Motion Estimation

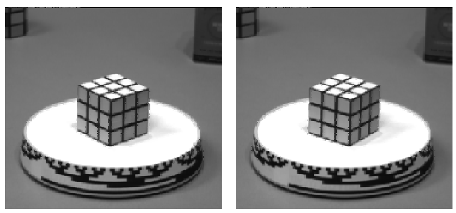
Why estimate motion?

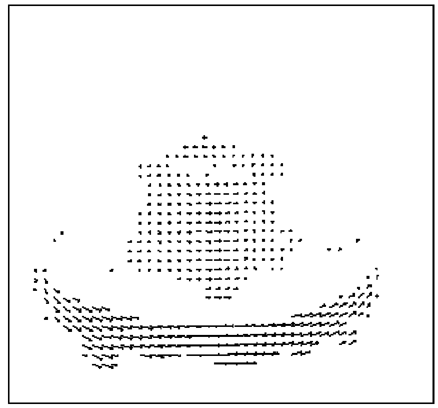
Lots of uses

- Motion Detection
- Track object behavior
- Correct for camera jitter (stabilization)
- Align images (mosaics)
- 3D shape reconstruction
- Video Compression

Optical flow

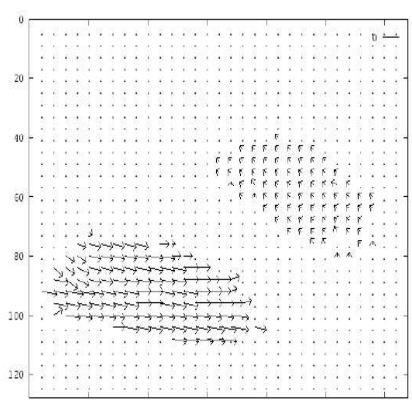
Measurement of motion at every pixel



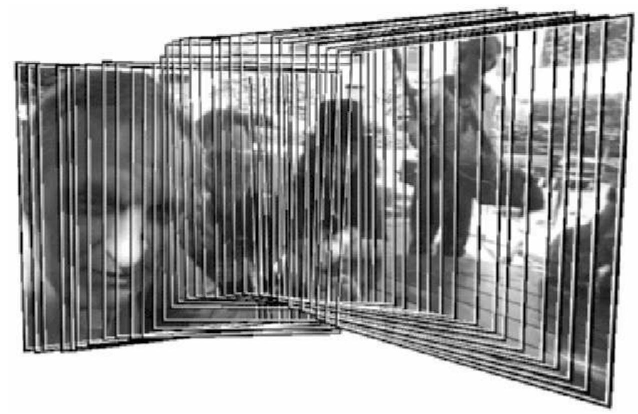


Optical flow

An image from Hamburg Taxi Sequence



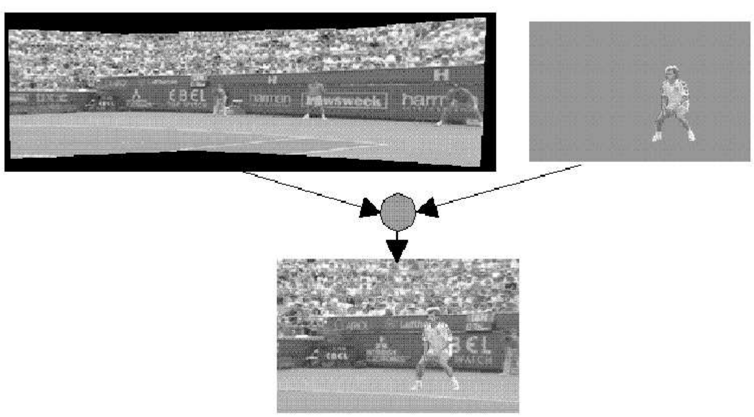
Video Mosaics



Video Mosaics

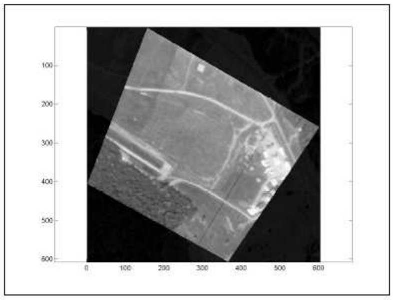
Video Mosaics

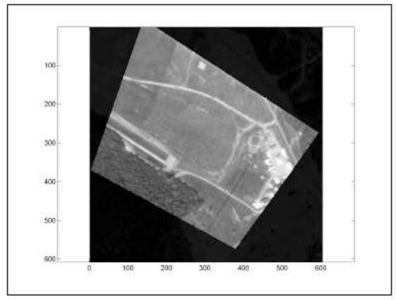
Video Compression



Geo Registration

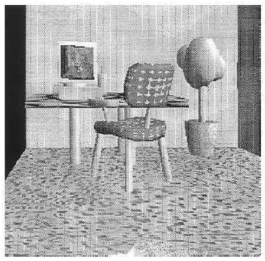
Results superimposed with the reference image

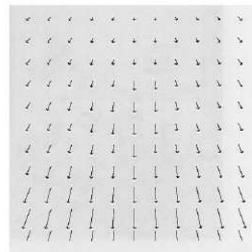


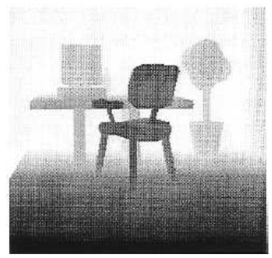


