CS 4495 Computer Vision
Motion Models

Aaron Bobick

School of Interactive Computing

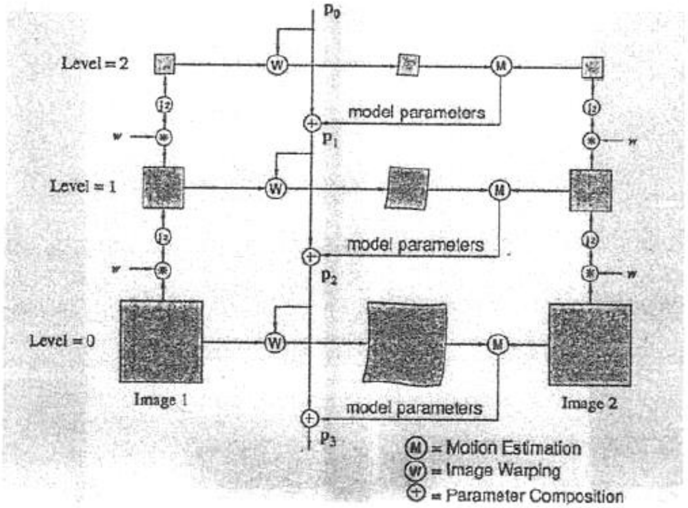


Fig. 1. Diagram of the hierarchical motion estimation framework.

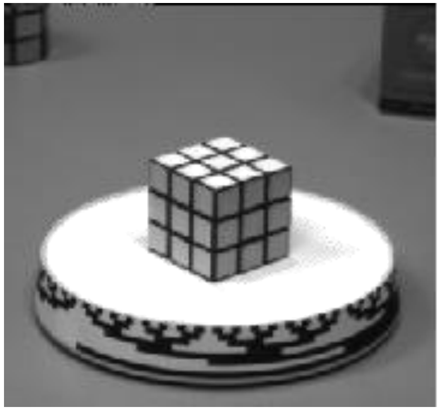
Outline

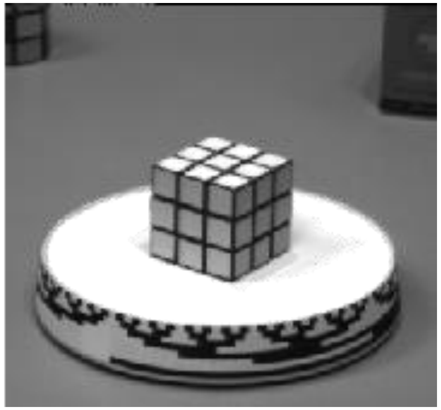
- Last time: dense motion: optic flow
  - Brightness constancy constraint equation
  - Lucas-Kanade
- 2D Motion models
  - Bergen, '92
  - Pyramids
  - Layers
- Motion fields from 3D motions
- Parametric motion

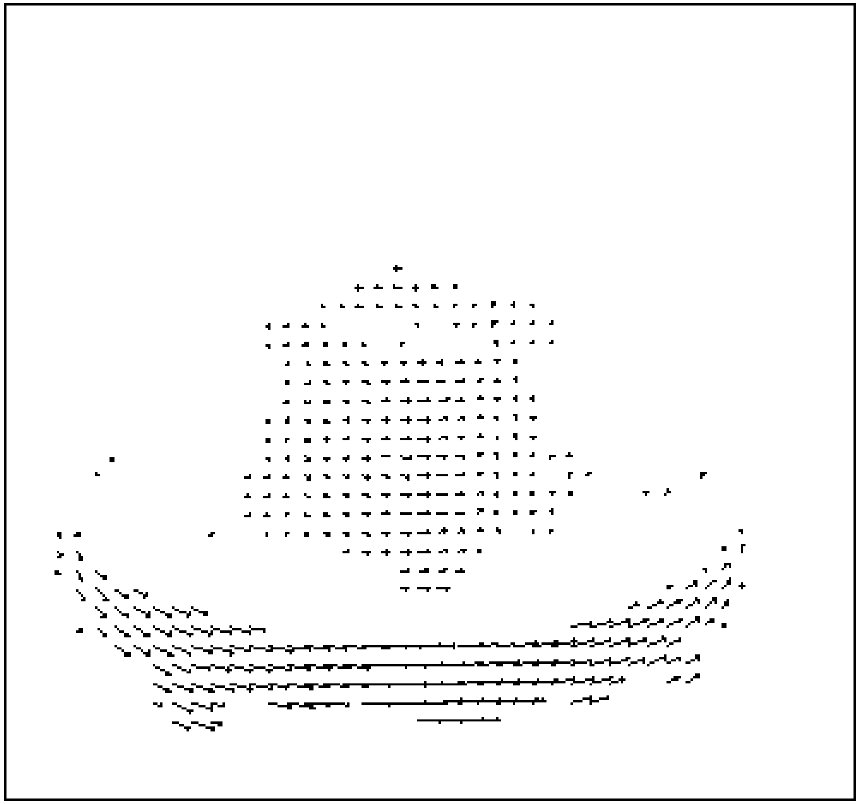
Visual motion

Many slides adapted from S. Seitz, R. Szeliski, M. Pollefeys, K. Grauman and others…

Motion estimation: Optical flow

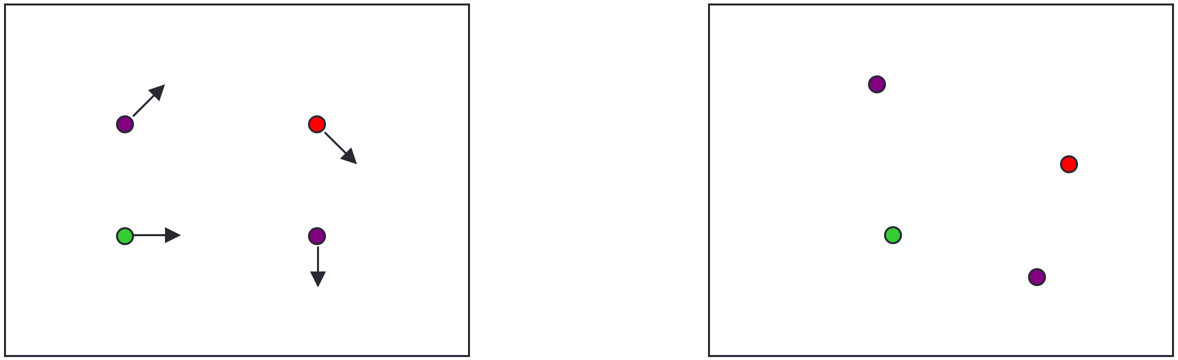






Will start by estimating motion of each pixel separately
Then will consider motion of entire image

Problem definition: optical flow



$I(x, y, t)$ $I(x, y, t+1)$

How to estimate pixel motion from image $I(x,y,t)$ to $I(x,y,t)$?

- Solve pixel correspondence problem
  - given a pixel in I(x,y,t), look for nearby pixels of the same color in *I(x,y,t+1)*

Key assumptions

- color constancy: a point in *I(x,y,* looks the same in I(x,y,t+1)
  - For grayscale images, this is brightness constancy
- small motion: points do not move very far

This is called the optical flow problem

Optical flow equation

- Combining these two equations

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$
\begin{aligned}
0 &= I(x+u, y+v, t+1) - I(x, y, t) \\
&\approx I(x, y, t+1) + I_x u + I_y v - I(x, y, t) \\
&\approx [I(x, y, t+1) - I(x, y)] + I_x u + I_y v \\
&\approx I_t + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot \langle u, v \rangle
\end{aligned}
$$

In the limit as u and v go to zero, this becomes exact

$$0 = I_t + \nabla I \cdot \langle u, v \rangle$$

Brightness constancy constraint equation

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

Optical flow equation

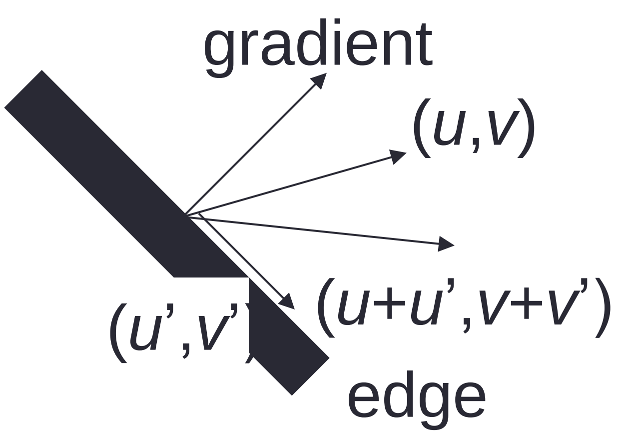
$$0 = I_t + \nabla I \cdot \langle u, v \rangle \quad \text{or} \quad I_x u + I_y v + I_t = 0$$

- Q: how many unknowns and equations per pixel?

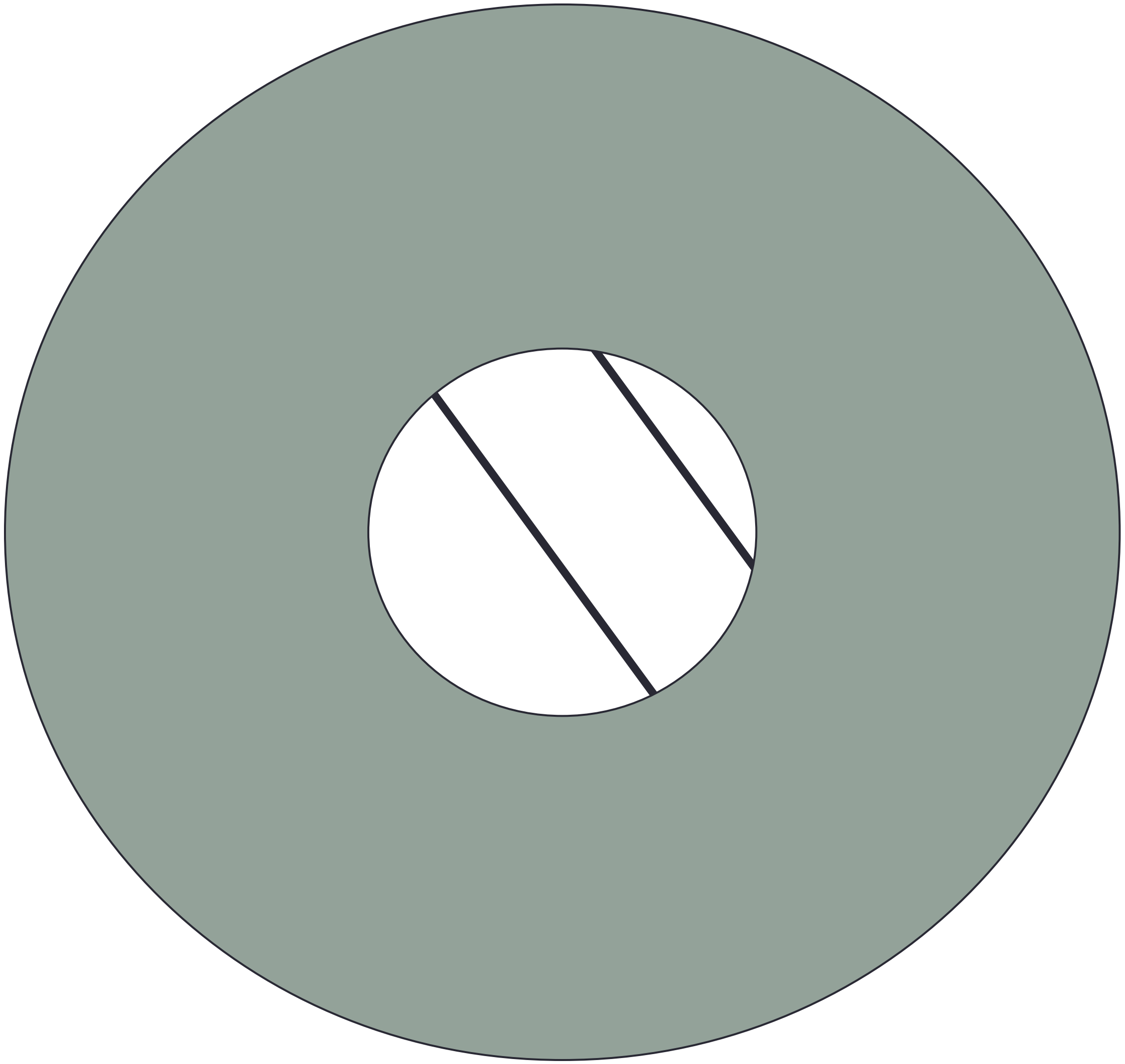
2 unknowns, one equation

Intuitively, what does this constraint mean?

- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown



Aperture problem



Solving the aperture problem

- How to get more equations for a pixel?
  - Basic idea: impose additional constraints
    - most common is to assume that the flow field is smooth locally
    - one method: pretend the pixel's neighbors have the same (u,v)
      - If we use a 5x5 window, that gives us 25 equations per pixel!

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \;\; v]$$

$$\underset{25\times 2}{\underbrace{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}}_{A}} \underset{2\times 1}{\underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d}} = - \underset{25\times 1}{\underbrace{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}_{b}}$$

Lukas-Kanade flow

- Prob: we have more equations than unknowns

$$\underset{25\text{x}2}{A}\ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

$$\underset{2\text{x}2}{(A^T A)}\ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = -\underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)

Eigenvectors of $A^T A$

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

- Recall the Harris corner detector: $M = A^T A$ is the *second moment matrix*
- The eigenvectors and eigenvalues of M relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it

Violating assumptions in Lucas-Kanade

- The motion is large (larger than a pixel)
  - Not-linear: Iterative refinement
  - Local minima: coarse-to-fine estimation
- A point does not move like its neighbors
  - Motion segmentation
- Brightness constancy does not hold
  - Do exhaustive neighborhood search with normalized correlation - tracking features – maybe SIFT – more later….

