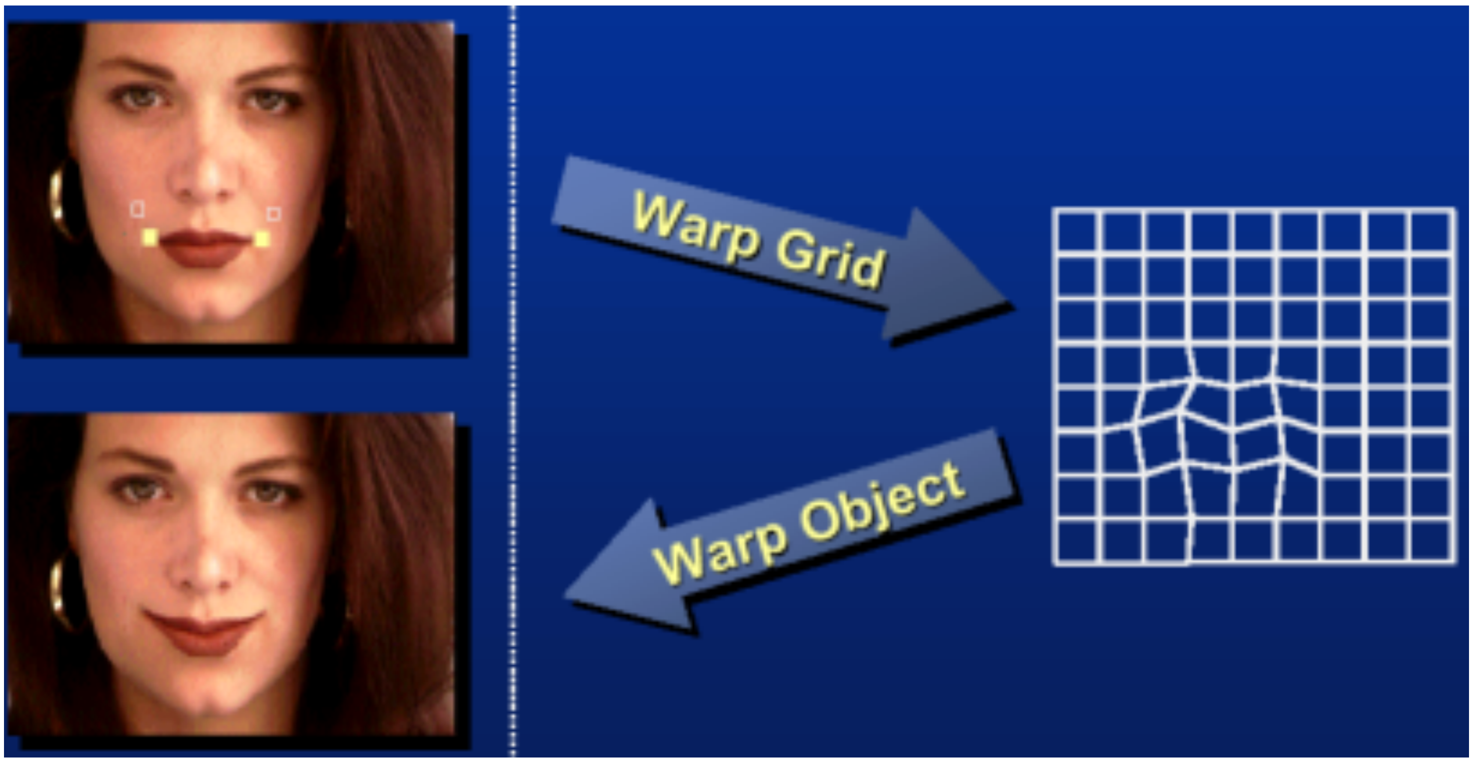
