MAEG 5720:
Computer Vision in Practice

Lecture 6:

Optical Flow

Dr. Terry Chang

2021-2022

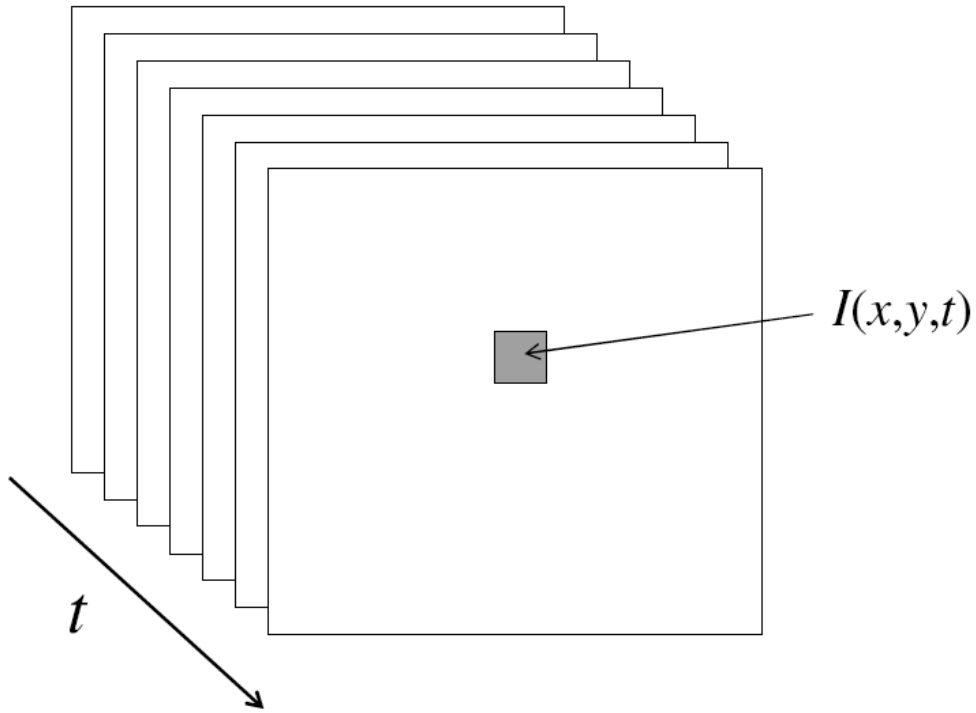
Semester 1

香港中文大學
The Chinese University of Hong Kong

Department of Mechanical and Automation Engineering
機械與自動化工程學系

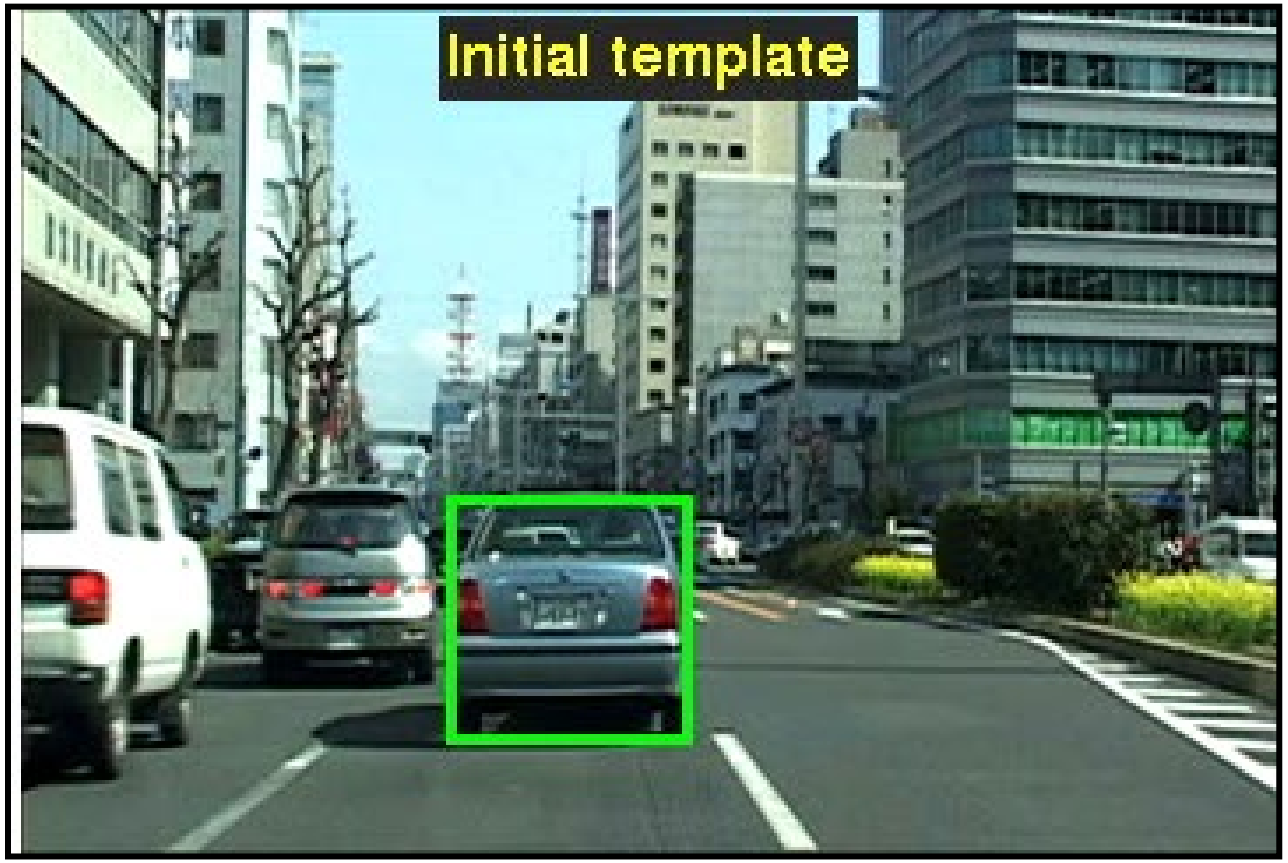
From Images to Videos

- A Video is a sequence of frames captured over time.
- Now the image data becomes a function of *space(x, y)* and *time (t)*



Motivation

- Tracking of object



Source: S. Narasimhan

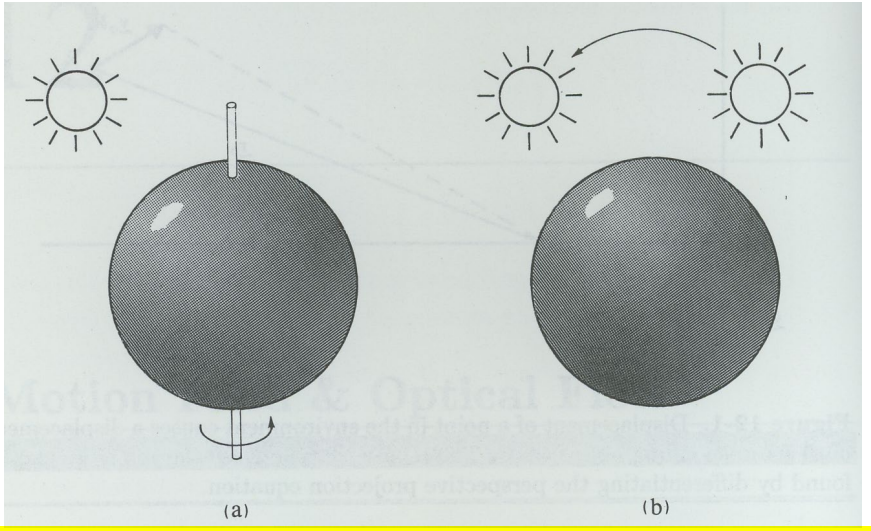
Motion Estimation techniques

- Feature-tracking
  - ***Extract*** of ***visual feature*** (corners) and ***"track"*** them over multiple frames.

- Optical flow
  - ***Recover image motion*** at each pixel from spatio-temporal image brightness variation

Optical flow

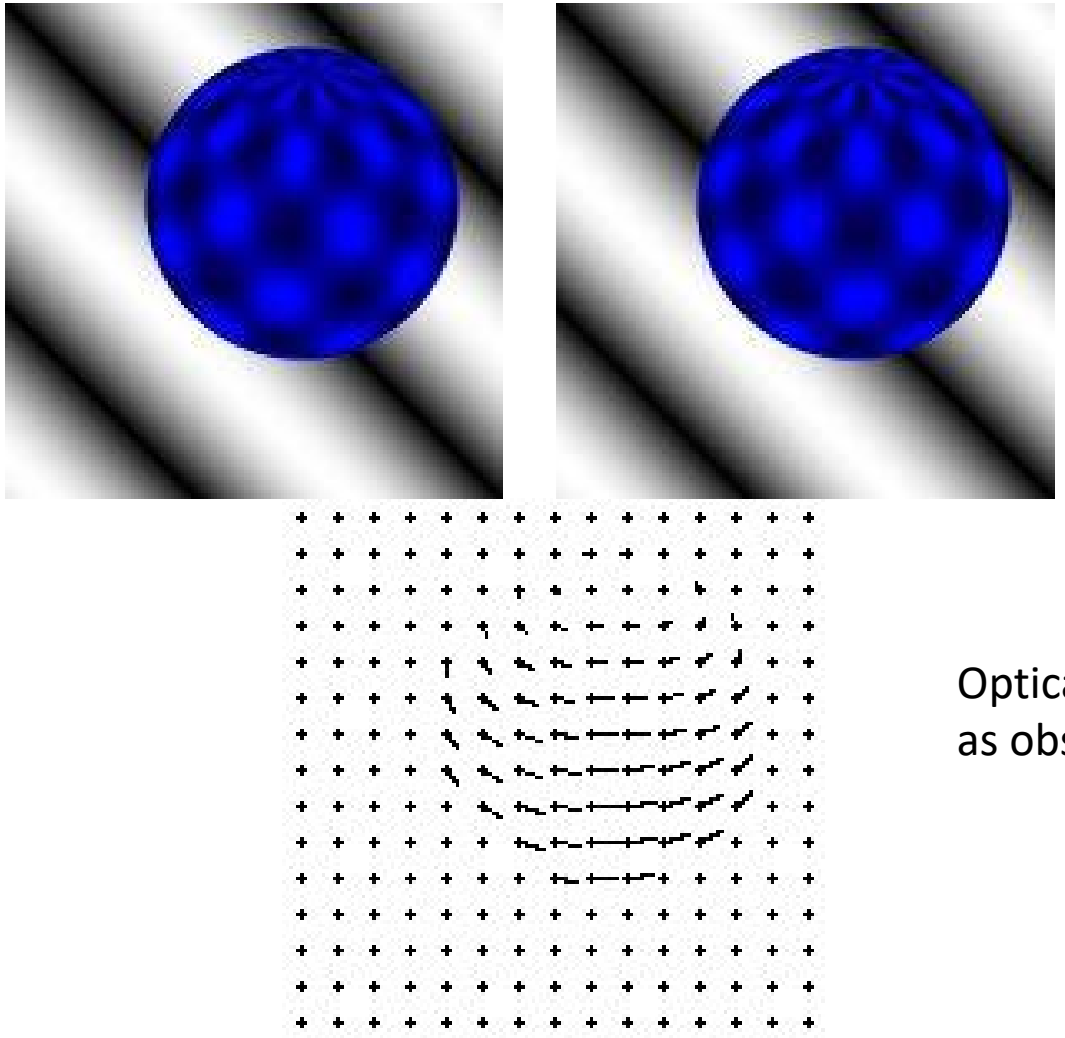
- **Definition:** ***optical flow*** is the apparent ***motion*** of brightness patterns in the images
- **Note:** apparent motion can be caused by ***lighting changes without*** any ***actual motion***



- **Goal:** To Recover the image motion at each pixel from optical

Source: S. Narasimhan

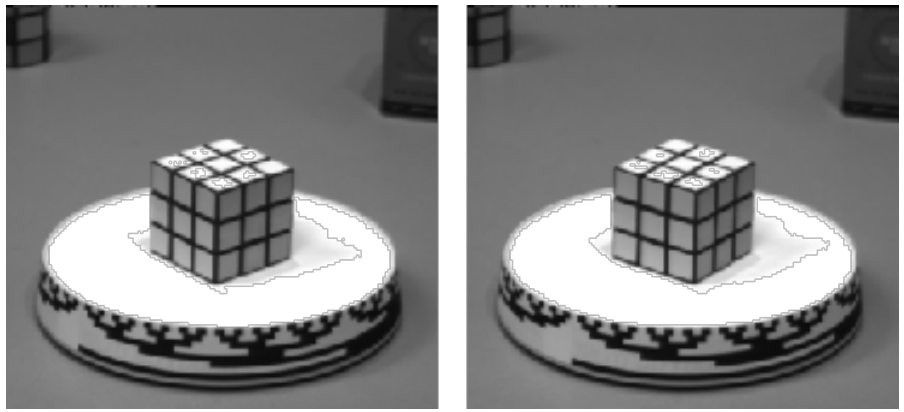
Optical Flow is NOT 3D motion field

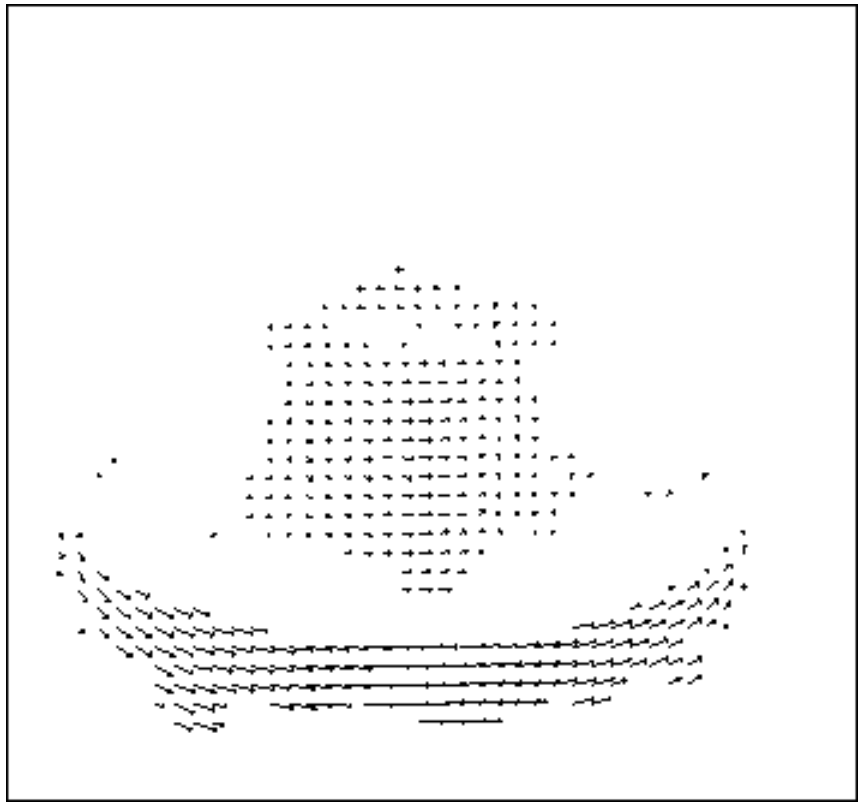


Optical flow: Pixel motion field as observed in image

http://of-eval.sourceforge.net/

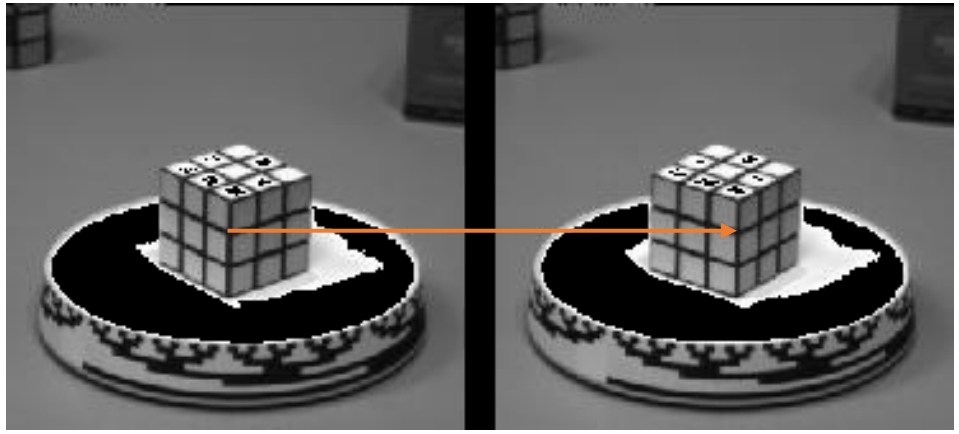
Dense Optical Flow





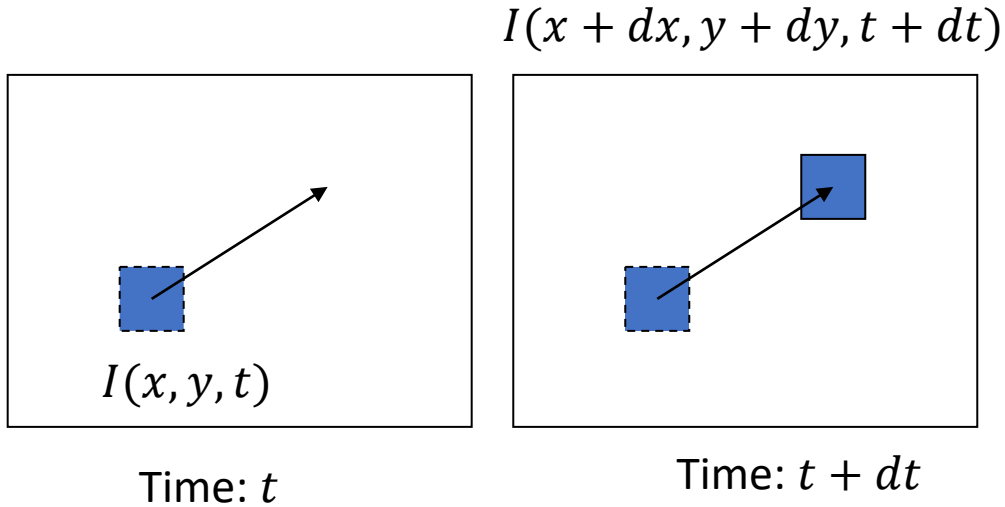
Optical Flow starts with Brightness Constancy

- Assume Points ***movement is small,*** and ***brightness is constant***



- Brightness constancy assumption

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$



Optical Flow Constraint Equation

- Optical Flow Starts with ***Brightness Constancy assumption***

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

- By Mutivariable Taylor's Expansion $f(x, y) \approx f(a, b) + f_x(a, b)(x - a) + f_y(a, b)(y - b)$

Assuming Small Motion

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt$$

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt$$

- Then we have (Brightness Constancy Equation)

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0$$

- In simplified notation

$$I_x dx + I_y dy + I_t dt = 0$$

$$I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t \frac{dt}{dt} = 0$$

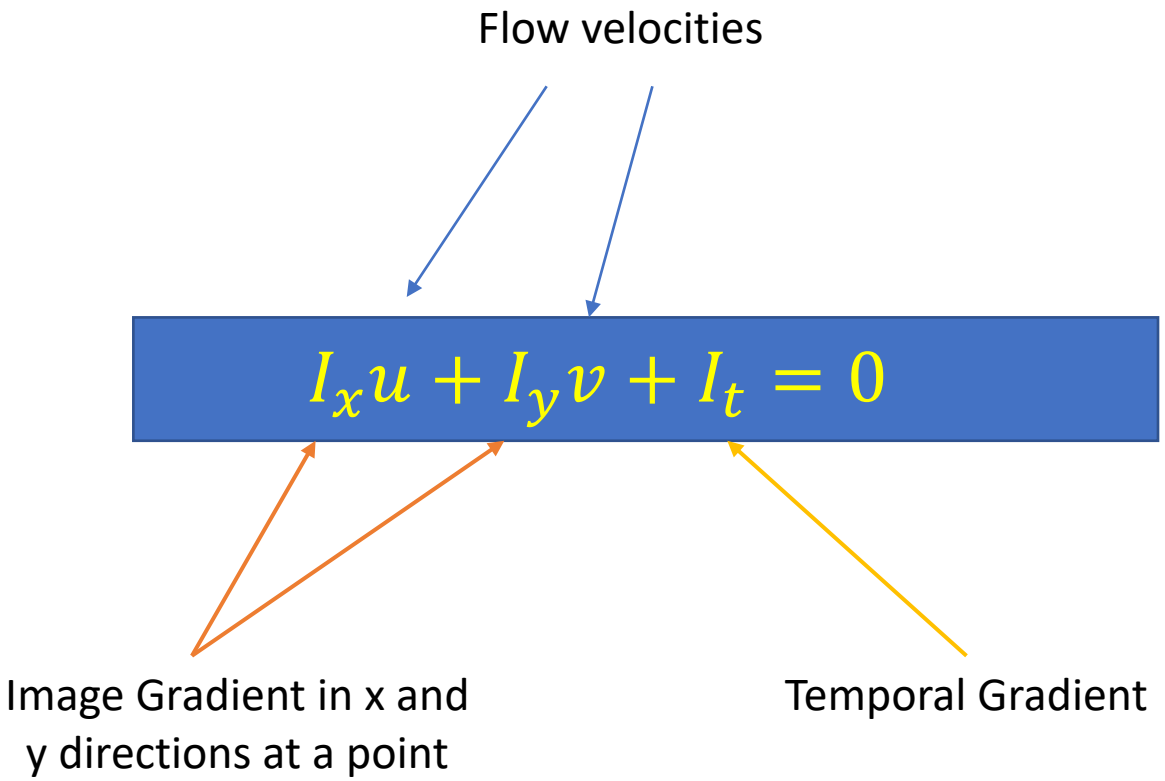
$$I_x u + I_y v + I_t = 0$$

- In vector form

$$\nabla I^T \boldsymbol{v} + I_t = 0$$

(1x2) (2x1)

What do the term represents?

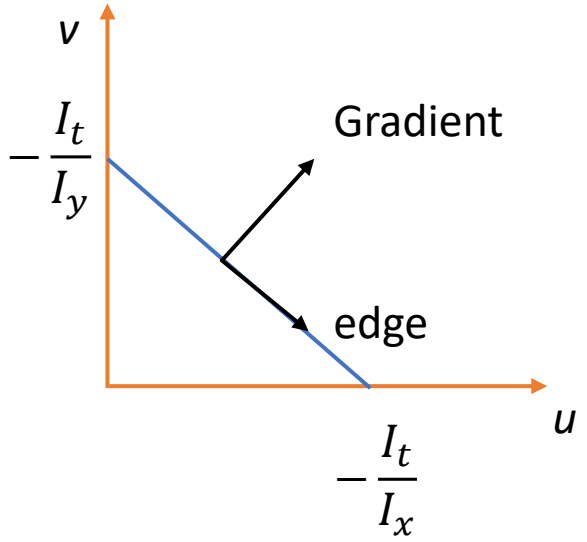


Interpretation of optical flow equation

Unknown (components of flow vector)

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

known (Spatial and Temporal gradients)



- $v = -\frac{I_x}{I_y} u - \frac{I_t}{I_y}$ ***Equation of line***

- Can we use this equation to recover image motion (u, v)?

