CS 664
Visual Motion (2)

Daniel Huttenlocher

Last Time

- Visual motion (optical flow)
  - Apparent motion of image pixels over time
- Brightness constancy assumption and optical flow constraint equation (gradient constraint)

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

- Direct and matching-based methods for estimating motion field (u,v)
  - Dense and sparse matching

Matching vs. Gradient Based

Consider image I translated by u_0, v_0

$$I_0(x,y) = I(x,y)$$
$$I_1(x+u_0, y+v_0) = I(x,y) + \eta_1(x,y)$$

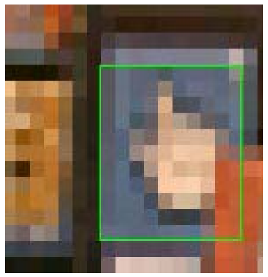
$$E(u,v) = \sum_{x,y}(I(x,y) - I_1(x+u, y+v))^2$$
$$= \sum_{x,y}(I(x,y) - I(x-u_0+u, y-v_0+v) - \eta_1(x,y))^2$$

Discrete search methods search for the best estimate, u(x,y),v(x,y).
Gradient methods linearize the intensity function and solve for the estimate.

Patch Matching

- Determining correspondences
  - Block matching or SSD (sum squared differences)

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



Dense Matching Based Methods

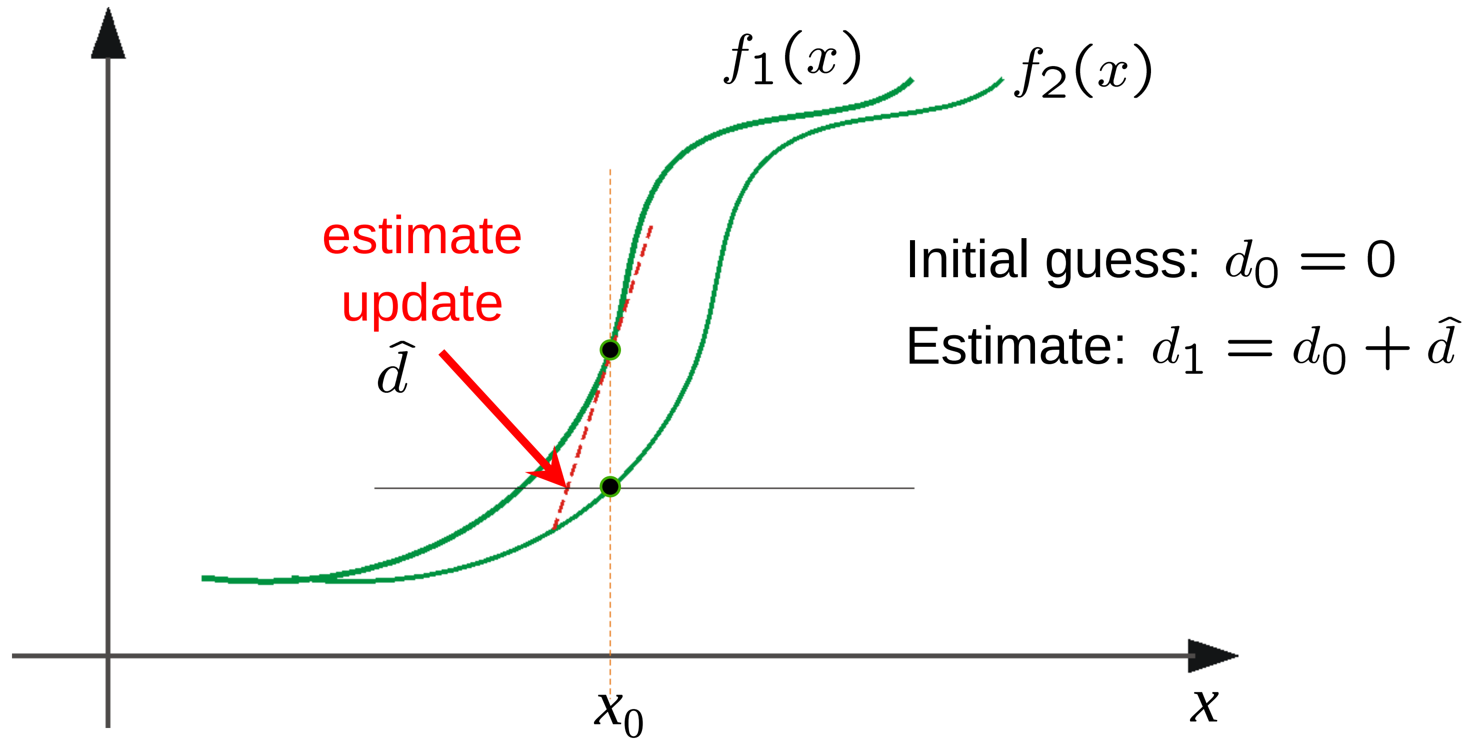
- Block matching for larger displacements
  - Define a small area around a pixel as the template
  - Match the template against each pixel within a search area in next image.
  - Use a match measure such as correlation, normalized correlation, or sum-of-squares difference
  - Choose the maximum (or minimum)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

Gradient Based Methods

- 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

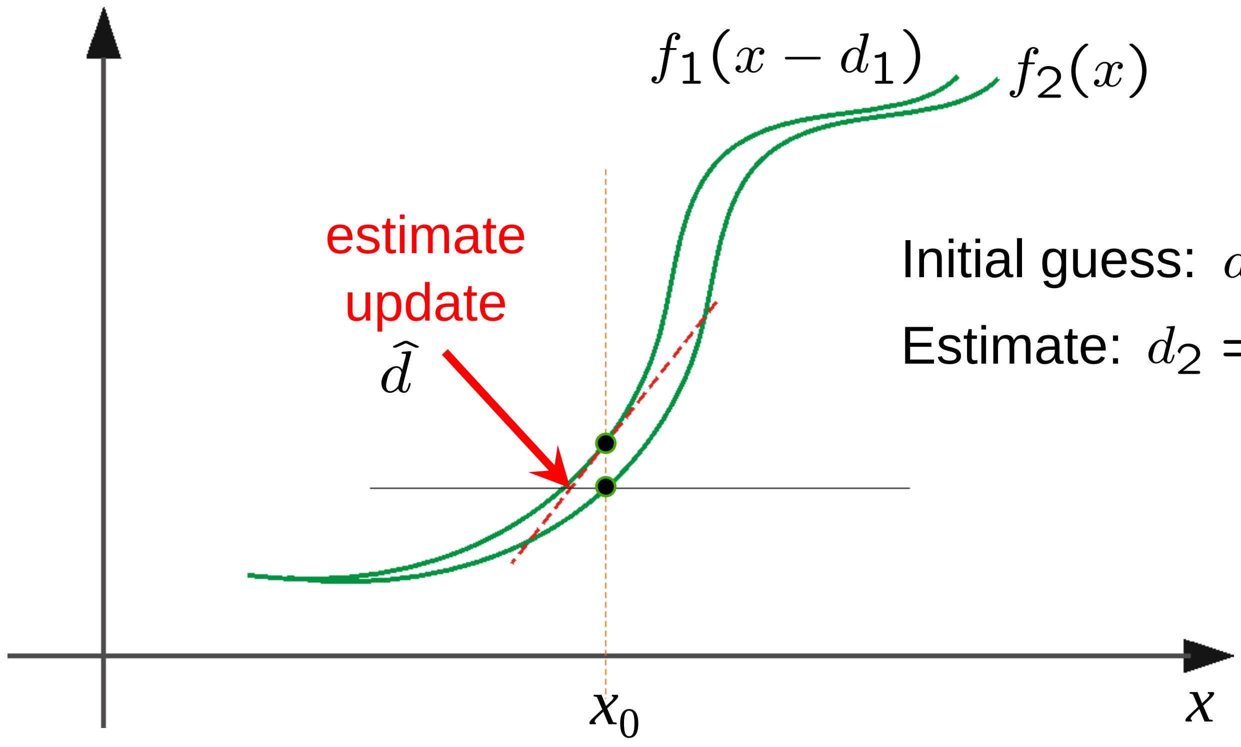
  (easier said than done)
- Refine estimate by repeating the process
- Coarse-to-fine process for larger motions

