Optic Flow and Motion Detection

Computer Vision and Imaging

Martin Jagersand

Readings:

Szeliski Ch 8

3DV Ma, Kosecka, Sastry Ch 4

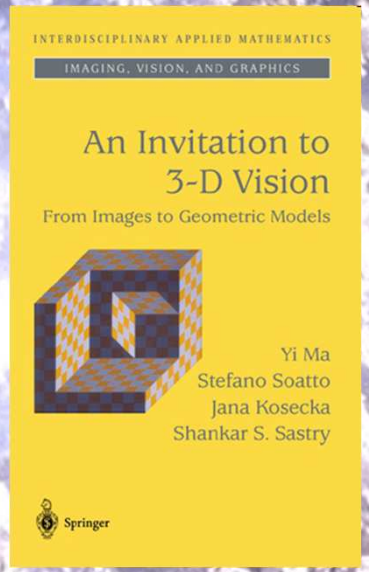
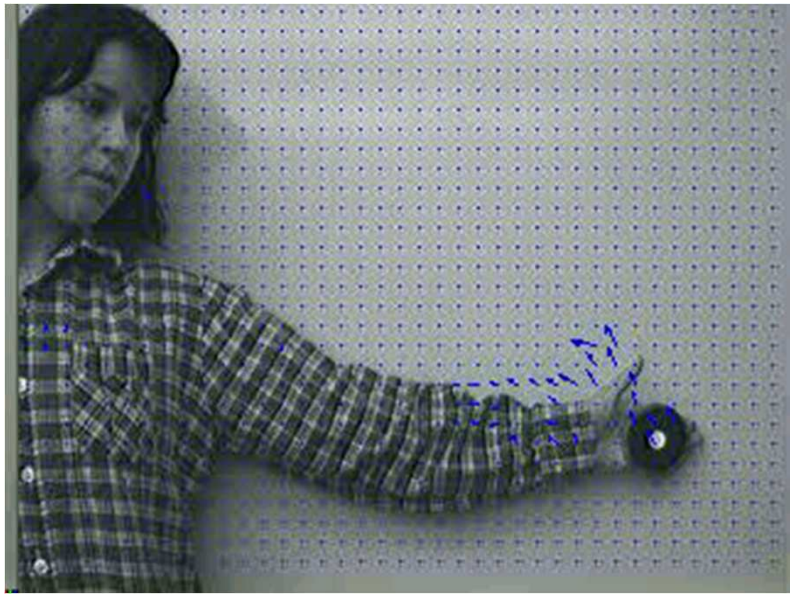