Violating assumptions in Lucas-Kanade

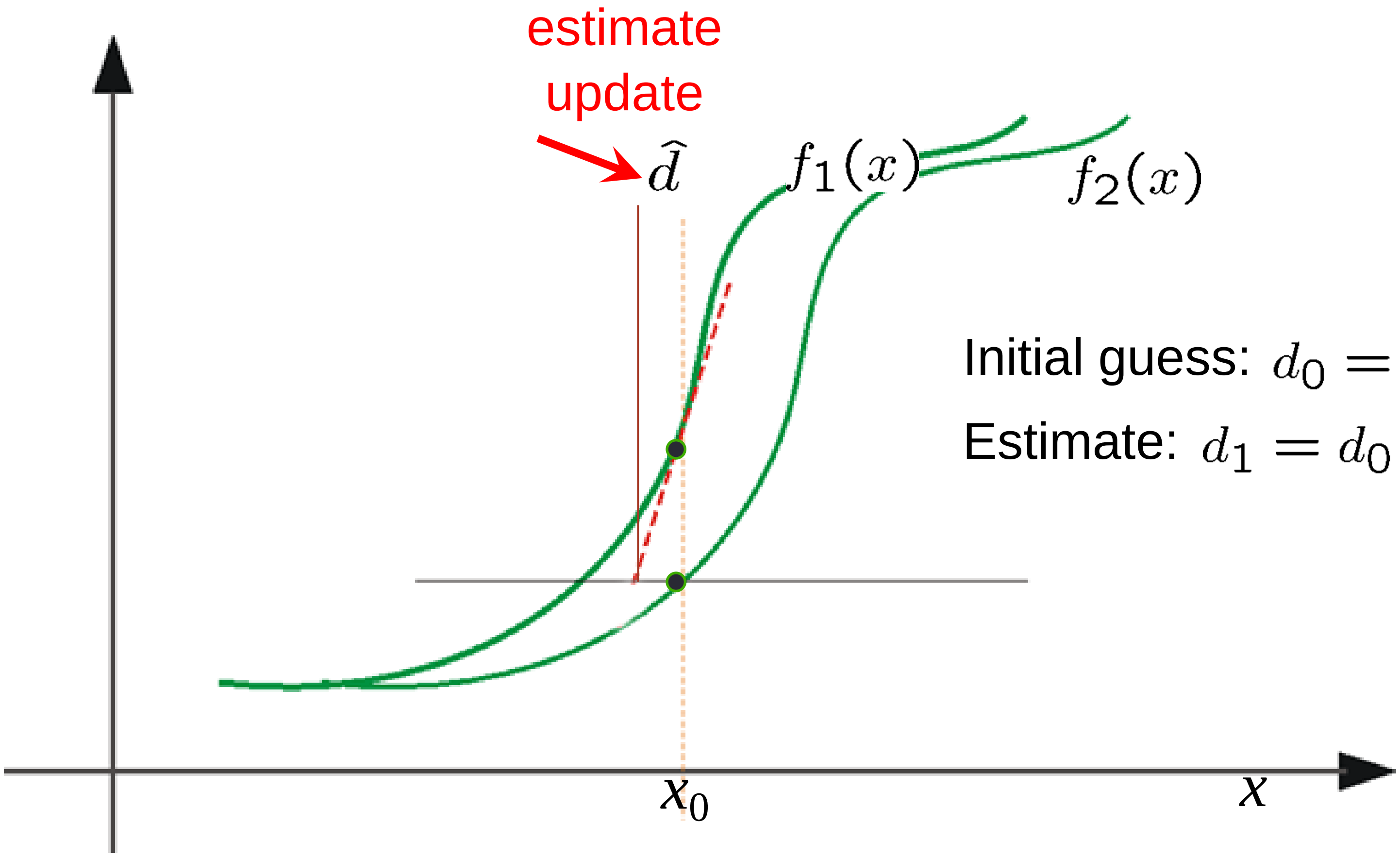
- The motion is large (larger than a pixel)
  - Not-linear: Iterative refinement
  - Local minima: coarse-to-fine estimation
- A point does not move like its neighbors
  - Motion segmentation
- Brightness constancy does not hold
  - Do exhaustive neighborhood search with normalized correlation - tracking features – maybe SIFT – more later….

Not tangent: Iterative Refinement

Iterative Lukas-Kanade Algorithm

1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp I_t towards I_{t+1} using the estimated flow field
    - *- use image warping techniques*
3. Repeat until convergence

Optical Flow: Iterative Estimation

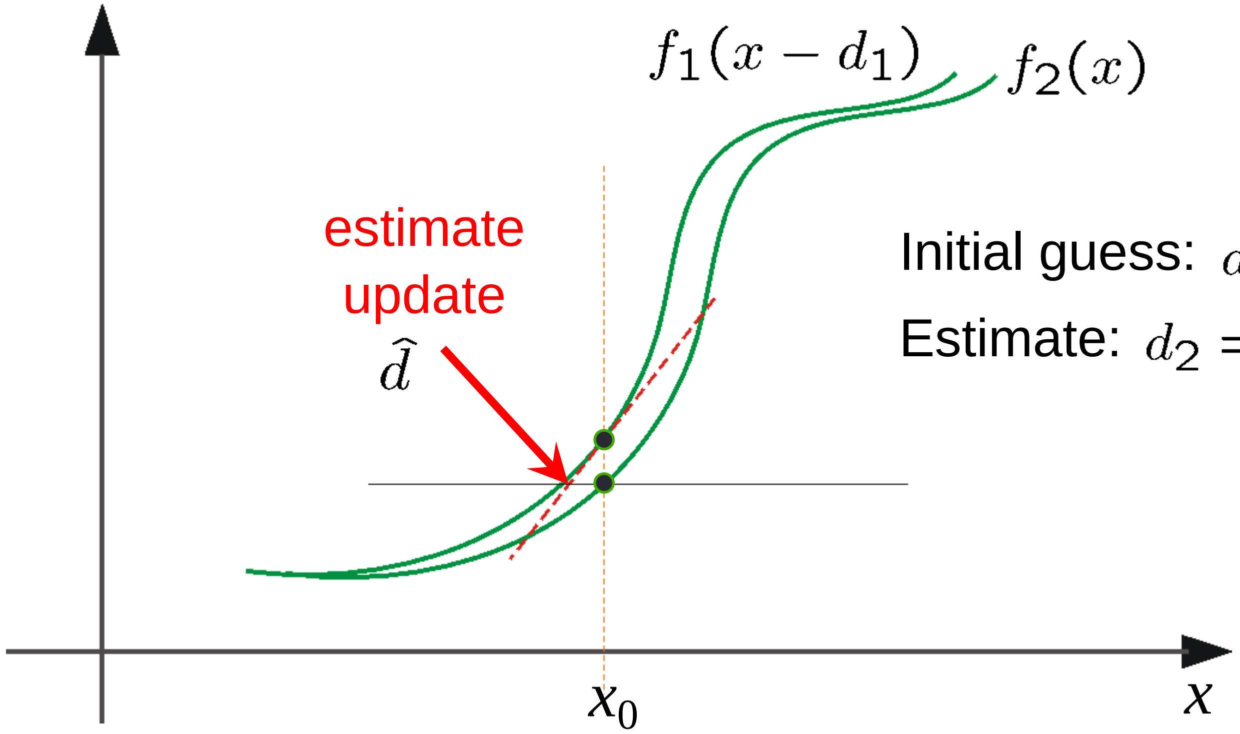


Initial guess: $d_0 = 0$

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

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

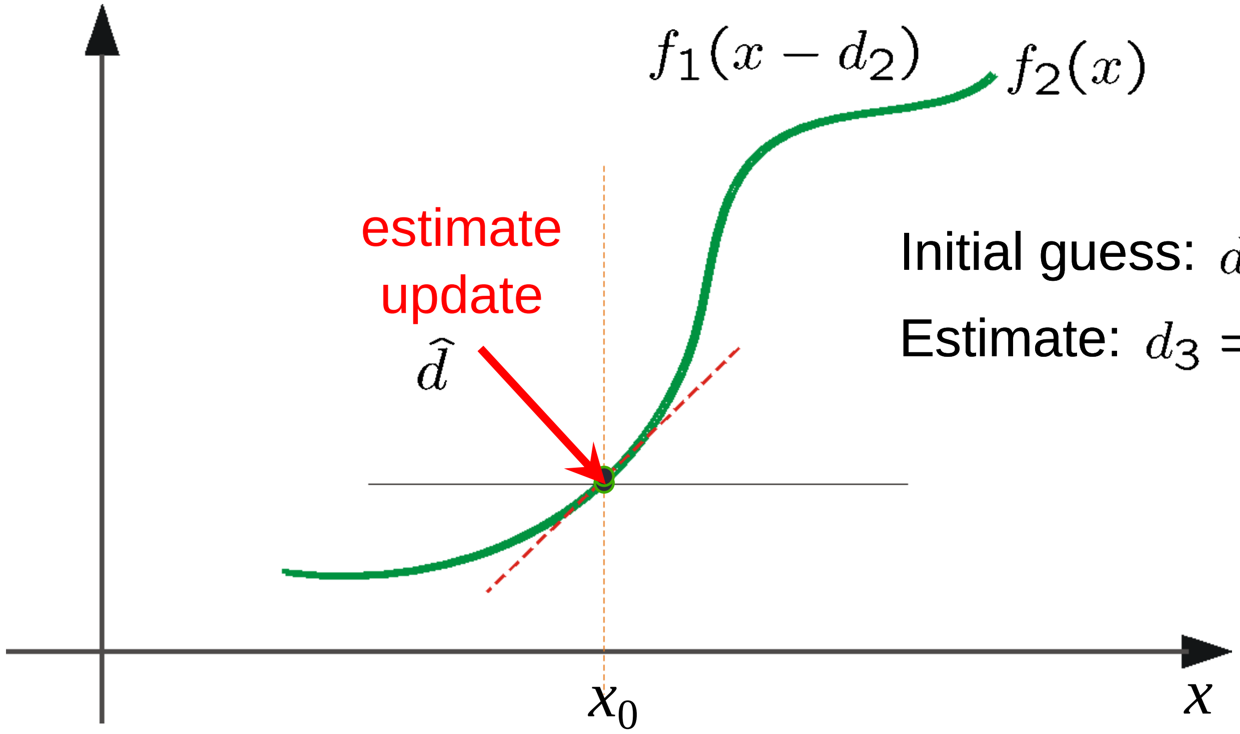
Optical Flow: Iterative Estimation



Initial guess: d_1

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

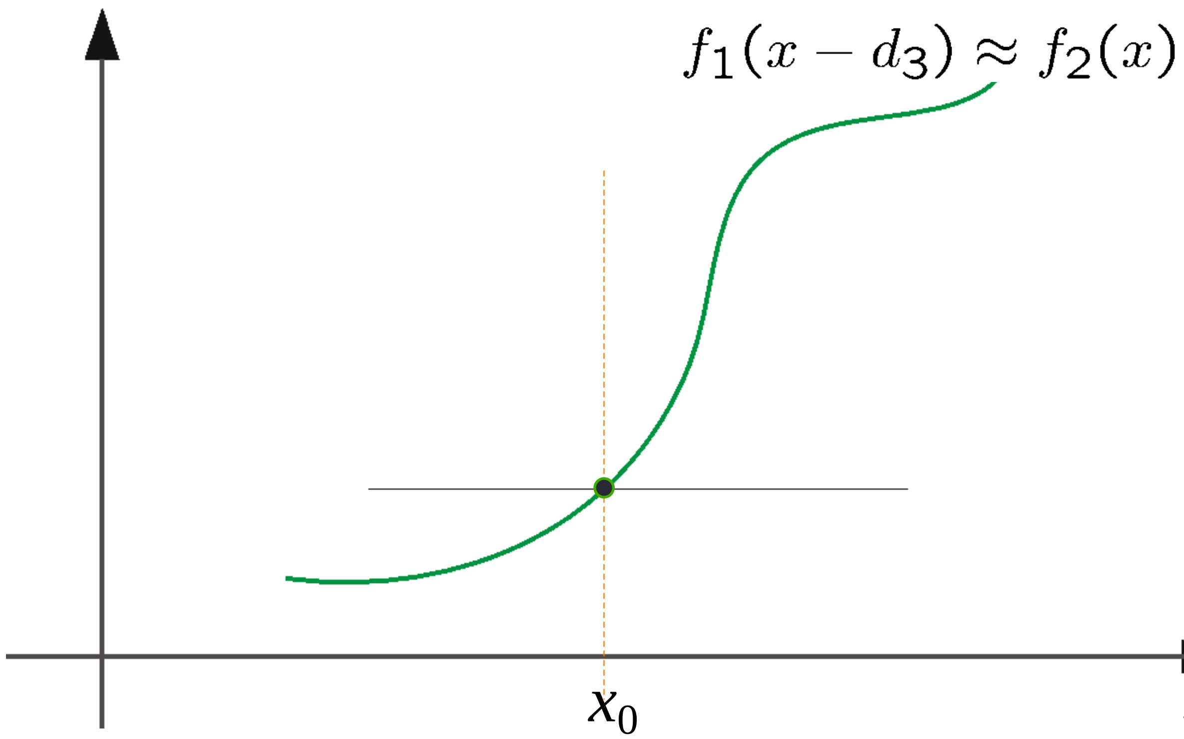
Optical Flow: Iterative Estimation



Initial guess: d_2

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

Optical Flow: Iterative Estimation



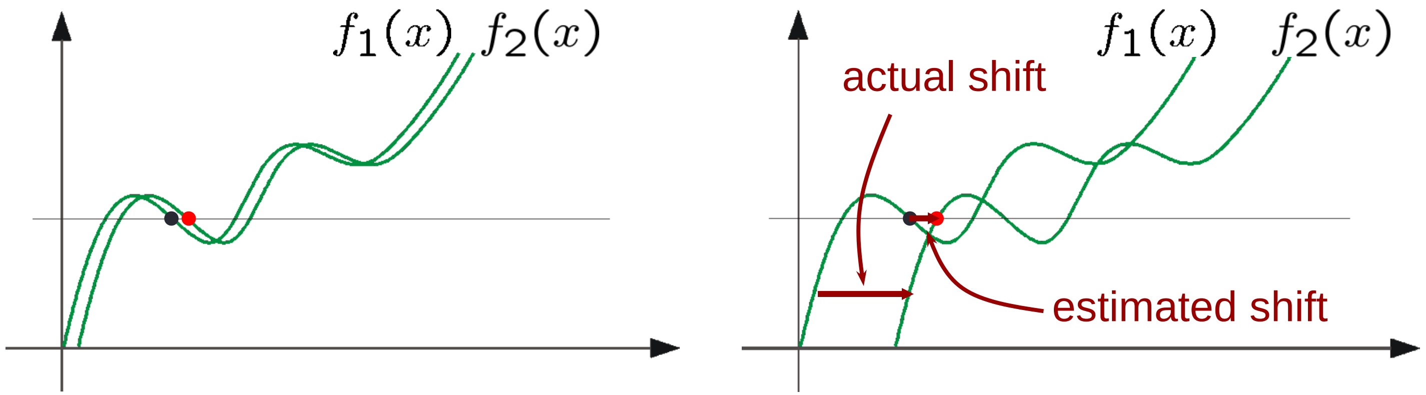
Revisiting the small motion assumption

- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)
  - How might we solve this problem?

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?