Optical Flow: Iterative Estimation



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

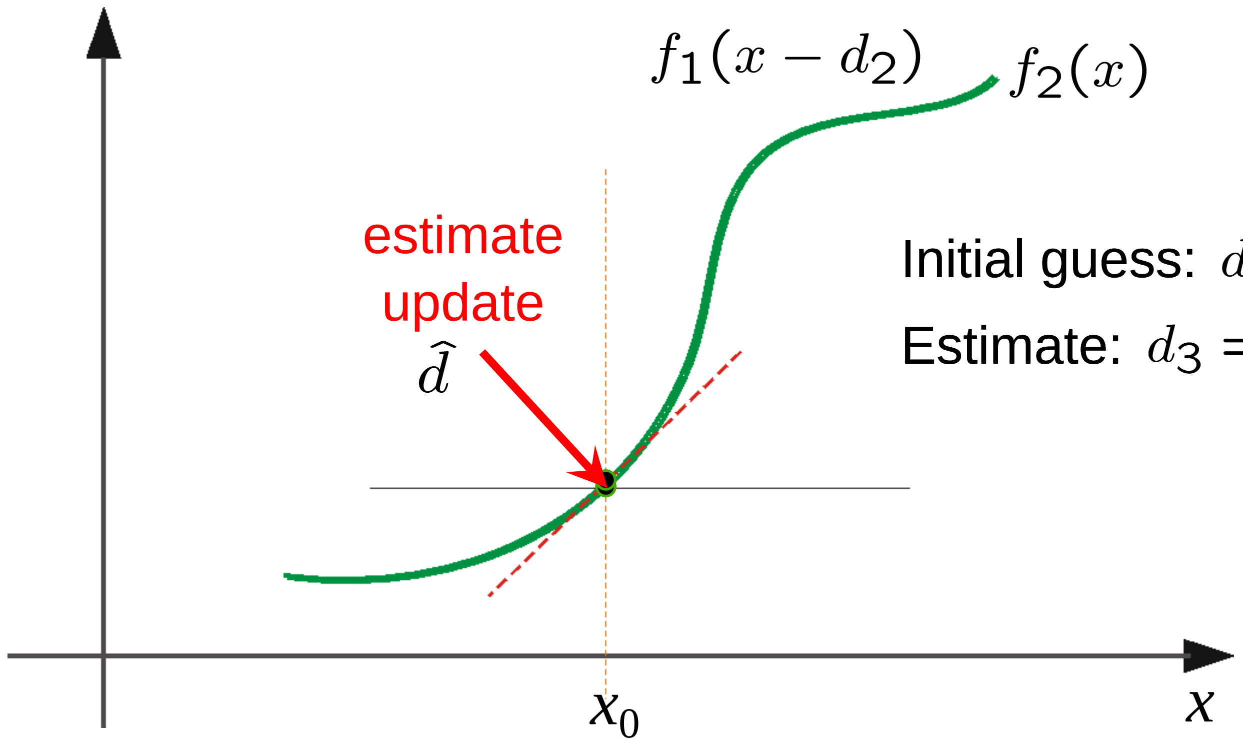
Optical Flow: Iterative Estimation



Initial guess: d_1

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

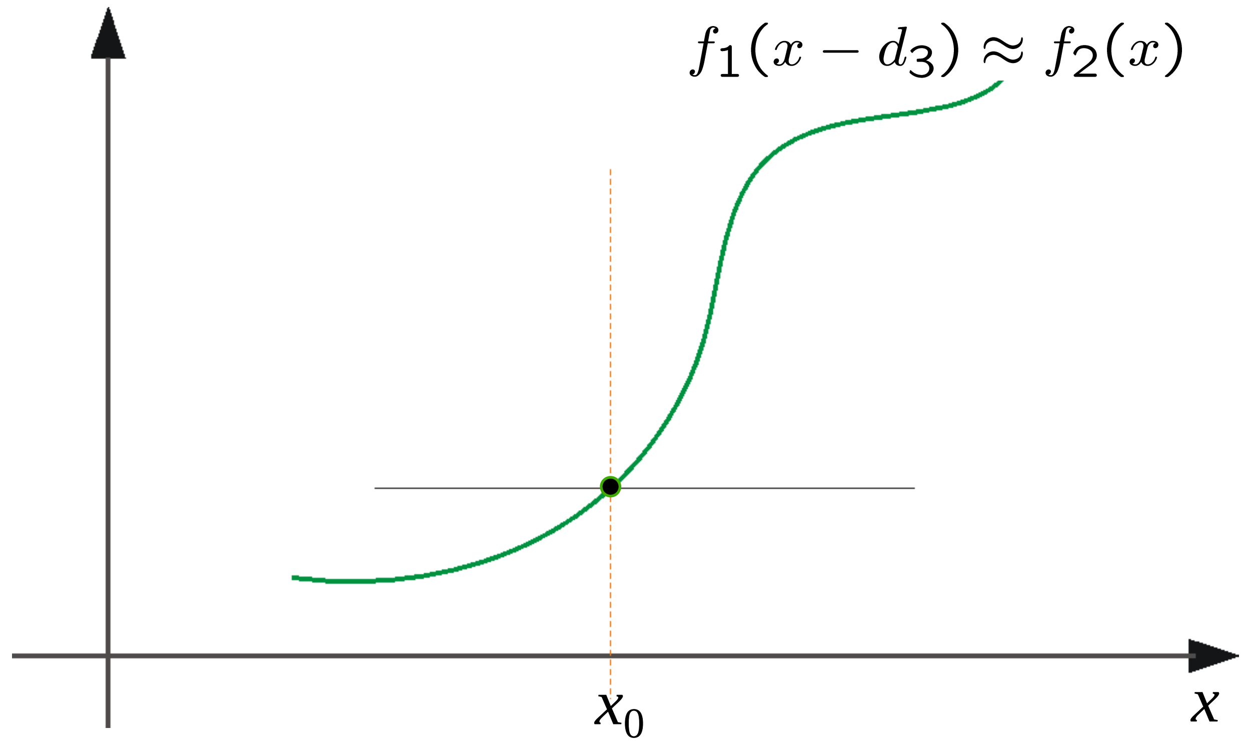
Optical Flow: Iterative Estimation



Initial guess: d_2

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

Optical Flow: Iterative Estimation



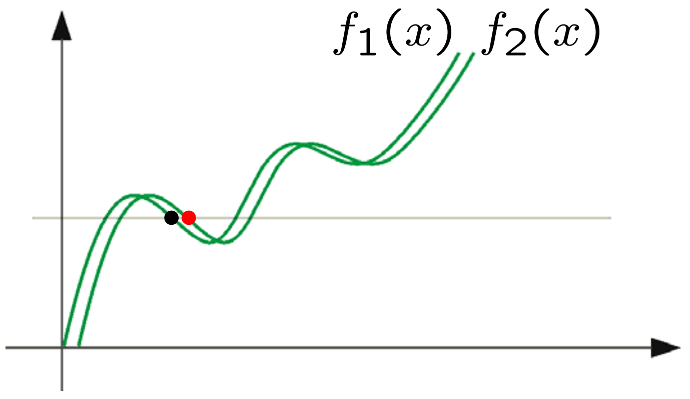
Optical Flow: Iterative Estimation

- Some issues:
  - Warping not easy (need to be sure 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)

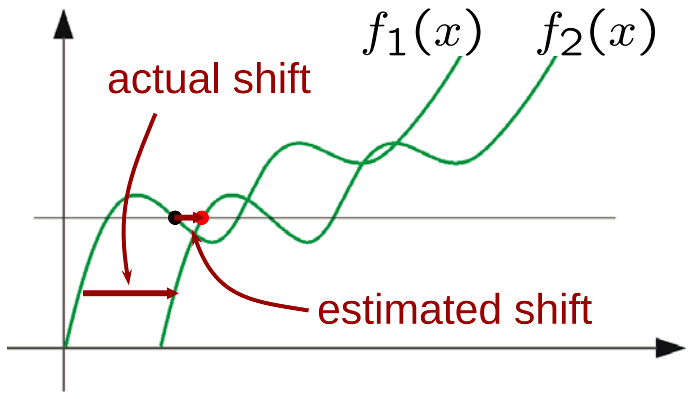
Optical Flow: Aliasing

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 correct
(no aliasing)



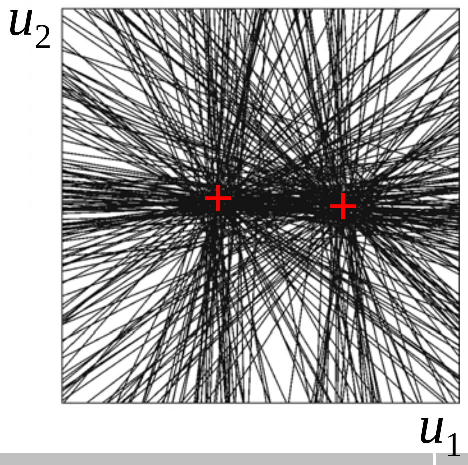
Nearest match incorrect
(aliasing)

To overcome aliasing: coarse-to-fine estimation.

