Optic Flow and Motion Detection

Computer Vision

Martin Jagersand

Readings:

3DV Ma, Kosecka, Sastry Ch 4.3

Szeliski Ch 5

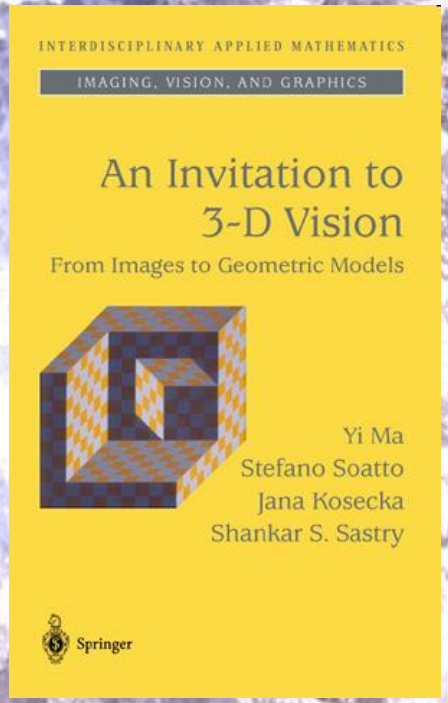
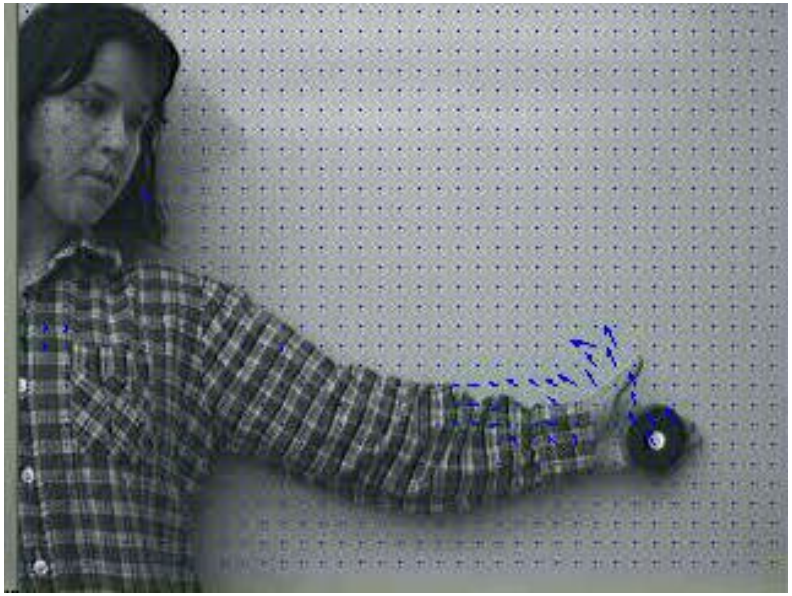


Image motion

- Somehow quantify the frame-to-frame differences in image sequences.

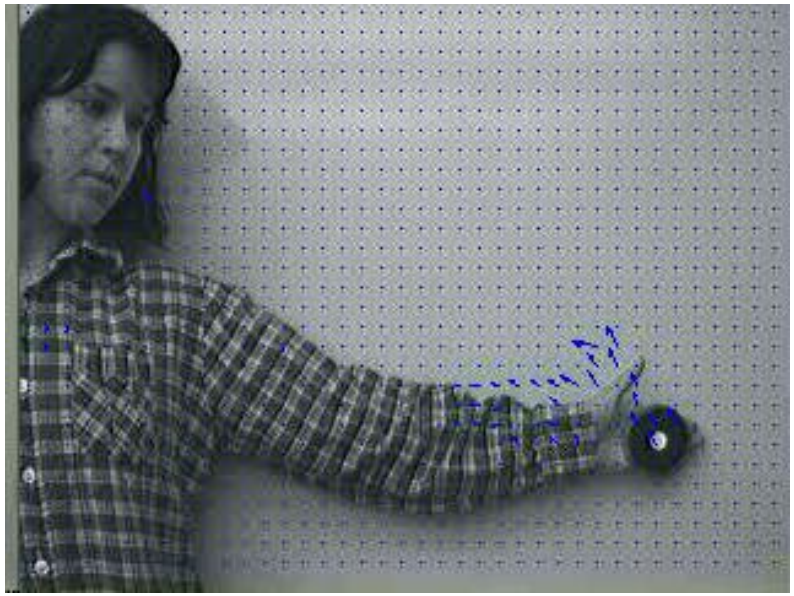


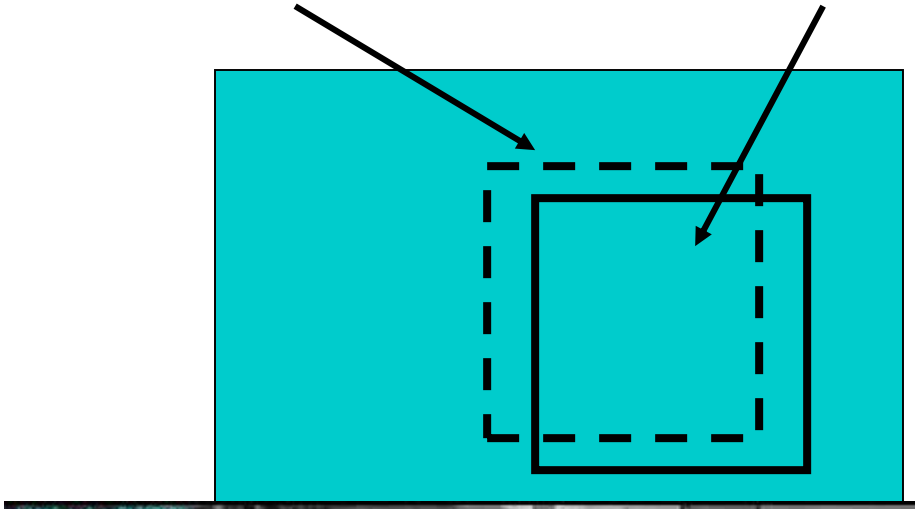
1. Image intensity difference.
2. Vector motion=optic flow
3. When computable?
4. Numerical conditioning!
5. Resolution pyramids
6. 3-6 dim image motion

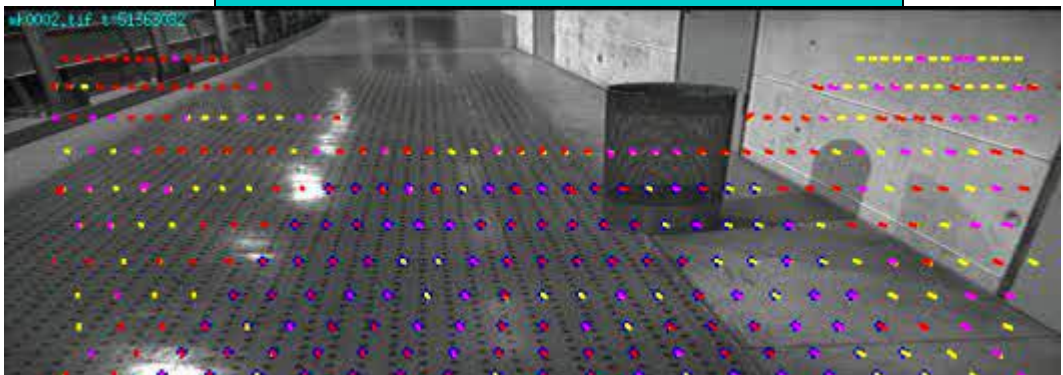
Low level video processing “Visual motion detecton / optic flow”

- Relating two adjacent frames: (small differences):

$$\mathrm{Im}(x + \delta x,\ y + \delta y,\ t + \delta t) = \mathrm{Im}(x,\ y,\ t)$$

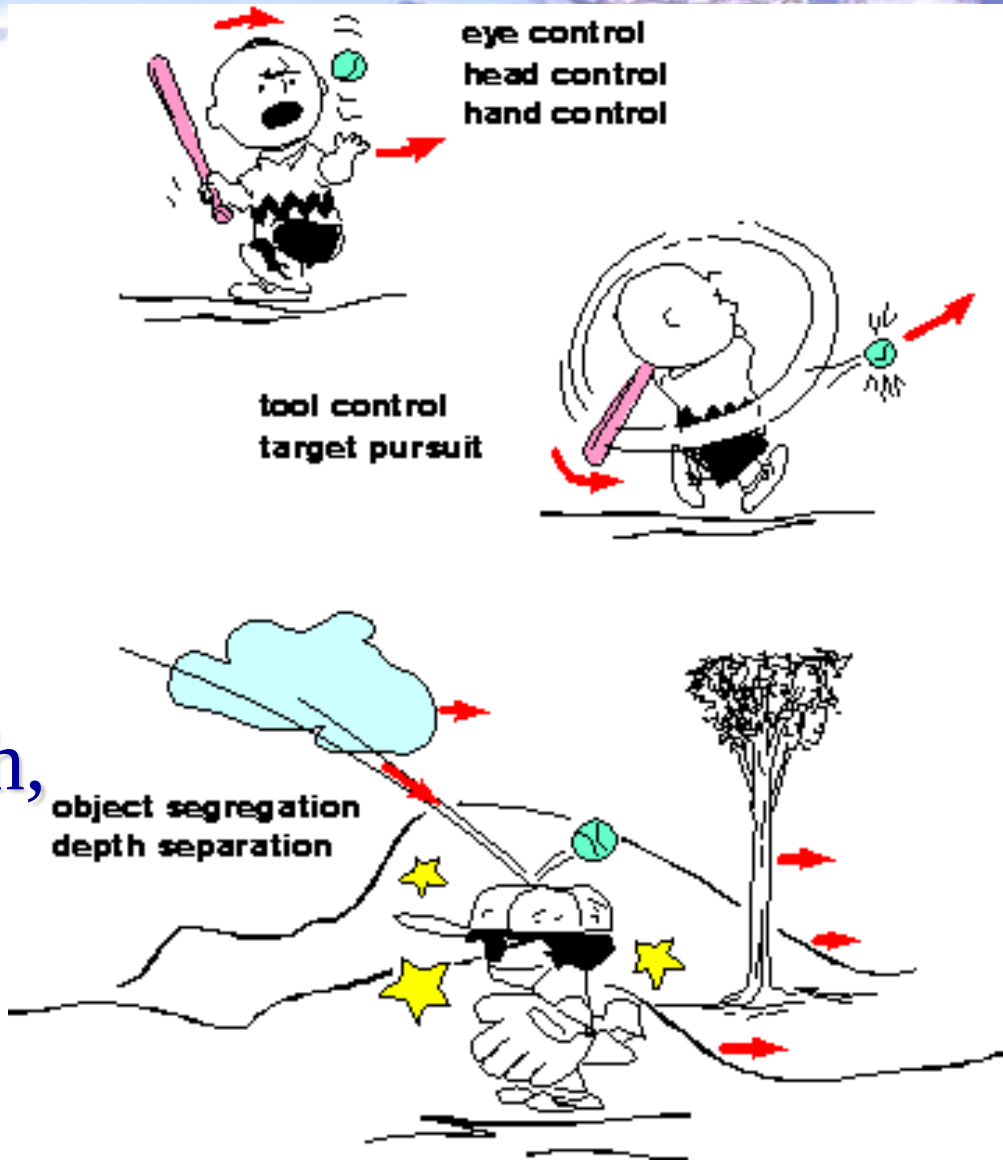




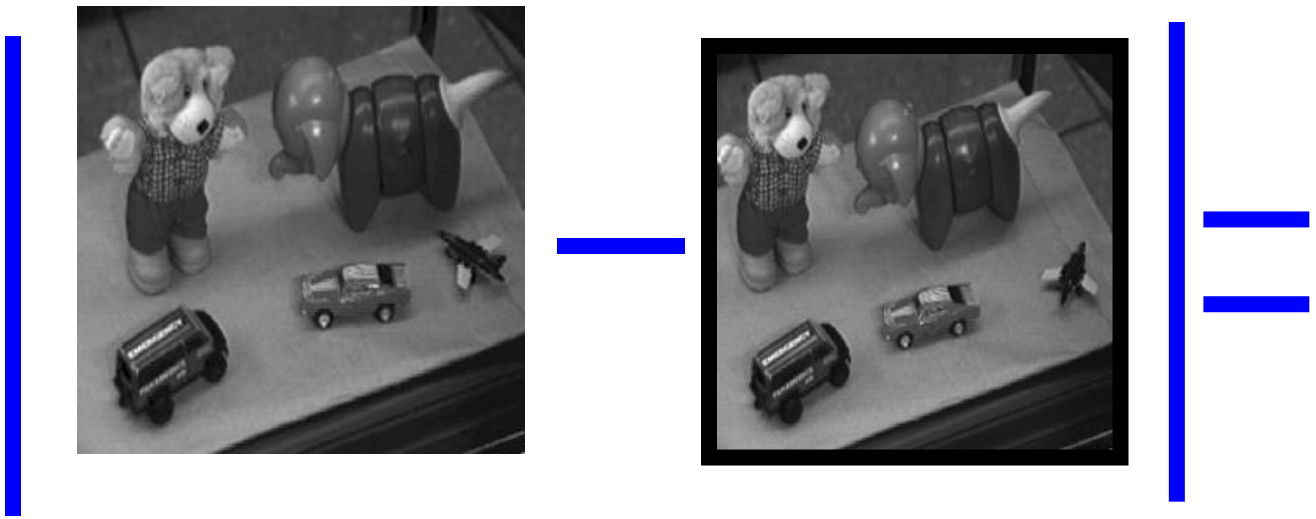


Motion is used to:

- **Attention:** Detect and direct using eye and head motions
- **Control:** Locomotion, manipulation, tools
- **Vision:** Segment, depth, trajectory



Small camera re-orientation



Note: Almost all pixels change!

Classes of motion

- Still camera, single moving object
- Still camera, several moving objects
- Moving cam
- Moving cam

Fixed video camera

- Background subtraction
  - A static camera is observing a scene
  - Goal: separate the static *background* from the moving *foreground*

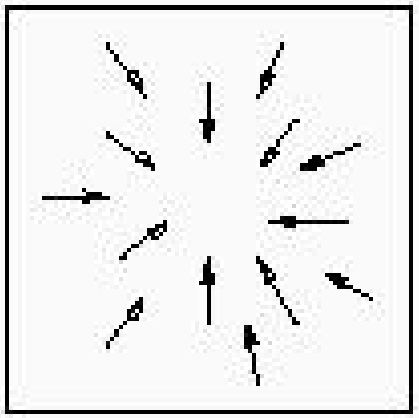
How to come up with background frame estimate without access to "empty" scene?

The optic flow field

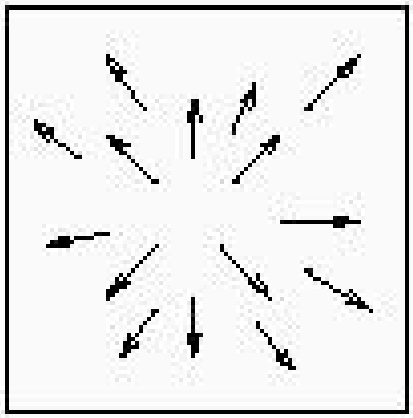
- Vector field over the image:

  [u,v] = f(x,y), u,v = Vel vector, x,y = Im pos

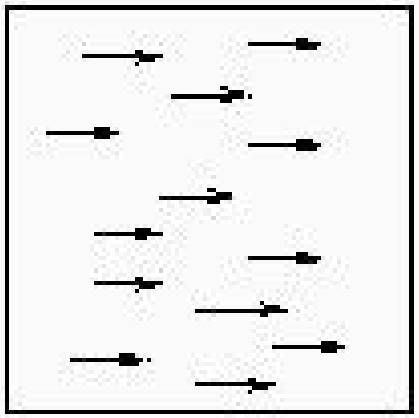
- **FOE, FOC** Focus of Expansion, Contraction



Zoom out



Zoom in



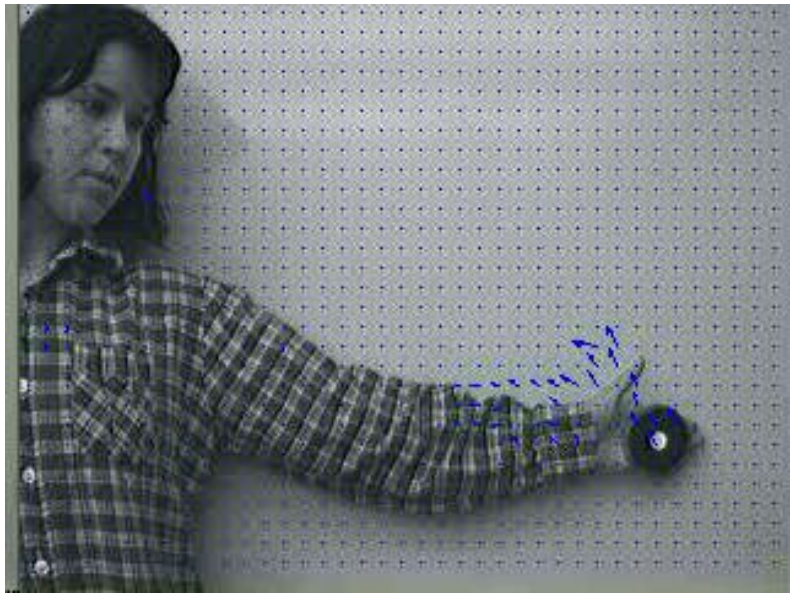
Pan Right to Left

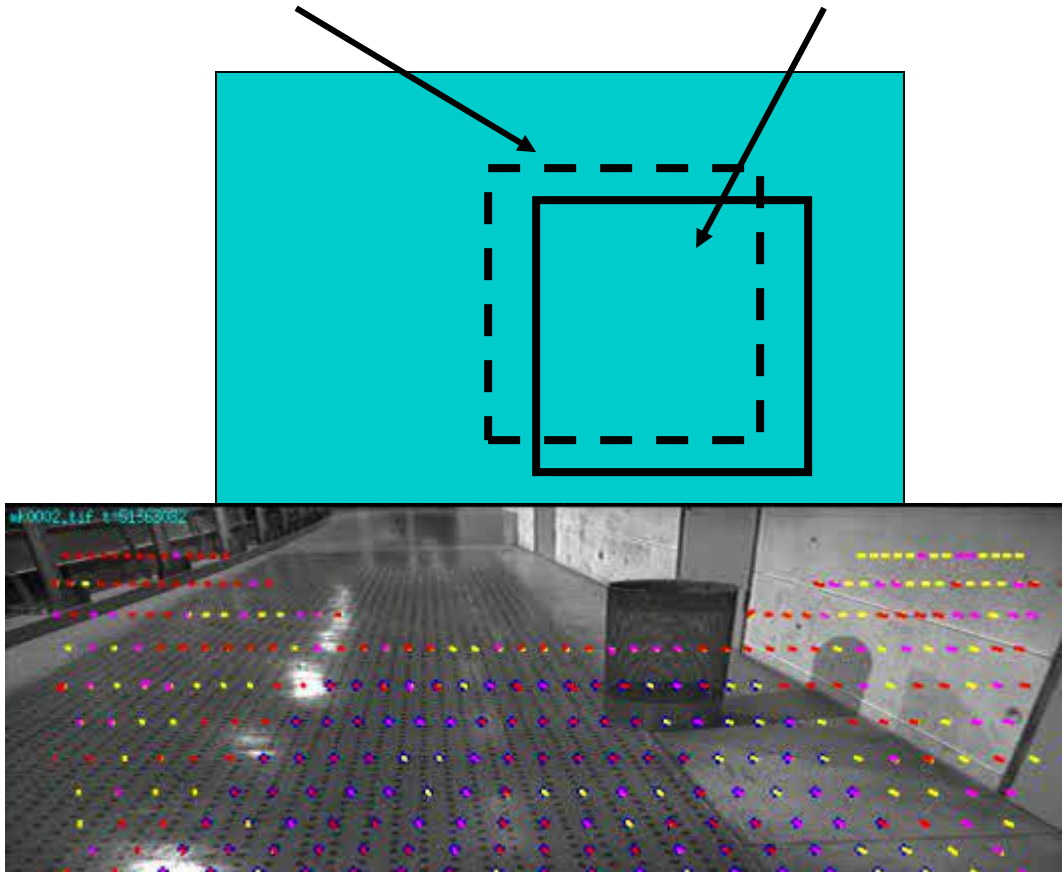
optic flow field

[u,v] = f(x,y), u,v = Vel vector, x,y = Im pos

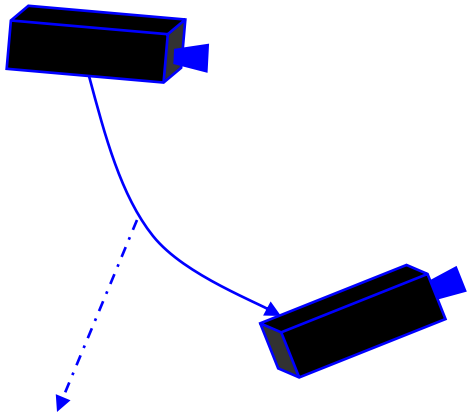
- Relating two adjacent frames: (small differences):

$$\mathrm{Im}(x + u,\ y + v,\ t + \delta t) = \mathrm{Im}(x,\ y,\ t)$$

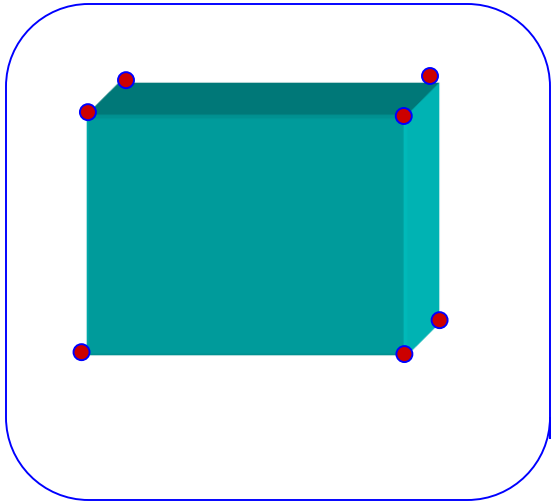




Correspondance for a box

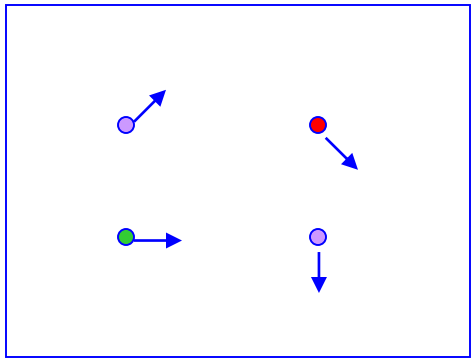


The change in spatial location
between the two cameras (the “motion”)

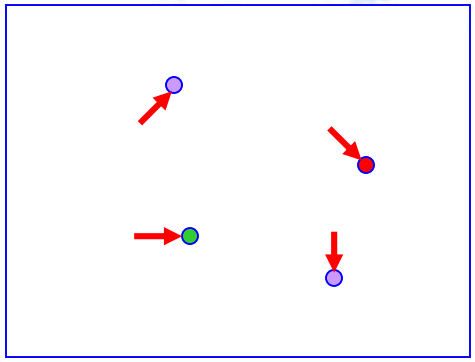


Locations of
points on the object
(the “structure”)

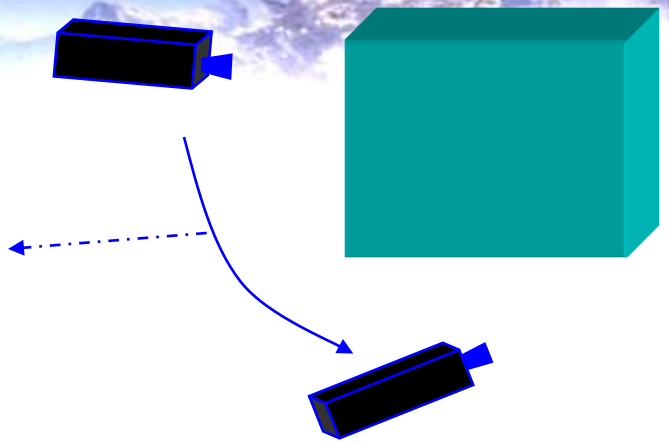
Motion/Optic flow vectors
How to compute?



$\mathrm{Im}(x,\ y,\ t)$



$\mathrm{Im}(x+\delta x,\ y+\delta y,\ t+\delta t)$



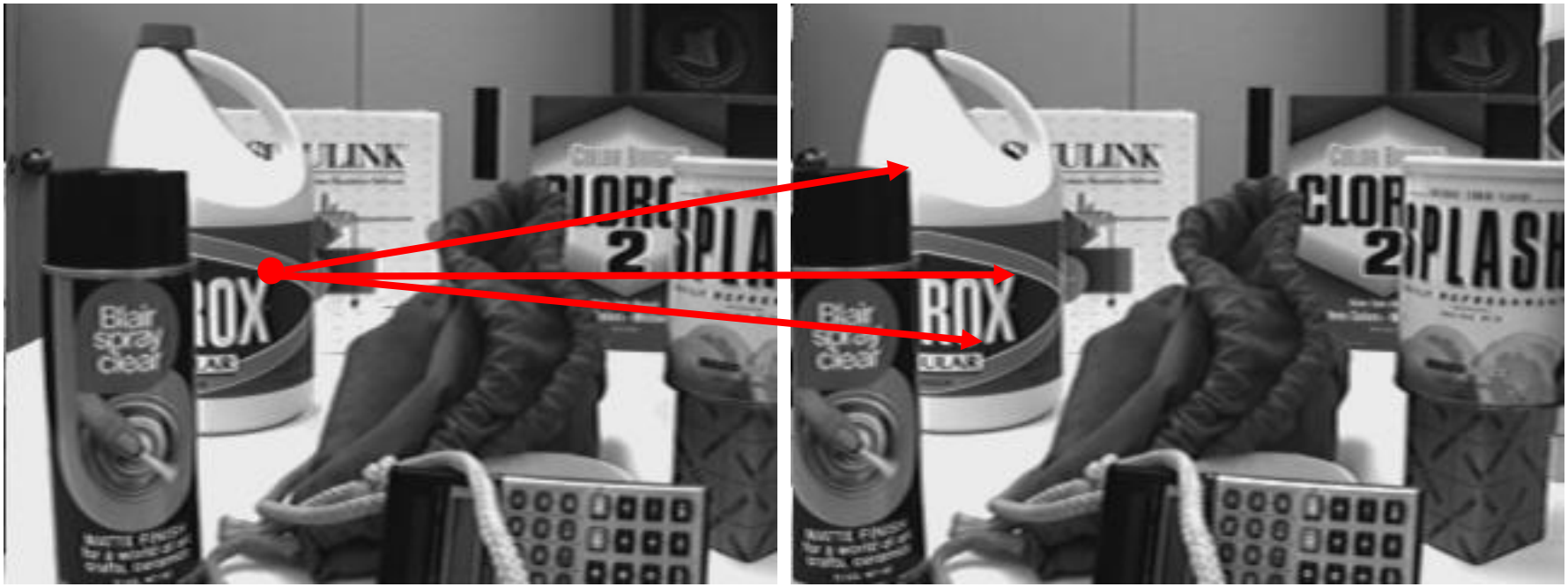
– Solve pixel correspondence problem

– given a pixel in Im1, look for same pixels in Im2

- Possible assumptions

1. **color constancy:** a patch in I(t) looks the same in I(t)
   – For grayscale images, this is **brightness constancy**
2. **small motion**: points do not move very far
   – This is called the **optical flow** problem

What pixels/patches correspond?



