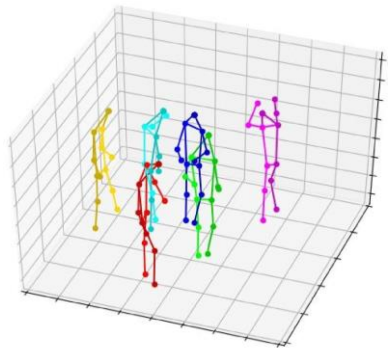
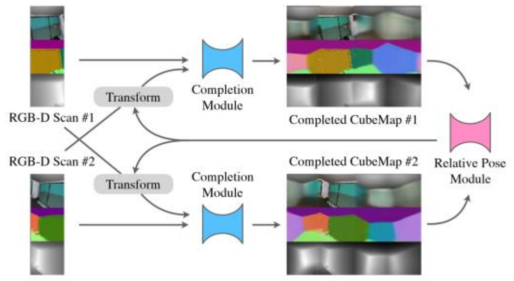
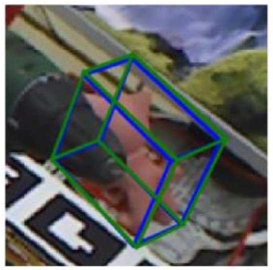
# CS376 Computer Vision
# Lecture 6: Optical Flow

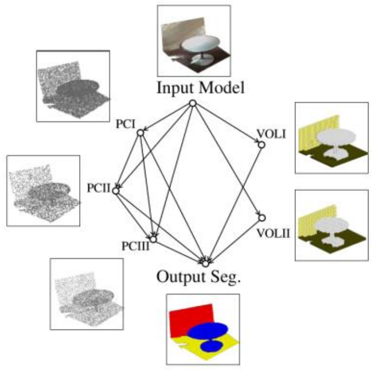




Qixing Huang

Feb. 11th 2019

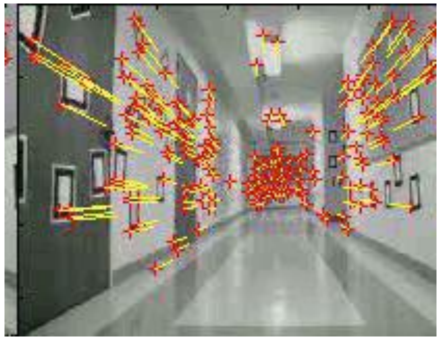




Slides Credit: Kristen Grauman and Sebastian Thrun, Michael Black, Marc Pollefeys

# Optical Flow

Image tracking



3D computation

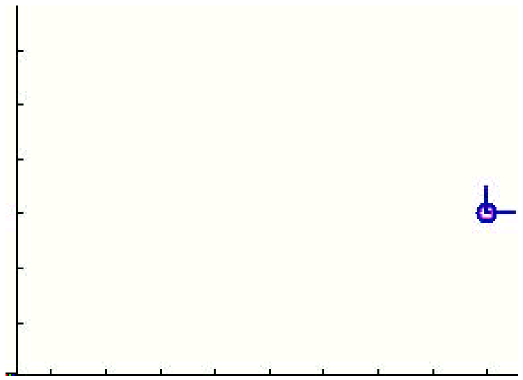
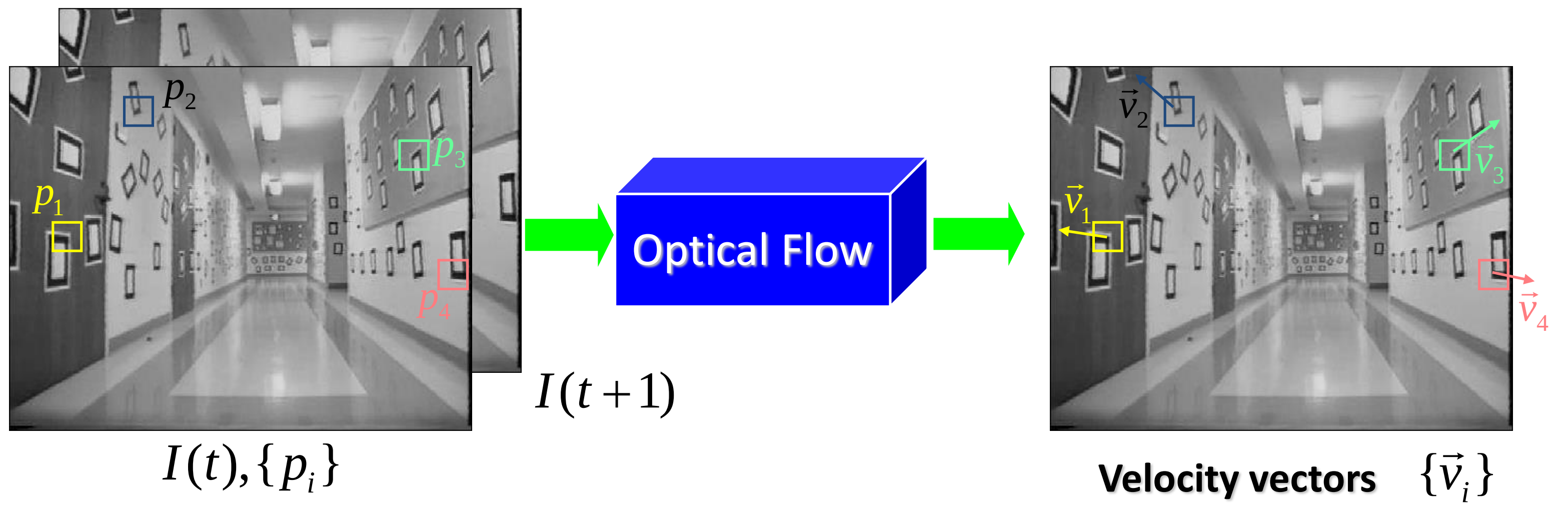


Image sequence
(single camera)