Robust Estimation

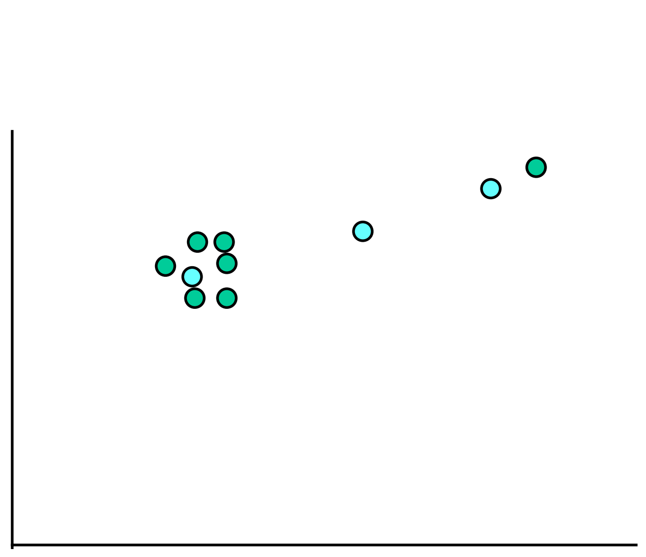
- Noise distributions are often non-Gaussian, having much heavier tails. Noise samples from the tails are called outliers.
- Sources of outliers (multiple motions):
  - specularities / highlights
  - jpeg artifacts / interlacing / motion blur
  - multiple motions (occlusion boundaries, transparency)

velocity space



Robust Estimation

Standard Least Squares Estimation allows too much influence for outlying points (similar in stereo and other correspondence problems)



$$E(m) = \sum_i \rho(x_i)$$

$$\rho(x_i) = (x_i - m)^2$$

Influence $\psi(x) = \frac{\partial \rho}{\partial x} = (x_i - m)$

Robust Estimation

- IRLS (iteratively reweighted least squares)

- Use of robust error functions
  - Robust gradient constraint

$$E_d(u_s, v_s) = \sum \rho\left(I_x u_s + I_y v_s + I_t\right)$$

  - Robust SSD

$$E_d(u_s, v_s) = \sum \rho\left(I(x, y) - J(x + u_s, y + v_s)\right)$$

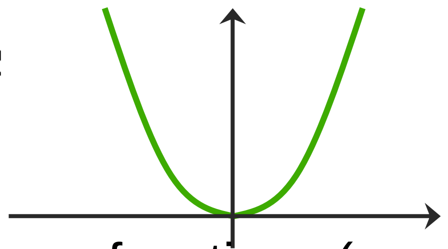
Robust Estimation

Problem: Least-squares estimators penalize deviations between data & model with quadratic error f^{n} (extremely sensitive to outliers)

error penalty function

influence function

$$\rho(\epsilon) = \epsilon^2$$



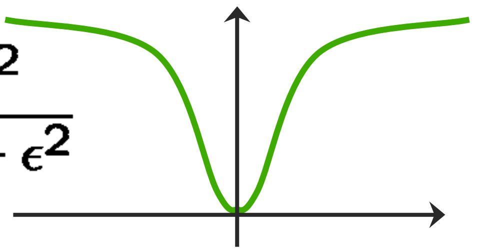
$$\psi(\epsilon) = \frac{\partial\rho(\epsilon)}{\partial\epsilon} = 2\,\epsilon$$

Redescending error functions (e.g., Geman-McClure) help to reduce the influence of outlying measurements.

error penalty function

influence function

$$\rho(\epsilon;\, s) = \frac{\epsilon^2}{s + \epsilon^2}$$



$$\psi(\epsilon;\, s) = \frac{2\,\epsilon\, s}{(s + \epsilon^2)^2}$$

“Global” (Nonlocal) Motion Estimation

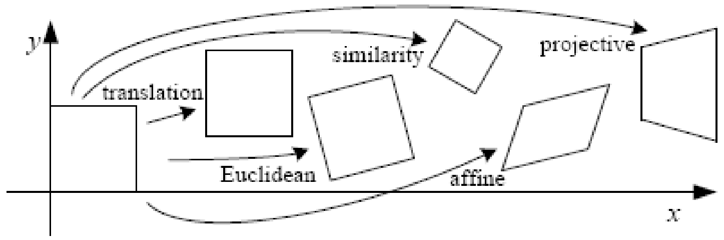
- Estimate motion vectors that are parameterized over some region
  - Each vector fits some low-order model of how vector field changes spatially
- Regions can be contiguous image patches or “layers” of some kind
- Most successful motion estimation techniques in practice use global motion estimates over patches and/or layers

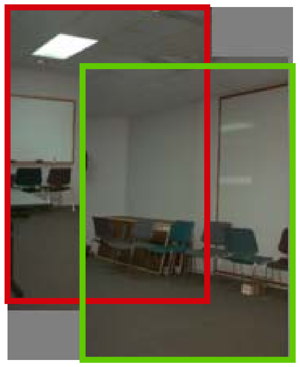
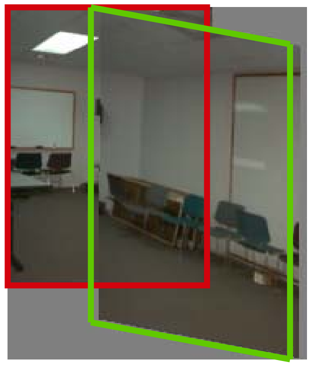
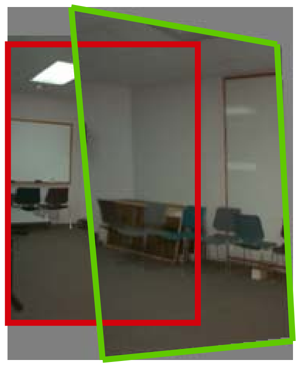
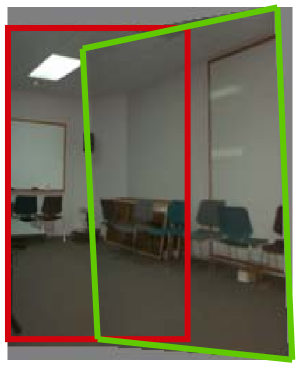
“Global” Motion Models

- Often referred to as parametric motion
- 2D Models
  - Affine
  - Quadratic
  - Planar projective transform (Homography)

- 3D Models
  - Homography+epipole
  - Plane+Parallax

Motion Models



Translation	Affine	Perspective	3D rotation
			
2 unknowns	6 unknowns	8 unknowns	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 B.C. Equation:

$$\boxed{I_x \cdot u + I_y \cdot v + I_t \approx 0}$$

$$\boxed{I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \approx 0}$$

Each pixel provides linear constraint on 6 (global) unknowns

Least Squares Minimization (over all pixels):

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

Other Global 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

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

3D Motion Models

Homography+Epipole

Global parameters: $\boxed{h_1, \ldots, h_9, t_1, t_2, t_3}$

Local Parameter: $\boxed{\gamma(x, y)}$

$$x' = \frac{h_1 x + h_2 y + h_3 + \gamma t_1}{h_7 x + h_8 y + h_9 + \gamma t_3}$$

$$y' = \frac{h_4 x + h_5 y + h_6 + \gamma t_1}{h_7 x + h_8 y + h_9 + \gamma t_3}$$