Video Segmentation

Structure From Motion

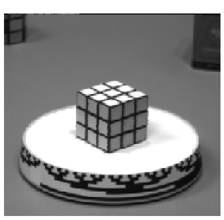


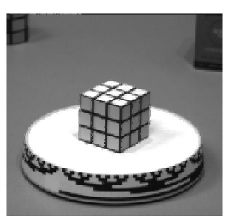


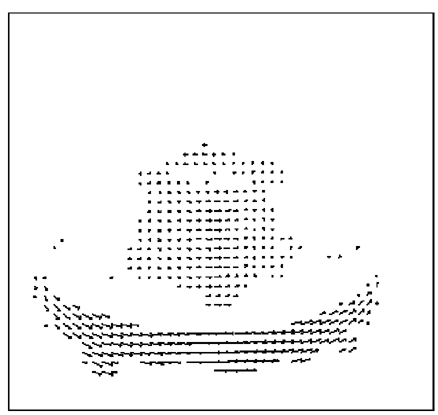


Optical flow

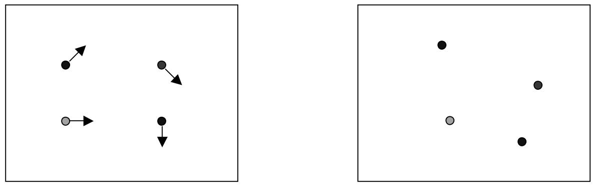
Measurement of motion at every pixel







Problem definition: optical flow



$H(x, y)$ $\qquad$ $I(x, y)$

How to estimate pixel motion from image H to image I?

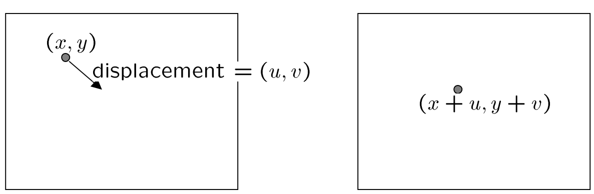
- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions

- **color constancy:** a point in H looks the same in I
  - For grayscale images, this is **brightness constancy**
- **small motion**: points do not move very far

This is called the **optical flow** problem

Optical flow constraints (grayscale images)



$H(x, y)$ $\qquad$ $I(x, y)$

Let's look at these constraints more closely

- brightness constancy: Q: what's the equation?

