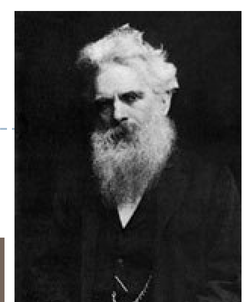
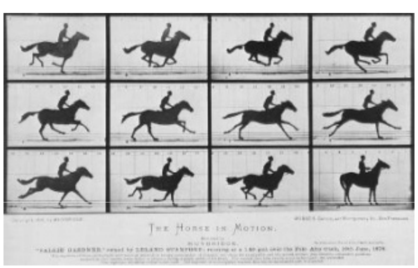
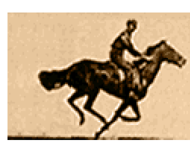
http://www.rit.edu/~andpph/photofile-b/stroboscopy-basketball-1a.jpg

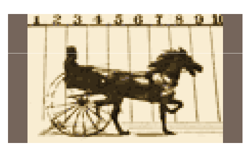
Motion

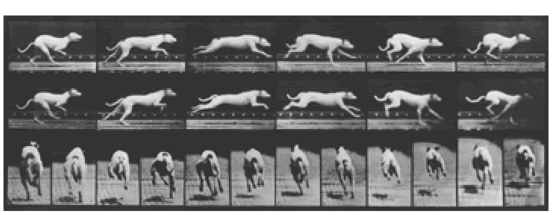
Muybridge: Father of Motion Photography

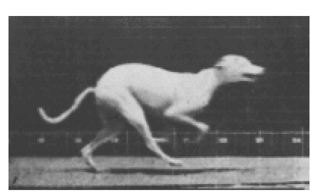




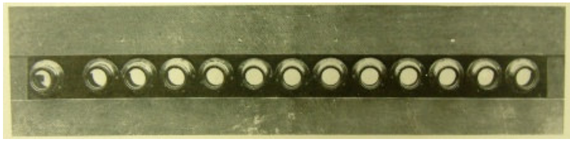


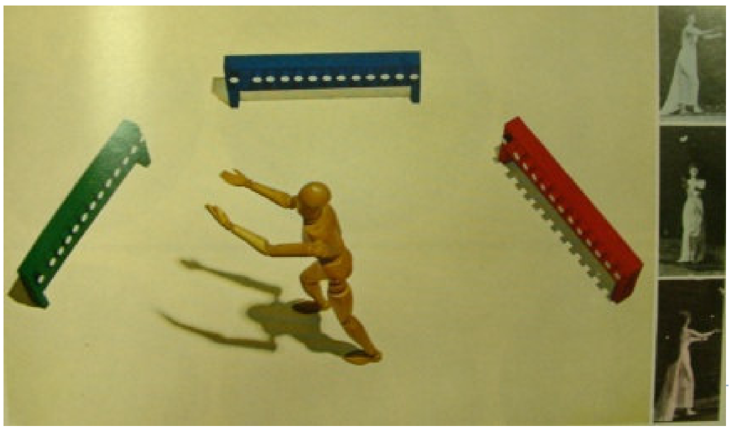


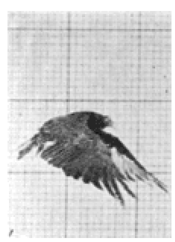


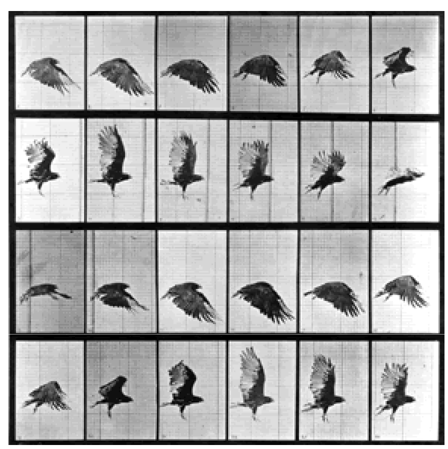


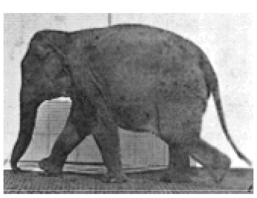
Muybridge’s Equipment

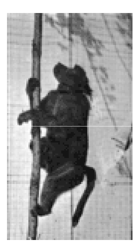


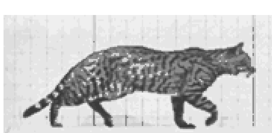


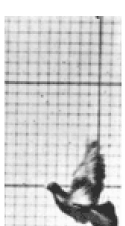






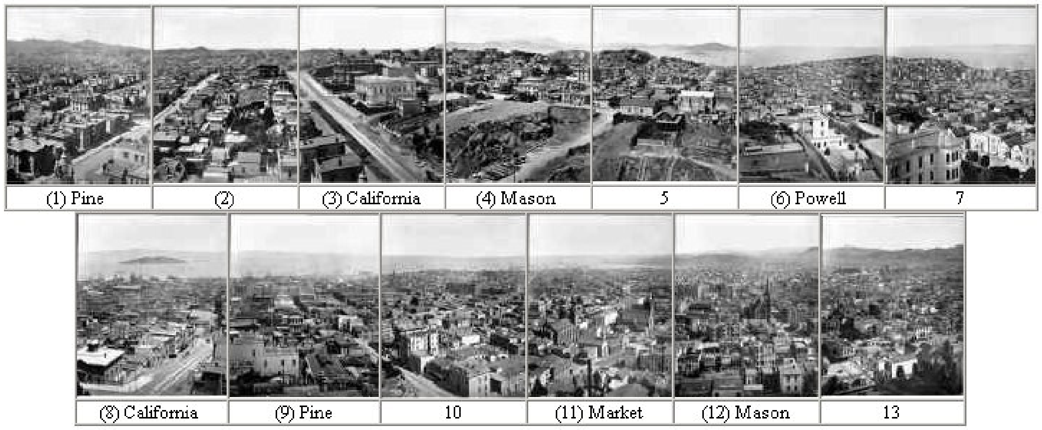






http://photo.ucr.edu/photographers/muybridge/default.html

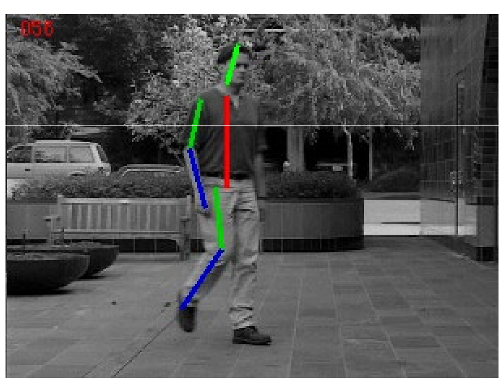
Muybridge’s Panoramas



http://americahurrah.com/SanFrancisco/Muybridge/Panorama.htm

More than 100 years later…

- Digital Muybridge Project



Motion

- Brightness constancy assumption:
  - Pixel color at location (*x, y*) in image taken at time *t* will be the same for some pixel (*x+dx, y+dy*) in image taken at time *t+dt*

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

Brightness Constancy Assumption

Motion

- Recall Taylor's Series

$$f(x) = f(a) + (x-a) f'(a) + \frac{1}{2!}(x-a)^2 f''(a) + \ldots$$

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

$$\Downarrow$$

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

Motion

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

$$\Downarrow$$

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

Motion

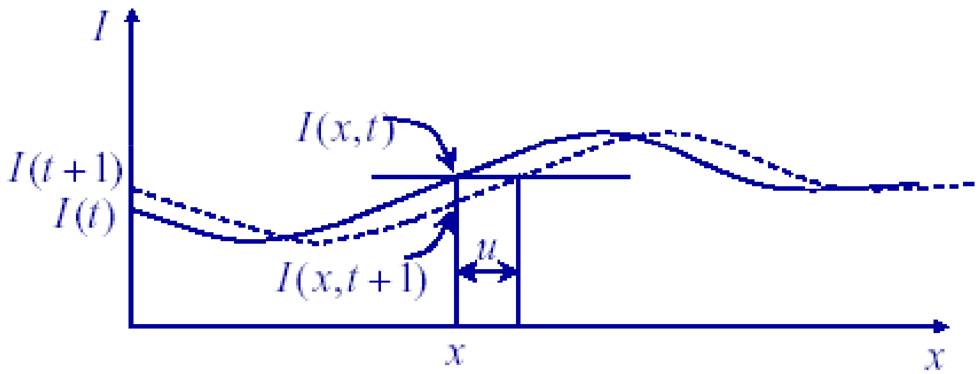
$$\frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial t} = 0$$

$$f_x \frac{dx}{dt} + f_y \frac{dy}{dt} + f_t = 0$$

$$f_x u + f_y v + f_t = 0 \qquad [f_x \quad f_y]\begin{bmatrix} u \\ v \end{bmatrix} = -f_t$$

Brightness Constancy Equation (BCE)

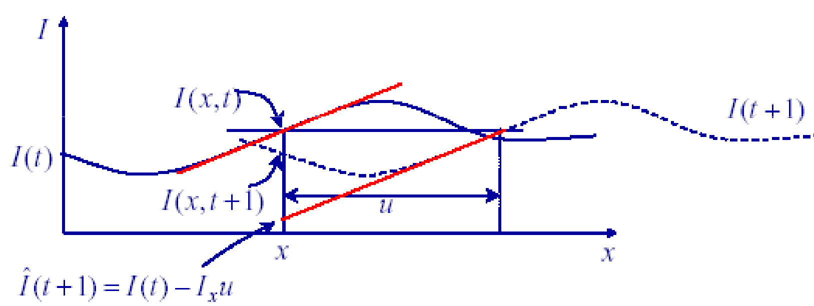
Interpretation of BCE



$$I_x u = -(I(t+1) - I(t)) \quad \text{or} \quad I_x u + I_t = 0$$

Ref: Hai Tao's Lecture Slides

For Large Motions…



The Brightness constancy equation does not hold for large motions…

Ref: Hai Tao's Lecture Slides

Interpretation of BCE

- Another Interpretation

  The change in brightness at a pixel if the motion is given by (u,v) is

$$-\left(f_x u + f_y v\right)$$

- The local brightness is modeled as a plane with slope f_x in x-direction and f_y in y-direction

Interpretation of BCE

$$\boxed{f_x u + f_y v + f_t = 0}$$

Assumptions:

1. Brightness Constancy
2. Image is differentiable
3. Well modeled by first order derivatives

u, v are unknowns here

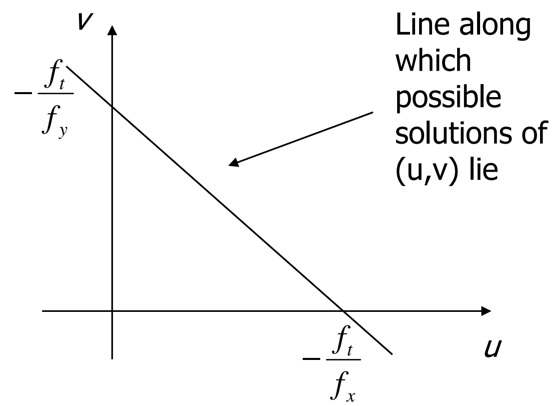
2 unknowns, 1 equation gives a linear constraint

$$v = -\frac{f_x}{f_y} u - \frac{f_t}{f_y}$$

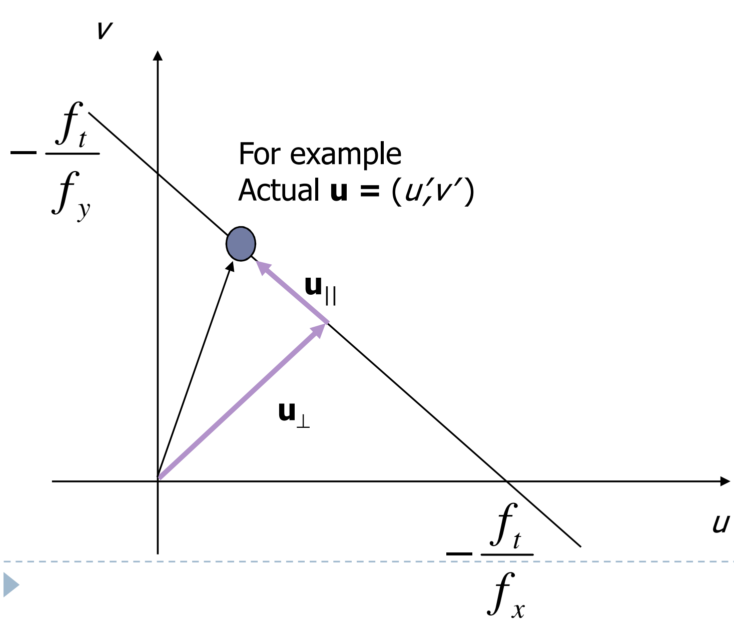
$\Rightarrow$ Of the form $y = mx + c$

Interpretation of BCE

$$v = -\frac{f_x}{f_y} u - \frac{f_t}{f_y}$$



Interpretation of BCE



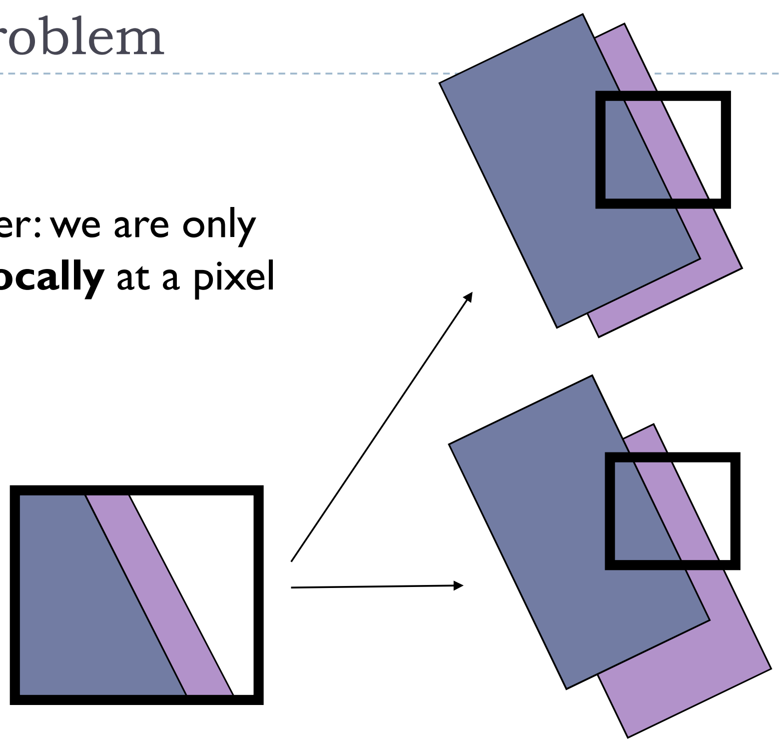
$$u_{\text{per}} = \frac{f_t}{\sqrt{f_x^2 + f_y^2}}$$

Interpretation of BCE

- For all points along the constraint line, the perpendicular component is the same, but the parallel component is different
- We can find the perpendicular component of motion but not the parallel component
- Physical Interpretation?

Aperture Problem

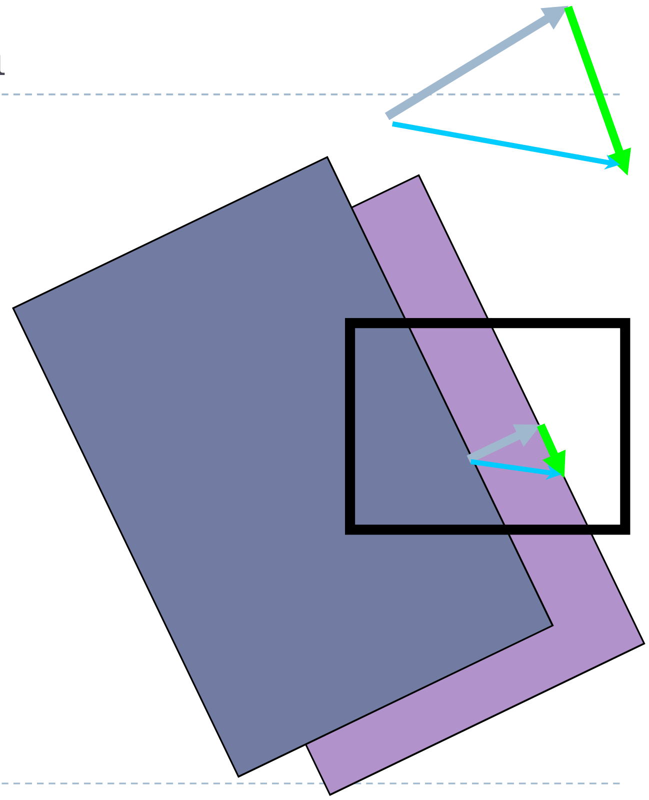
- Remember: we are only looking **locally** at a pixel



Aperture Problem

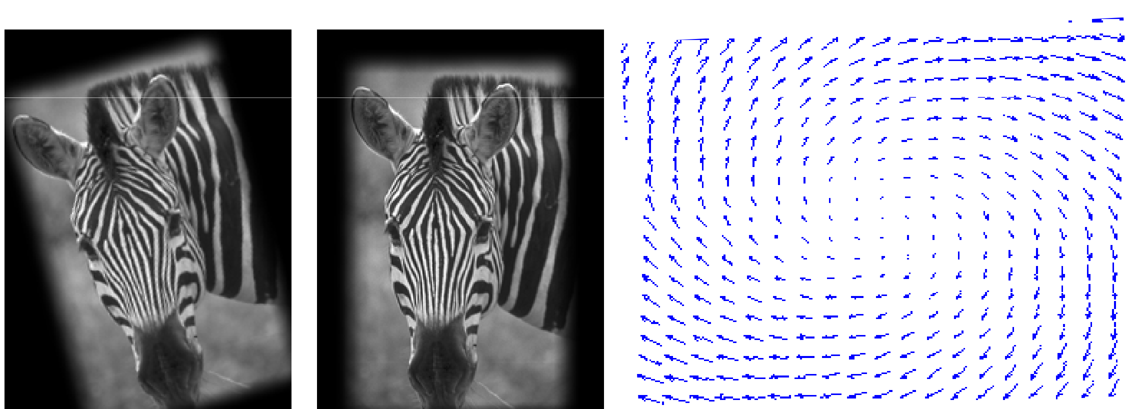
- **Definition**

  "The component of **motion field** in the direction orthogonal to the spatial image gradient is not constrained by the brightness constancy equation"



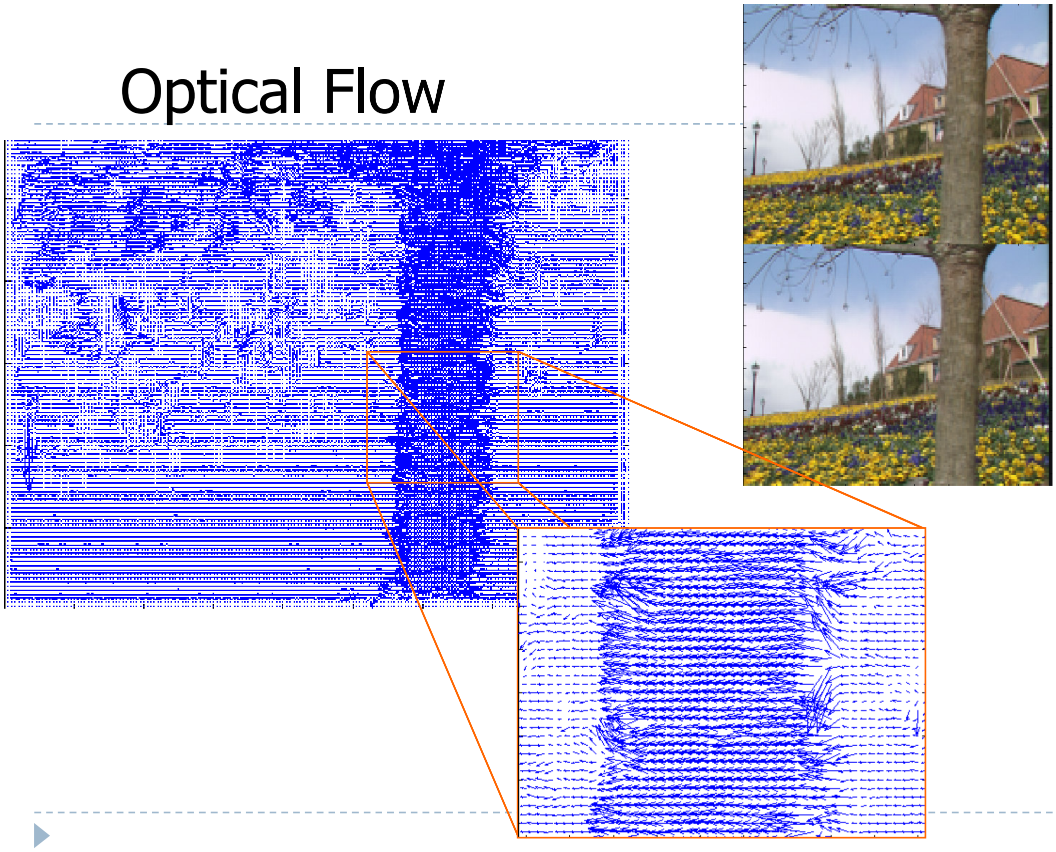
Optical Flow

- Computing motion field at every pixel



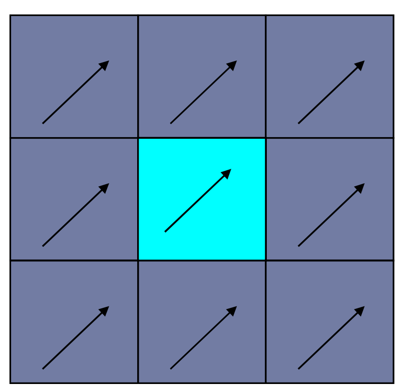
Optical Flow

Optical Flow



Computing Optical Flow

- To find motion at each pixel, we have 2 unknowns for each pixel but only one equation
- Solution?
- Assume that the motion is constant over a small neighborhood
- Known as the Lucas-Kanade Method



Lucas-Kanade Method

- Assume constant optical flow in a 3x3 window
- Yields __ equations and __ unknowns
- Over constrained system
- Solve?
- Using Least Squares

Lucas-Kanade Method

$$f_x u + f_y v = -f_t$$

Consider same (u,v) for a 3x3 window

$$f_{x1} u + f_{y1} v = -f_{t1}$$

$$f_{x2} u + f_{y2} v = -f_{t2}$$

$$\vdots$$

$$f_{x9} u + f_{y9} v = -f_{t9}$$

$$\begin{bmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t1} \\ -f_{t2} \\ \vdots \\ -f_{t9} \end{bmatrix}$$