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

Residual Planar Parallax Motion

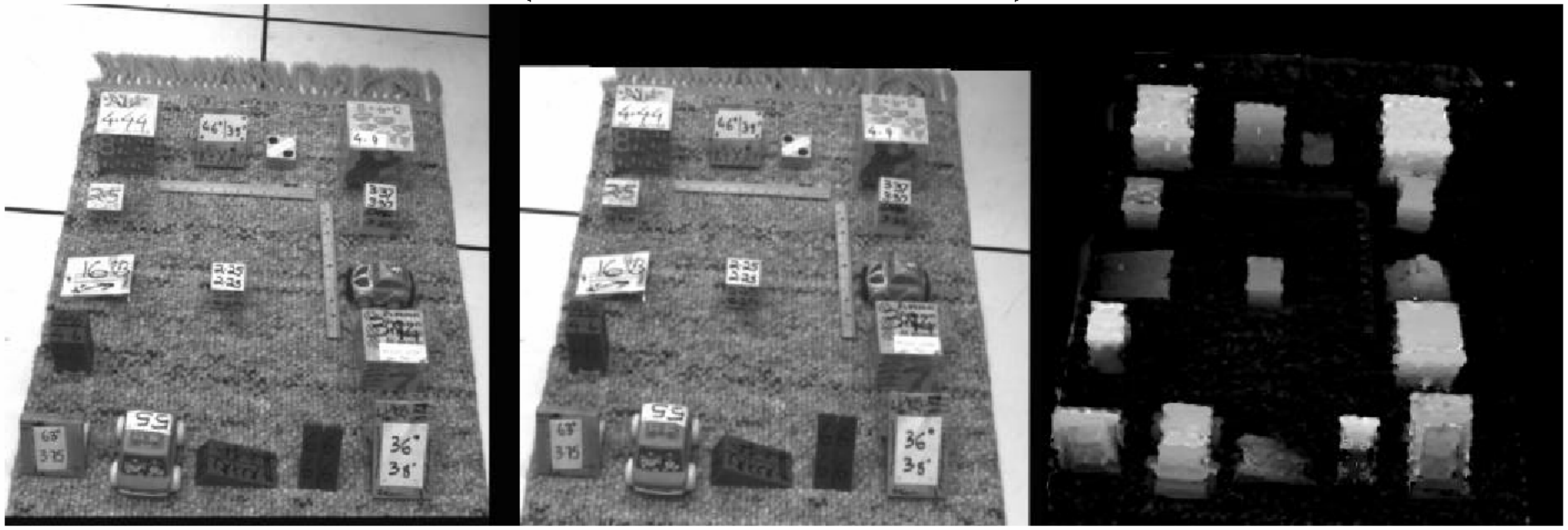
Global parameters: $\boxed{t_1, t_2, t_3}$

Local Parameter: $\boxed{\gamma(x, y)}$

$$u = x^w - x = \frac{\gamma}{1 + \gamma t_3}(t_3 x - t_1)$$

$$v = y^w - x = \frac{\gamma}{1 + \gamma t_3}(t_3 y - t_2)$$

Residual Planar Parallax Motion (Plane+Parallax)

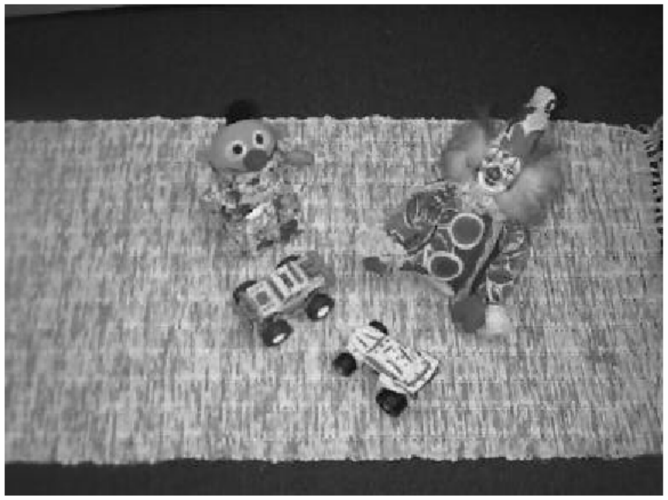


Original sequence **Plane-aligned sequence** **Recovered shape**

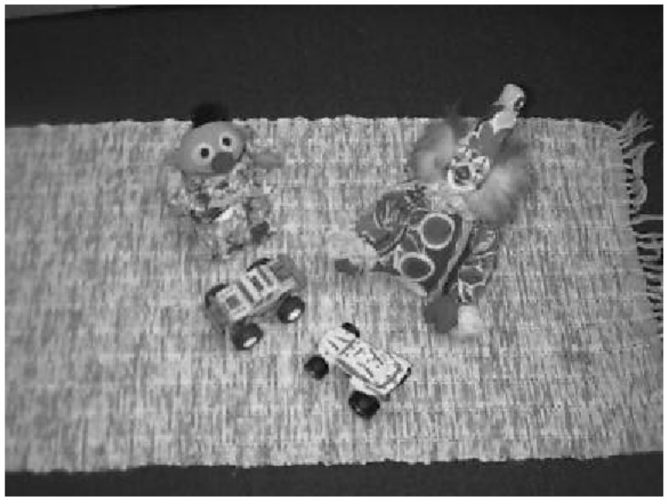
Block sequence from [Kumar-Anandan-Hanna'94]

"Given two views where motion of points on a parametric surface has been compensated, the residual parallax is an epipolar field"

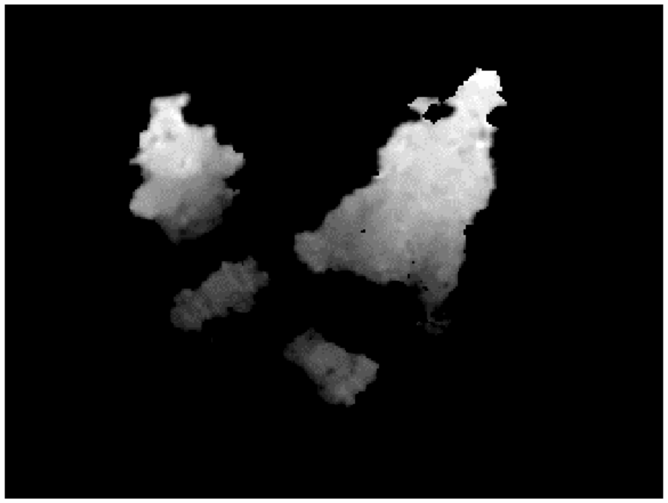
Dense 3D Reconstruction
(Plane+Parallax)



Original sequence



Plane-aligned sequence



Recovered shape

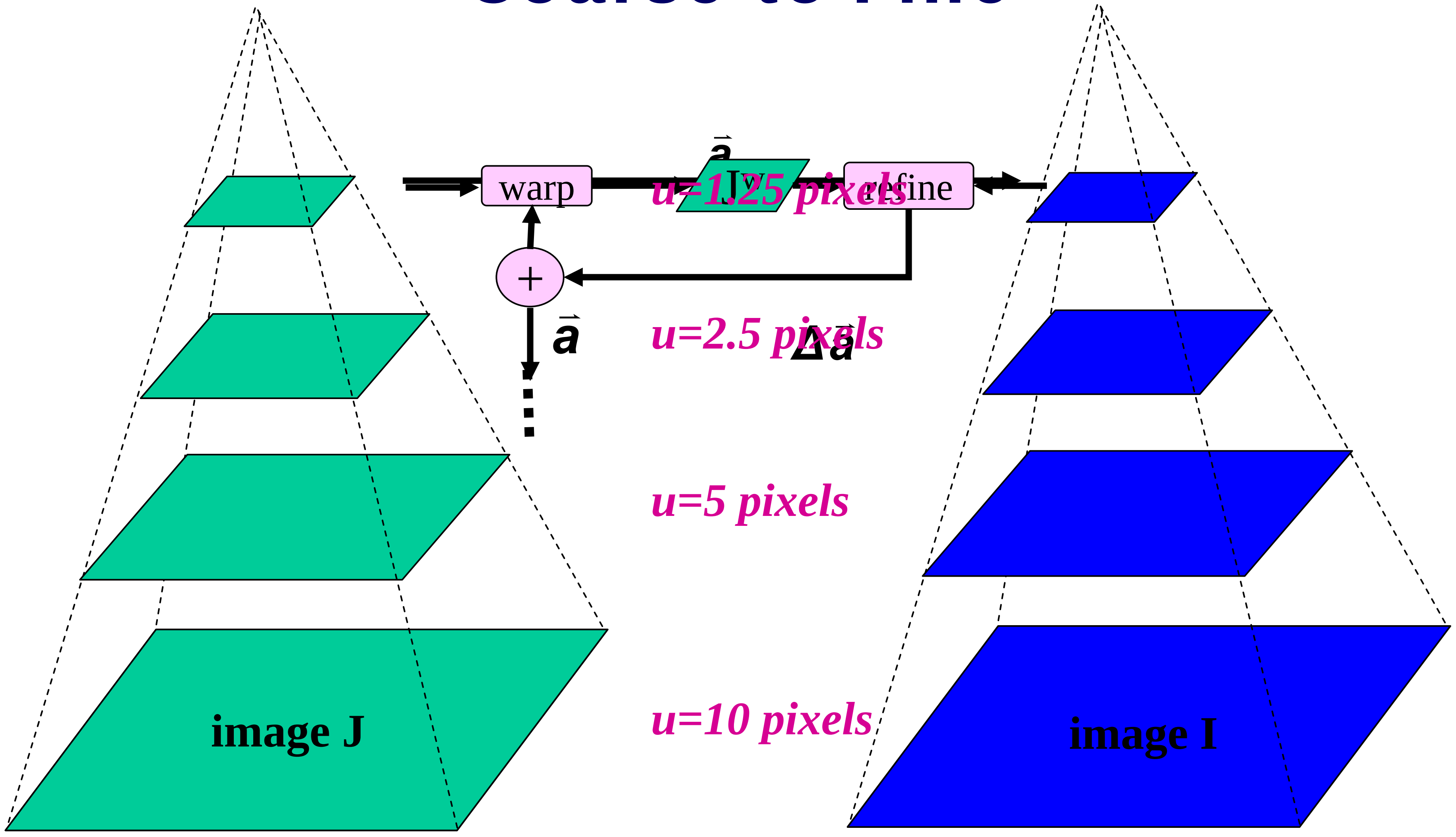
Multiple (Layered) Motions

- Combining global parametric motion estimation with robust estimation
  - Calculate predominant parameterized motion over entire image (e.g., affine)
  - Corresponds to largest planar surface in scene under orthographic projection
    - If doesn't occupy majority of pixels robust estimator will probably fail to recover its motion
  - Outlier pixels (low weights in IRLS) are not part of this surface
    - Recursively try estimating their motion
    - If no good estimate, then remain outliers