Tracked sequence

3D structure
+
3D trajectory

# What is Optical Flow?



**Optical flow** is *the 2D projection of the physical movement of points relative to the observer*

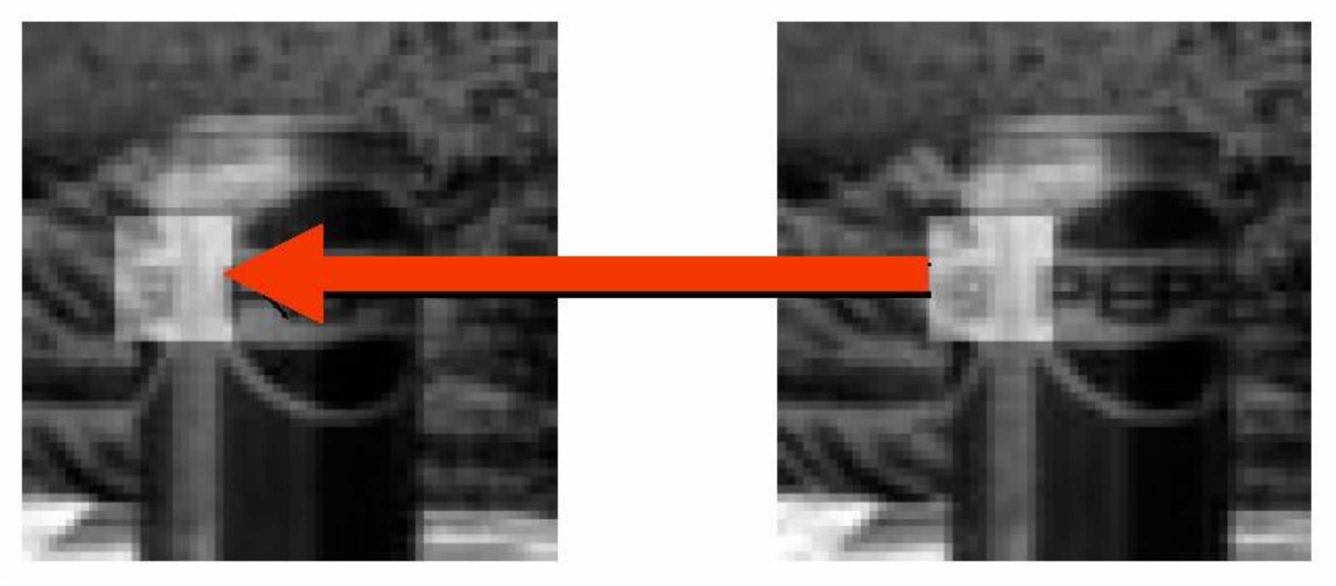
A common assumption is brightness constancy:

$$I(p_i, t) = I(p_i + \vec{v}_i, t+1)$$

# When does Brightness Assumption Break down?

- TV is based on illusory motion
  - the set is stationary yet things seem to move
- A uniform rotating sphere
  - nothing seems to move, yet it is rotating
- Changing directions or intensities of lighting can make things seem to move
  - for example, if the specular highlight on a rotating sphere moves
- Muscle movement can make some spots on a cheetah move opposite direction of motion

# Optical Flow Assumptions: Brightness Constancy



Assumption

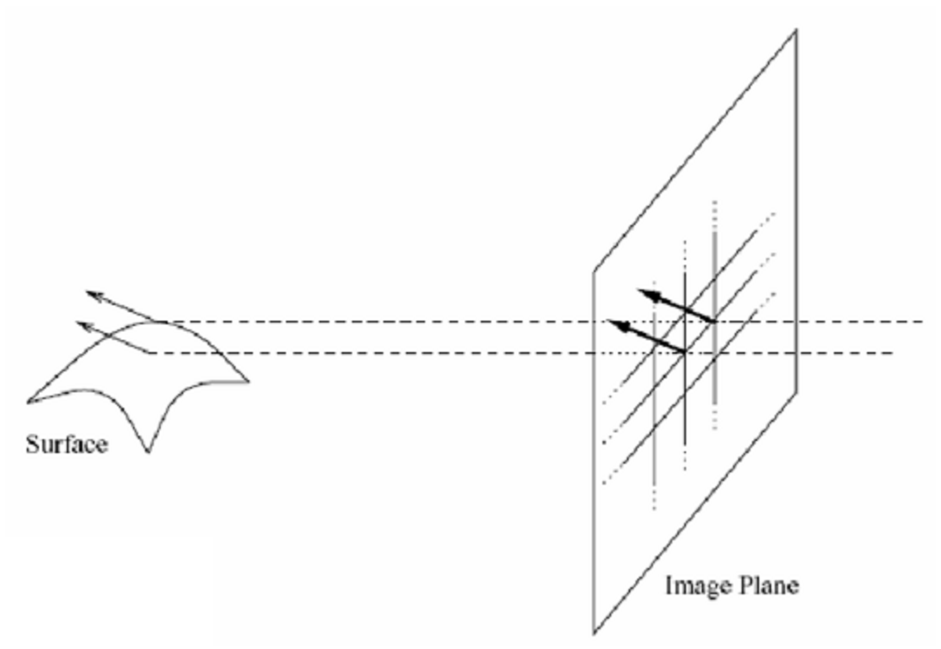
Image measurements (e.g. brightness) in a small region remain the same although their location may change.

$$I(x+u, y+v, t+1) = I(x, y, t)$$

(assumption)