Image motion

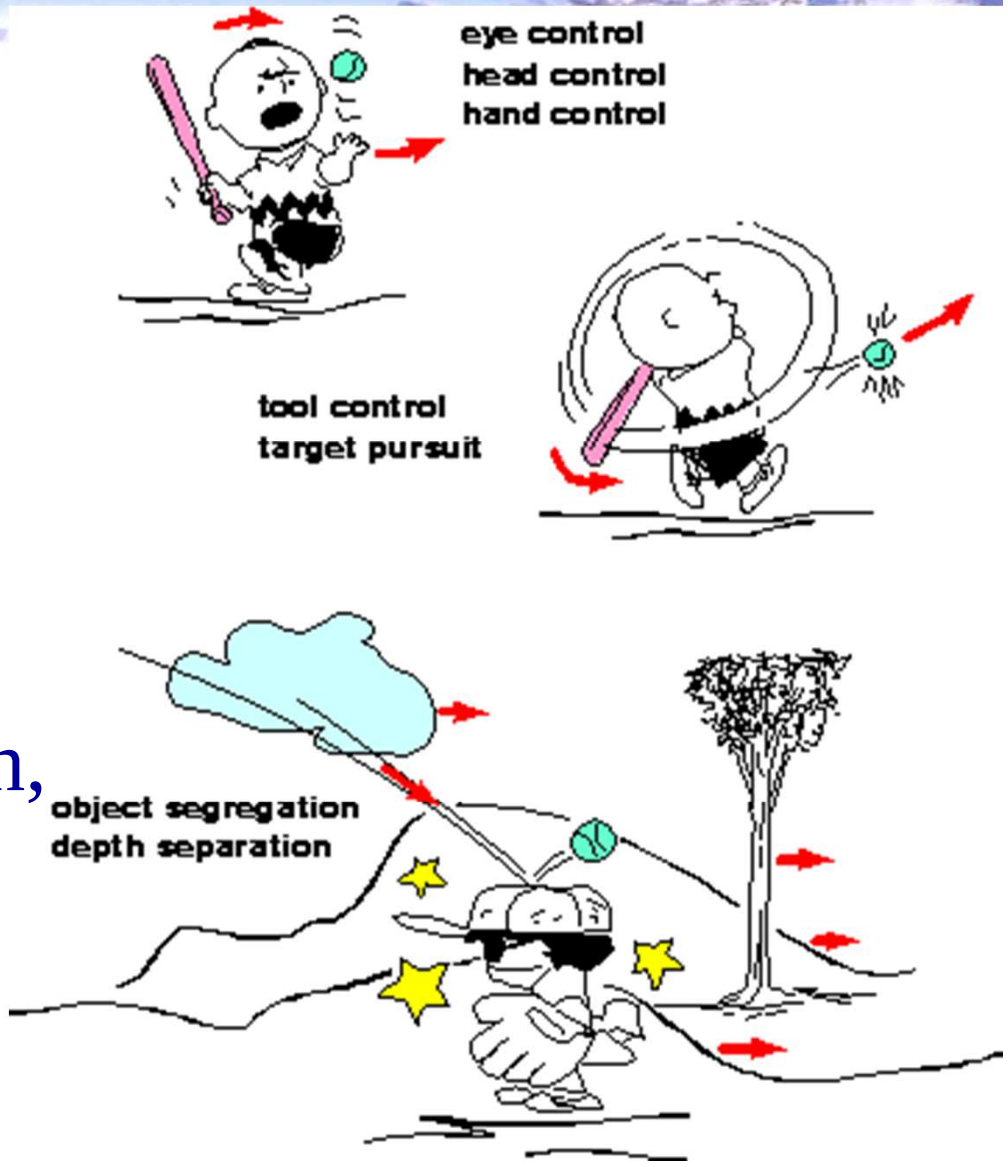
- Somehow quantify the frame-to-frame differences in image sequences.

1. Image intensity difference.
2. Optic flow
3. 3-6 dim image motion computation

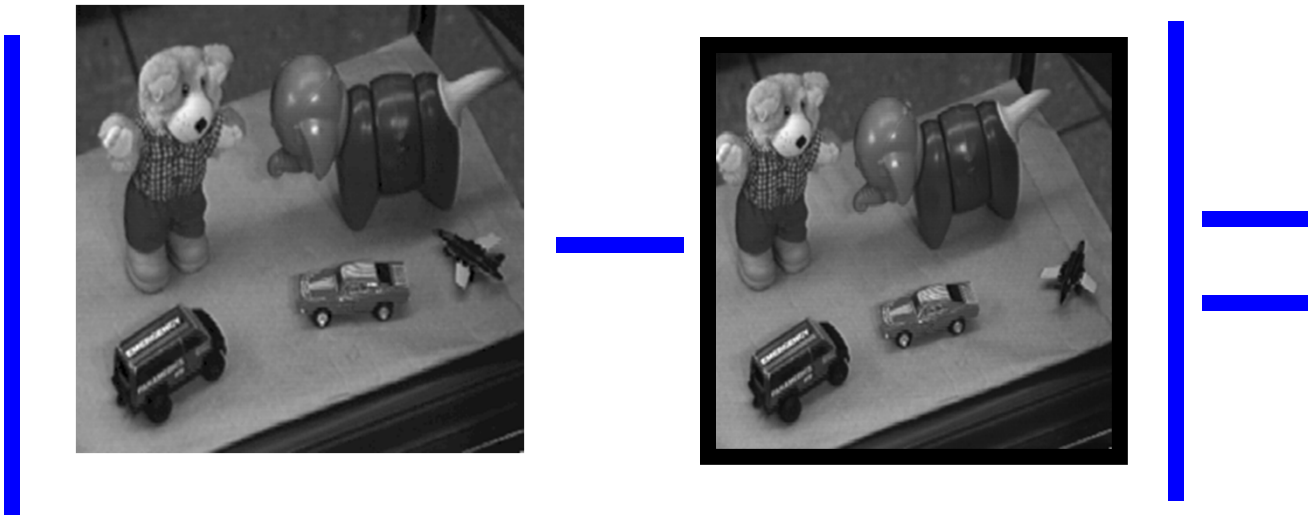


Motion is used to:

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



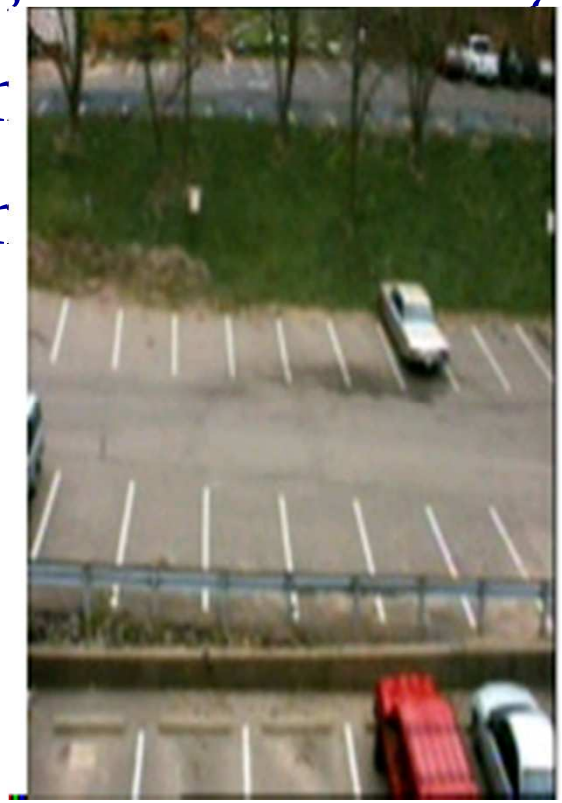
Small camera re-orientation



Note: Almost all pixels change!

Classes of motion

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

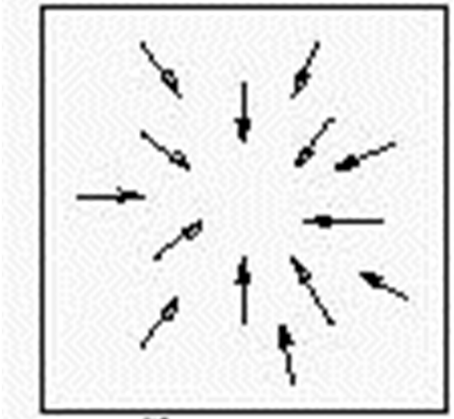


The optic flow field

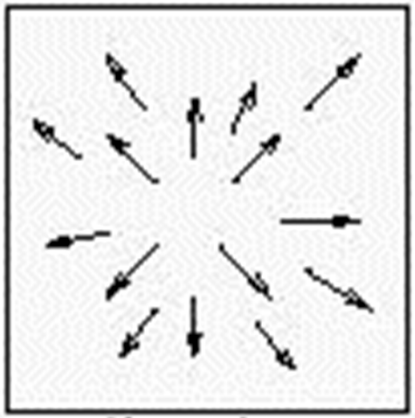
- Vector field over the image:

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

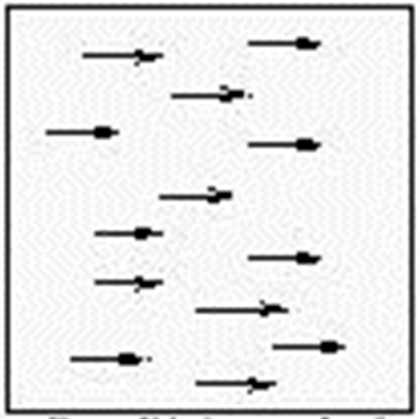
- **FOE, FOC** Focus of Expansion, Contraction