Limits of Gradient Based Methods

- Fails when
  - Intensity structure in window is poor
  - Displacement is large (typical operating range is motion of 1 pixel)
    - Linearization of brightness is suitable only for small displacements
- Brightness not strictly constant in images
  - Less problematic than appears, since can pre-filter images to make them look similar
- Large displacements can be addressed by coarse-to-fine (challenge to do locally)

Coarse to Fine



Pyramid of image J

Pyramid of image I

Coarse to Fine Estimation

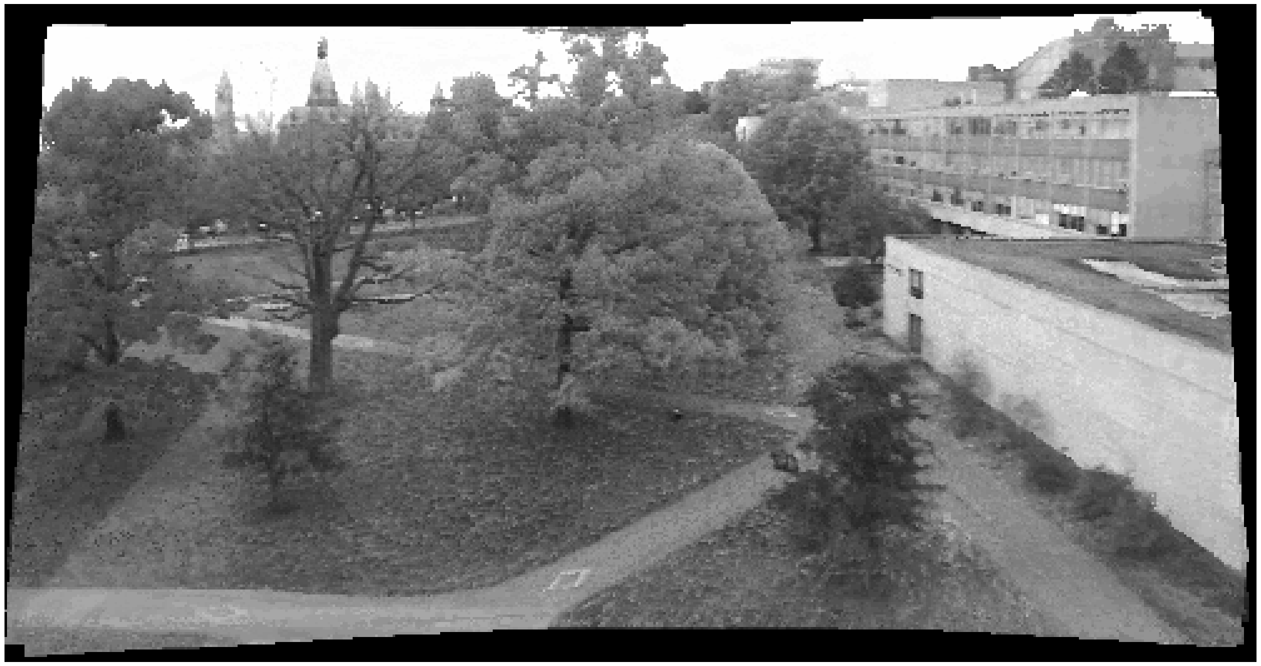
- Compute M^k, estimate of motion at level k
  - Can be local motion estimate (u^k,v^k)
    - Vector field with motion of patch at each pixel
  - Can be global motion estimate
    - Parametric model (e.g., affine) of dominant motion for entire image
  - Choose max k such that motion about one pixel
- Apply M^k at level k-1 and estimate remaining motion at that level, iterate
  - Local estimates: shift I^k by $2(u^k,v^k)$
  - Global estimates: apply inverse transform to J^{k-1}

Global Motion Coarse to Fine

- Compute transformation T^k mapping pixels of I^k to J^k
- Warp image J^{k-1} using T^k
  - Apply inverse of T^k
  - Double resolution of T^k (translations double)
- Compute transformation T^{k-1} mapping pixels of I^k to <u>warped</u> J^{k-1}
  - Estimate of "residual" motion at this level
  - Total estimate of motion at this level is composition of T^{k-1} and resolution doubled T^k
    - In case of translation just add them

Affine Mosaic Example

- Coarse-to-fine affine motion
  - Pan tilt camera sweeping repeatedly over scene
- Moving objects removed from background
  - Outliers in motion estimate removed

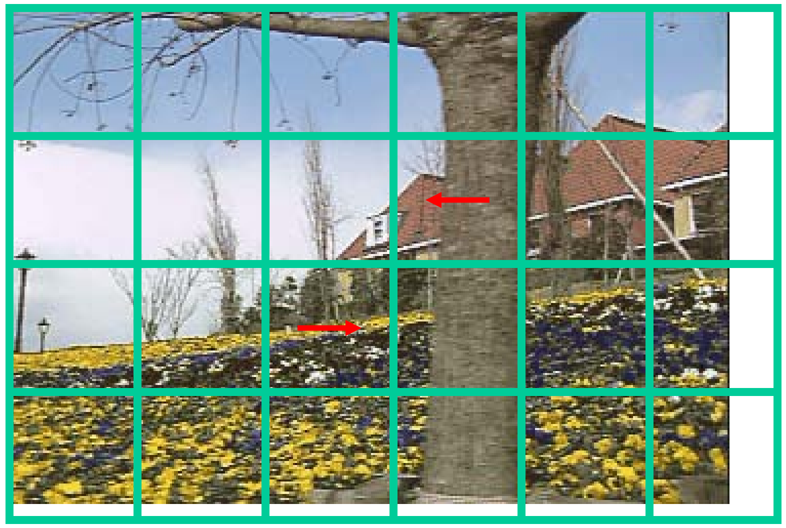


Motion Representations

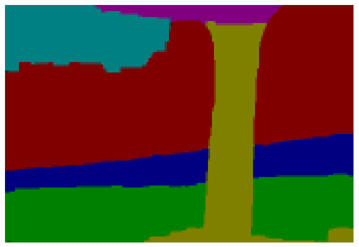
- How can we describe this scene?

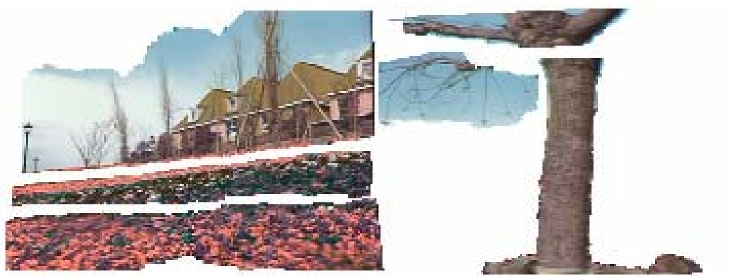
Block-based Motion Prediction

- Break image up into square blocks
- Estimate translation for each block
- Use this to predict next frame, code difference (MPEG-2)



Layered Motion

- Break image sequence up into “layers”:
- 

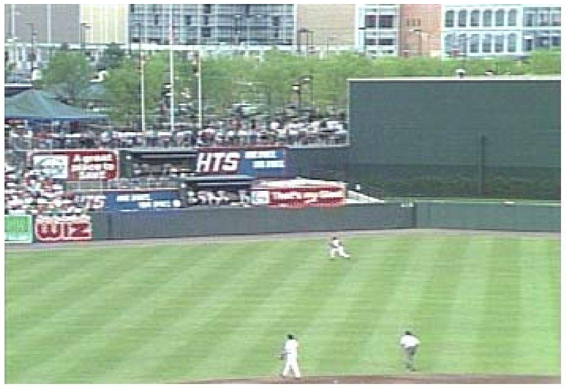


- Describe each layer’s motion (generally parametrically)

Layered Motion

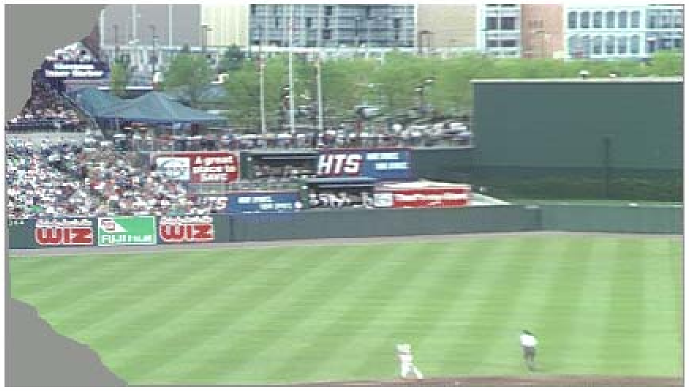
- Advantages:
  - Can represent occlusions / disocclusions
  - Each layer’s motion can be smooth
  - Video segmentation for semantic processing
- Difficulties/challenges:
  - How do we determine the correct number of layers (independent motions)?
  - How do we assign pixels to layers?
  - How do we model the motions?