Lucas-Kanade Method

$$\begin{bmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t1} \\ -f_{t2} \\ \vdots \\ -f_{t9} \end{bmatrix}$$

$$\mathbf{Au} = \mathbf{f}_t$$
$$\mathbf{A}^T\mathbf{Au} = \mathbf{A}^T\mathbf{f}_t$$
$$\mathbf{u} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{f}_t$$

Least Squares Solution

Lucas-Kanade Method

- Another way to write the same system of equations is

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{xi}^2 & \sum_i f_{xi} f_{yi} \\ \sum_i f_{xi} f_{yi} & \sum_i f_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{xi} f_{ti} \\ -\sum_i f_{yi} f_{ti} \end{bmatrix}$$

Why can we not apply BCE for large motions

- The relationship between x- y- and t-derivatives has to hold

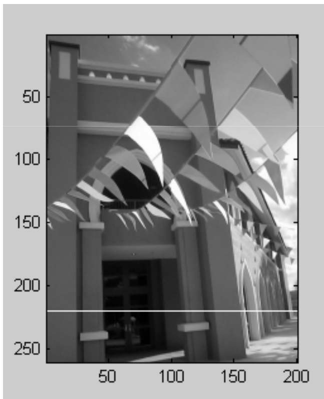
$$f_x u + f_y v + f_t = 0$$

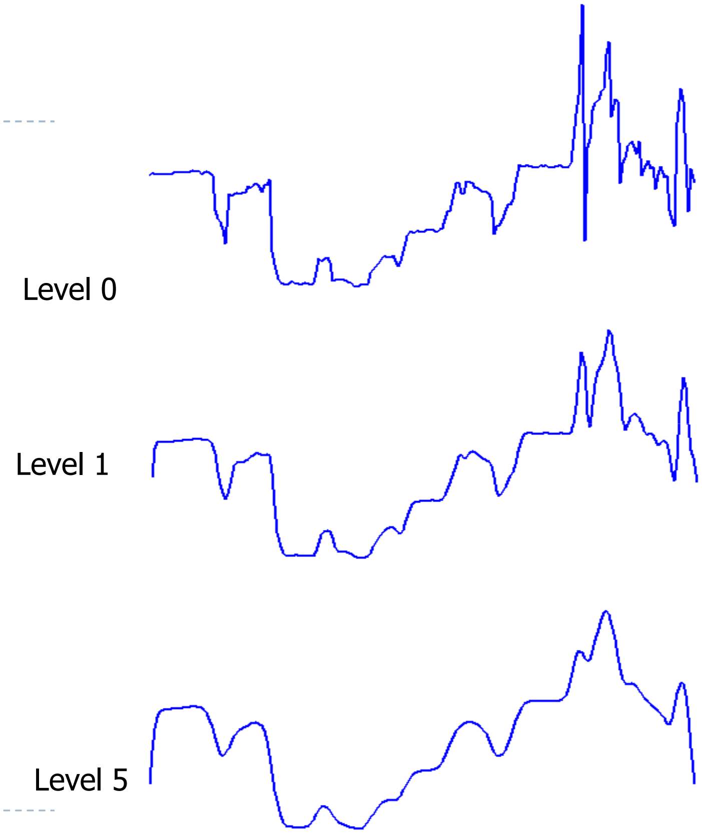
- The assumption that the time derivative is linearly related to the spatial derivatives breaks down
- If motion is very large, derivative window might be too small to capture it

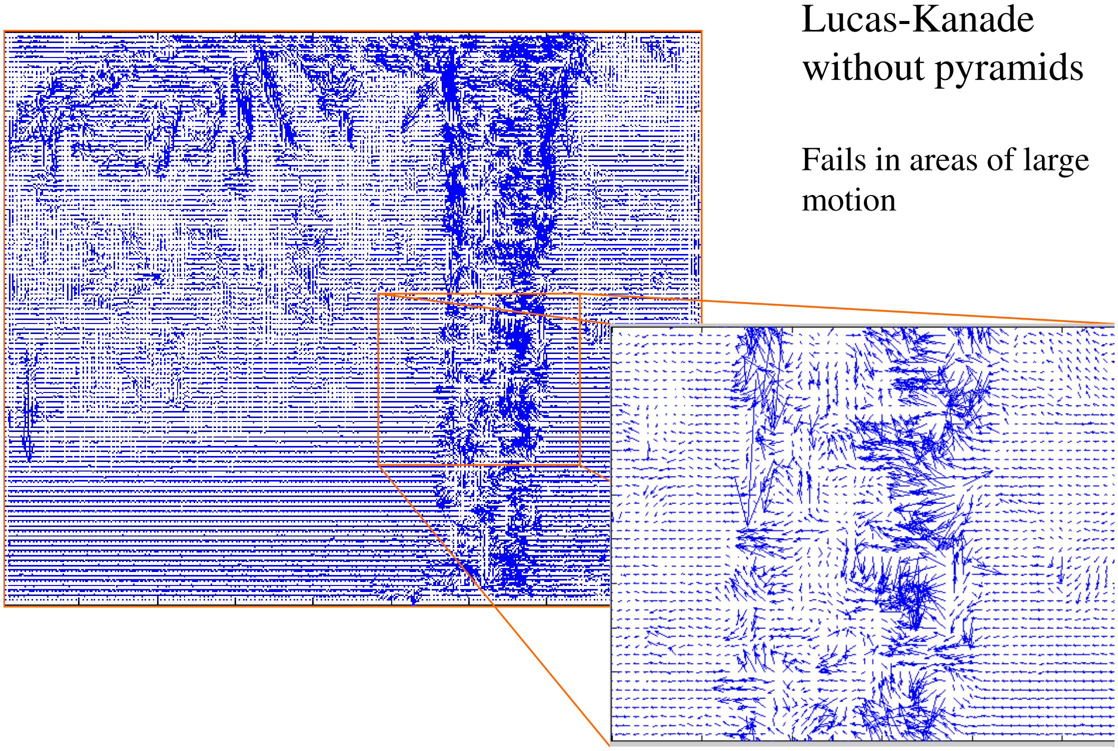
For large motions…

- The variations in derivatives may be reduced
- i.e. the image may be smoothed out
- Then, a larger mask may be used
- OR, smaller images may be used and operations done incrementally
- Solution: **Pyramids**

Reduce



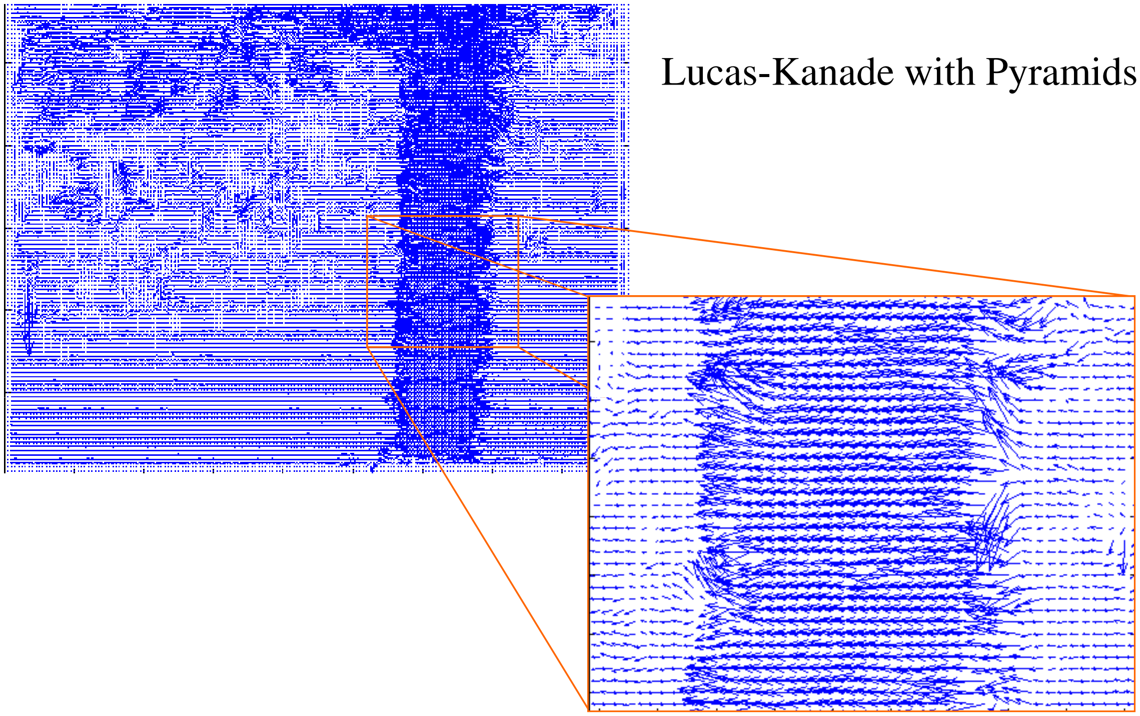




Lucas-Kanade
without pyramids

Fails in areas of large motion

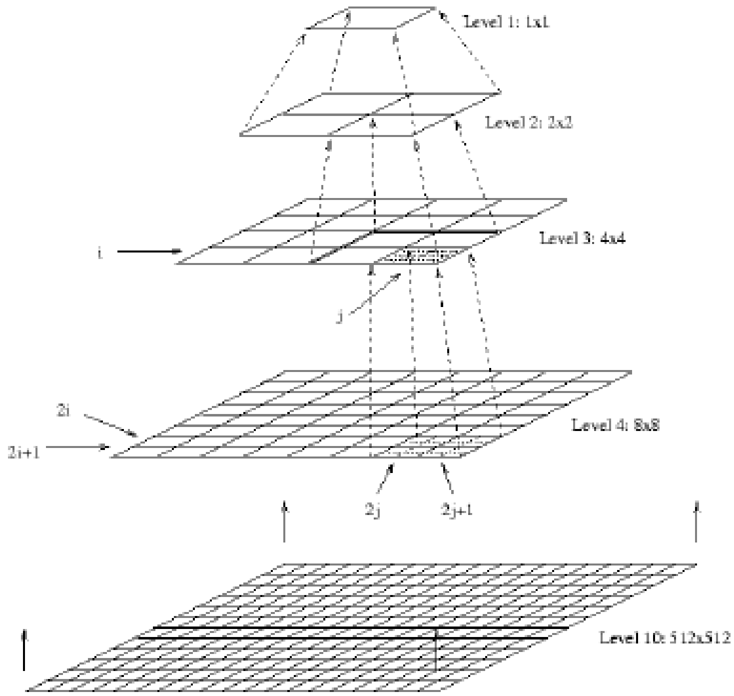
Lucas-Kanade with Pyramids



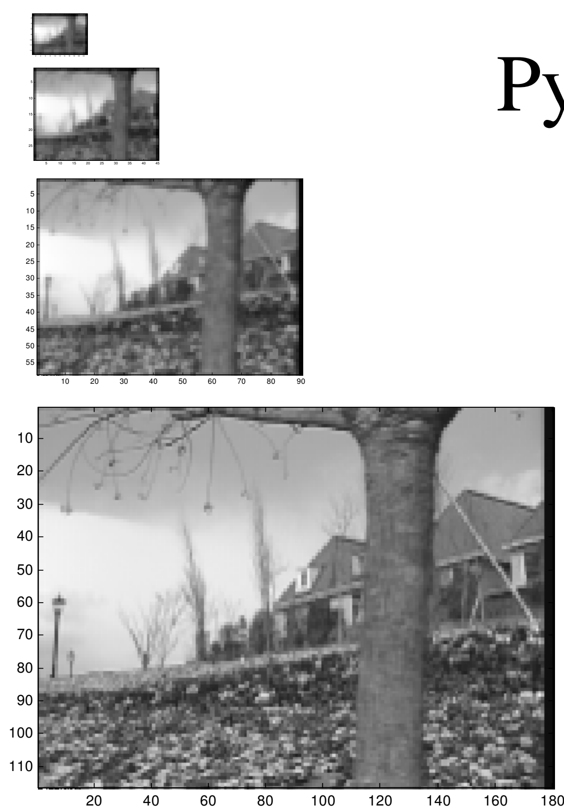
Pyramids

- Very useful image representation
- Pyramid representation has multiple copies of the image
- Each "level" is ¼ the size of the previous level
- Lowest level is highest resolution
- Highest level is lowest resolution