- small motion: (u and v are less than 1 pixel)
  - suppose we take the Taylor series expansion of I:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$
$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

Optical flow equation

Combining these two equations

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

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

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

$$0 = I_t + \nabla I \cdot [\tfrac{\partial x}{\partial t} \;\; \tfrac{\partial y}{\partial t}]$$

Optical flow equation

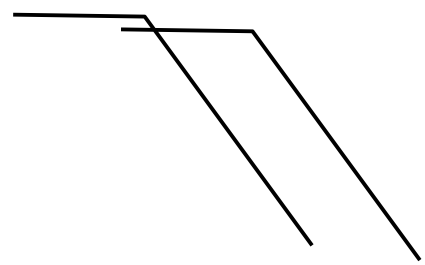
$$0 = I_t + \nabla I \cdot [u \;\; v]$$

Q: how many unknowns and equations per pixel?

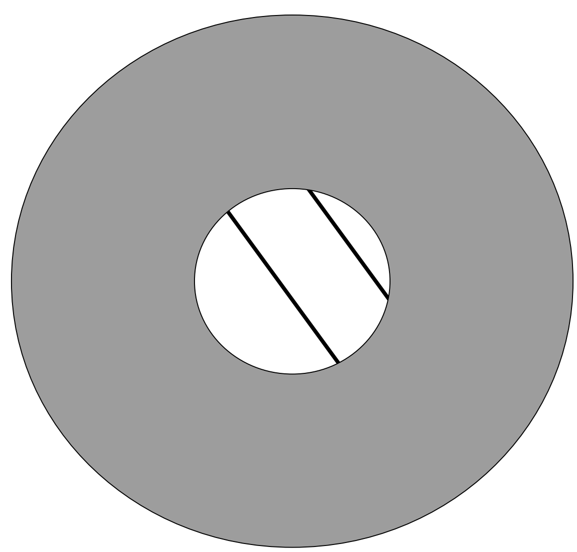
Intuitively, what does this constraint mean?

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

Aperture problem



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!

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

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

RGB version

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*3 equations per pixel!

$$0 = I_t(\mathbf{p_i})[0,1,2] + \nabla I(\mathbf{p_i})[0,1,2] \cdot [u \ v]$$

$$\underset{75\times 2}{\begin{bmatrix} I_x(\mathbf{p_1})[0] & I_y(\mathbf{p_1})[0] \\ I_x(\mathbf{p_1})[1] & I_y(\mathbf{p_1})[1] \\ I_x(\mathbf{p_1})[2] & I_y(\mathbf{p_1})[2] \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}})[0] & I_y(\mathbf{p_{25}})[0] \\ I_x(\mathbf{p_{25}})[1] & I_y(\mathbf{p_{25}})[1] \\ I_x(\mathbf{p_{25}})[2] & I_y(\mathbf{p_{25}})[2] \end{bmatrix}}_{A} \underset{2\times 1}{\begin{bmatrix} u \\ v \end{bmatrix}}_{d} = - \underset{75\times 1}{\begin{bmatrix} I_t(\mathbf{p_1})[0] \\ I_t(\mathbf{p_1})[1] \\ I_t(\mathbf{p_1})[2] \\ \vdots \\ I_t(\mathbf{p_{25}})[0] \\ I_t(\mathbf{p_{25}})[1] \\ I_t(\mathbf{p_{25}})[2] \end{bmatrix}}_{b}$$

Lukas-Kanade flow

Prob: we have more equations than unknowns

$$\underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

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

$$\underset{2\times2}{(A^T A)}\ \underset{2\times1}{d} = \underset{2\times1}{A^T b}$$

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

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

Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

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

When is This Solvable?

- **$A^T A$** should be invertible
- **$A^T A$** should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of **$A^T A$** should not be too small
- **$A^T A$** should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Eigenvectors of A^TA

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

Suppose (x,y) is on an edge. What is A^TA?