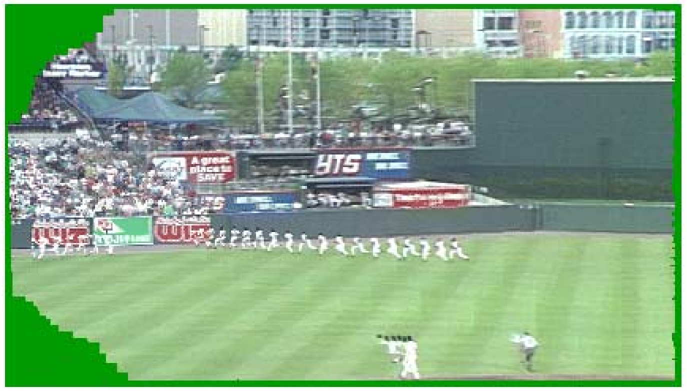
Layers for Video Summarization


Frame 0

Frame 50

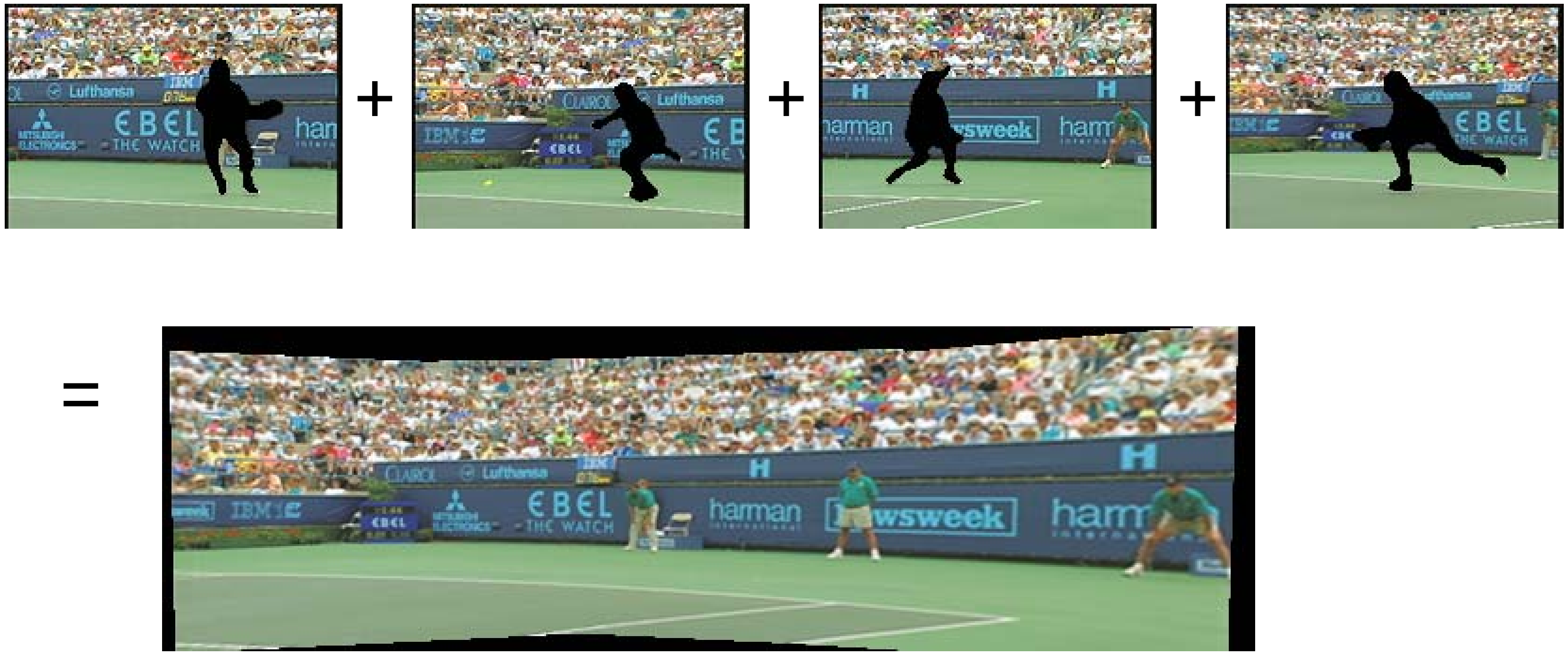
Frame 80


Background scene (players removed)


Complete synopsis of the video

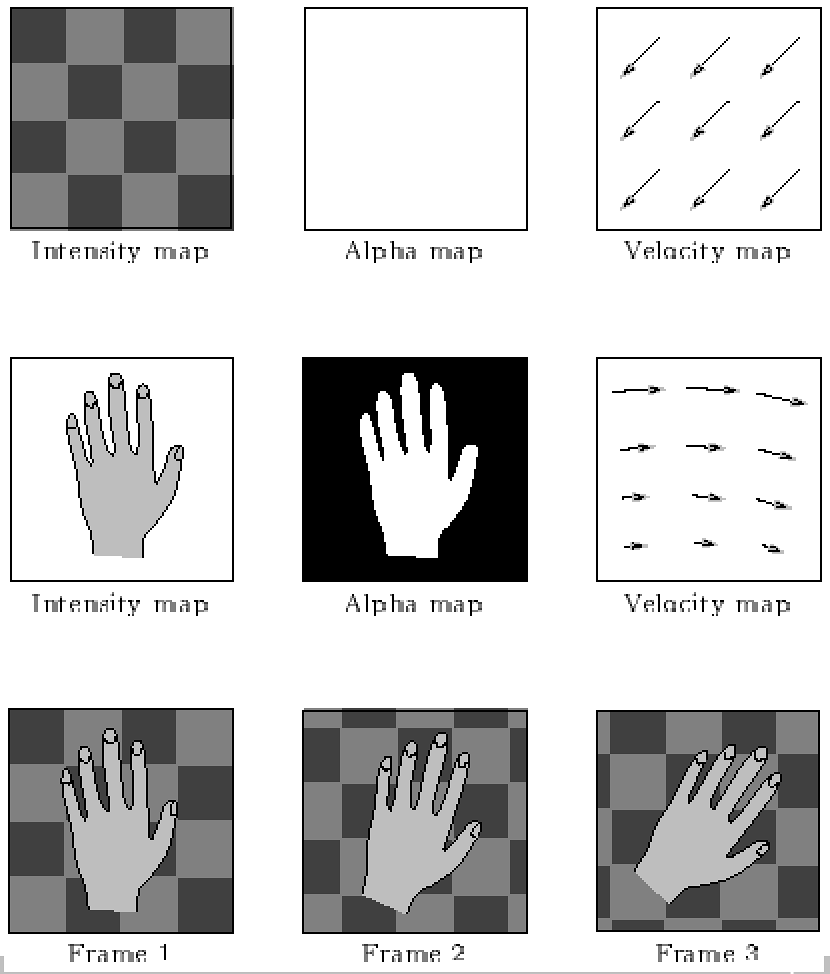
Background Modeling (MPEG-4)

- Convert masked images into a background sprite for layered video coding

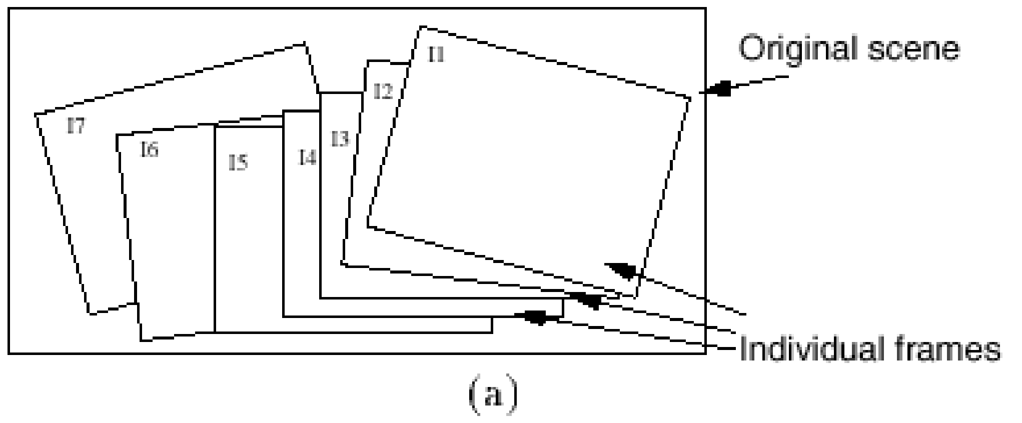


What are Layers?