Time **t** **t+1**

(Correspondence between image points – a common challenge:
Optic flow, Tracking, Features)

Image correspondence: Three assumptions

- Brightness consistency
- Spatial coherence
- Temporal persistence

Brightness consistency

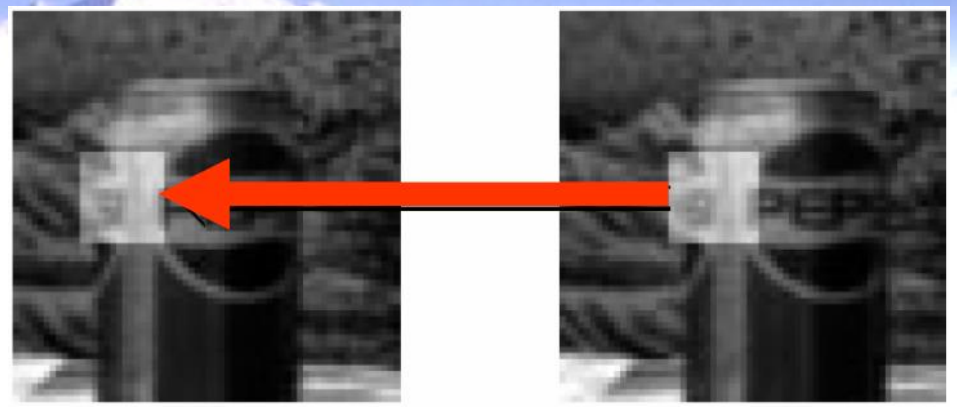
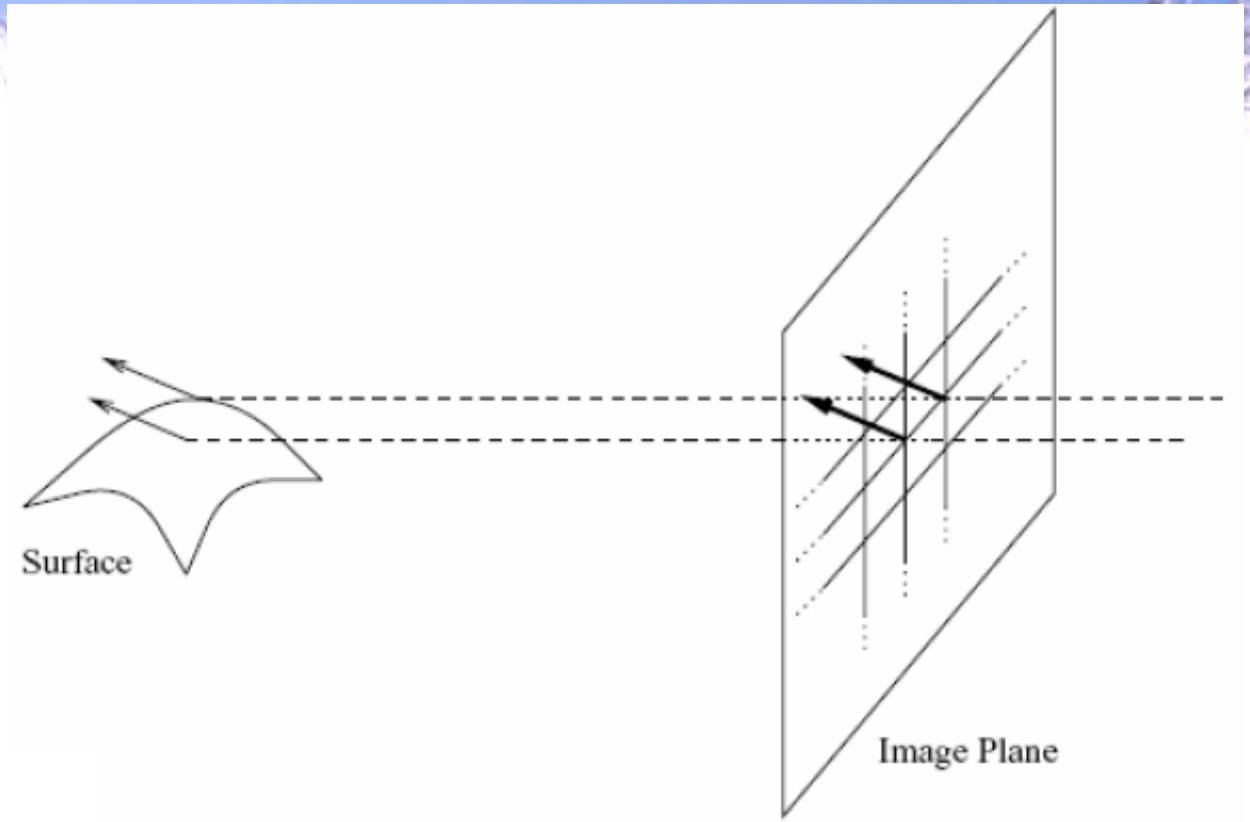


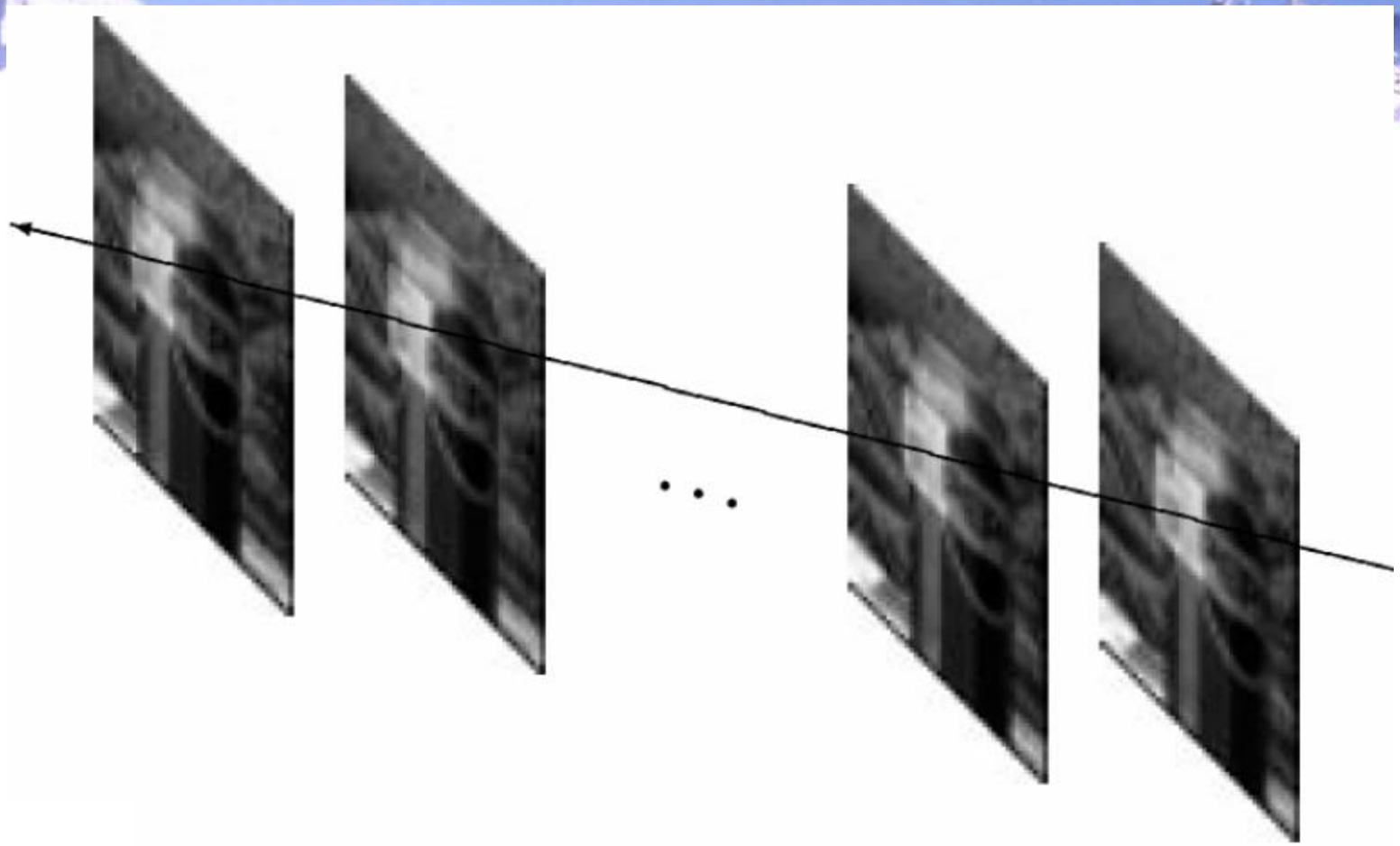
Image measurement (e.g. brightness) in a small physical surface region remain the same although their image location may change.

Spatial coherence



- Neighboring points in the scene typically **belong to the same surface** and hence typically have similar motions.
- Since they also project to nearby pixels in the image, we expect spatial coherence in image flow.

Temporal persistence

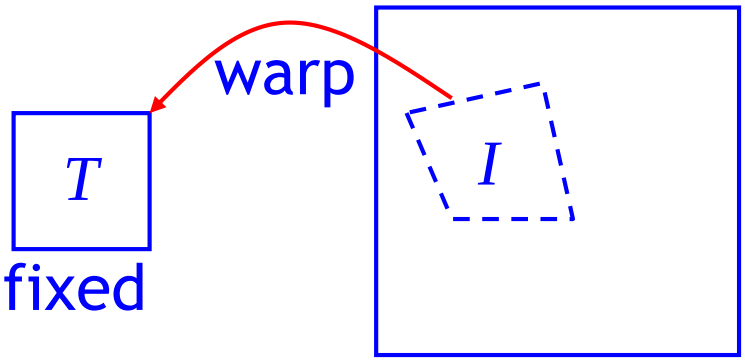


The image motion of a surface patch changes gradually over time – no more than 2-5pixels/frame

Image registration: Three applications

Goal: register a template image $T(x)$ and an input image $I(x)$, where $x=(x,y)^T$. (warp I so that it matches T)

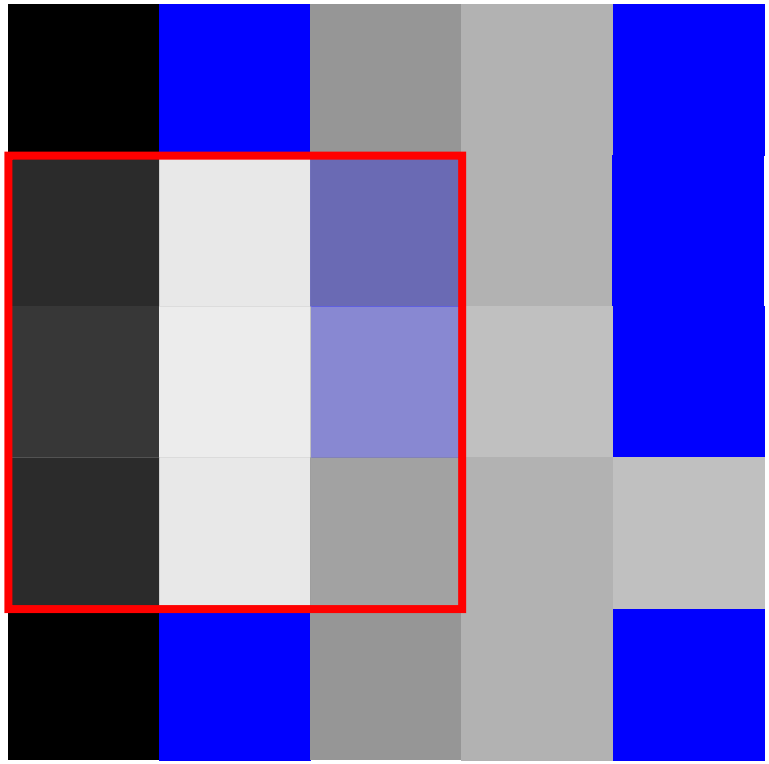
1. Image alignment: $I(x)$ and $T(x)$ are two images
2. Tracking: $T(x)$ is a small patch around a point p in the first video image, $t=0$. $I(x)$ is the image at time $t+1$.
3. Optical flow: $T(x)$ and $I(x)$ are patches of images at t and $t+1$.

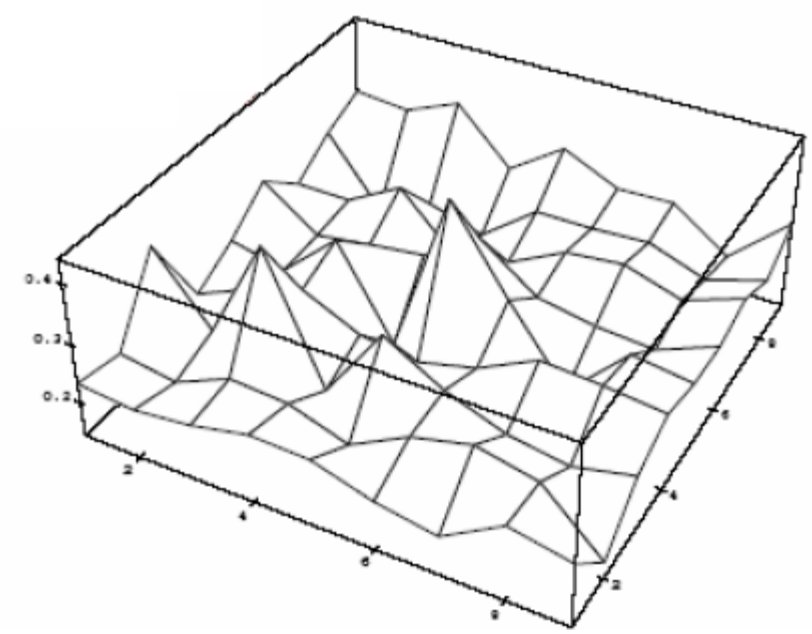


Simple approach (for translation)

- Minimize brightness difference

$$E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - T(x,y) \right)^2$$





Simple image registration algorithm
SSD error norm

For each offset *(u, v)*

compute *E(u,v);*

$$E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - T(x,y) \right)^2$$

Choose *(u, v)* which minimizes *E(u,v);*

Problems:

- Not efficient
- No sub-pixel accuracy

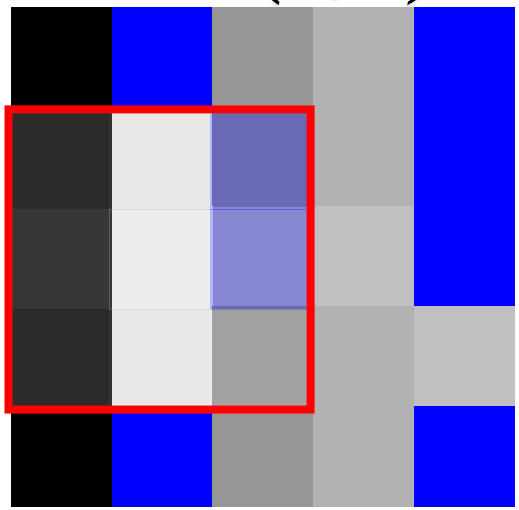
Simple image registration algorithm
SSD error norm

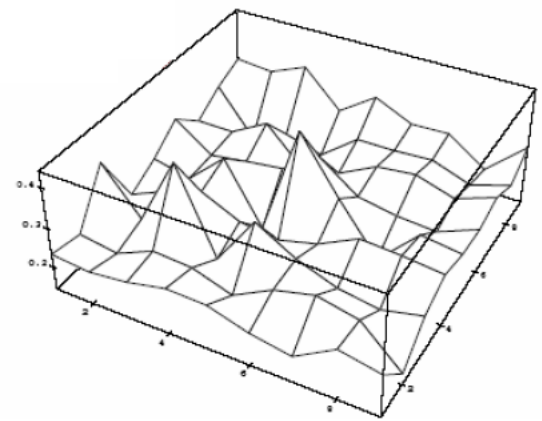
$$E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - T(x,y) \right)^2$$

For each offset *(u, v)*

compute *E(u,v);*

Choose *(u, v)* which minimizes *E(u,v);*





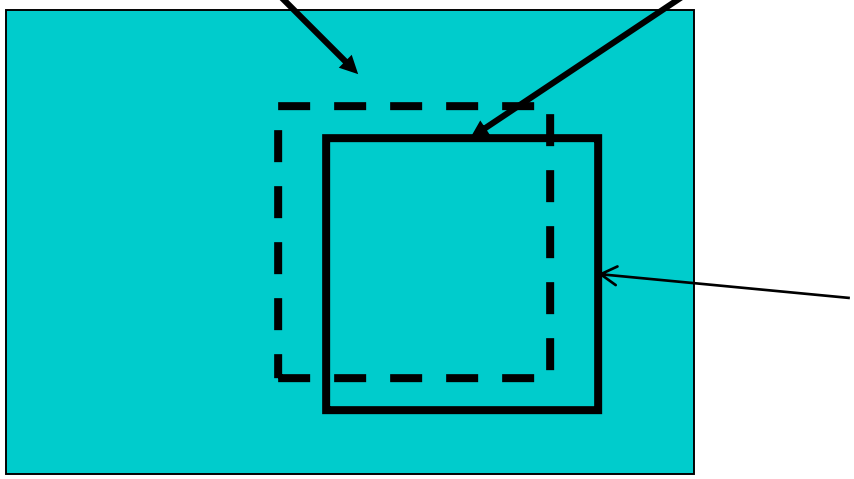
Optic/image flow

Assume:

1. Image intensities from object points remain constant over time
2. Image displacement/motion small

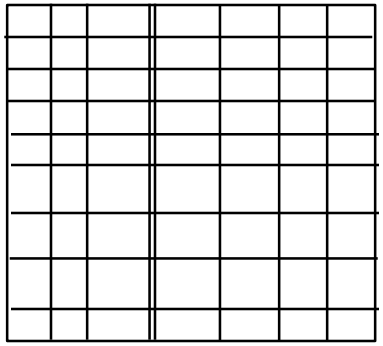
$$\mathrm{Im}(x + \delta x,\ y + \delta y,\ t + \delta t) = \mathrm{Im}(x,\ y,\ t)$$

Brightness consistency



Small rectange of image

(Later: Subdivide image)



Taylor expansion of intensity variation

Taylor expansion:

$$\mathrm{Im}(x+\delta x, y+\delta y, t+\delta t) = \mathrm{Im}(x,y,t) + \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t + h.o.t.$$

Keep linear terms

$$(\exists \delta x, \delta y) s.t. \;\; \mathrm{Im}(x+\delta x, \;\; y+\delta y, \;\; t+\delta t) \;\; = \;\; \mathrm{Im}(x, \;\; y, \;\; t)$$

← Brightness consistency

- Use consistency assumption and rewrite:

$$0 = \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t$$

- Notice: Linear constraint, but no unique solution

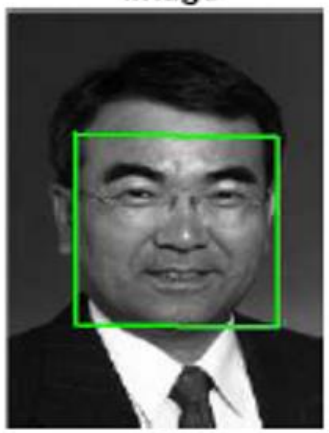
What are the partial derivatives?

$$0 = \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t$$

What are the partial derivatives?

$$0 = \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t$$

Image



$I(\mathbf{x})$

Image Gradient X

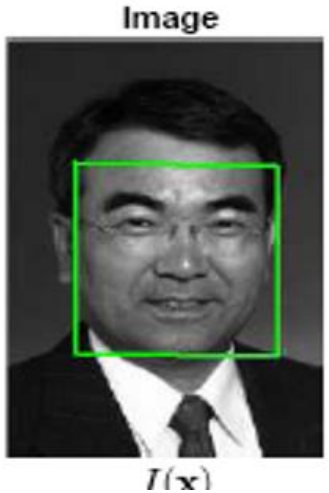
Image Gradient Y

What are the partial derivatives?

$$0 = \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t$$

dIm_t = I(t+1)-I(t)

Heath Ch 9

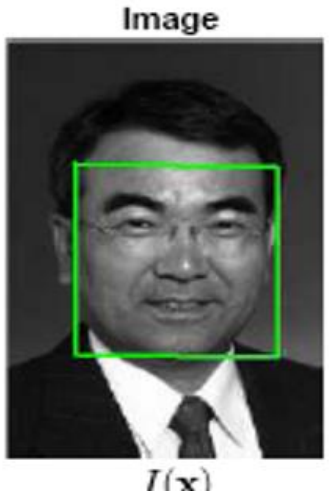


What are the partial derivatives?

$$0 = \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t$$

dIm_t = I(t+1)-I(t)

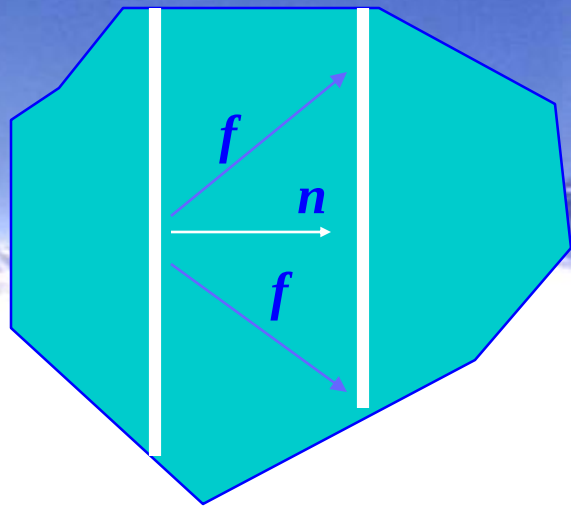
Heath Ch 9



dIm_x = I(x+1,y)-I(x,y)

Matlab: dIm_x = I(:,2:n)-I(:,1:n-1)

Solving for optic flow



- Rewrite as dot product

$$-\frac{\partial \mathrm{Im}}{\partial t}\delta t = \left(\frac{\partial \mathrm{Im}}{\partial x}, \frac{\partial \mathrm{Im}}{\partial y}\right) \cdot \begin{pmatrix}\delta x \\ \delta y\end{pmatrix} = \nabla \mathrm{Im} \cdot \begin{pmatrix}\delta x \\ \delta y\end{pmatrix}$$

- Each pixel gives one equation in two unknowns:

k = n*f

Image spatial gradient normal ***n:*** $\nabla \mathcal{I}$m, later: M

The image motion / optic flow ***f*** = $(\delta x \psi \delta y)^{\mathrm{T}}$, later u

Image temporal gradient k: ∂Im/∂t, later dIm

- Min length solution: Can only detect vectors normal to gradient direction
- The motion of a line cannot be recovered using only local information

Flatten 2D images into vectors

Equation for one pixel:

$$-\frac{\partial \mathrm{Im}}{\partial t}\delta t = \left(\frac{\partial \mathrm{Im}}{\partial x}, \frac{\partial \mathrm{Im}}{\partial y}\right) \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = \nabla \mathrm{Im} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

Equations for many pixels:

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

Solve for optic flow using several simultaneous equations

- Typically solve for motion in 2x2, 4x4, 8x8 or larger image patches.
- Over determined equation system:

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

dIm = M*u

- Can be solved in least squares sense using Matlab

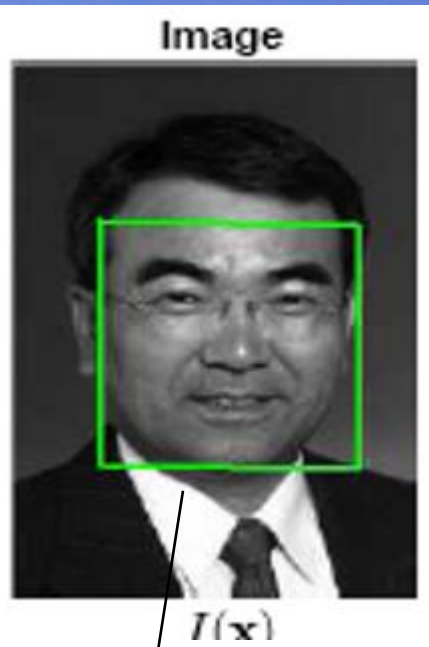
u = M\dIm

- Can also be expressed using QR factorization:

$(Q^T M)u = Q^T dIm$ [Q, R] = qr(M), $Q^T M=R$

(Don't compute $u = (M^T M)^{-1} * M^T dIm$ Ill conditioned – see 340)

Intuitive connection images - equations



I(t+1)-I(t);

=

∇I_x

$\delta x +$

∇I_y

δy

Template

$T(\mathbf{x})$

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

Summary: Solve for optic flow using several simultaneous equations

- Taylor expansion for each patch:

$$\mathrm{Im}(x+\delta x, y+\delta y, t+\delta t) = \mathrm{Im}(x,y,t) + \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t + h.o.t.$$

$$(\exists \delta x, \delta y) s.t.\ \ \mathrm{Im}(x+\delta x,\ \ y+\delta y,\ \ t+\delta t) \ \ = \ \ \mathrm{Im}(x,\ \ y,\ \ t)$$

← Brightness consistency

$$0 = \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t$$

- Over determined equation system:

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

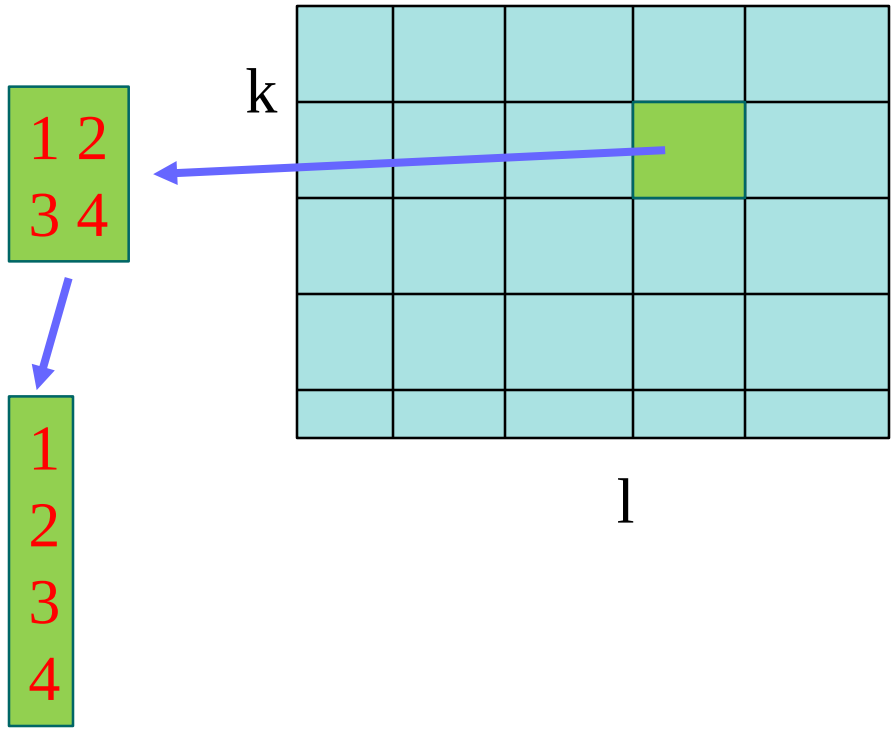
dIm = M*u

- Can be solved in least squares sense using Matlab

u = M\dIm

Matlab

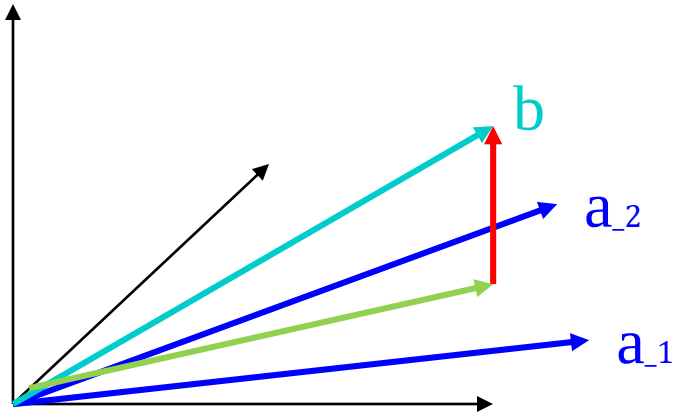
```
dIm = I(t+1)-I(t);
[dUm,dVm] = gradient(I(t)) % Or mean of I(t), I(t+1))
%Get an 8x8 patch:
dI = dIm(k:k+8, l:l+8);
(same for dU, dV)
% Flatten into vector
dIv = dI(:);
M = [dUv dVv];
u = M\dIv
```



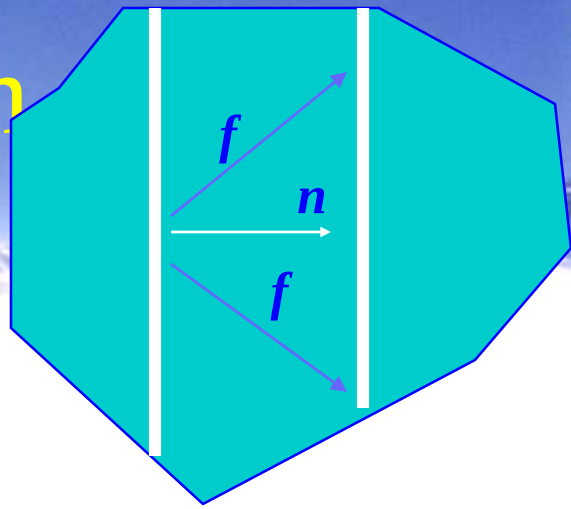
Geometric view of overdetermined equations