- gradients along edge all point the same direction
- gradients away from edge have small magnitude

$$\left(\sum \nabla I (\nabla I)^T\right) \approx k \nabla I \nabla I^T$$

$$\left(\sum \nabla I (\nabla I)^T\right) \nabla I = k \|\nabla I\| \nabla I$$

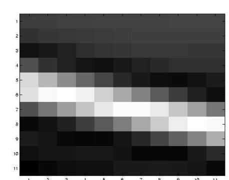
- ∇I is an eigenvector with eigenvalue $k\|\nabla I\|$
- What's the other eigenvector of A^TA?
  - let N be perpendicular to ∇I

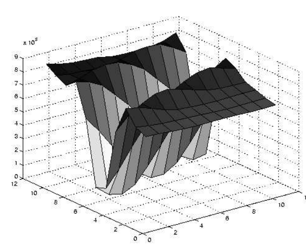
$$\left(\sum \nabla I (\nabla I)^T\right) N = 0$$

  - N is the second eigenvector with eigenvalue 0

The eigenvectors of A^TA relate to edge direction and magnitude

Edge



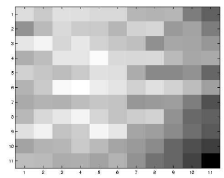


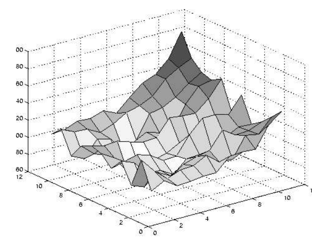
$$\sum \nabla I (\nabla I)^T$$

– large gradients, all the same

– large λ_1, small λ_2

Low texture region

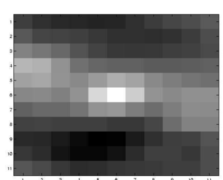


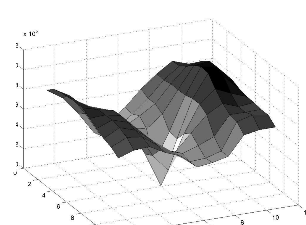


$\sum \nabla I (\nabla I)^T$

– gradients have small magnitude
– small λ_1, small λ_2

High textured region





$\sum \nabla I (\nabla I)^T$

– gradients are different, large magnitudes
– large λ_1, large λ_2

Observation

This is a two image problem BUT

- Can measure sensitivity by just looking at one of the images!
- This tells us which pixels are easy to track, which are hard
  - very useful later on when we do feature tracking...

Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

- Suppose A^TA is easily invertible
- Suppose there is not much noise in the image

When our assumptions are violated

- Brightness constancy is **not** satisfied
- The motion is **not** small
- A point does **not** move like its neighbors
  - window size is too large
  - what is the ideal window size?

Improving accuracy

Recall our small motion assumption

$$0 = I(x+u, y+v) - H(x,y)$$
$$\approx I(x,y) + I_x u + I_y v - H(x,y)$$

This is not exact

- To do better, we need to add higher order terms back in:

$$= I(x,y) + I_x u + I_y v + \text{higher order terms} - H(x,y)$$

This is a polynomial root finding problem

- Can solve using **Newton's method**
  - Also known as **Newton-Raphson** method

- Lukas-Kanade method does one iteration of Newton's method
  - Better results are obtained via more iterations

Iterative Refinement

Iterative Lukas-Kanade Algorithm

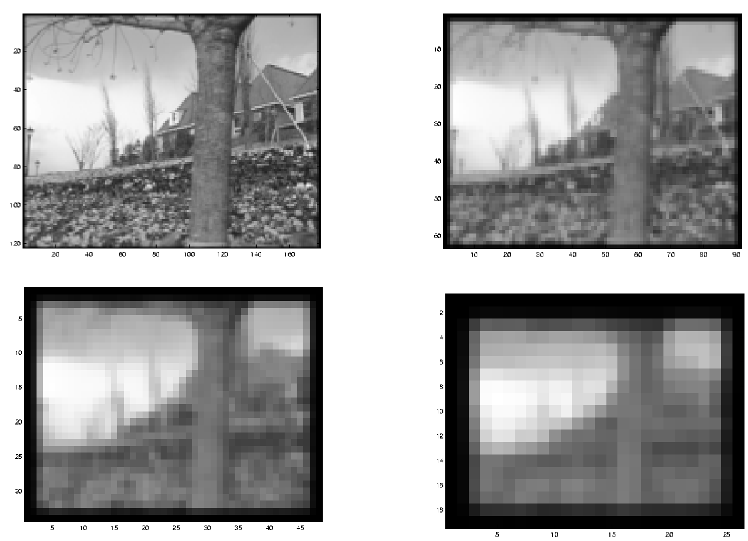
1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp H towards I using the estimated flow field
   - use image warping techniques
3. Repeat until convergence