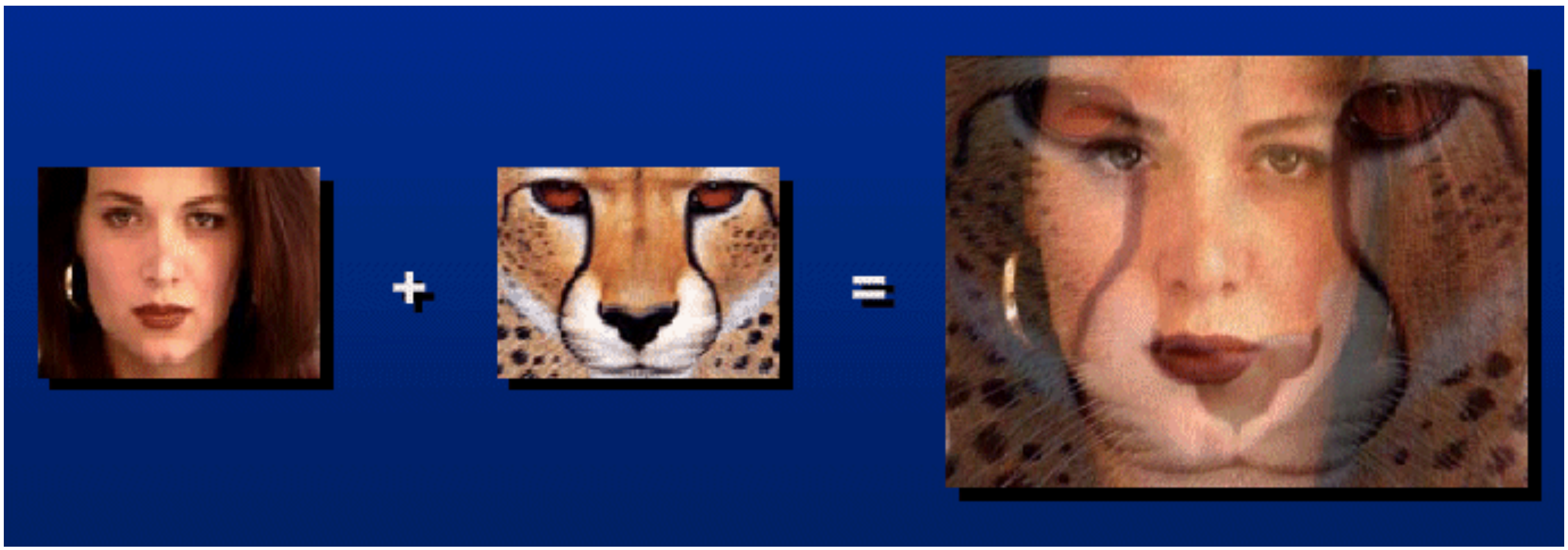