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

$$b = Ax$$



Aperture problem

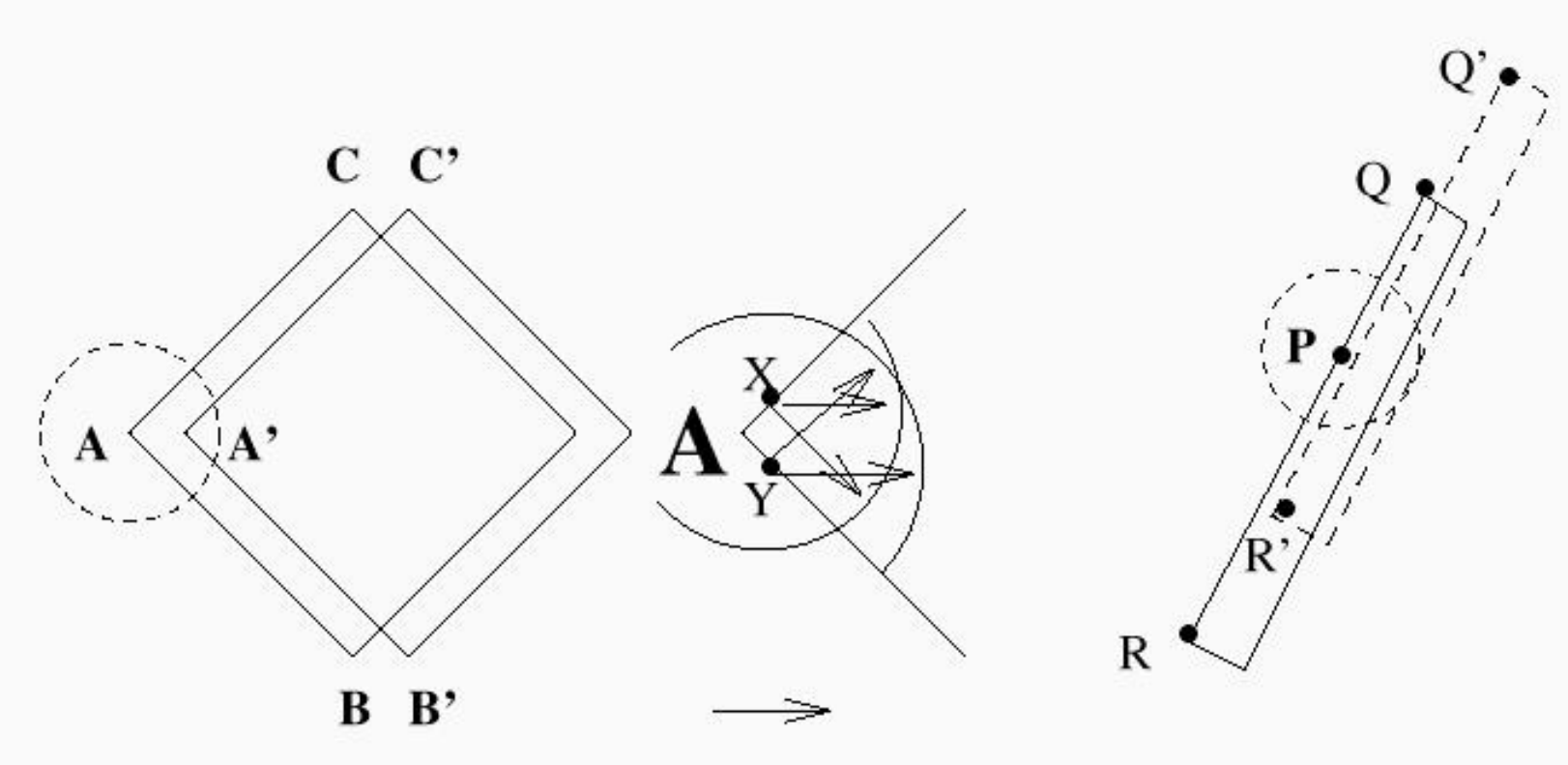


- Rewrite as dot product

$$-\frac{\partial \mathrm{Im}}{\partial t}\delta t = \left(\frac{\partial \mathrm{Im}}{\partial x}, \frac{\partial \mathrm{Im}}{\partial y}\right) \cdot \begin{pmatrix}\delta x \\ \delta y\end{pmatrix} = \nabla \mathrm{Im} \cdot \begin{pmatrix}\delta x \\ \delta y\end{pmatrix}$$

- Each pixel gives one equation in two unknowns:
  $\boldsymbol{n}*\boldsymbol{f} = \mathrm{k}$
- Min length solution: Can only detect vectors normal to gradient direction
- The motion of a line cannot be recovered using only local information

Aperture problem 2

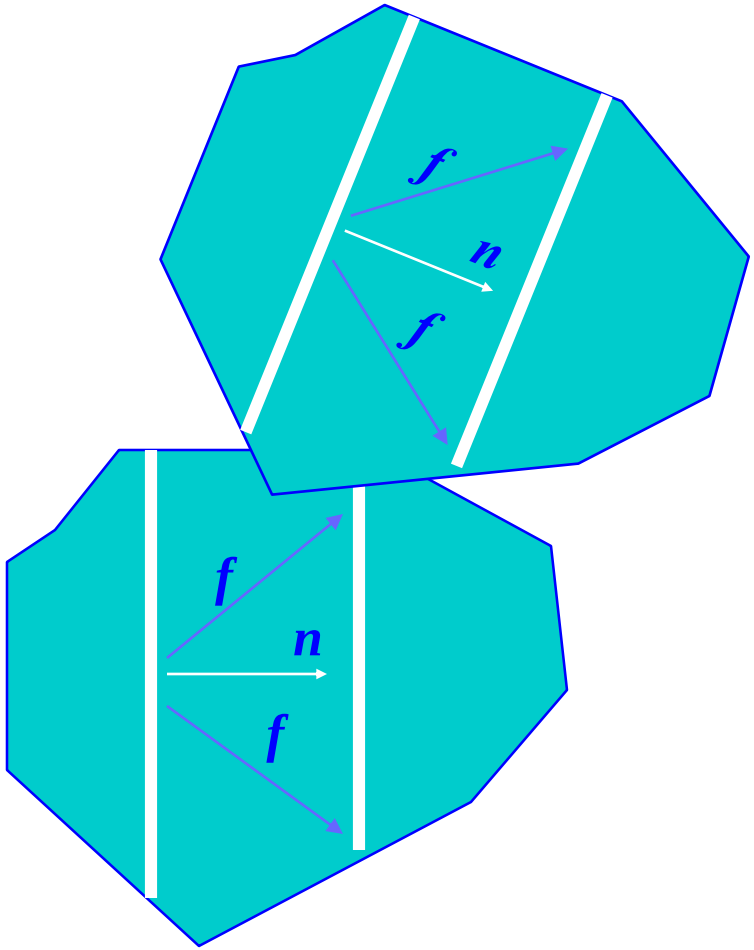


The flow continuity constraint

- Flows of nearby pixels or patches are (nearly) equal
- Two equations, two unknowns:

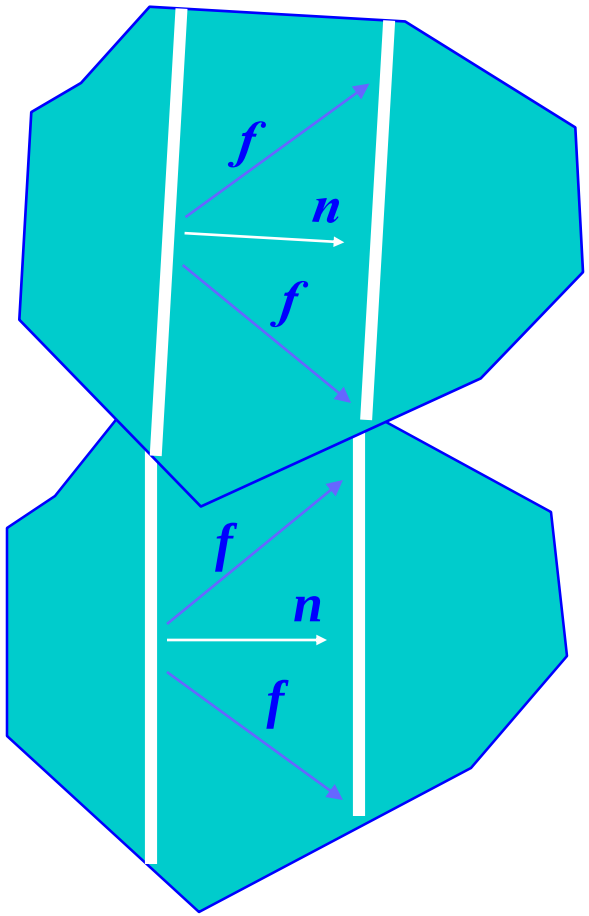
$$\boldsymbol{n}_1 * \boldsymbol{f} = \mathrm{k}_1$$
$$\boldsymbol{n}_2 * \boldsymbol{f} = \mathrm{k}_2$$

- Unique solution $\boldsymbol{f}$ exists, provided $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ not parallel



Sensitivity to error

- $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ might be ***almost*** parallel
- Tiny errors in estimates of k's or $\boldsymbol{n}$'s can lead to huge errors in the estimate of $\boldsymbol{f}$



Conditions for solvability

– SSD Optimal (u, v) satisfies Optic Flow equation

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

$A^T A$ $\qquad$ $A^T b$

Better: [u,s,v] = svd(A), s=diag(λ_1, λ_2) Heath Ch3.5

When is this solvable?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ entries should not be too small (noise)
- $\mathbf{A^T A}$ should be well-conditioned
- Study eigenvalues:
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

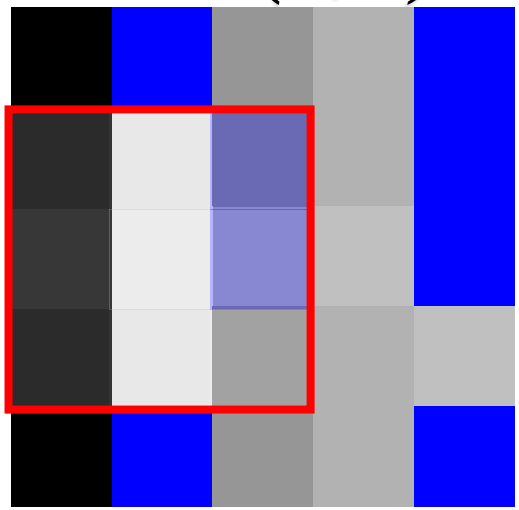
Simple image registration algorithm
SSD error norm

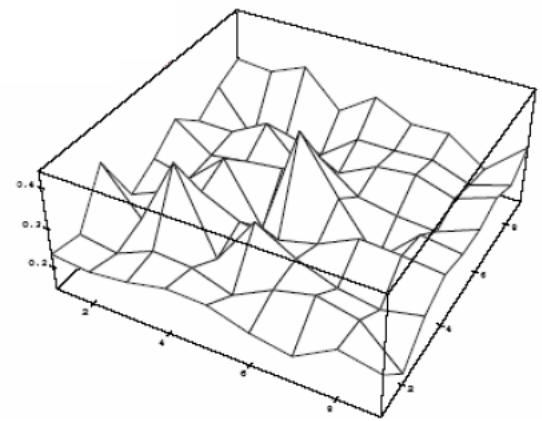
$$E(u,v) = \sum_{x,y} \left(I(x+u, y+v) - T(x,y) \right)^2$$

For each offset *(u, v)*

compute *E(u,v);*

Choose *(u, v)* which minimizes *E(u,v);*

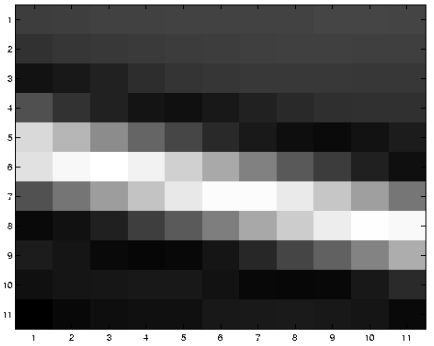


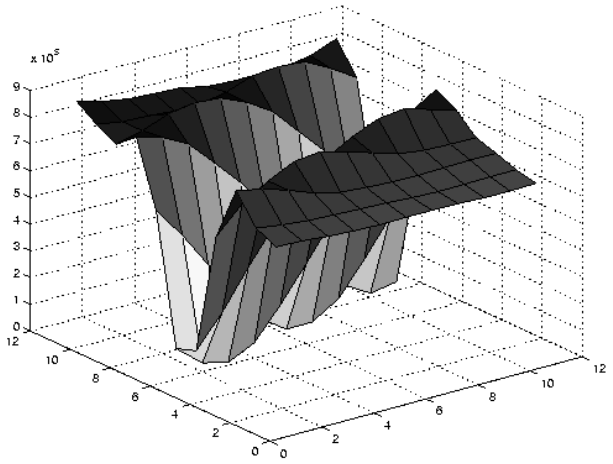


Optic Flow Real Image Challenges:

- Can we solve for accurate optic flow vectors everywhere using this image sequence?

Edge



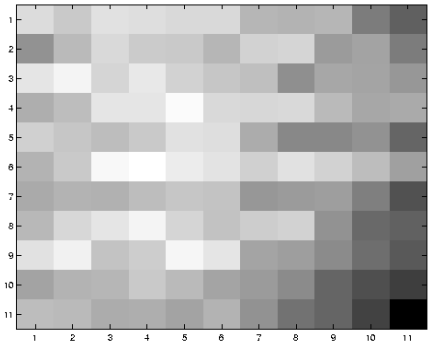


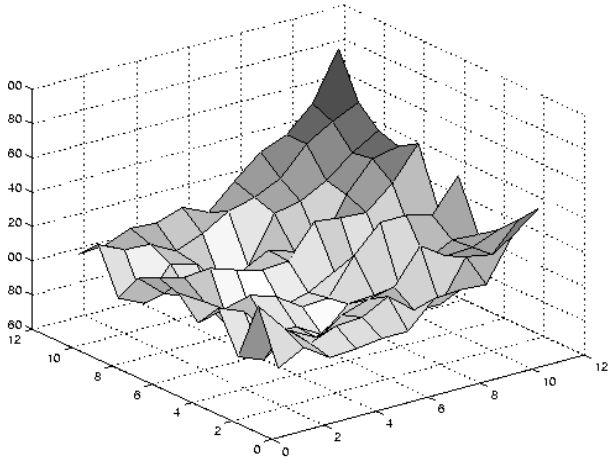
$$\sum \nabla I (\nabla I)^T$$

– gradients very large or very small

– large λ_1, small λ_2

Low texture region

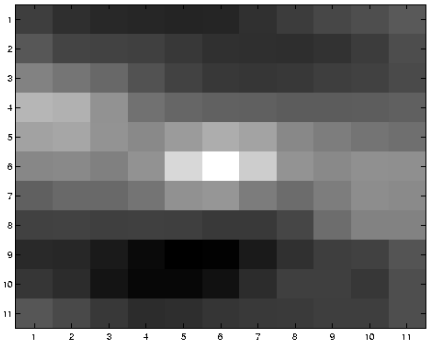


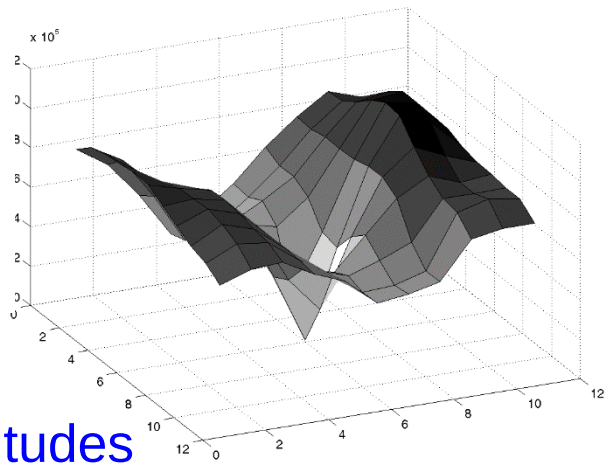


$$\sum \nabla I (\nabla I)^T$$

- gradients have small magnitude
- small λ_1, small λ_2

High textured region





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

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

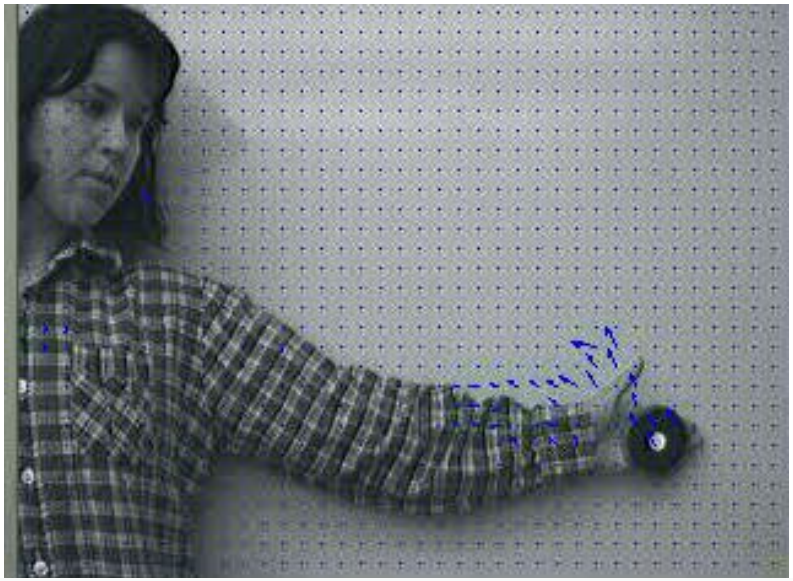
Observation

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

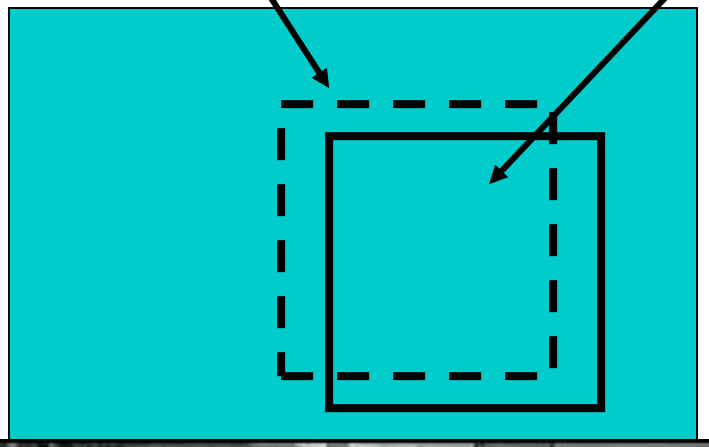
Review "Visual motion detection / optic flow"

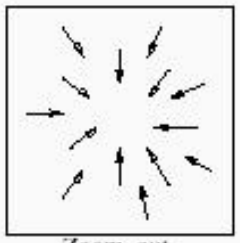
- Relating two adjacent frames: (small differences):

$$\mathrm{Im}(x+\delta x,\ y+\delta y,\ t+\delta t) = \mathrm{Im}(x,\ y,\ t)$$

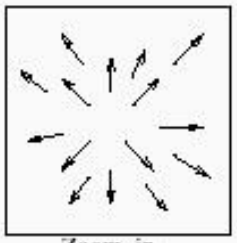


Vector Field $[\delta x\ \delta y]$

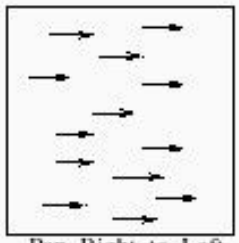




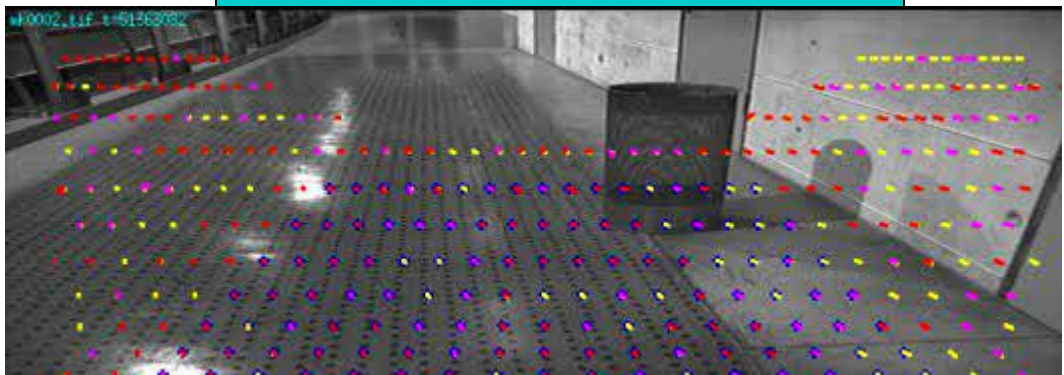
Zoom out



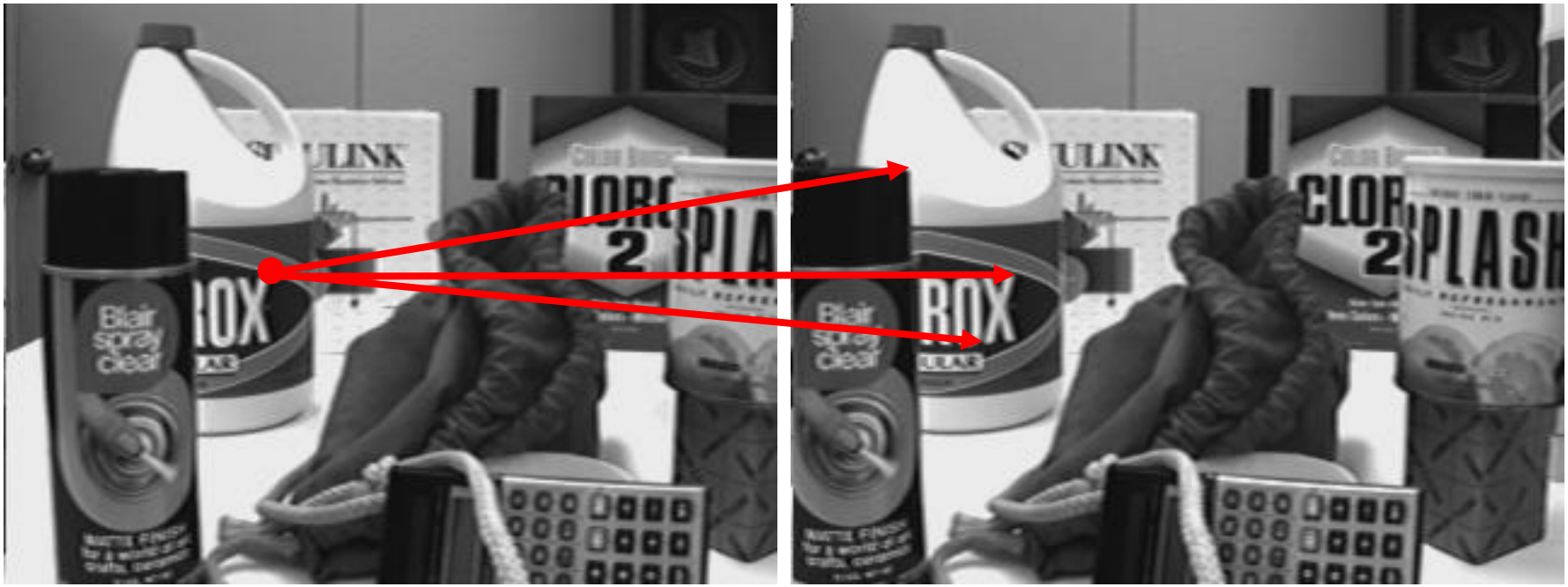
Zoom in



Pan Right to Left



Correspondence between image points
Motion, Optic flow, Tracking, Features

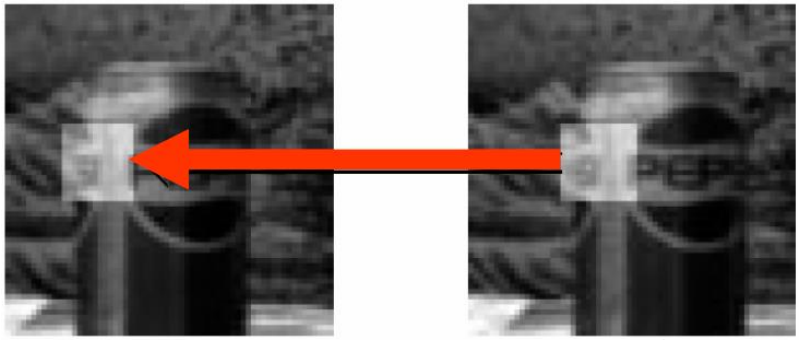


Time **t**

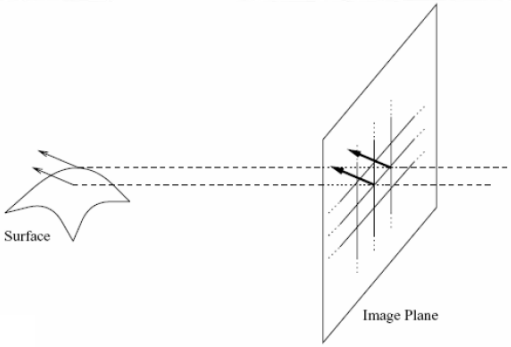
t+1

Image correspondence: Three assumptions

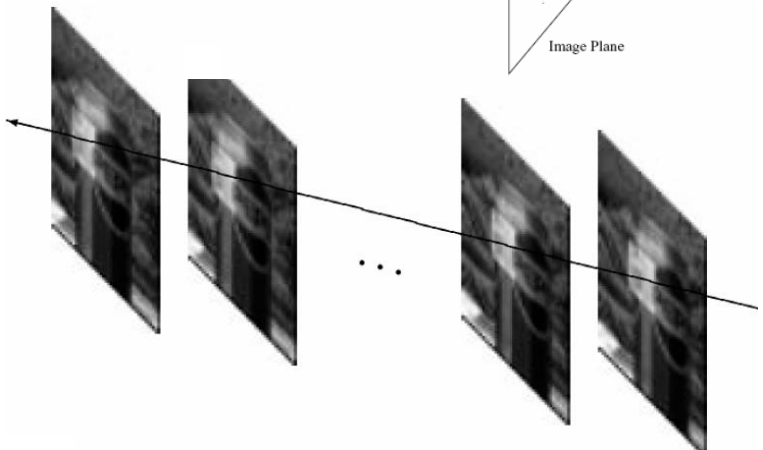
- Brightness consistency



- Spatial coherence



- Temporal persistence



Solve for optic flow using several simultaneous equations

- Taylor expansion for each patch:

$$\mathrm{Im}(x+\delta x, y+\delta y, t+\delta t) = \mathrm{Im}(x,y,t) + \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t + h.o.t.$$

$$(\exists \delta x, \delta y) s.t.\ \ \mathrm{Im}(x+\delta x,\ \ y+\delta y,\ \ t+\delta t) \ \ = \ \ \mathrm{Im}(x,\ \ y,\ \ t)$$

← Brightness consistency

$$0 = \frac{\partial \mathrm{Im}}{\partial x}\delta x + \frac{\partial \mathrm{Im}}{\partial y}\delta y + \frac{\partial \mathrm{Im}}{\partial t}\delta t$$

- Over determined equation system:

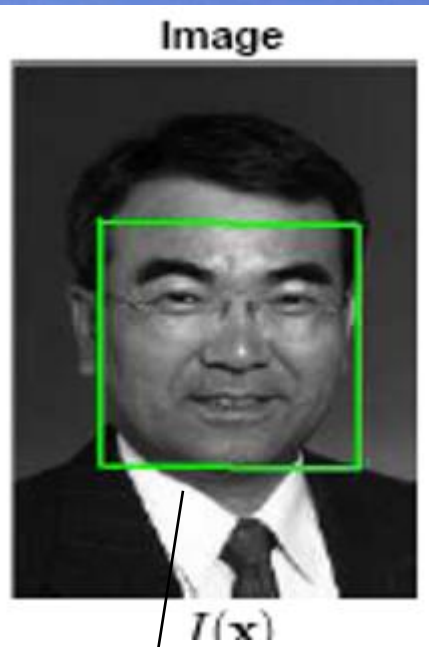
$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

dIm = M*u

- Can be solved in least squares sense using Matlab

u = M\dIm

Intuitive connection images - equations



I(t+1)-I(t);

=

∇I_x

$\delta x +$

∇I_y

δy

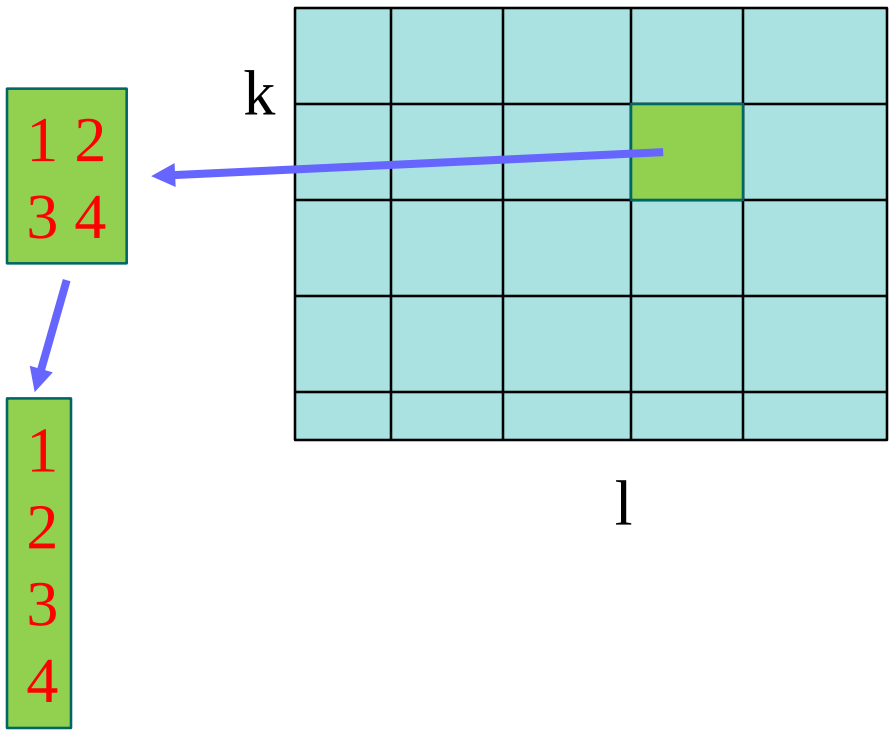
Template

$T(\mathbf{x})$

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

Matlab

```
dIm = I(t+1)-I(t);
[dUm,dVm] = gradient(I(t)) %dUm = I(x+1,y,t) - I(x,y,t)
%Get an 8x8 patch:
dI = dIm(k:k+8, l:l+8);
(same for dU, dV)
% Flatten into vector
dIv = dI(:);
M = [dUv dVv];
u = M\dIv
```



Problems in Optic flow computation

- 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?

Iterative Refinement

- Used in SSD/Lucas-Kanade tracking algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp / shift pixels I(t+1) = H towards I(t) using the estimated flow field

     - *use image warping techniques – ie OpenGL texture rendering*
  3. Repeat until convergence

  (Iteration just like in 340 Newton methods, Heath Ch 5.)