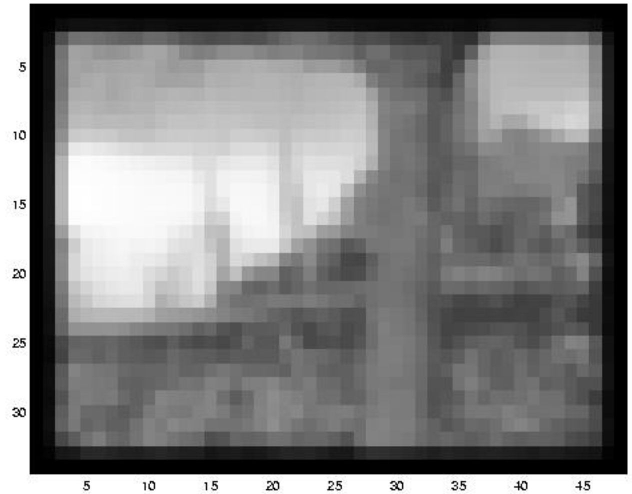


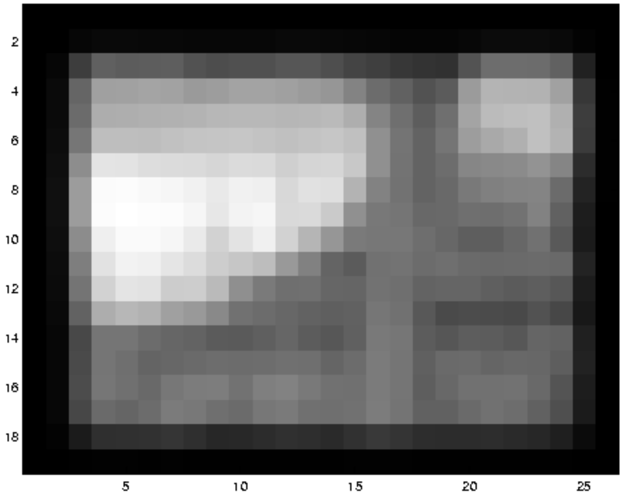
nearest match is correct (no aliasing)

nearest match is incorrect (aliasing)

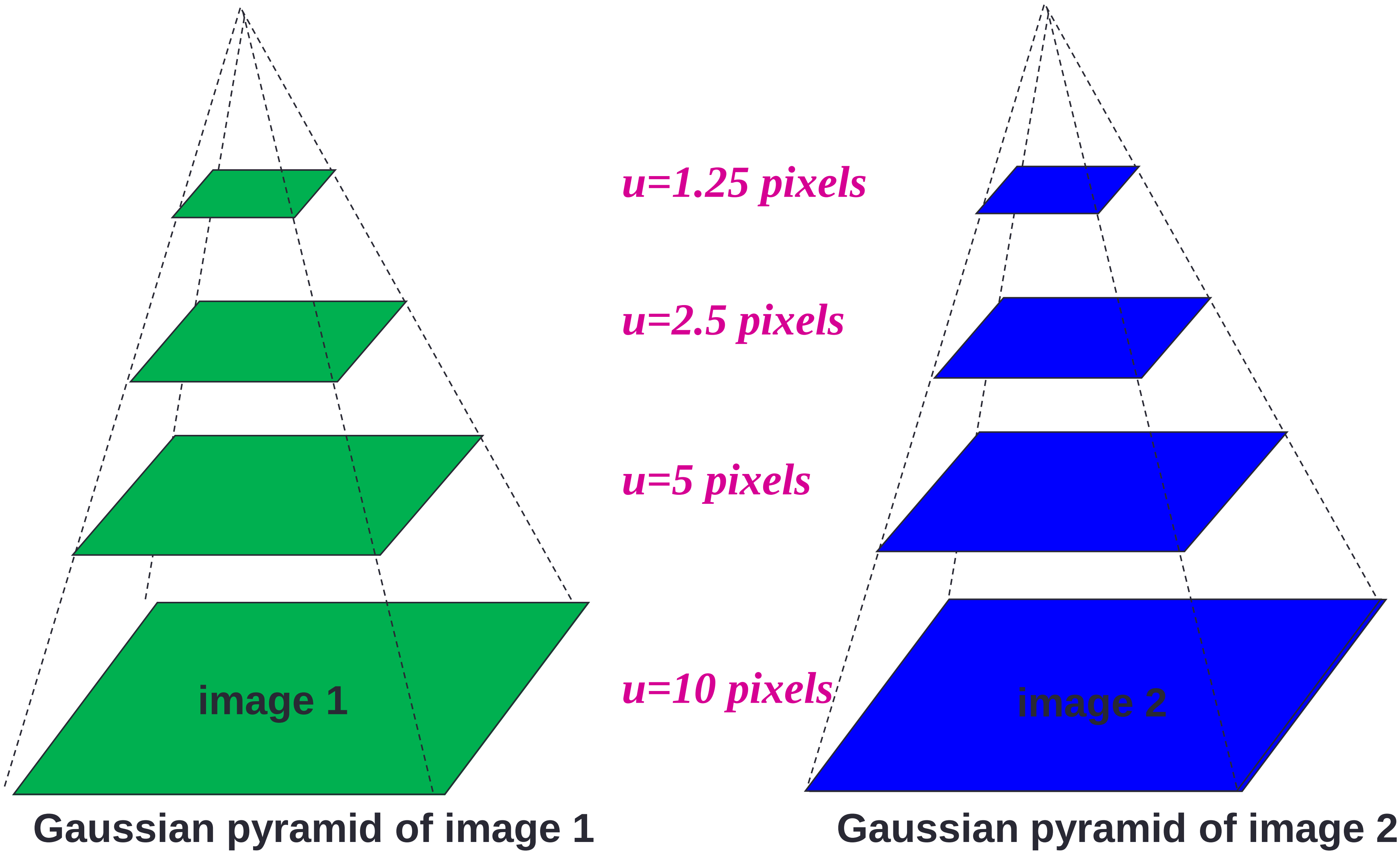
To overcome aliasing: coarse-to-fine estimation.

Reduce the resolution!

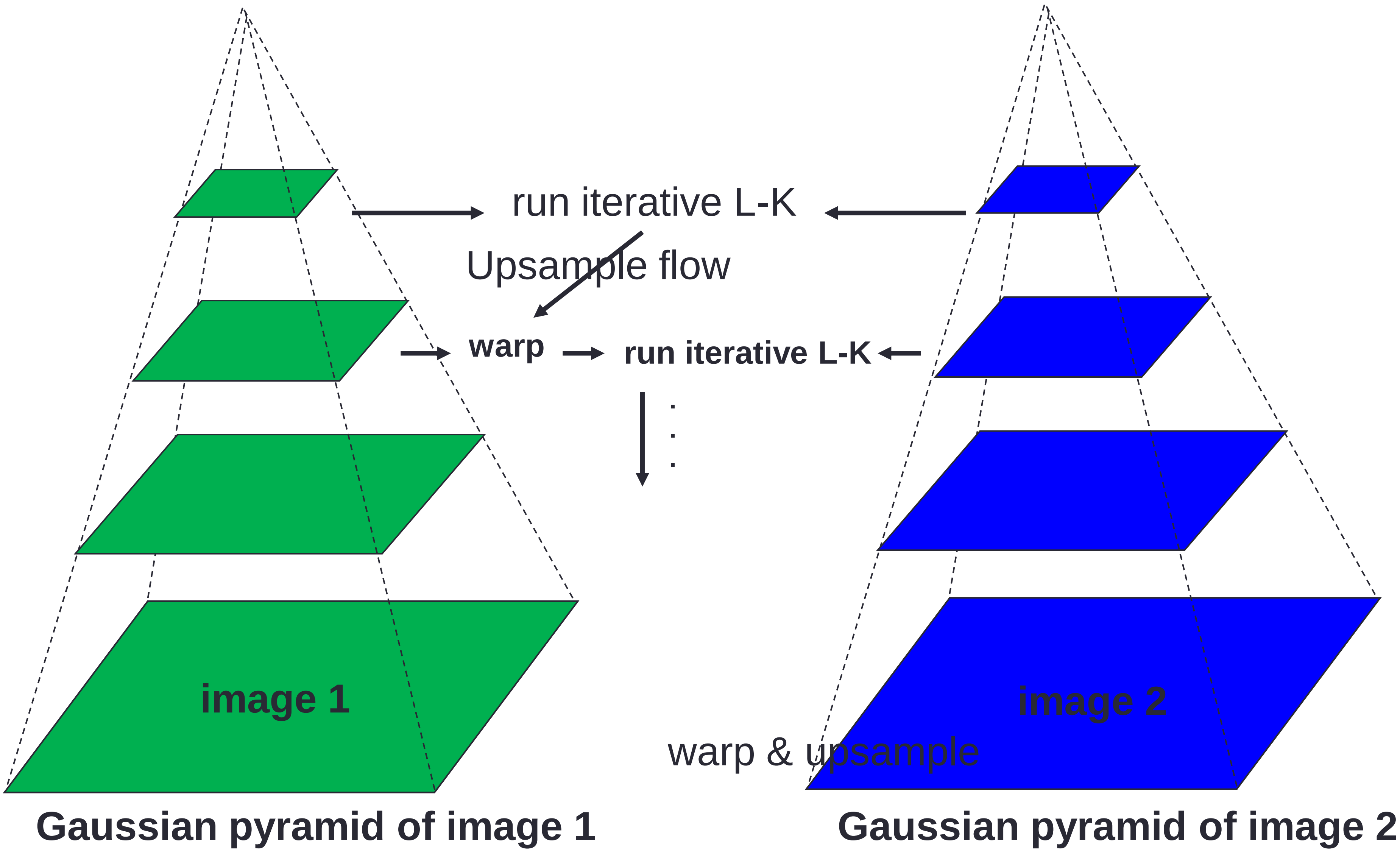




Coarse-to-fine optical flow estimation

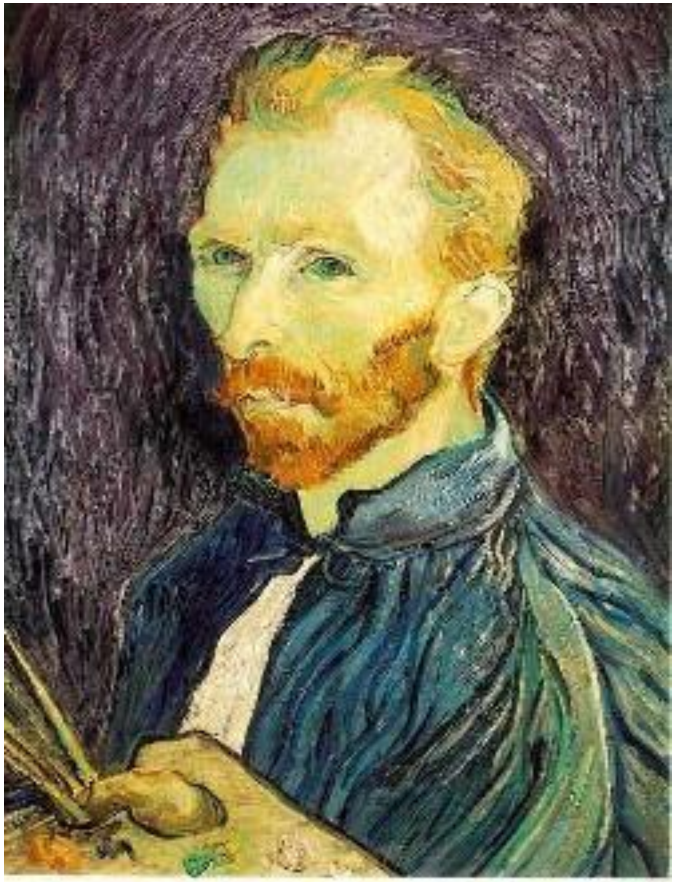


Coarse-to-fine optical flow estimation



Multi-scale

Remember: Image sub-sampling



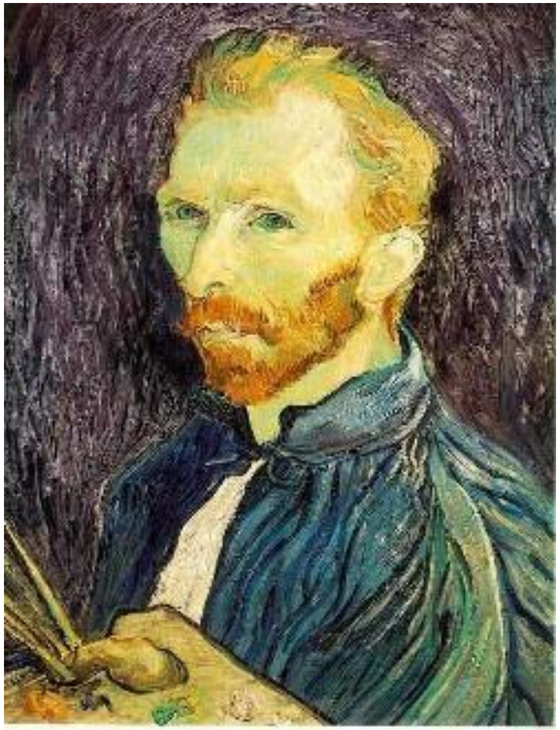
1/4

1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

S. Seitz

Bad image sub-sampling



1/2

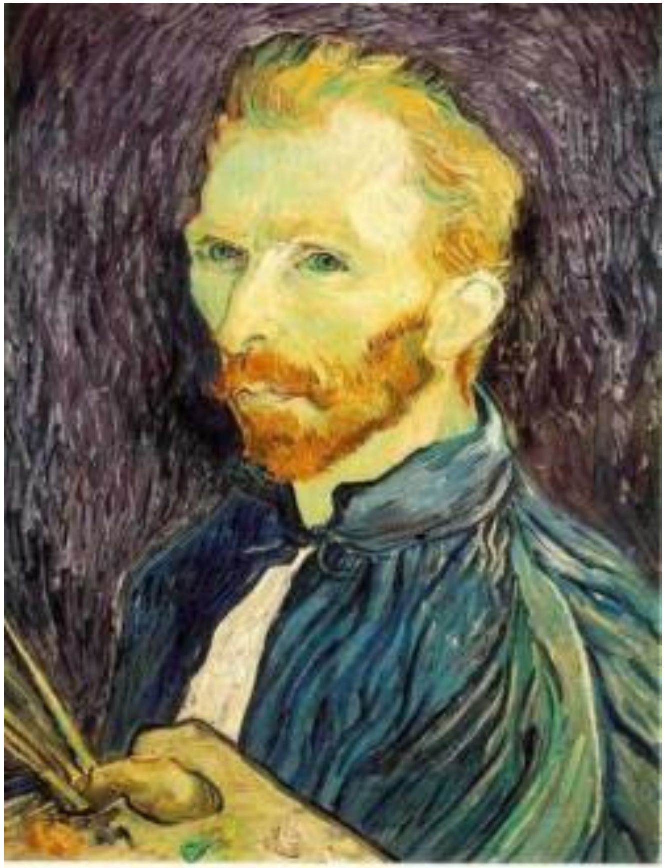
1/4 (2x zoom)

1/8 (4x zoom)

Aliasing! What do we do?