* Slide from Michael Black, CS143 2003

## Optical Flow Assumptions:

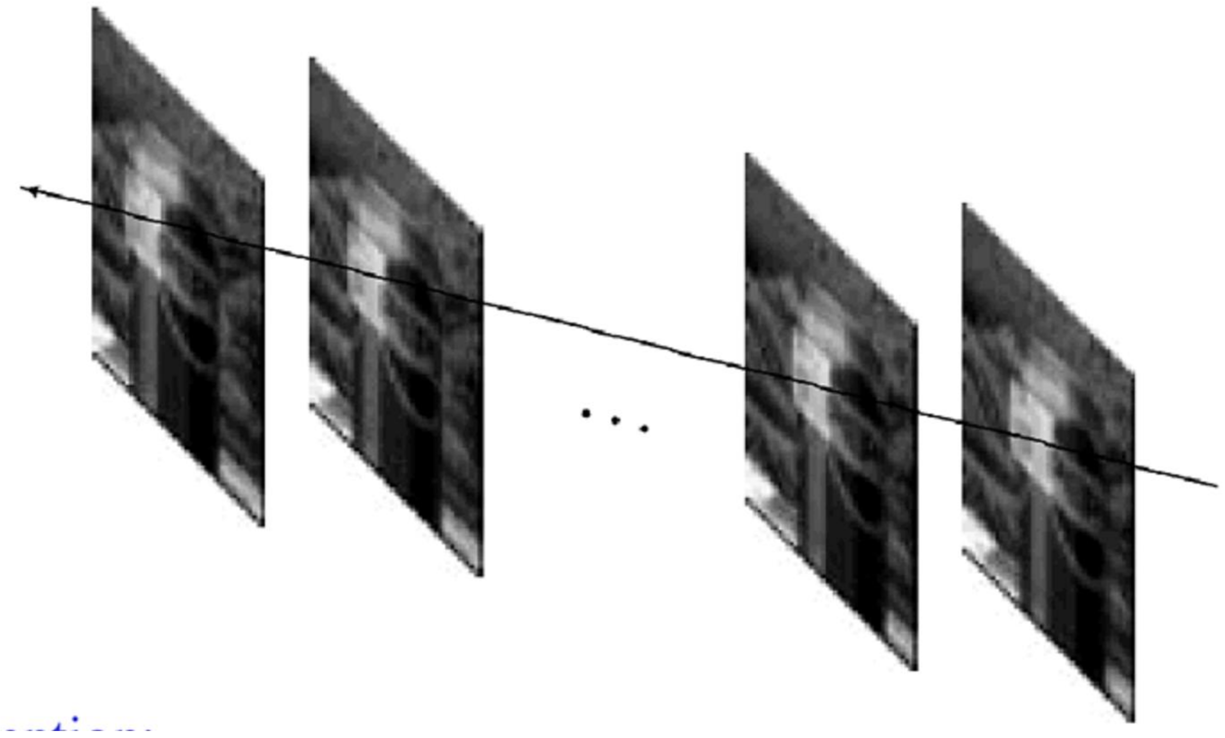


Neighboring pixels tend to have similar motions

When does this break down?

# Optical Flow Assumptions:

- The image motion of a surface path changes gradually over time



## 1D Optical Flow

# Optical Flow: 1D Case

Brightness Constancy Assumption:

$$f(t) \equiv \underbrace{I(x(t),t)} = I(x(t+dt),t+dt)$$

$$\frac{\partial f(x)}{\partial t} = 0$$ Because no change in brightness with time

$$\left.\frac{\partial I}{\partial x}\right|_t \left(\frac{\partial x}{\partial t}\right) + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

$I_x$ $v$ $I_t$

$$\Rightarrow v = \frac{I_t}{I_x}$$

## 2D Optical Flow

# From 1D to 2D tracking

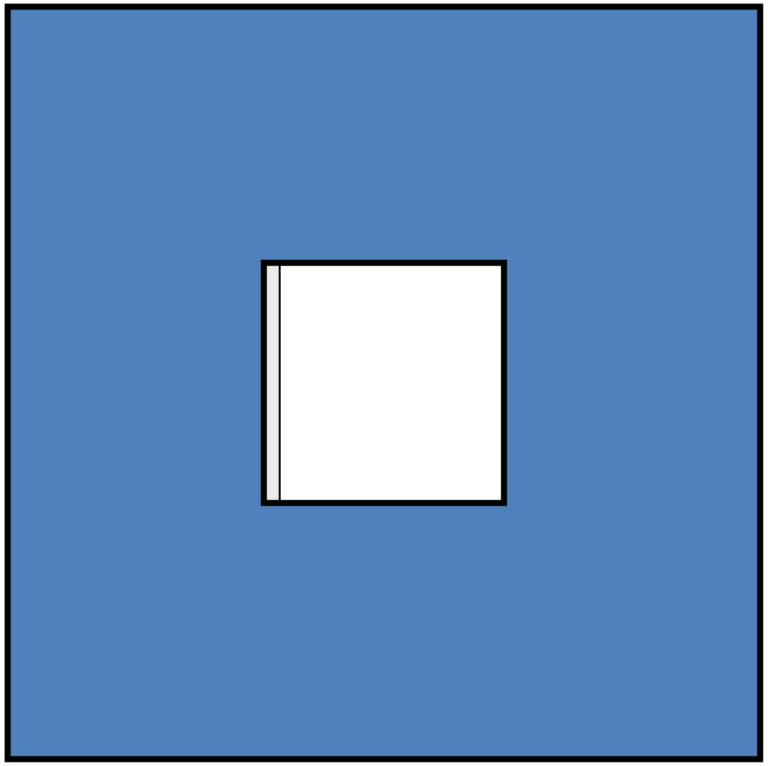
1D: $$\left.\frac{\partial I}{\partial x}\right|_{t}\left(\frac{\partial x}{\partial t}\right)+\left.\frac{\partial I}{\partial t}\right|_{x(t)}=0$$

2D: $$\left.\frac{\partial I}{\partial x}\right|_{t}\left(\frac{\partial x}{\partial t}\right)+\left.\frac{\partial I}{\partial y}\right|_{t}\left(\frac{\partial y}{\partial t}\right)+\left.\frac{\partial I}{\partial t}\right|_{x(t)}=0$$

$$\left.\frac{\partial I}{\partial x}\right|_{t}u+\left.\frac{\partial I}{\partial y}\right|_{t}v+\left.\frac{\partial I}{\partial t}\right|_{x(t)}=0$$