Consider 1D Case

By the Brightness Constancy Assumption:

$$I(x,t) = I(x + dx, t + dt)$$

By Taylor Expansion,

$$I_x \frac{dx}{dt} + I_t \frac{dt}{dt} = 0$$

$$v = \frac{dx}{dt} = -\frac{I_t}{I_x}$$

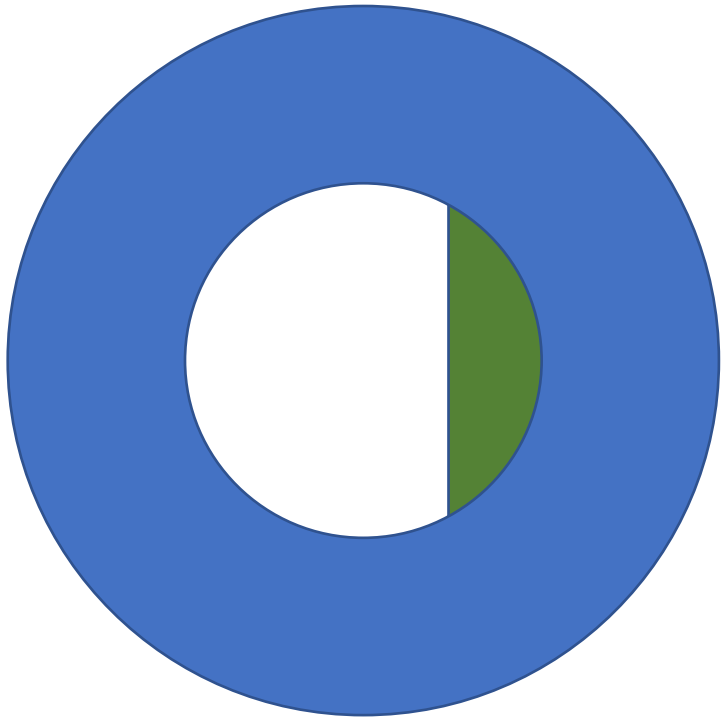
In 2D Case

- In 1D: $I_x \frac{dx}{dt} + I_t \frac{dt}{dt} = 0$
- In 2D: $I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t \frac{dt}{dt} = 0$

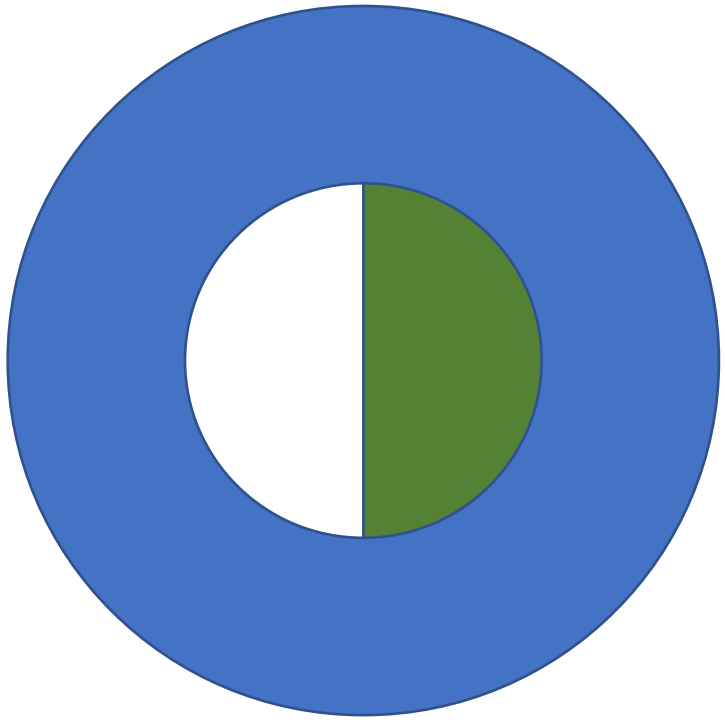
$$I_x u + I_y v + I_t = 0$$

- One equation and two unknown velocity (u, v)
- Under constraint! Need more constraints

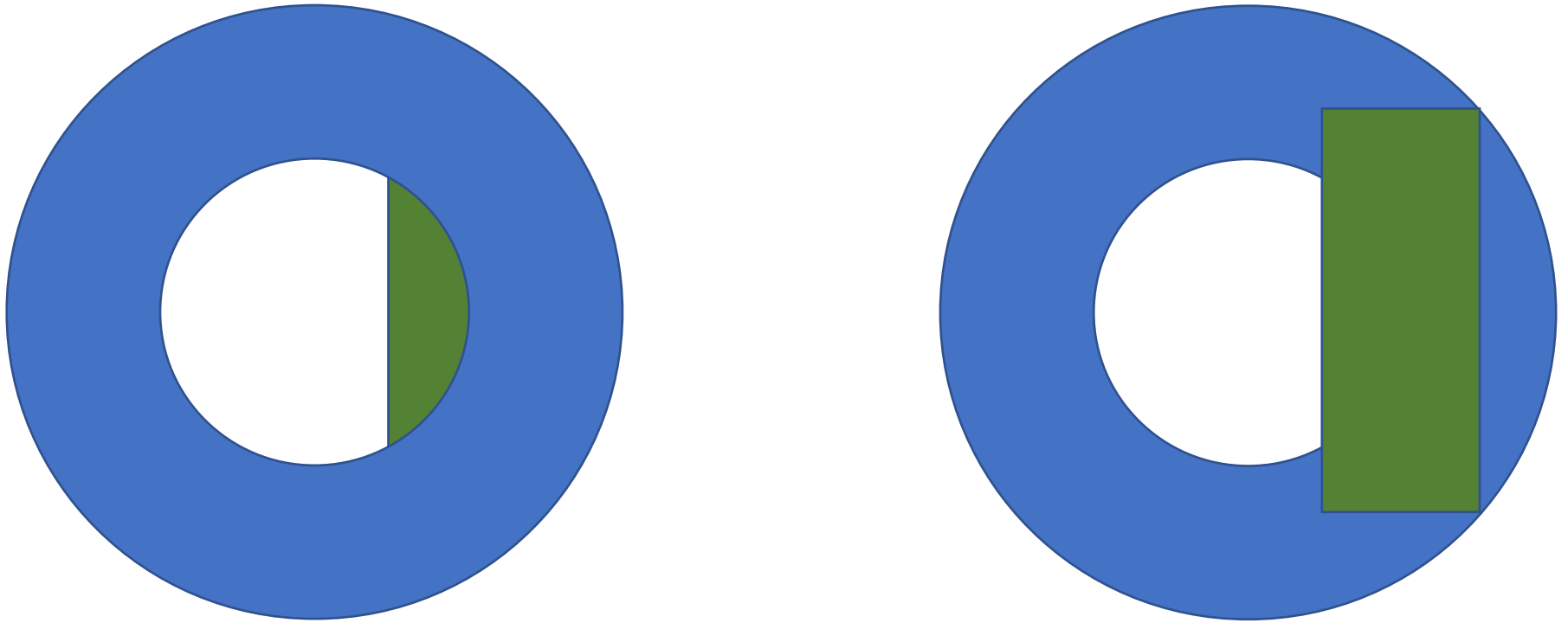
The Aperture Problem



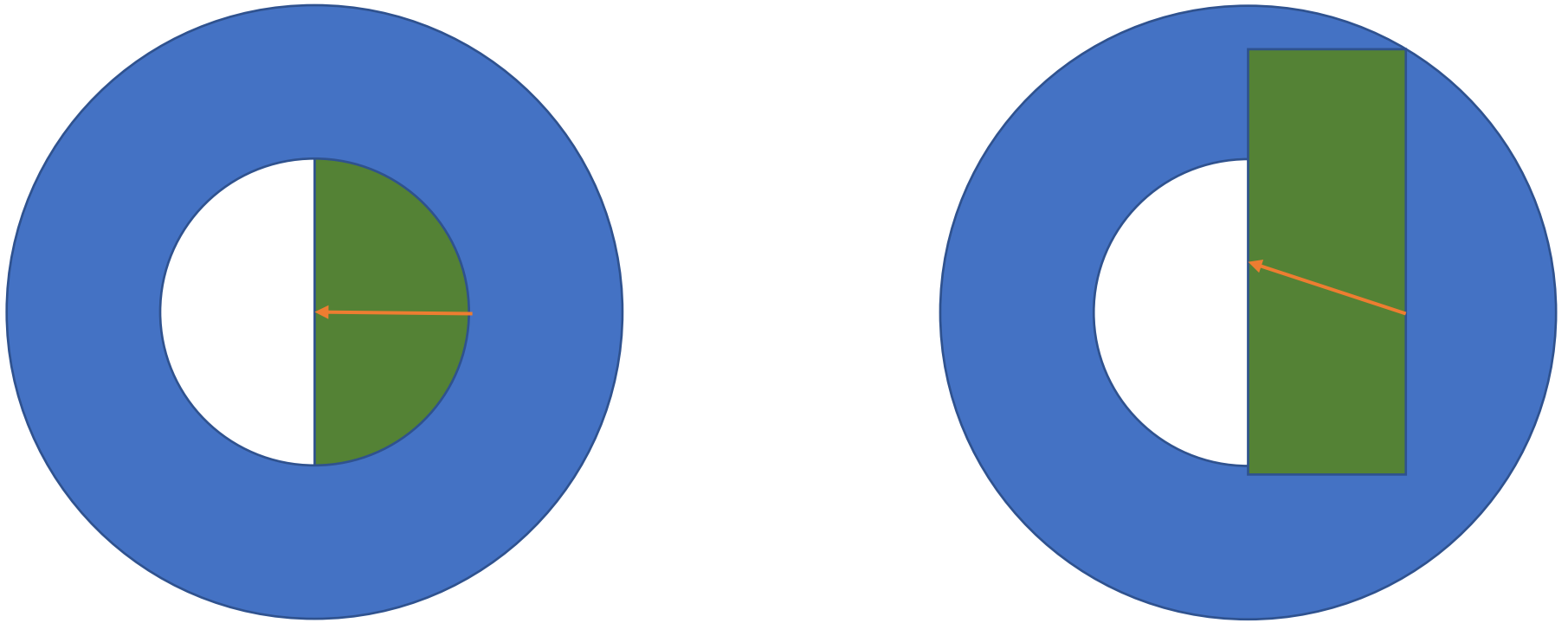
The Aperture Problem



The Aperture Problem

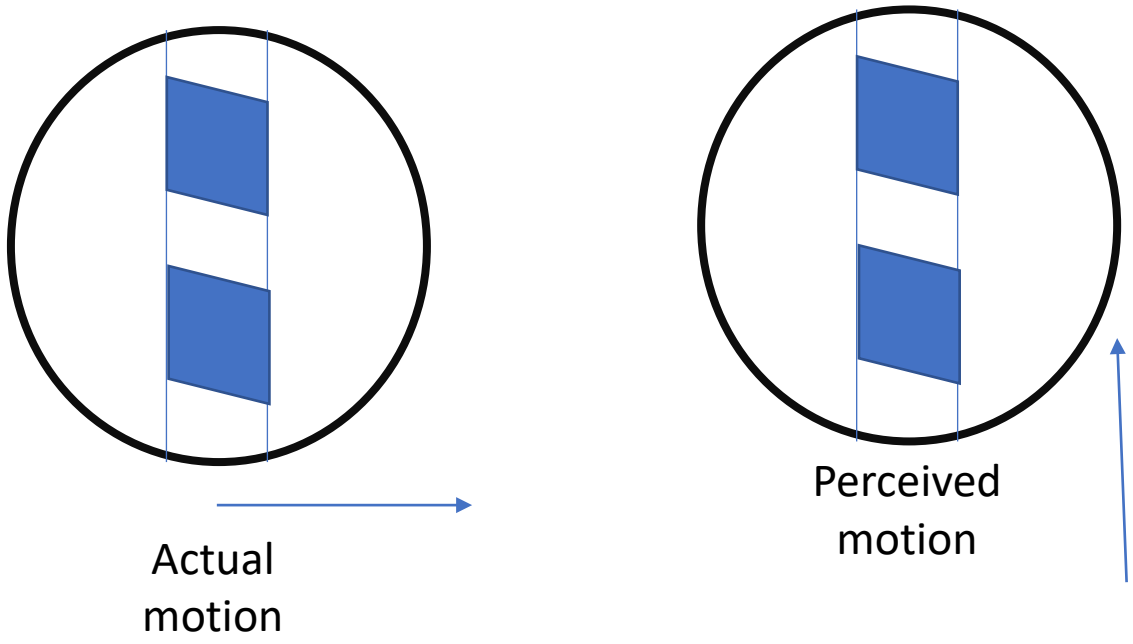


The Aperture Problem



Aperture problem

- Barber Pole illusion

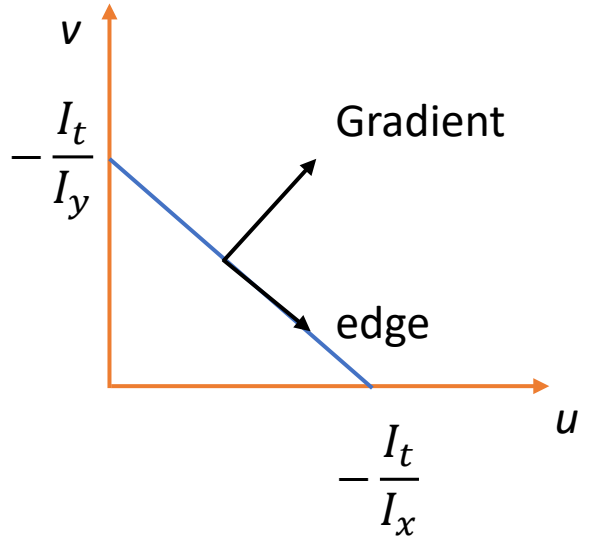


http://www.liv.ac.uk/~marcob/Trieste/barberpole.html

Aperture Problem and Normal Flow

- One question and two unknowns

$$v = -\frac{I_x}{I_y}u - \frac{I_t}{I_y}$$



- What does this constraints mean?
  - The component of the flow in the *gradient direction* is determined (*Normal flow*)
  - The component of the *flow parallel to an edge is unknown*

Computing Optical Flow

- Algorithm for computing Optical flow are two types
  - *Regularization Methods – Horn and Schunck (1980)*
    - By imposing smoothness constraint
  - *Local Methods – Lucas Kanade (1981)*
    - By assuming constant flow

Horn and Schunck

- Proposed by Berthold K.P. Horn and Brian G. Schunck (1980)
  - Apply smoothness constraint
  - Iterative method was proposed

Determining Optical Flow

Berthold K.P. Horn and Brian G. Schunck
Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

ABSTRACT
Optical flow cannot be computed locally, since only one independent measurement is available from the image sequence at a point, while the flow velocity has two components. A second constraint is needed. A method for finding the optical flow pattern is presented which assumes that the apparent velocity of the brightness pattern varies smoothly almost everywhere in the image. An iterative implementation is shown which successfully computes the optical flow for a number of synthetic image sequences. The algorithm is robust in that it can handle image sequences that are quantized rather coarsely in space and time. It is also insensitive to quantization of brightness levels and additive noise. Examples are included where the assumption of smoothness is violated at singular points or along lines in the image.

1. Introduction

Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image. Optical flow can arise from relative motion of objects and the viewer [6, 7]. Consequently, optical flow can give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement [8]. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects [27]. Attempts have been made to perform such segmentation using differences between successive image frames [15, 16, 17, 20, 25]. Several papers address the problem of recovering the motions of objects relative to the viewer from the optical flow [10, 18, 19, 21, 29]. Some recent papers provide a clear exposition of this enterprise [30, 31]. The mathematics can be made rather difficult, by the way, by choosing an inconvenient coordinate system. In some cases information about the shape of an object may also be recovered [3, 18, 19].

These papers begin by assuming that the optical flow has already been determined. Although some reference has been made to schemes for comput-

Artificial Intelligence **17** (1981) 185–203
0004–3702/81/0000–0000/$02.50

Horn & Schunk (Regularization)

- Brightness Constancy Equation

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

- How to solve the equation?

- Imposing smoothness Constraints

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$$

- The gradient of optical flow velocity u and v should small

Horn & Schunk

- Error in optical flow:

$E(u,v)=\iint\{E_c+\lambda E_s\}dxdy$

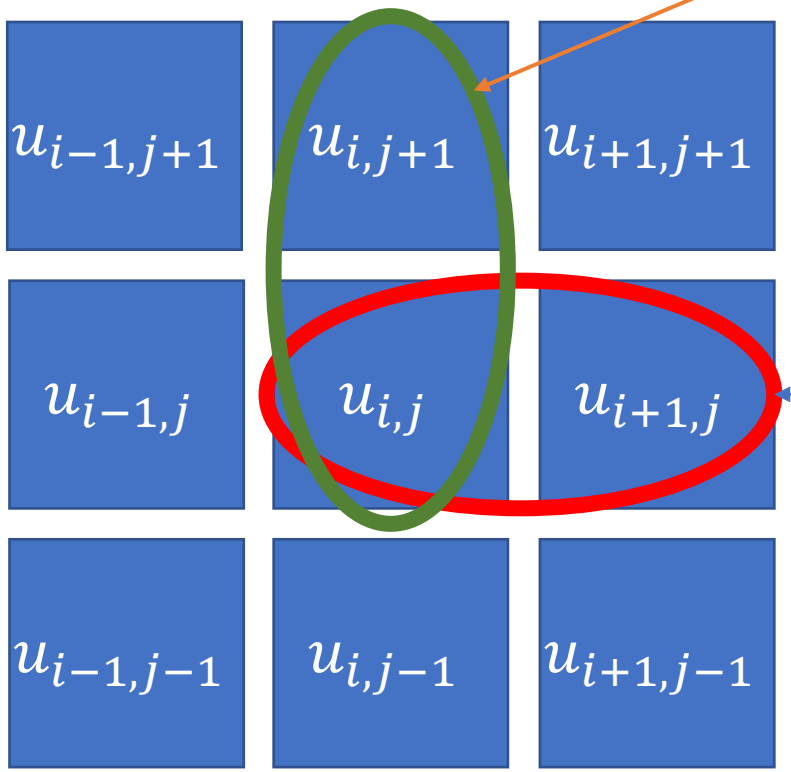
  - Smoothness constraint is added.
  - λ is a parameter
  - u and v are x and y components of motion respectively

$E_c=\left(I_x u+I_y v+I_t\right)^2 \qquad E_s=u_x^2+u_y^2+v_x^2+v_y^2$

$E(u,v)=\iint\left\{\left(I_x u+I_y v+I_t\right)^2+\lambda(u_x^2+u_y^2+v_x^2+v_y^2)\right\}dxdy$

What is smooth flow?

- Smooth flow means

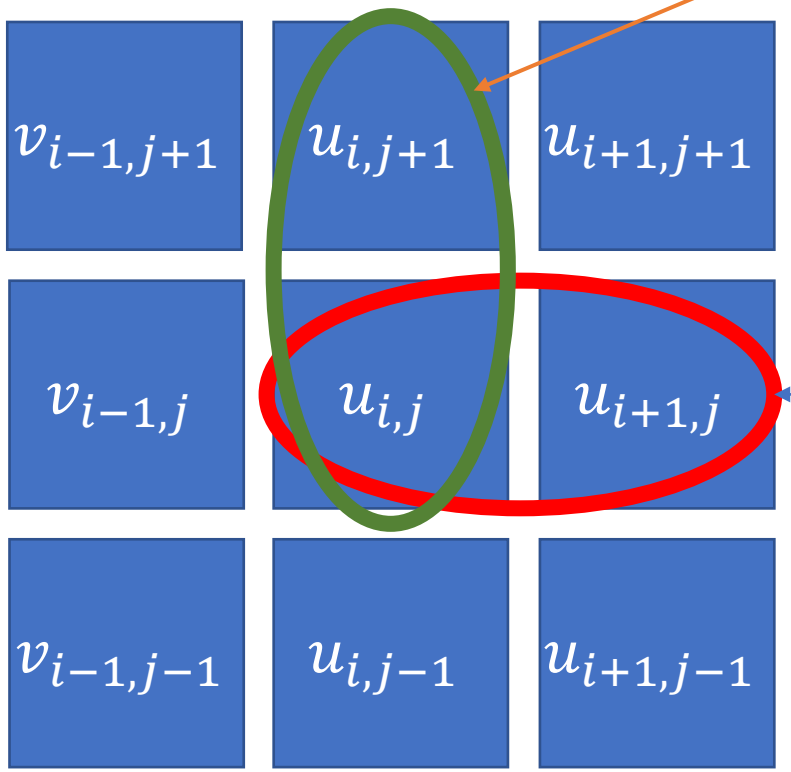


$$\min_u (u_{i,j+1} - u_{i,j})^2$$

$$\min_u (u_{i+1,j} - u_{i,j})^2$$

What is smooth flow?