Image Morphing

How can we *in-between* two images?

1. Cross-dissolve

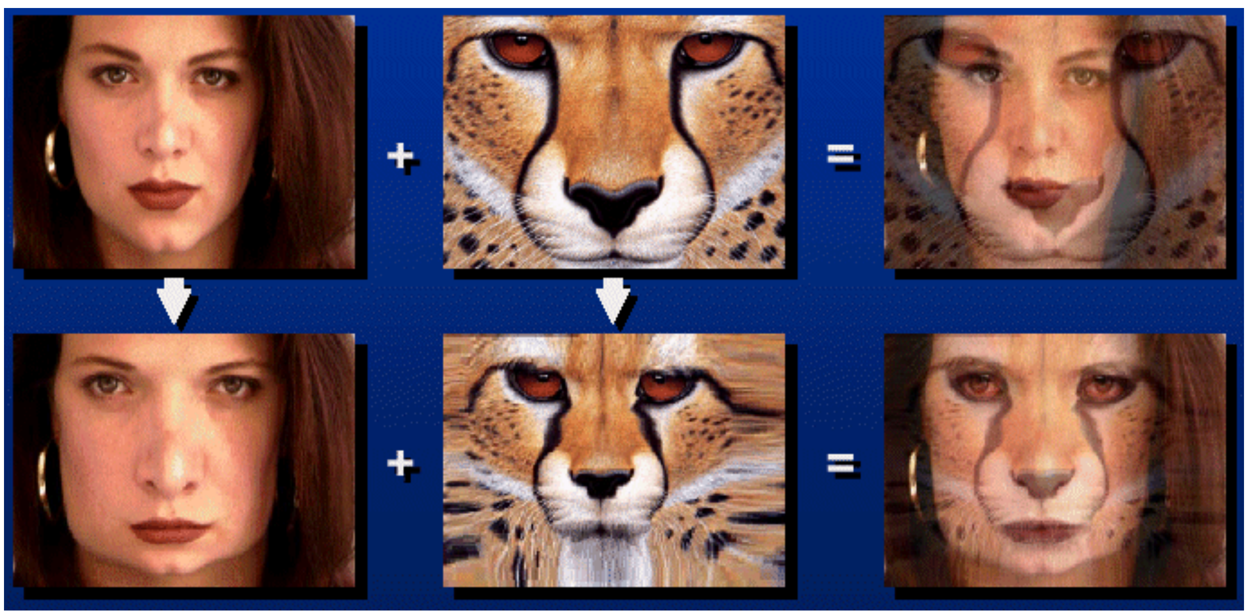


(all examples from [Gomes *et al.*' 99])

Image Morphing

How can we *in-between* two images?

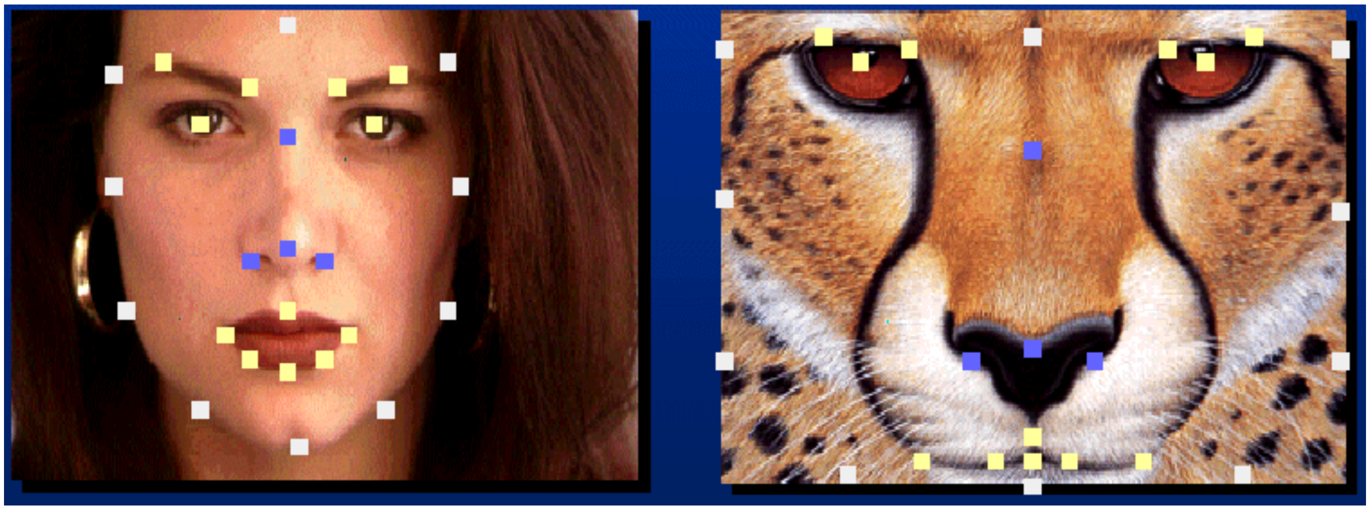
2. Warp then cross-dissolve = *morph*



Warp specification

How can we specify the warp?