S. Seitz

Gaussian (lowpass) pre-filtering



Gaussian 1/2

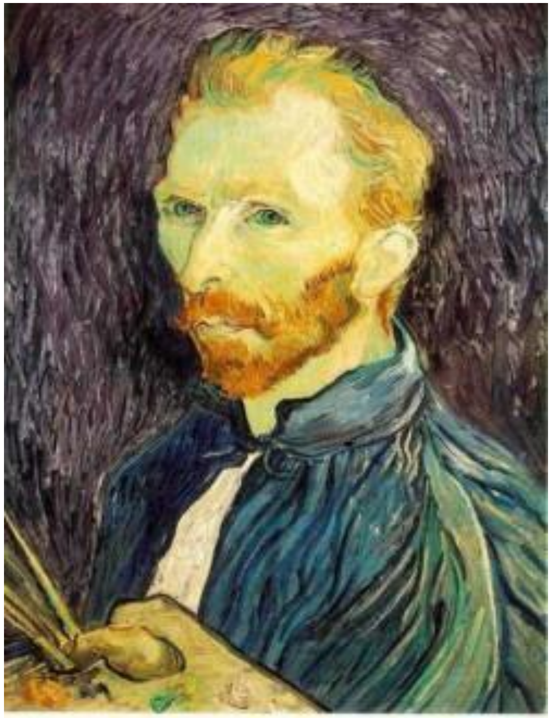
G 1/4

G 1/8

Solution: filter the image, *then* subsample

S. Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2

G 1/4

G 1/8

S. Seitz

Band-pass filtering

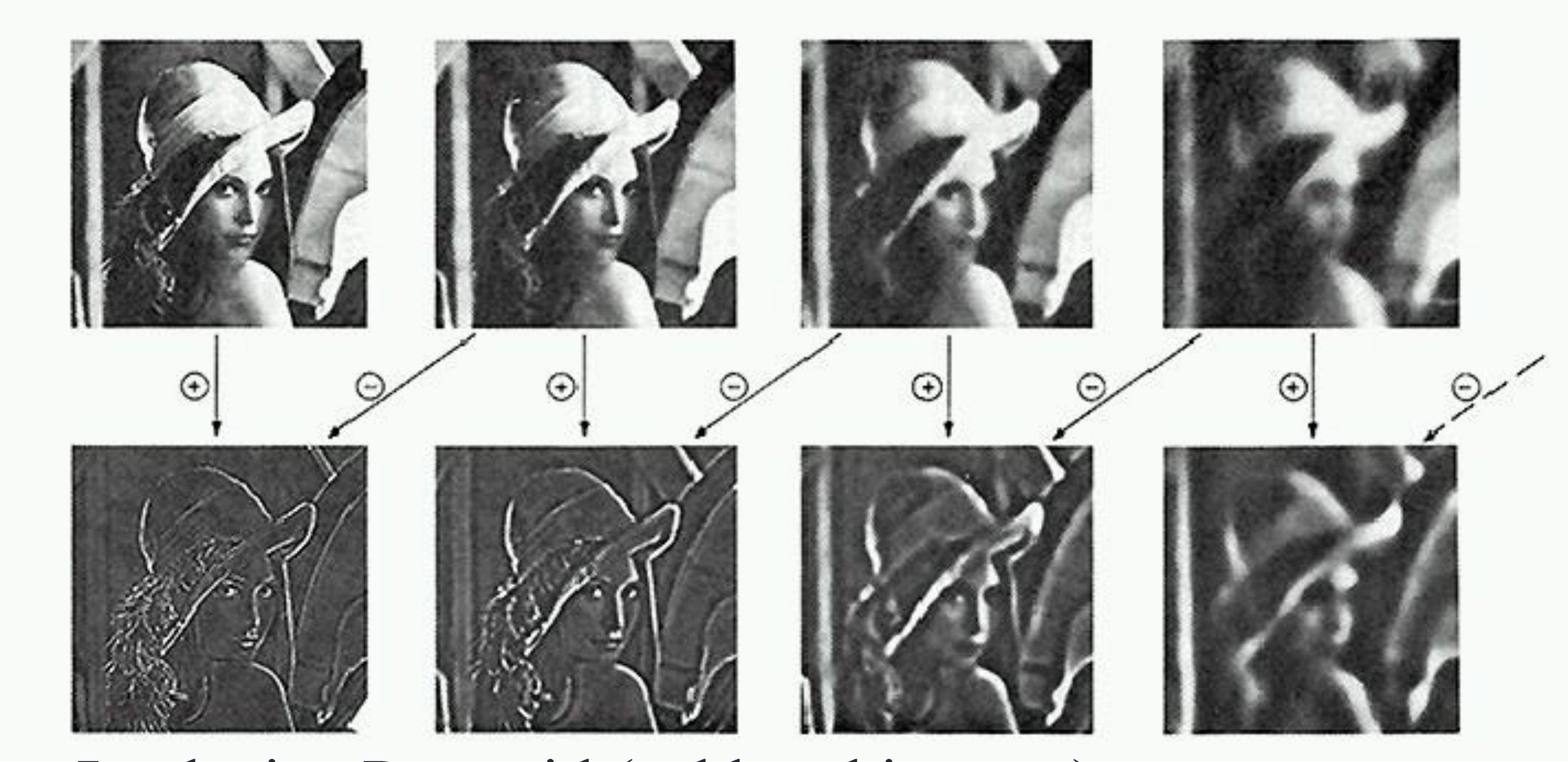
Gaussian Pyramid (low-pass images)

Laplacian Pyramid (subband images)

These are "bandpass" images (almost).

Band-pass filtering

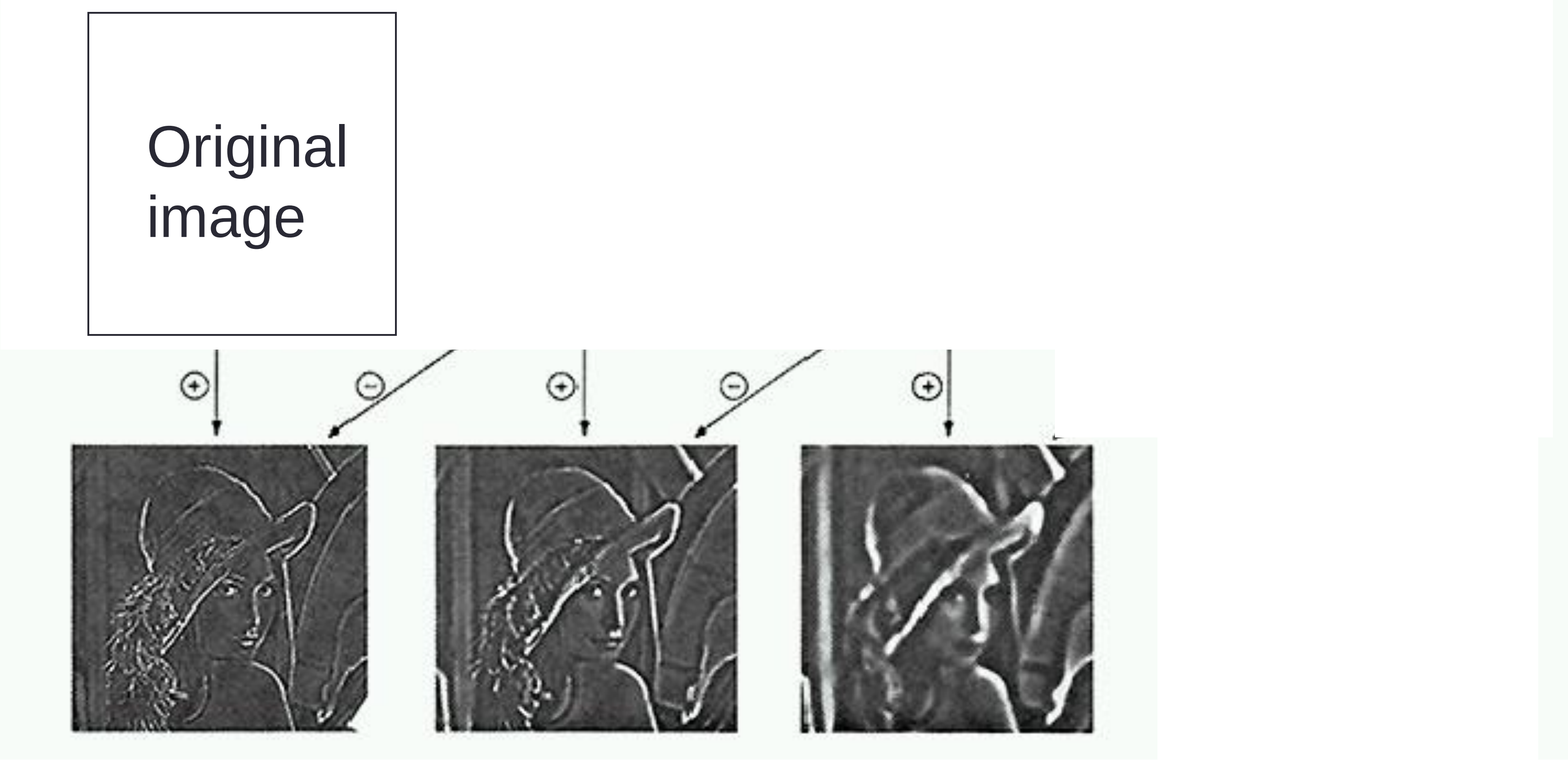
Gaussian Pyramid (low-pass images)



Laplacian Pyramid (subband images)

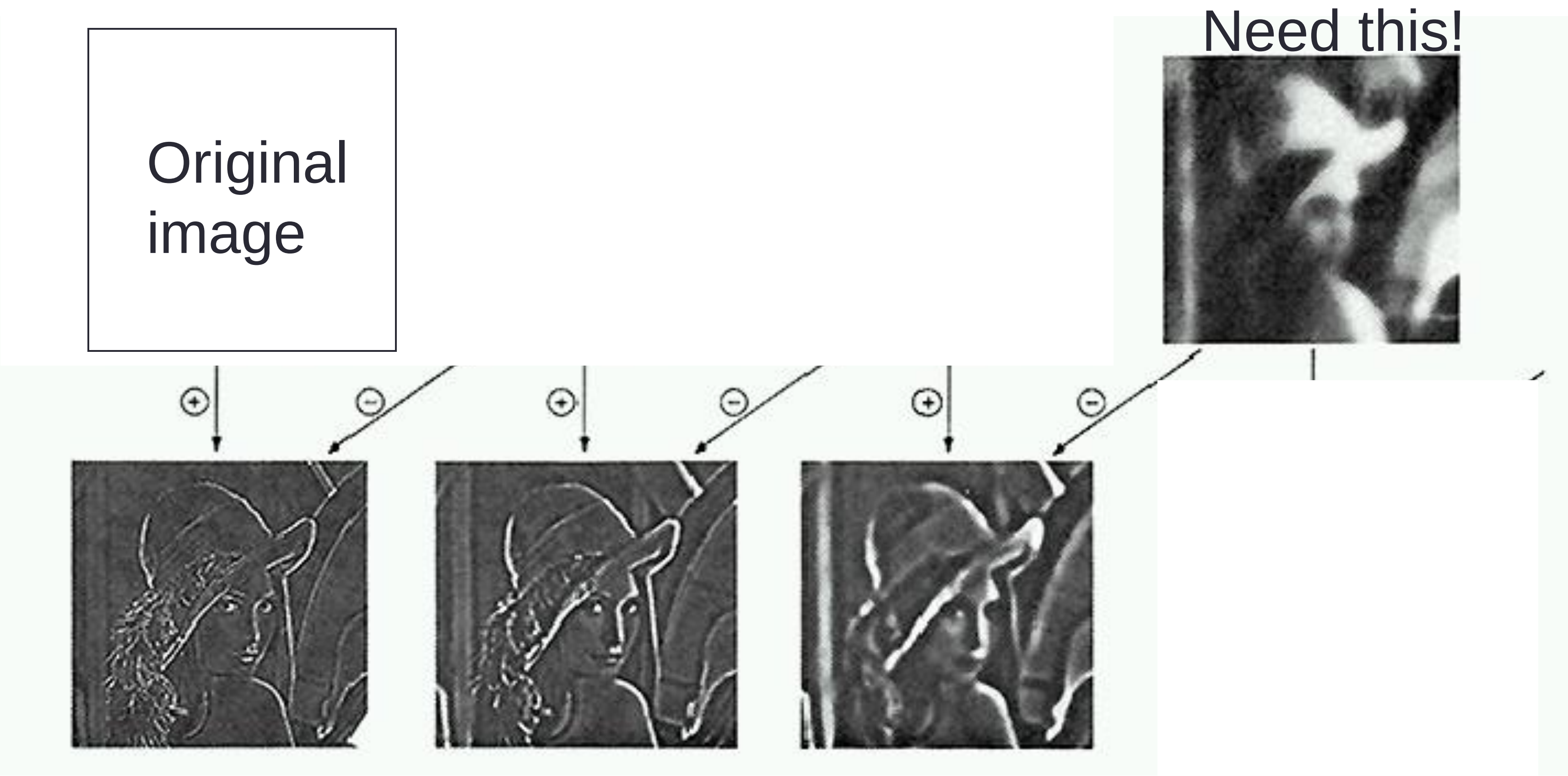
These are "bandpass" images (almost).