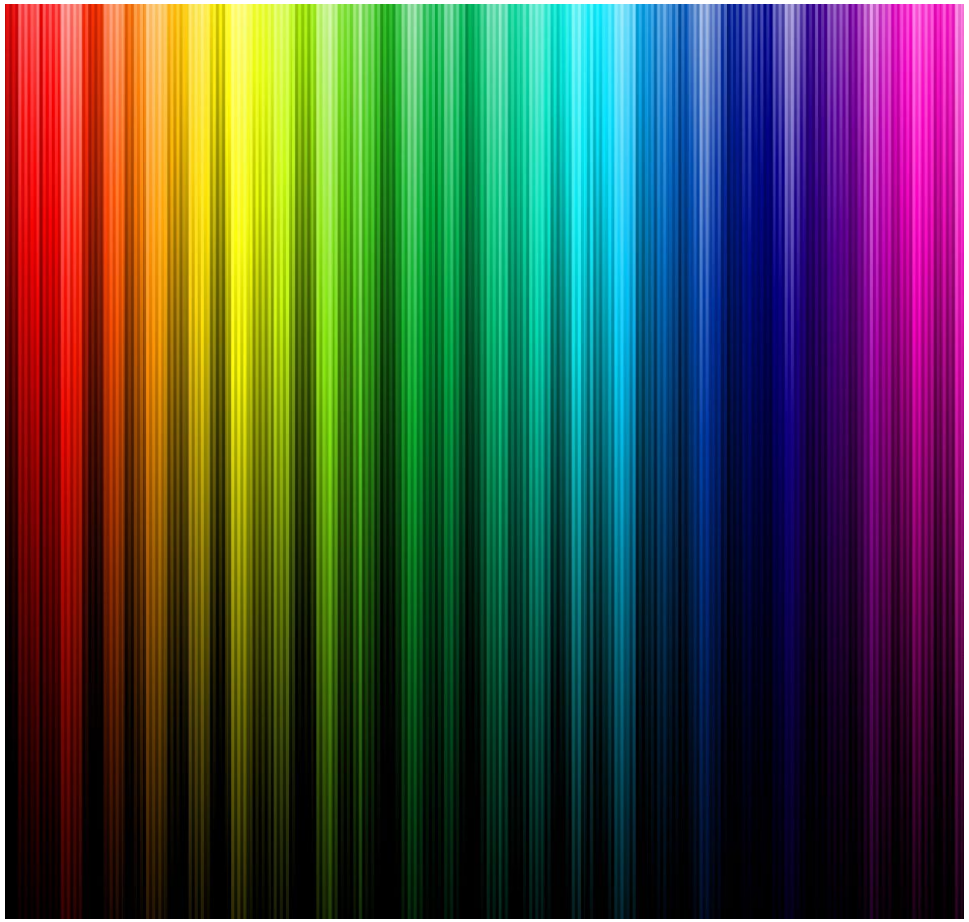
- Smooth flow means

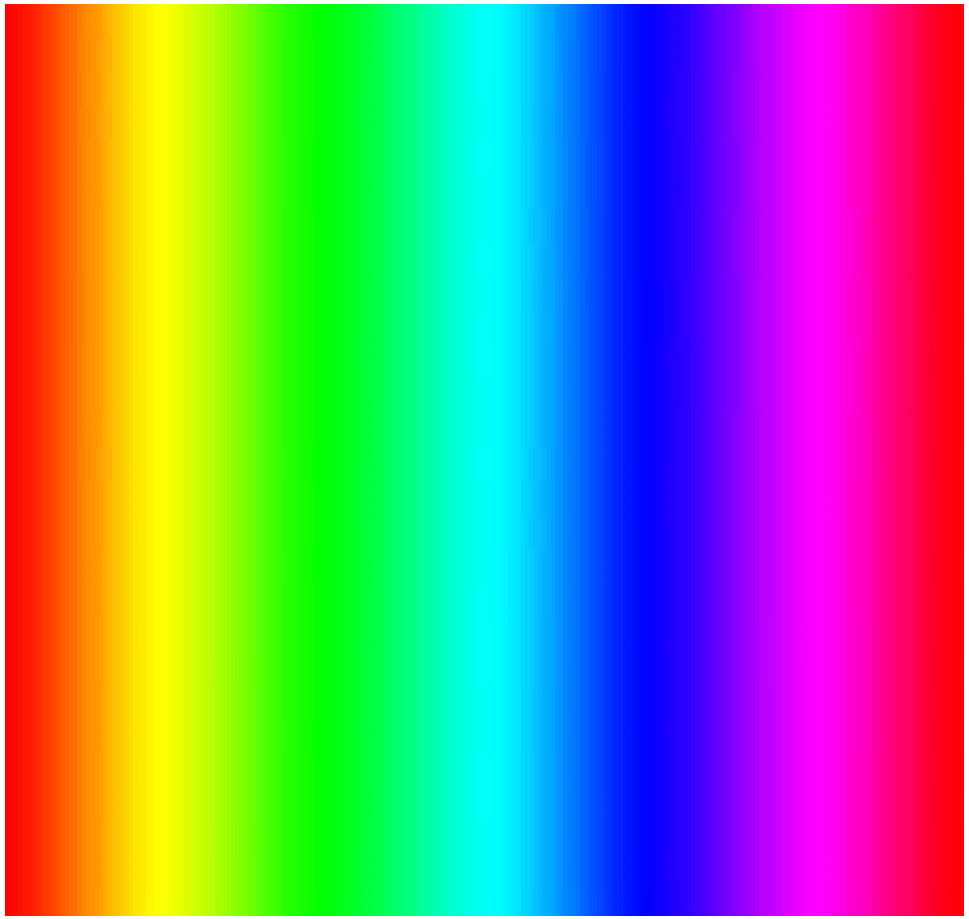


$$\min_u (u_{i,j+1} - u_{i,j})^2$$

$$\min_u (u_{i+1,j} - u_{i,j})^2$$

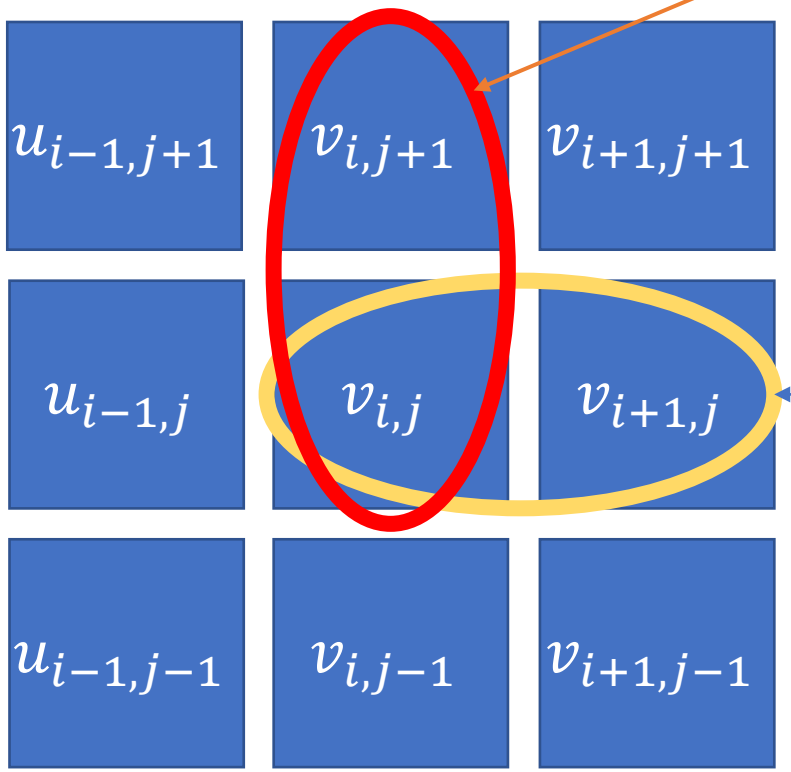
The flow is smooth?





What is smooth flow?

- Smooth flow means



$$\min_{v}(v_{i,j+1} - v_{i,j})^2$$

$$\min_{u}(v_{i+1,j} - v_{i,j})^2$$

Horn & Schunk

- We have, error in optical flow

$$E(u,v) = \iint \left\{ \left(I_x u + I_y v + I_t\right)^2 + \lambda \overbrace{\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right)}^{f} \right\} dxdy$$

Brightness Constancy constraint Smoothness constraint

- Find (u, v) at each image that minimize error
  - Take partial derivative w.r.t. u and v

$$\frac{\partial E}{\partial u} = 2I_x\left(I_x u + I_y v + I_t\right) + \lambda\left(\frac{\partial f}{\partial u}\right)$$

$$\frac{\partial E}{\partial v} = \left(I_x u + I_y v + I_t\right)I_y + \lambda\left(\frac{\partial f}{\partial u}\right)$$

$$\frac{\partial f}{\partial u} = \frac{\partial}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial}{\partial u}\frac{\partial u}{\partial y} = 2(u_{xx} + u_{yy})$$

$$\frac{\partial f}{\partial v} = \frac{\partial}{\partial v}\frac{\partial v}{\partial x} + \frac{\partial}{\partial v}\frac{\partial v}{\partial y} = 2(v_{xx} + v_{yy})$$

Laplacian of u

Laplacian of v

$$\Delta^2 u = u_{xx} + u_{yy}$$

Taking Partial Derivative

- Taking Partial Derivative w.r.t. *u*

$$\frac{\partial E}{\partial u} = 2I_x\left(I_x u + I_y v + I_t\right) + \lambda\left(\frac{\partial f}{\partial u}\right) \quad where\ f = u_x^2 + u_y^2 + v_x^2 + v_y^2$$

- What is $\frac{\partial f}{\partial u}$?

$$\frac{\partial f}{\partial u} = \frac{\partial}{\partial u}\frac{\partial f}{\partial x} + \frac{\partial}{\partial u}\frac{\partial f}{\partial y}$$
$$\frac{\partial f}{\partial u} = \frac{\partial}{\partial u}2u_x + \frac{\partial}{\partial u}2u_y$$
$$\frac{\partial f}{\partial u} = 2(u_{xx} + u_{yy})$$

Laplacian of u

$$\Delta^2 u = u_{xx} + u_{yy}$$

Taking Partial Derivative

- Taking Partial Derivative w.r.t. *u*

$$\frac{\partial E}{\partial u} = 2I_x\left(I_x u + I_y v + I_t\right) + \lambda\left(\frac{\partial f}{\partial u}\right) \quad where\ f = u_x^2 + u_y^2 + v_x^2 + v_y^2$$

- We have $\frac{\partial E}{\partial u}$

$$\frac{\partial E}{\partial u} = 2I_x\left(I_x u + I_y v + I_t\right) + \lambda\left(\Delta^2 u\right) = 0$$

Taking Partial Derivative

- Taking Partial Derivative w.r.t. *v*

$$\frac{\partial E}{\partial v} = 2I_y\left(I_x u + I_y v + I_t\right) + \lambda\left(\frac{\partial f}{\partial v}\right) \quad where\ f = u_x^2 + u_y^2 + v_x^2 + v_y^2$$

- We have $\frac{\partial E}{\partial v}$

$$\frac{\partial E}{\partial v} = 2I_y\left(I_x u + I_y v + I_t\right) + \lambda\left(\Delta^2 v\right) = 0$$

Laplacian Mask

- The Laplacian is estimated by subtracting the value at a point from a weighted average of the values at neighboring points.

$\Delta^2 f = f - f_{av}$

$$\begin{bmatrix} 0 & -\frac{1}{4} & 0 \\ -\frac{1}{4} & 1 & -\frac{1}{4} \\ 0 & -\frac{1}{4} & 0 \end{bmatrix}$$

Or

$$\begin{bmatrix} -\frac{1}{12} & -\frac{1}{6} & -\frac{1}{12} \\ -\frac{1}{6} & 1 & -\frac{1}{6} \\ -\frac{1}{12} & -\frac{1}{6} & -\frac{1}{12} \end{bmatrix}$$

Horn & Schunk (cont'd)

- Error in optical flow
  - $\iint \left\{ \left(I_x u + I_y v + I_t \right)^2 + \lambda (u_x^2 + u_y^2 + v_x^2 + v_y^2 \right\} dxdy$

- Find (u, v) at each image that minimize error
  - $\left(I_x u + I_y v + I_t \right) I_x + \lambda \left(\Delta^2 u \right) = 0$ $\quad \Delta^2 u = u_{xx} + u_{yy}$
  - $\left(I_x u + I_y v + I_t \right) I_y + \lambda \left(\Delta^2 v \right) = 0$

- Rearranging the equations
  - $\left(I_x u + I_y v + I_t \right) I_x + \lambda (u - u_{av}) = 0$ Two equations & two unknowns
  - $\left(I_x u + I_y v + I_t \right) I_y + \lambda (v - v_{av}) = 0$

Source: M. Shah

Horn & Schunk (cont'd)

- Discrete version
  - $(I_x u + I_y v + I_t) I_x + \lambda(u - u_{av}) = 0$
  - $(I_x u + I_y v + I_t) I_y + \lambda(v - v_{av}) = 0$

- Then we have
  - $u = u_{av} - I_x \frac{P}{D}$
  - $v = v_{av} - I_y \frac{P}{D}$

    - $P = I_x u_{av} + I_y v_{av} + I_t$
    - $D = \lambda + I_x^2 + I_y^2$

Source: M. Shah

Steps for Horn and Schunck

- $u = u_{av} - I_x \frac{P}{D}$
- $v = v_{av} - I_y \frac{P}{D}$

  - $P = I_x u_{av} + I_y v_{av} + I_t$
  - $D = \lambda + I_x^2 + I_y^2$

- We can iteratively compute u and V
  - Initially assume u, v are zero
  - Compute u_{av} and v_{av} in the neighborhood

Iterative Solution

- *Constrained minimization problem*
  - $u = u_{av} - I_x \frac{P}{D}$
  - $v = v_{av} - I_y \frac{P}{D}$
  - Where $P = I_x u_{av} + I_y v_{av} + I_t \quad \& \quad D = \lambda + I_x^2 + I_y^2$

- By Gauss-Seidel method to
  - $u^{n+1} = u_{av}^n - I_x \frac{P^n}{D}$
  - $v^{n+1} = v_{av}^n - I_x \frac{P^n}{D}$
  - where
    - $P^n = I_x u_{av}^n + I_y v_{av}^n + I_t$

Discrete Optical Flow

- Consider image pixel (i, j)

Error in Optical flow constraint equation

$$c_{ij} = \left(I_x^{ij}\, u_{i,j} + I_y^{ij}\, v_{i,j} + I_t^{ij}\right)^2$$

From smoothness Constraint

$$S_{ij} = \frac{1}{4}\left[\left(u_{i+1,j} - u_{i,j}\right)^2 + \left(u_{i,j+1} - u_{i,j}\right)^2 + \left(v_{i+1,j} - v_{i,j}\right)^2 + \left(v_{i,j+1} - v_{i,j}\right)^2\right]$$

- We seek the set $\{u_{ij}\}$ & $\{v_{ij}\}$that minimize:

$$e = \sum_i \sum_j (\, S_{ij} + \lambda\, c_{ij})$$

Minimization

- We seek the set $\{u_{ij}\}$ & $\{v_{ij}\}$ that minimize:

$$e = \sum_i \sum_j (S_{ij} + \lambda\, c_{ij})$$

- For the smoothness term, we have

$$S_{ij} = \frac{1}{4}[(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2]$$

$$u_{i+1,j}{}^2 - 2u_{i+1,j}\, u_{i,j} + u_{i,j}{}^2 \qquad u_{i,j+1}{}^2 - 2u_{i,j+1}\, u_{i,j} + u_{i,j}{}^2$$

Discrete Optical Flow

- Differentiating e w.r.t. v_{kl} & u_{kl}:

- $\frac{\partial e}{\partial u_{kl}} = 2(u_{kl} - \bar{u}_{kl}) + 2\lambda\left(I_x^{kl}u_{kl} + I_y^{kl}v_{kl} + I_t^{kl}\right)I_x^{kl}$

- $\frac{\partial e}{\partial v_{kl}} = 2(v_{kl} - \bar{v}_{kl}) + 2\lambda\left(I_x^{kl}u_{kl} + I_y^{kl}v_{kl} + I_t^{kl}\right)I_x^{kl}$

- $\bar{u}_{kl}$ & $\bar{v}_{kl}$ are ***average of*** (u_{kl}, v_{kl}) at pixel (k, l)

Finding the Extrema

$$\frac{\partial e}{\partial u_{kl}} = 2(u_{kl} - \bar{u}_{kl}) + 2\lambda\left(I_x^{kl}u_{kl} + I_y^{kl}v_{kl} + I_t^{kl}\right)I_x^{kl} = 0$$
$$\frac{\partial e}{\partial v_{kl}} = 2(v_{kl} - \bar{v}_{kl}) + 2\lambda\left(I_x^{kl}u_{kl} + I_y^{kl}v_{kl} + I_t^{kl}\right)I_y^{kl} = 0$$

Rearranging terms

$$(1 + \lambda {I_x^{kl}}^2)u_{kl} + \lambda I_x^{kl} I_y^{kl} v_{kl} = -\lambda I_x^{kl} I_t^{kl} + \bar{u}_{kl}$$
$$\lambda I_x^{kl} I_y^{kl} u_{kl} + \left(1 + \lambda {I_y^{kl}}^2\right) v_{kl} = -\lambda I_y^{kl} I_t^{kl} + \bar{v}_{kl}$$

- This is a linear system $\boldsymbol{Ax} = \boldsymbol{b}$, we can solve by Cramer's Rule.

Cramer's Rule

Solving the system

Putting into $\boldsymbol{Ax = b}$

$$\begin{bmatrix} 1+\lambda(I_x^{kl})^2 & \lambda I_x^{kl} I_y^{kl} \\ \lambda I_x^{kl} I_y^{kl} & 1+\lambda(I_y^{kl})^2 \end{bmatrix} \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix} = \begin{bmatrix} -\lambda I_x^{kl} I_t^{kl} + \bar{u}_{kl} \\ -\lambda I_y^{kl} I_t^{kl} + \bar{v}_{kl} \end{bmatrix}$$

$$D = \begin{vmatrix} 1+\lambda(I_x^{kl})^2 & \lambda I_x^{kl} I_y^{kl} \\ \lambda I_x^{kl} I_y^{kl} & 1+\lambda(I_y^{kl})^2 \end{vmatrix} = (1+\lambda(I_x^{kl})^2)(1+\lambda(I_y^{kl})^2) - (\lambda I_x^{kl} I_y^{kl})(\lambda I_x^{kl} I_y^{kl})$$

$$D = 1+\lambda(I_x^{kl})^2 + \lambda(I_y^{kl})^2$$

Solving the system

Putting into $\boldsymbol{Ax = b}$

$$\begin{bmatrix} 1+\lambda(I_x^{kl})^2 & \lambda I_x^{kl} I_y^{kl} \\ \lambda I_x^{kl} I_y^{kl} & 1+\lambda(I_y^{kl})^2 \end{bmatrix} \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix} = \begin{bmatrix} -\lambda\, I_x^{kl}\, I_t^{kl} + \bar{u}_{kl} \\ -\lambda\, I_y^{kl}\, I_t^{kl} + \bar{v}_{kl} \end{bmatrix}$$

$$D_u = \begin{vmatrix} -\lambda\, I_x^{kl}\, I_t^{kl} + \bar{u}_{kl} & \lambda I_x^{kl} I_y^{kl} \\ -\lambda\, I_y^{kl}\, I_t^{kl} + \bar{v}_{kl} & 1+\lambda(I_y^{kl})^2 \end{vmatrix} \qquad D_v = \begin{vmatrix} 1+\lambda(I_x^{kl})^2 & -\lambda\, I_x^{kl}\, I_t^{kl} + \bar{u}_{kl} \\ \lambda I_x^{kl} I_y^{kl} & -\lambda\, I_y^{kl}\, I_t^{kl} + \bar{v}_{kl} \end{vmatrix}$$

$$u = \frac{D_u}{D} \qquad v = \frac{D_v}{D}$$

Solving the system

$$\begin{bmatrix} 1+\lambda(I_x^{kl})^2 & \lambda I_x^{kl} I_y^{kl} \\ \lambda I_x^{kl} I_y^{kl} & 1+\lambda(I_y^{kl})^2 \end{bmatrix} \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix} = \begin{bmatrix} -\lambda\, I_x^{kl}\, I_t^{kl} + \bar{u}_{kl} \\ -\lambda\, I_y^{kl}\, I_t^{kl} + \bar{v}_{kl} \end{bmatrix}$$

$$D_u = \begin{vmatrix} -\lambda\, I_x^{kl}\, I_t^{kl} + \bar{u}_{kl} & \lambda I_x^{kl} I_y^{kl} \\ -\lambda\, I_y^{kl}\, I_t^{kl} + \bar{v}_{kl} & 1+\lambda(I_y^{kl})^2 \end{vmatrix}$$

$$D_u = (-\lambda\, I_x^{kl}\, I_t^{kl} + \bar{u}_{kl})(1+\lambda(I_y^{kl})^2) - (-\lambda\, I_y^{kl}\, I_t^{kl} + \bar{v}_{kl})(\lambda I_x^{kl} I_y^{kl})$$

$$D_u = -\lambda\, I_x^{kl}\, I_t^{kl} + \bar{u}_{kl} - \lambda^2\, I_x^{kl}\, I_t^{kl} I_y^{kl\,2} + \lambda(I_y^{kl})^2\, \bar{u}_{kl} + \lambda\, I_y^{kl}\, I_t^{kl}\, \lambda I_x^{kl} I_y^{kl} - \bar{v}_{kl}\, \lambda I_x^{kl} I_y^{kl}$$

$$D_u = \bar{u}_{kl}(1 + \lambda(I_y^{kl})^2 + \lambda(I_x^{kl})^2) - \lambda(I_x^{kl})^2\, \bar{u}_{kl} - \lambda\, I_x^{kl}\, I_t^{kl} - \lambda^2\, I_x^{kl}\, I_t^{kl} I_y^{kl\,2} - \bar{v}_{kl}\, \lambda I_x^{kl} I_y^{kl} + + \lambda\, I_y^{kl}\, I_t^{kl}\, \lambda I_x^{kl} I_y^{kl}$$

- $D_u = \bar{u}_{kl}(1 + \lambda(I_y^{kl})^2 + \lambda(I_x^{kl})^2) - \lambda(I_x^{kl})^2\,\bar{u}_{kl} - \lambda\, I_x^{kl}\, I_t^{kl} - \lambda^2\, I_x^{kl}\, I_t^{kl} I_y^{kl\,2} - \bar{v}_{kl}\,\lambda I_x^{kl} I_y^{kl} + \lambda\, I_y^{kl}\, I_t^{kl}\,\lambda I_x^{kl} I_y^{kl}$

$$u = \frac{D_u}{D}$$

$$= \frac{\bar{u}_{kl}(1 + \lambda(I_y^{kl})^2 + \lambda(I_x^{kl})^2) - \lambda(I_x^{kl})^2\,\bar{u}_{kl} - \lambda\, I_x^{kl}\, I_t^{kl} - \lambda^2\, I_x^{kl}\, I_t^{kl} I_y^{kl\,2} - \bar{v}_{kl}\,\lambda I_x^{kl} I_y^{kl} + \lambda I_y^{kl}\, I_t^{kl}\,\lambda I_x^{kl} I_y^{kl}}{1 + \lambda(I_x^{kl})^2 + \lambda(I_y^{kl})^2}$$

$$u = \bar{u}_{kl} - \frac{\lambda I_x^{kl}\bar{u}_{kl} + \bar{v}_{kl}\,\lambda I_y^{kl} + \lambda I_t^{kl}}{1 + \lambda(I_x^{kl})^2 + \lambda(I_y^{kl})^2} I_x^{kl}$$

$$u = \bar{u}_{kl} - \frac{P}{D} I_x^{kl}$$

$$P = I_x^{kl}\bar{u}_{kl} + \bar{v}_{kl}\, I_y^{kl} + I_t^{kl}$$

$$D = 1/\lambda + (I_x^{kl})^2 + (I_y^{kl})^2$$

Discrete Optical Flow Update Rule

- Update Rule:

  - $u_{kl}^{n+1} = \bar{u}_{kl}^{n} - \frac{I_x^{kl}\bar{u}_{kl}^{n} + I_y^{kl}\bar{v}_{kl}^{n} + I_t^{kl}}{\frac{1}{\lambda} + \left[\left(I_x^{kl}\right)^2 + \left(I_y^{kl}\right)^2\right]} I_x^{kl}$

  - $v_{kl}^{n+1} = \bar{v}_{kl}^{n} - \frac{I_x^{kl}\bar{u}_{kl}^{n} + I_y^{kl}\bar{v}_{kl}^{n} + I_t^{kl}}{\frac{1}{\lambda} + \left[\left(I_x^{kl}\right)^2 + \left(I_y^{kl}\right)^2\right]} I_y^{kl}$

Horn and Schunck Algorithm

- Compute image gradient I_x & I_y
- Compute the temporal gradient I_t
- Set initial flow $u = 0, v = 0$

- Do
  - Compute $\bar{u}$ and $\bar{v}$
  - Compute flow field for each pixel

$$u_{kl}^{n+1} = \bar{u}_{kl}^{n} - \frac{I_x^{kl}\bar{u}_{kl}^{n} + I_y^{kl}\bar{v}_{kl}^{n} + I_t^{kl}}{\frac{1}{\lambda} + \left[\left(I_x^{kl}\right)^2 + \left(I_y^{kl}\right)^2\right]} I_x^{kl}$$

$$v_{kl}^{n+1} = \bar{v}_{kl}^{n} - \frac{I_x^{kl}\bar{u}_{kl}^{n} + I_y^{kl}\bar{v}_{kl}^{n} + I_t^{kl}}{\frac{1}{\lambda} + \left[\left(I_x^{kl}\right)^2 + \left(I_y^{kl}\right)^2\right]} I_y^{kl}$$

Until converge

Example

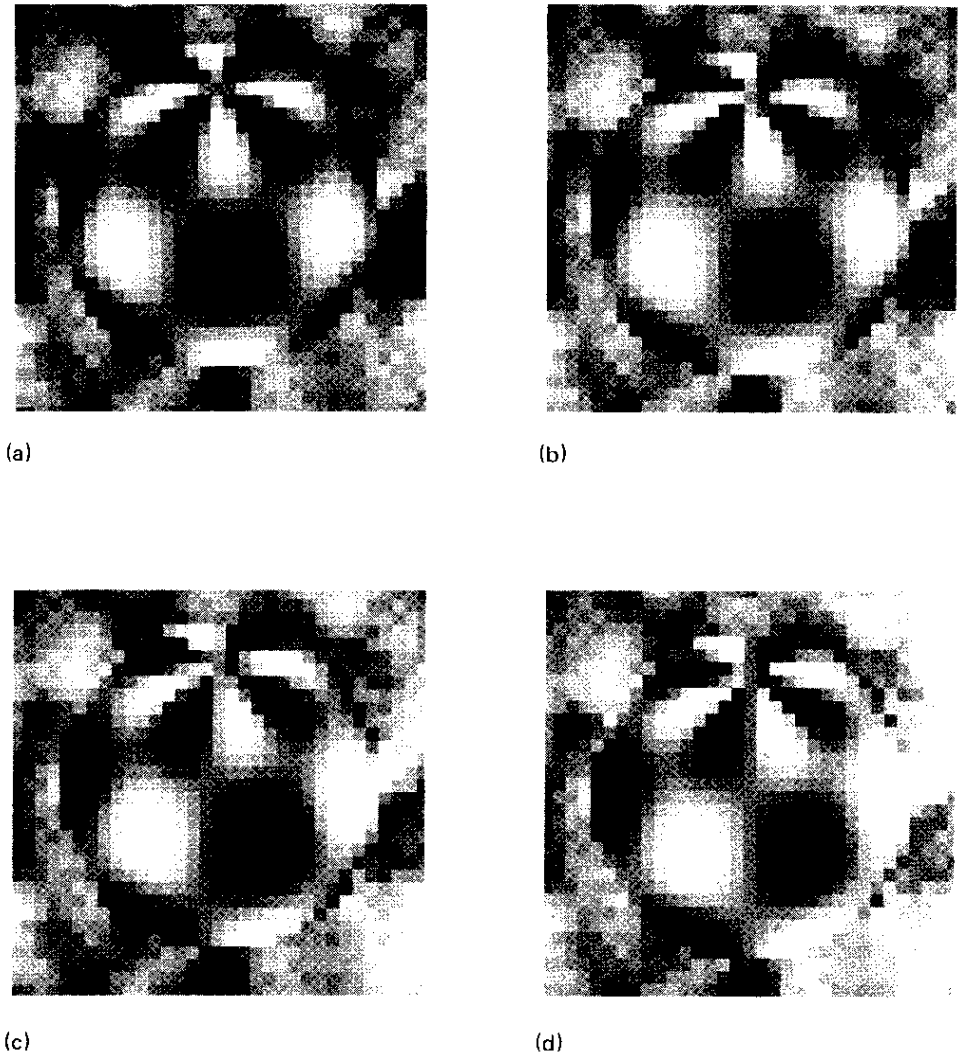


Figure 12-8. Four frames of a synthetic image sequence showing a sphere slowly rotating in front of a randomly patterned background.