One equation but two velocity *(u,v)* unknowns…

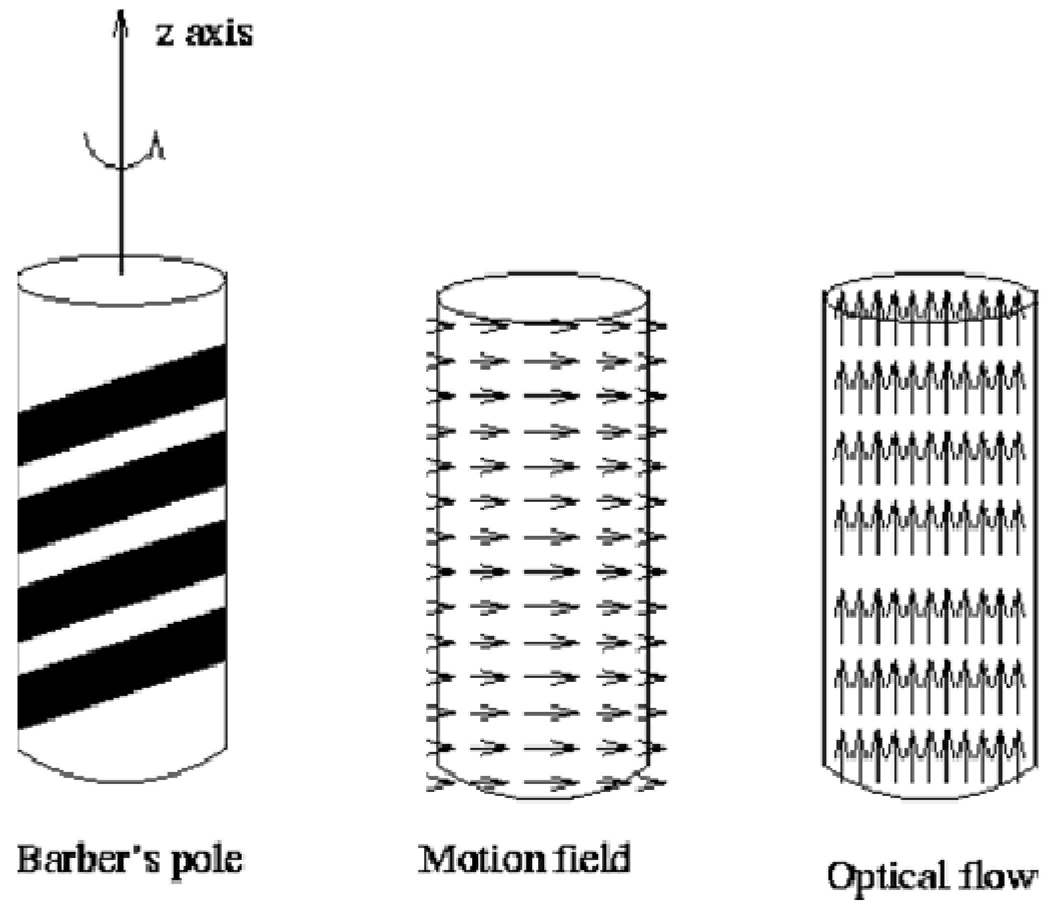
## How does this show up visually? Known as the “Aperture Problem”



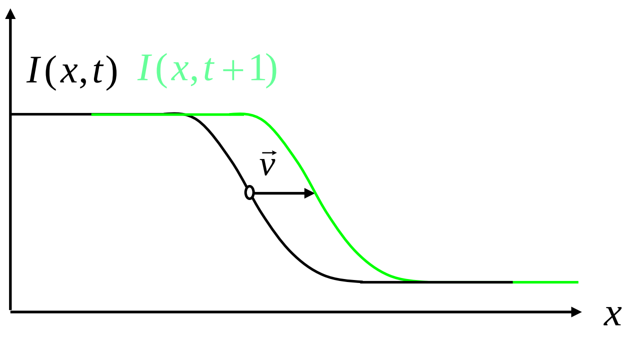
# Aperture Problem in Real Life

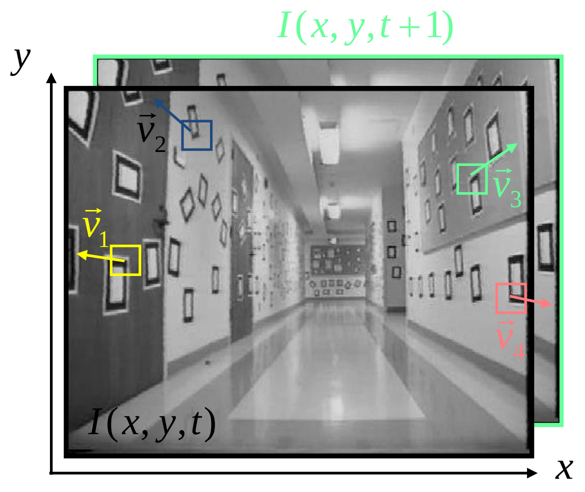
## Aperture Problem

Barber pole illusion



# From 1D to 2D tracking





**The Math is very similar:**

$$\vec{v} \approx -\frac{I_t}{I_x} \quad \Longrightarrow \quad \begin{cases} \vec{v} \approx -G^{-1} b \\ G = \sum_{\text{window around } p} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \\ b = \sum_{\text{window around } p} \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \end{cases}$$

Aperture problem

**Window size here ~ 11x11**

## More Detail: Solving the aperture problem

- How to get more equations for a pixel? -- 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)

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

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

Suppose a 5x5 window

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

# 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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

# 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 $\lambda_1$ and $\lambda_2$ of **$A^T A$** should not be too small
- **$A^T A$** should be well-conditioned
  – $\lambda_1 / \lambda_2$ should not be too large ($\lambda_1$ = larger eigenvalue)

# Eigenvectors of A$^T$A

$$A^T A = \left[\begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array}\right] = \sum \left[\begin{array}{c} I_x \\ I_y \end{array}\right] [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

- Suppose (x,y) is on an edge. What is A$^T$A?
  - gradients along edge all point the same direction
  - gradients away from edge have small magnitude