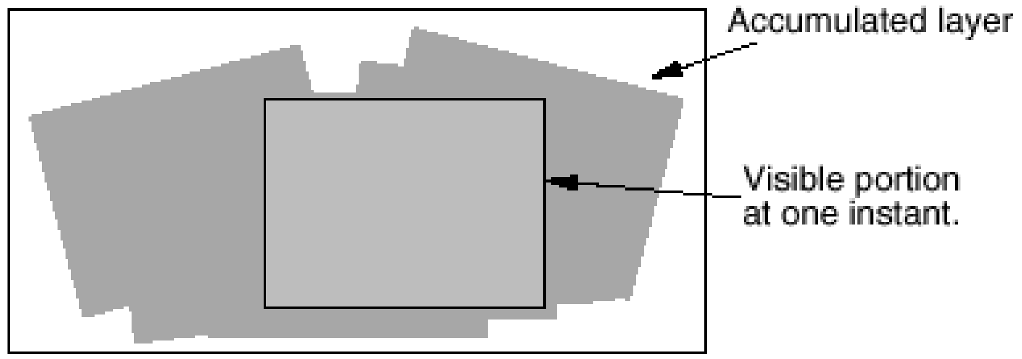
- [Wang & Adelson, 1994]
- Intensities
- Alphas
- Velocities



Forming Layers



(a)



(b)

Forming Layers

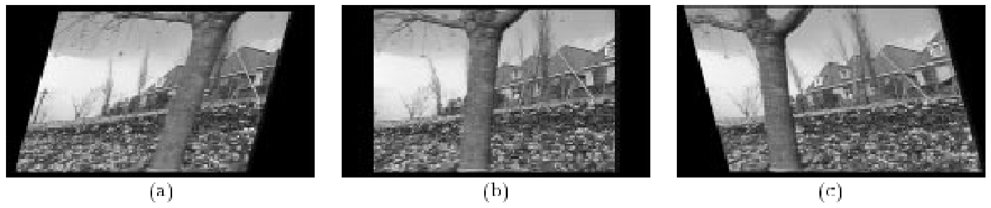


Figure 7: (a) Frame 1 warped with an affine transformation to align the flowerbed region with that of frame 15. (b) Original frame 15 used as reference. (c) Frame 30 warped with an affine transformation to align the flowerbed region with that of frame 15.

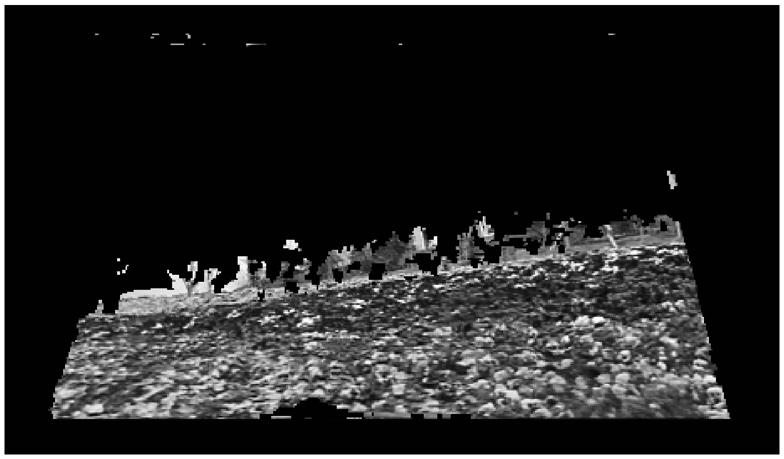
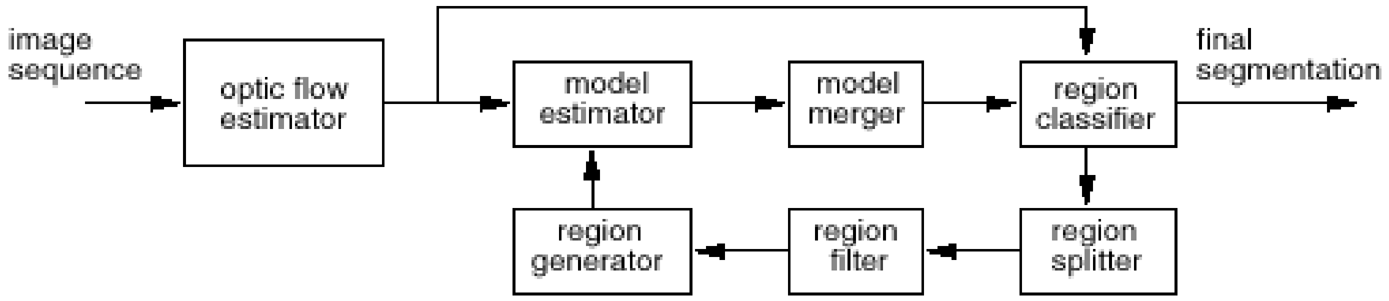


Figure 8: Accumulation of the flowerbed. Image intensities are obtained from a temporal median operation on the motion compensated images. Only the regions belonging to the flowerbed layer is accumulated in this image. Note also occluded regions are correctly recovered by accumulating data over many frames.

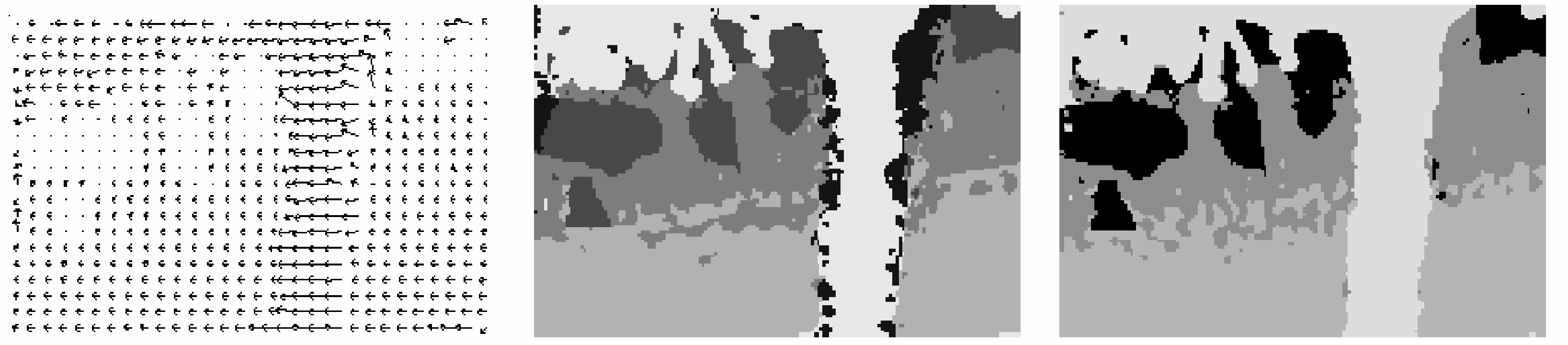
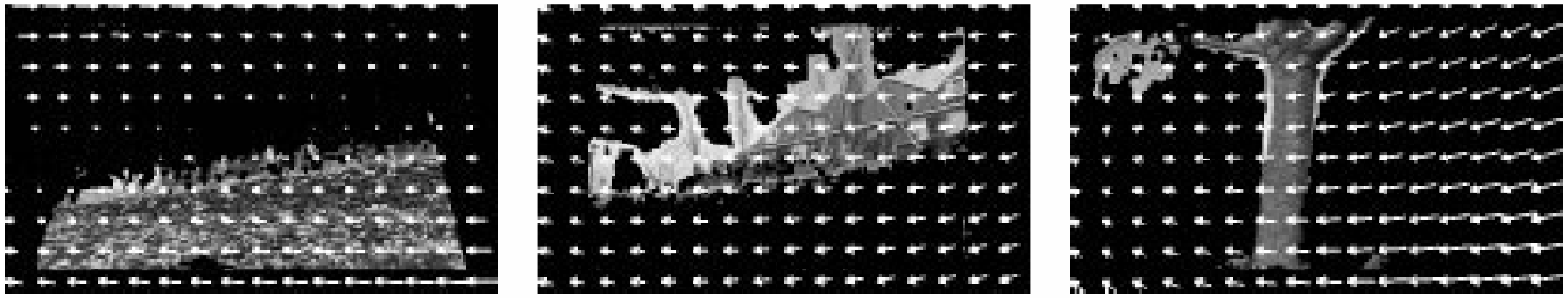
How do we estimate the layers?