Tracking Lucas-Kanade algorithm

Solving for the translational motion of a patch H

Over determined equation system:

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

-Im_t = M u

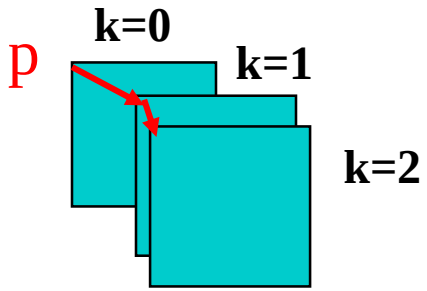
- Solve least squares using matlab

u = M\Im_t

- Update: "Newton step" ……

p = p + u

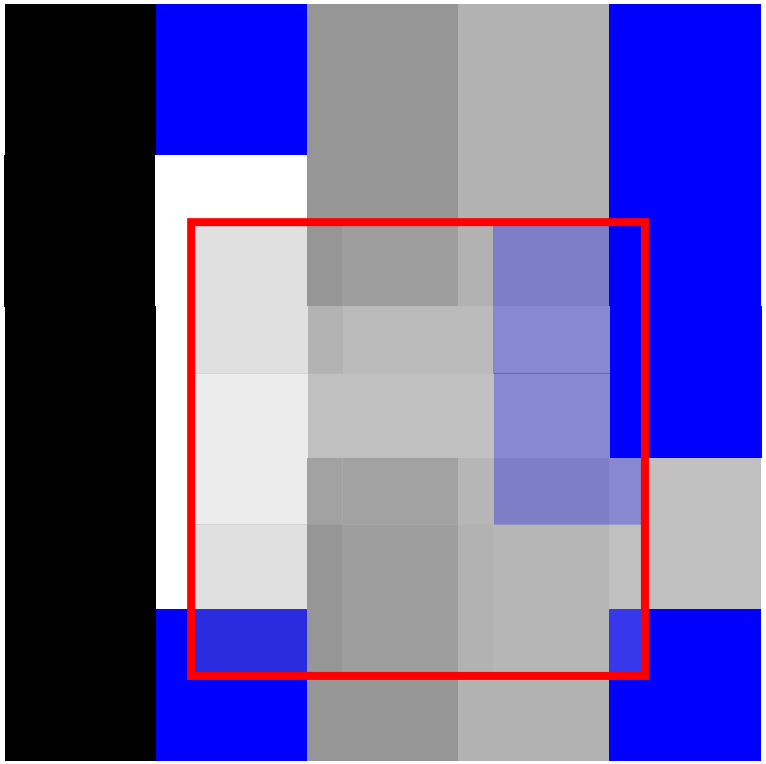
- For k=1,2,… Im_t = Im(t+1,x+p) – Im(t, x)

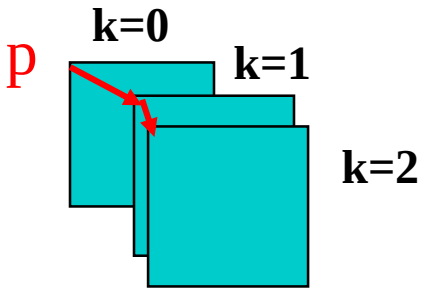


How to source Im(.,x+p) ?

- When p is not integer interpolate pixels

$$Im(.,x+[1.2,1.5]) = .8*.5*Im(.,x+1,y+1) +.2*.5*Im(.,x+2,y+1)+\ldots$$

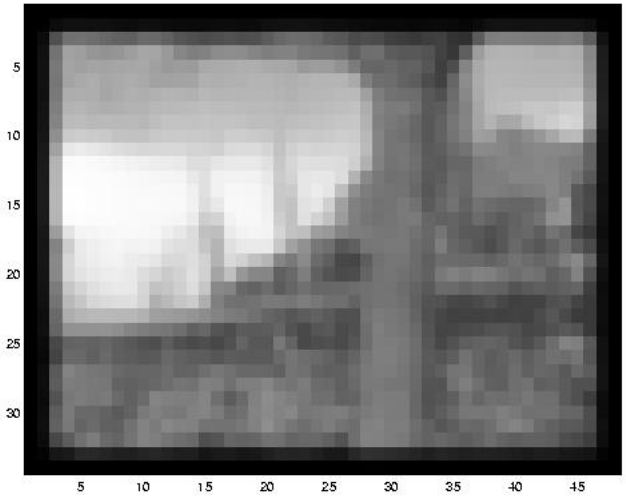




Revisiting the small motion assumption

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

Reduce the resolution!



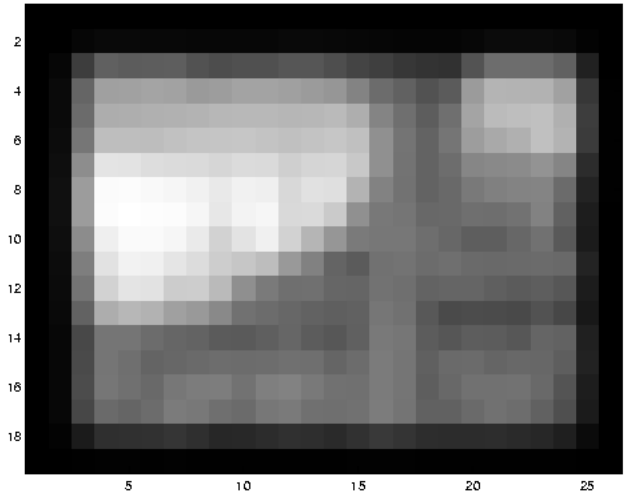
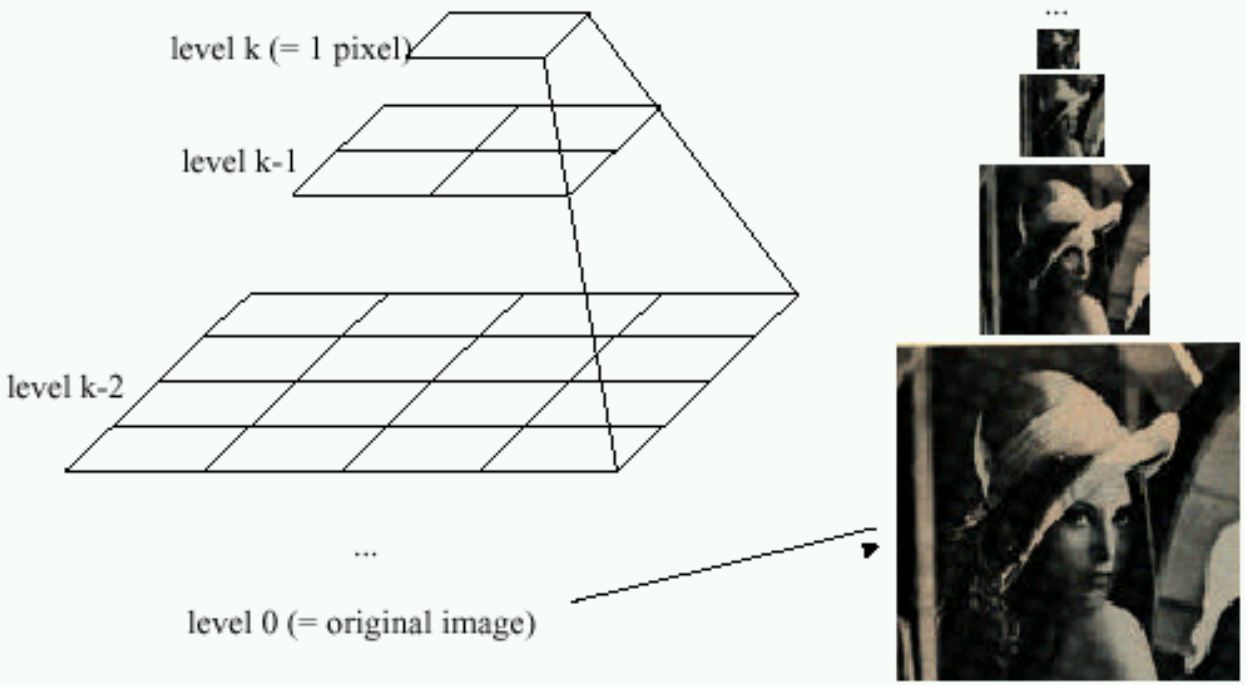
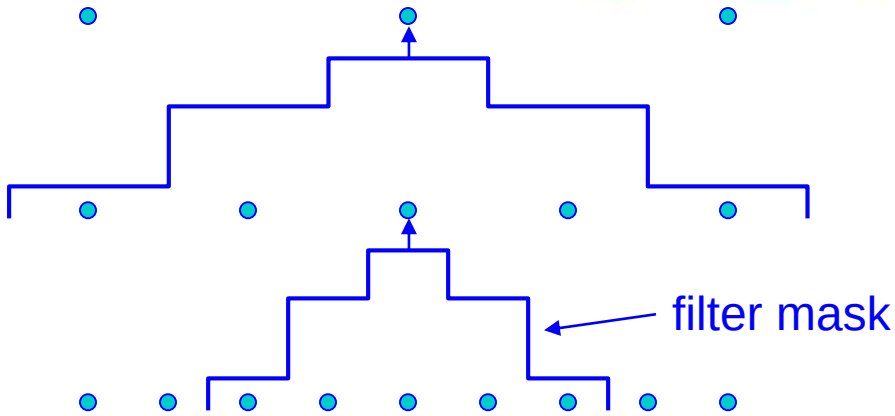


Image Pyramids

Idea: Represent NxN image as a "pyramid" of 1x1, 2x2, 4x4,…, 2^kx2^k images (assuming N=2^k)

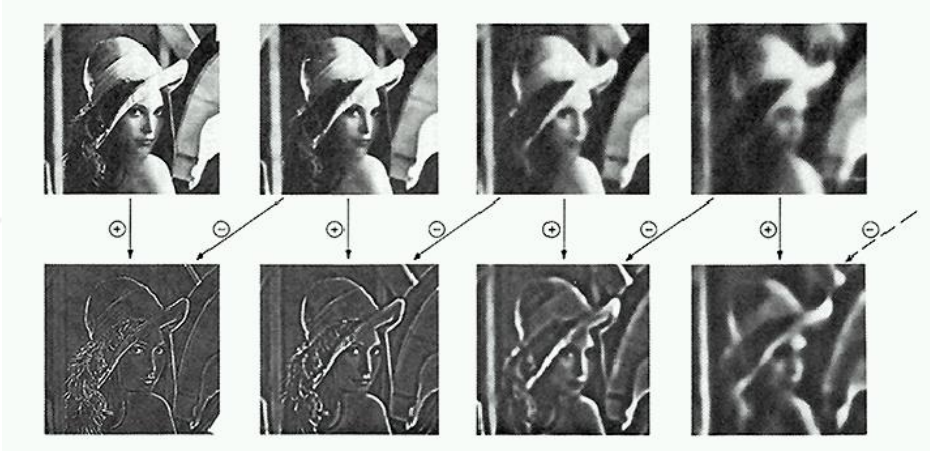


Pyramid Creation



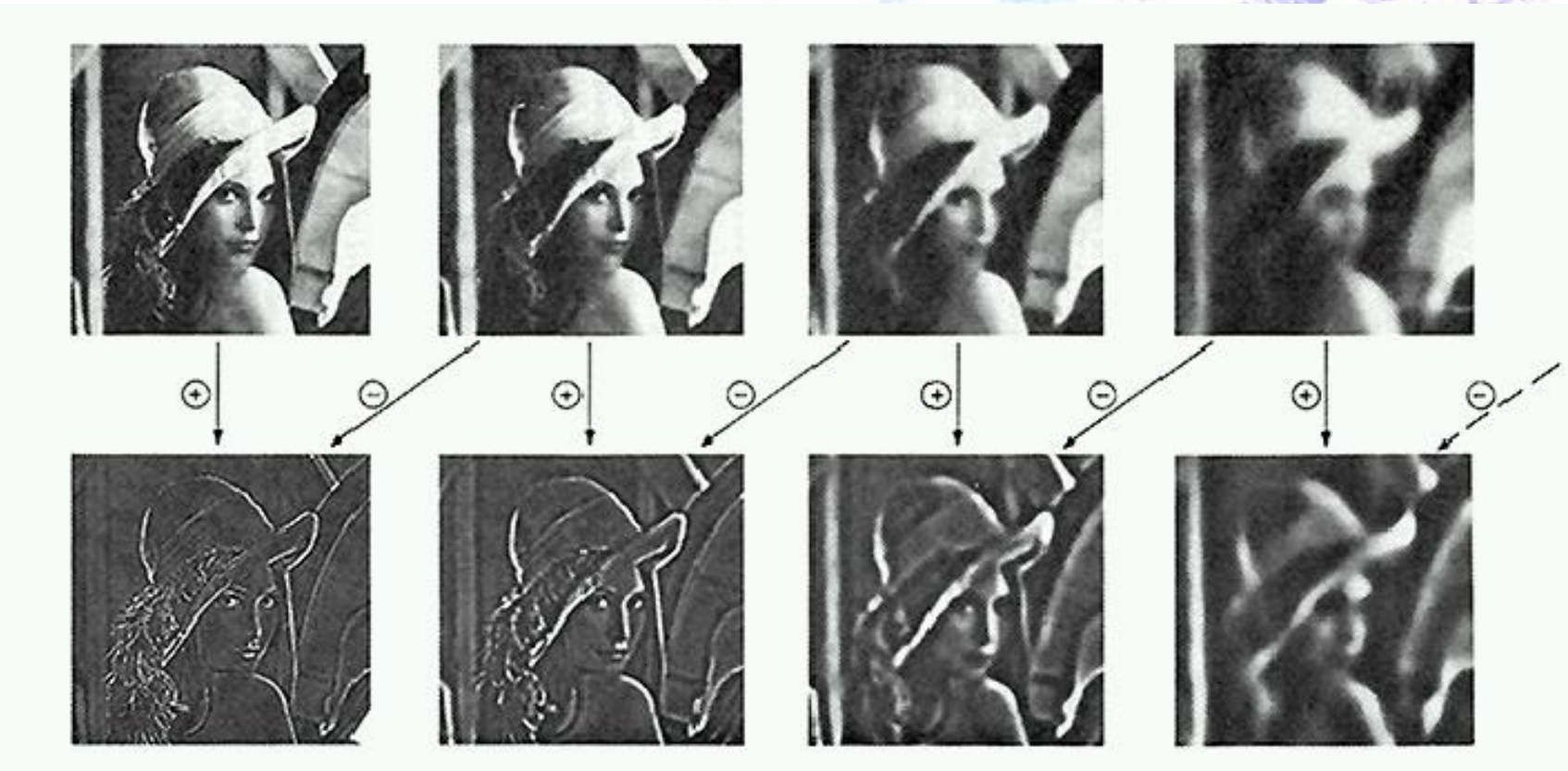
"Gaussian" Pyramid

- "Laplacian" Pyramid
  - Created from Gaussian pyramid by subtraction
    $L_l = G_l - \text{expand}(G_{l+1})$



Octaves in the Spatial Domain

Lowpass Images



- Bandpass Images

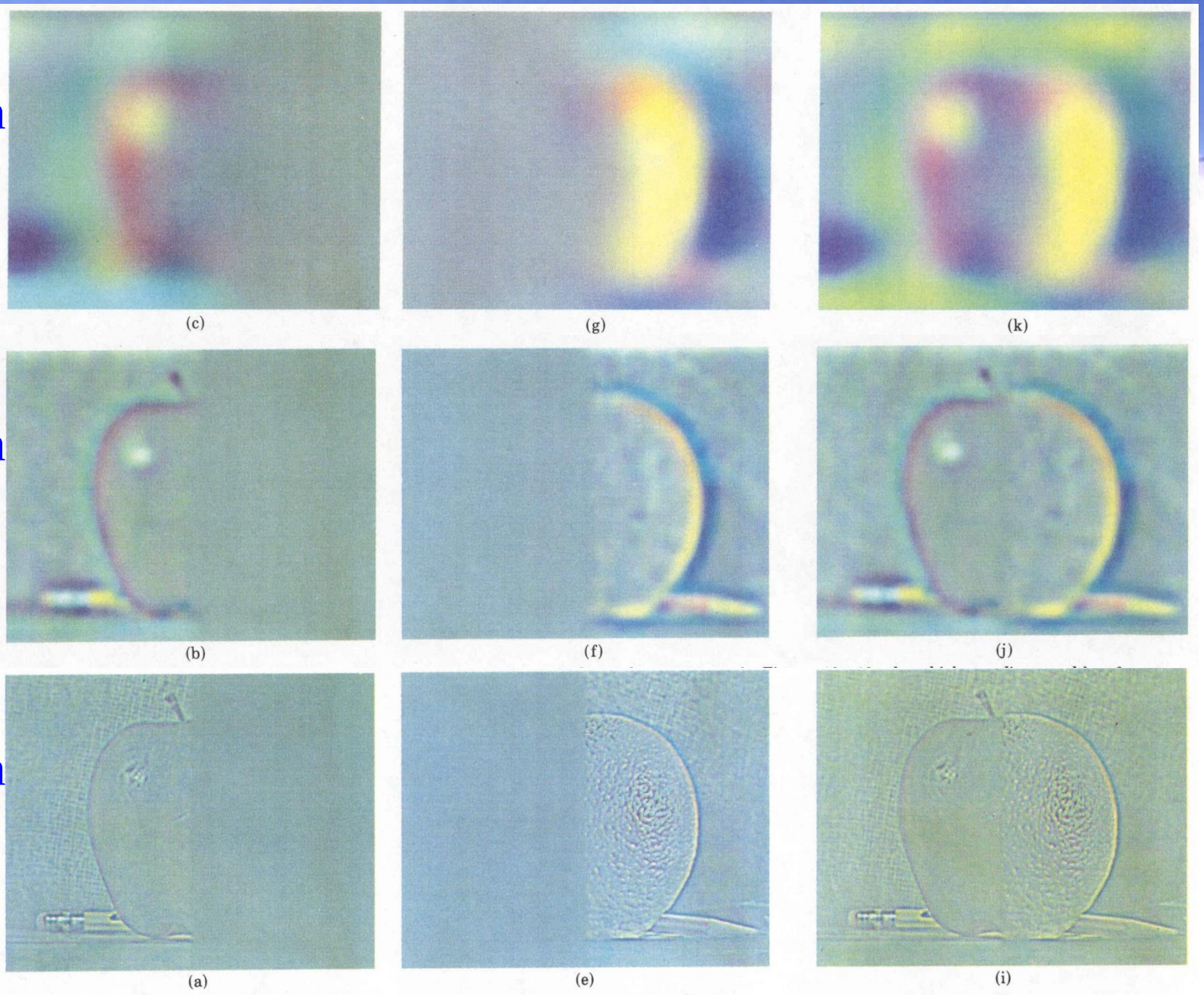
Pyramids

- Advantages of pyramids
  - Faster than Fourier transform
  - Avoids “ringing” artifacts
- Many applications
  - small images faster to process
  - good for multiresolution processing
  - compression
  - progressive transmission
- Known as “mip-maps” in graphics community
- Precursor to wavelets
  - Wavelets also have these advantages

Laplacian level 4

Laplacian level 2

Laplacian level 0

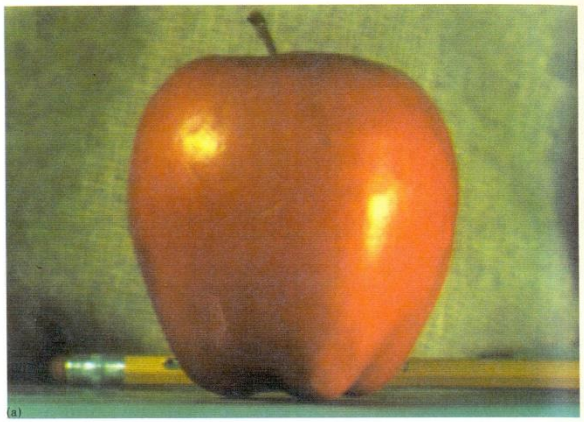


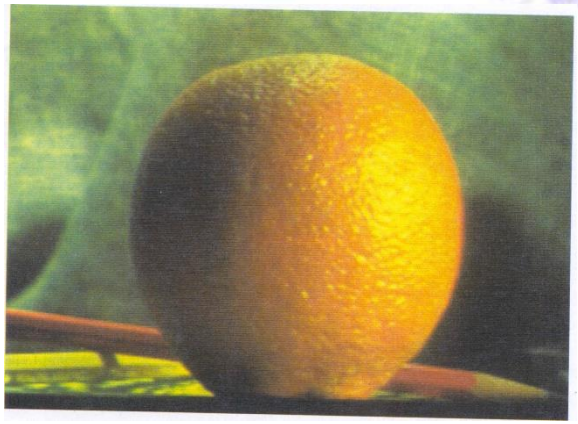
left pyramid

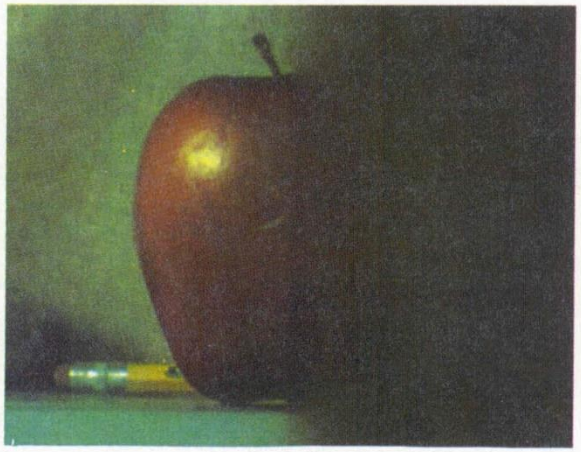
right pyramid

blended pyramid

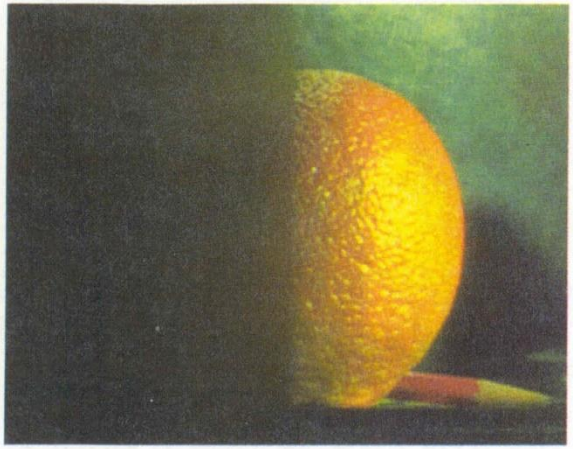
Pyramid Blending



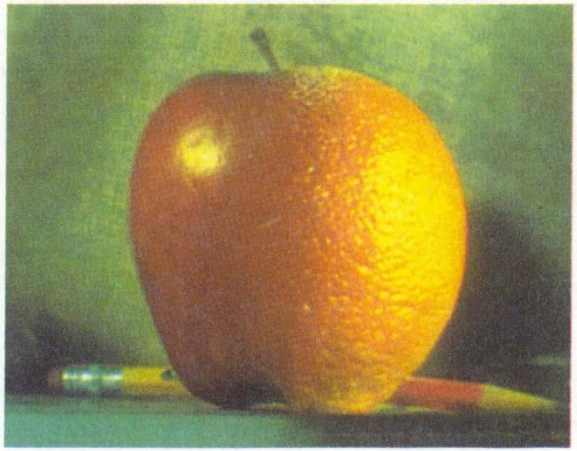




(d)

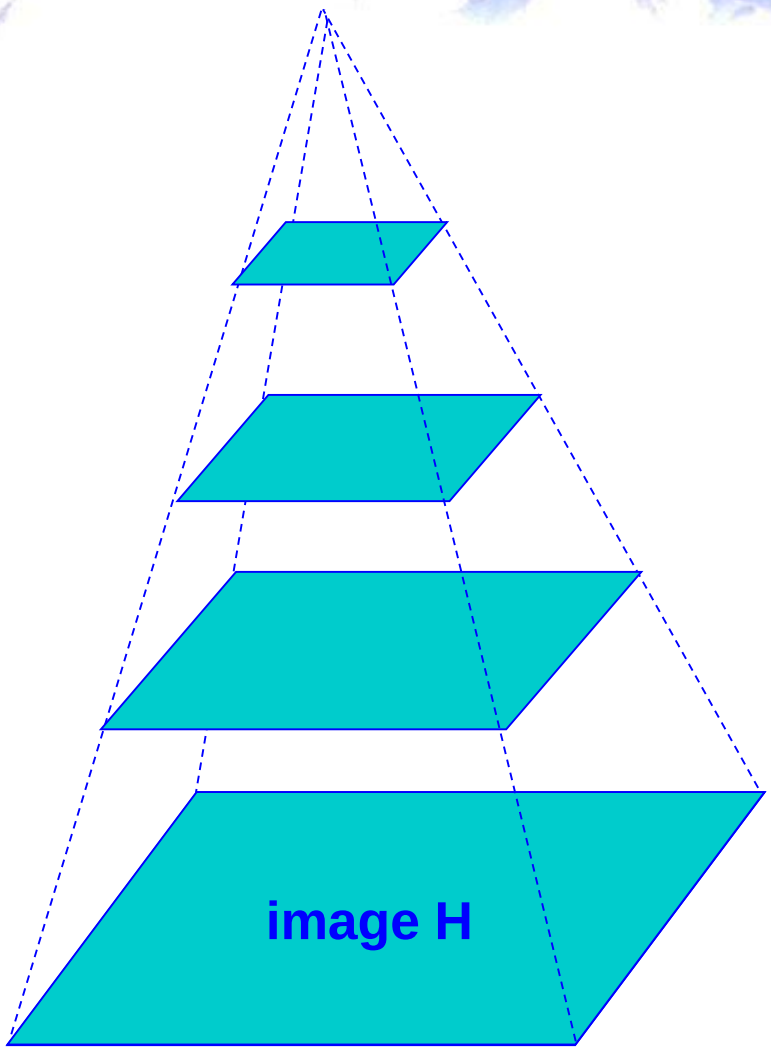


(h)



(l)