Zoom out

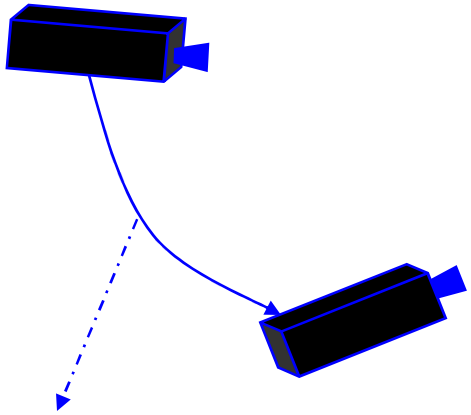


Zoom in

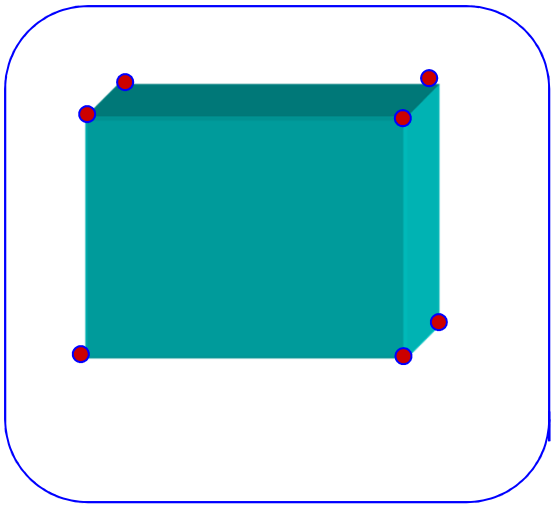


Pan Right to Left

MOVING CAMERAS ARE LIKE STEREO



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

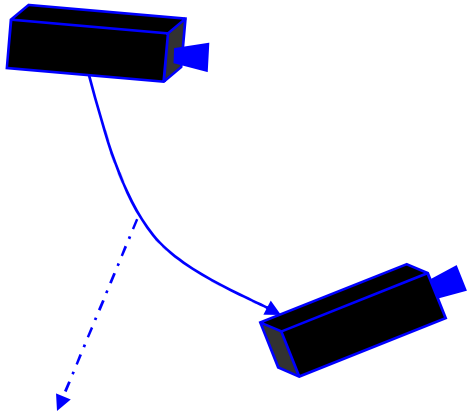


Locations of
points on the object
(the “structure”)

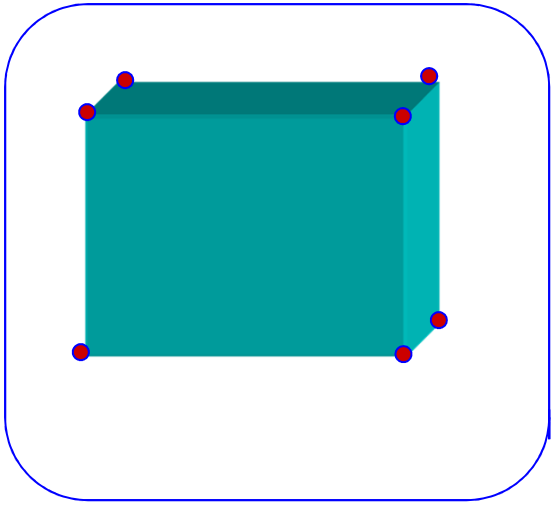
Image Correspondance problem

Given an image point in left image, what is the (corresponding) point in the right image, which is the projection of the same 3-D point

Correspondance for a box

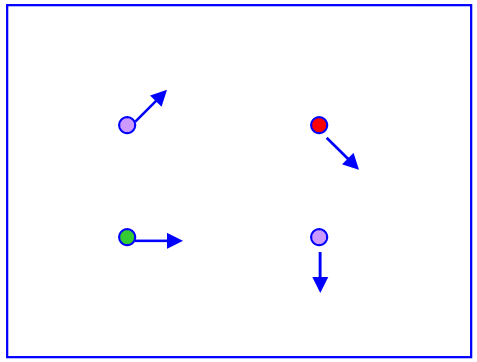


The change in spatial location
between the two cameras (the “motion”)

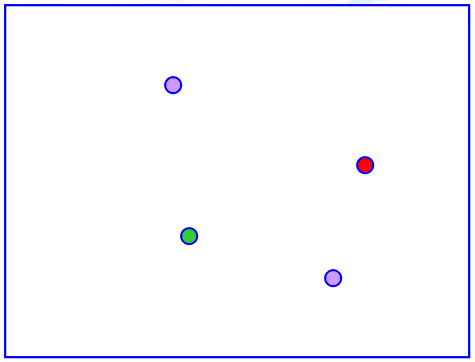


Locations of
points on the object
(the “structure”)