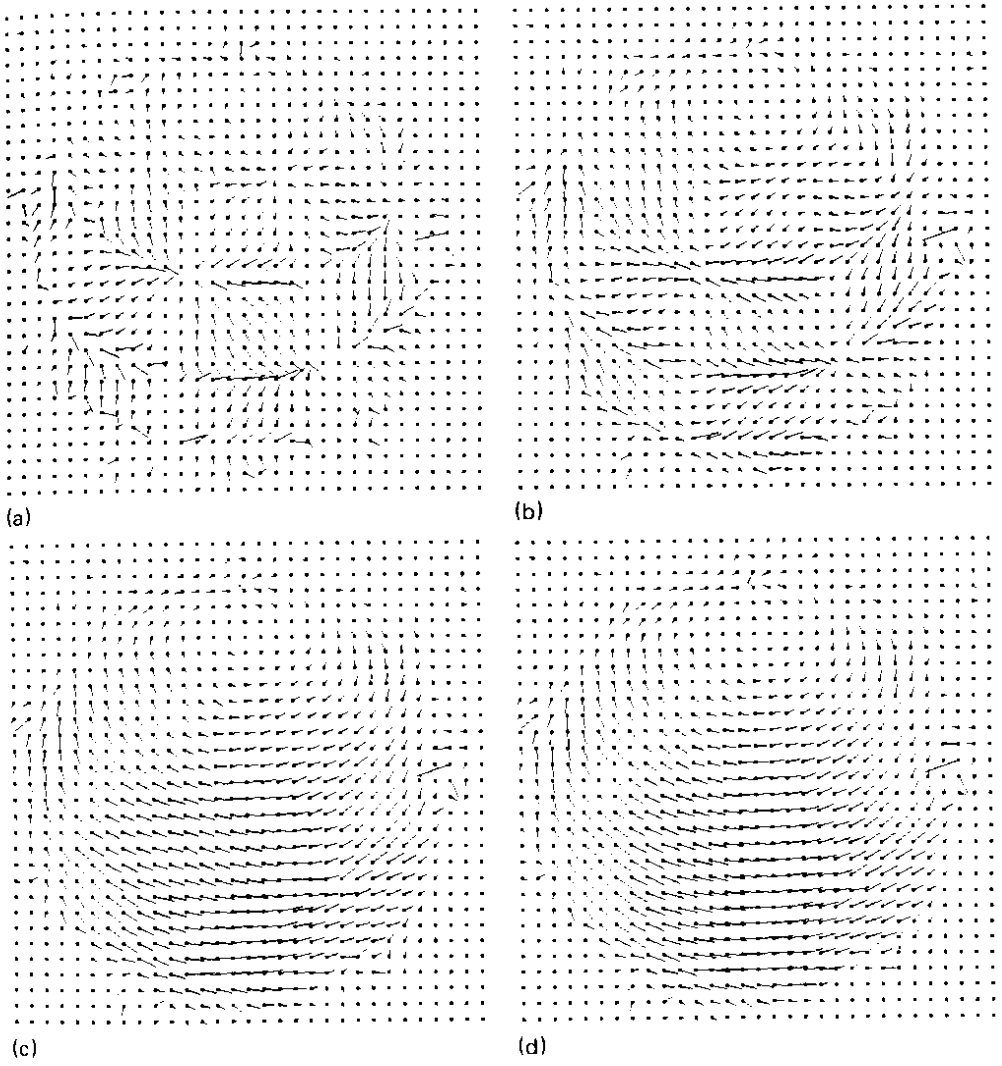
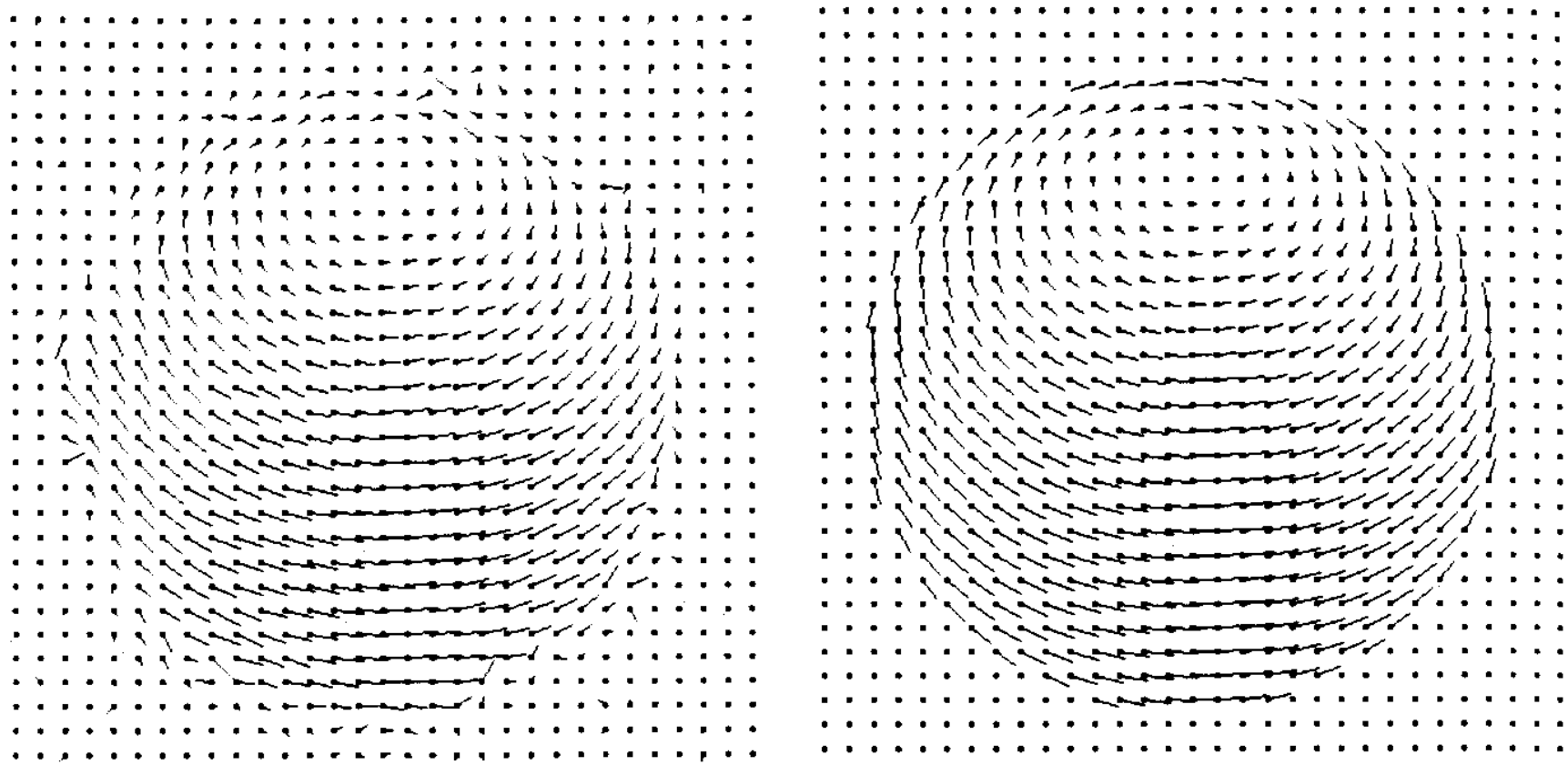


Figure 12-9. Estimates of the optical flow shown in the form of needle diagrams after 1, 4, 16, and 64 iterations of the algorithm.

Results



(a) (b)

Figure 12-10. (a) The estimated optical flow after several more iterations. (b) The computed motion field.

Another Example

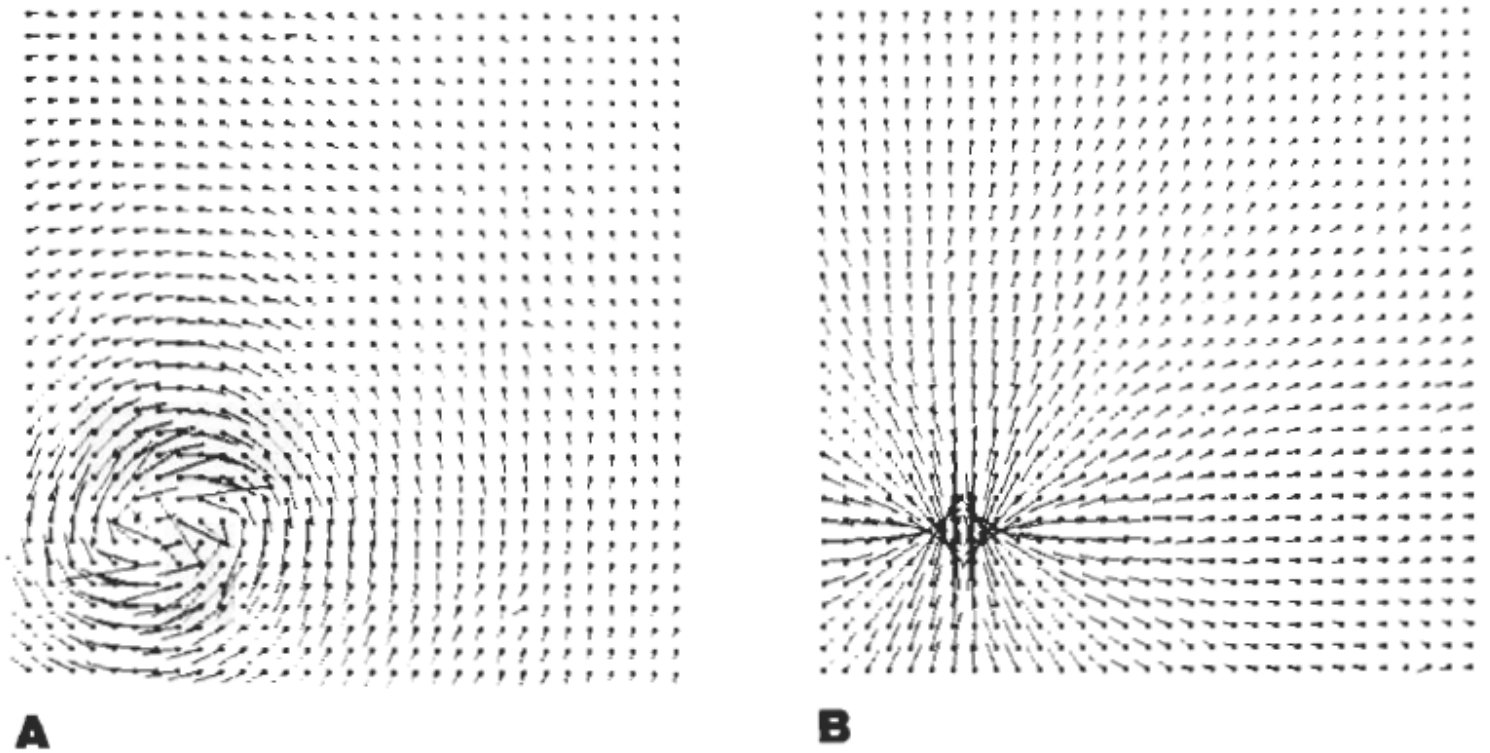


FIG. 9. Flow patterns computed for flow around a line vortex and two dimensional flow into a sink. In each case the estimates after 32 iterations are shown.

Horn & Schunck - Matlab Example

- Compute Derivatives

```
fx = conv2(im1,[-1 1; -1 1], 'valid'); % partial on x
fy = conv2(im1, [-1 -1; 1 1], 'valid'); % partial on y
ft = conv2(im1, ones(2), 'valid') + conv2(im2, -ones(2), 'valid'); % partial on t
```

- Laplacian Mask and averaging

```
kernel_1=[0 1/4 0;1/4 0 1/4;0 1/4 0];

% or

kernel_1=[1/12 1/6 1/12;1/6 0 1/6;1/12 1/6 1/12];
```

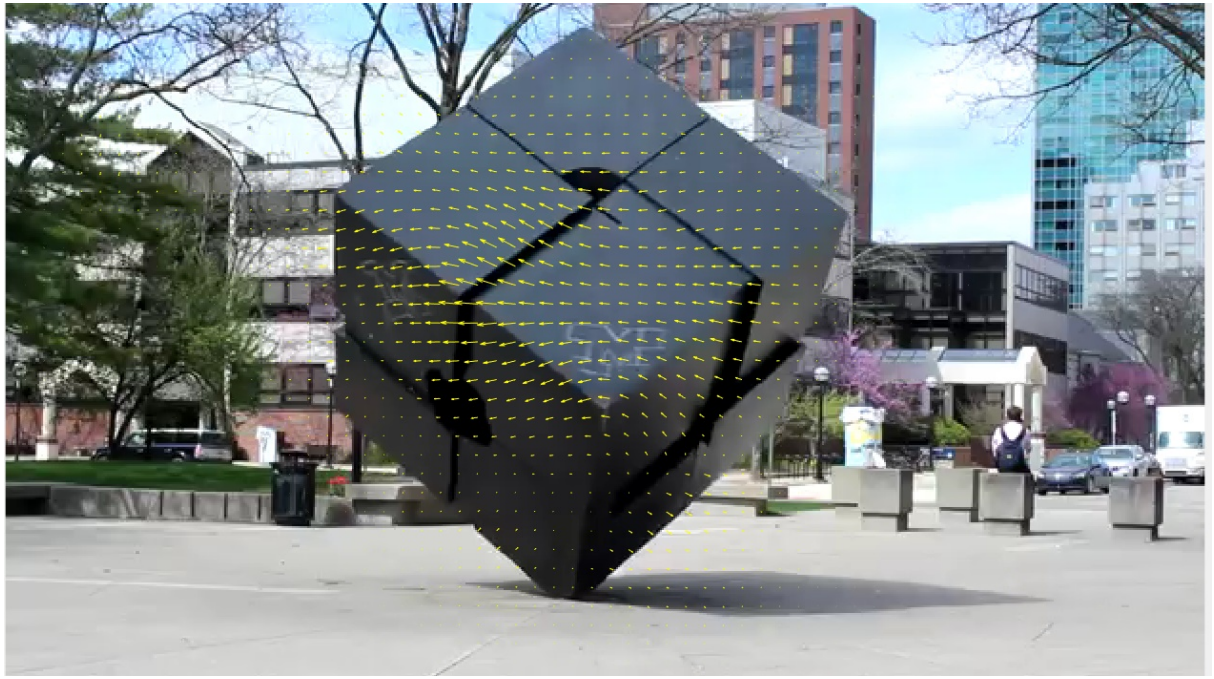
```
uAvg=conv2(u,kernel_1,'same');
vAvg=conv2(v,kernel_1,'same');
```

- Iteration

```
alpha = (fx.*uAvg+fy.*vAvg+ft)./(1+lamda*(fx.^2+fy.^2));
u = uAvg-alpha.*fx;
v = vAvg-alpha.*fy;
```

https://www.mathworks.com/matlabcentral/fileexchange/22756-horn-schunck-optical-flow-method

Result



Lucas Kanade

- Proposed by Bruce D. Lucas, and Takeo Kanade in 1981
  - Original proposed to solve stereo vision registration issue.
  - Make use of spatial intensity gradient of the images for matching

From *Proceedings of Imaging Understanding Workshop*, pp. 121-130 (1981).

An Iterative Image Registration Technique with an Application to Stereo Vision

Bruce D. Lucas
Takeo Kanade

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Image registration finds a variety of applications in computer vision. Unfortunately, traditional image registration techniques tend to be costly. We present a new image registration technique that makes use of the spatial intensity gradient of the images to find a good match using a type of Newton-Raphson iteration. Our technique is faster because it examines far fewer potential matches between the images than existing techniques. Furthermore, this registration technique can be generalized to handle rotation, scaling and shearing. We show show our technique can be adapted for use in a stereo vision system.

1. Introduction

Image registration finds a variety of applications in computer vision, such as image matching for stereo vision, pattern recognition, and motion analysis. Untortunately, existing techniques for image registration tend to be costly. Moreover, they generally fail to deal with rotation or other distortions of the images.

In this paper we present a new image registration technique that uses spatial intensity gradient information to direct the search for the position that yields the best match. By taking more information about the images into account, this technique is able to find the best match between two images with far fewer comparisons of images than techniques which examine the possible positions of registration in some fixed order. Our technique takes advantage of the fact that in many applications the two images are already in approximate registration. This technique can be generalized to deal with arbitrary linear distortions of the image, including rotation. We then describe a stereo vision system that uses this registration technique, and suggest some further avenues for research toward making effective use of this method in stereo image understanding.

2. The registration problem

The translational image registration problem can be characterized as follows: We are given functions $F(x)$ and $G(x)$ which give the respective pixel values at each location x in two images, where x is a vector. We wish to find the disparity vector h which minimizes some measure of the difference between $F(x + h)$ and $G(x)$, for x in some region of interest R. (See figure 1).

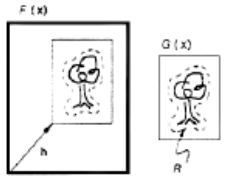


Figure 1: The image registration problem

Typical measures of the difference between $F(x + h)$ and $G(x)$ are:

- L_1 norm = $\sum_{x \in R} |F(x+h) - G(x)|$
- L_2 norm = $\left(\sum_{x \in R} [F(x+h) - G(x)]^2\right)^{1/2}$
- negative of normalized correlation

$$= \frac{-\sum_{x \in R} F(x+h)G(x)}{\left(\sum_{x \in R} F(x+h)^2\right)^{1/2}\left(\sum_{x \in R} G(x)^2\right)^{1/2}}$$

We will propose a more general measure of image difference, of which both the L_2 norm and the correlation are special cases. The L_1 norm is chiefly of interest as an inexpensive approximation to the L_2 norm.

Lucas and Kanade flow

- Energy function

$$E(u,v) = \sum \left(I_x u + I_y v + I_t\right)^2$$

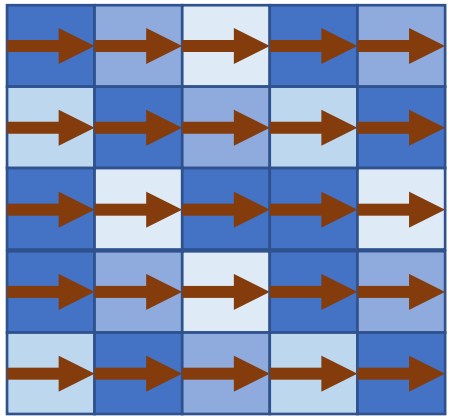
- We want to minimize this
  - $\frac{\partial E}{\partial u} = \sum 2I_x\left(I_x u + I_y v + I_t\right) = 0$
  - $\frac{\partial E}{\partial v} = \sum 2I_y\left(I_x u + I_y v + I_t\right) = 0$
- One equation and two unknowns. Aperture problem
- How to solve?