Coarse-to-fine optical flow estimation

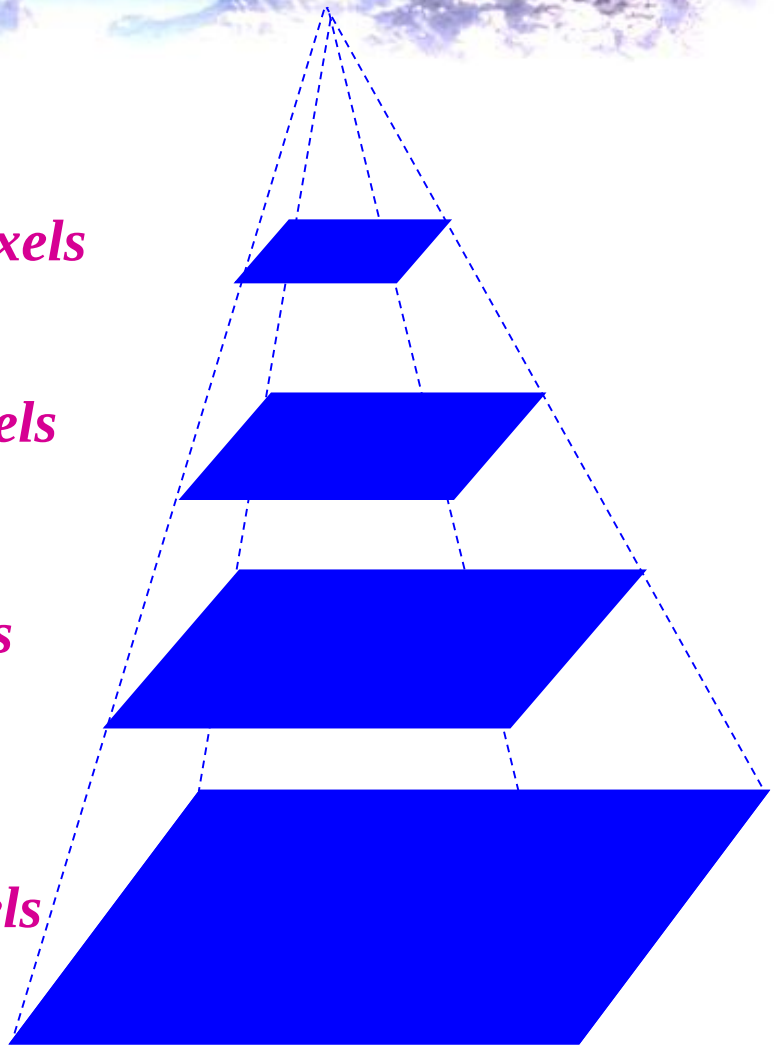


u=1.25 pixels

u=2.5 pixels

u=5 pixels

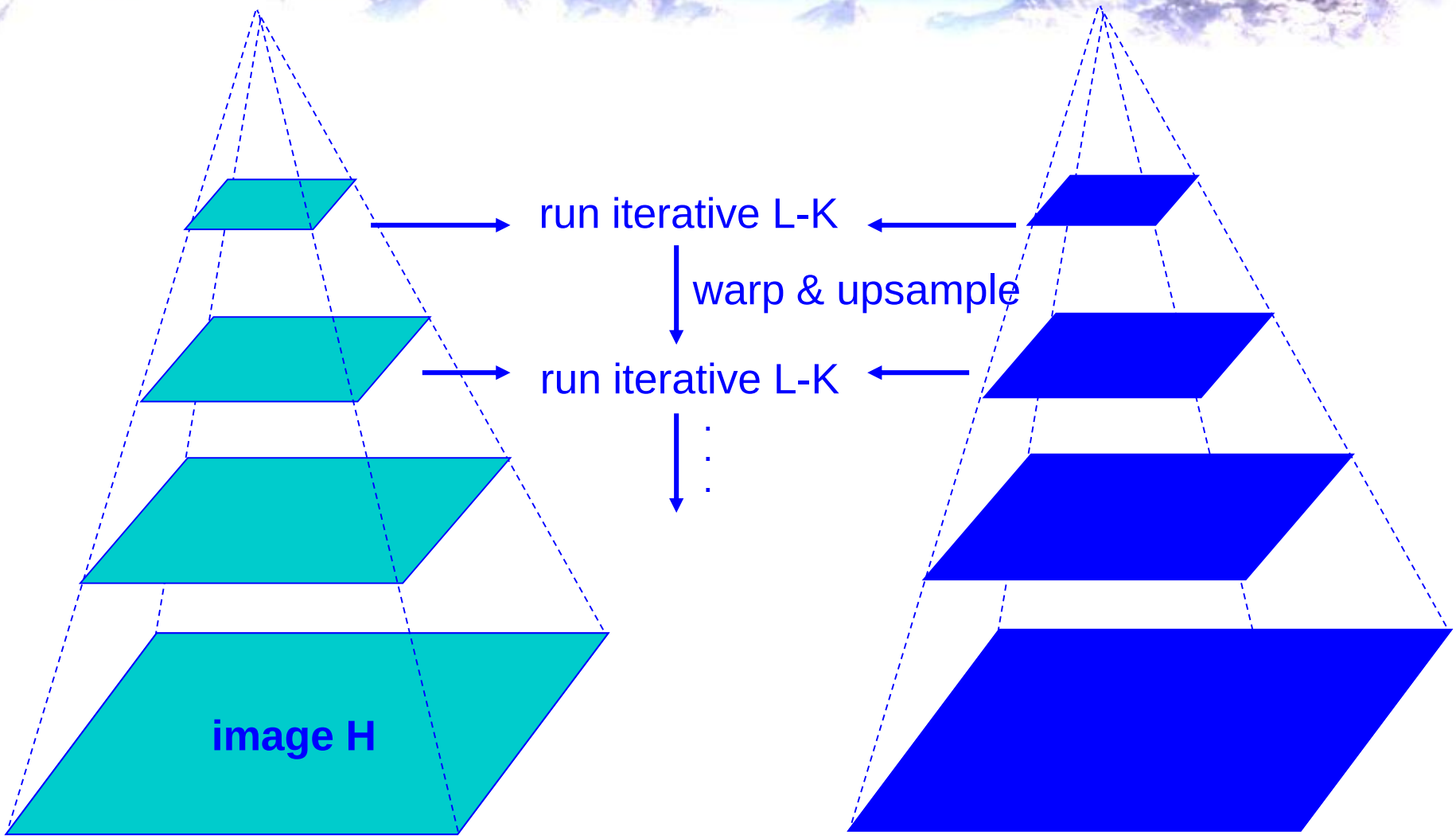
u=10 pixels



Gaussian pyramid of image H

Gaussian pyramid of image I

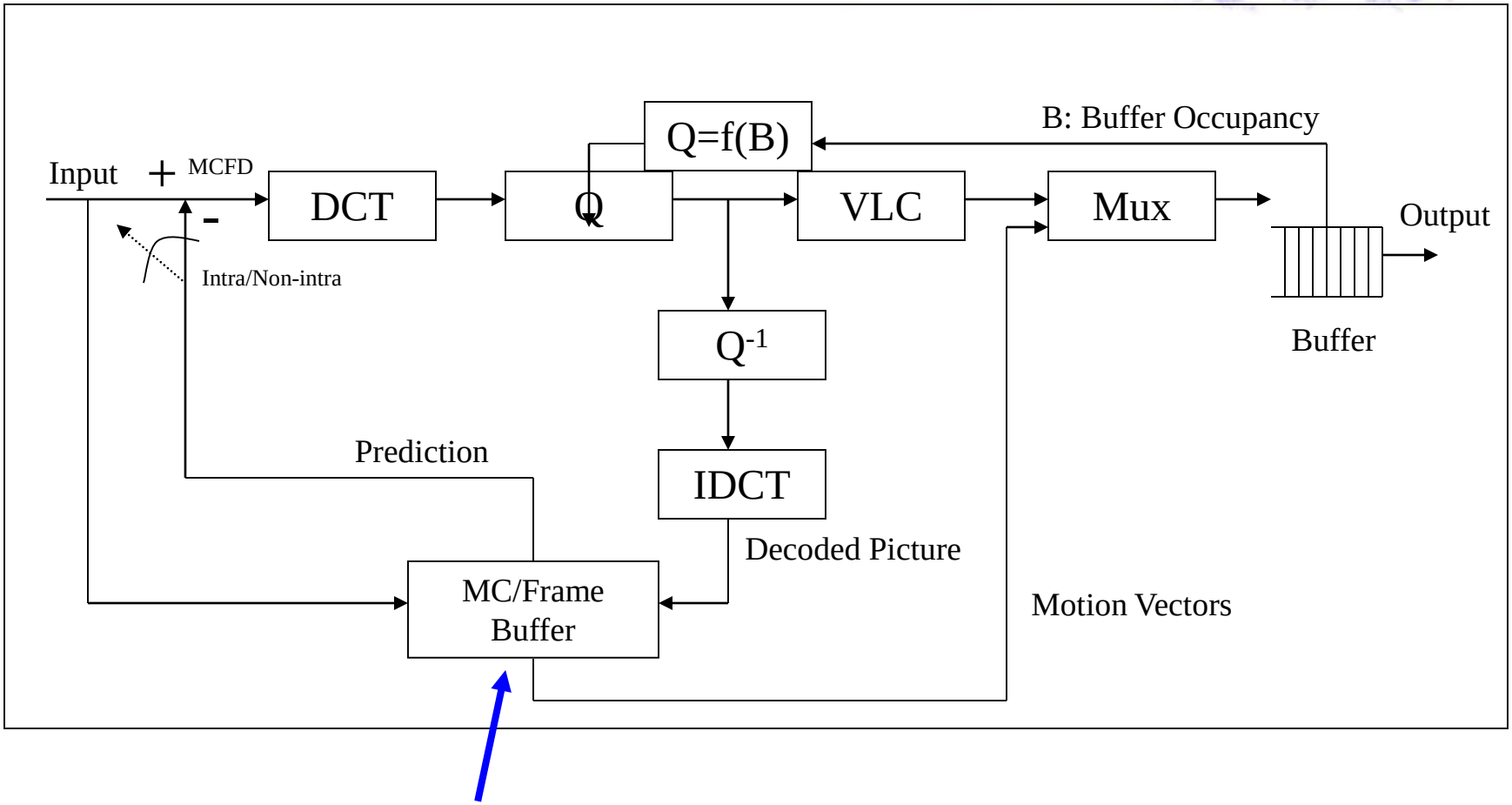
Coarse-to-fine optical flow estimation



Gaussian pyramid of image H

Gaussian pyramid of image I

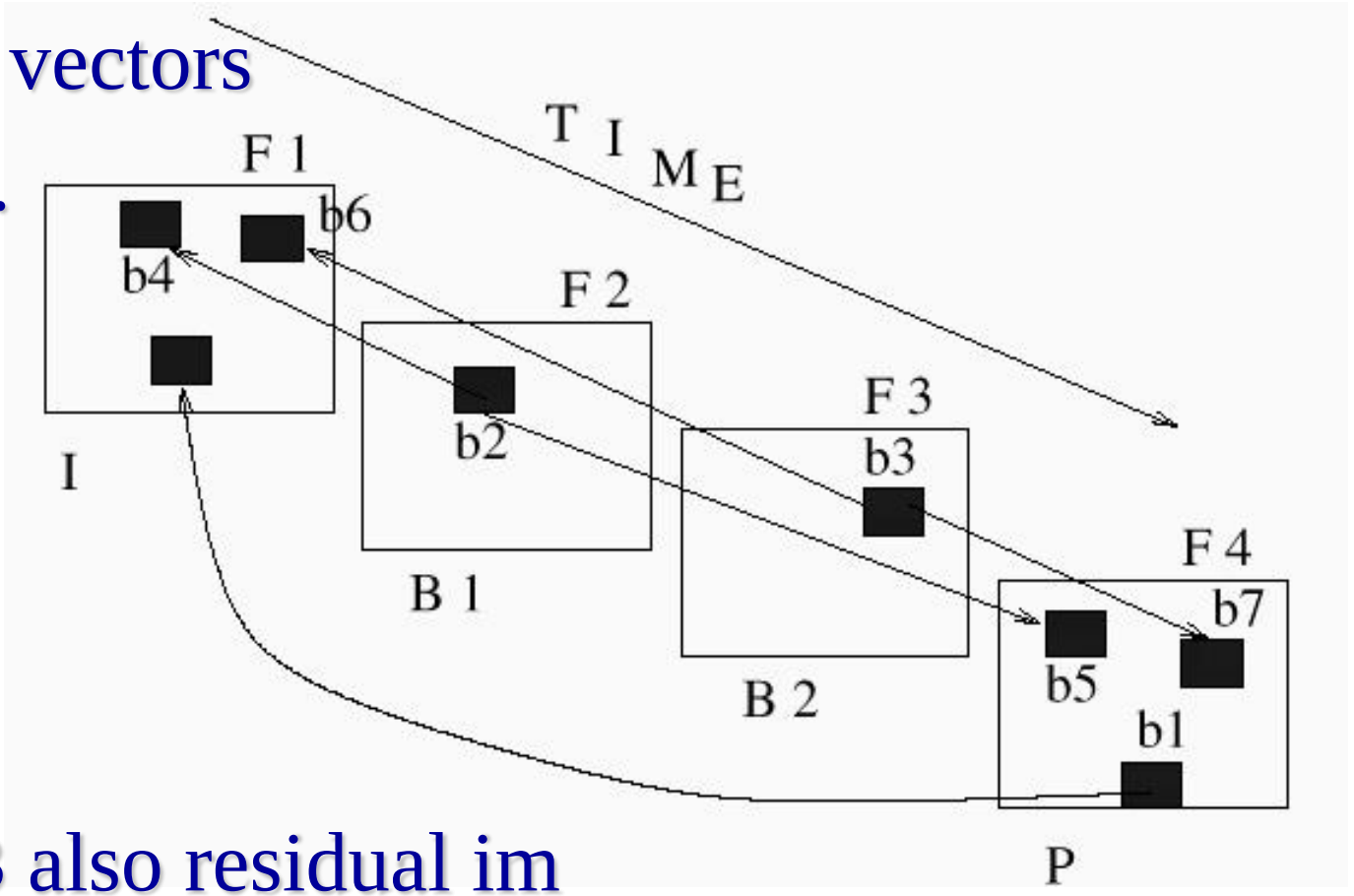
Video compression codec



- Uses optic flow computation

Compressed movie decoding

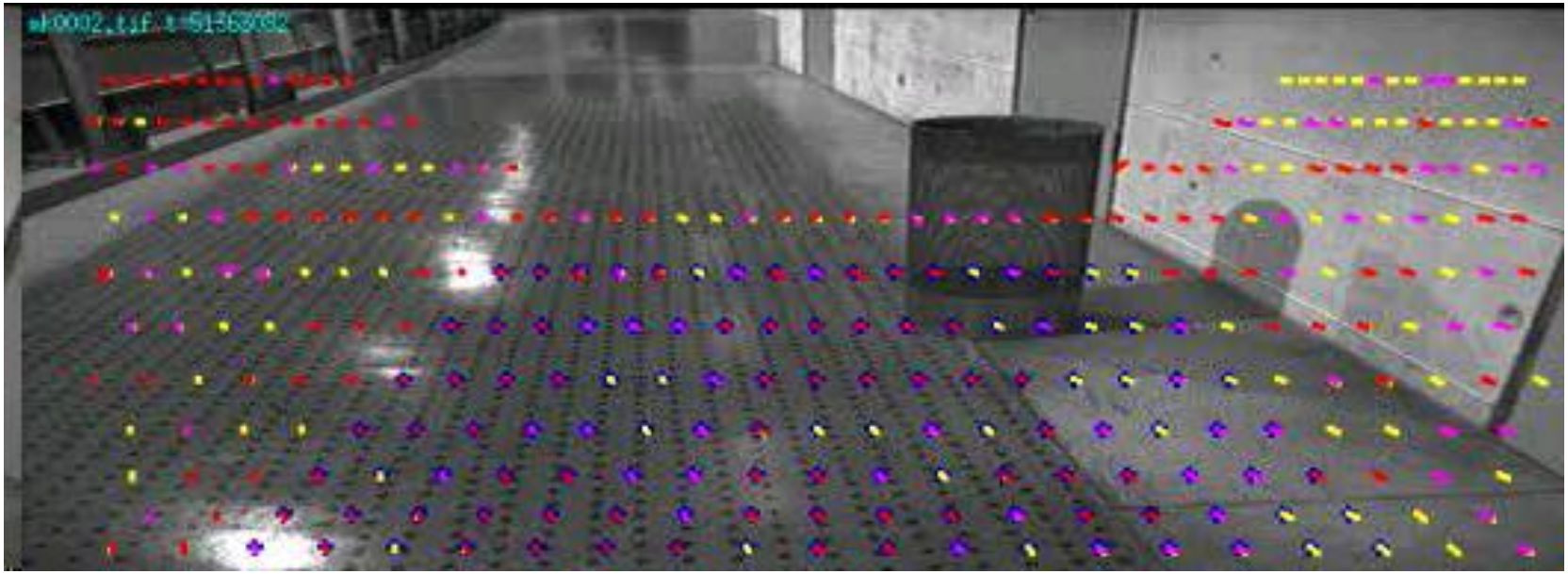
- I = intensity frame(s) (DCT compressed)
- P = motion vectors
- B=reconstr. interpolated motion



Stream:

- IPBBB…
- Frames P,B also residual im

HW accelerated computation of flow vectors



- Norbert's trick: Use an mpeg-card to speed up motion computation

Other applications:

- Video tracking – Next lecture topic and L1.2
- Motion control, robots and animals (we will cover later)
- Image segmentation
- Recursive depth recovery:

Lab:

- Assignment1:
- Purpose:
  - Intro to image capture and processing
  - Hands on optic flow experience
- See www page for details.
- Suggestions welcome!

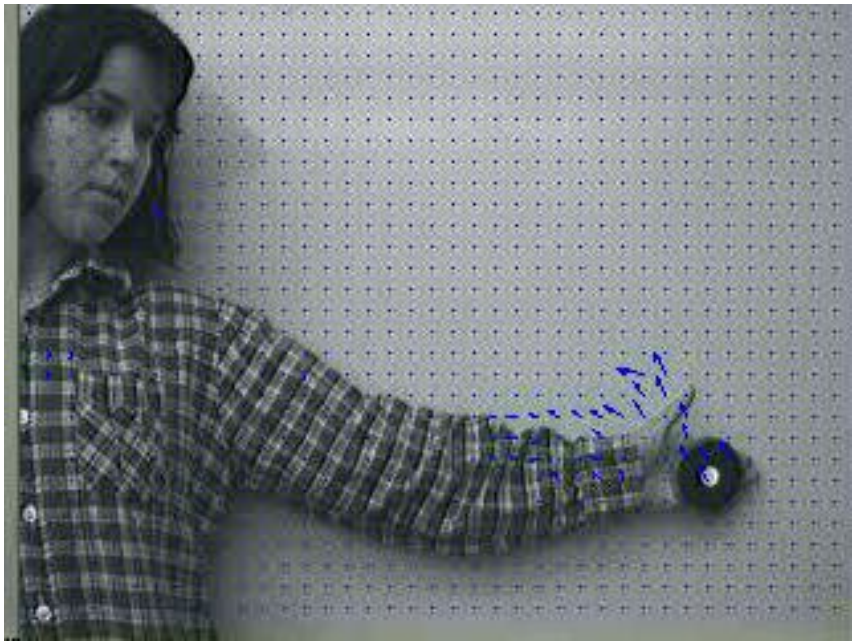
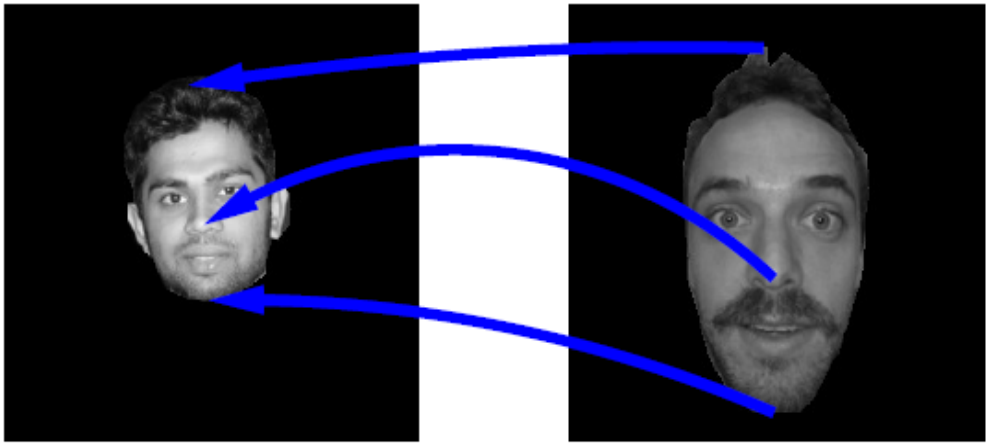


Image registration

WHAT is image registration



Transform
a “source” image
to **match** a “target” image

Image registration

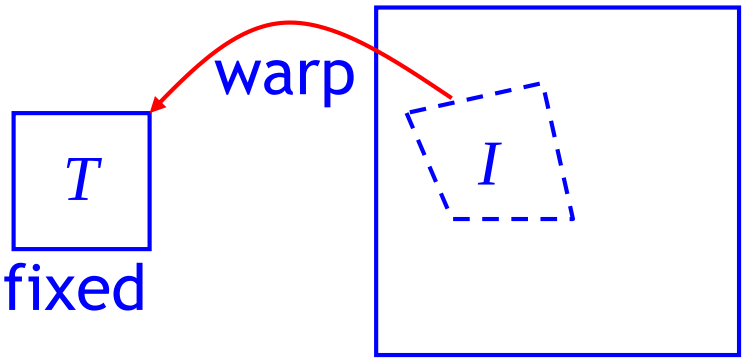
WHAT is image registration

Transform a “source” image
match a “target” image

Image registration: Three applications

Goal: register a template image $T(x)$ and an input image $I(x)$, where $x=(x,y)^T$. (warp I so that it matches T)

1. Image alignment: $I(x)$ and $T(x)$ are two images
2. Tracking: $T(x)$ is a small patch around a point p in the first video image, $t=0$. $I(x)$ is the image at time $t+1$.
3. Optical flow: $T(x)$ and $I(x)$ are patches of images at t and $t+1$.
4. Quadrilaterals, triangles, pixels…

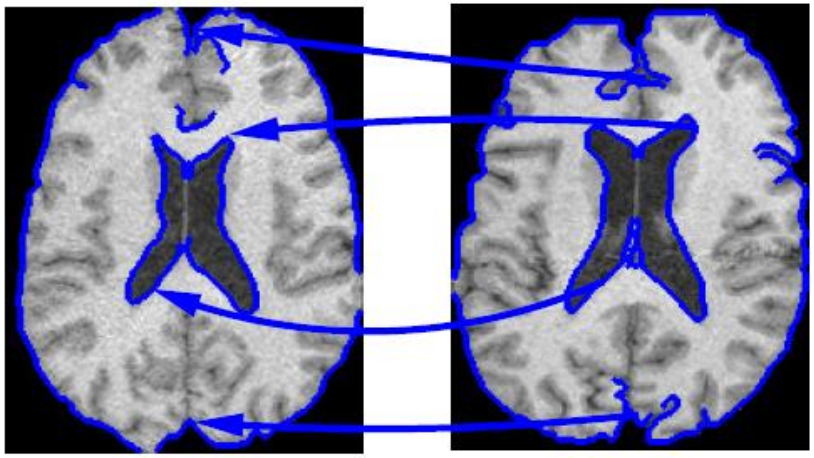


Medical image registration

WHAT is image registration

Transform a “source” image
match a “target” image

Medical image registration

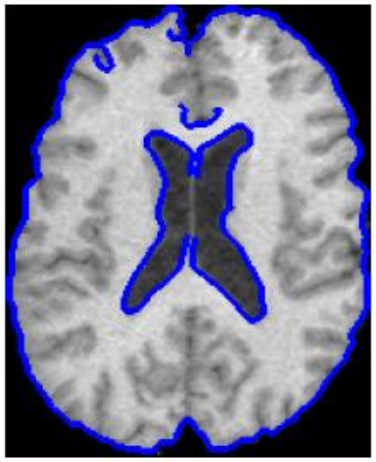


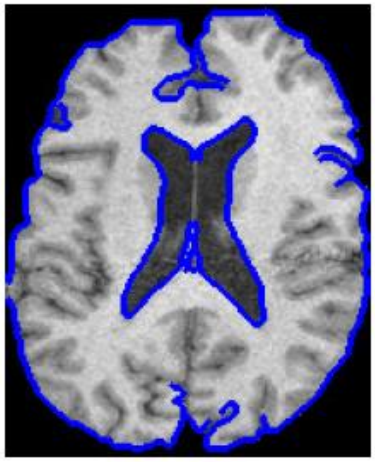
Medical image registration

WHAT is image registration

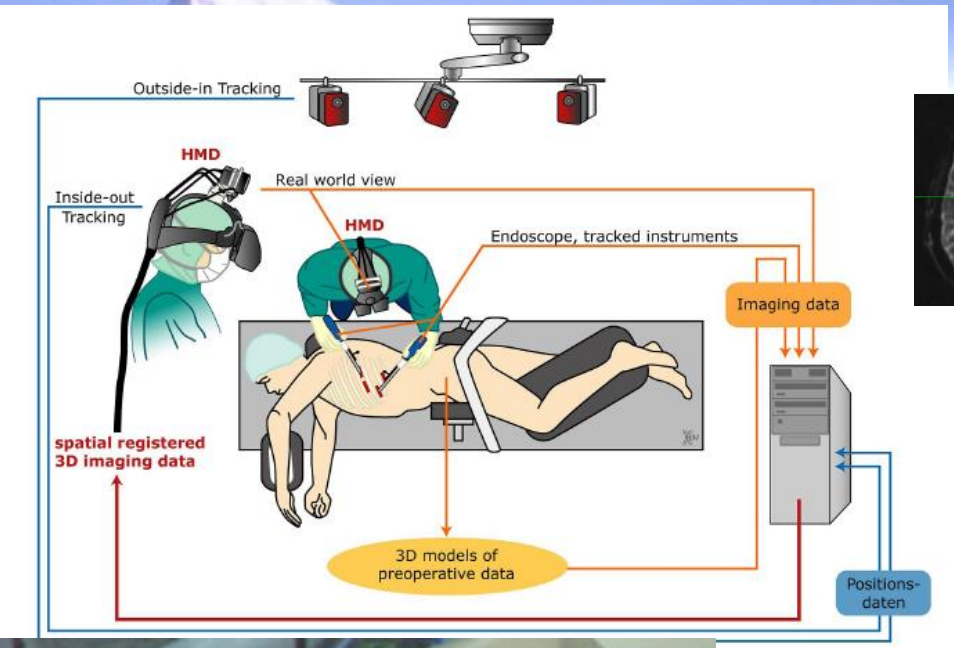
Transform a “source” image
match a “target” image

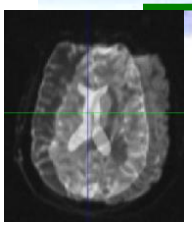
Medical image registration



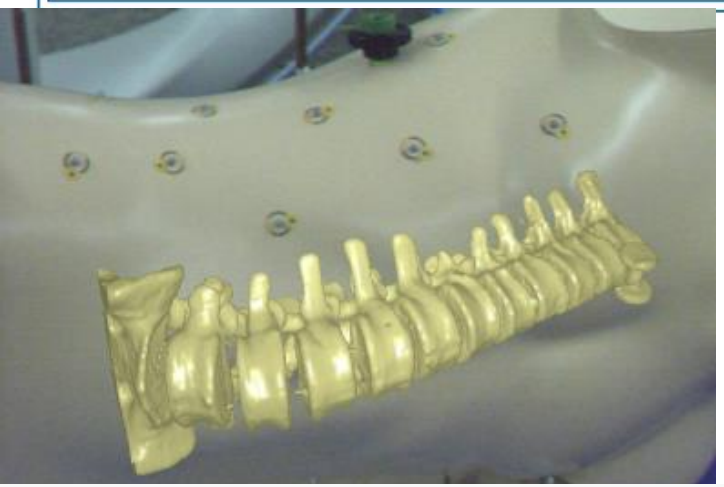


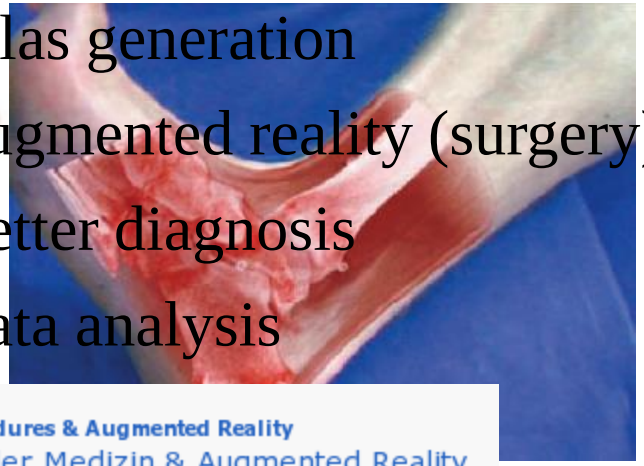
Medical applications





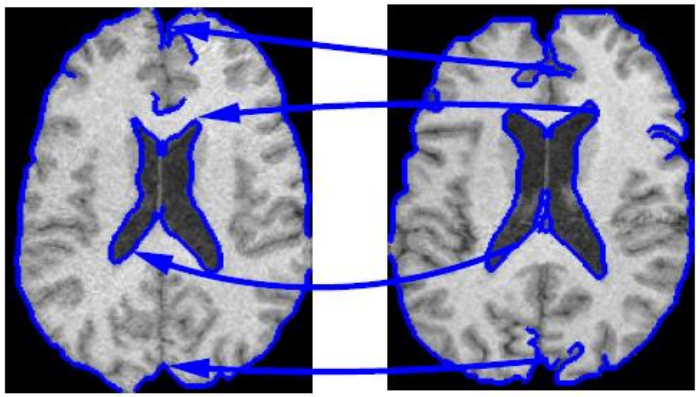
- **Data (source, target)**
  - different medical images modalities (MRI, XRay, CT…)
  - pre-acquired medical image with real-time images (video)
  - patient data with an atlas
- **For:**
  - atlas generation
  - augmented reality (surgery)
  - better diagnosis
  - data analysis





Formulation

Very similar to tracking and optic flow.



I_A I_B

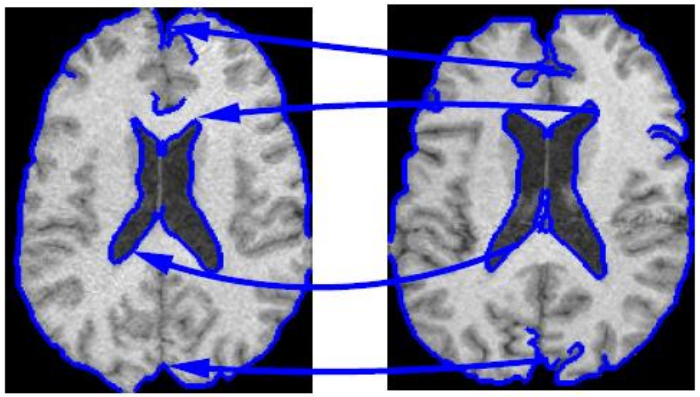
Transform a “source” image to **match** a “target” image

Find best transformation T through the minimization of an energy

$\min_T$ **Sim**(I_A - **T**(I_B))

Formulation

Very similar to tracking and optic flow.