Laplacian Pyramid



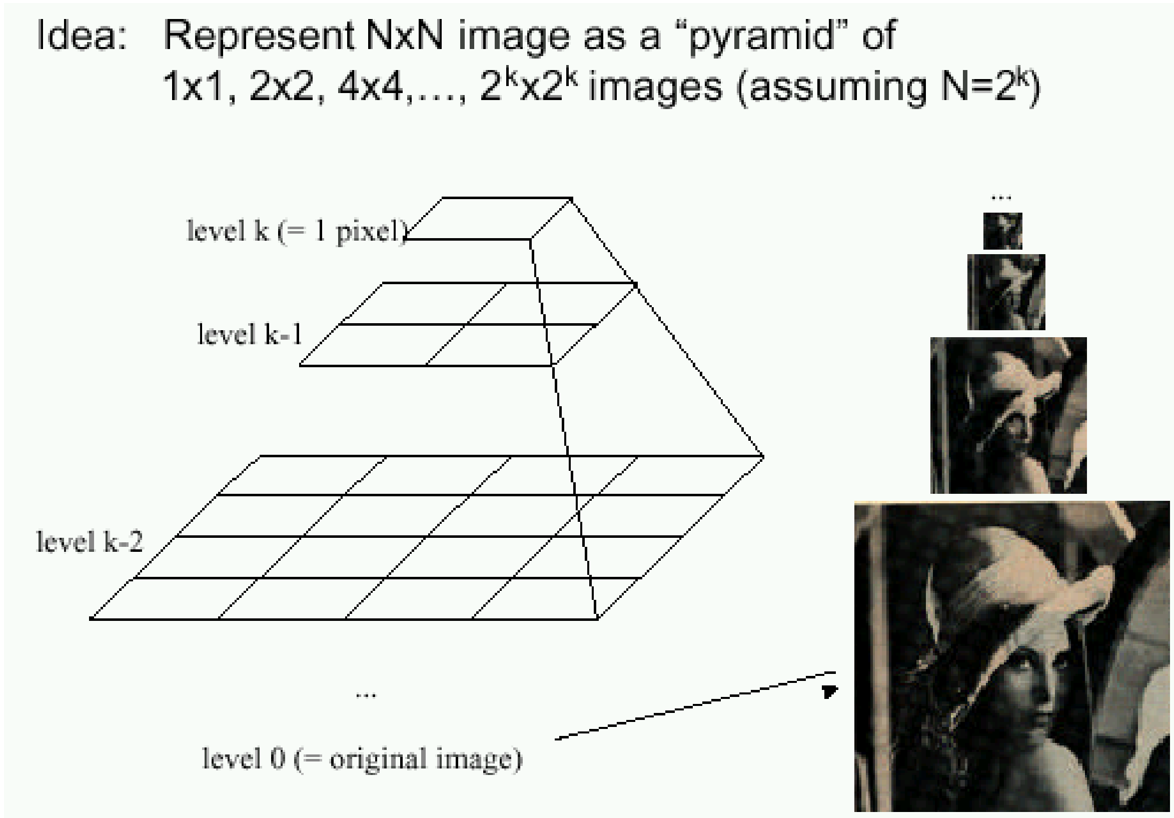
- How can we reconstruct (collapse) this pyramid into the original image?

Laplacian Pyramid



- How can we reconstruct (collapse) this pyramid into the original image?

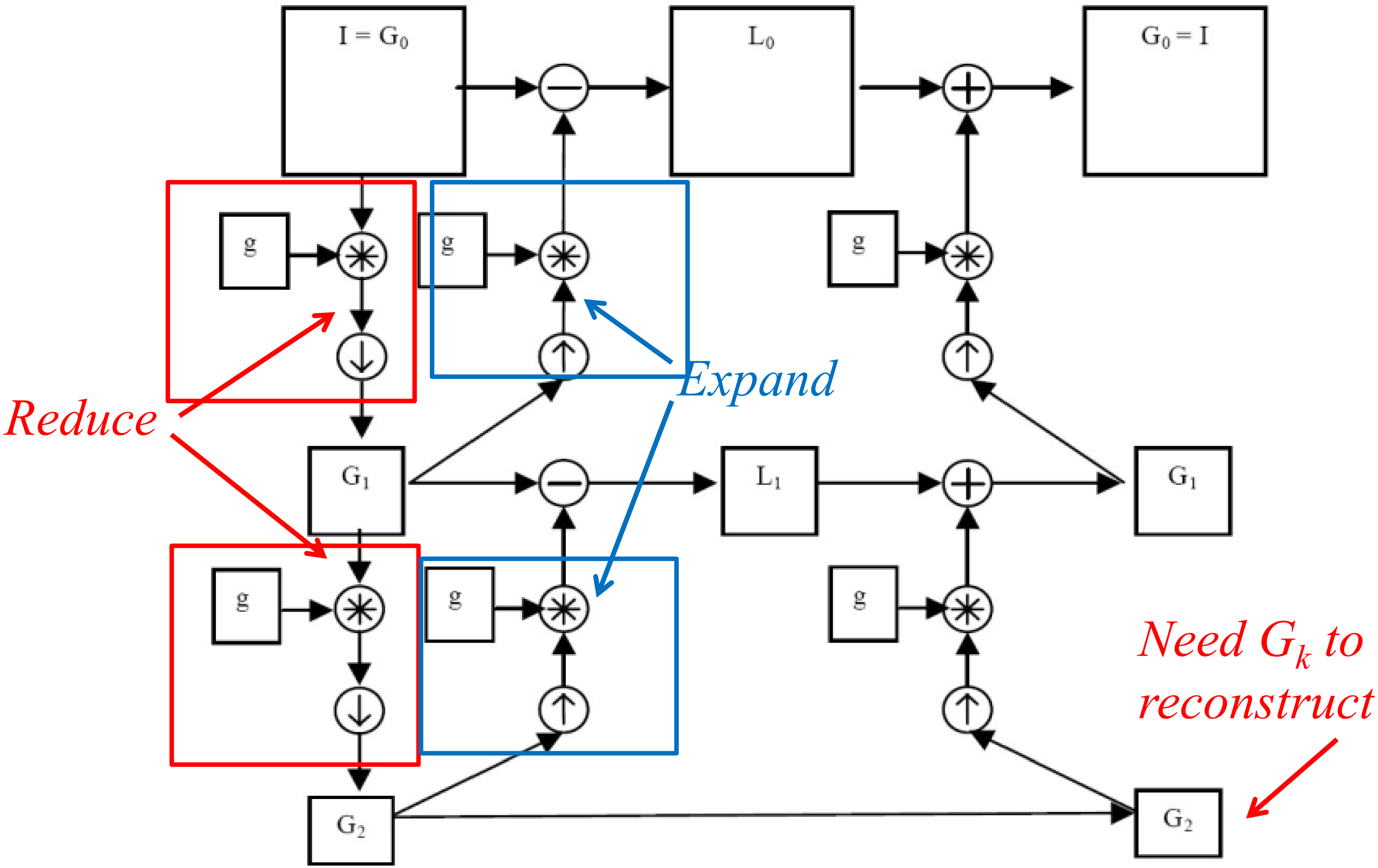
Image Pyramids



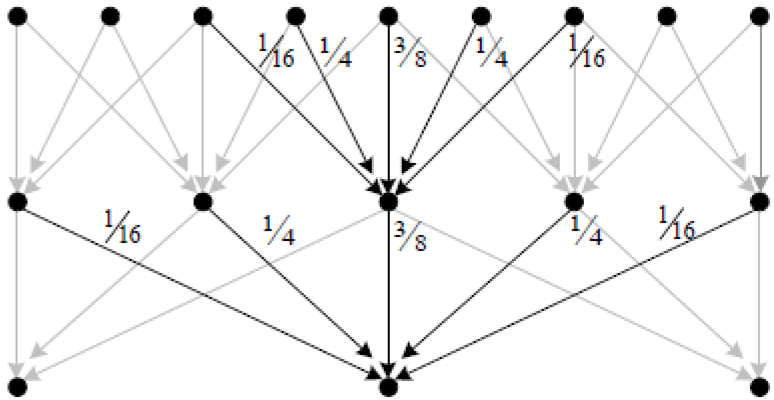
Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

S. Seitz

Computing the Laplacian Pyramid



Reduce and Expand

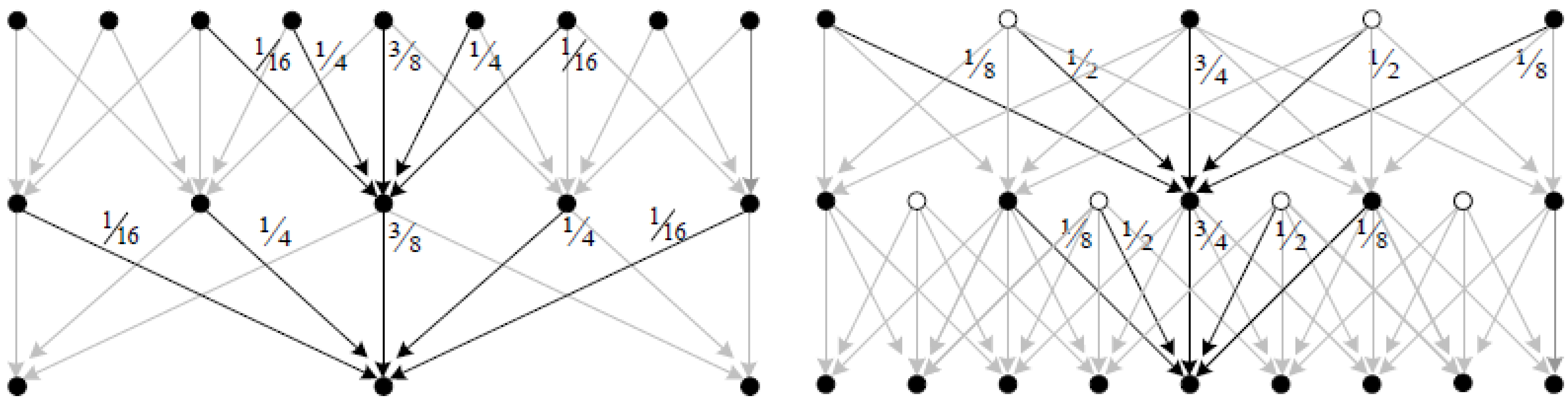


(a)

Reduce

Apply “5-tap” ***separable*** filter to make reduced image.

Reduce and Expand



(a) (b)

Reduce

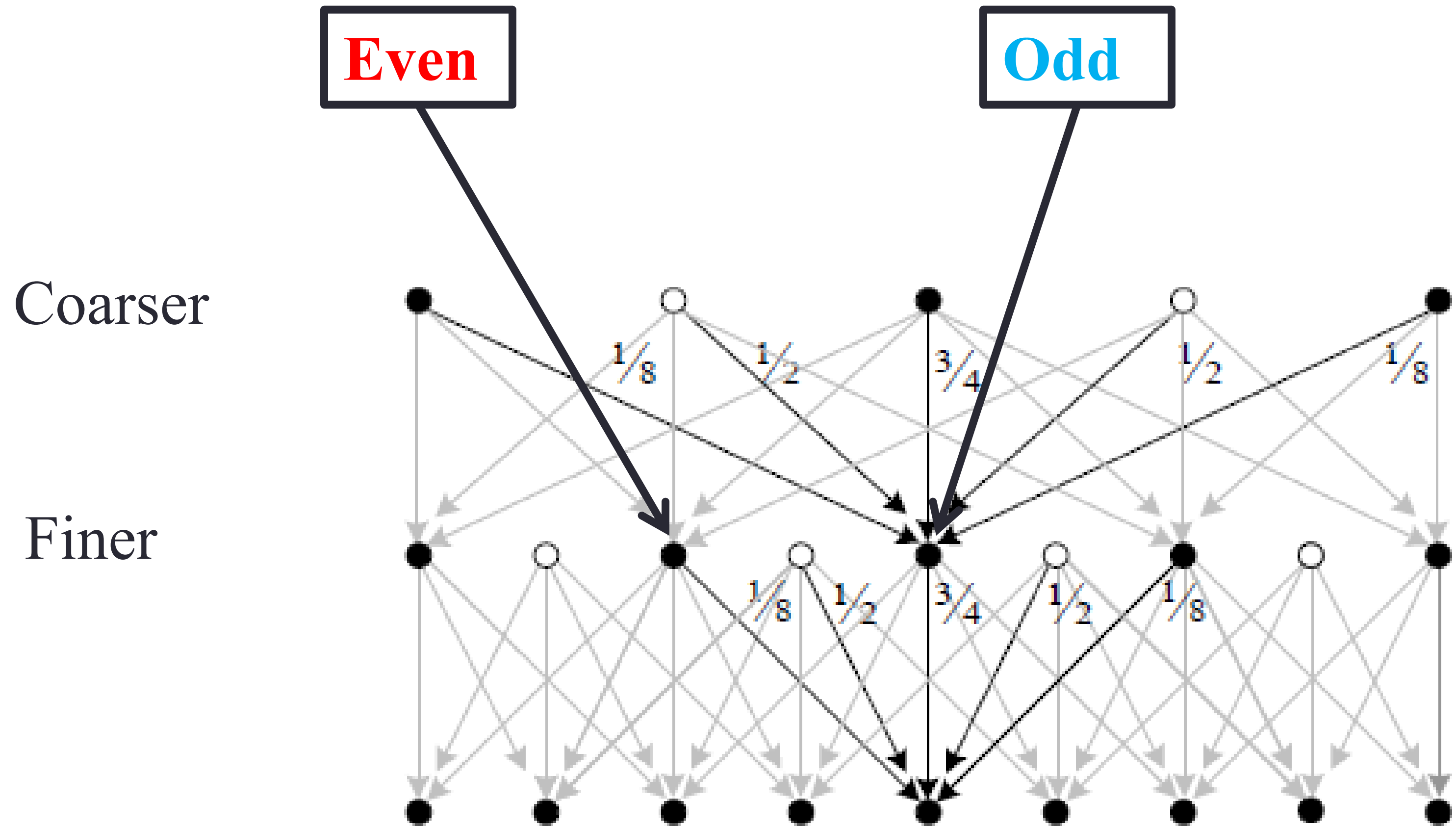
Apply "5-tap" separable filter to make reduced image.

Expand

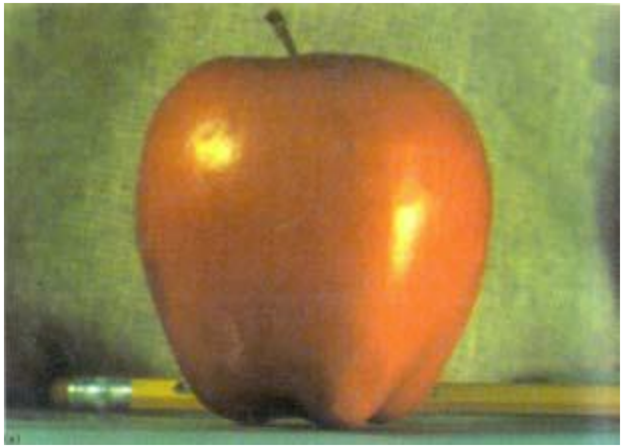
Apply different "3-tap" separable filters for even and odd pixels to make expanded image...