Lucas and Kanade flow

- Energy function

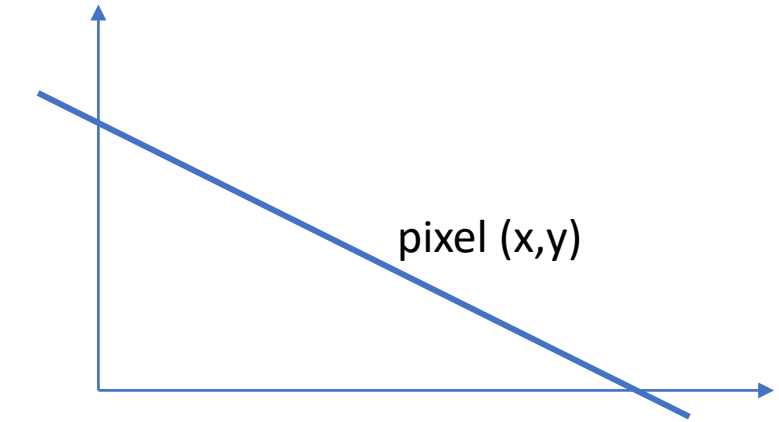
$$E(u,v) = \sum \left(I_x u + I_y v + I_t\right)^2$$

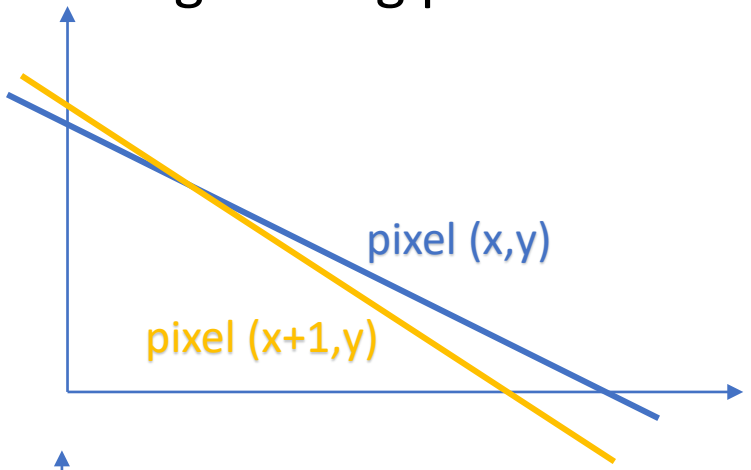
- We want to minimize this
  - $\frac{\partial E}{\partial u} = \sum 2I_x\left(I_x u + I_y v + I_t\right) = 0$
  - $\frac{\partial E}{\partial v} = \sum 2I_y\left(I_x u + I_y v + I_t\right) = 0$
- Assume the flow field is ***smooth locally***
- Pixel neighbors have the *same (u,v)*

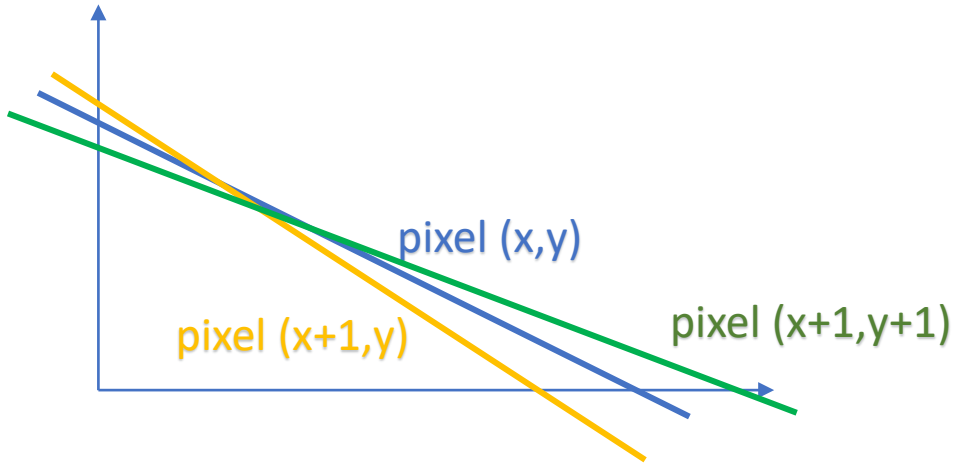


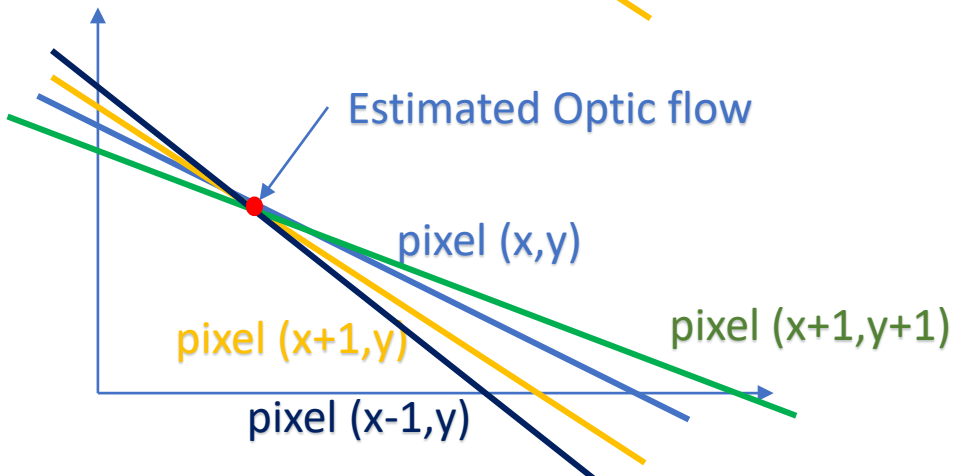
Lucas Kanade Algorithm

- Optic flow equation is specified $I_x u + I_y v + I_t = 0$ for all pixel in the patch.
- Consider the straight lines defining the u,v of the neighboring pixels.





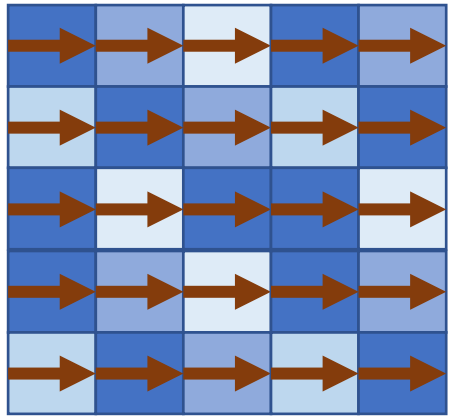




Lucas and Kanade flow (Least Square)

- Assume the flow field is smooth locally
- Pixel neighbors have the same (u,v)
- We use 5x5 windows (25 equations)

$$
\begin{aligned}
I_{x1}u + I_{y1}v &= -I_{t1} \\
I_{x2}u + I_{y2}v &= -I_{t2} \\
&\vdots \\
I_{x25}u + I_{y25}v &= -I_{t25}
\end{aligned}
$$



$$
\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y1} \\ \vdots & \vdots \\ I_{x25} & I_{y25} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{t25} \end{bmatrix}
$$

25x2 2x1 25x1

RGB version

- Assume the flow field is smooth locally
- Pixel neighbors have the same (u,v)
- We use 5x5 windows (75 equations)

$$\begin{bmatrix} I_{x1}[0] & I_{y1}[0] \\ I_{x1}[1] & I_{y1}[1] \\ I_{x1}[2] & I_{y1}[2] \\ \vdots & \vdots \\ I_{x25}[0] & I_{y25}[0] \\ I_{x25}[1] & I_{y25}[1] \\ I_{x25}[2] & I_{y25}[2] \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1}[0] \\ -I_{t1}[1] \\ -I_{t1}[2] \\ \vdots \\ -I_{t25}[0] \\ -I_{t25}[1] \\ -I_{t25}[2] \end{bmatrix}$$

75x2 2x1 75x1

- Question:
- RGB has three channels, therefore three equations and *two unknowns*. Can we solve the for U and V?
- No! Note that RGB are correlated and cannot be used for solve the U and V alone.
- It just provide better gradient

Solving the ambiguity

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y1} \\ \vdots & \vdots \\ I_{x25} & I_{y25} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{t25} \end{bmatrix} \Rightarrow \text{Minimize } \|\mathrm{A}\mathbf{u} - \mathbf{f_t}\|^2$$

- This is an over-constraint linear system, we use least square solution to find u,v to minimize the error
- $$\begin{bmatrix} I_{x1} & \dots & I_{x25} \\ I_{y1} & \dots & I_{x25} \end{bmatrix} \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y1} \\ \vdots & \vdots \\ I_{x25} & I_{y25} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_{x1} & \dots & I_{x25} \\ I_{y1} & \dots & I_{x25} \end{bmatrix} \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{t25} \end{bmatrix}$$

$$\mathbf{Au} = \mathbf{f_t}$$
$$\boldsymbol{A^T}\mathbf{Au} = \boldsymbol{A^T}\mathbf{f_t}$$
$$\mathbf{u} = (\boldsymbol{A^T}\mathbf{A})^{-1}\boldsymbol{A^T}\mathbf{f_t}$$

- $$\begin{bmatrix} \sum I_x I_x & \sum I_x I f_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Similar to Second moment matrix

Source: M. Shah

Lucas Kanade Matlab Example

- Compute Derivatives

```
fx = conv2(im1,[-1 1; -1 1], 'valid'); % partial on x
fy = conv2(im1, [-1 -1; 1 1], 'valid'); % partial on y
ft = conv2(im1, ones(2), 'valid') + conv2(im2, -ones(2), 'valid'); % partial on t
```

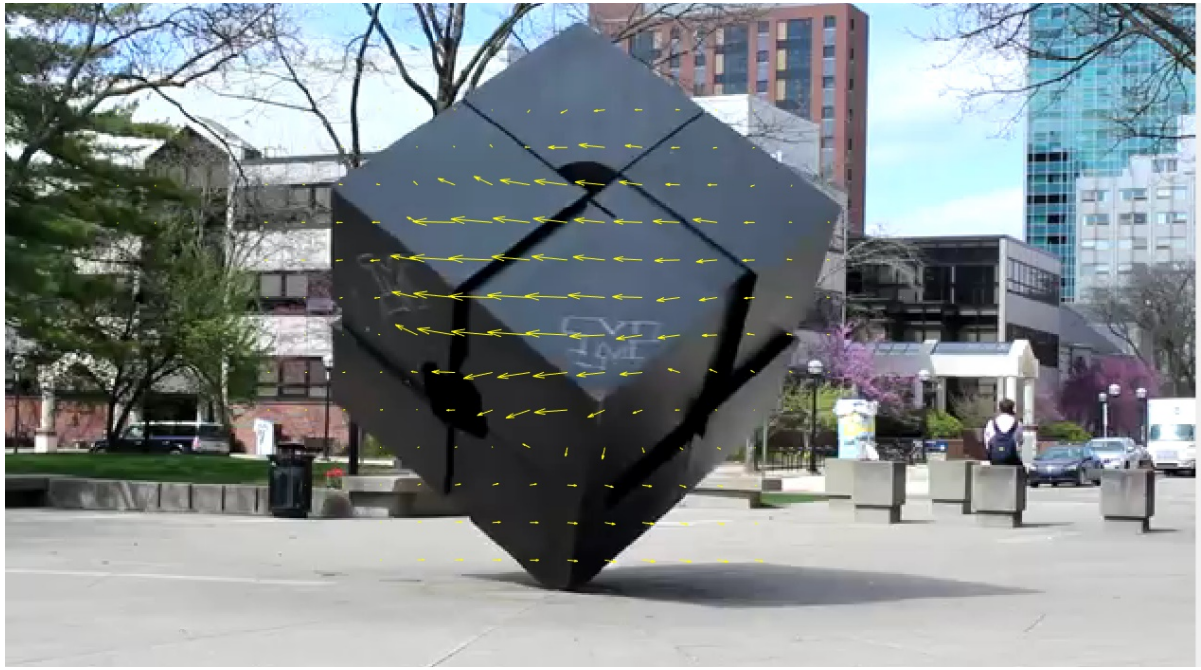
- Define the local windows

```
for i = w+1:size(Ix_m,1)-w
   for j = w+1:size(Ix_m,2)-w
      % Copy Ix, Iy, It to the windows
      Ix = Ix_m(i-w:i+w, j-w:j+w);
      Iy = Iy_m(i-w:i+w, j-w:j+w);
      It = It_m(i-w:i+w, j-w:j+w);

      Ix = Ix(:);
      Iy = Iy(:);
      b = -It(:); % This defines b

      A = [Ix Iy]; % This defines a
      vel = pinv(A)*b; % vel = pesudo inverse(a)*b

      u(i,j)=Vel(1);
      v(i,j)=Vel(2);
   end;
end;
```



https://www.mathworks.com/matlabcentral/fileexchange/48744-lucas-kanade-tutorial-example-1

Solving the ambiguity

- All neighbors satisfy
- $\begin{bmatrix} \sum I_x I_x & \sum I_x I f_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$
- Let $A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$
- Then the u,v are given below:
  - $\begin{bmatrix} u \\ v \end{bmatrix} = -(A^T A)^{-1} \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$

Source: Silvio Savarese

Conditions for solvability

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

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I f_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \qquad A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$$

- When is this solvable?
  - $A^T A$ is invertible
  - Eigenvalues λ_1 and λ_2 doesn't equal to zero
  - λ_1 / λ_2 are not too large

Lucas & Kanade flow

- $A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$ is second moment matrix

- Does it remind you something?

- **Harris corner detector**

Recall and comparison with Harris Corner Detector

- $E(u,v) = \sum \overbrace{w(x,y)}^{window\ function} \left[\overbrace{I(x+u,y+v)}^{shifted\ intensity} - \overbrace{I(x,y)}^{intensity} \right]^2$

Harris: shift in single frame
OF: Displacement across two frames

Current frame

previous frame

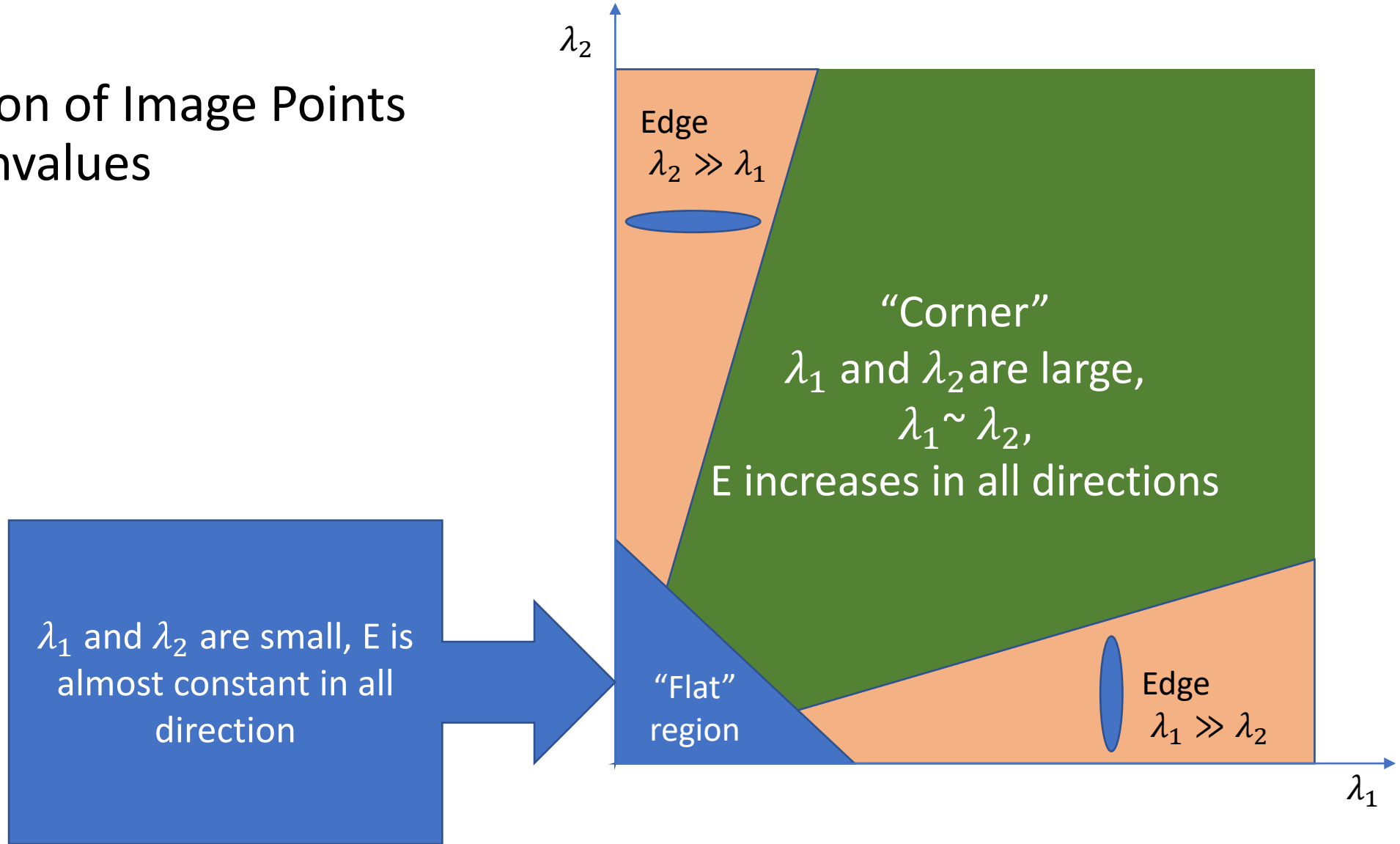
- $\mathrm{A}^{\mathrm{T}}\mathrm{A} = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$ is second moment matrix

- Eigenvectors and eigenvalues of $A^T A$ relate to edge direction and magnitude

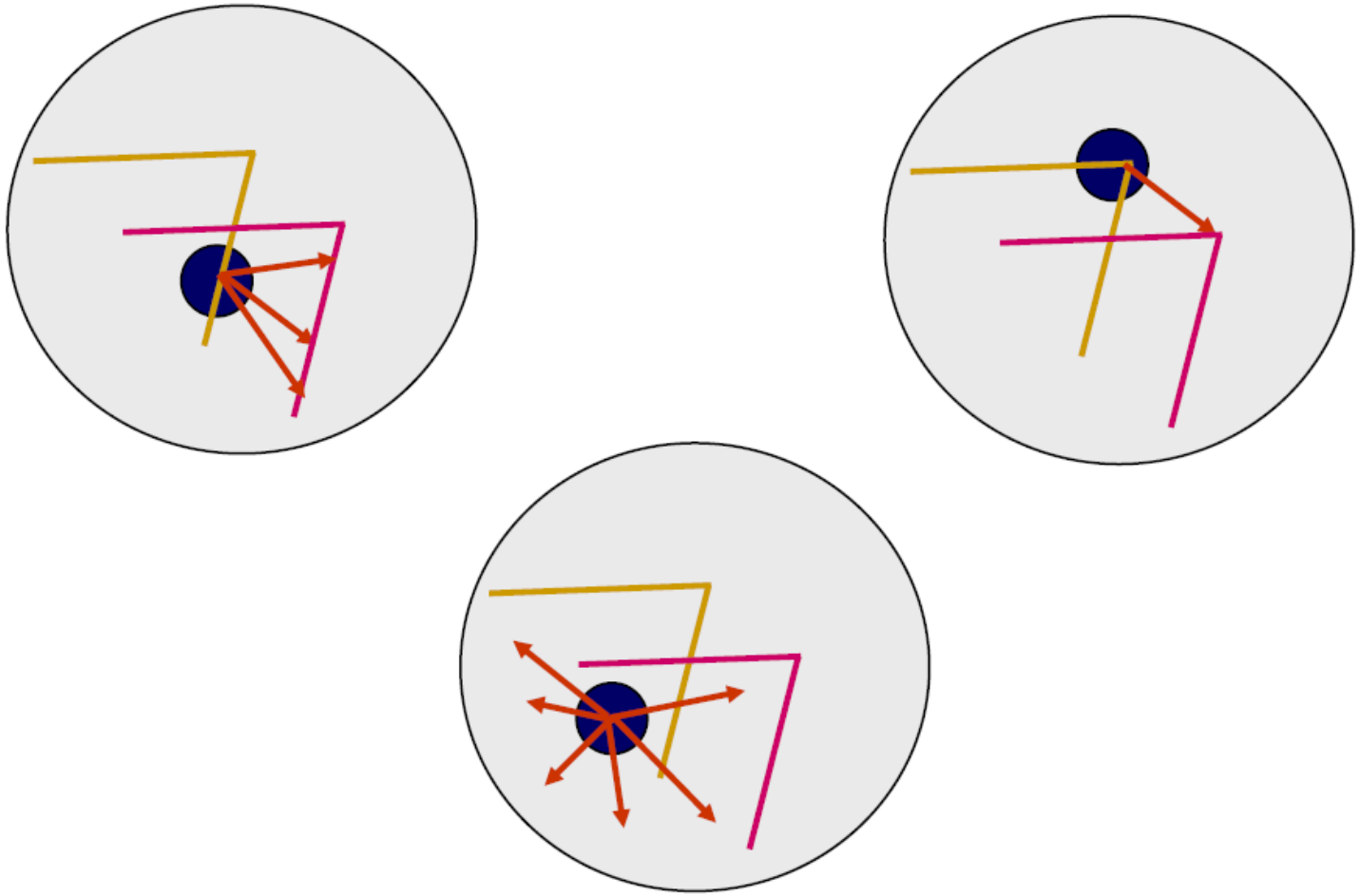
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it.

Classification of Image Points

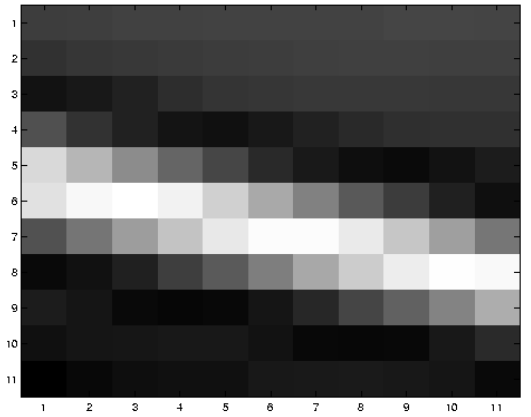
- Classification of Image Points using eigenvalues

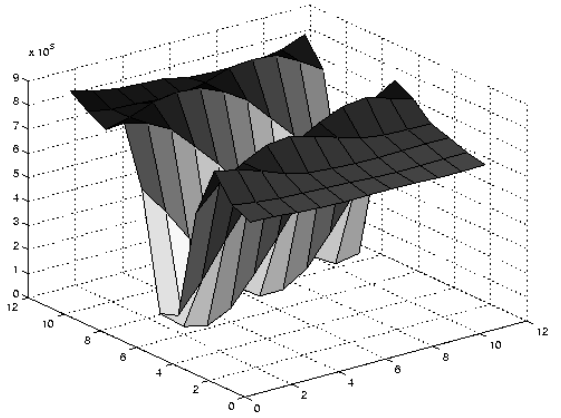


Local Patch Analysis



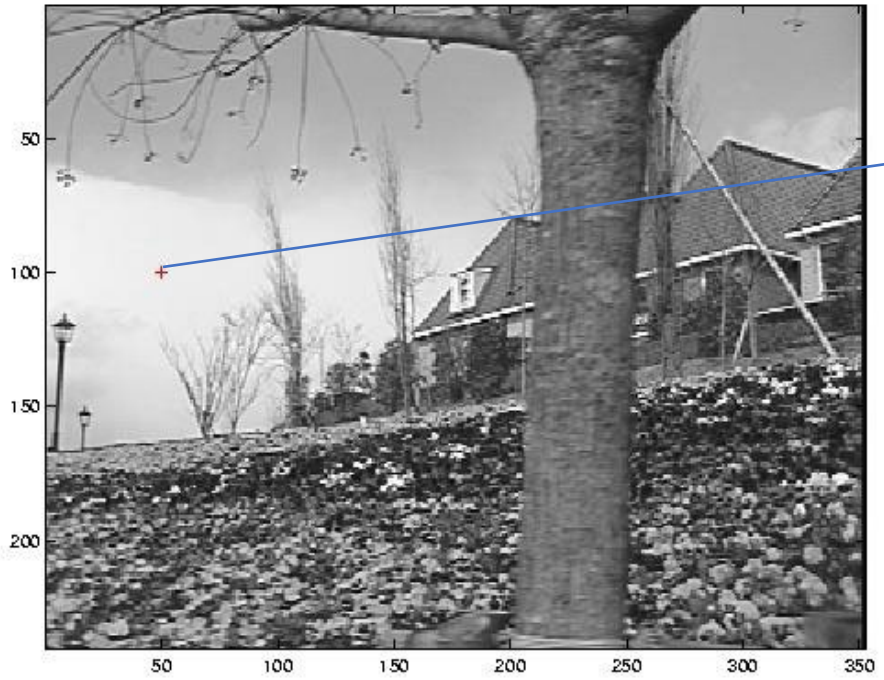
Recall - Edge

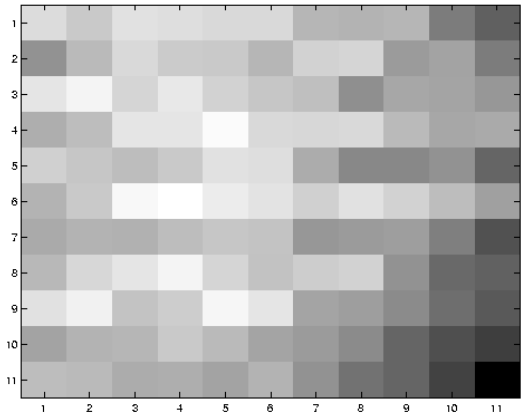


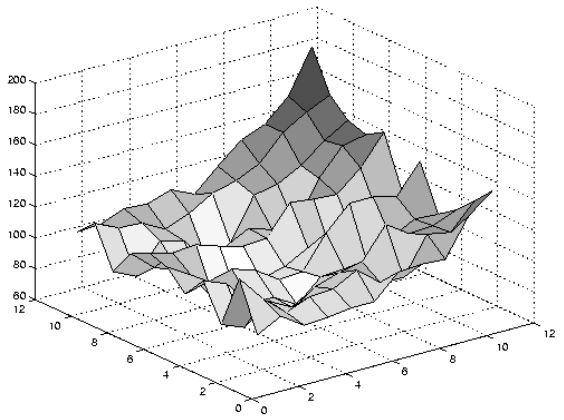


Large I_x and small I_y

Recall - low texture

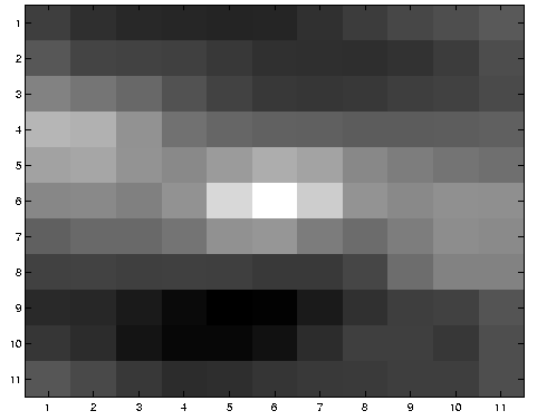


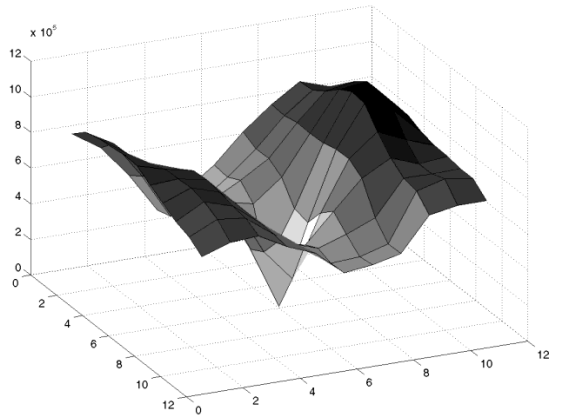




I_x, I_y are small

Recall – High texture region





I_x, I_y are large

Implications

- Corners are when λ_1, λ_2 are *large*, Lucas-Kanade works best at *corners*
- Corners has two different directions of gradient
- Aperture problem doesn't exist at corners.
- Corners are good place for computing optical flow.