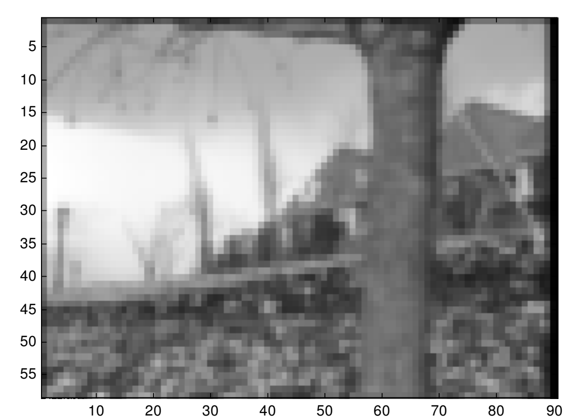
Pyramid

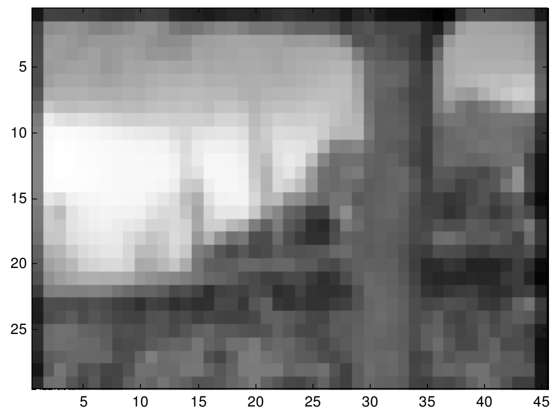


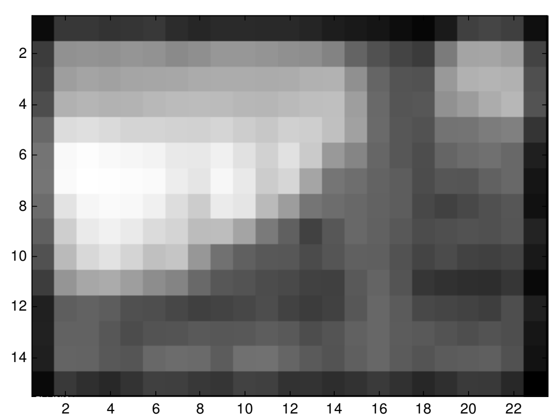
Pyramids




10
20
30
40
50
60
70
80
90
100
110
20
40
60
80
100
120
140
160
180

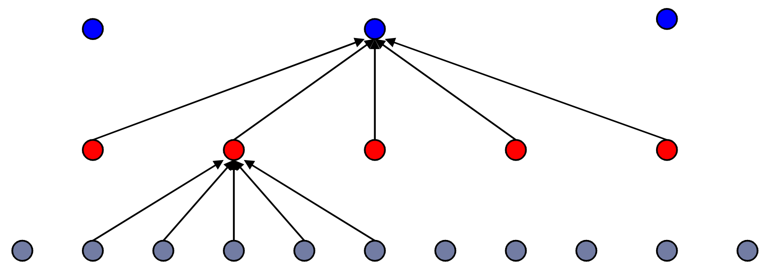

5
10
15
20
25
30
35
40
45
50
55
10
20
30
40
50
60
70
80
90


5
10
15
20
25
5
10
15
20
25
30
35
40
45


2
4
6
8
10
12
14
2
4
6
8
10
12
14
16
18
20
22

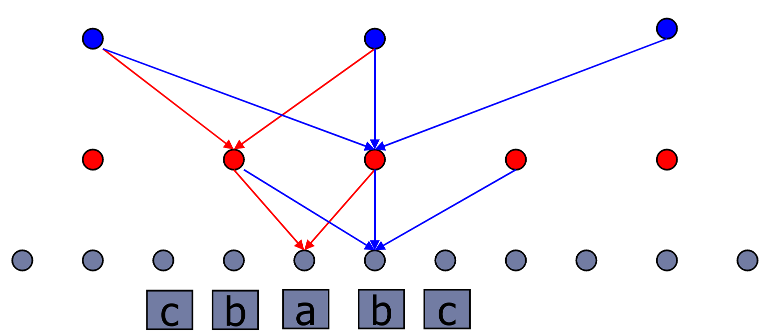
Pyramids… implementation

Reduce Operation



Pyramids

Expand Operation



Convolution Mask

- Separable

$$w(m,n) = \hat{w}(m)\hat{w}(n)$$

- Symmetric

$$\hat{w}(i) = \hat{w}(-i)$$

$$[c, b, a, b, c]$$

Convolution Mask

- The sum of mask should be 1.

$$a + 2b + 2c = 1$$

- All nodes at a given level must contribute the same total weight to the nodes at the next higher level.

$$a + 2c = 2b$$

Gaussian Pyramids

- Where weights are distributed according to a Gaussian Distribution
- Special property that Gaussian masks are separable

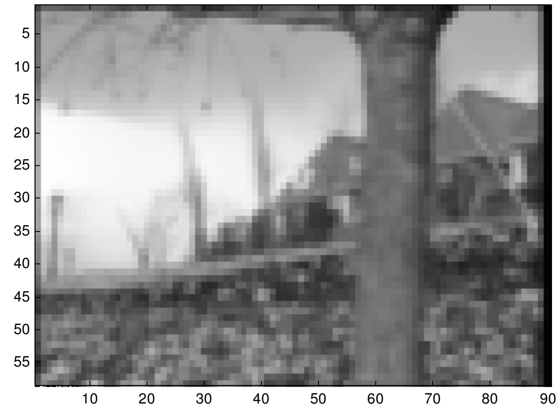
$$w(n,m) = \hat{w}(n) \otimes \hat{w}(m)$$

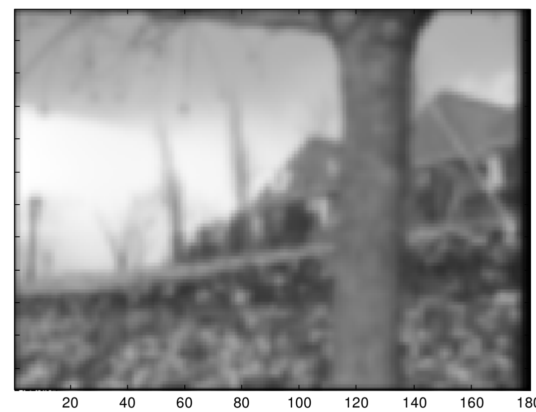
- `[0.05 0.25 0.4 0.25 0.05]`

Generating 2D Pyramid

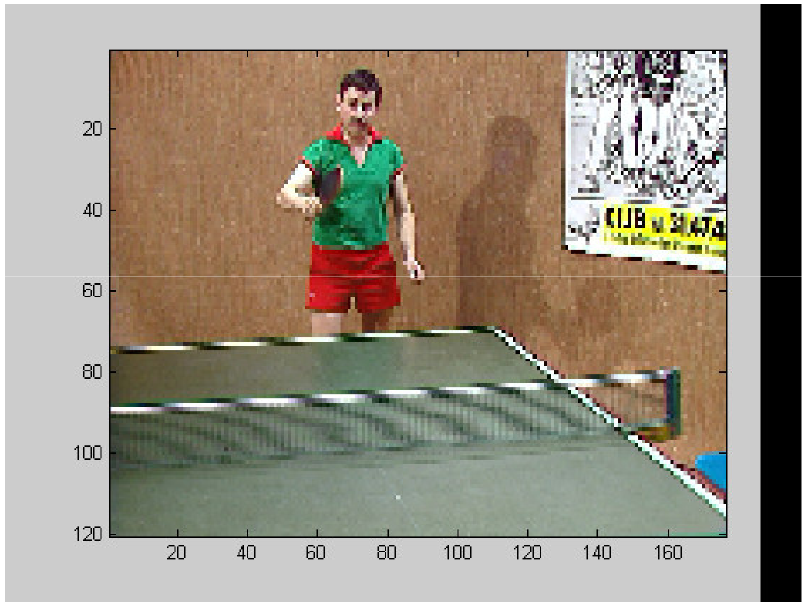
- Since Gaussian is separable,
- Apply mask to alternate pixels in row direction
- Apply mask to alternate columns of resultant image

Reduce followed by Expand





Why not just simply sub-sample?



20
40
60
80
100
120
20
40
60
80
100
120
140
160

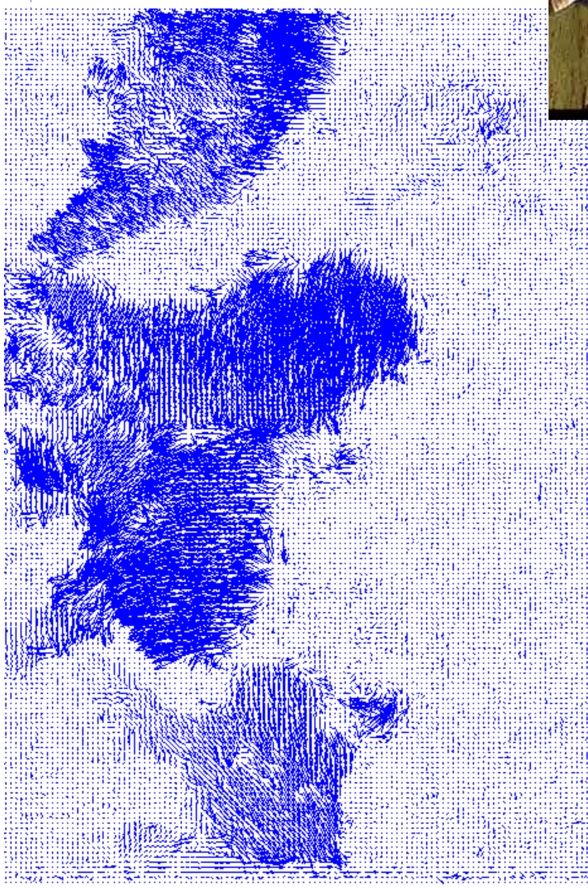
20
40
60
80
100
120
20
40
60
80
100
120
140
160

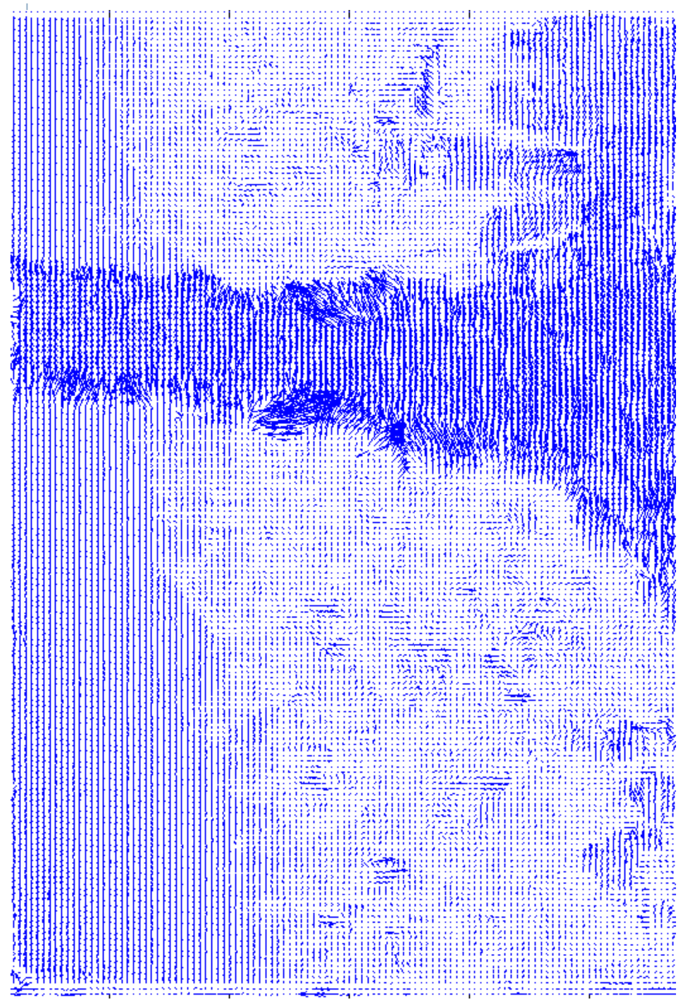
Why not just simply sub-sample?