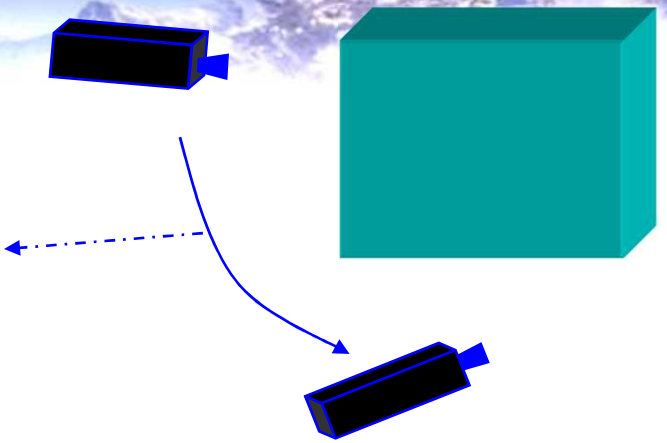
Motion/Optic flow vectors
How to compute?



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



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



– Solve pixel correspondence problem

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

- Possible assumptions

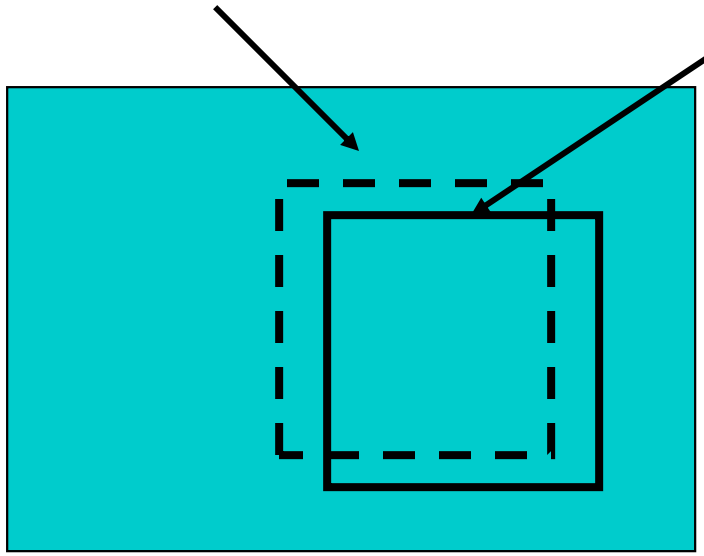
1. **color constancy:** a point in H looks the same in I
   – For grayscale images, this is **brightness constancy**
2. **small motion**: points do not move very far
   – This is called the **optical flow** problem

Optic/image flow

Assume:

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

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



Taylor expansion of intensity variation

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

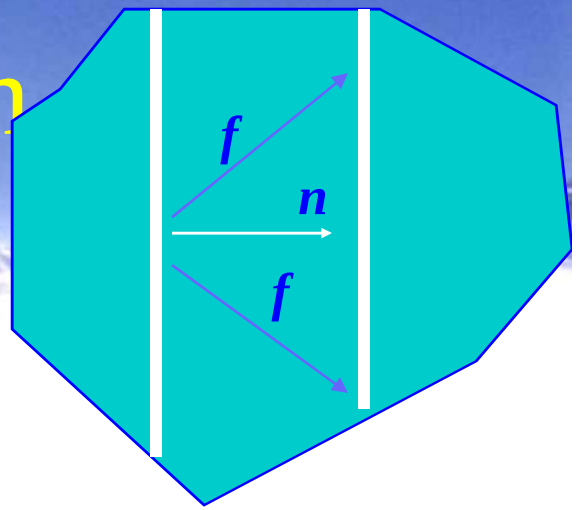
Keep linear terms

- Use constancy 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

Aperture problem

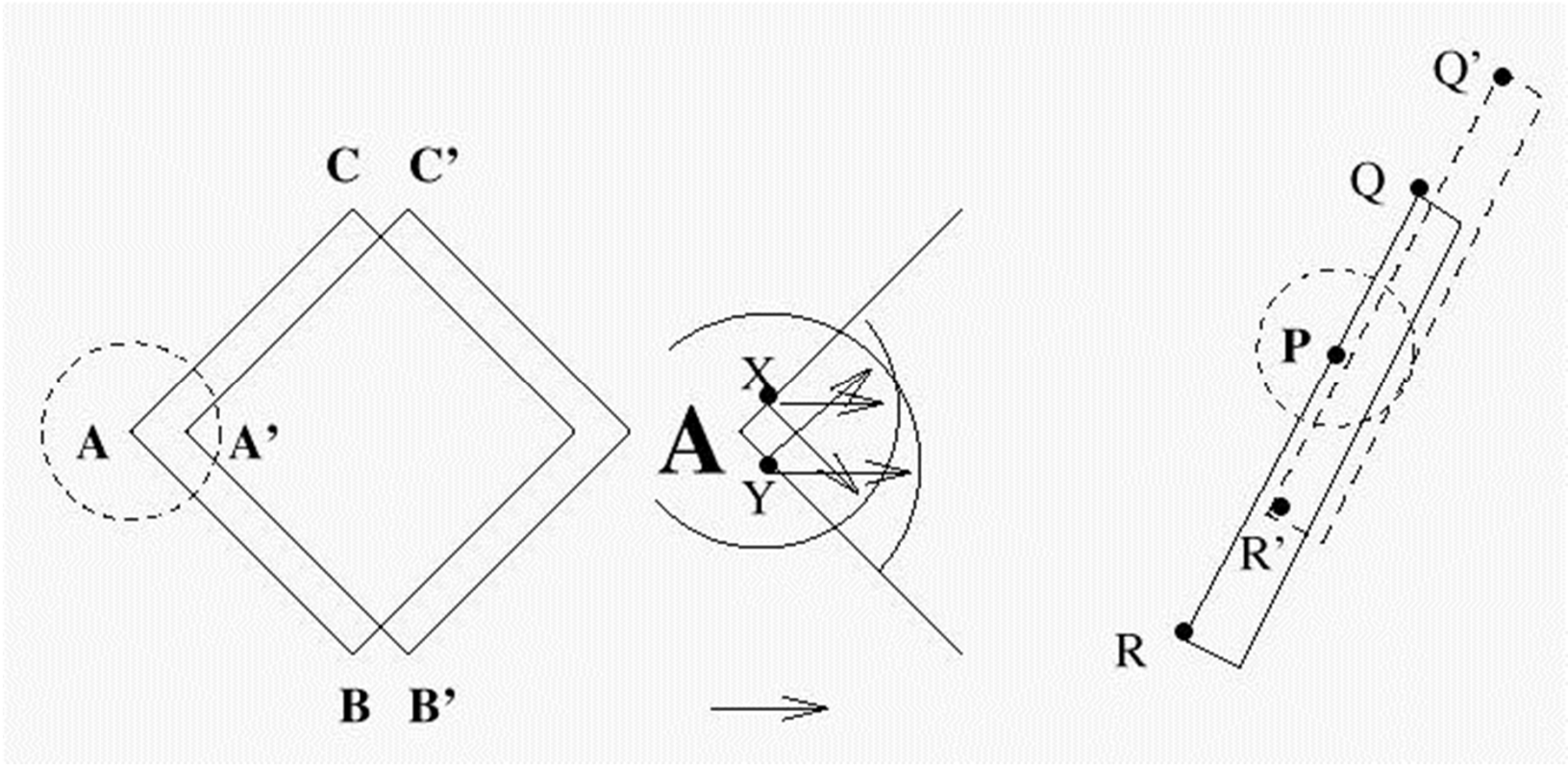


- Rewrite as dot product

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

- 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



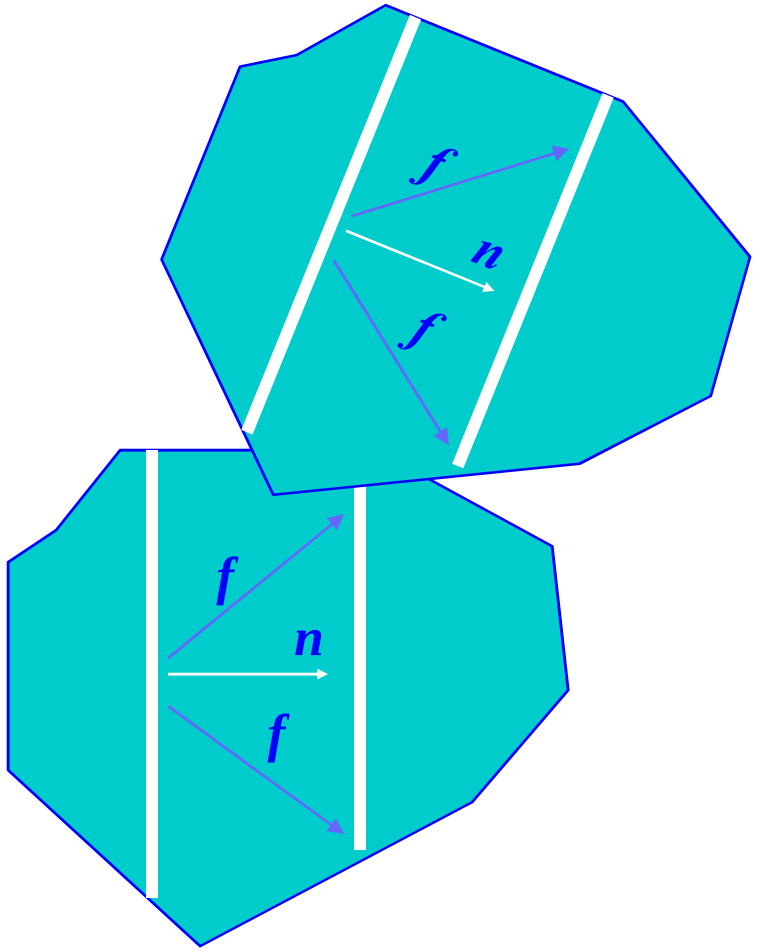
The flow continuity constraint

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

  $\boldsymbol{n}_1 * \boldsymbol{f} = k_1$