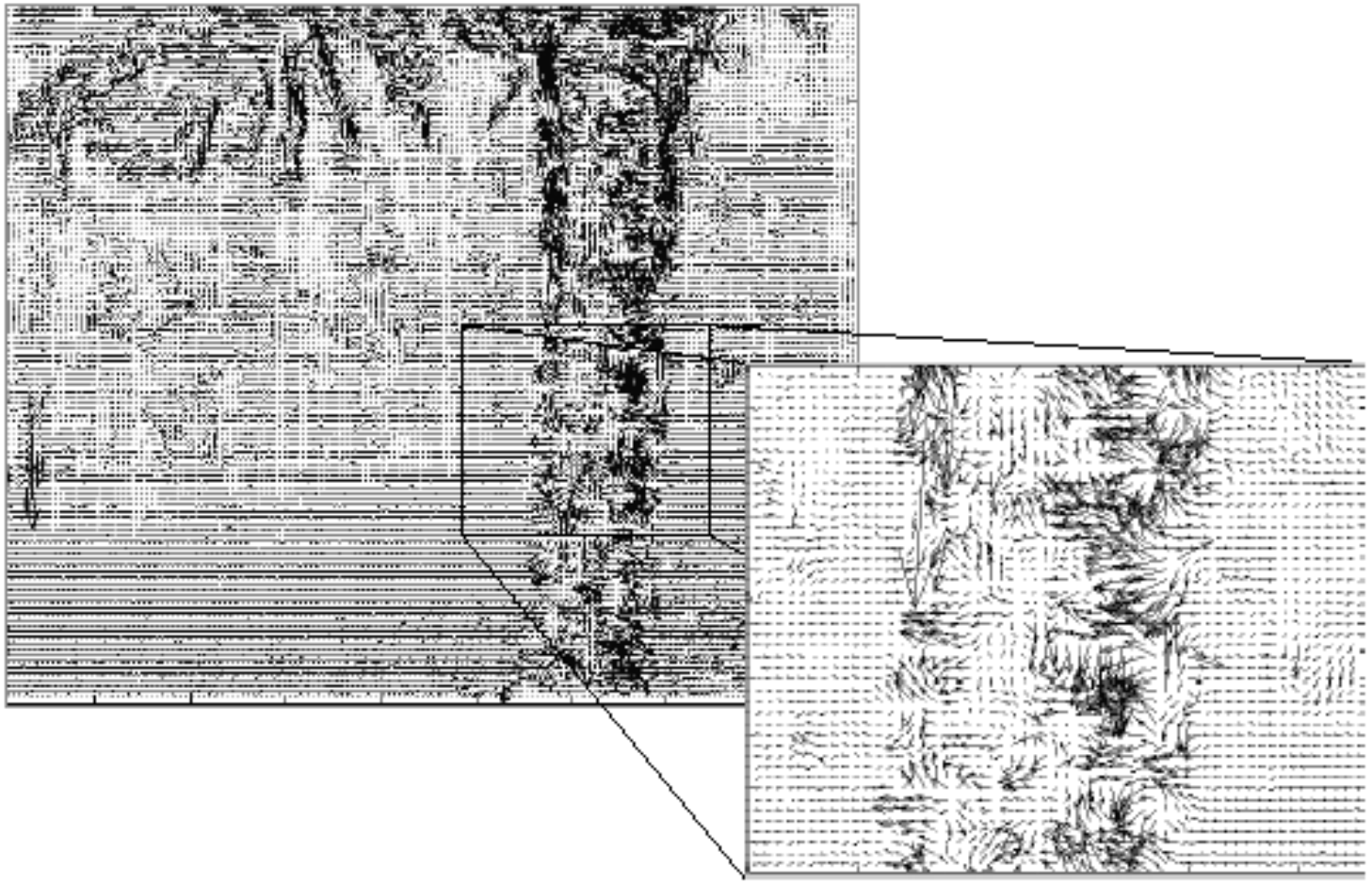
Key Assumptions for Lucas and Kanade Optical flow

- Assumptions
  - Small motion: small displacement
  - Brightness constancy: point has the same brightness in every frame
  - Spatial coherence: neighbors move the same u, v
- Errors in Lucas-kanade
  - *Motion is large* – either object moves fast or brightness changes rapidly, derivative masks fail to estimate spatiotemporal derivatives
  - *Neighbor has different u, v* boundary of objects

Small motion?

- Is this motion small enough?
  - Probably not—it's much larger than one How might we solve this problem?

Optical Flow Result



Small motion

- If the motion is not small? > 1 pixel?
- How to solve this problem?

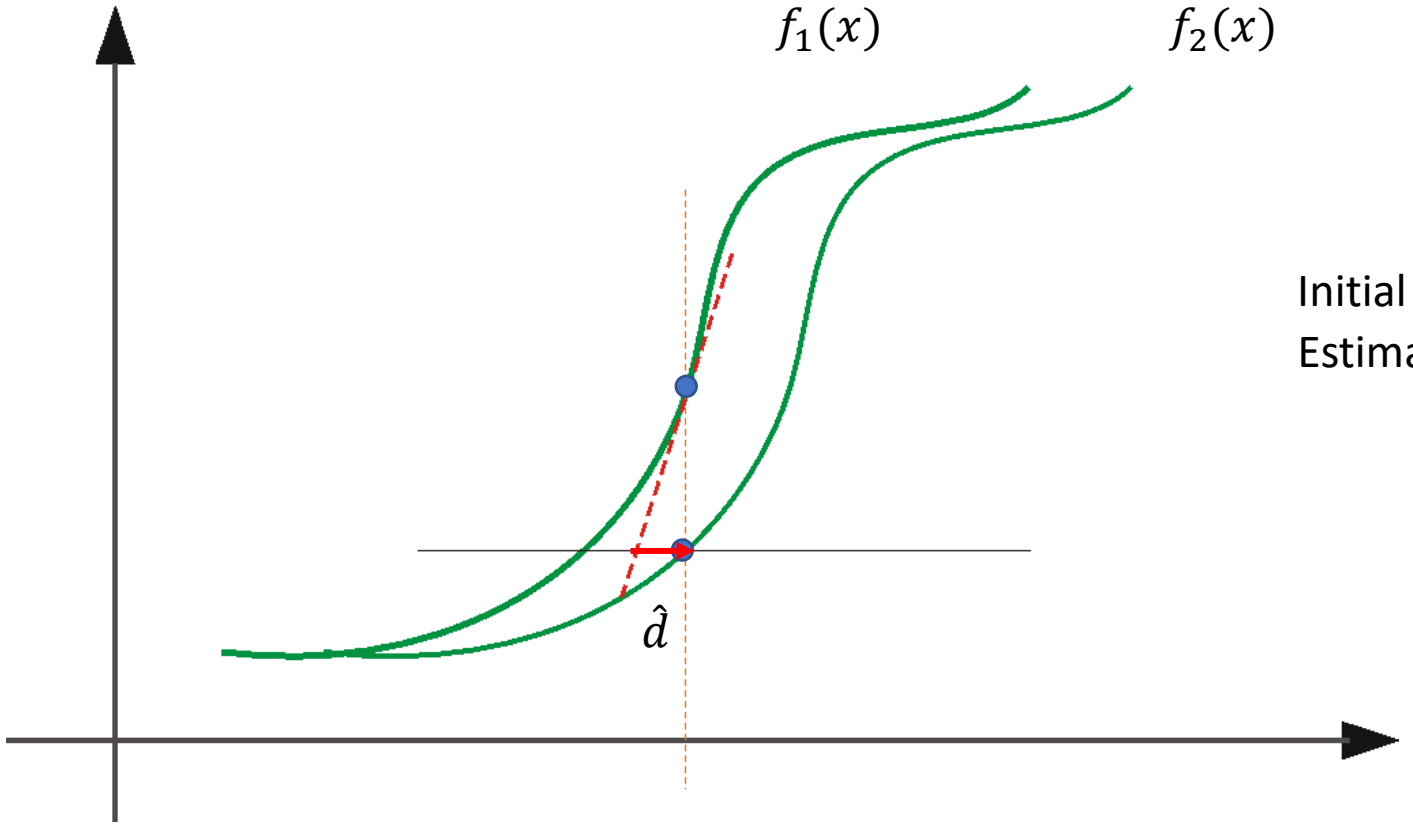
- Reduce the resolution

- Method?
  - Iterative Refinement method
  - Pyramid – Course-to-fine flow estimation

Method 1: Iterative Refinement

- Iterative Lucas Kanade Algorithm
  - Estimate velocity at each pixel using one iteration of Lucus Kanade estimation
  - Warp one image toward the other using the estimated flow field
  - Repeat the above process to refine the estimation.