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

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

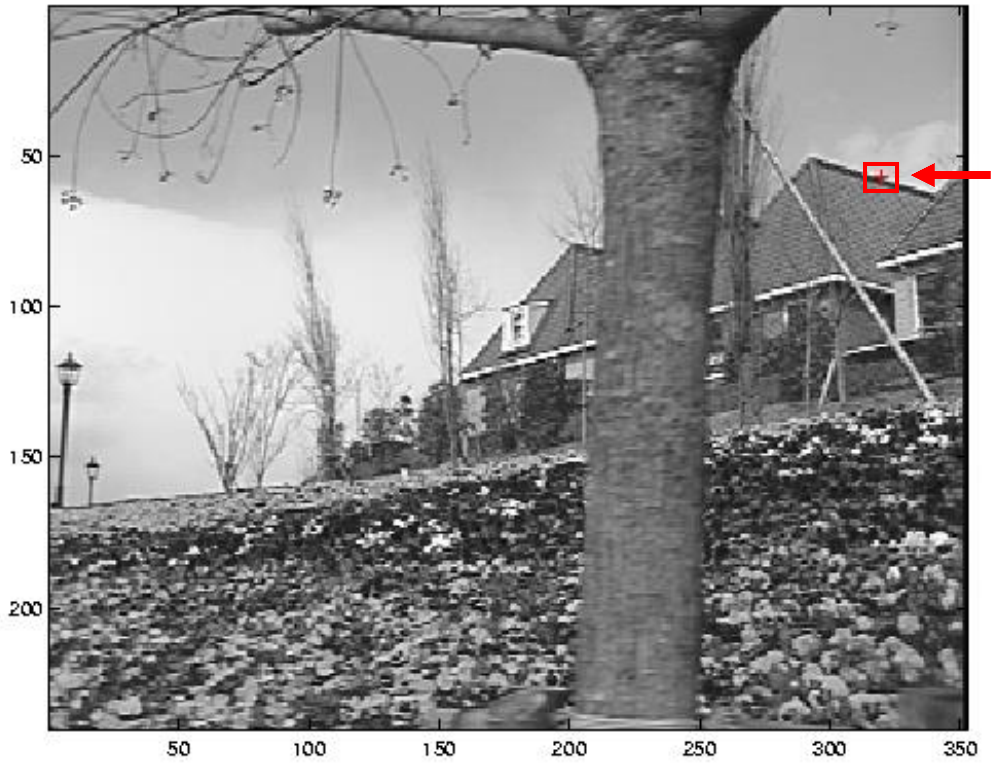
  - $\nabla I$ is an eigenvector with eigenvalue $k\|\nabla I\|$
  - What's the other eigenvector of A$^T$A?
    - let N be perpendicular to $\nabla I$

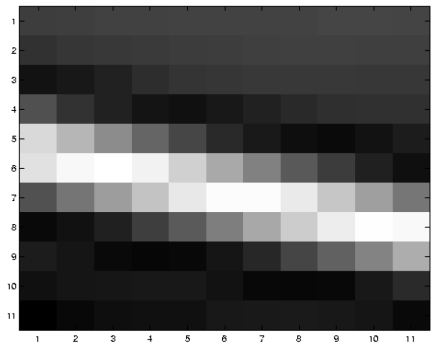
$$\left(\sum \nabla I (\nabla I)^T\right) N = 0$$

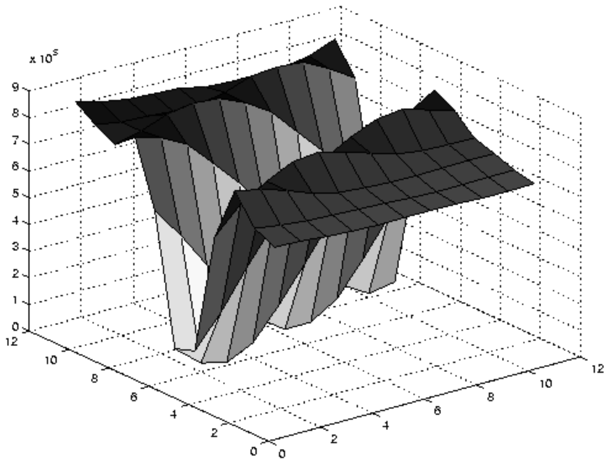
    - N is the second eigenvector with eigenvalue 0
- The eigenvectors of A$^T$A relate to edge direction and magnitude