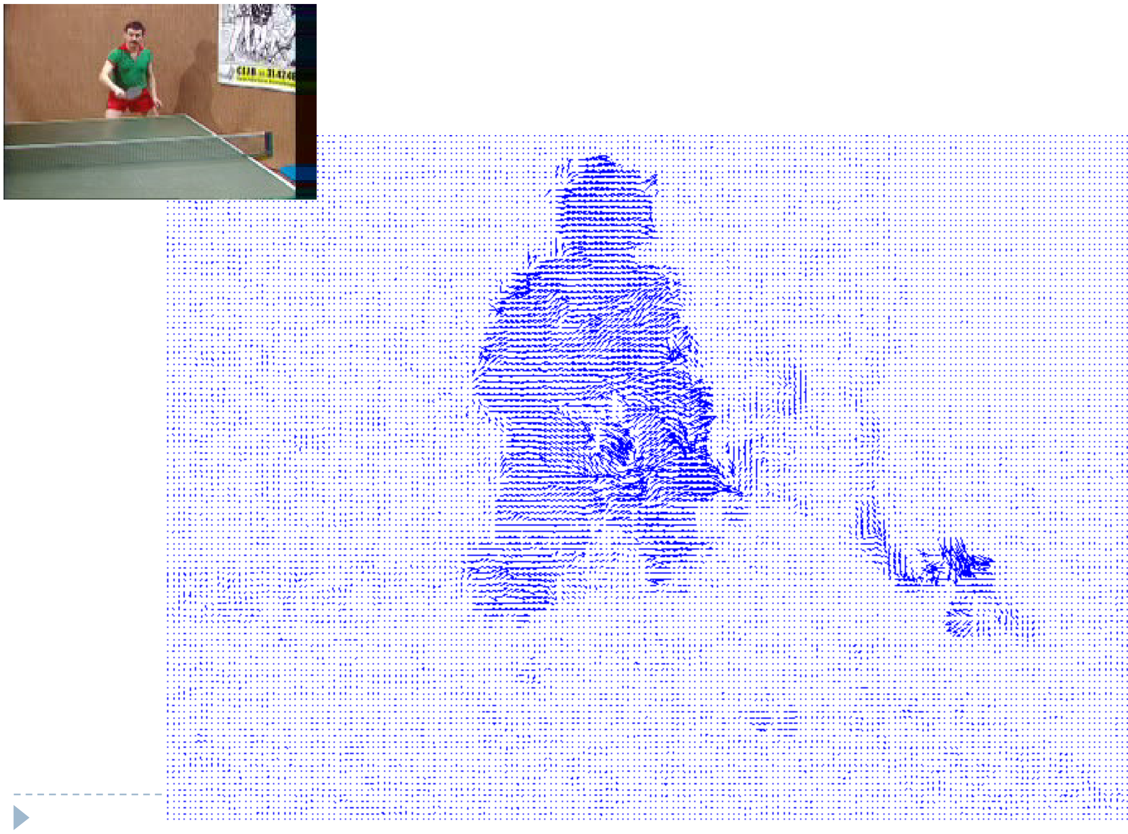
- Aliasing will occur on simple sub-sampling
- Follows from Nyquist Theorem
- 'Expand' can be substituted by Bilinear interpolation – filtering step would be discarded

Lucas Kanade with Pyramids

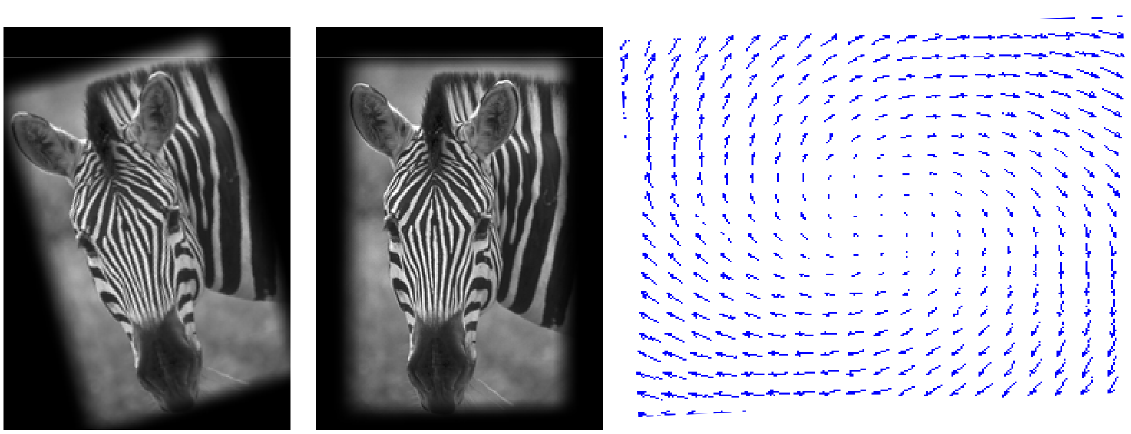
- Compute 'simple' LK at highest level
- At level *i*
  - Take flow u_{i-1}, v_{i-1} from level i-1
  - bilinear interpolate (or expand) it, multiply by 2 to create u_i^*, v_i^* matrices of twice resolution
  - compute f_t from a block displaced by $u_i^*(x,y), v_i^*(x,y)$
  - Apply LK to get $u_i'(x,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

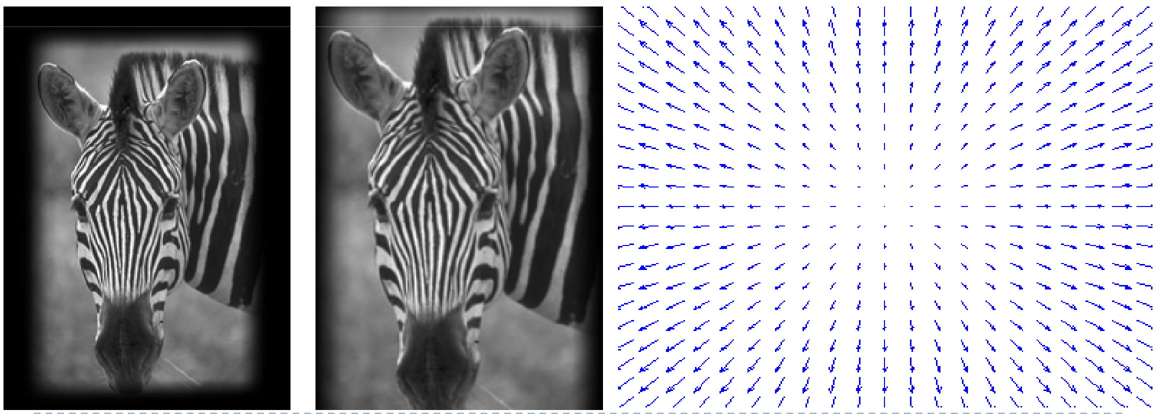






Global Flow



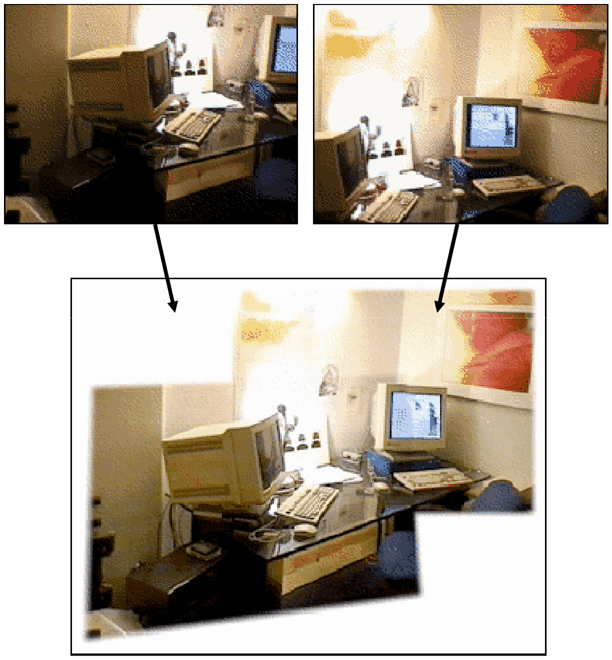


Global Flow

- Dominant Motion in the scene
  - Motion of all points in the scene
  - Motion of most of the points in the scene
  - A Component of motion of all points in the scene
- Global Motion is caused by
  - Motion of sensor (Ego Motion)
  - Motion of a rigid scene
- Estimation of Global Motion can be used to
  - Video Mosaics
  - Image Alignment (Registration)
  - Removing Camera Jitter
  - Tracking (By neglecting camera motion)
  - Video Segmentation etc.