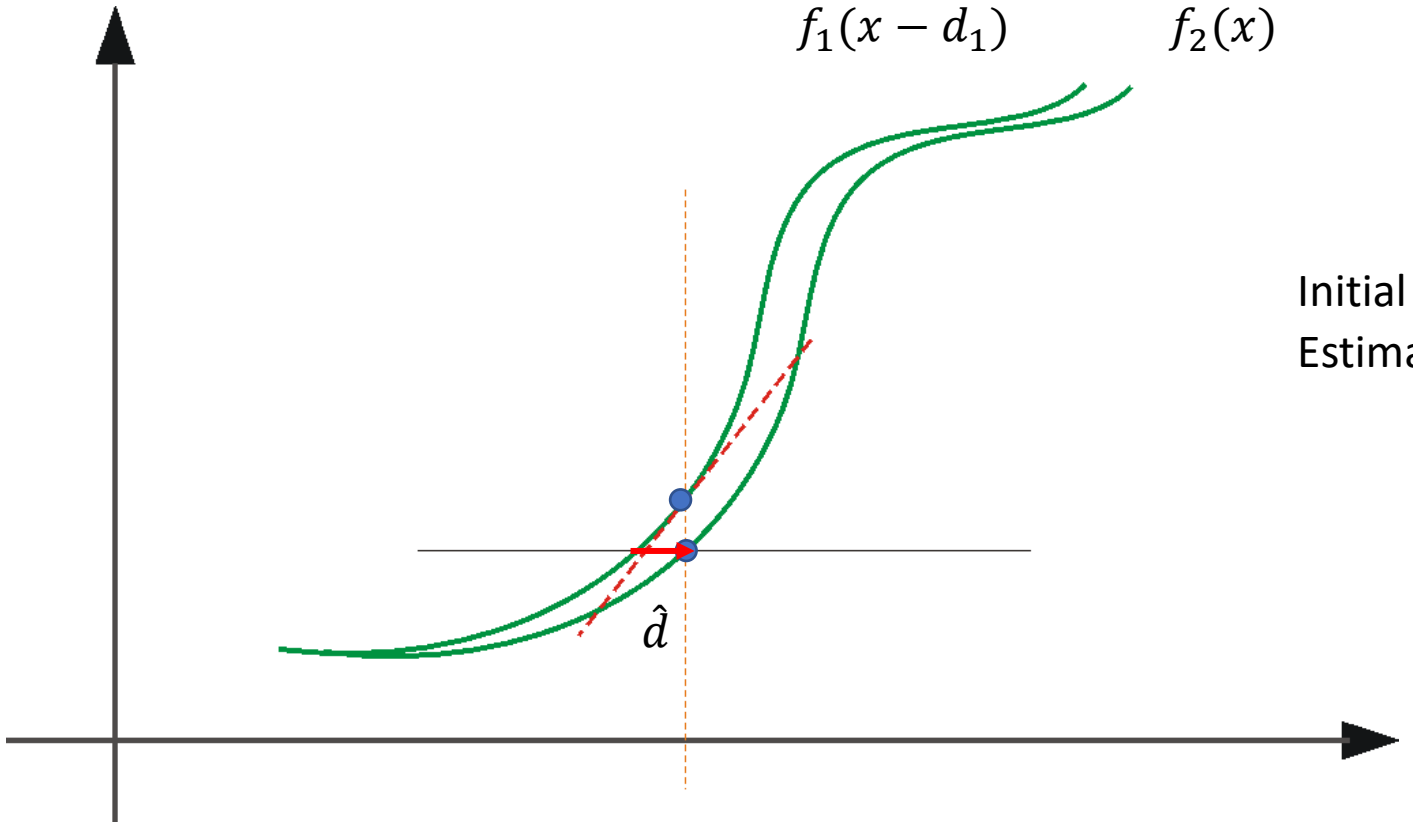
Optical Flow: Iterative Refinement



Initial guess: $d_0 = 0$

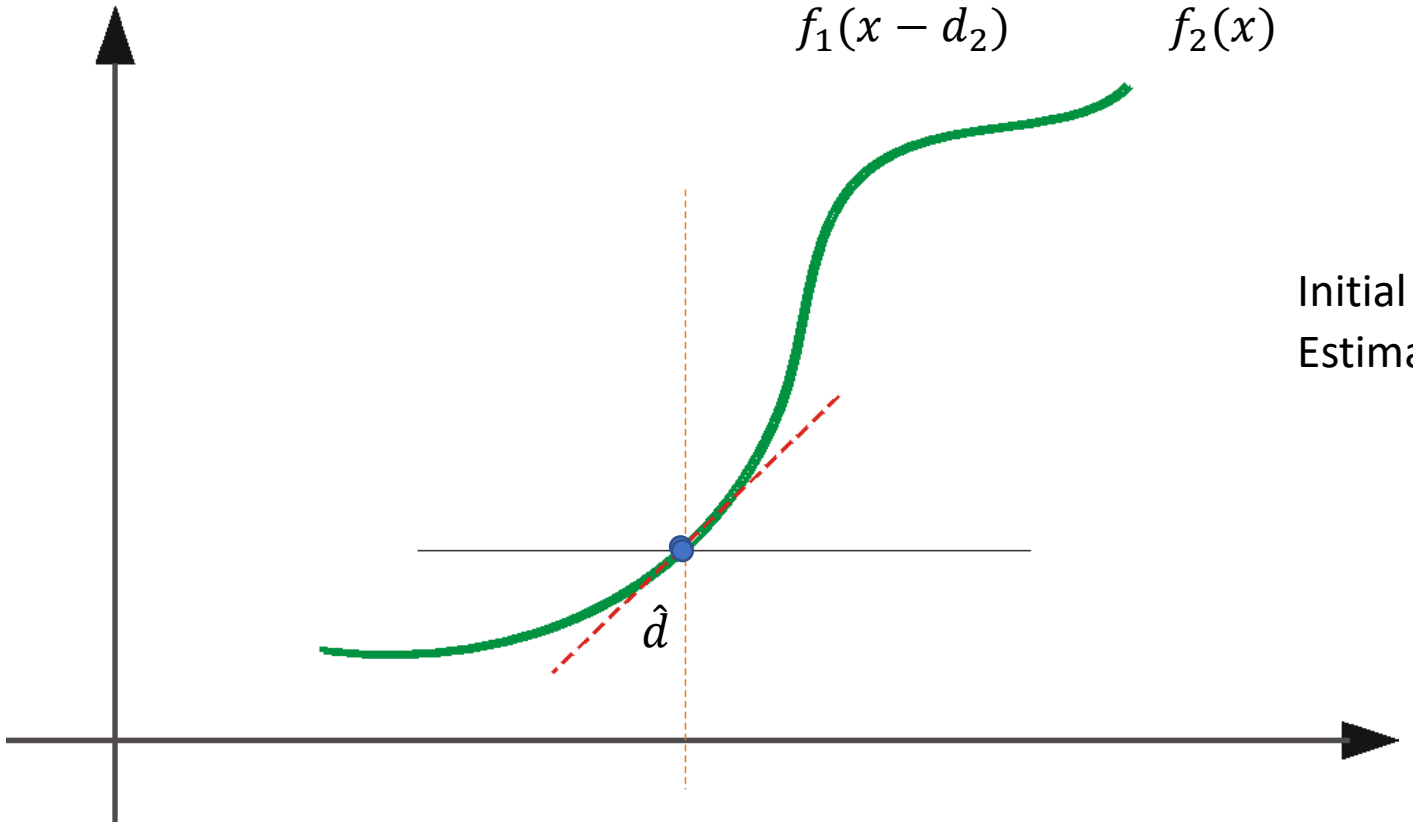
Estimate: $d_1 = d_0 + \hat{d}$

Optical Flow: Iterative Refinement



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

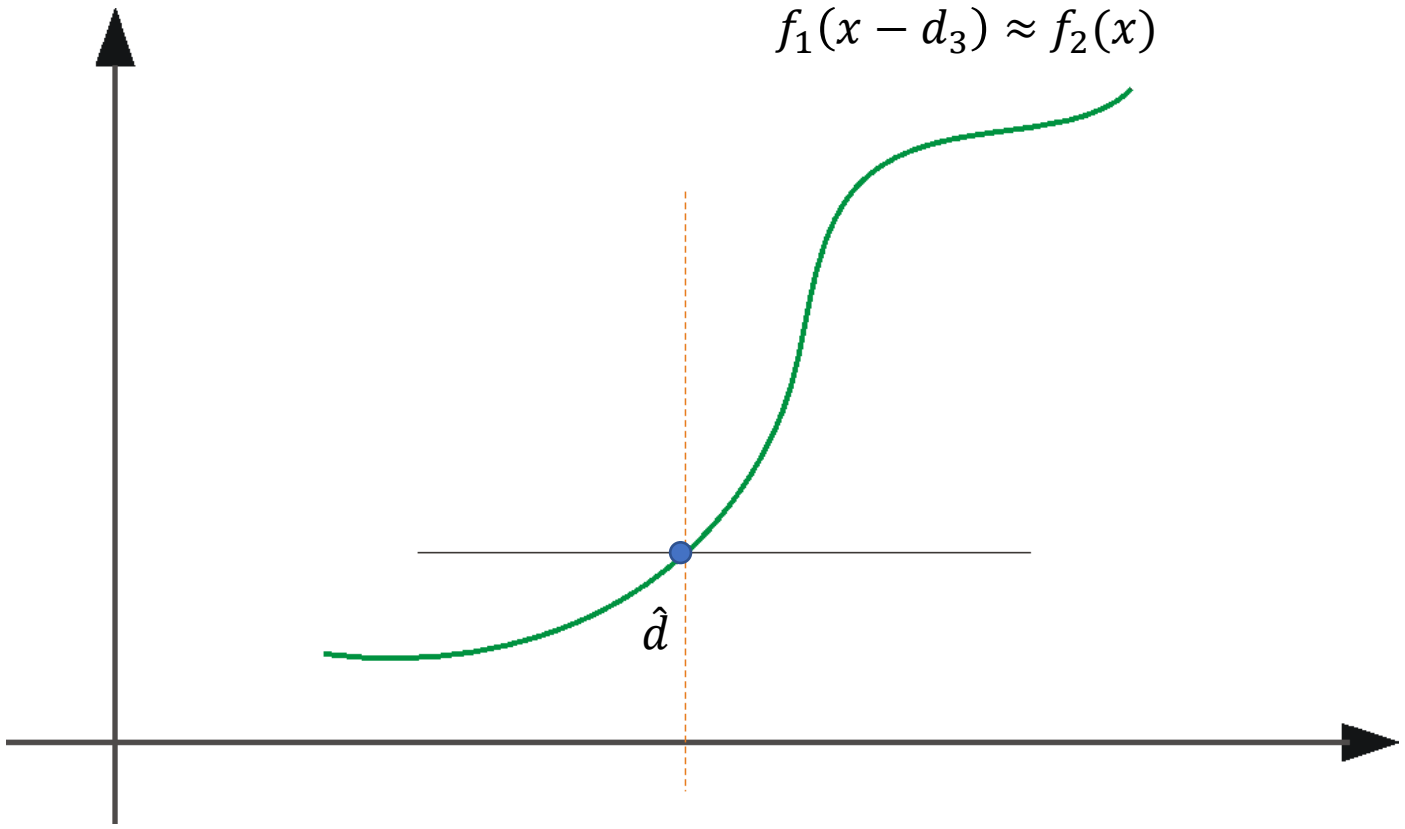
Optical Flow: Iterative Refinement



Initial guess: d_2

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

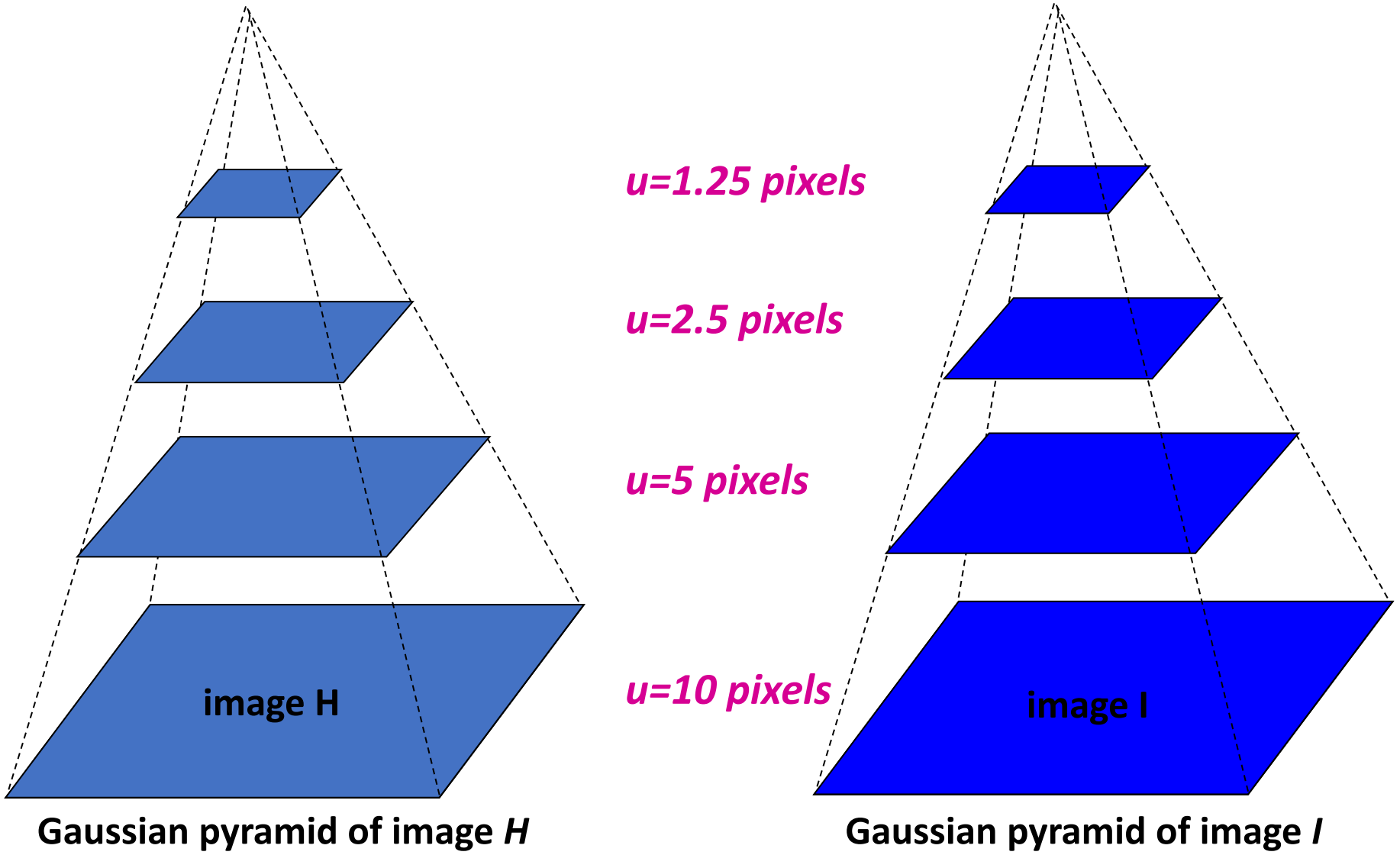
Optical Flow: Iterative Refinement



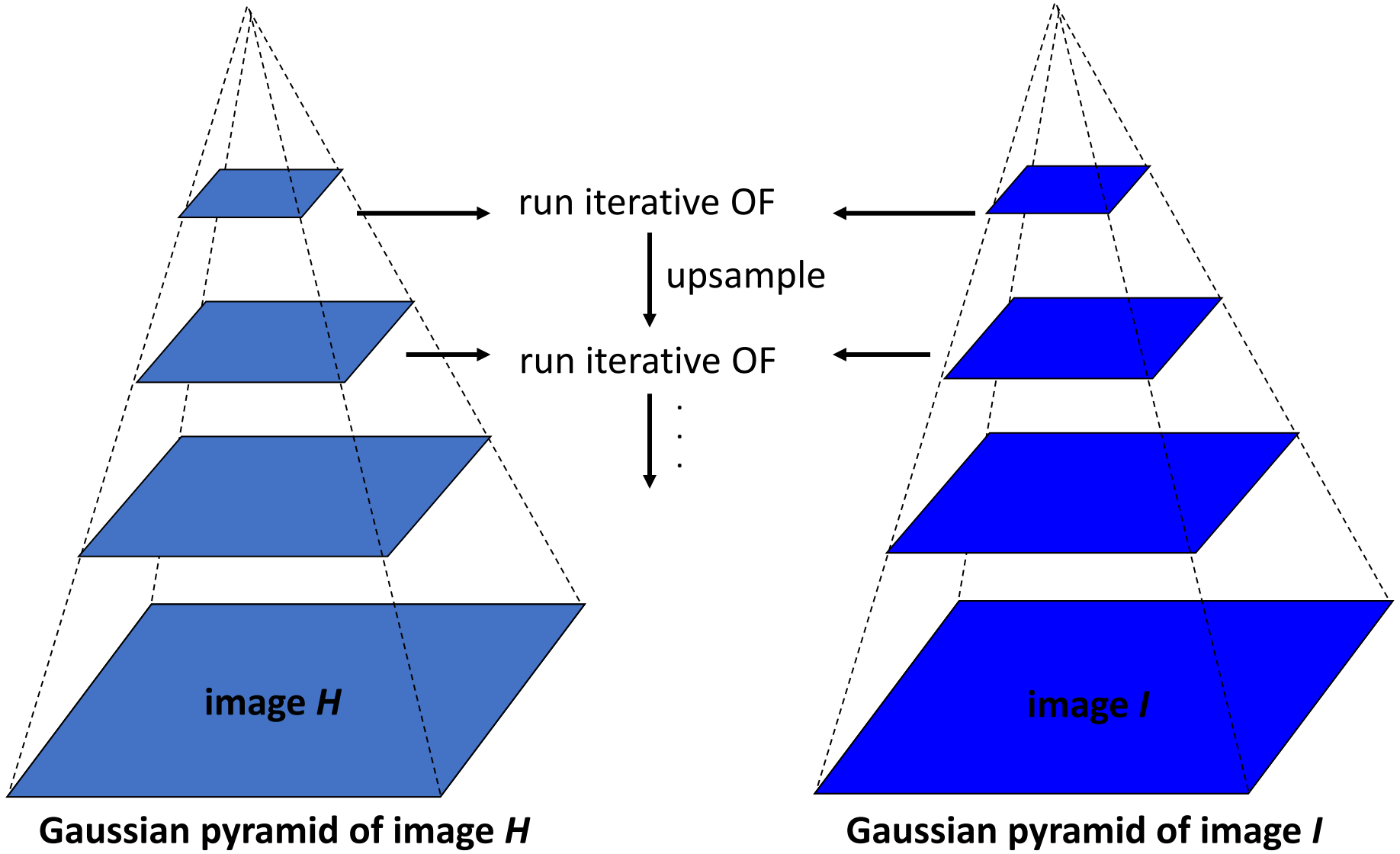
Optical Flow: Iterative Refinement

- Implementation Issues:
  - Warping one image, take derivative of the other image to avoid re-computation of gradient at each iteration
  - Useful to apply low-pass filter to images before motion estimation. This gives better derivative estimation and linear approximation to image intensity.

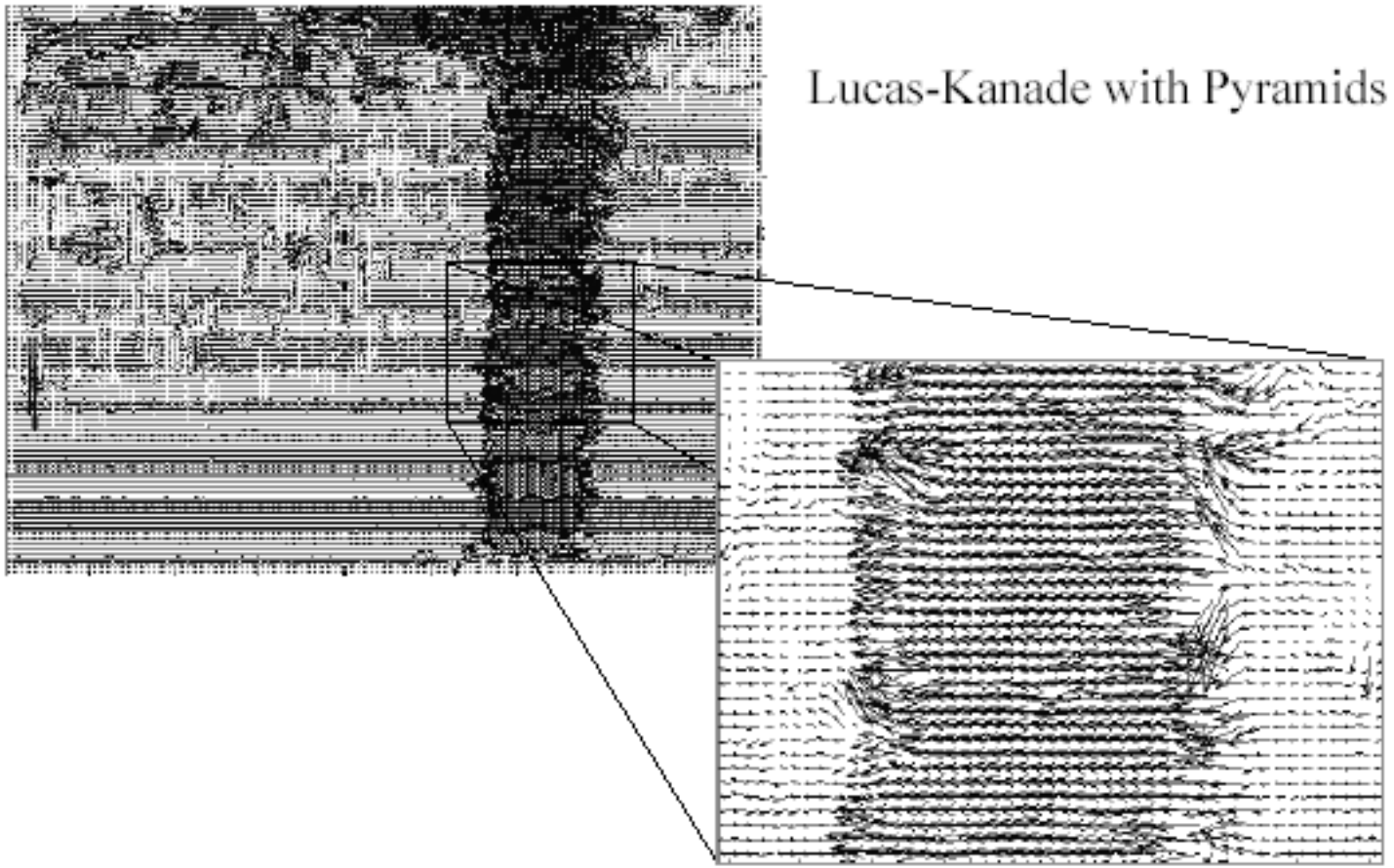
Coarse-to-fine Optical Flow Estimation



Coarse-to-fine Optical Flow Estimation

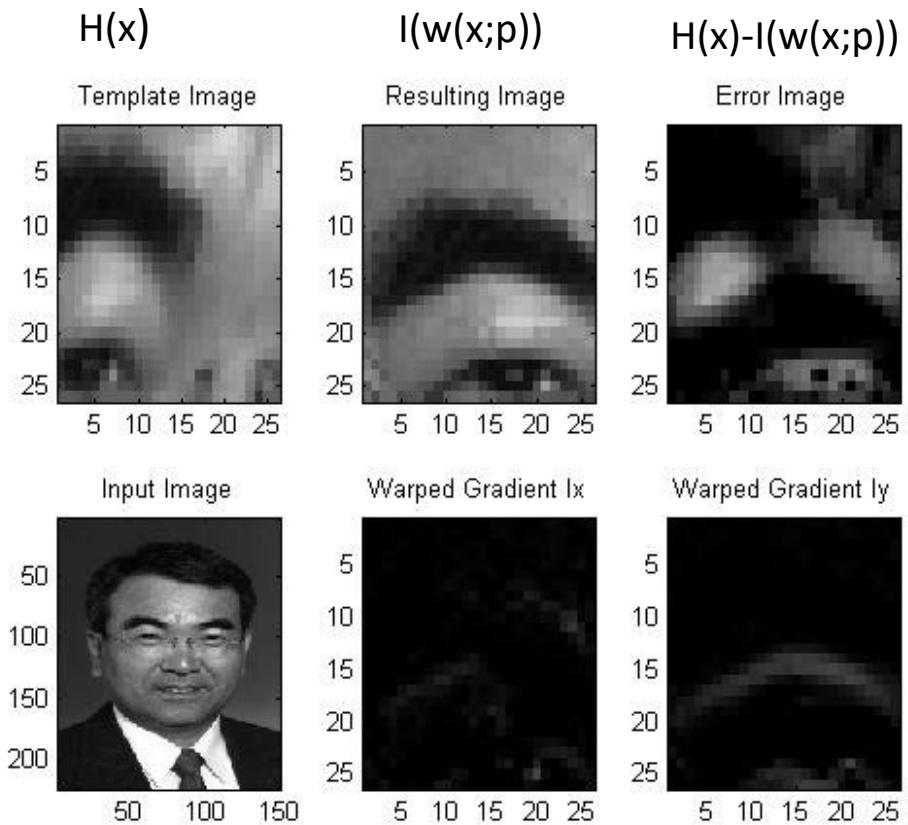


Optical Flow Results



Lucas-Kanade Algorithm

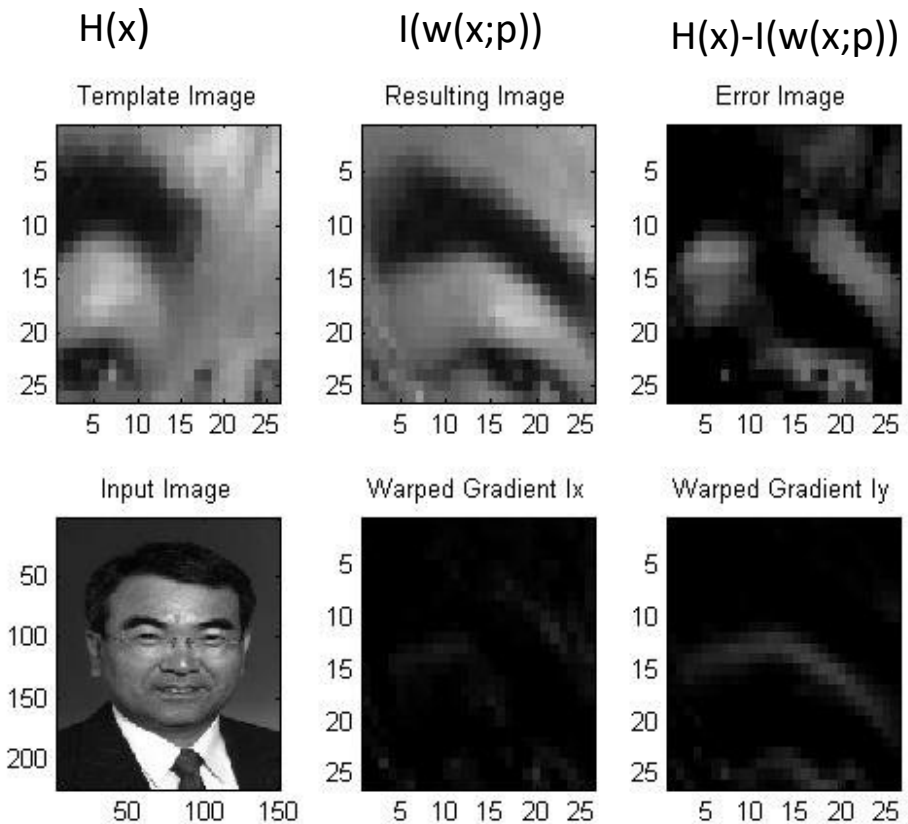
Iteration 1:



Source: J. Chai

Lucas-Kanade Algorithm

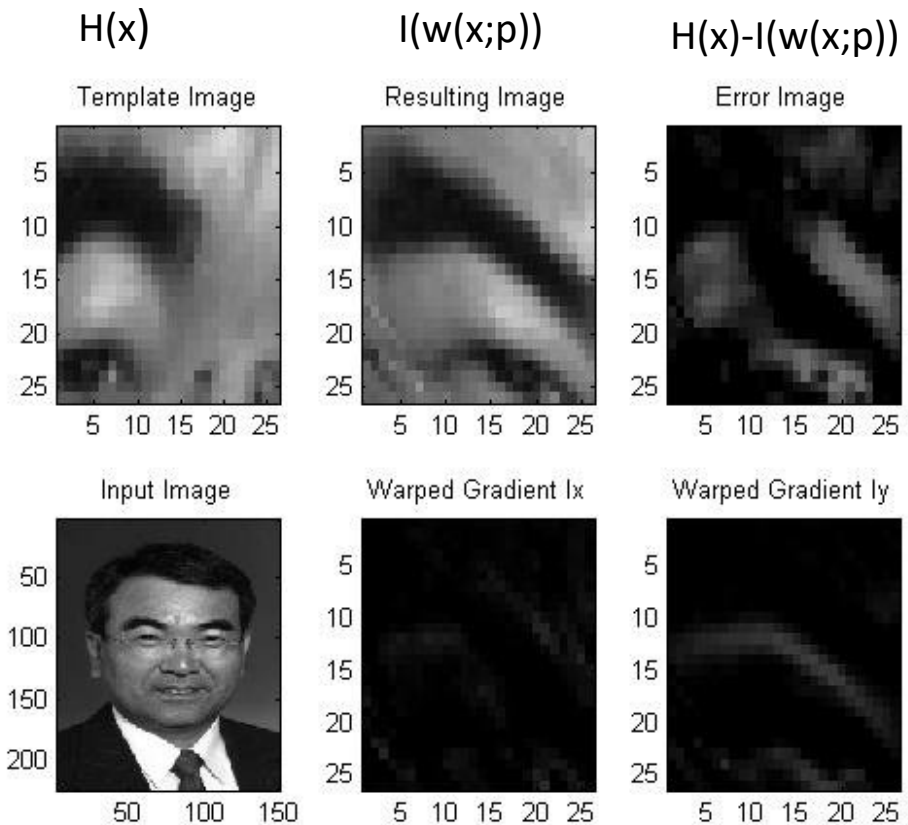
Iteration 2:



Source: J. Chai

Lucas-Kanade Algorithm

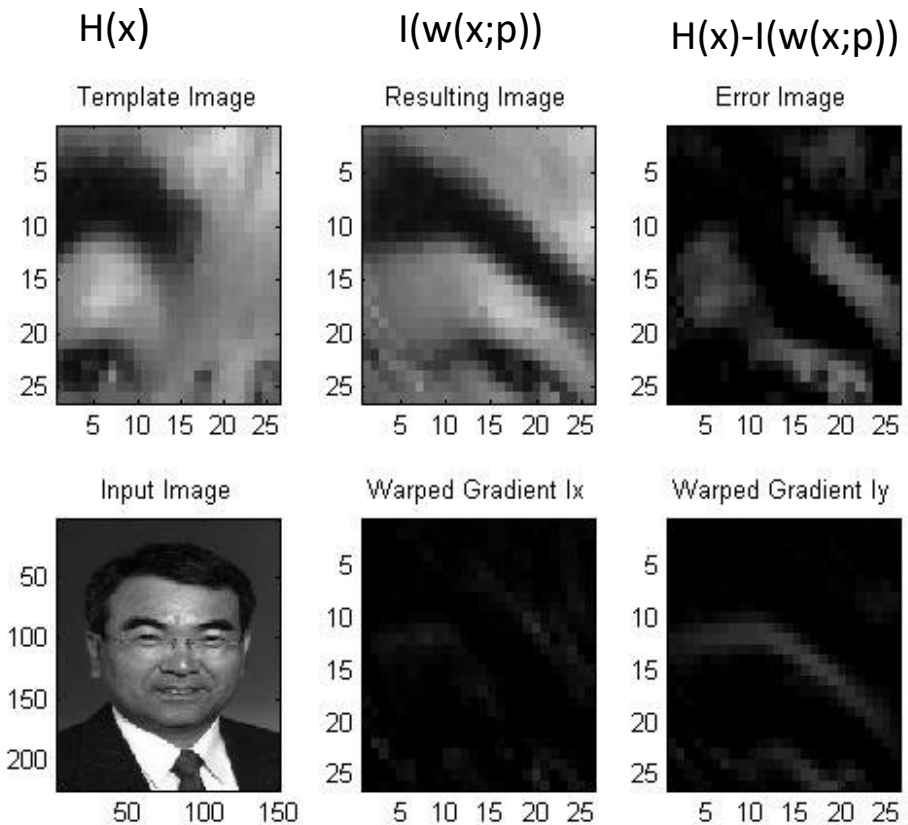
Iteration 3:



Source: J. Chai

Lucas-Kanade Algorithm

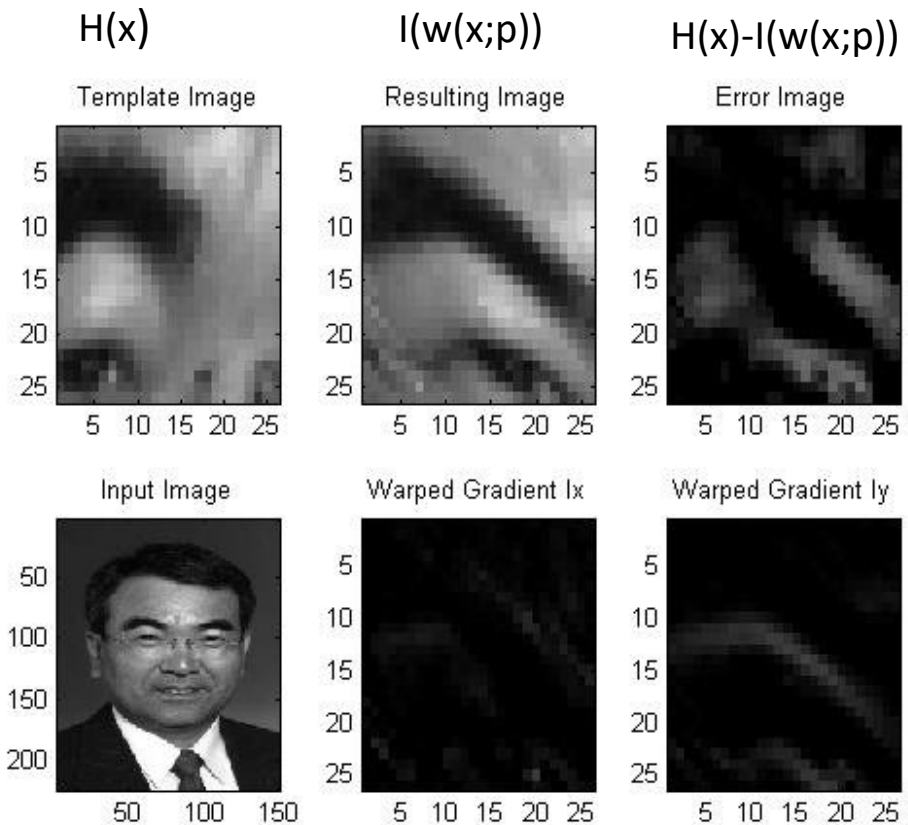
Iteration 4:



Source: J. Chai

Lucas-Kanade Algorithm

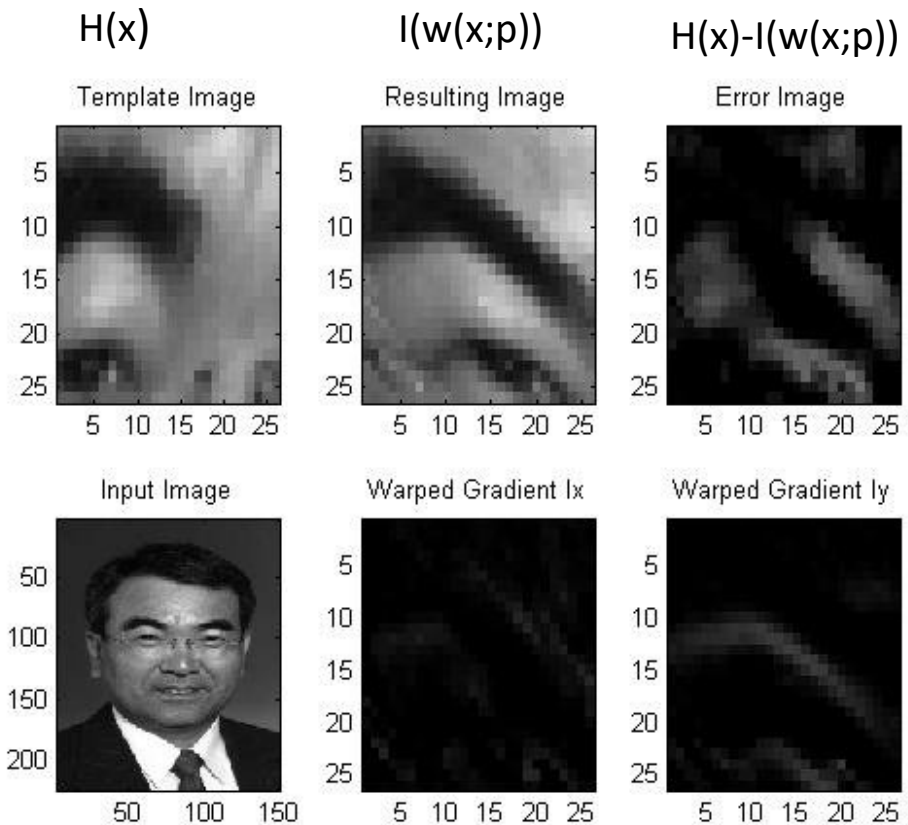
Iteration 5:



Source: J. Chai

Lucas-Kanade Algorithm

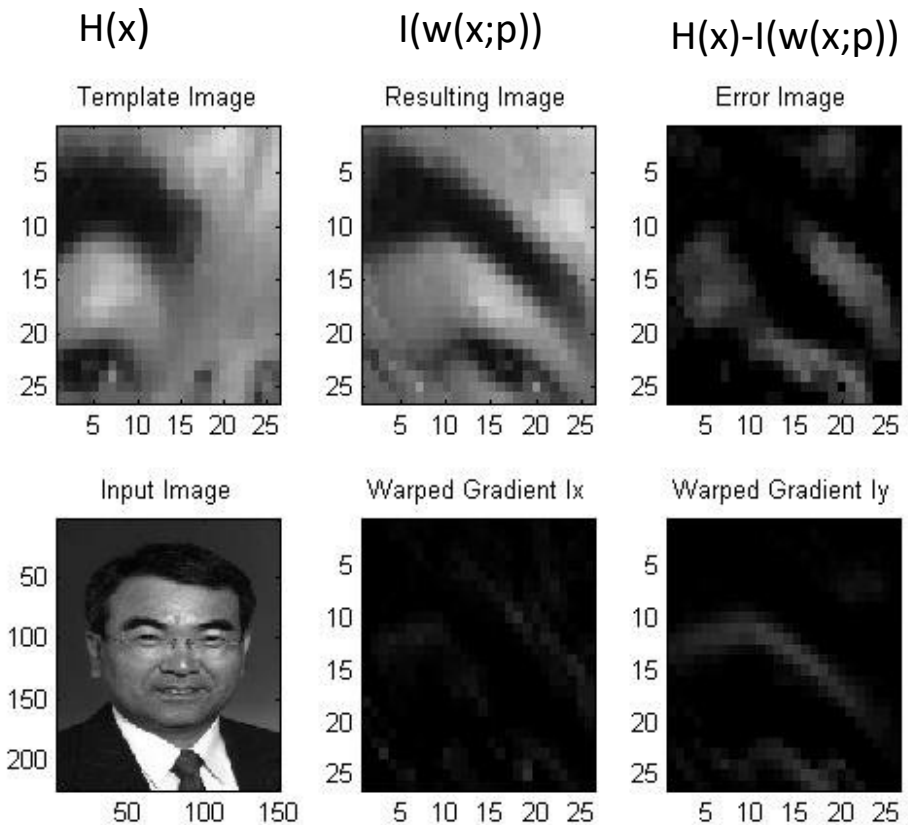
Iteration 6:



Source: J. Chai

Lucas-Kanade Algorithm

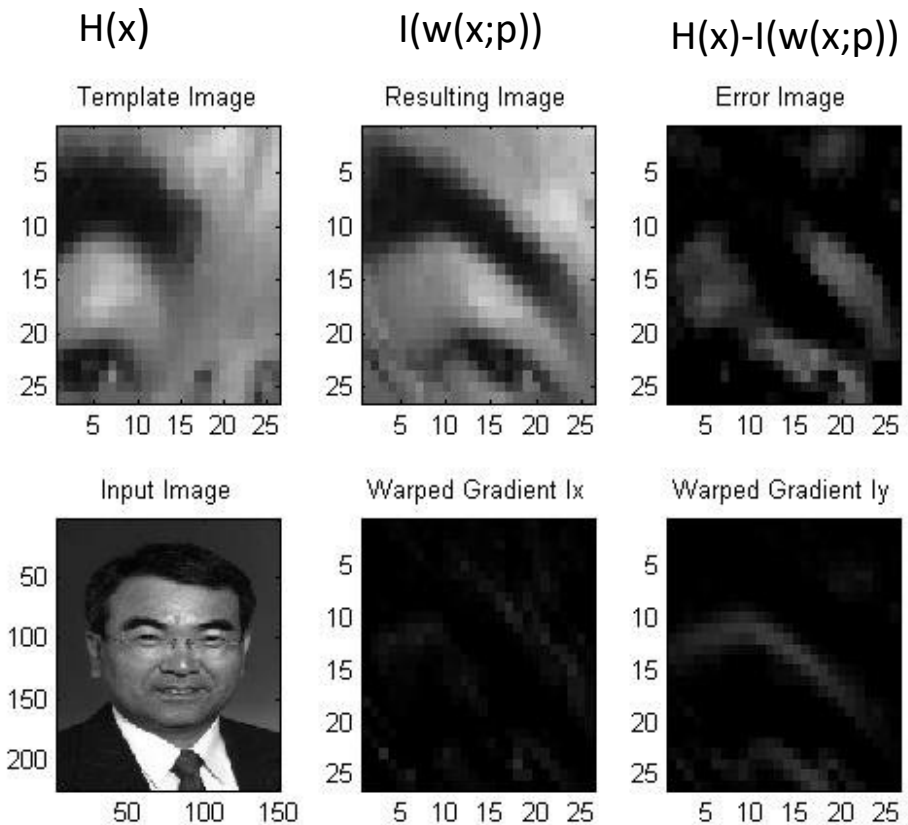
Iteration 7:



Source: J. Chai

Lucas-Kanade Algorithm

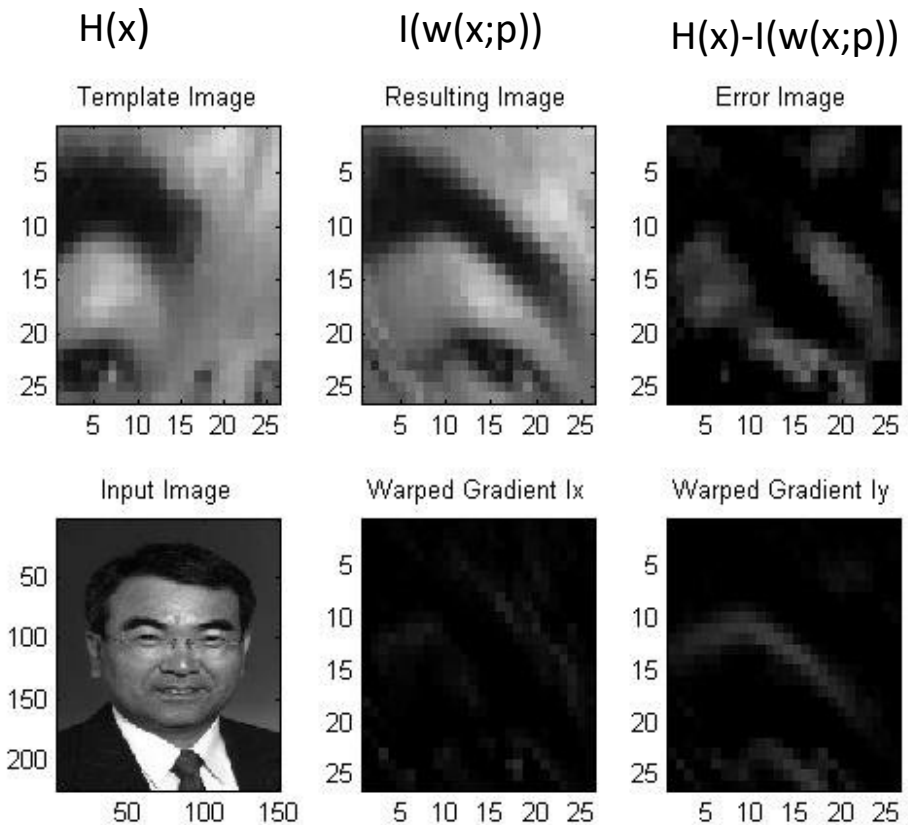
Iteration 8:



Source: J. Chai

Lucas-Kanade Algorithm

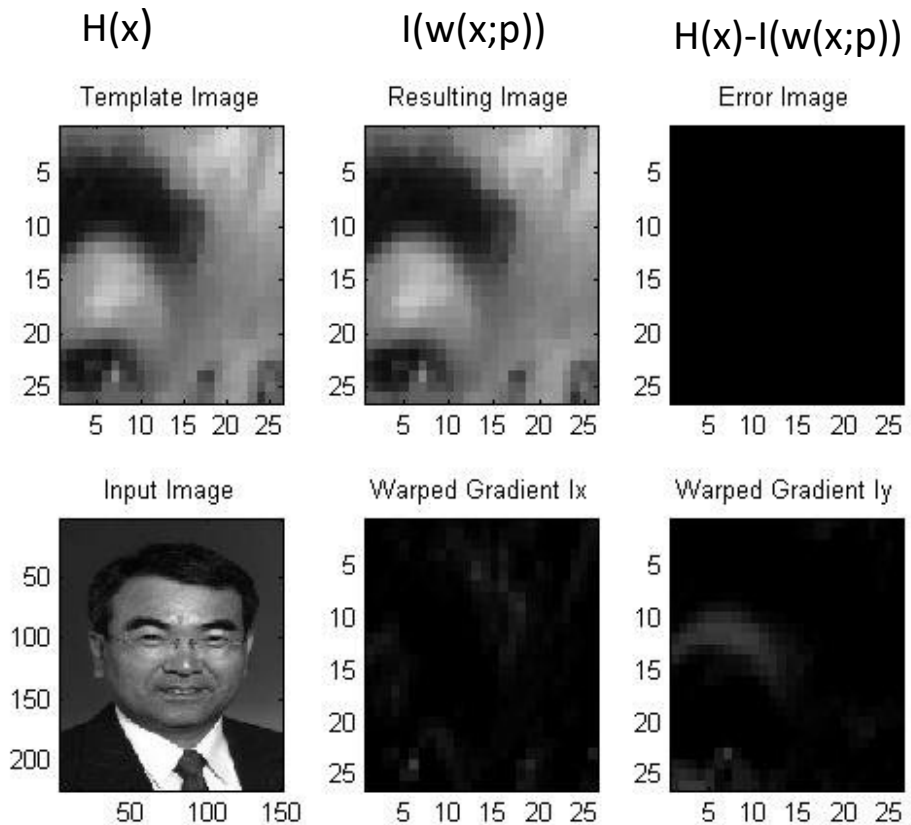
Iteration 9:



Source: J. Chai

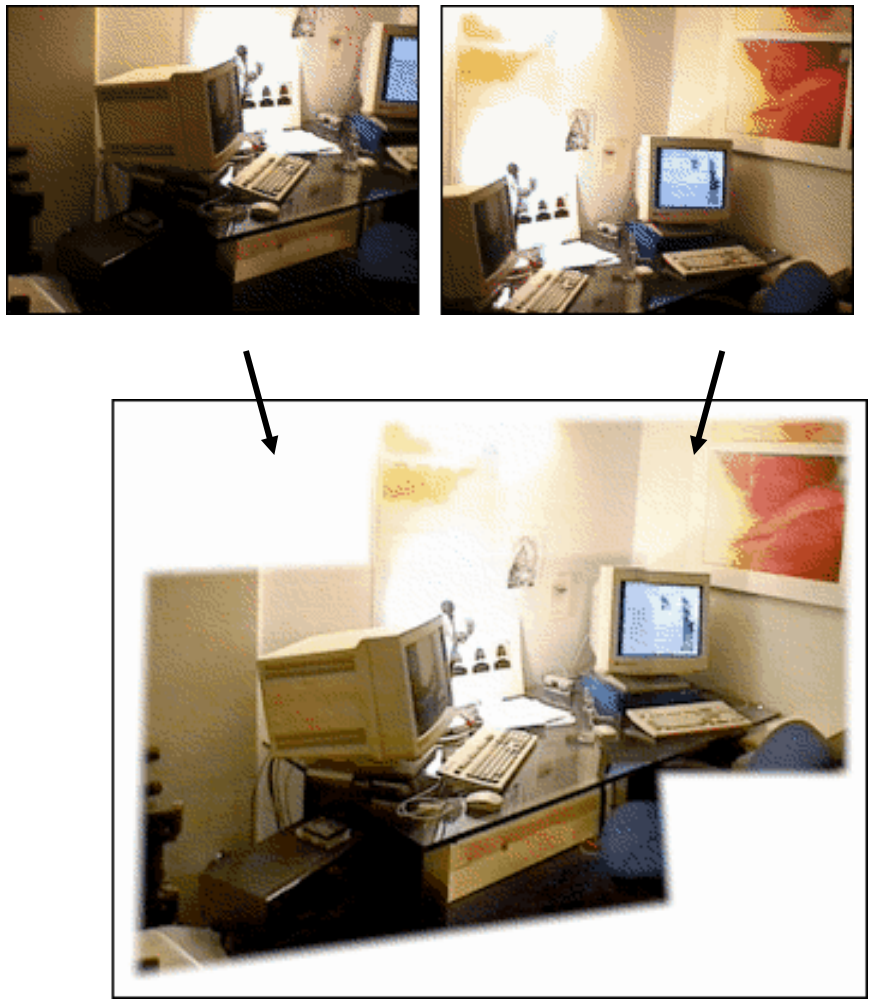
Lucas-Kanade Algorithm

Final result:



Source: J. Chai

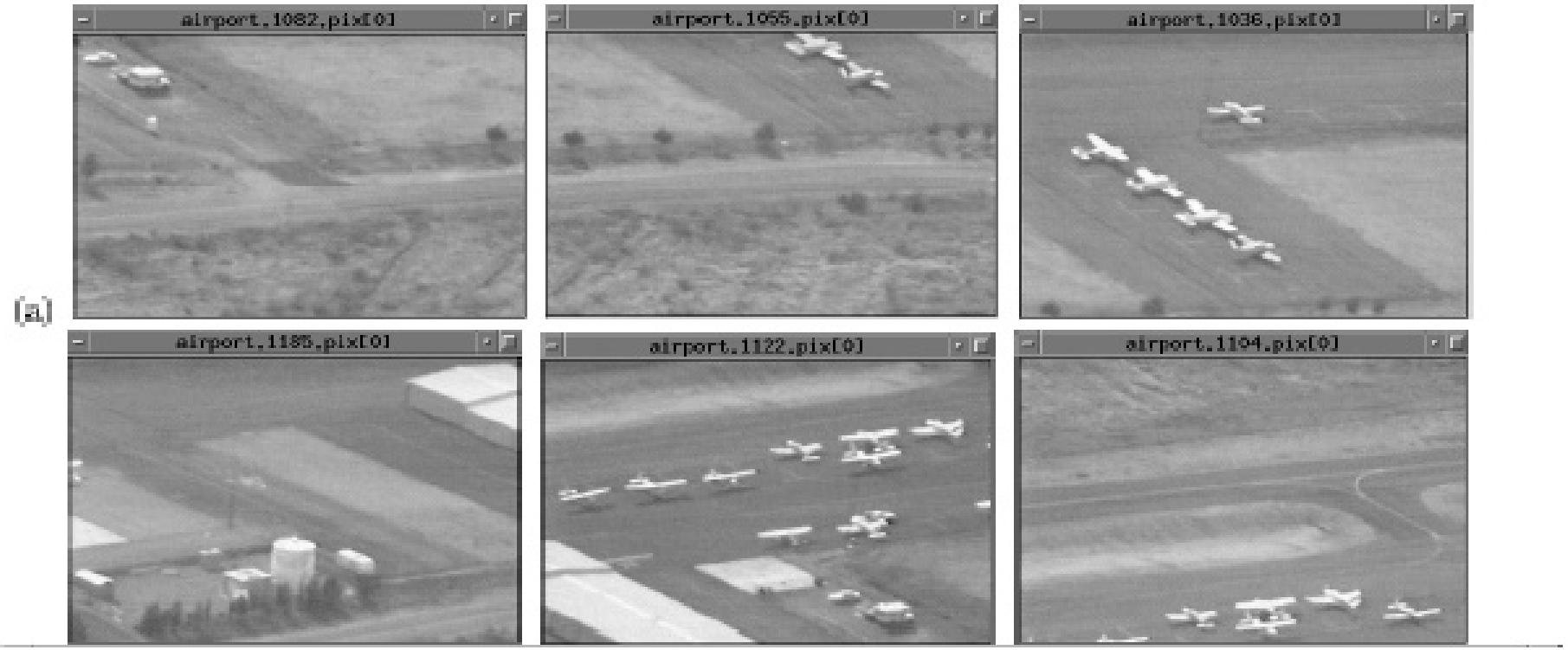
Image Alignment



- Goal: Estimate single *(u,v)* translation (transformation) for entire image

Slide source: S. Narasimhan

Example: Image Mosaic



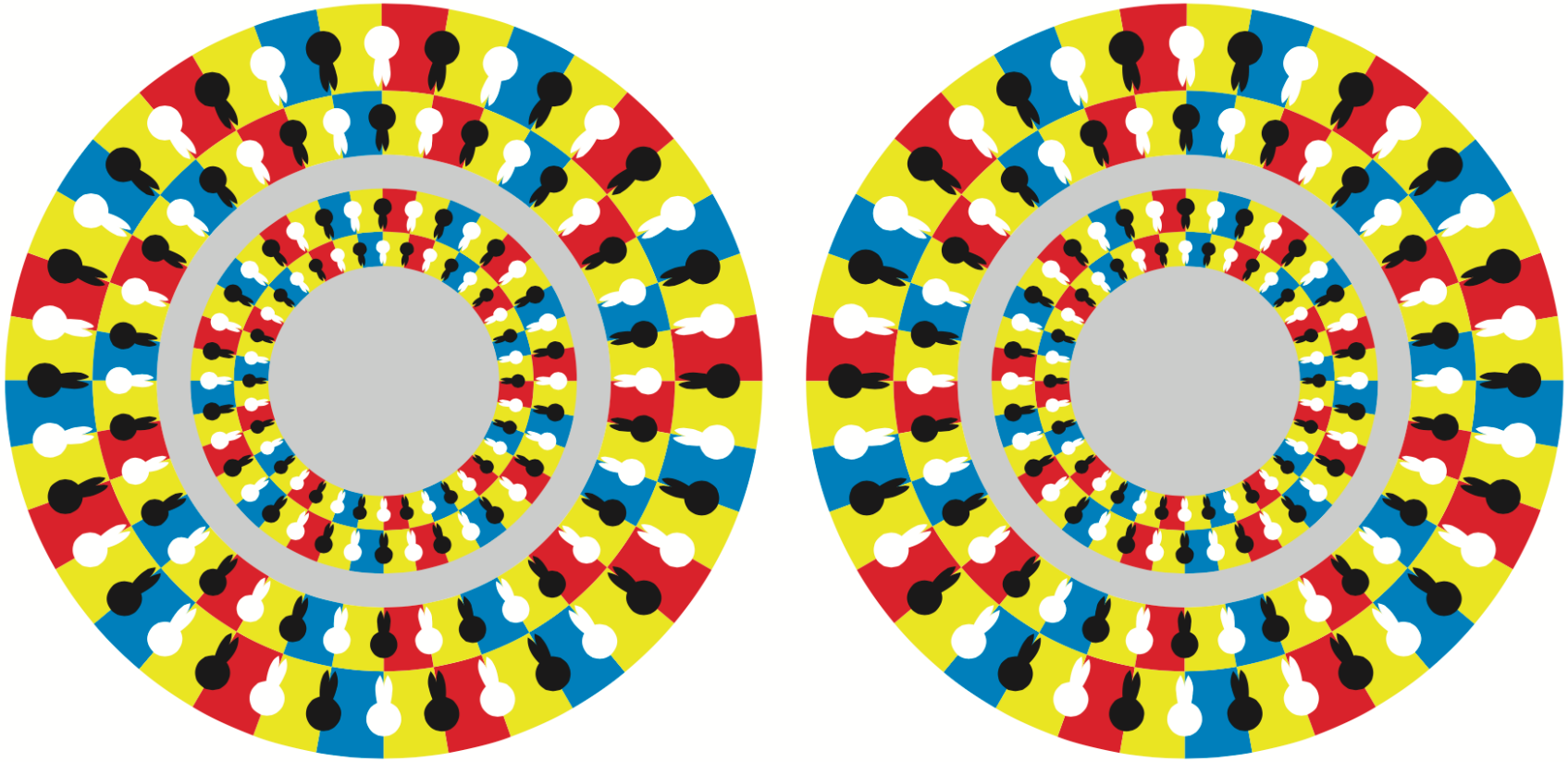
(Michal Irani, Weizmann)

Image Mosaic

. 1. Static background mosaic of an airport video clip.
(a) A few representative frames from the minute-long video clip. The video shows an airport being imaged from the air with a moving camera. The scene itself is static (i.e., no moving objects). (b) The static background mosaic image which provides an extended view of the entire scene imaged by the camera in the one-minute video clip.

(Michal Irani, Weizmann)

Motion?



http://www.psy.ritsumei.ac.jp/~akitaoka/uzumakie.html

Motion magnification using optical flow

How would you achieve this effect?

original

motion-magnified

- Compute optical flow from frame to frame.
- Magnify optical flow velocities.
- Appropriately warp image intensities.

How would you achieve this effect?

naïvely motion-magnified

motion-magnified

- Compute optical flow from frame to frame.
- Magnify optical flow velocities.
- Appropriately warp image intensities.

In practice, many additional steps are required for a good result.

Some more examples

Some more examples

Summary

- In this lecture:

- Definition: optical flow is the apparent motion of brightness patterns in the images.
- Aperture Problem of optical flow
- Additional constraints for optical flow
  - Horn and Schunck method
  - Lucas Kanade Optical Flow

Further Reading

- Berthold K. P. Horn and Brian G. Schunck.(1980) Determining Optical Flow. MIT, Cambridge, MA.

- Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application in stereo vision. In Seventh International Joint Conference on Artificial Intelligence (IJCAI-81), pp. 674–679, Vancouver.

- Richard Szeliski, Ch. 8.4 P. 409~419

Appendix

Cramer's Rule

- Considering a system

$$Ax = b$$

  - Where A is *nxn* matrix and $\det(\mathrm{A}) \neq 0$
  - vector $x = (x_1 \dots x_n)^T$

$$x_i = \frac{\det(A_i)}{\det(A)}, i = 1, \dots n$$

  - Where A_i is the matrix formed by replacing the *i-th* column of A by the column of b

Example for Cramer's Rule

- Consider the linear system

$$\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}$$

- In Matrix form

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

$$x = \frac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} = \frac{c_1 b_2 - b_1 c_2}{a_1 b_2 - b_1 a_2}$$

$$y = \frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} = \frac{a_1 c_2 - a_2 c_1}{a_1 b_2 - b_1 a_2}$$

Back