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

# Edge







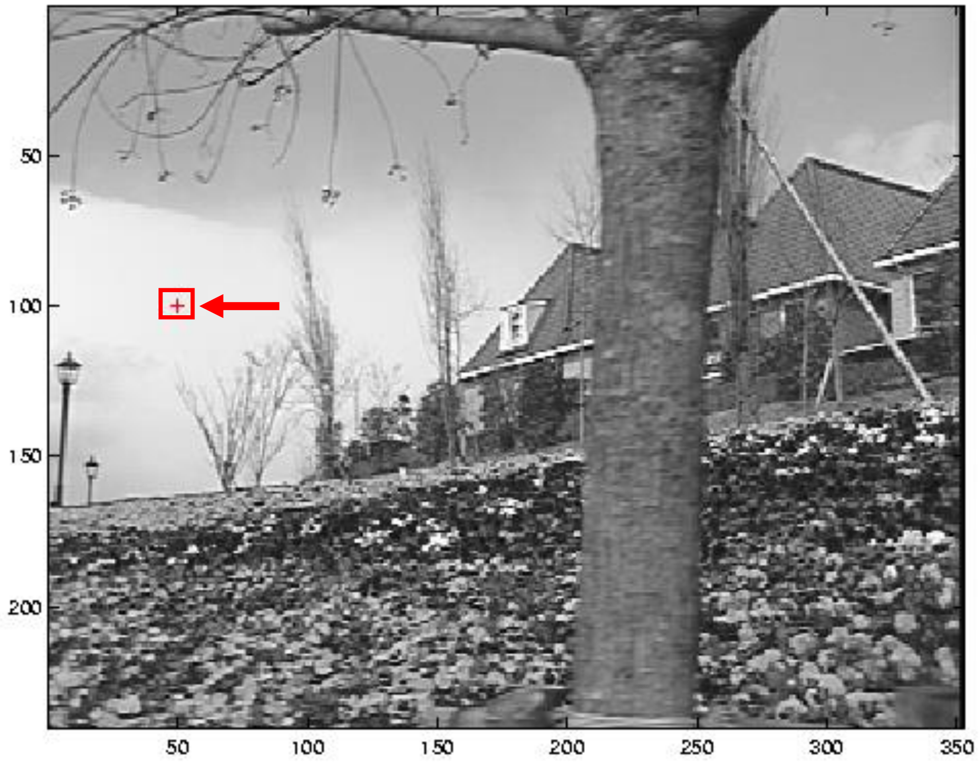
$$\sum \nabla I (\nabla I)^T$$

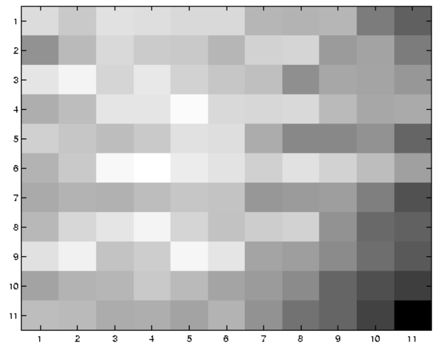
– large gradients, all the same

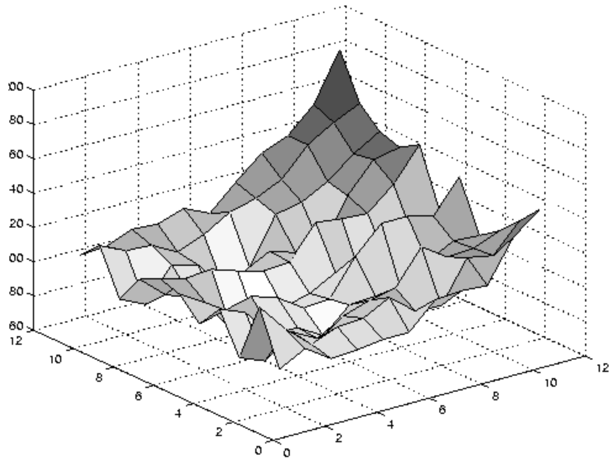
– large $\lambda_1$, small $\lambda_2$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

# Low texture region







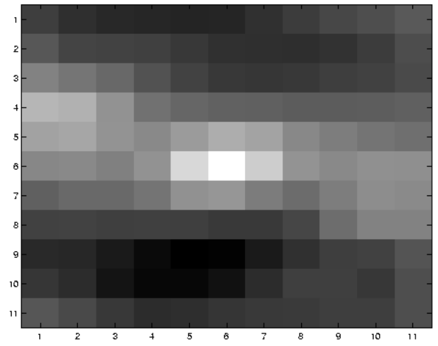
$$\sum \nabla I (\nabla I)^T$$

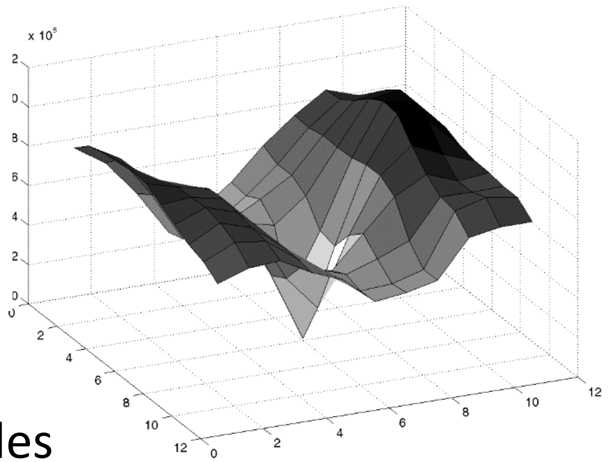
– gradients have small magnitude

– small $\lambda_1$, small $\lambda_2$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

# High textured region





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

– gradients are different, large magnitudes

– large $\lambda_1$, large $\lambda_2$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

## 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?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

## Improving accuracy

- Recall our small motion assumption

$$0 = I(x+u, y+v) - \boldsymbol{I_{t-1}(x,y)}$$

$$\approx I(x,y) + I_x u + I_y v - \boldsymbol{I_{t-1}(x,y)}$$

- This is not exact
  - To do better, we need to add higher order terms back in:

$$= I(x,y) + I_x u + I_y v + \text{higher order terms} - \boldsymbol{I_{t-1}(x,y)}$$

- 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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

## Iterative Refinement

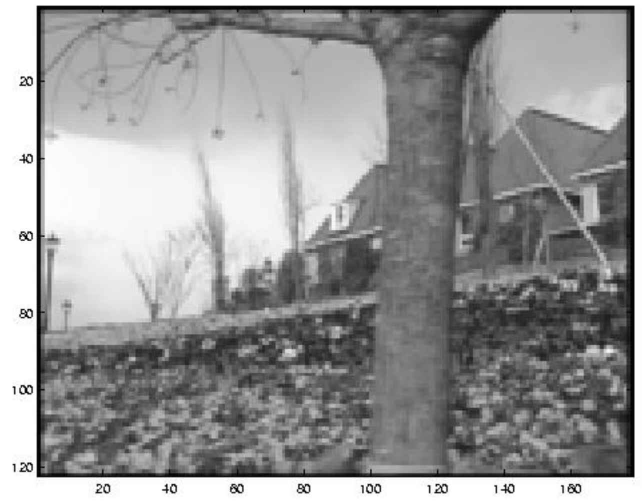
- Iterative Lukas-Kanade Algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp I(t-1) towards I(t) using the estimated flow field
     - *use image warping techniques*
  3. Repeat until convergence

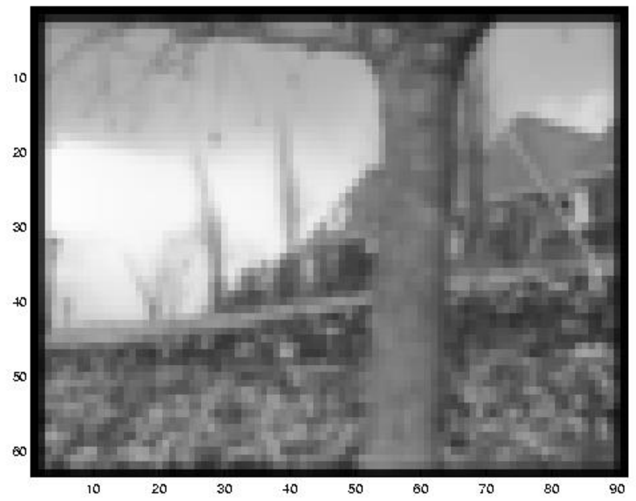
## Revisiting the small motion assumption

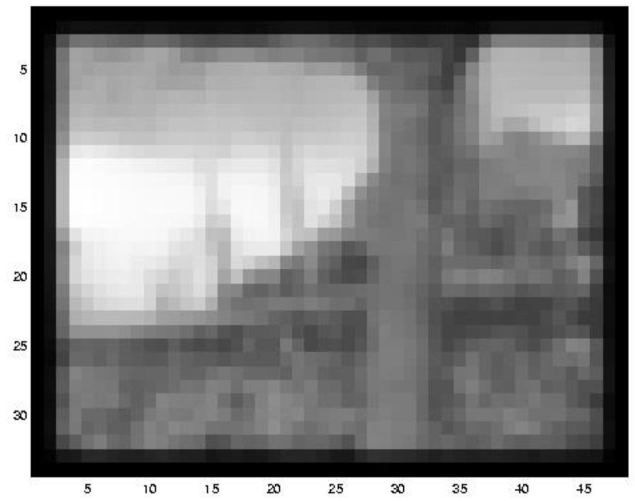
- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2nd order terms dominate)
  - How might we solve this problem?