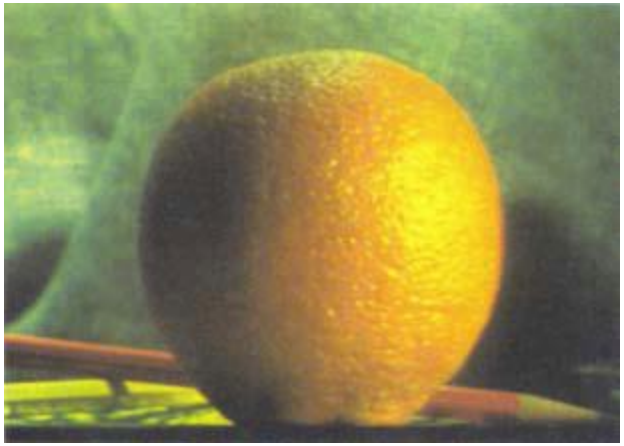
Just Expand

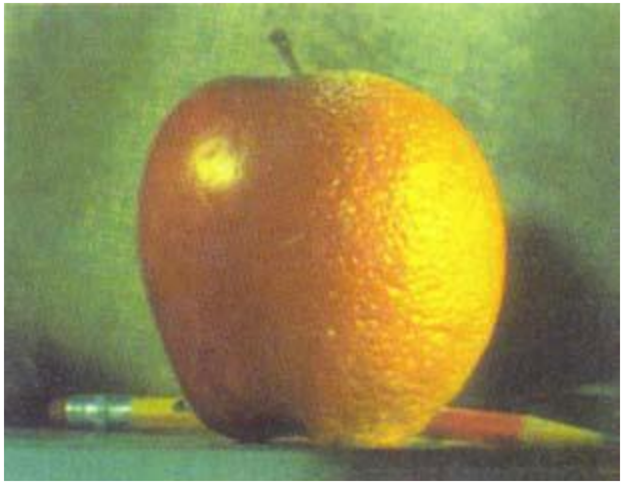
Apply different "3-tap" separable filters for even and odd pixels to make expanded image.



What can you do with band limited imaged?







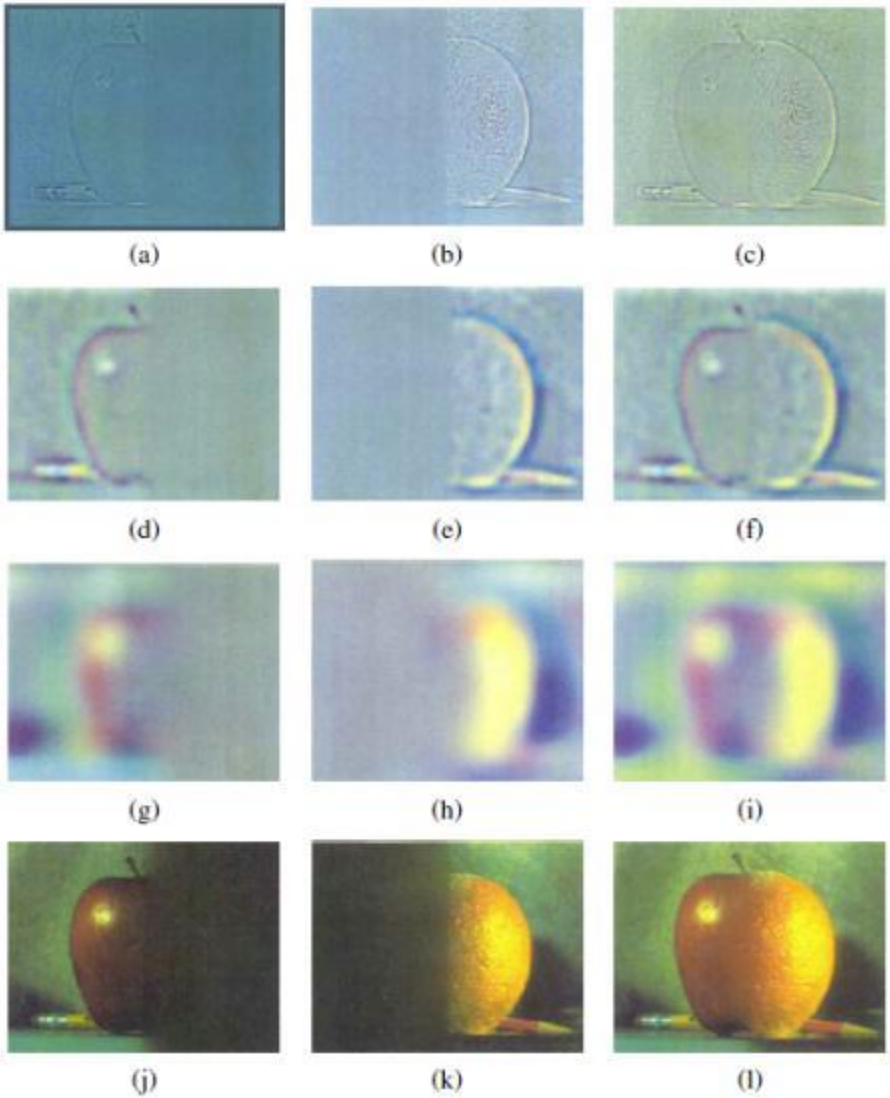
Apples and Oranges in bandpass

L_0

L_2

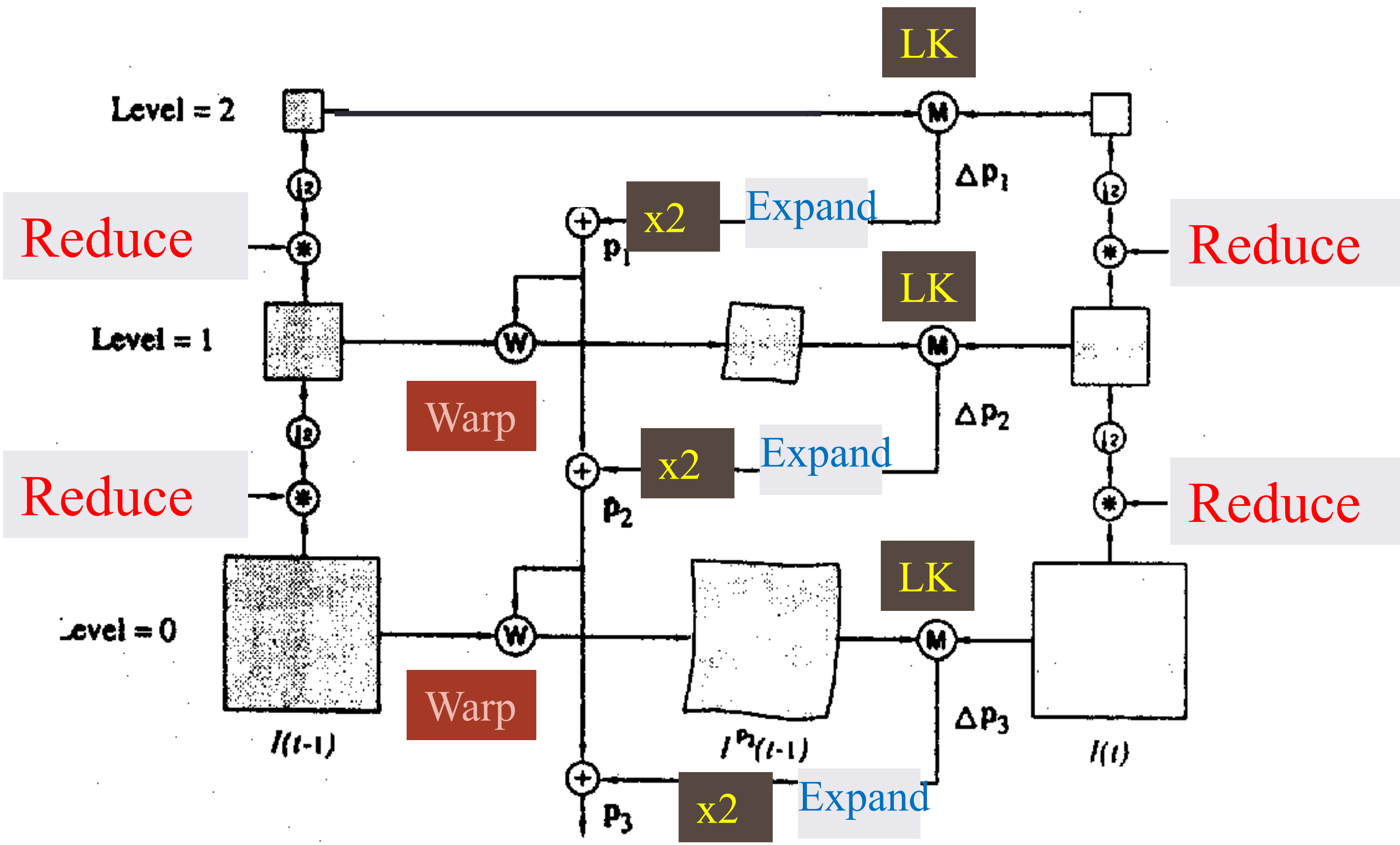
L_4

Reconstructed



Applying pyramids to LK

Coarse-to-fine global motion estimation



Final <u(x,y), v(x,y)>

Multi-resolution Lucas Kanade Algorithm

Compute Iterative LK at highest level

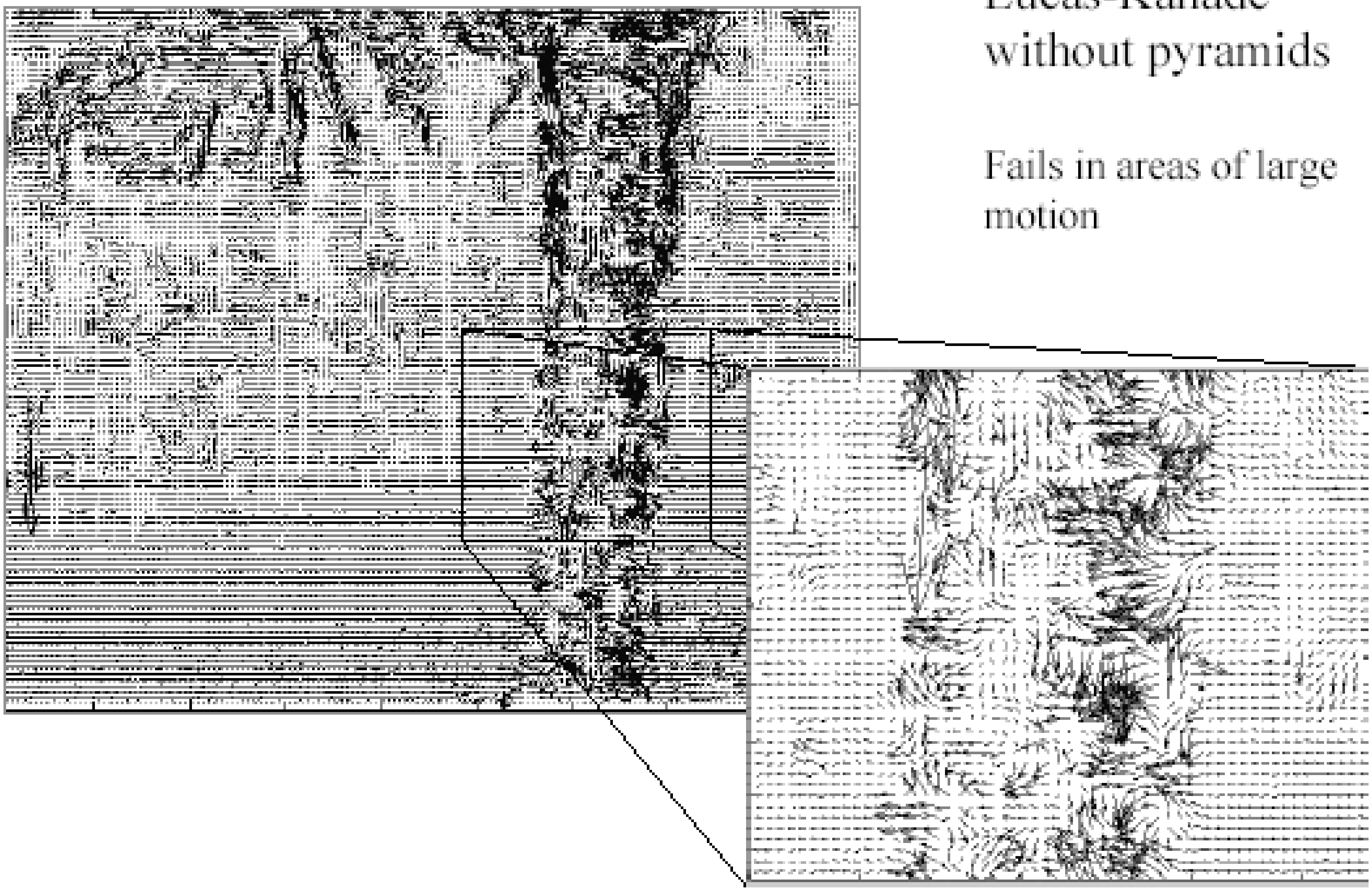
Initialize $u_{K+1}, v_{K+1} = 0$ at size of level K+1

For Each Level i from K to 0

- Upsample u_{i+1}, v_{i+1} to create u_i^p, v_i^p flow fields of now twice resolution as level $i+1$.
- Multiply u_i^p, v_i^p by 2 to get predicted flow
- Warp I_2 according to predicted flow
- Compute I_t –temporal derivative
- Apply LK to get u_i^δ, v_i^δ (the correction in flow)
- Add corrections to obtain the flow $u(i)$, $v(i)$ at i^{th} level, i.e.,