Ref: Khurram Shafique, UCF

Global Flow



Application: Image Alignment

Computing Global Flow - Approach

- Brightness Constancy Equation is derived for a **single pixel**
- If we want to measure the motion of the **whole image**…
- We need to combine BCE with a **global displacement model**
  - Affine
  - Projective

Computing Global Flow

Assuming affine model for spatial displacement

$$x' = a_1 x + a_2 y + b_1$$

$$y' = a_3 x + a_4 y + b_2$$

What will be the global flow model?

$$x' - x = (a_1 - 1)x + a_2 y + b_1$$

$$y' - y = a_3 x + (a_4 - 1)y + b_2$$

$$\boxed{\begin{aligned} u &= a_1' x + a_2 y + b_1 \\ v &= a_3 x + a_4' y + b_2 \end{aligned}}$$

Computing Global Flow

$$\boxed{\begin{aligned} u &= a_1' x + a_2 y + b_1 \\ v &= a_3 x + a_4' y + b_2 \end{aligned}}$$

or

$$\begin{bmatrix} u \\ v \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} + B$$

Can be rearranged as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2 \end{bmatrix}$$

$$\mathbf{u} = \mathbf{Xa}$$

This is the global affine flow model. Now we need to combine this with the Brightness Constancy Constraint

Computing Global Flow

$$\mathbf{f}_{\mathbf{x}}^{T}\mathbf{u}+f_{t}=0$$

BCE

$$\mathbf{f}_{\mathbf{x}}^{T}\mathbf{X}\mathbf{a}+f_{t}=0$$

$$\mathbf{u}=\mathbf{X}\mathbf{a}$$

Global Affine Flow Model

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

Computing Global Flow

$$\mathbf{f}_{\mathbf{x}}^{T}\mathbf{X}\mathbf{a}+f_{t}=0$$

This term should be zero for every pixel in the image

Define error term as the sum of the square of the value of this term over the whole image

$$e=\sum_{\forall(x,y)\in I}\left(\mathbf{f}_{\mathbf{x}}^{T}\mathbf{X}\mathbf{a}+f_{t}\right)^{2}$$

Computing Global Flow

$$e = \sum_{\forall (x,y) \in I} \left(\mathbf{f}_{\mathbf{x}}^{T} \mathbf{X} \mathbf{a} + f_t \right)^2$$

If we choose the correct parameters **a**, this error should be minimum

$$\min \sum_{\forall (x,y) \in I} \left[\mathbf{f}_{\mathbf{x}}^{T} \mathbf{X} \mathbf{a} + f_t \right]^2 \Longrightarrow \left[\sum_{\forall (x,y) \in I} \mathbf{X}^T (\mathbf{f}_{\mathbf{x}})(\mathbf{f}_{\mathbf{x}})^T \mathbf{X} \right] \mathbf{a} = - \sum_{\forall (x,y) \in I} f_t \mathbf{X}^T \mathbf{f}_{\mathbf{x}}$$

$$\mathbf{A}\mathbf{a} = \mathbf{B}$$

Implementation

$$\left[\sum_{\forall (x,y) \in I} \mathbf{X}^T (\mathbf{f}_{\mathbf{x}})(\mathbf{f}_{\mathbf{x}})^T \mathbf{X} \right] \mathbf{a} = - \sum_{\forall (x,y) \in I} f_t \mathbf{X}^T \mathbf{f}_{\mathbf{x}}$$

- This equation should give answer in one step
- Practically, that frequently does not happen
- Why?
- BCE holds only for small motions, and so we need to iteratively refine our estimates using pyramids

Example

Image 1

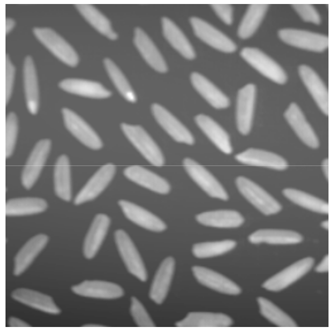
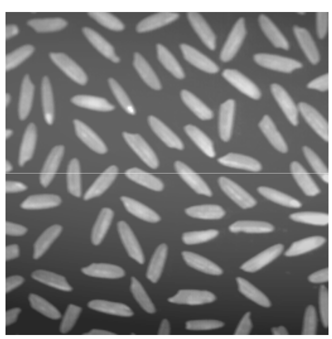
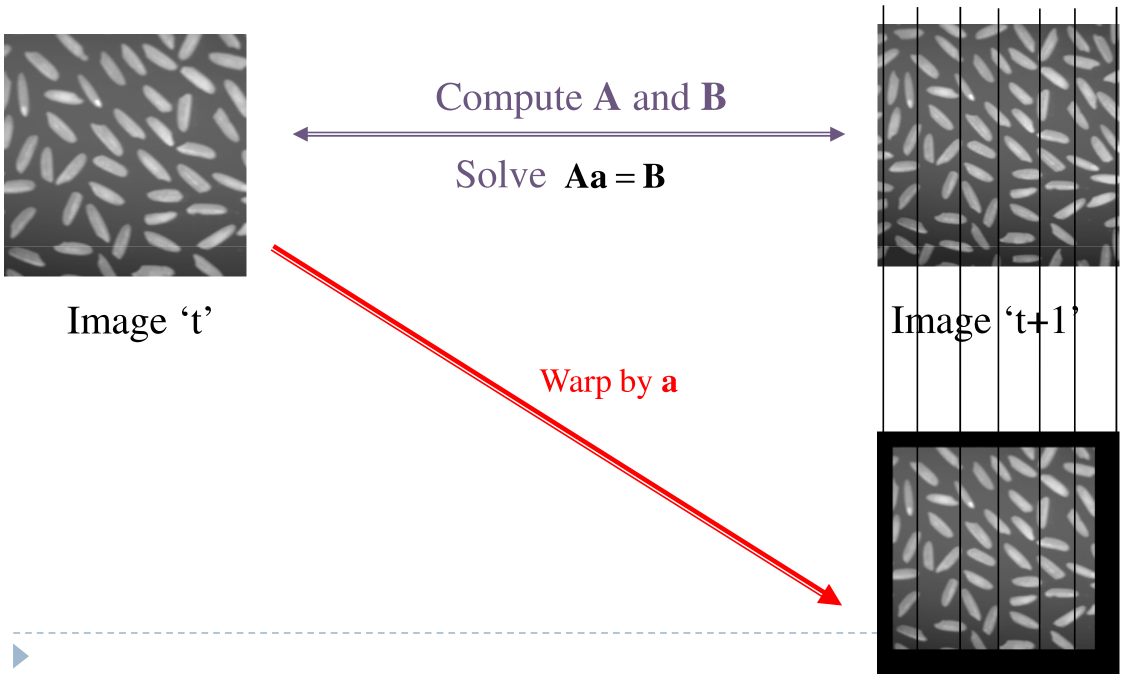


Image 2



Estimation of Global Flow

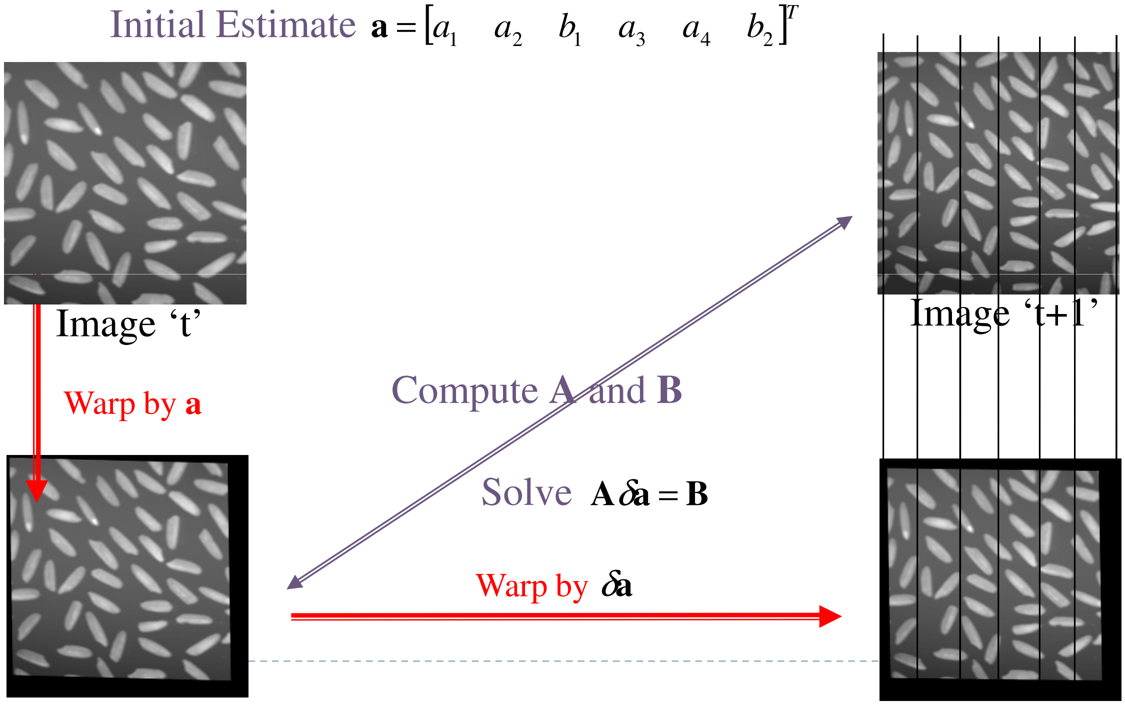
Single Iteration



Estimation of Global Flow

Iterative

Initial Estimate $\mathbf{a} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$



Estimation of Global Flow

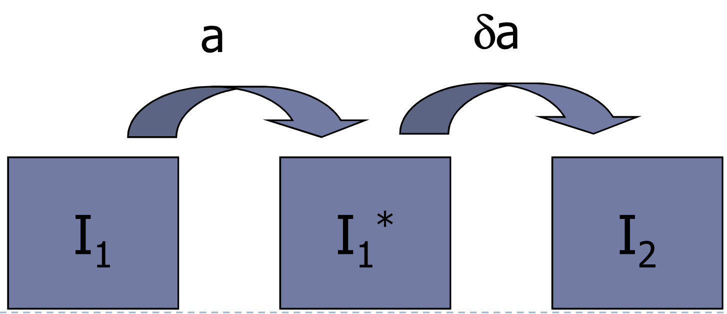
Parameters Update

Initial Estimate $\mathbf{a} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$

Computed Parameters $\delta\mathbf{a} = [\delta a_1 \quad \delta a_2 \quad \delta b_1 \quad \delta a_3 \quad \delta a_4 \quad \delta b_2]^T$

Update

Should have the combined effect of transformations



Estimation of Global Flow

Iterative

Initial Estimate $\mathbf{a} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$

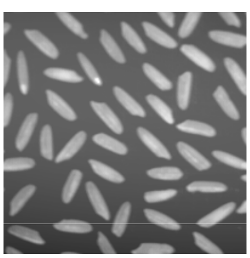


Image ‘t’

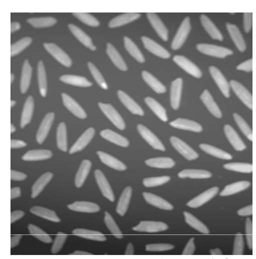
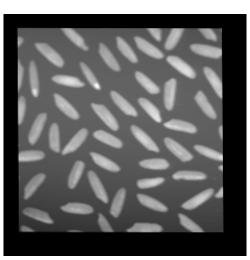
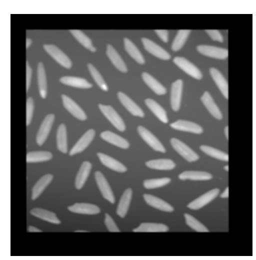


Image ‘t+1’