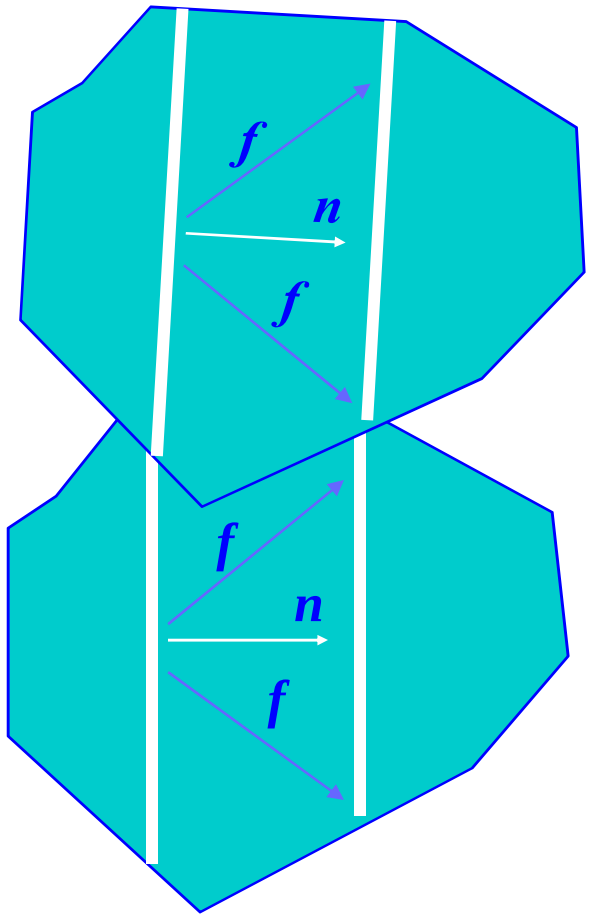
  $\boldsymbol{n}_2 * \boldsymbol{f} = k_2$

- 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 $\boldsymbol{f}$



Using several points

- Typically solve for motion in 2x2, 4x4 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 = Mu

- Can be solved in least squares sense using Matlab

u = M\dIm

- Can also be solved be solved using normal equations

$u = (M^T M)^{-1} * M^T dIm$

Conditions for solvability

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

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

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

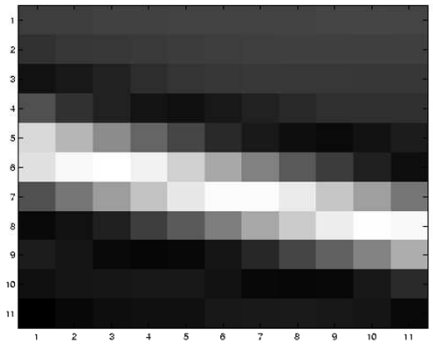
When is this solvable?