$$u_i = u_i^p + u_i^\delta ; v_i = v_i^p + v_i^\delta$$

Optical Flow Results

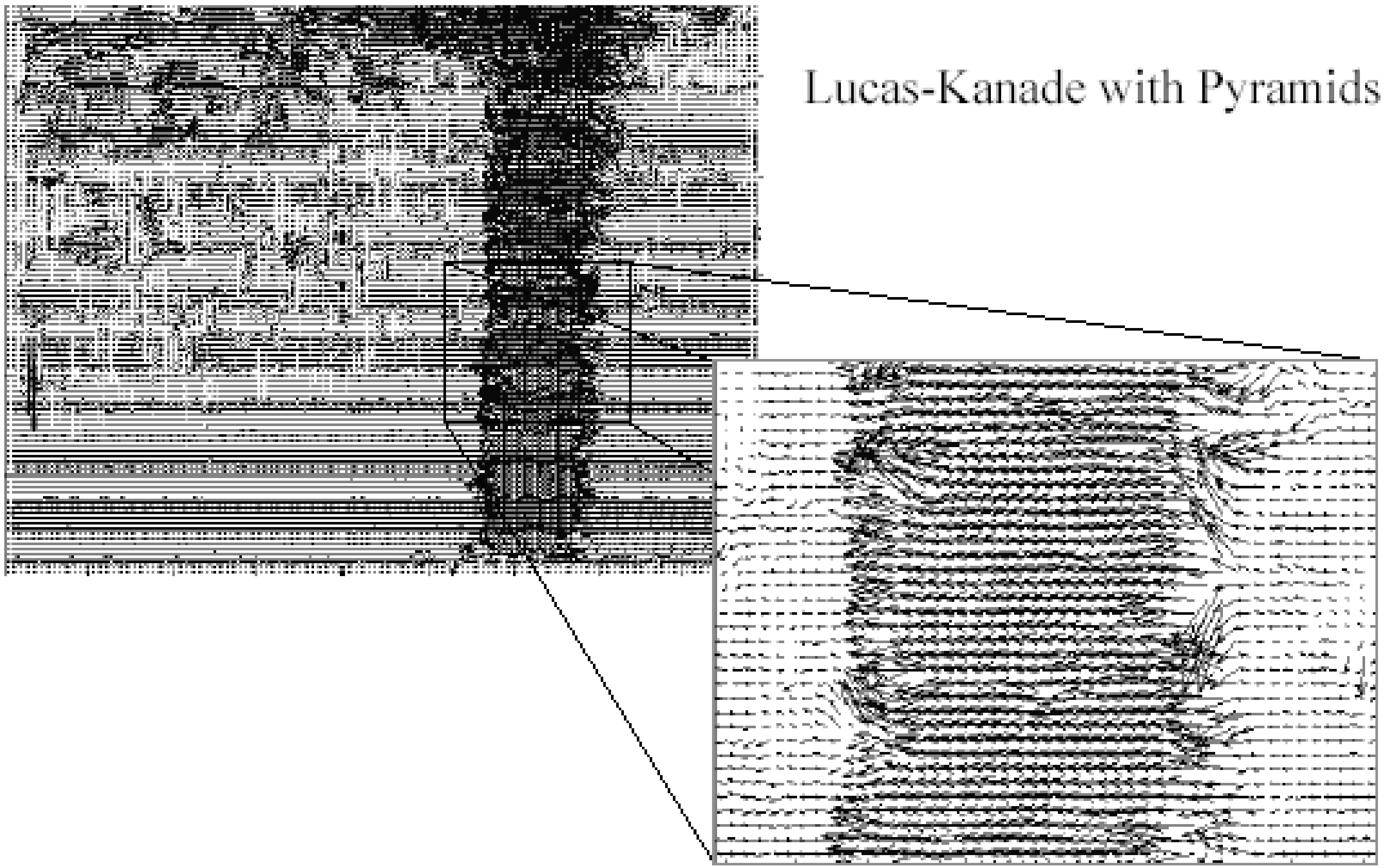


Lucas-Kanade without pyramids

Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Moving to models

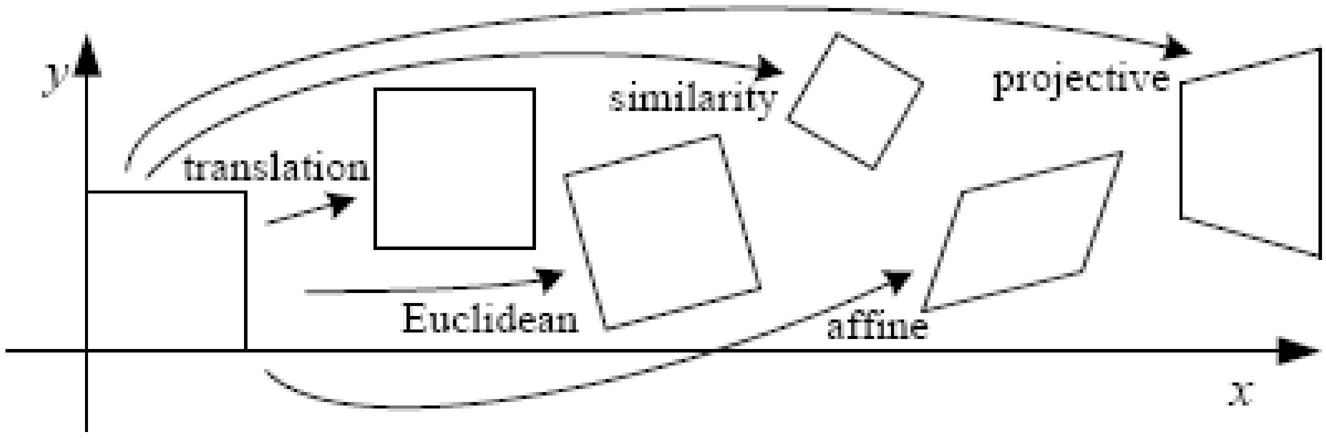
- Previous method(s) give dense flow with little or no constraint between locations (smoothness is either explicit or implicit).

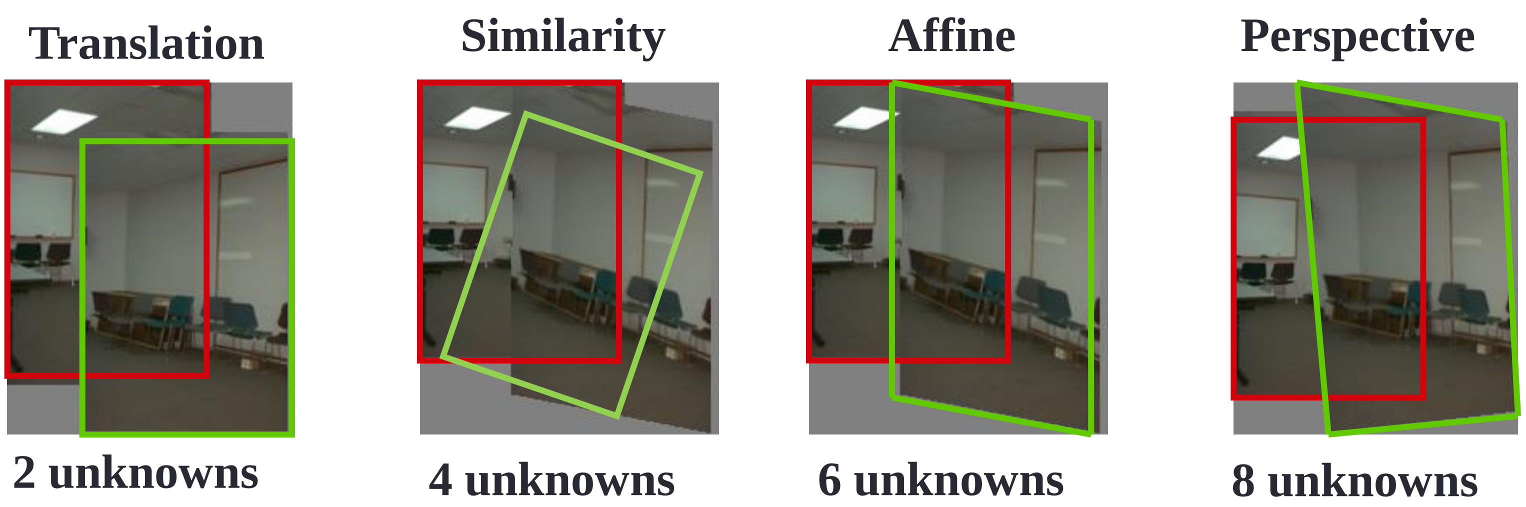
- Suppose you "know" that motion is constrained, e.g.
  - Small rotation about horizontal or vertical axis (or both) that is very close to a translation.
  - Distant independent moving objects

- In this case you might "model" the flow…

  Ready for another old slide?

Motion models





Focus of Expansion (FOE) - Example

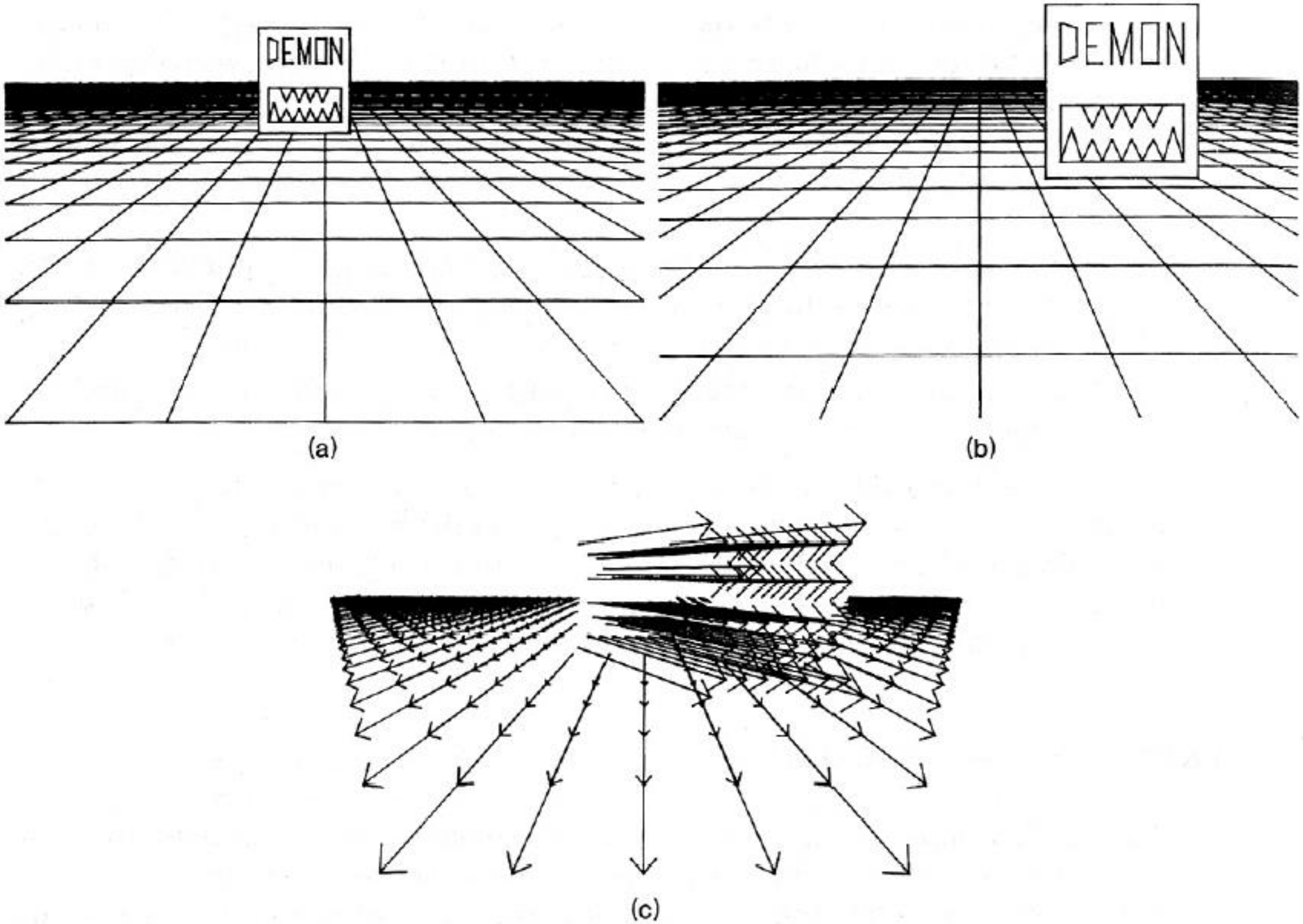


Fig. 7.3 FOE for rectilinear observer motion. (a) An image. (b) Later image. (c) Flow shows different FOEs for static floor and moving object.

Full motion model

- From physics or elsewhere:

$$V = \Omega \times R + T$$

$$[a_x] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \approx \begin{bmatrix} 0 & -\omega_Z & \omega_Y \\ \omega_Z & 0 & -\omega_X \\ -\omega_Y & \omega_X & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} V_{T_x} \\ V_{T_Y} \\ V_{T_Z} \end{bmatrix}$$

$\begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} =$ Velocity Vector

$\begin{bmatrix} V_{T_X} \\ V_{T_Y} \\ V_{T_Z} \end{bmatrix} =$ Translational Component of Velocity

$\begin{bmatrix} \omega_X \\ \omega_Y \\ \omega_Z \end{bmatrix} =$ Angular Velocity

General motion

Take derivatives:

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

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

$$u = v_x = f\frac{ZV_X - XV_Z}{Z^2} = f\frac{V_X}{Z} - \left(f\frac{X}{Z}\right)\frac{V_Z}{Z} = f\frac{V_X}{Z} - x\frac{V_Z}{Z}$$

$$v = v_y = f\frac{ZV_Y - YV_Z}{Z^2} = f\frac{V_Y}{Z} - \left(f\frac{Y}{Z}\right)\frac{V_Z}{Z} = f\frac{V_Y}{Z} - y\frac{V_Z}{Z}$$

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \frac{1}{Z(x,y)}\mathbf{A}(x,y)\mathbf{T} + \mathbf{B}(x,y)\mathbf{\Omega}$$

Why is Z only here?

$$\mathbf{A}(x,y) = \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \quad \mathbf{B}(x,y) = \begin{bmatrix} (xy)/f & -(f+x^2)/f & y \\ (f+y^2)/f & -(xy)/f & -x \end{bmatrix}$$

Where **T** is translation vector, Ω is rotation

If a plane and perspective…

$$aX + bY + cZ + d = 0$$

$$u(x, y) = a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy$$

$$v(x, y) = a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2$$

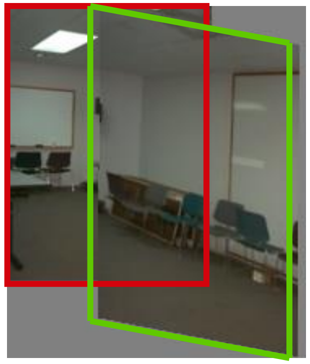
If a plane and orthographic…

$$u(x, y) = a_1 + a_2 x + a_3 y$$

$$v(x, y) = a_4 + a_5 x + a_6 y$$

Affine!

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 brightness constancy 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 1 linear constraint in 6 unknowns
- Least squares minimization:

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

Affine motion

- Can sum gradients over window or entire image:

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

- Minimize squared error (robustly)

$$\begin{bmatrix} I_x & I_x x_1 & I_x y_1 & I_y & I_y x_1 & I_y y_1 \\ I_x & I_x x_2 & I_x y_2 & I_y & I_y x_2 & I_y y_2 \\ & & \vdots & & & \\ & & \vdots & & & \\ I_x & I_x x_n & I_x y_n & I_y & I_y x_n & I_y y_n \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = - \begin{bmatrix} I_t^1 \\ I_t^2 \\ \vdots \\ \vdots \\ I_t^n \end{bmatrix}$$

- This is an example of parametric flow – can substitute any linear model easily. Others with some work.