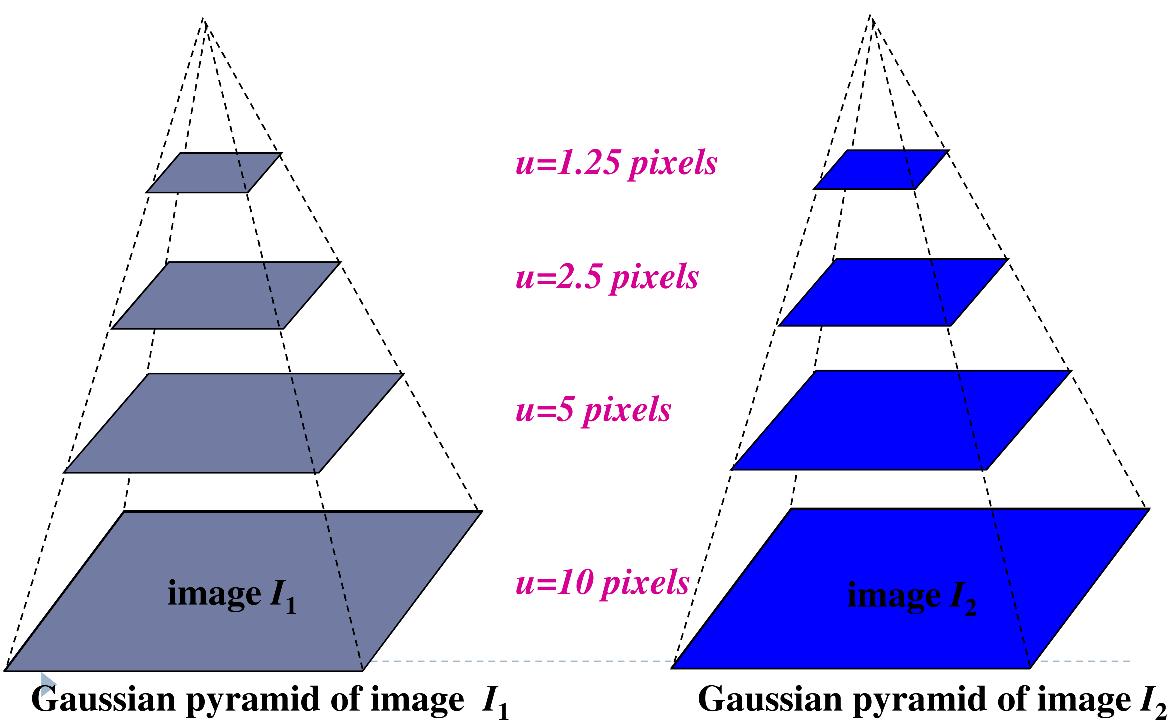
1.2000	-0.0000	-19.9997
-0.0000	1.2000	-19.9986



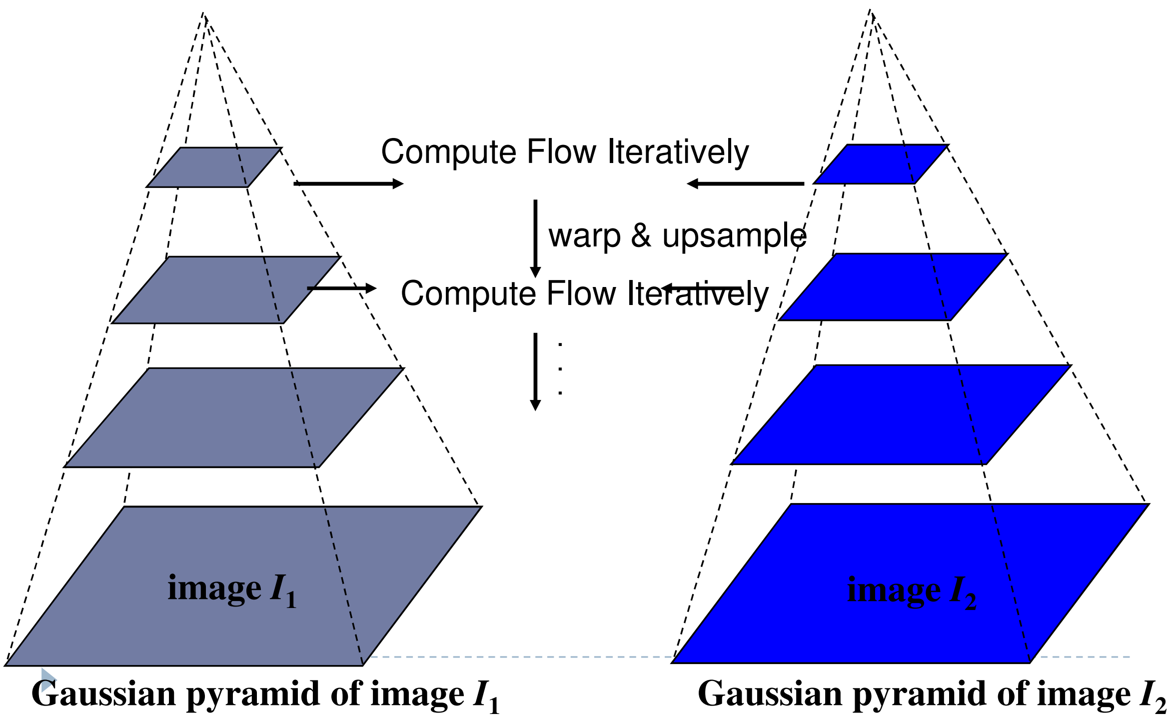
Iterative Refinement

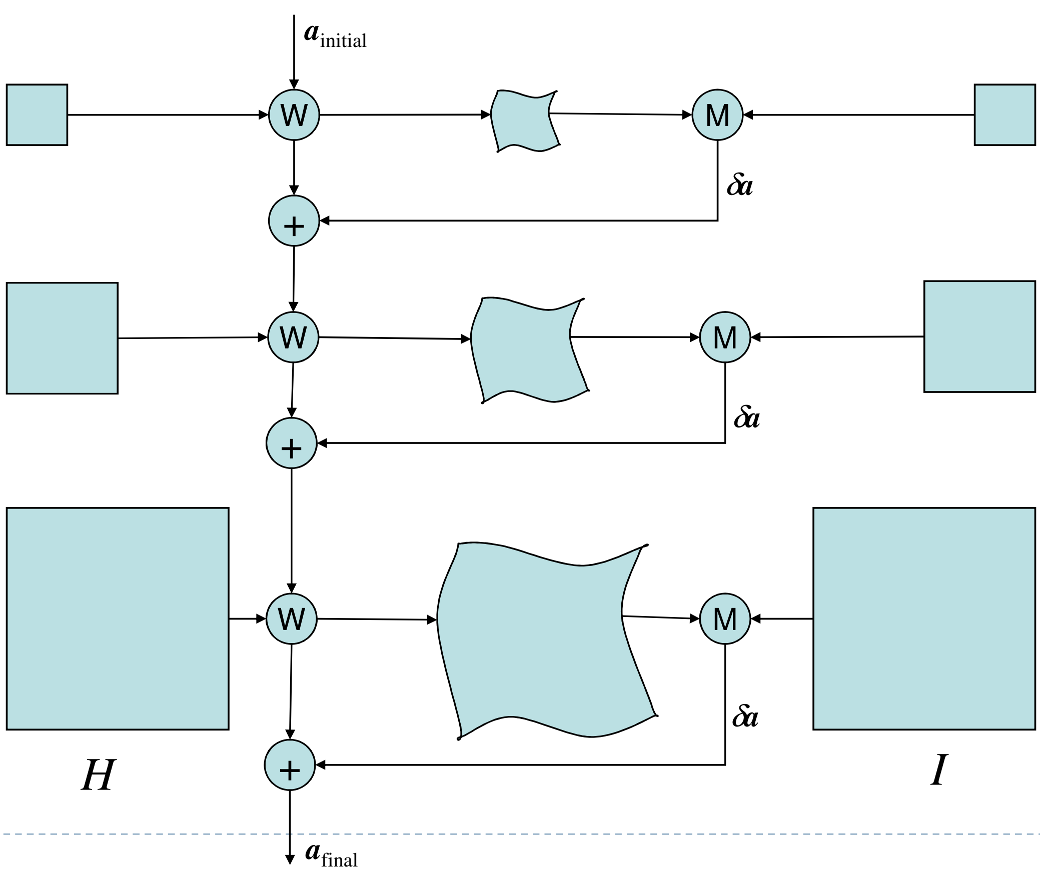
- Basic Idea
  1. Estimate global flow parameters **a** by solving the linear system of the form **Aa=B**
  2. Use **warping** to warp I_1 towards I_2 using the estimated parameters
  3. Move to next level of pyramid

Coarse-to-fine global flow estimation



Coarse-to-fine global flow estimation

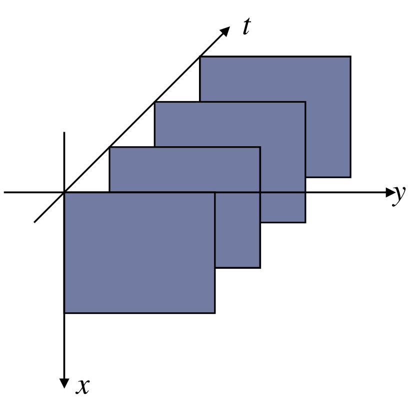




Implementation: Basic Components

- Pyramid Construction
- Computation of Derivatives
- Motion Estimation
- Update of Parameters
- Image Warping
- Coarse-to-fine refinement
- Generating Output by Blending

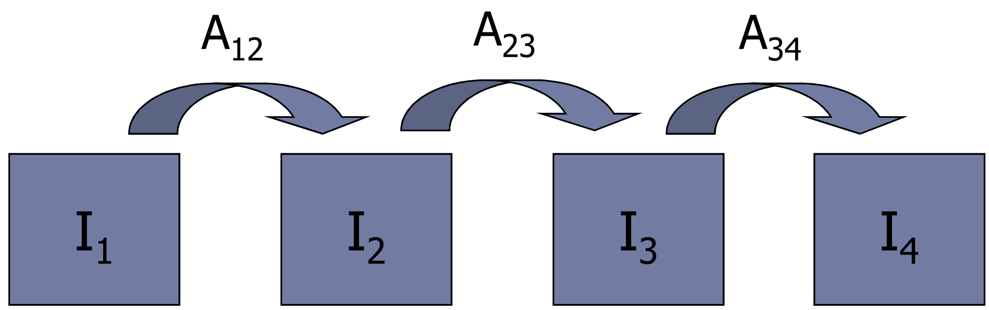
Implementation - Derivatives



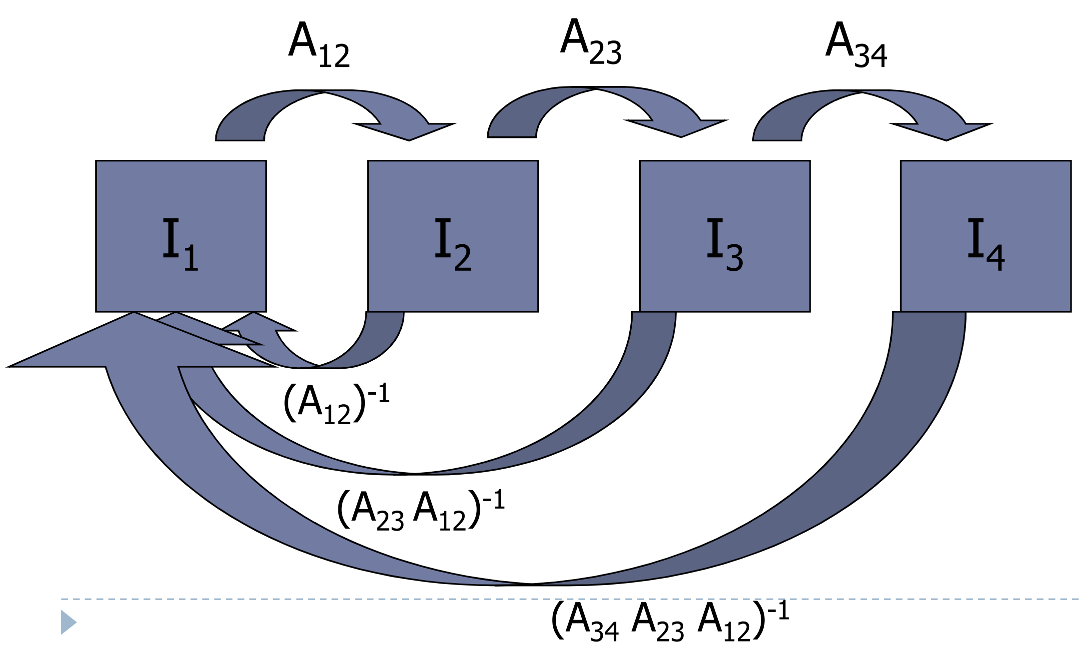
	f_x		f_y		f_t	
I_t	-1	-1	-1	1	-1	-1
	1	1	-1	1	-1	-1
I_{t+1}	-1	-1	-1	1	1	1
	1	1	-1	1	1	1