I_A I_B

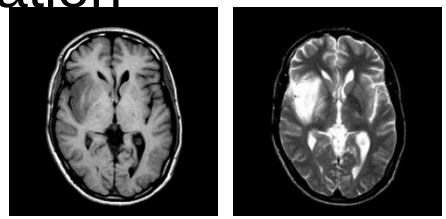
Transform a “source” image to **match** a “target” image

Find best transformation T through the minimization of an energy

$\min_T$ **Sim**(I_A - **T**(I_B))

Maching – similarity score : Sim

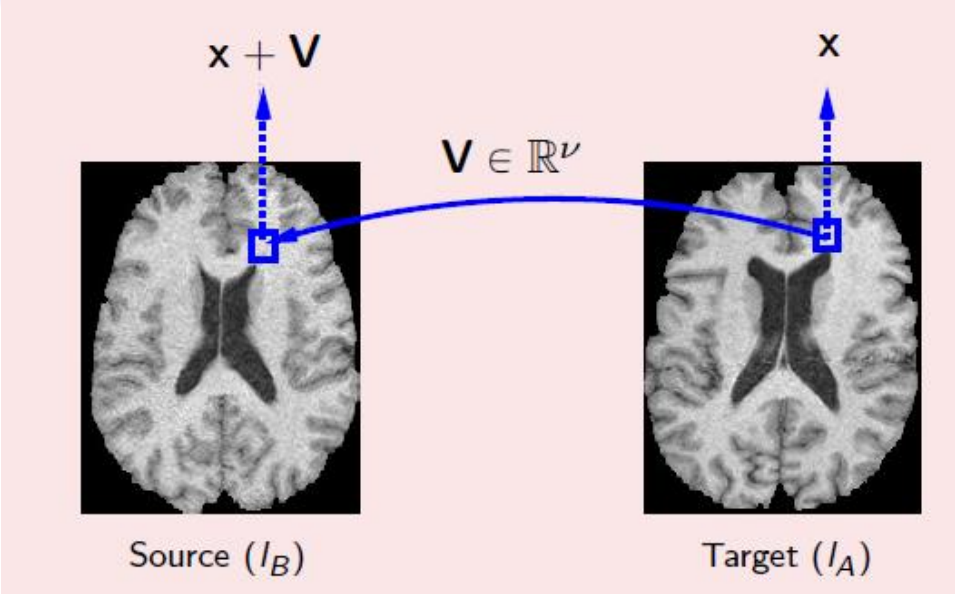
- depends on data
- simple – same type of data - SSD : sum $(I_A(x) - T(I_B(x)))^2$
- different illumination : NCC normalized cross correlation
- different imaging modalities : MI mutual information

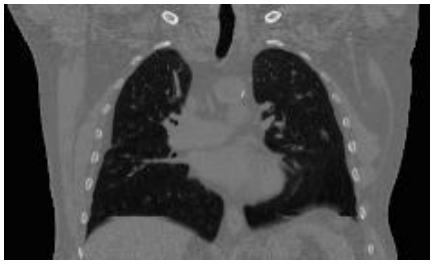


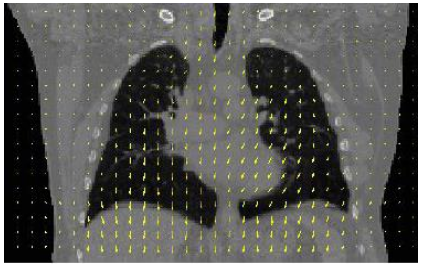
Transformation : T

- (linear) rigid, affine [ex. Same patient]
-(nonliear) image points are allowed to move differently

Non-rigid registration







Looking for a deformation field (vector field) v that will move each voxel in image A to the corresponding voxel in image B

$\min_{\mathbf{v}} \text{sum}_x (I_A(x) - I_B(x+\mathbf{v}))^2$

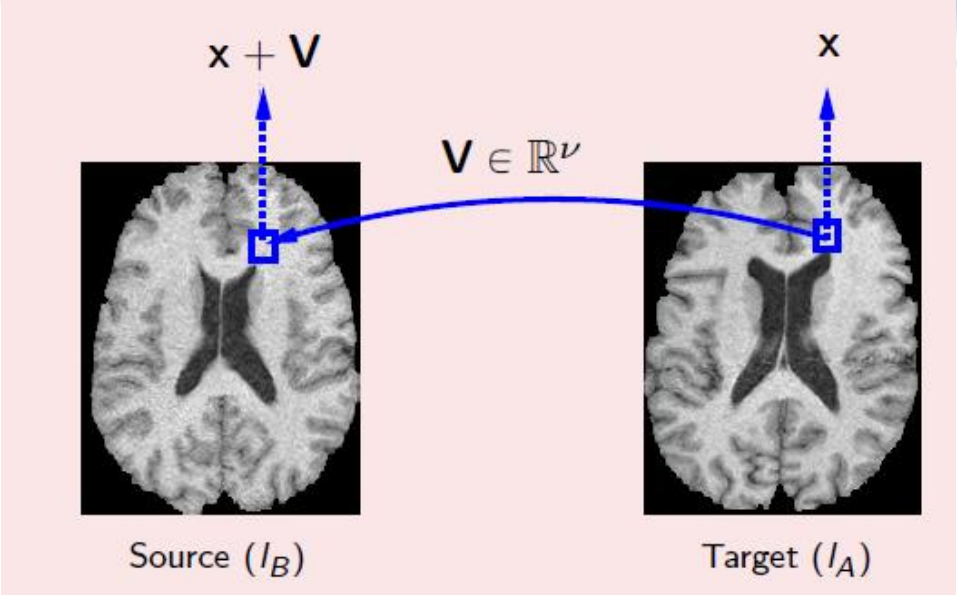
Gradient descent: solve for v iteratively adding small updates delta **δv**

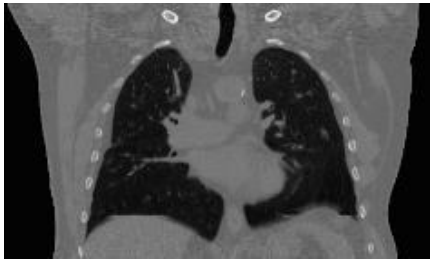
Each step is similar to an **optic flow problem**

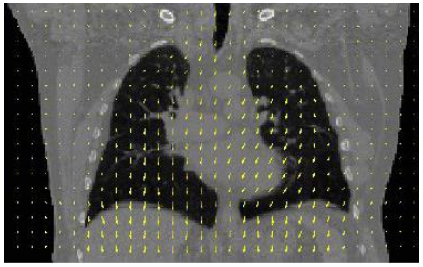
$\min_{\delta\mathbf{v}} \text{sum}_x (I_A(x) - I_B(x+\mathbf{v}+\delta\mathbf{v}))^2$

$\delta\mathbf{v} = -(I_A(x) - I_B(x+\mathbf{v}))/\text{grad}\, I_B(x+\mathbf{v})$

Non-rigid registration







Looking for a deformation field (vector field) v that will move each voxel in image A to the corresponding voxel in image B

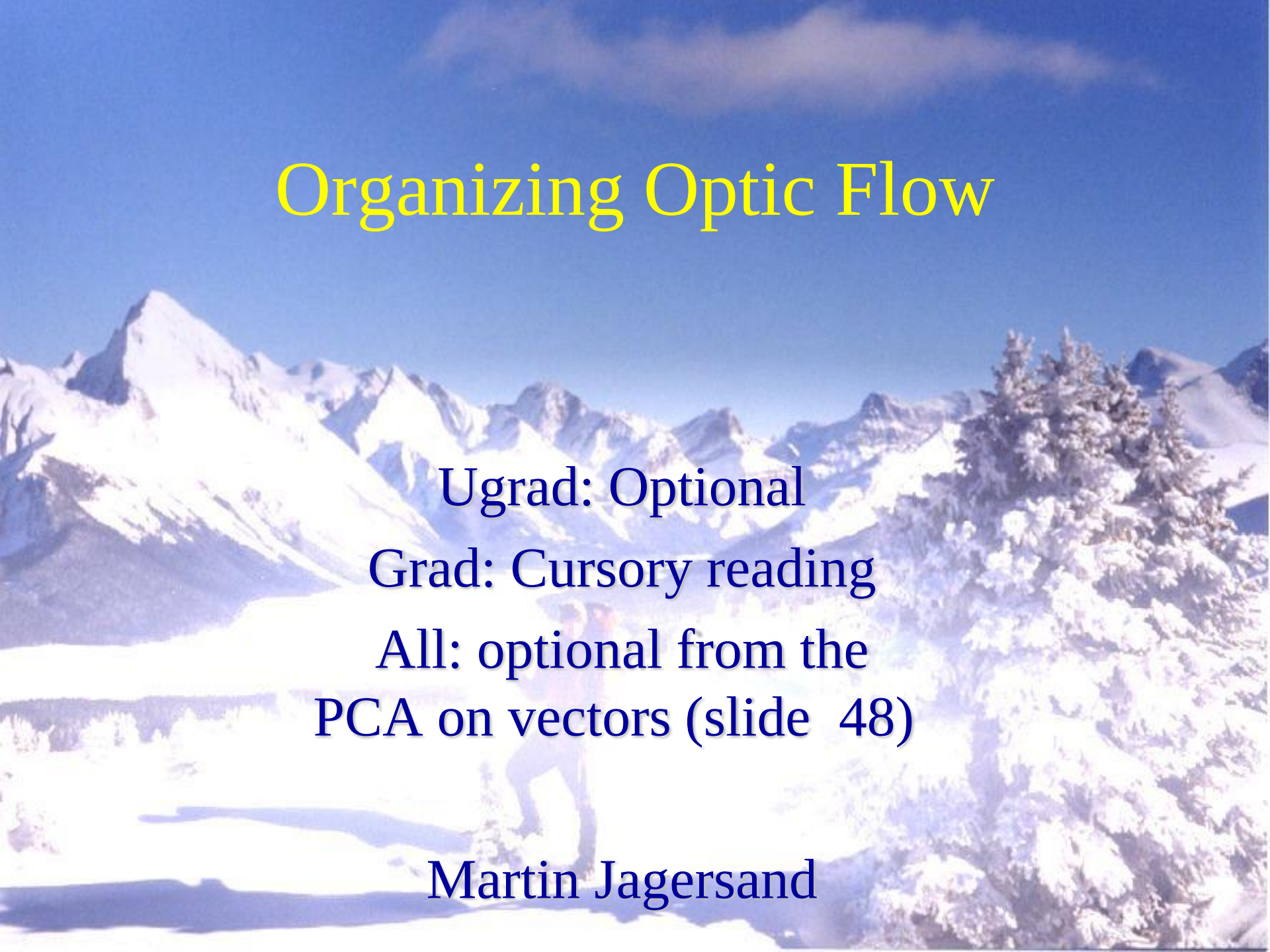
$\min_{\mathbf{v}} \text{sum}_x (I_A(x) - I_B(x+\mathbf{v}))^2$

Gradient descent: solve for v iteratively adding small updates delta **δv**

Each step is similar to an **optic flow problem**

In practice – motion between images is not small > needs **regularization** and **image pyramid** to solve robustly

$\min_{\mathbf{v}} \text{sum} (I_A(x) - I_B(x+\mathbf{v}(x)))^2 + R(v)$



Organizing Optic Flow

Ugrad: Optional

Grad: Cursory reading

All: optional from the PCA on vectors (slide 48)

Martin Jagersand

Organizing different kinds of motion

Two examples:

1. Greg Hager paper: Planar motion

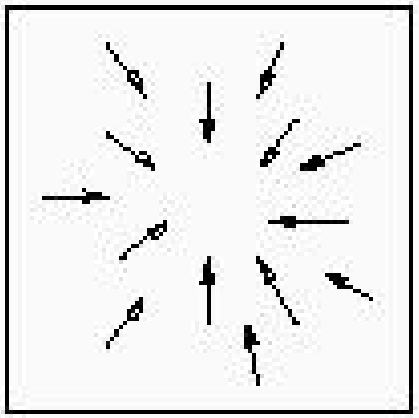
2. Mike Black, et al: Attempt to find a low dimensional subspace for complex motion

Remember: The optic flow field

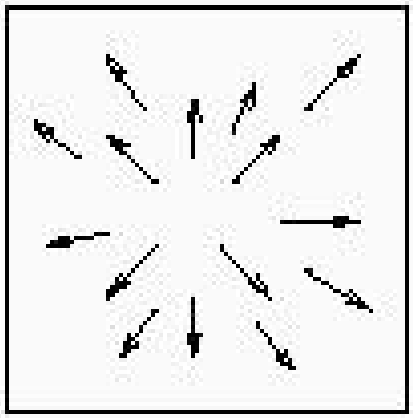
- Vector field over the image:

  [u,v] = f(x,y), u,v = Vel vector, x,y = Im pos

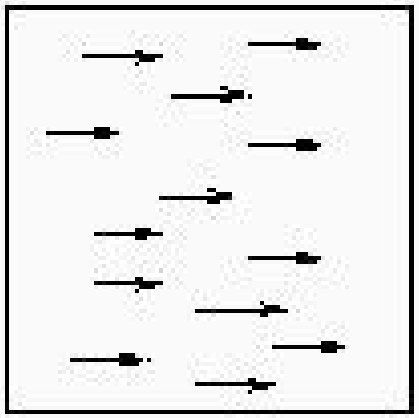
- **FOE, FOC** Focus of Expansion, Contraction



Zoom out



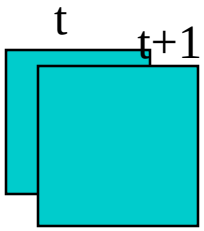
Zoom in



Pan Right to Left

Remember last lecture:

- Solving for the motion of a patch



Over determined equation system:

$$\begin{pmatrix} \vdots \\ -\frac{\partial \mathrm{Im}}{\partial t} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial \mathrm{Im}}{\partial x} & \frac{\partial \mathrm{Im}}{\partial y} \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

$$\mathrm{Im}_t = \mathrm{Mu}$$

- Can be solved in e.g. least squares sense using matlab $u = M \backslash \mathrm{Im}_t$

3-6D Optic flow

- Generalize to many freedooms (DOFs)

$$\delta I = \|I - I_i\|$$

$$\delta I = M\delta\mu$$

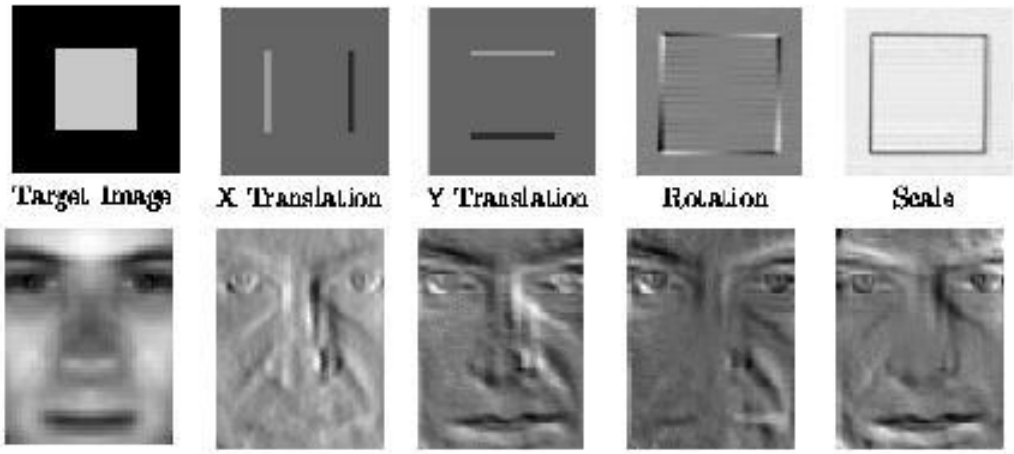
$$M = [I_x|I_y|I_r|I_s]$$

$$I_x = I(x,y) - I(x-1,y)$$

$$I_y = I(x,y) - I(x,y-1)$$

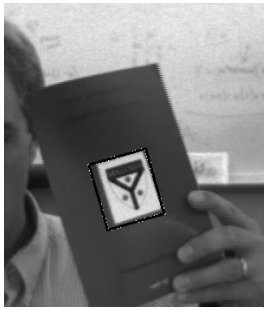
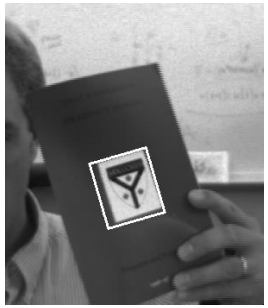
$$I_r = -yI_x + xI_y$$

$$I_s = \frac{1}{\sqrt{x^2+y^2}}(xI_x + yI_y)$$



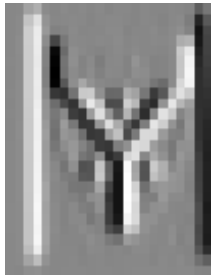
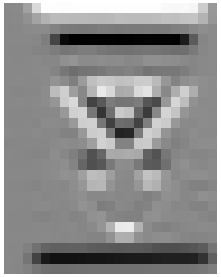
Im = Mu

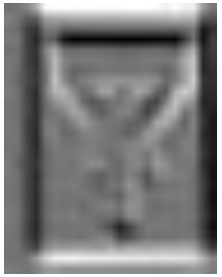
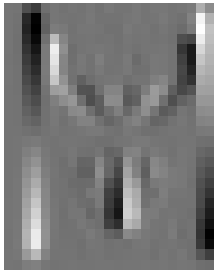
Example: All 6 freedoms


Template

Difference images $\mathrm{M(u)} = \partial \mathrm{Im} / \partial \mathrm{u}$

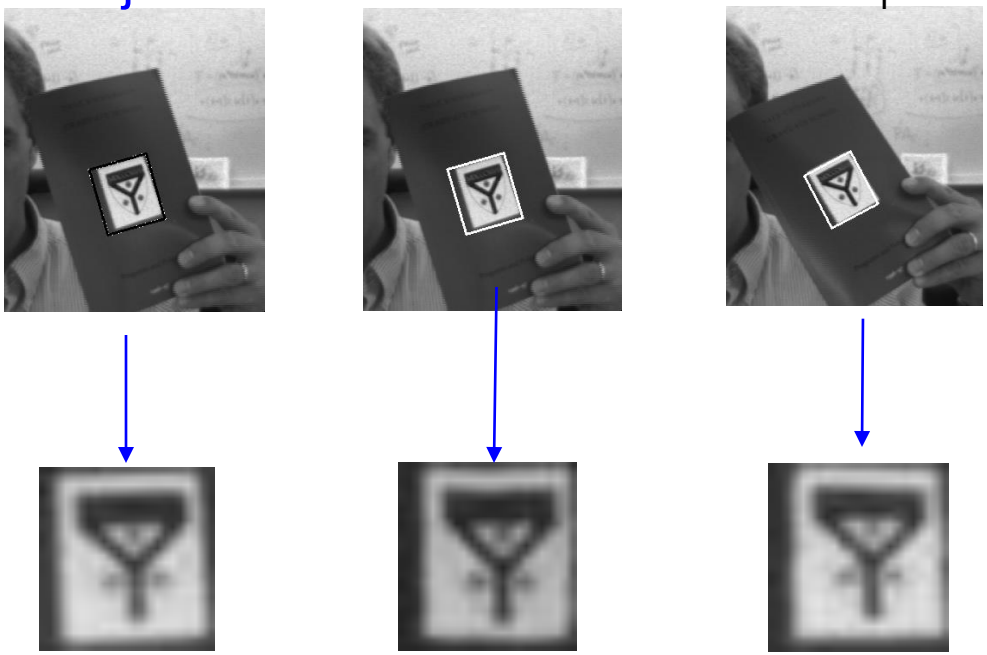



X **Y** **Rotation** **Scale** **Aspect** **Shear**

Know what type of motion
(Greg Hager, Peter Belhumeur)

E.g. Planar Object => Affine motion model: $u'_i = A u_i + d$



$$I_t = g(p_t, I_0)$$

Mathematical Formulation

- Define a "warped image" g
  - f(p,x) = x' (warping function), p warp parameters
  - I(x,t) (image a location x at time t)
  - $g(p,I_t) = (I(f(p,x_1),t), I(f(p,x_2),t), \ldots I(f(p,x_N),t))'$
- Define the Jacobian of warping function
  - $M(p,t) = \left[\frac{\partial \mathbf{I}}{\partial p}\right]$
- Model
  - $I_0 = g(p_t, I_t)$ (image I, variation model g, parameters p)
  - $\Delta I = \mathbf{M}(p_t, I_t)\,\Delta p$ (local linearization **M**)
- Compute motion parameters
  - $\Delta p = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \Delta I$ where $\mathbf{M} = \mathbf{M}(p_t, I_t)$ (Remember solve with QR or SVD)

Planar 3D motion

From geometry we know that the correct plane-to-plane transform is

1. for a perspective camera the ***projective homography***

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \mathcal{W}_h(\mathbf{x}_h, \mathbf{h}) = \frac{1}{1+h_7u+h_8v} \begin{bmatrix} h_1u & h_3v & h_5 \\ h_2u & h_4v & h_6 \end{bmatrix}$$

1. for a linear camera (orthographic, weak-, para-perspective) the ***affine warp***

$$\begin{bmatrix} u_w \\ v_w \end{bmatrix} = \mathcal{W}_a(\mathbf{p}, \mathbf{a}) = \begin{bmatrix} \mathbf{a_3} & \mathbf{a_4} \\ \mathbf{a_5} & \mathbf{a_6} \end{bmatrix} \mathbf{p} + \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}$$

Planar Texture Variability 1
Affine Variability

- Affine warp function

$$\begin{bmatrix} u_w \\ v_w \end{bmatrix} = \mathcal{W}_a(\mathbf{p}, \mathbf{a}) = \begin{bmatrix} \mathbf{a_3} & \mathbf{a_4} \\ \mathbf{a_5} & \mathbf{a_6} \end{bmatrix} \mathbf{p} + \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}$$

- Corresponding image variability

$$\Delta \mathbf{I}_a = \sum_{i=1}^{6} \frac{\partial}{\partial a_i} \mathbf{I}_w \Delta a_i = \left[\frac{\partial I}{\partial u}, \frac{\partial I}{\partial v}\right] \begin{bmatrix} \frac{\partial u}{\partial a_1} & \cdots & \frac{\partial u}{\partial a_6} \\ \frac{\partial v}{\partial a_1} & \cdots & \frac{\partial v}{\partial a_6} \end{bmatrix} \begin{bmatrix} \Delta a_1 \\ \vdots \\ \Delta a_6 \end{bmatrix}$$

- Discretized for images

$$\Delta \mathbf{I_a} = \left[\frac{\partial \mathbf{I}}{\partial \mathbf{u}}, \frac{\partial \mathbf{I}}{\partial \mathbf{v}}\right] \begin{bmatrix} \mathbf{1} & \mathbf{0} & * \mathbf{u} & \mathbf{0} & * \mathbf{v} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & * \mathbf{u} & \mathbf{0} & * \mathbf{v} \end{bmatrix} \begin{bmatrix} \mathbf{y_1} \\ \vdots \\ \mathbf{y_6} \end{bmatrix}$$

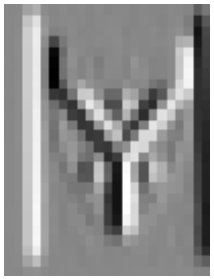
$$= [\mathbf{B}_1 \ldots \mathbf{B}_6][y_1, \ldots, y_6]^T = B_a \mathbf{y_a}$$

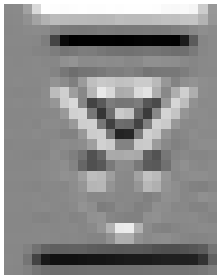
On The Structure of M

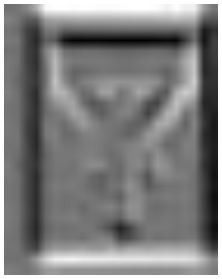
Planar Object + linear (infinite) camera

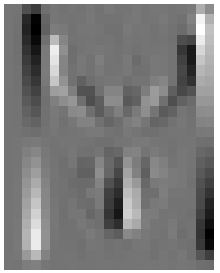
-> Affine motion model $u'_i = \mathbf{A}\, u_i + d$

$$M(p) = \partial g / \partial p$$







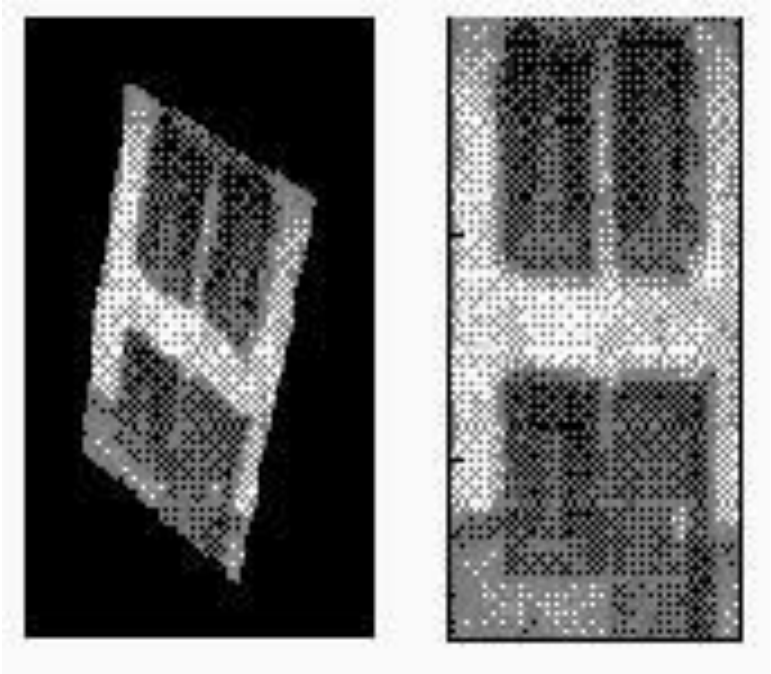


X **Y** **Rotation** **Scale** **Aspect** **Shear**

$$\begin{bmatrix} a_3 & a_4 \\ a_5 & a_6 \end{bmatrix} = sR(\Theta) \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ h & 1 \end{bmatrix}$$

Planar motion under perspective projection

- Perspective plane-plane transforms defined by homographies



$$\begin{bmatrix} u_{1t_1} & \cdots & u_{Nt_1} \\ v_{1t_1} & \cdots & v_{Nt_1} \\ \xi_{1t_1} & \cdots & \xi_{Nt_1} \end{bmatrix} = H \begin{bmatrix} u_{1t_2} & \cdots & u_{Nt_2} \\ v_{1t_2} & \cdots & v_{Nt_2} \\ 1 & \cdots & 1 \end{bmatrix}$$

Planar Texture Variability 2
Projective Variability

- Homography warp

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \mathcal{W}_h(\mathbf{x}_h, \mathbf{h}) = \frac{1}{1+h_7u+h_8v} \begin{bmatrix} h_1u & h_3v & h_5 \\ h_2u & h_4v & h_6 \end{bmatrix}$$

- Projective variability:

$$\Delta \mathbf{I}_h = \frac{1}{c_1}\left[\frac{\partial \mathbf{I}}{\partial u}, \frac{\partial \mathbf{I}}{\partial v}\right] \begin{bmatrix} u & 0 & v & 0 & 1 & 0 & -\frac{uc_2}{c_1} & -\frac{vc_2}{c_1} \\ 0 & u & 0 & v & 0 & 1 & -\frac{uc_3}{c_1} & -\frac{vc_3}{c_1} \end{bmatrix} \begin{bmatrix} \Delta h_1 \\ \vdots \\ \Delta h_8 \end{bmatrix}$$

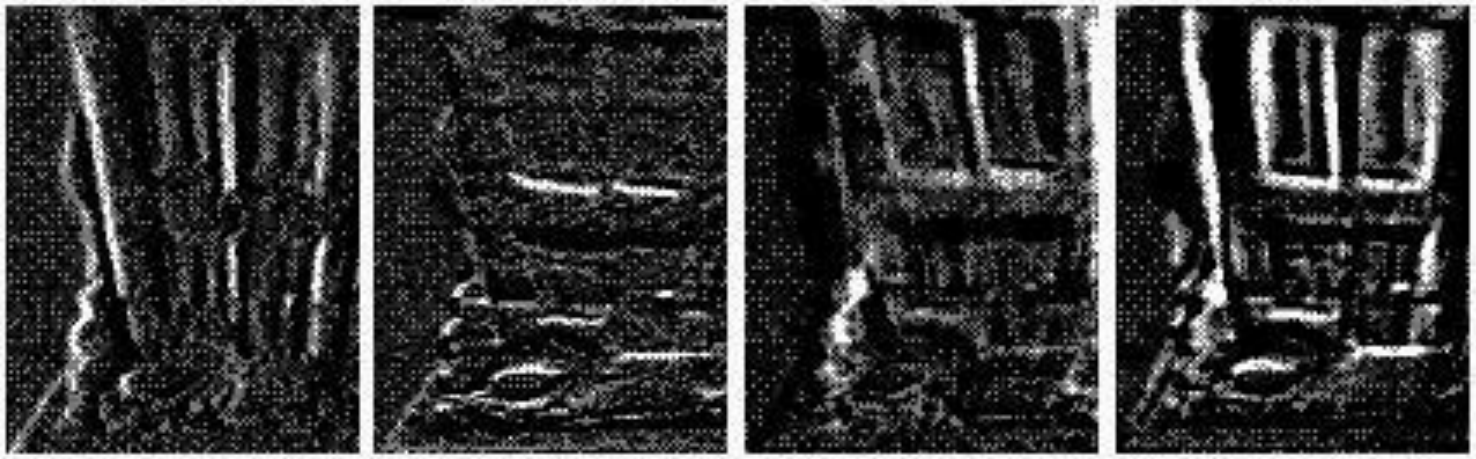
$$= [\mathbf{B}_1 \ldots \mathbf{B}_8][y_1, \ldots, y_8]^T = B_h \mathbf{y}_h$$

- Where $c_1 = 1 + h_7u + h_8v$, $c_2 = h_1u + h_3v + h_5$ and $c_3 = h_2u + h_4v + h_6$

Planar-perspective motion 3

- In practice hard to compute 8 parameter model stably from one image, and impossible to find out-of plane variation
- Estimate variability basis from several images:

Computed Estimated

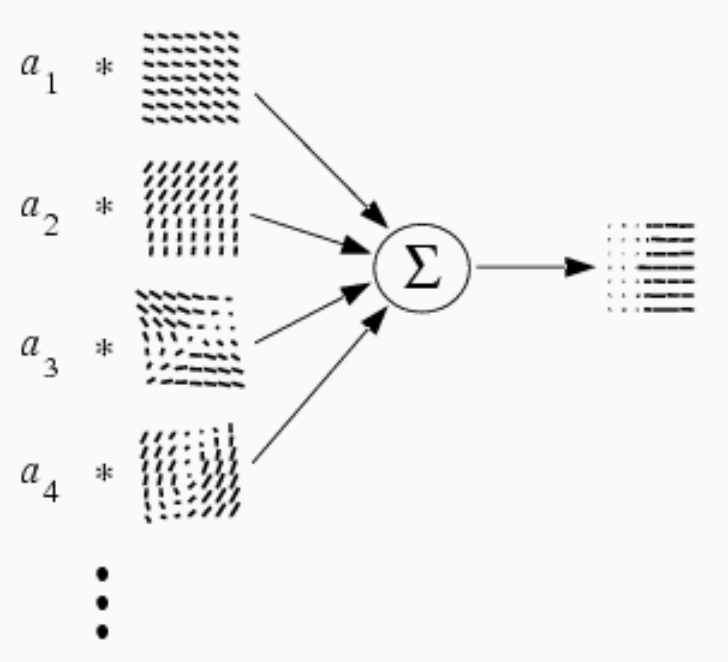


Another idea Black, Fleet) Organizing flow fields

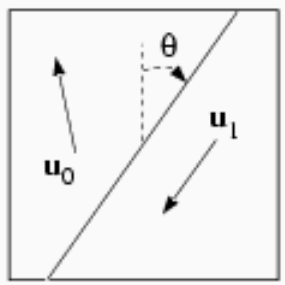
- Express flow field f in subspace basis m