Revisiting the small motion assumption

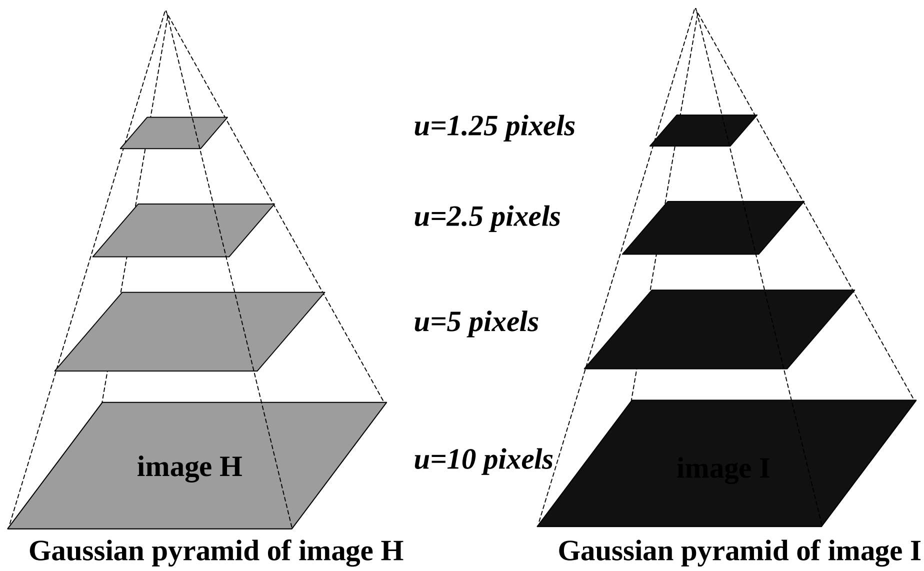
Is this motion small enough?

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

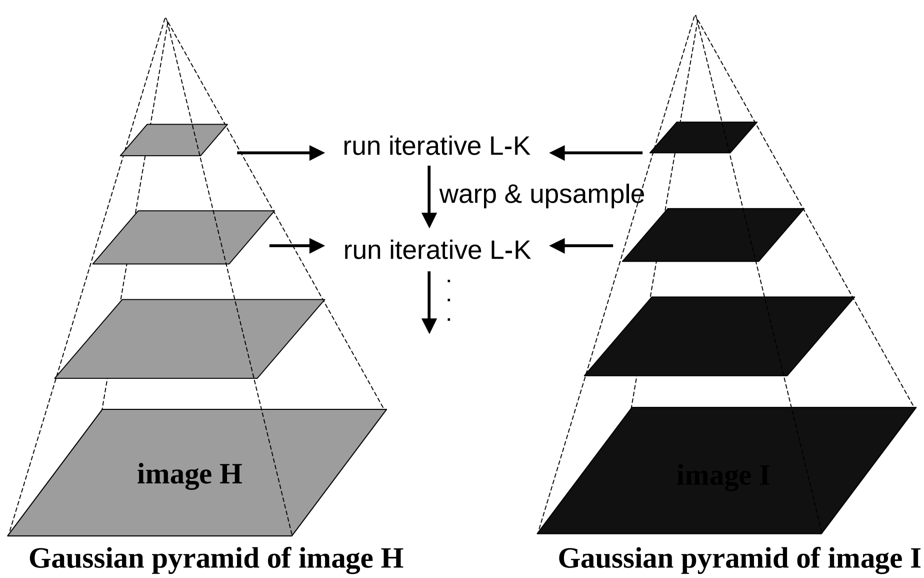
Reduce the resolution!