Hierarchical model-based flow

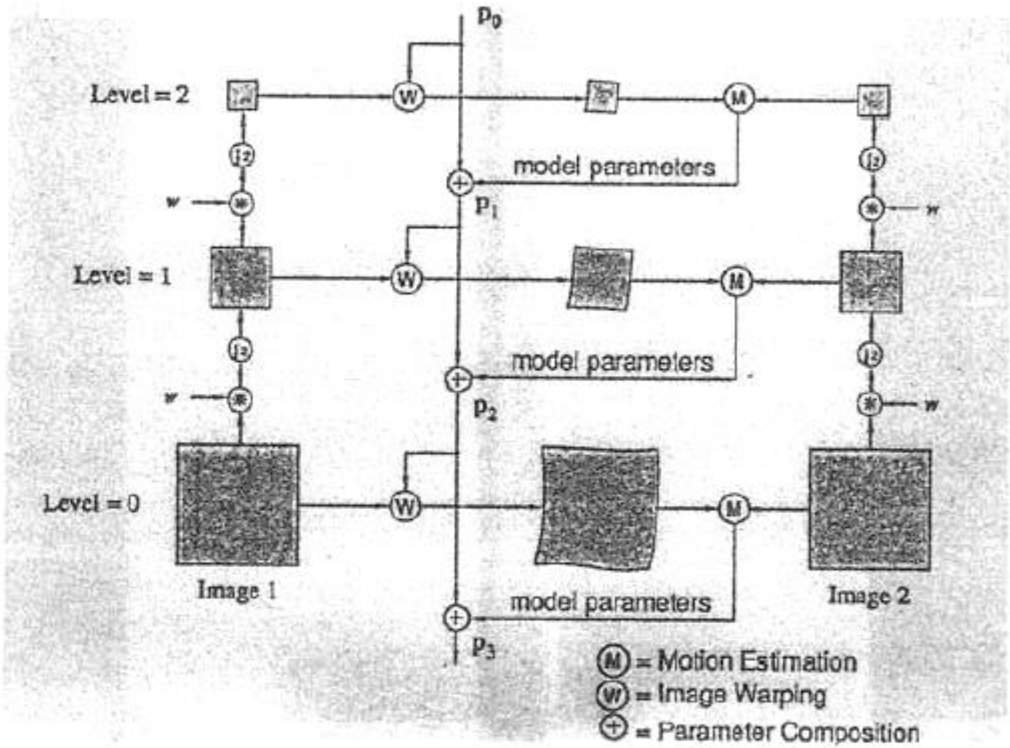


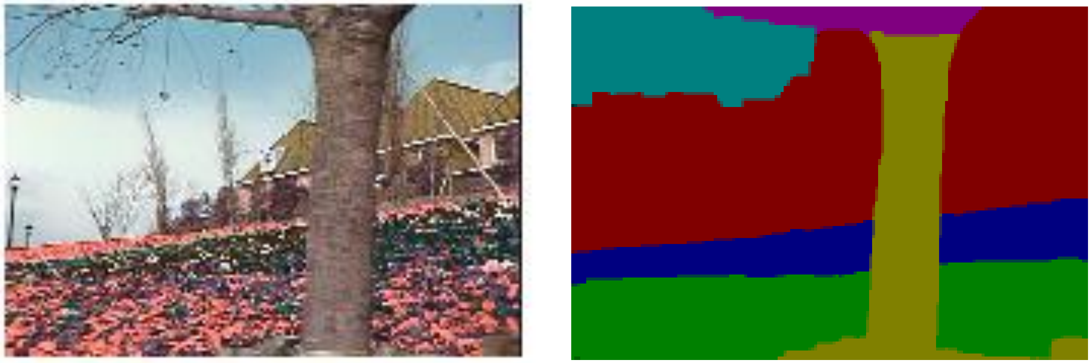
Fig. 1. Diagram of the hierarchical motion estimation framework.

James R. Bergen, P. Anandan, Keith J. Hanna, Rajesh Hingorani: “Hierarchical Model-Based Motion Estimation," ECCV 1992: 237-252

Now, if different motion regions…

Layered motion

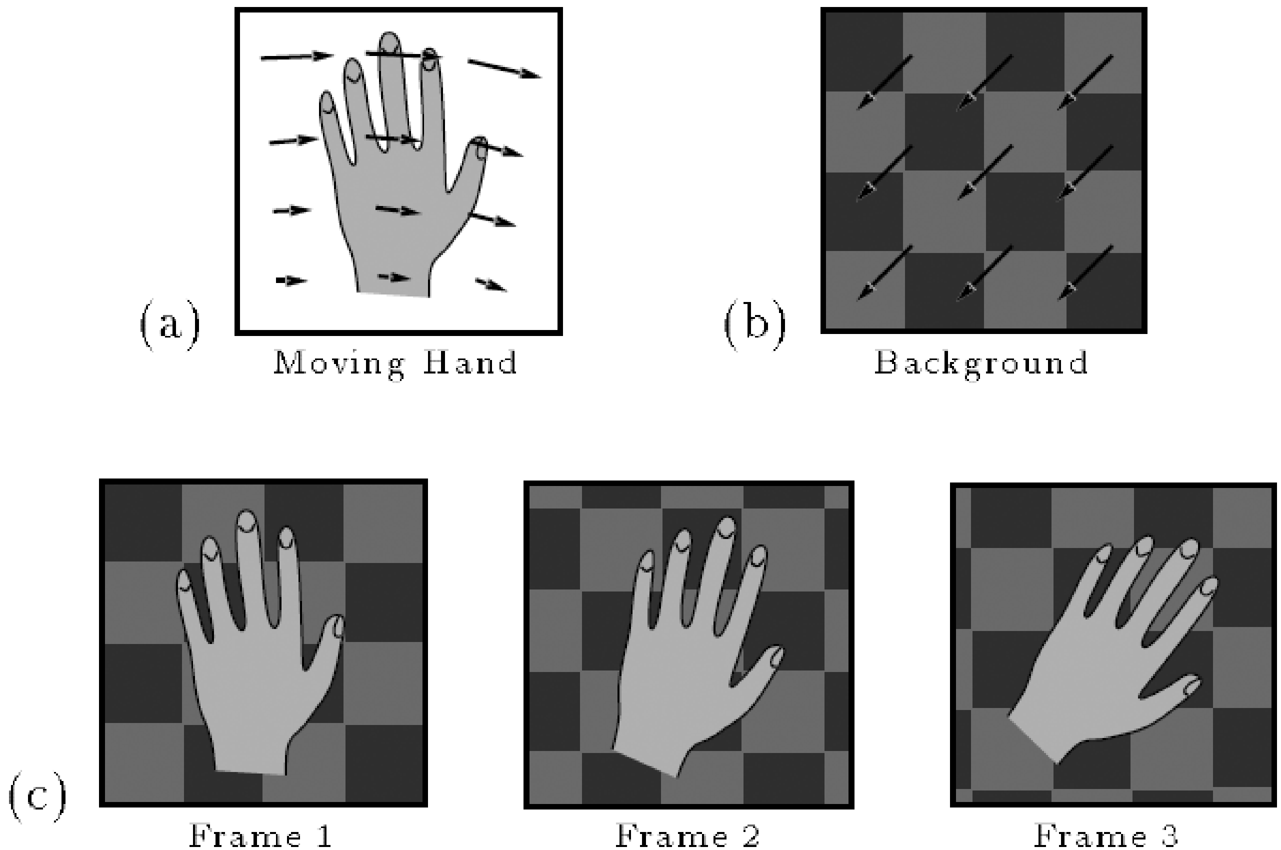
- Basic idea: break image sequence into "layers" each of which has a coherent motion



J. Wang and E. Adelson. Layered Representation for Motion Analysis. *CVPR 1993*.

What are layers?

- Each layer is defined by an alpha mask and an affine motion model



J. Wang and E. Adelson. [Layered Representation for Motion Analysis.](#) *CVPR 1993*.

Motion segmentation with an affine model

$$\boxed{u(x,y)} = a_1 + a_2 x + a_3 y$$

$$\boxed{v(x,y)} = a_4 + a_5 x + a_6 y$$

Local flow estimates

J. Wang and E. Adelson. Layered Representation for Motion Analysis. *CVPR 1993*.

Motion segmentation with an affine model

$$\boxed{u(x,y) = a_1 + a_2 x + a_3 y}$$

$$v(x,y) = a_4 + a_5 x + a_6 y$$

Equation of a plane (parameters a_1, a_2, a_3 can be found by least squares)

J. Wang and E. Adelson. [Layered Representation for Motion Analysis.](#) *CVPR 1993*.

Motion segmentation with an affine model

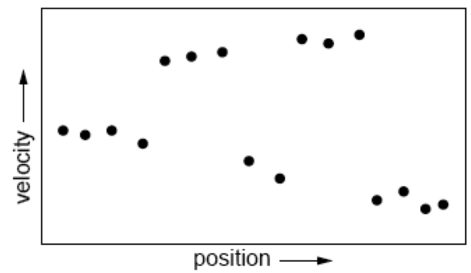
$$\boxed{u(x,y) = a_1 + a_2 x + a_3 y}$$

$$v(x,y) = a_4 + a_5 x + a_6 y$$

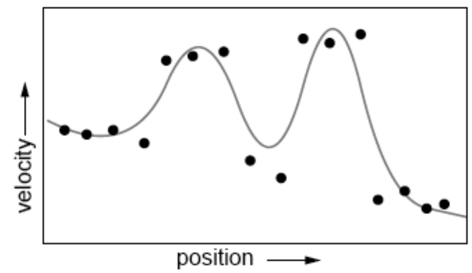
Equation of a plane (parameters a_1, a_2, a_3 can be found by least squares)

1D example

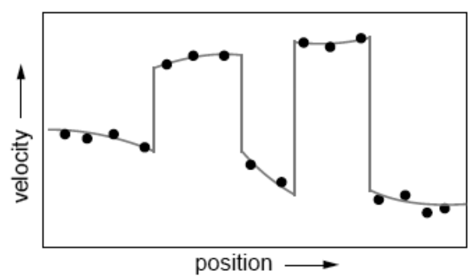
$u(x,y)$



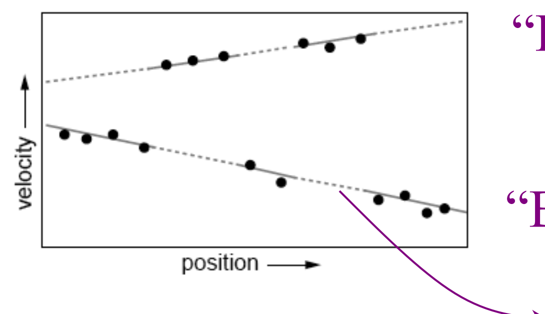
True flow



Local flow estimate



Segmented estimate



Line fitting

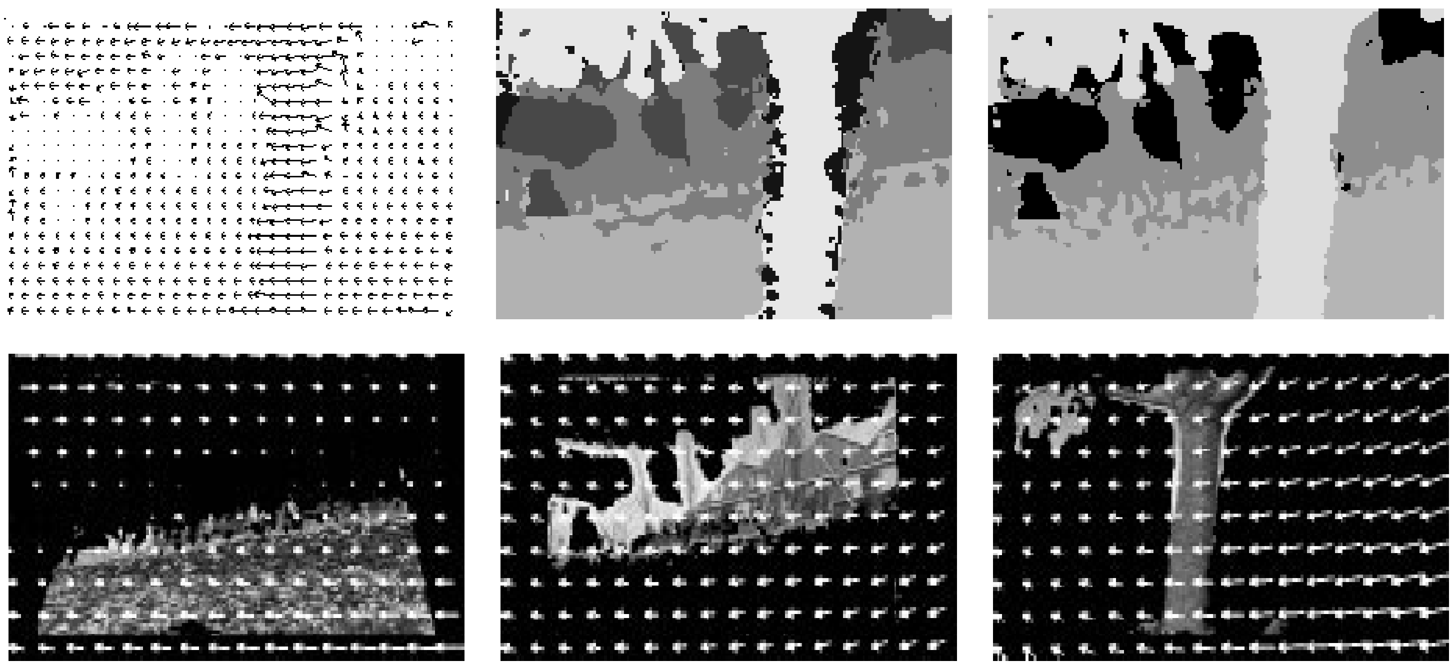
J. Wang and E. Adelson. [Layered Representation for Motion Analysis.](#) *CVPR 1993*.

How do we estimate the layers?

- Compute local flow in a coarse-to-fine fashion
- Obtain a set of initial affine motion hypotheses
  - Divide the image into blocks and estimate affine motion parameters in each block by least squares
    - Eliminate hypotheses with high residual error
  - Perform k-means clustering on affine motion parameters
    - Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene
- Iterate until convergence:
  - Assign each pixel to best hypothesis
    - Pixels with high residual error remain unassigned
  - Perform region filtering to enforce spatial constraints
  - Re-estimate affine motions in each region

J. Wang and E. Adelson. Layered Representation for Motion Analysis. *CVPR 1993*.

Example result



J. Wang and E. Adelson. [Layered Representation for Motion Analysis.](#) *CVPR 1993*.

Recovering image motion

- Feature-based methods (e.g. SIFT, Ransac, regression)
  - Extract visual features (corners, textured areas) and track them - sometimes over multiple frames
  - Sparse motion fields, but possibly robust tracking
    - Good for global motion
  - Suitable especially when image motion is large (10-s of pixels)
  - **PS4!**

- Direct-methods (e.g. optical flow)
  - Directly recover image motion from spatio-temporal image brightness variations
  - Dense, local motion fields, but more sensitive to appearance variations
  - Suitable for video and when image motion is small (< 10 pixels)
  - **PS5!!!**

End CS4495