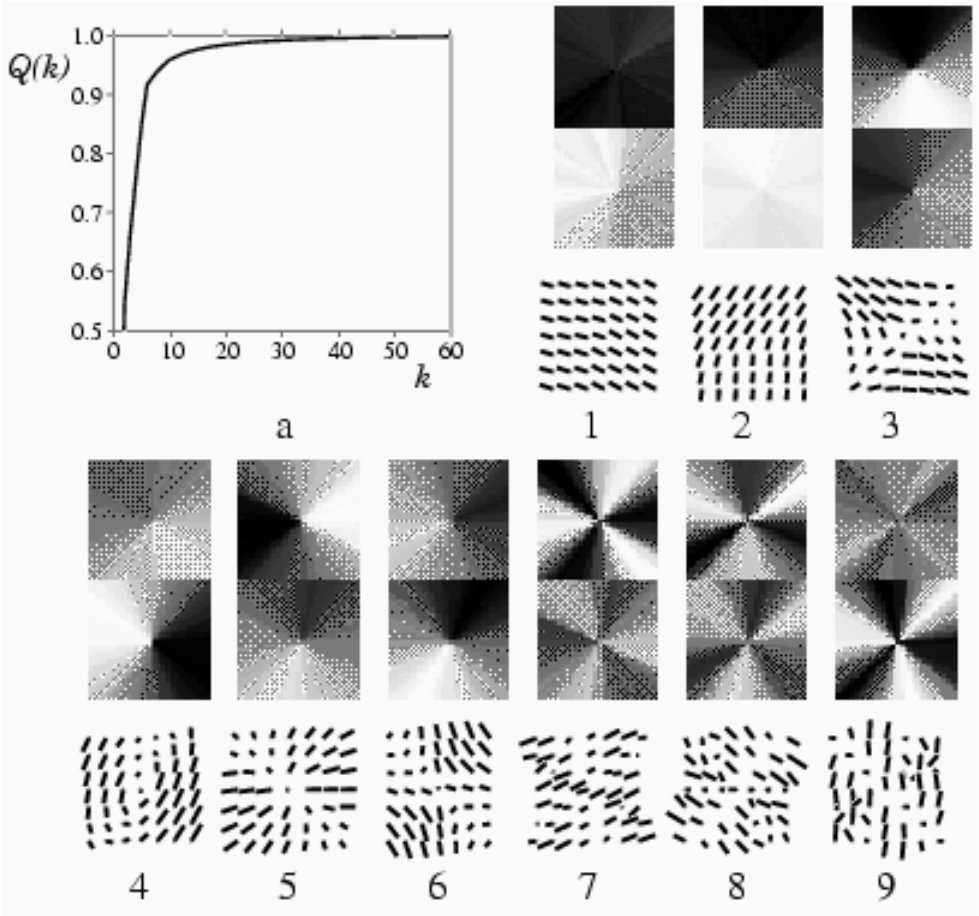
$$\vec{f}_k = \sum_{i=1}^{k} a_i \vec{m}_i$$

- Different "mixing" coefficients a correspond to different motions



Imag





Mathematical formulation

Let: $\rho(r,\sigma) = r^2/(\sigma^2+r^2)$

Robust error norm

Mimimize objective function:

Motion basis

$$E(\vec{b};\vec{a}) = \sum_{\vec{x}\in R} \rho(I(\vec{x}+\vec{u}(\vec{x};\vec{a}+\vec{b}),t+1)-I(\vec{x},t),\ \sigma)$$

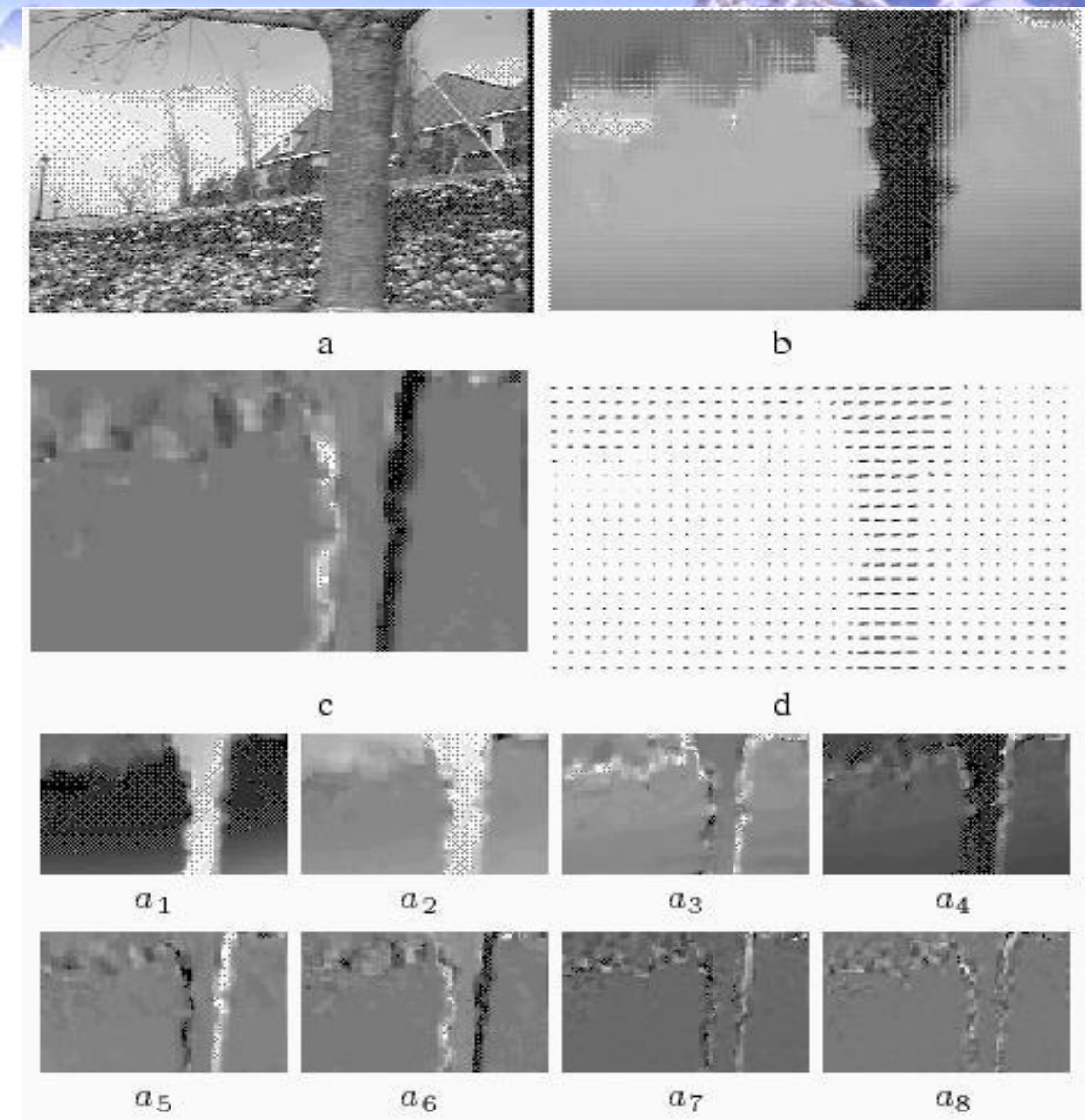
=

$$\sum_{\vec{x}\in R} \rho(\vec{u}(\vec{x};\vec{b})\cdot\vec{\nabla}I(\vec{x}+\vec{u}(\vec{x};\vec{a}),t+1)+r(\vec{x},\vec{a}),\ \sigma)$$

Where $\vec{\nabla}I(\vec{x}+\vec{u}(\vec{x};\vec{a}),t+1) = [I_x, I_y]^T$

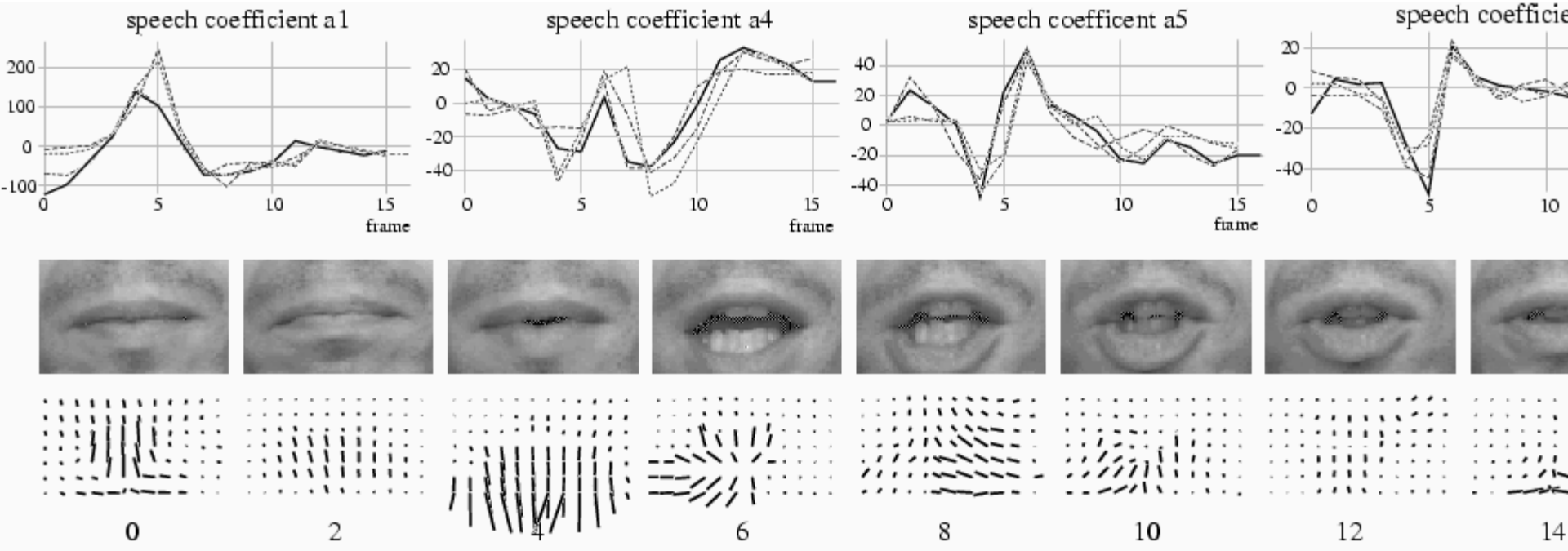
Experiment
Moving camera

- 4x4 pixel patches
- Tree in foreground separates well



Experiment: Characterizing lip motion

- Very non-rigid!



Questions to think about

Readings: Book chapter, Fleet et al. paper.

Compare the methods in the paper and lecture

1. Any major differences?
2. How dense flow can be estimated (how many flow vectore/area unit)?
3. How dense in time do we need to sample?

Summary

- Three types of visual motion extraction
  1. Optic (image) flow: Find x,y – image velocities
  2. 3-6D motion: Find object pose change in image coordinates based more spatial derivatives (top down)
  3. Group flow vectors into global motion patterns (bottom up)
- Visual motion still not satisfactorily solved problem

(Parenthesis)
Euclidean world motion -> image

Let us assume there is one rigid object moving with velocity T and $w = d R / dt$

For a given point P on the object, we have

$p = f P/z$

The apparent velocity of the point is

$V = -T - w \times P$

Therefore, we have $v = dp/dt = f (z V - V_z P)/z^2$

Component wise:

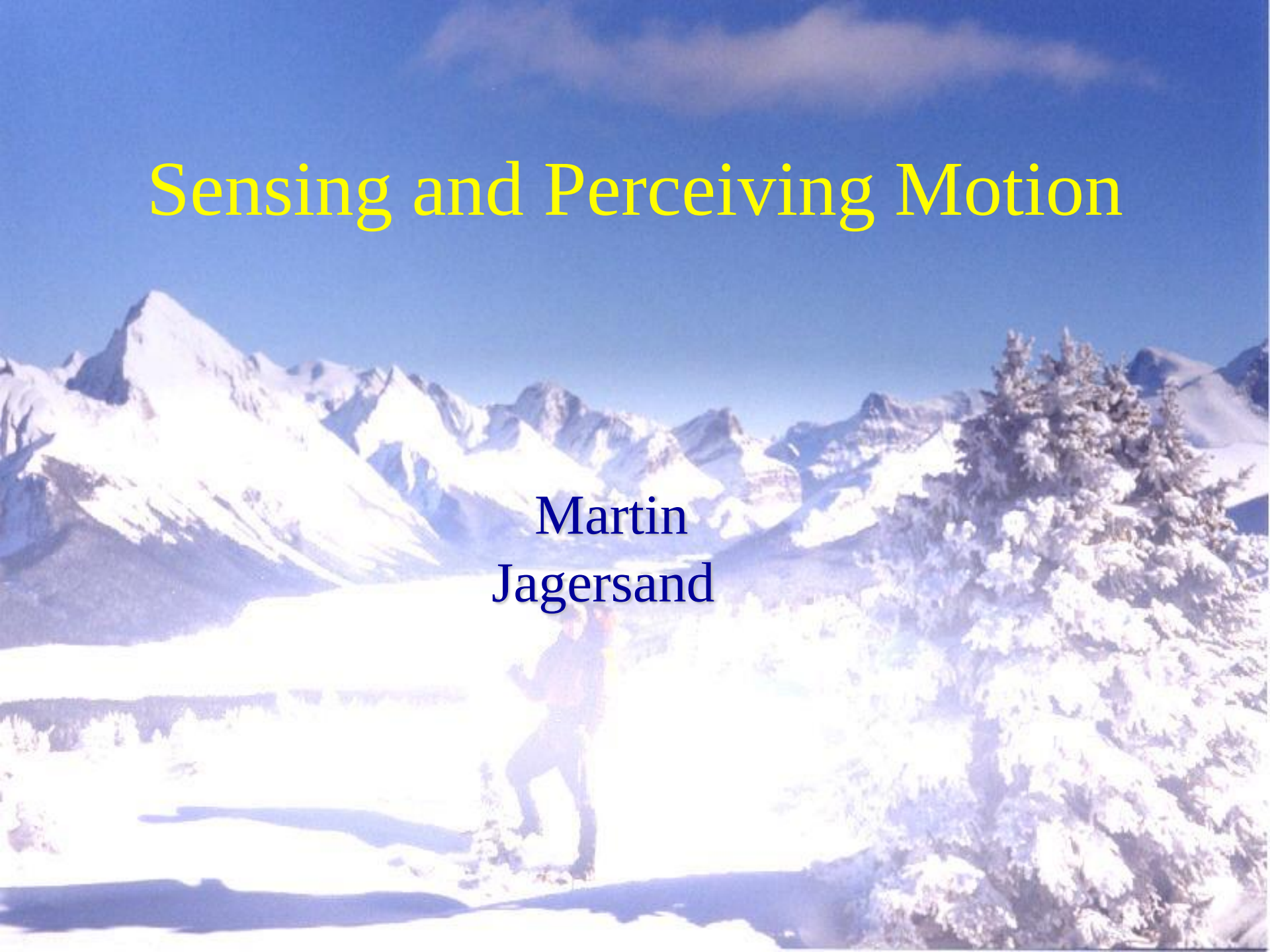
$$v_x = f\frac{T_z x - T_x f}{z} - w_y f + w_z y + \frac{w_x xy - w_y x^2}{f}$$

$$v_y = f\frac{T_z x - T_y f}{z} - w_x f + w_z x + \frac{w_y xy - w_x y^2}{f}$$

Motion due to translation: depends on depth

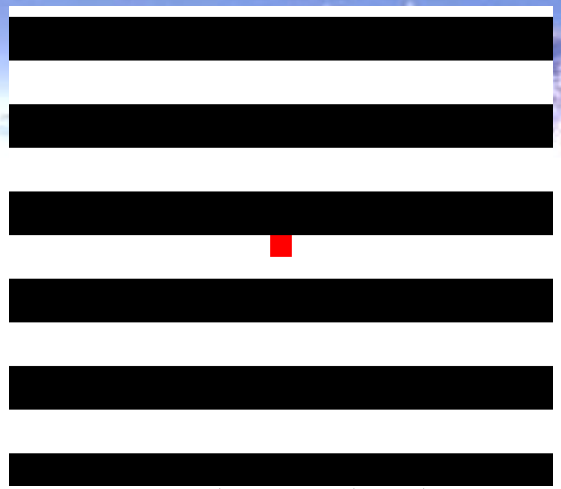
Motion due to rotation: independent of depth

Sensing and Perceiving Motion



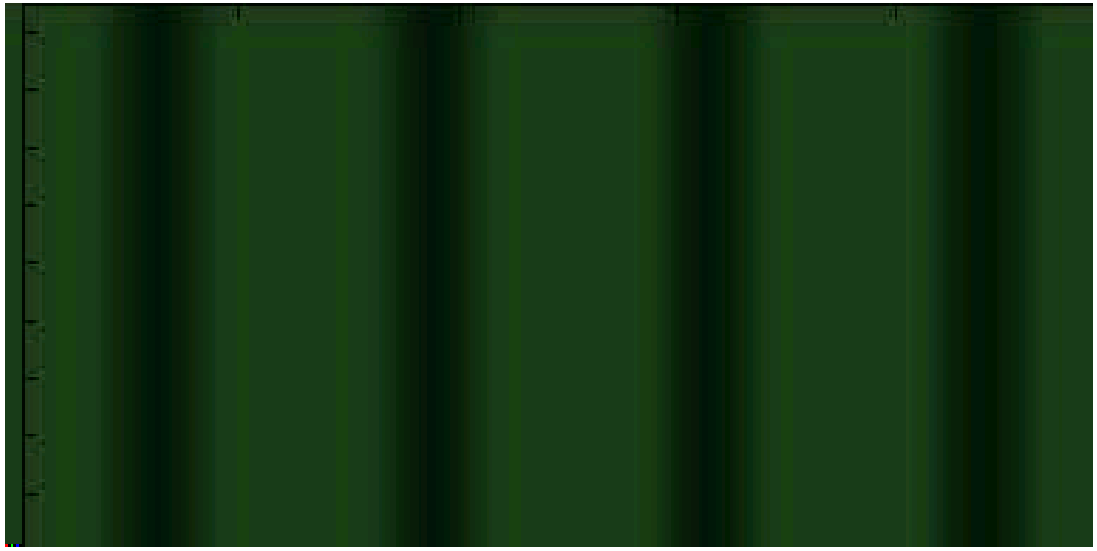
Martin
Jagersand

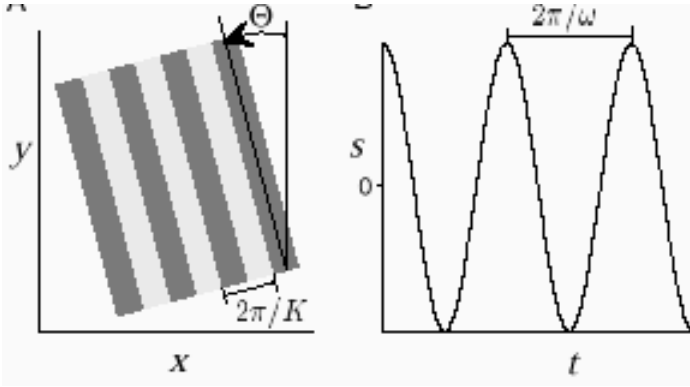
Counterphase sin grating



- Spatio-temporal pattern
  - Time t, Spatial x,y

$$s(x, y, t) = A\cos(Kx\cos\Theta + Ky\sin\Theta - \Phi)\cos(\omega t)$$





Counterphase sin g

- Spatio-temporal pattern
  - Time t, Spatial x,y

$$s(x, y, t) = A\cos(Kx\cos\Theta + Ky\sin\Theta - \Phi)\cos(\omega t)$$

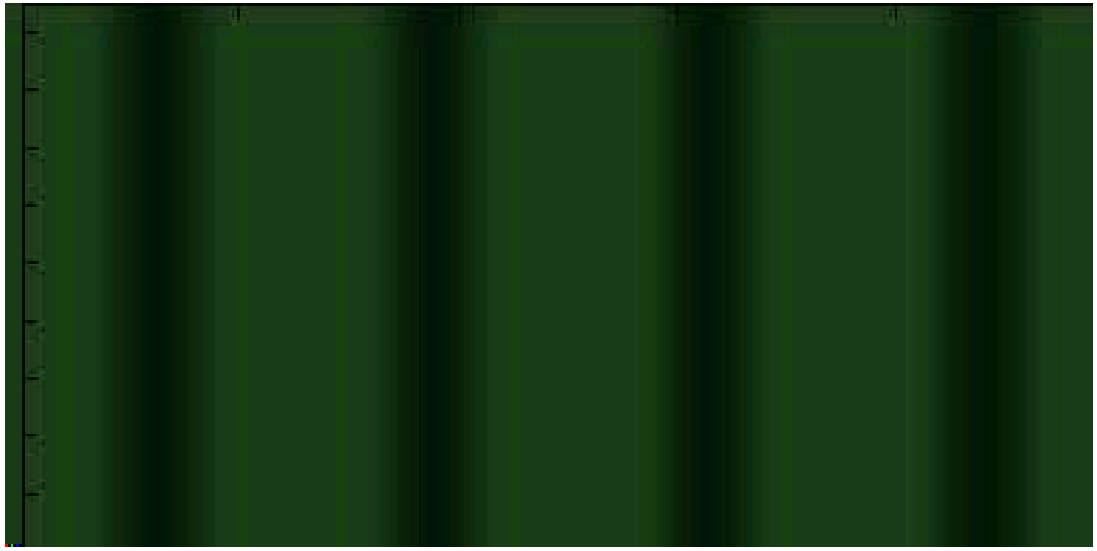
Rewrite as dot product:

$$\tfrac{1}{2}(\cos([a, b]\begin{bmatrix} x \\ y \end{bmatrix} - \omega t) + \cos([a, b]\begin{bmatrix} x \\ y \end{bmatrix} + \omega t)$$

Result: Standing wave is superposition of two moving waves

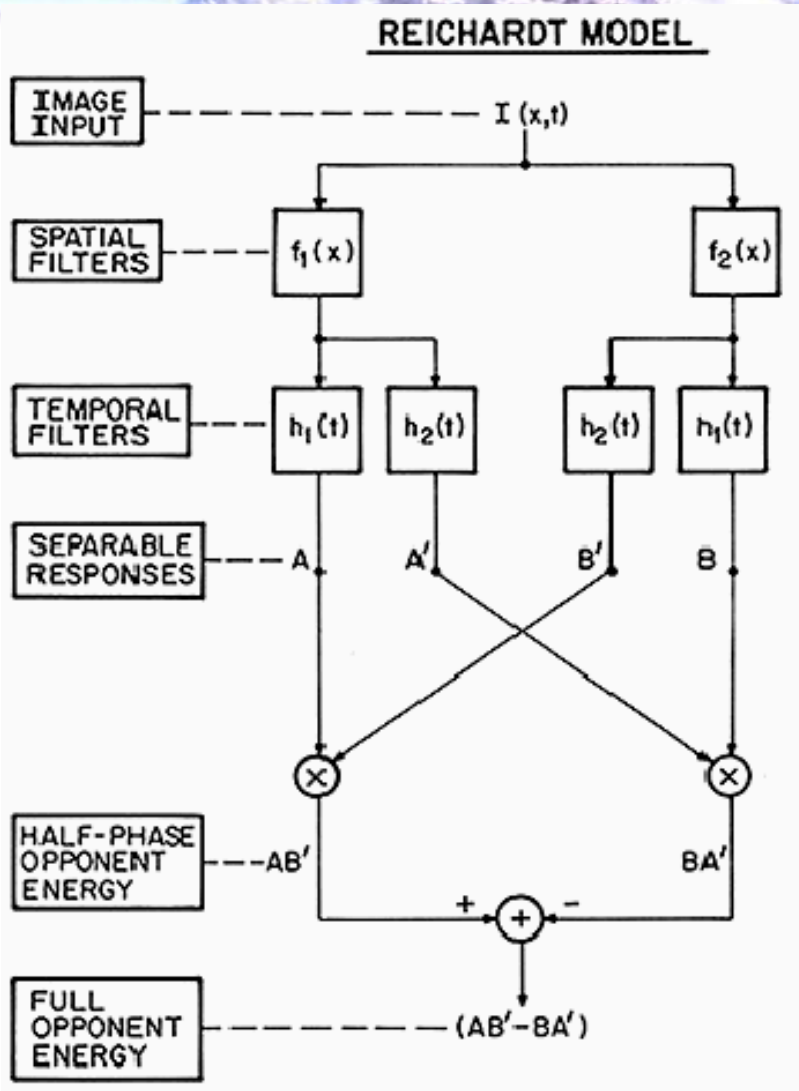
Analysis:

- Only one term: Motion left or right
- Mixture of both: Standing wave
- Direction can flip between left and right



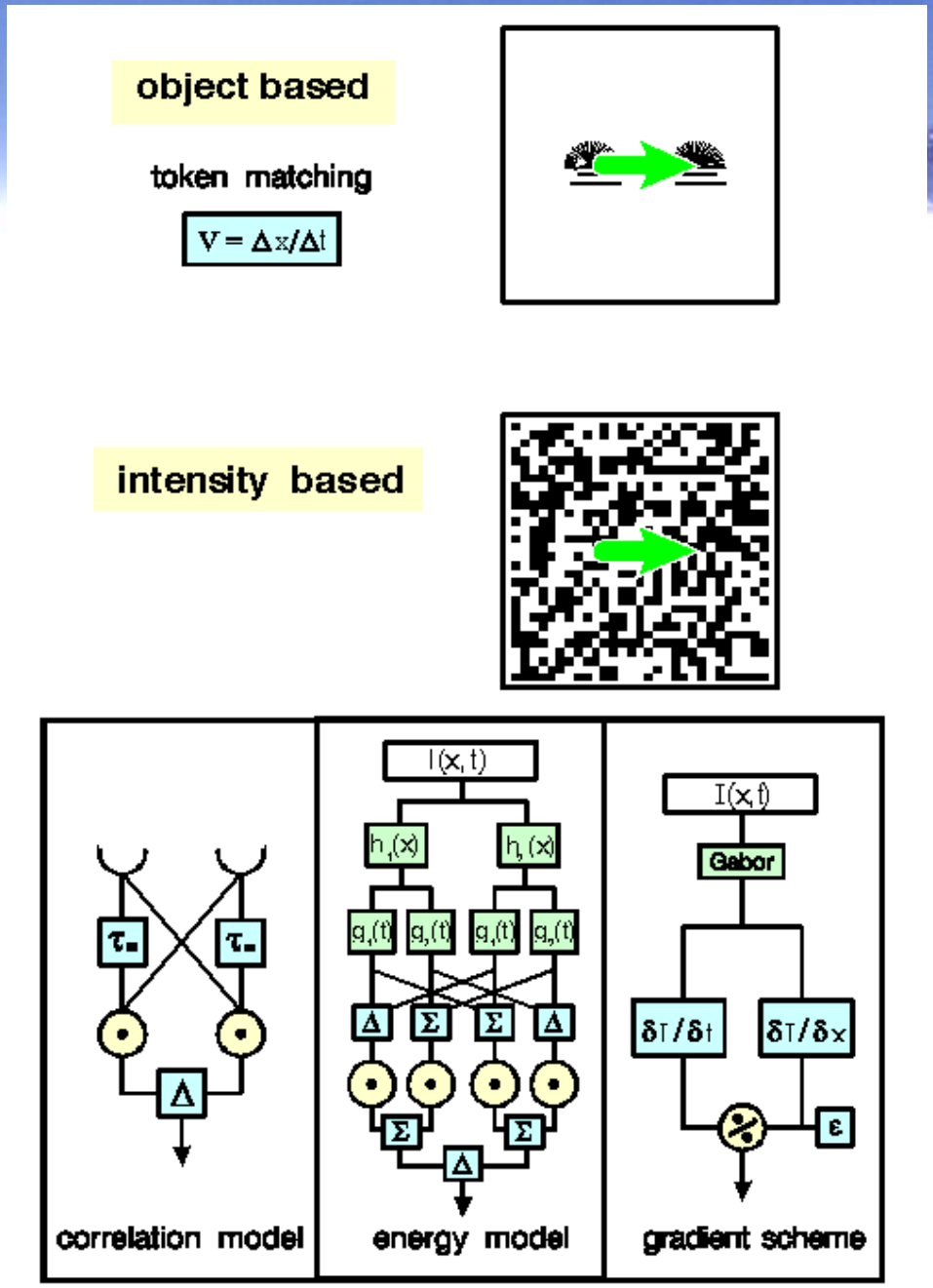
Reichardt detector

- QT movie



Several motion models

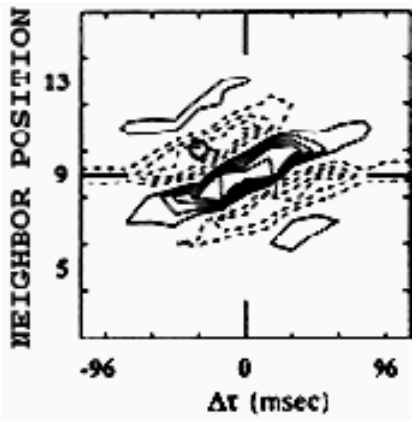
- Gradient: in Computer Vision
- Correlation: In bio vision
- Spatiotemporal filters: Unifying model