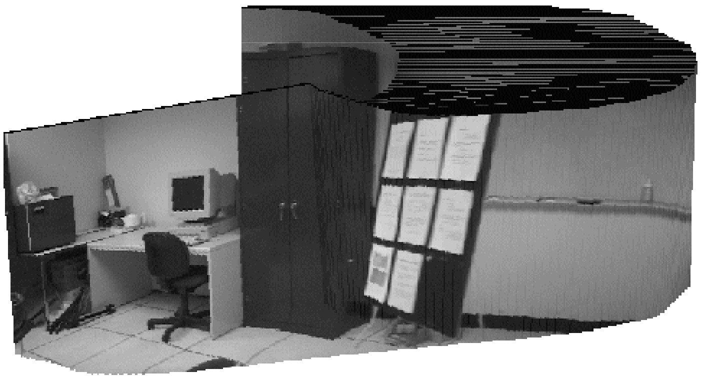
Video Mosaic

- Recover parameters between each pair of images

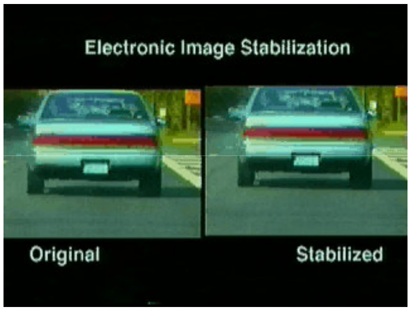


Video Mosaic

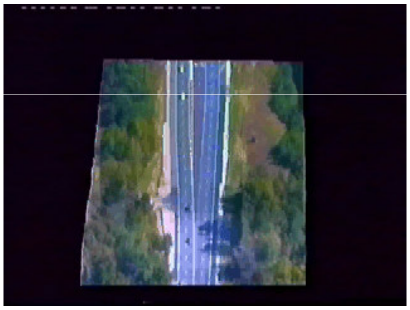




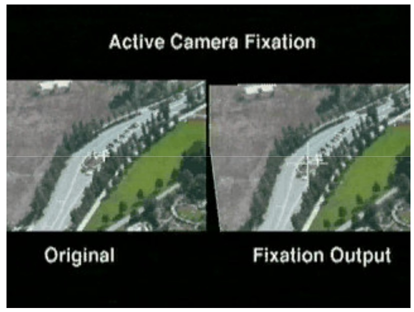
Credit: Sarnoff Corp



Credit: Sarnoff Corp



Credit: Sarnoff Corp



Credit: Sarnoff Corp

Algorithm

- Global Affine Alignment Algo
  - Given two images I_1 I_2
  - Make Pyramid by **REDUCE** operation
  - **Warp** I_1 to I_1^* using current **a**
  - Apply Global Affine equation on I_1^* and I_2 to get correction d**a**
  - **Combine** d**a** and **a** to get better new **a**
  - Repeat at each level several times, proceed down the pyramid

Blending

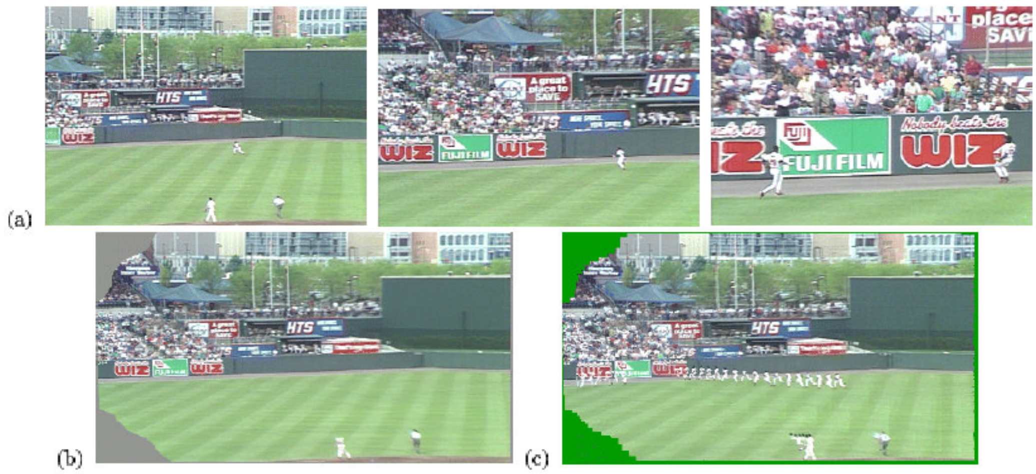


Fig. 2. Visual summaries of a baseball video clip.
(a) A few representative frames from the video clip. The video shows two outfielders running, while the camera is panning to the left and zooming on the two baseball players. (b) The static background mosaic image which provides an extended view of the entire scene captured by the camera in the video clip. The "missing" regions at the top-left and bottom-left were never imaged by the camera, because at that point it was zoomed on the two players (e.g., frame 80). (c) The synopsis mosaic which provides a visual summary of the entire event. It shows the trajectories of the two outfielders in the context of the mosaic image.

Michal Irani, P. Anandan, "Video Indexing Based on Mosaic Representations"

Hierarchical Dominant Motion Estimation

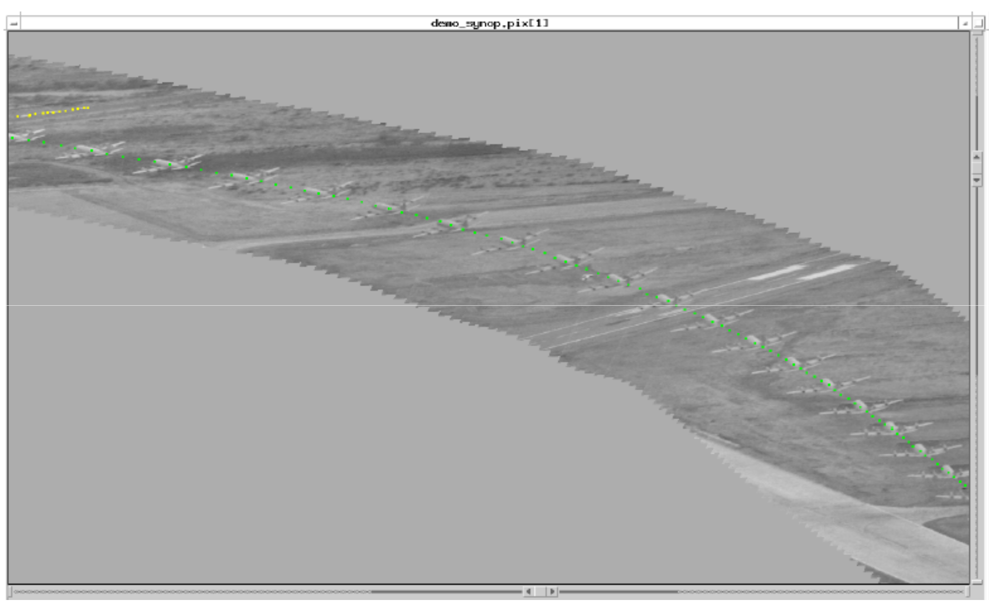
- If there are multiple motions in an image
- Eg, the player sequence in the previous slide
- We can track objects of interest by analyzing global motion

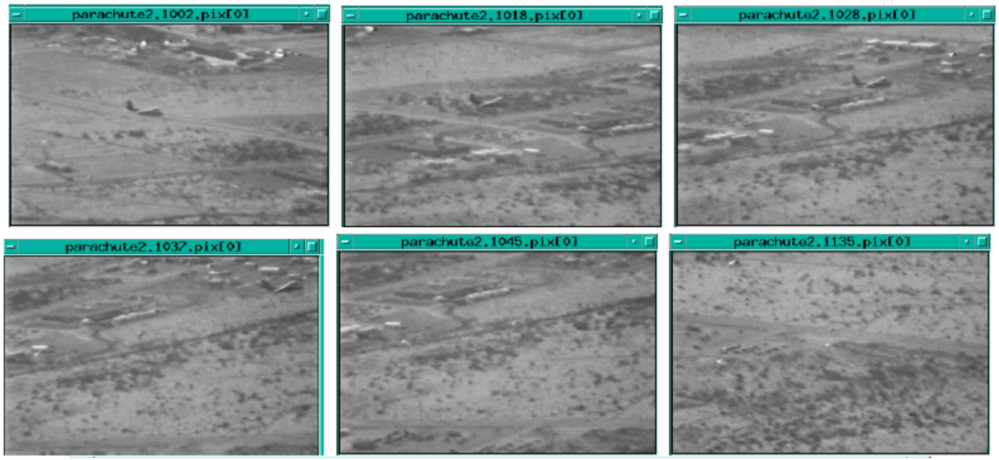
Hierarchical Dominant Motion Estimation

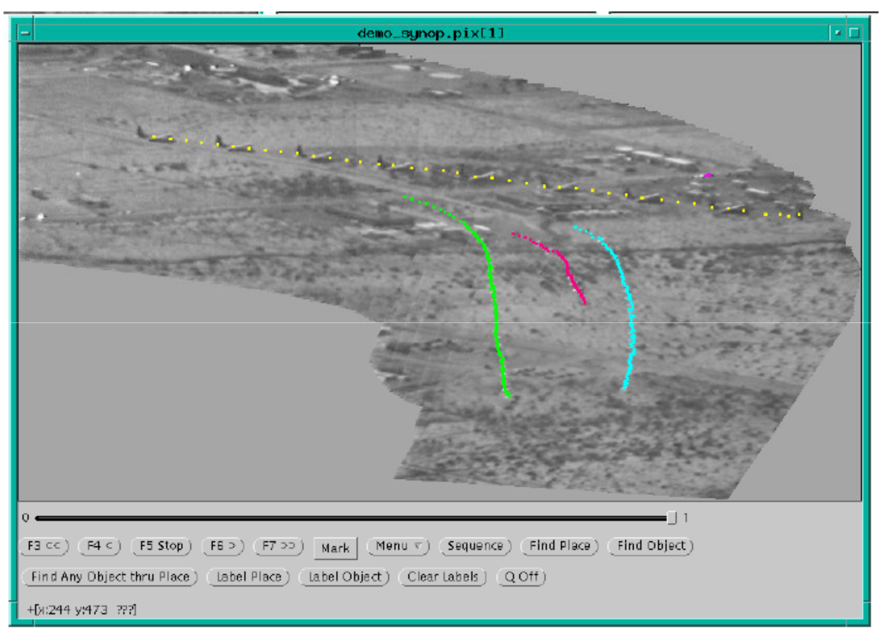
- Compute Dominant Motion
  - Use standard formulation just described
- Compensate for that motion, so that the dominant motion appears as stationary
- Now compute the next dominant motion
- This will be the object of interest
- Can be repeated if there are multiple motions within the object of interest
  - Eg the table tennis player

http://www.irisa.fr/vista/Themes/Logiciel/Motion-2D/Motion-2D.english.html


airplane2.1099.pix[0]
airplane2.1109.pix[0]
airplane2.1146.pix[0]
airplane2.1174.pix[0]
airplane2.1196.pix[0]
airplane2.1204.pix[0]


demo_synop.pix[1]





Michal Irani, P. Anandan, "Video Indexing Based on Mosaic Representations"