Coarse-to-fine optical flow estimation



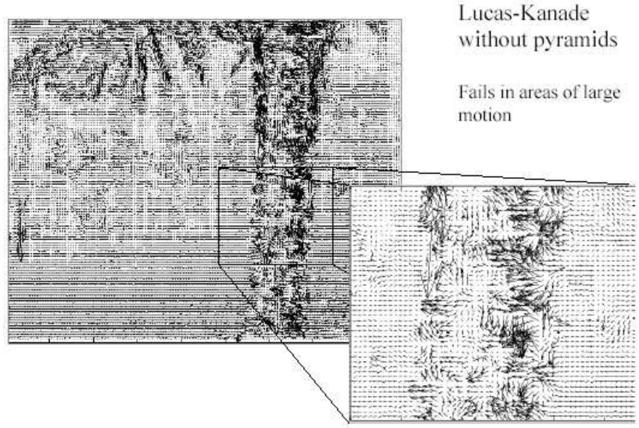
Coarse-to-fine optical flow estimation



Multi-resolution Lucas Kanade Algorithm

- Compute 'simple' LK at highest level
- At level i
  - Take flow u_{i-1}, v_{i-1} from level i-1
  - bilinear interpolate it to create u_i^*, v_i^* matrices of twice resolution for level i
  - multiply u_i^*, v_i^* by 2
  - compute f_t from a block displaced by $u_i^*(x,y)$, $v_i^*(x,y)$
  - Apply LK to get $u_i'(x, y)$, $v_i'(x, y)$ (the correction in flow)
  - Add corrections u_i' v_i', *i.e.* $u_i = u_i^* + u_i'$, $v_i = v_i^* + v_i'$.

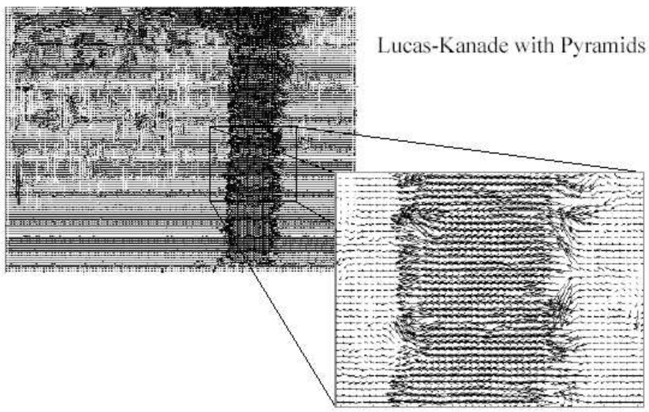
Optical Flow Results



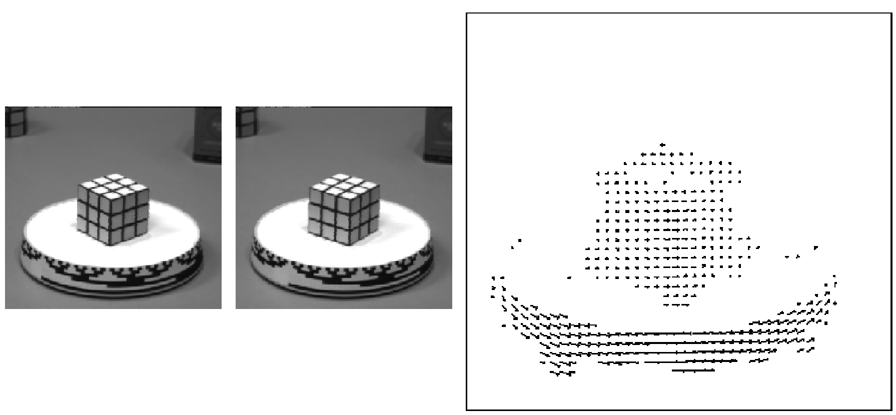
Lucas-Kanade without pyramids

Fails in areas of large motion

Optical Flow Results



Optical flow Results



Suggested Readings

- Chapter 8, Emanuele Trucco, Alessandro Verri, “Introductory Techniques for 3-D Computer Vision”