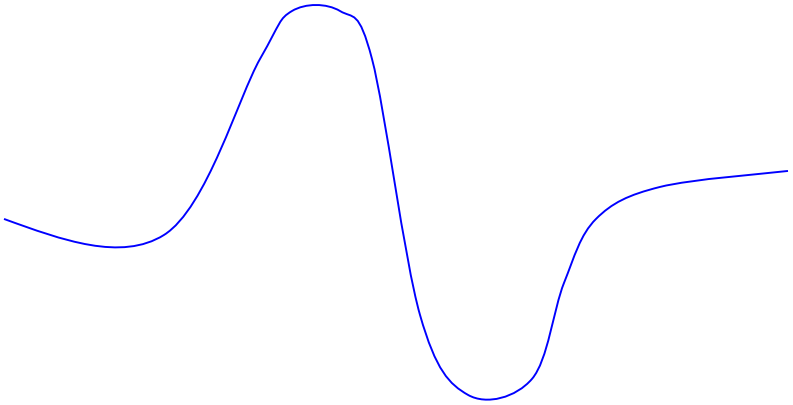


Spatial response: Gabor function

- Definition:

$$D_s(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right) \cos(kx - \phi)$$





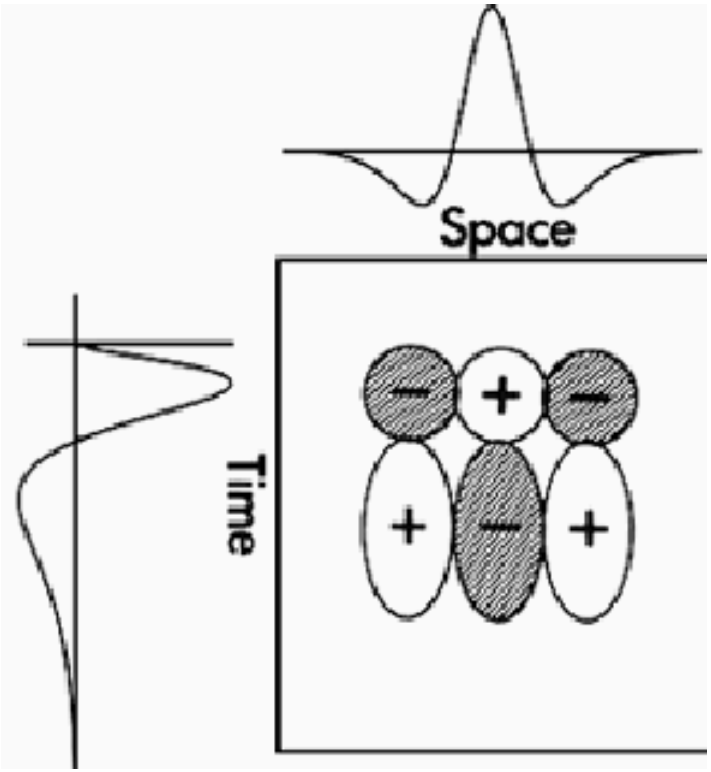
Temporal response:

Adelson, Bergen ’85

$$D_t(\tau) = \alpha \exp(-\alpha\tau)\left(\frac{(\alpha\tau)^5}{5!} - \frac{(\alpha\tau)^7}{7!}\right)$$

Note: Terms from taylor of sin(t)

Spatio-temporal $D=D_sD_t$



Receptor response to Counterphase grating

- Separable convolution

$$L(t) = L_s L_t(t),$$

$$L_s = \int dxdy\, D_s(x, y)\, A \cos\left(Kx\cos(\Theta) + Ky\sin(\Theta) - \Phi\right)$$

$$L_t(t) = \int_0^\infty d\tau\, D_t(\tau) \cos\left(\omega(t-\tau)\right).$$

Simplified:

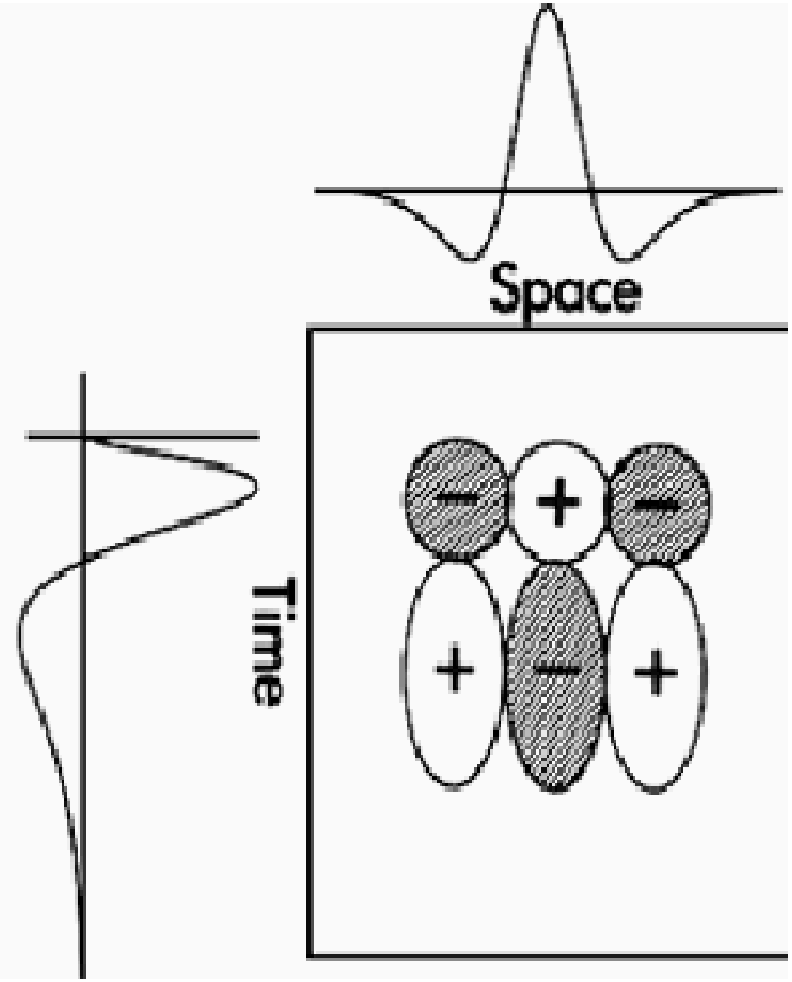
- For our grating: (Theta=0)

$$L_s = \frac{\mathrm{A}}{2}\exp\left(\frac{-\sigma^2(k-K)^2}{2}\right)\cos(\phi - \Phi)$$

- Write as sum of components:

= exp(…)*(acos… + bsin…)

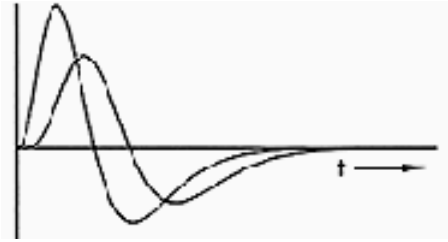
Space-time receptive field



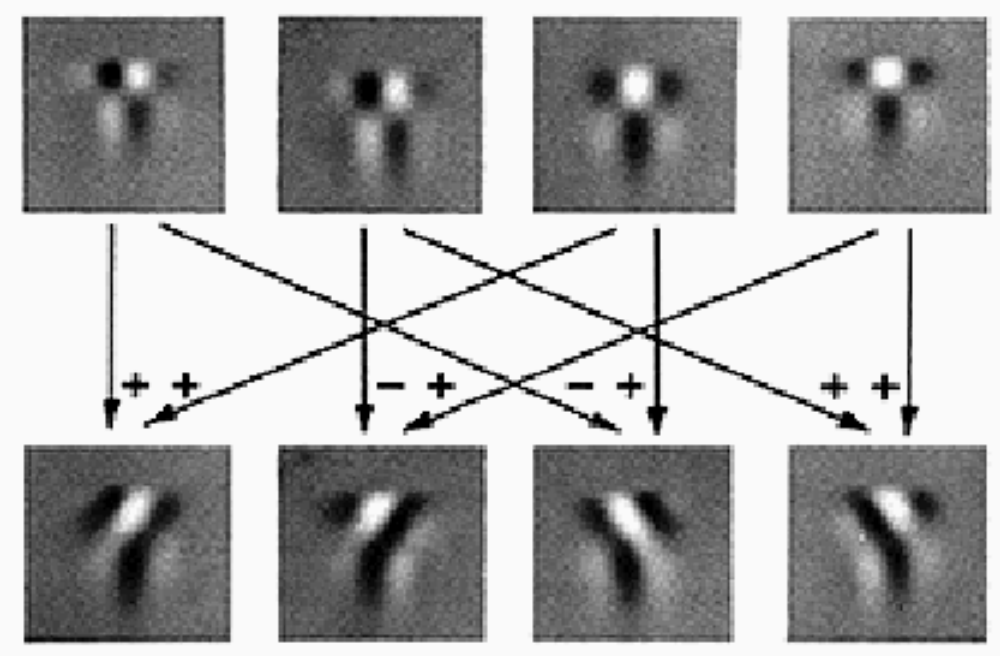
Combined cells

- Spat:

Temp:



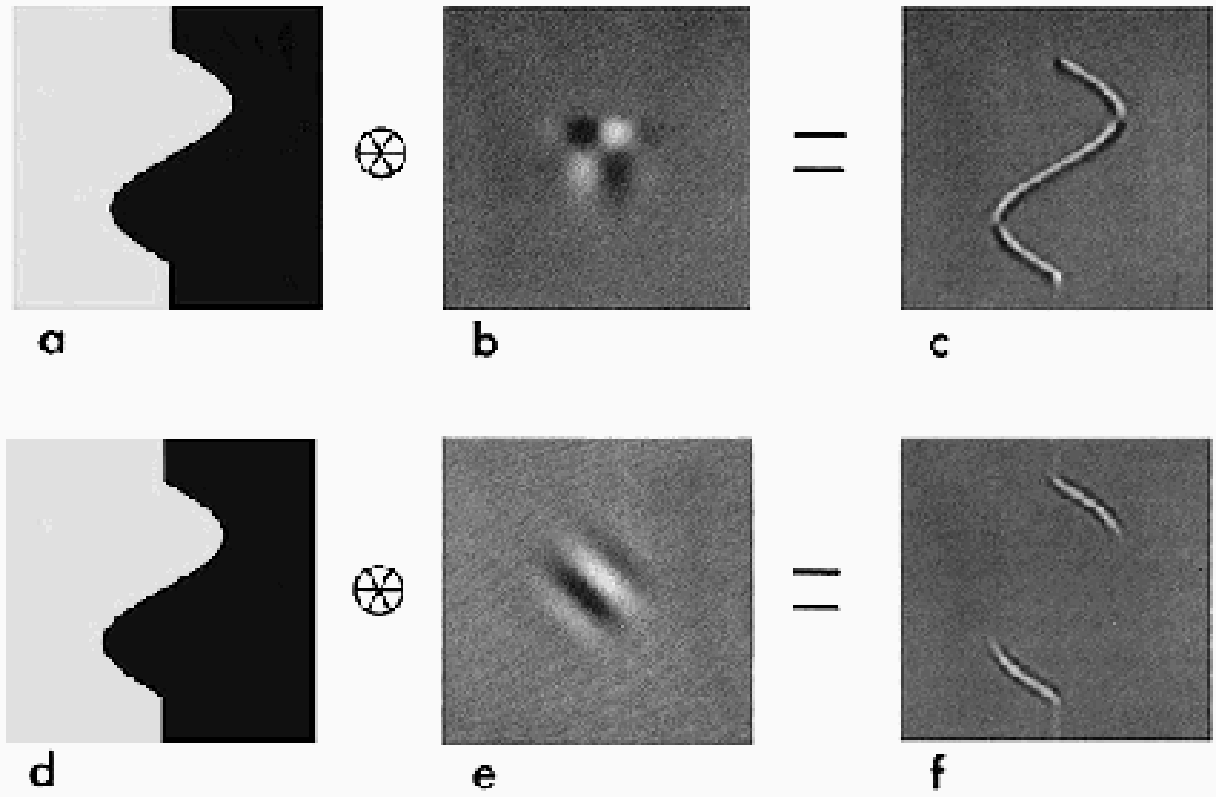
- Both:



- Comb:

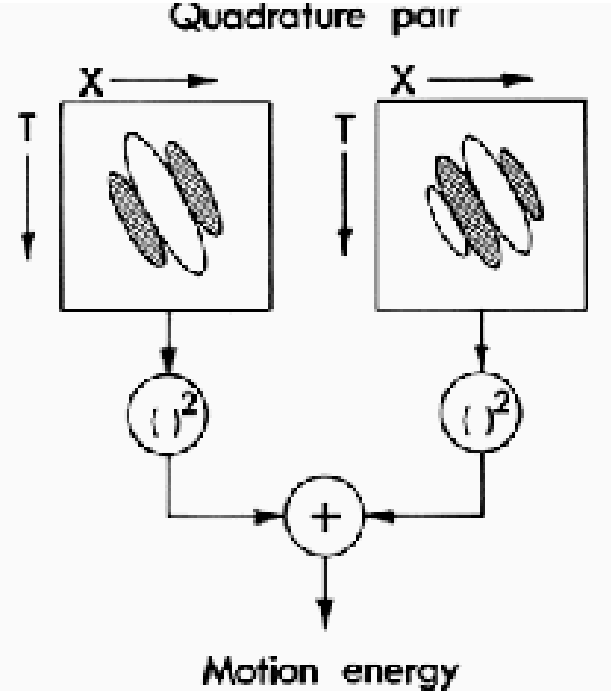
Result:

- More directionally specific response

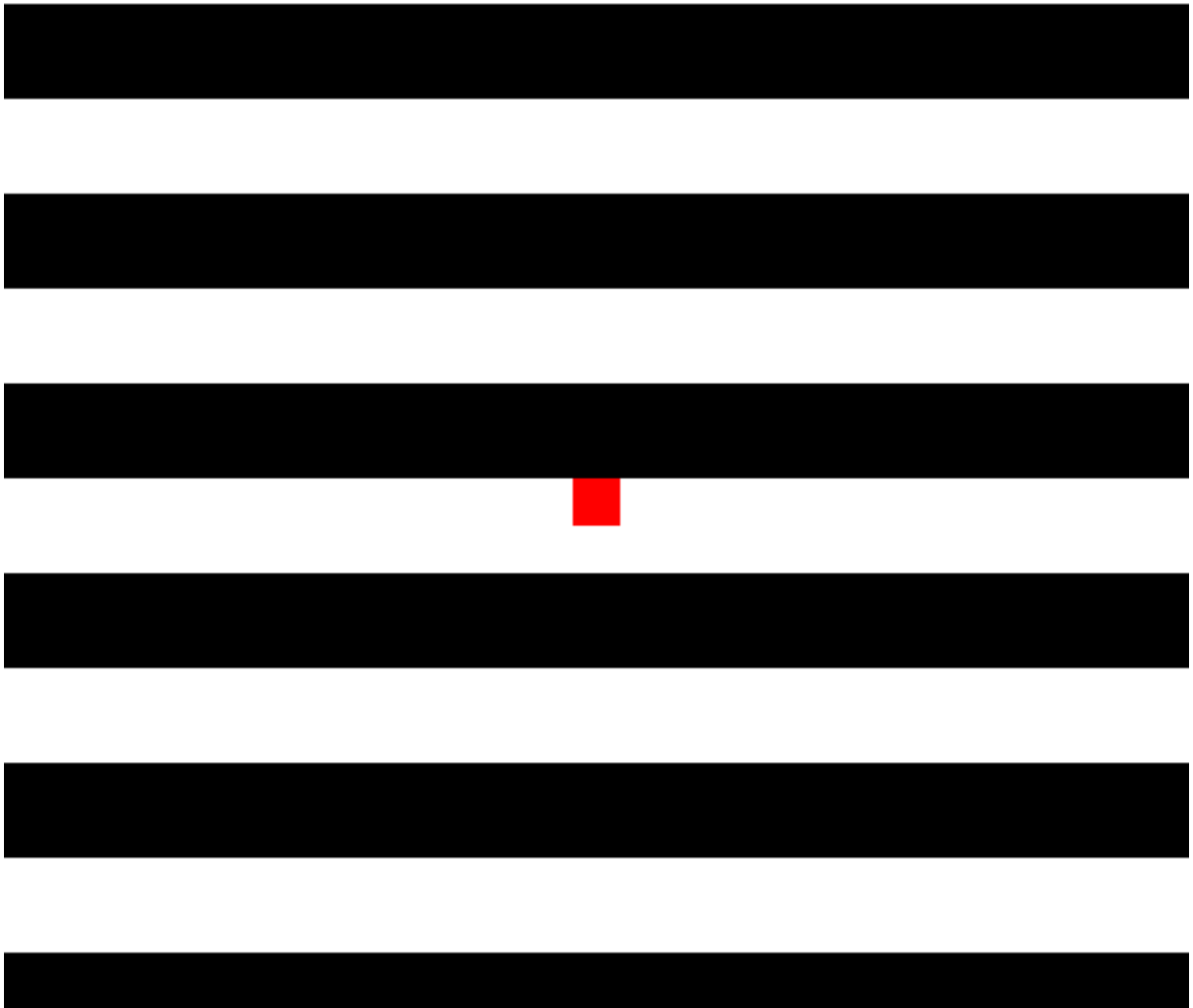


Energy model:

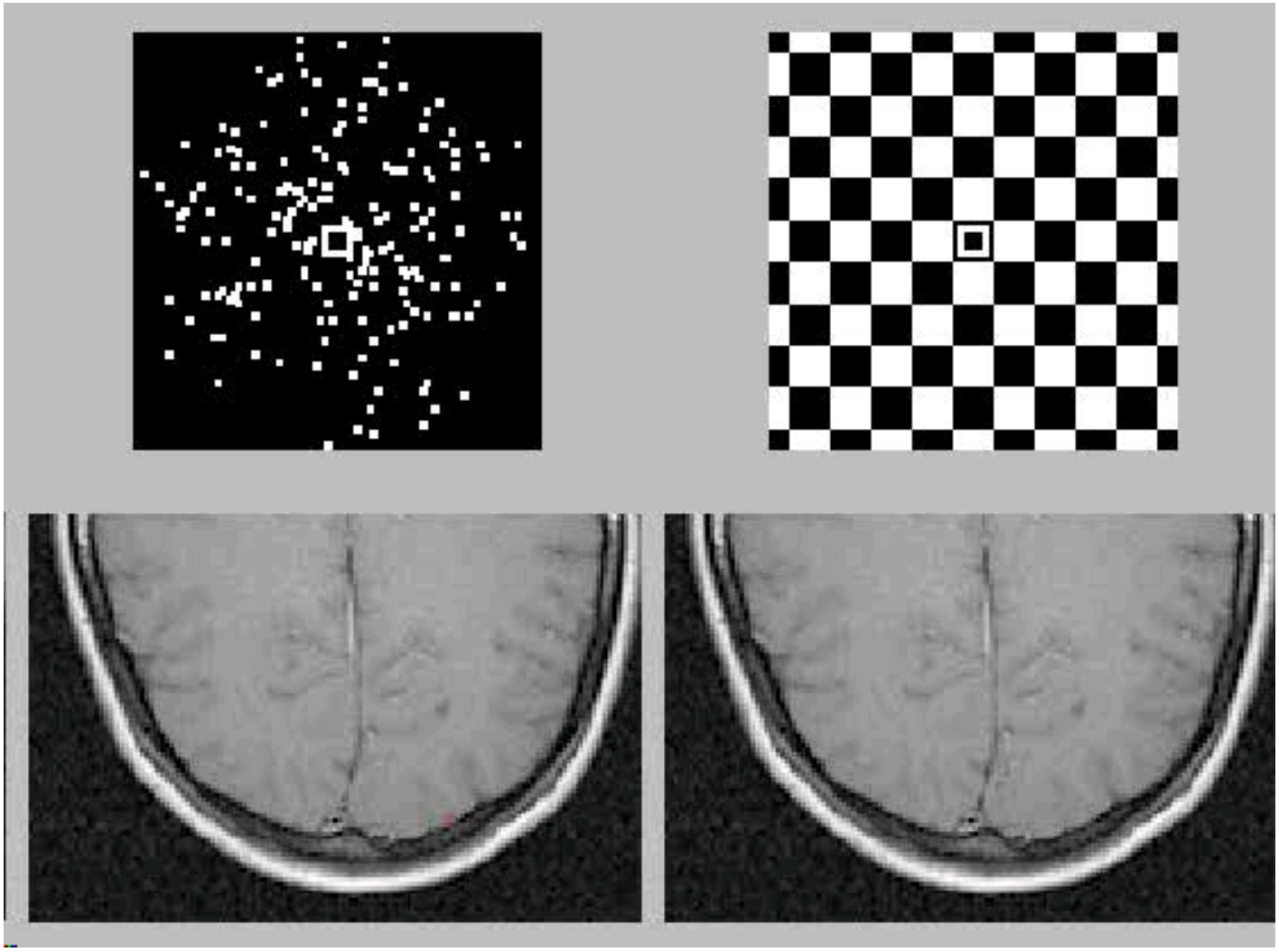
- Sum odd and even phase components
- Quadrature rectifier



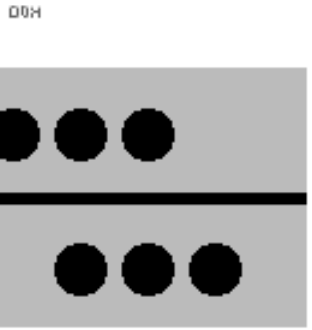
Adaption:
Motion aftereffect

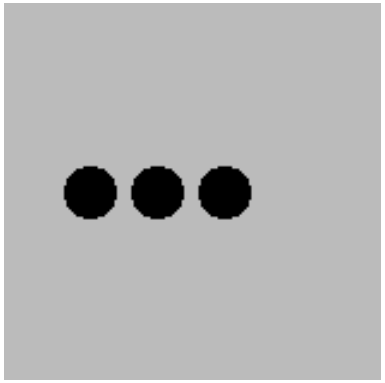


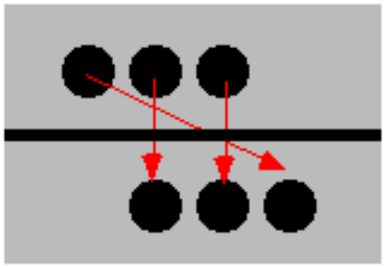
Where is motion processed?

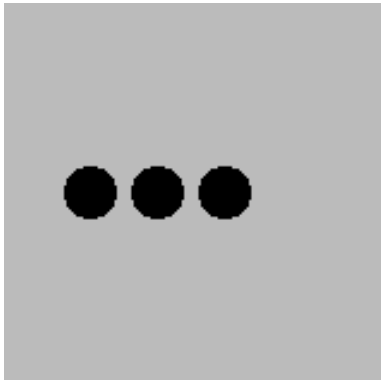


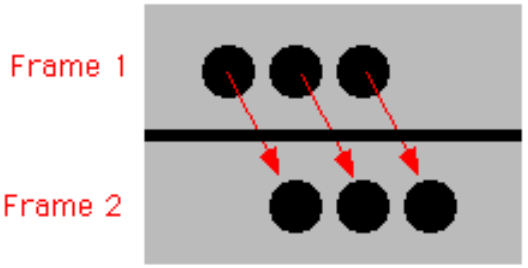
Higher effects:



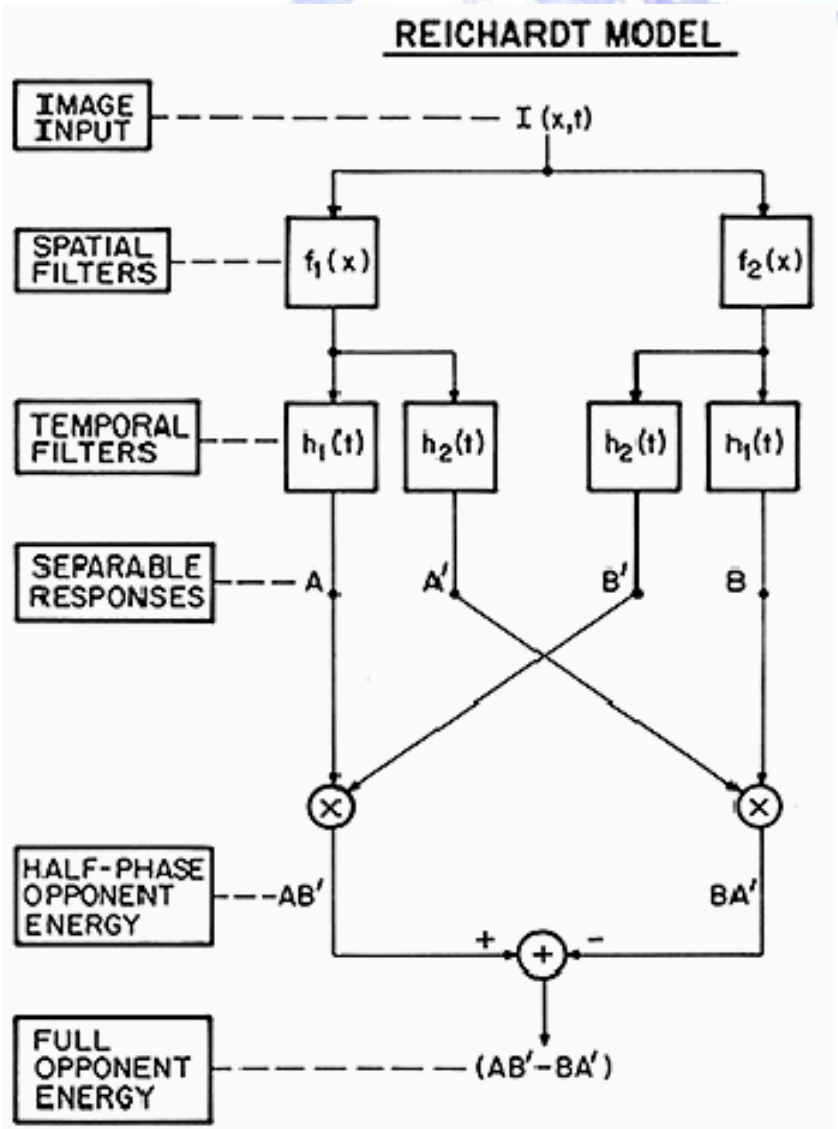


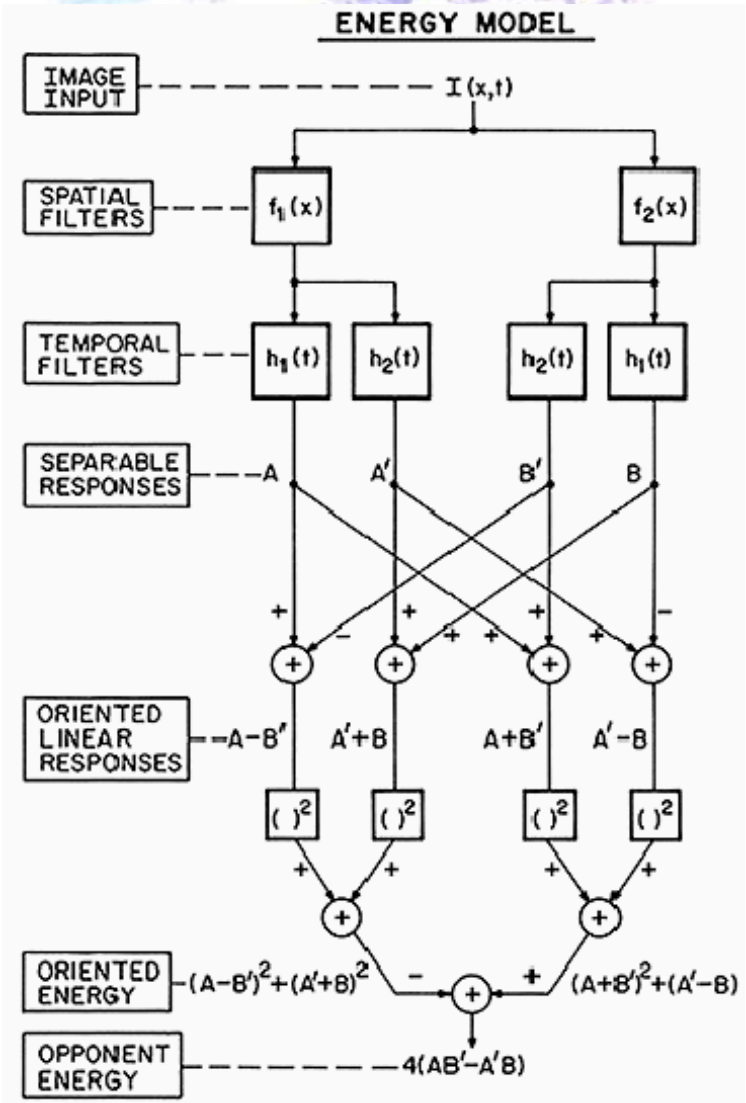






Equivalence: Reich and Spat





Conclusion

- Evolutionary motion detection is important
- Early processing modeled by Reichardt detector or spatio-temporal filters.
- Higher processing poorly understood