1. Compute coarse-to-fine flow
2. Estimate affine motion in blocks (regression)
3. Cluster with *k-means*
4. Assign pixels to best fitting affine region
5. Re-estimate affine motions in each region



Layer Synthesis

- For each layer:
- Stabilize the sequence with the affine motion
- Compute median value at each pixel
- Determine occlusion relationships

Sparse Feature Matching

- 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”
     - Unmatched features

Feature Tracking Example

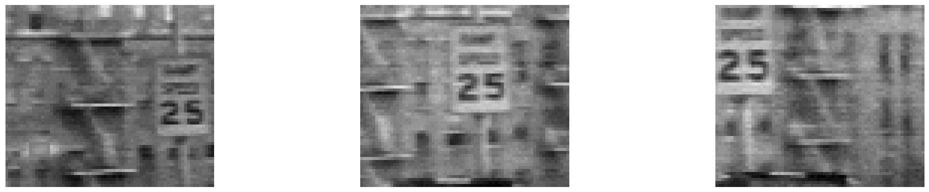


Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

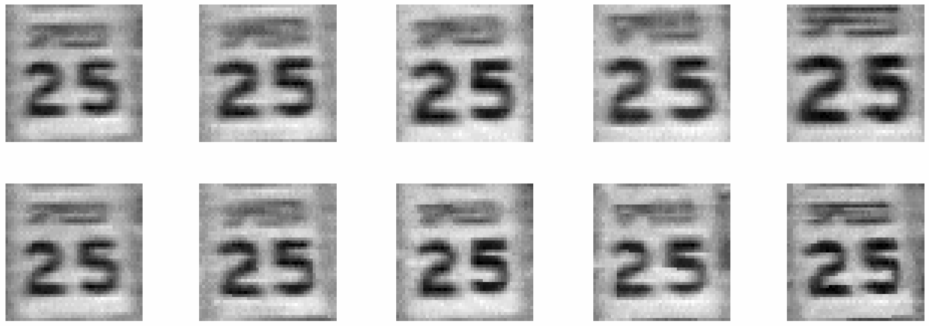


Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

Feature Tracking: Motion Models

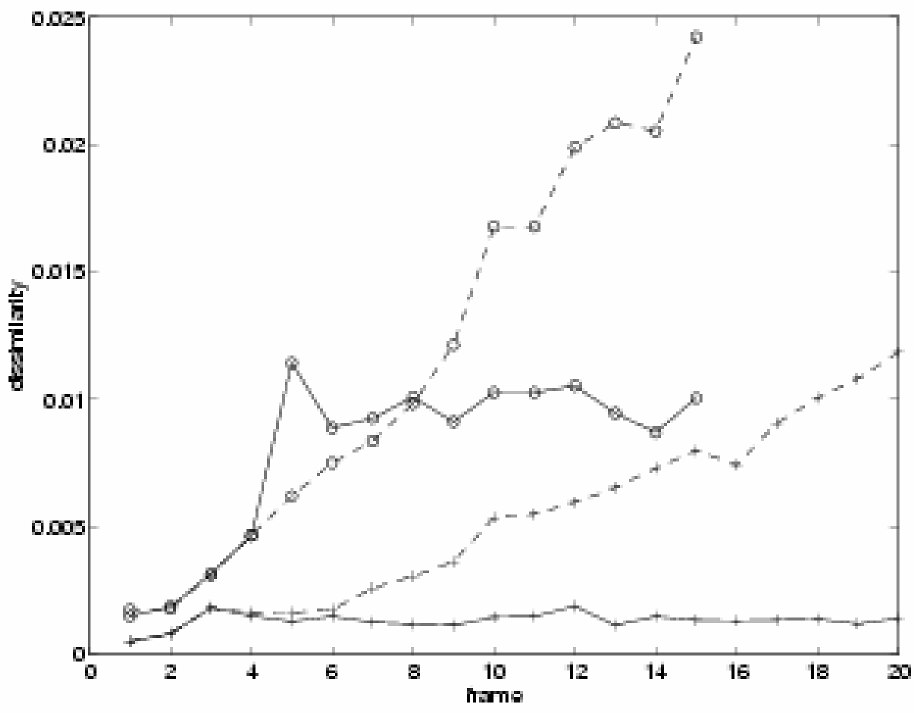


Figure 3: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 1 (plusses) and 4 (circles).

Feature Tracking Results

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

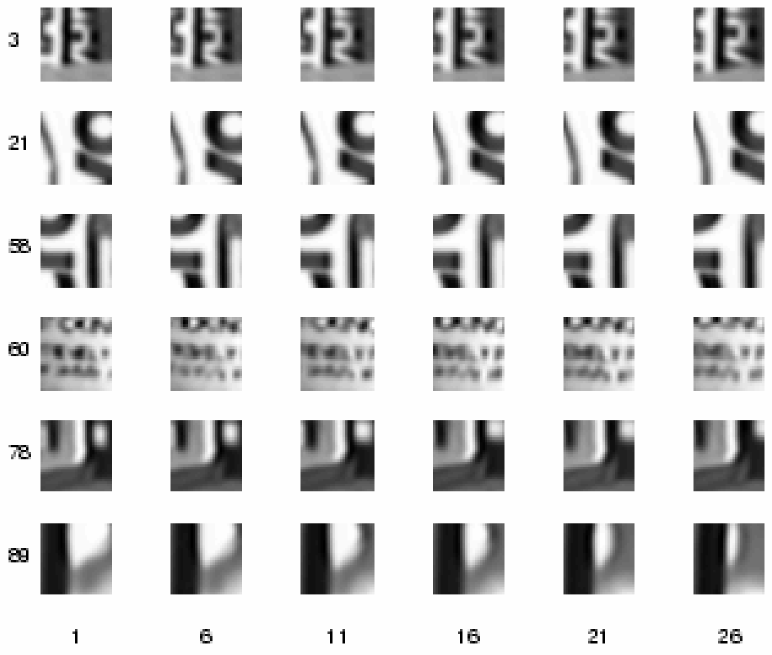


Figure 14: Six sample features through six sample frames.

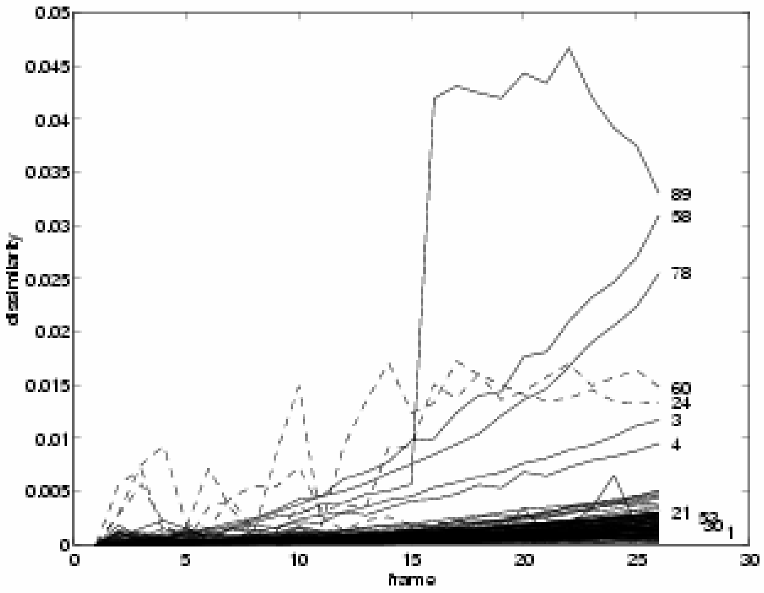


Figure 15: Affine motion dissimilarity for the features in figure 11. Notice the good discrimination between good and bad features. Dashed plots indicate aliasing (see text).

Features 24 and 60 deserve a special discussion, and