1. Specify corresponding *points*
   - *interpolate* to a complete warping function

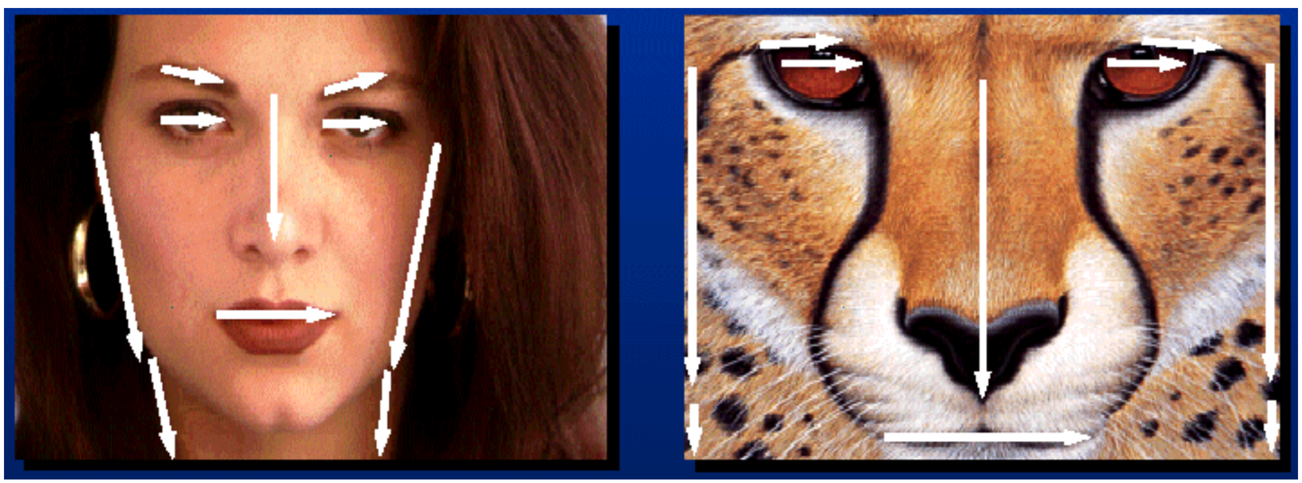


- Nielson, *Scattered Data Modeling*, IEEE CG&A' 93]

Warp specification

How can we specify the warp?

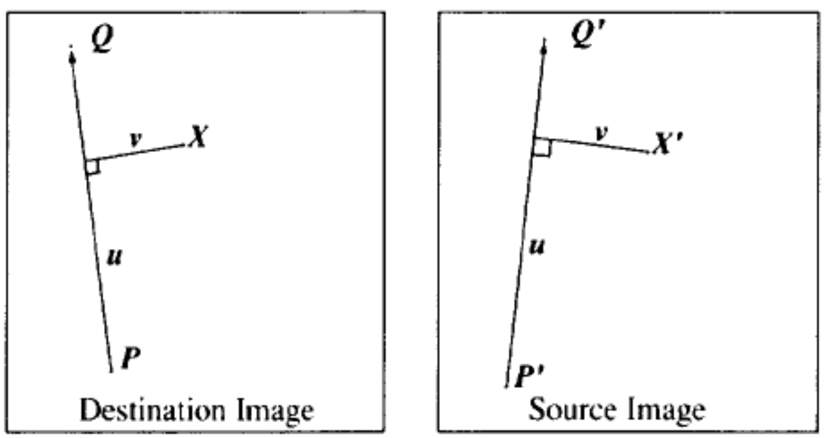
2. Specify corresponding *vectors*
    - *interpolate* to a complete warping function



Warp specification

How can we specify the warp?

2. Specify corresponding *vectors*
    - *interpolate* [Beier & Neely, SIGGRAPH' 92]



For each pixel X in the destination
  $DSUM = (0,0)$
  $weightsum = 0$
  For each line $P_i\, Q_i$
    calculate u,v based on $P_i\, Q_i$
    calculate X'_i based on u,v and $P_i' Q_i'$
    calculate displacement $D_i = X_i' - X_i$ for this line
    $dist$ = shortest distance from X to $P_i\, Q_i$
    $weight = (length^p / (a + dist))^b$
    $DSUM \mathrel{+}= D_i * weight$
    $weightsum \mathrel{+}= weight$
  $X' = X + DSUM / weightsum$
  destinationImage(X) = sourceImage(X')

Warp specification

How can we specify the warp?

3. Specify corresponding *spline control points*
   - *interpolate* to a complete warping function