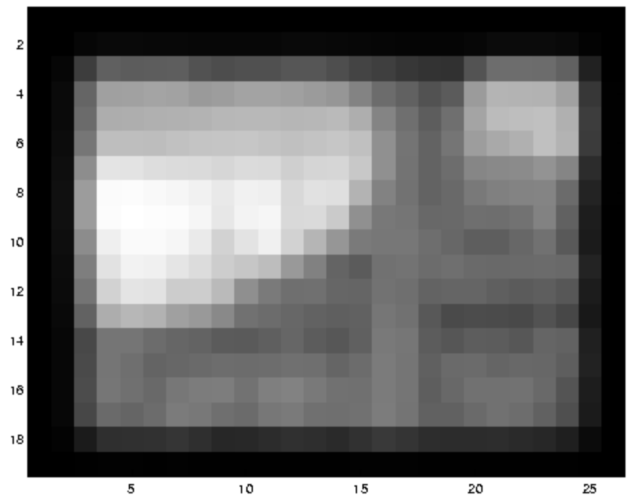
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

# Reduce the resolution!

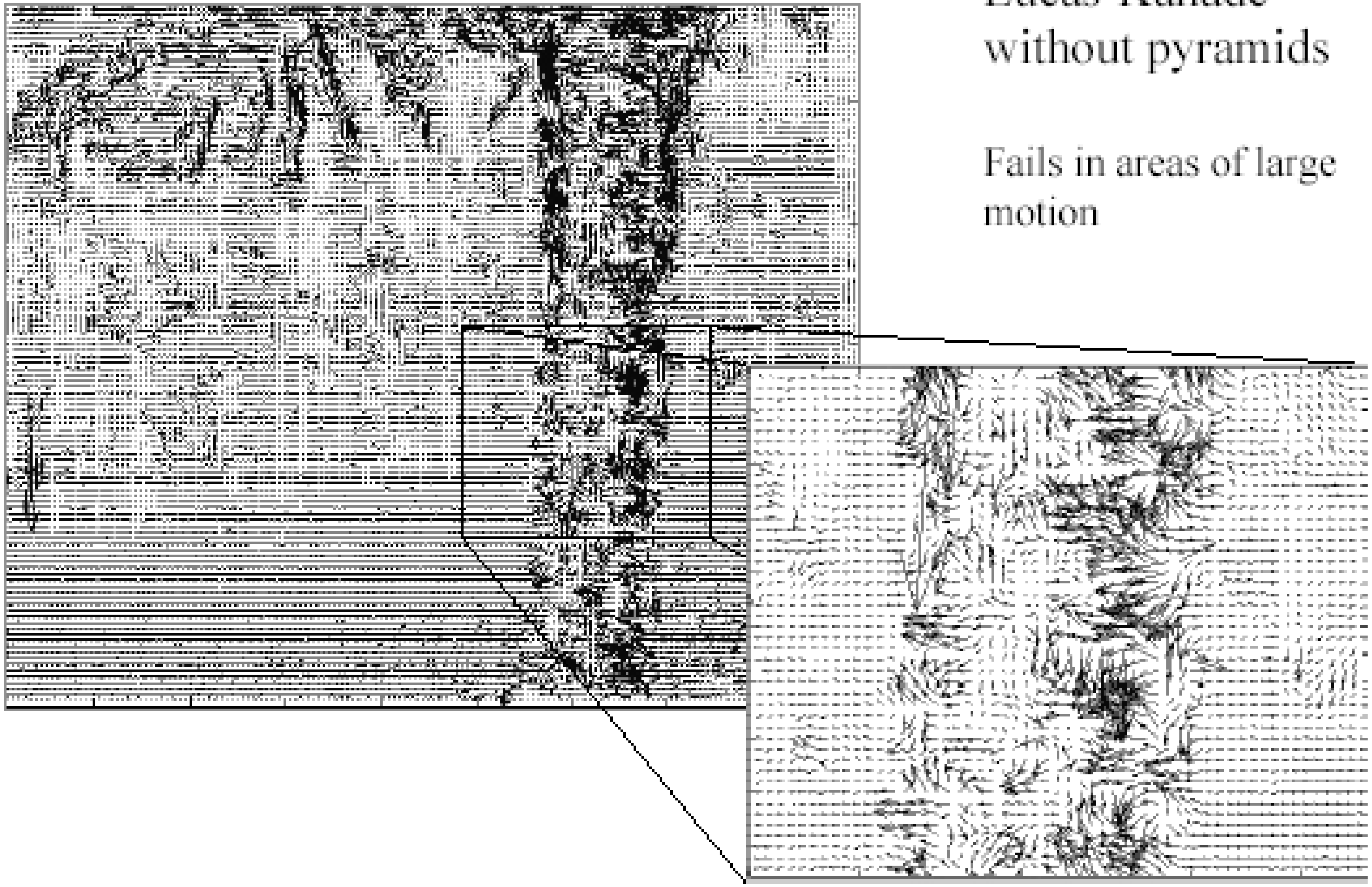








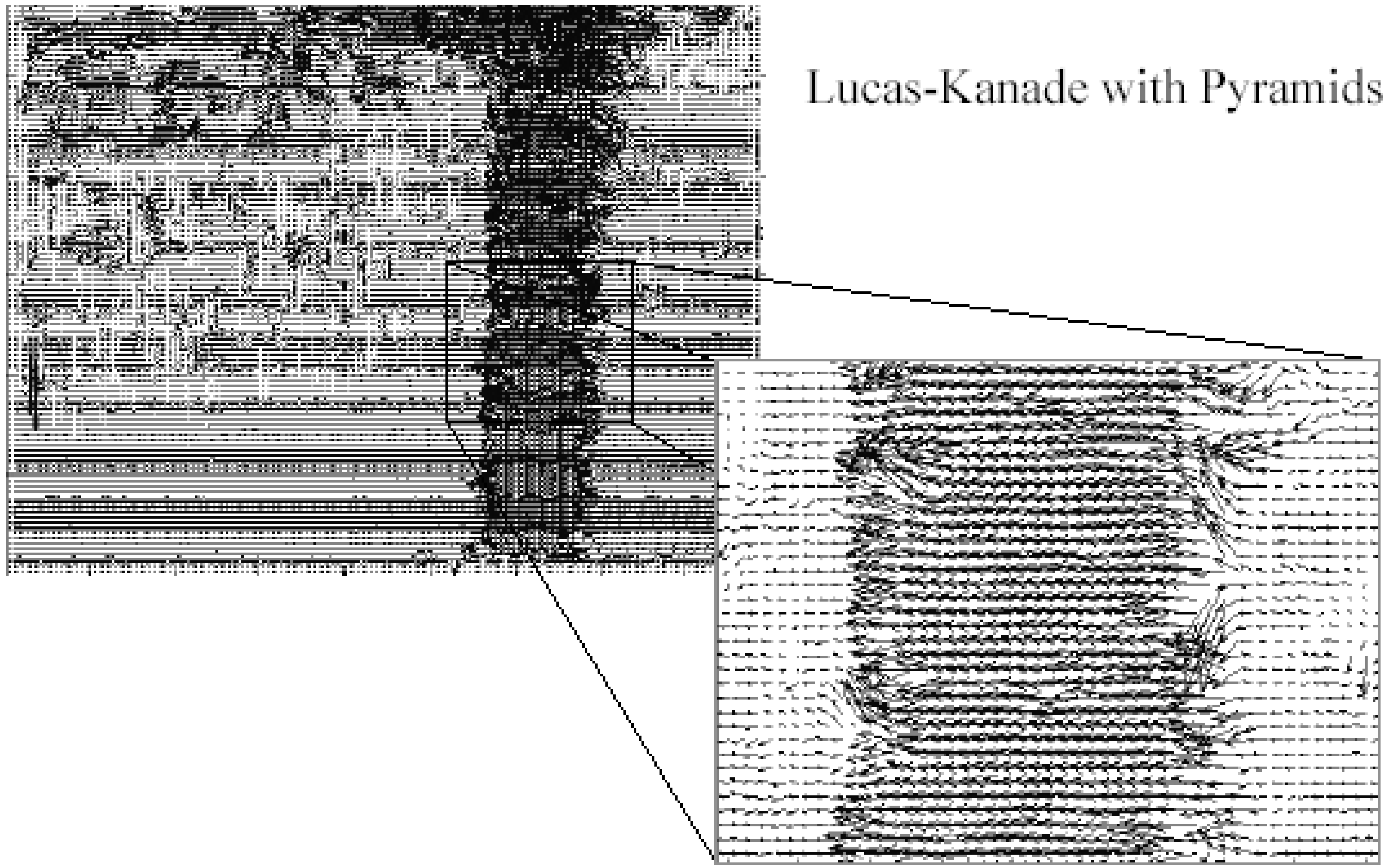
# Optical Flow Results



Lucas-Kanade without pyramids

Fails in areas of large motion

# Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

## What about other types of motion?

## Generalization

- Transformations/warping of image

$$E(\mathbf{A}, \mathbf{h}) = \sum_{\mathbf{x} \in R} \left[ I(\mathbf{A}\mathbf{x} + \mathbf{h}) - I_0(\mathbf{x}) \right]^2$$

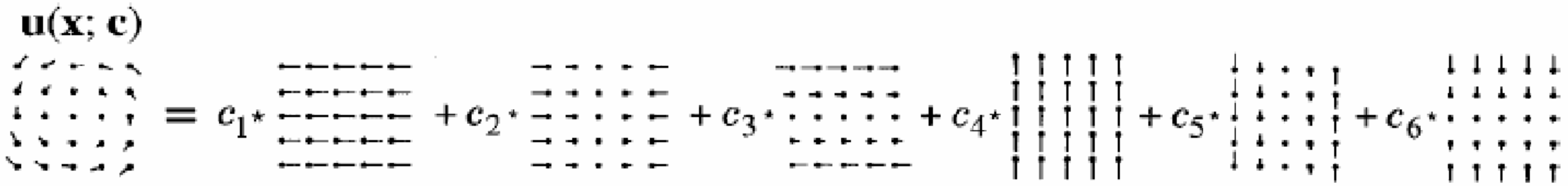
Affine: $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\mathbf{h} = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$

* From Marc Pollefeys COMP 256 2003

# Affine Flow

## Linear Basis

You can think of this as just another set of linear basis functions!



$$\mathbf{u}(\mathbf{x};\,\mathbf{c}) = \sum_{j=1}^{n} a_j \mathbf{b}_j(\mathbf{x})$$

* Slide from Michael Black, CS143 2003

## Horn & Schunck algorithm

Additional smoothness constraint :

$$e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2))dxdy,$$

besides Opt. Flow constraint equation term

$$e_c = \iint (I_x u + I_y v + I_t)^2 dxdy,$$

$\boxed{\text{minimize } e_s + \alpha e_c}$

→

* From Marc Pollefeys COMP 256 2003

## Horn & Schunck algorithm

In simpler terms: If we want dense flow, we need to regularize what happens in ill conditioned (rank deficient) areas of the image. We take the old cost function:

$$d = \arg\min_d \sum_{x \in N} \left( I(x,t) - I(x+d, t+1) \right)^2$$

And add a regularization term to the cost:

$$d = \arg\min_d \sum_{x \in N} \left( I(x,t) - I(x+d, t+1) \right)^2 + \alpha \left\| d \right\|$$

Convex Program!

We will see a lot of such formulations in in robust regression!

## Discussion: What are the other methods to improve optical flows?