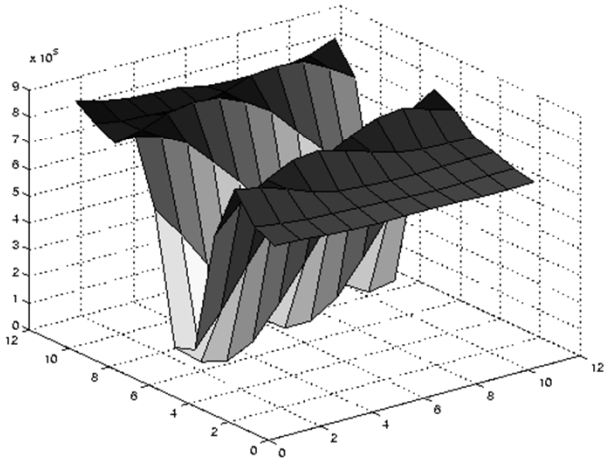
- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ entries should not be too small (noise)
- $\mathbf{A^T A}$ should be well-conditioned
- Study eigenvalues:
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Optic Flow Real Image Challenges:

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

Edge

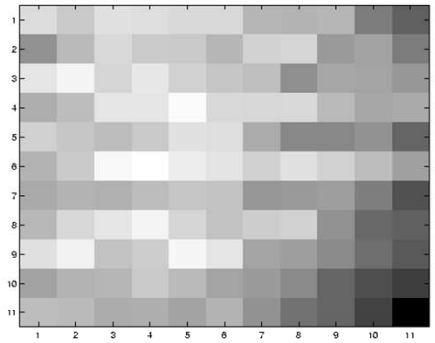


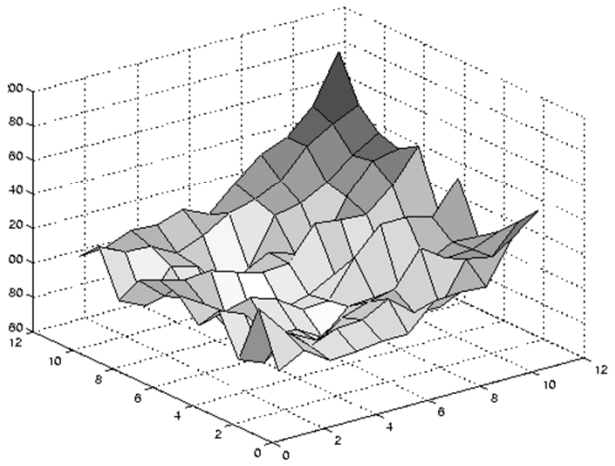


$$\sum \nabla I (\nabla I)^T$$

- gradients very large or very small
- large λ_1, small λ_2

Low texture region

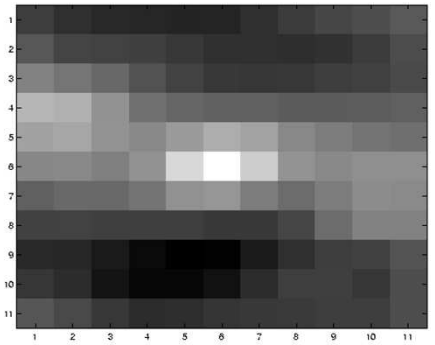


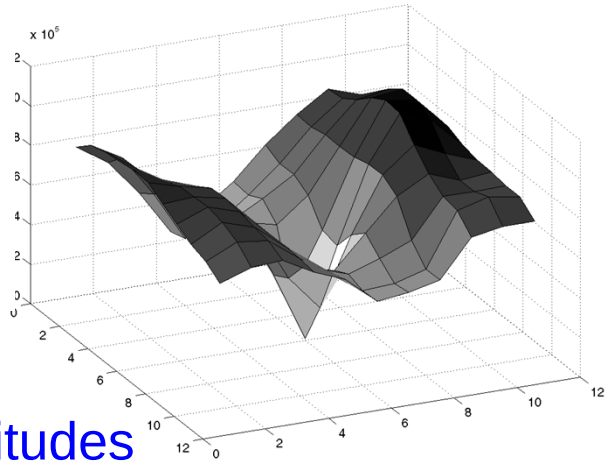


$$\sum \nabla I (\nabla I)^T$$

- 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...

Errors 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?

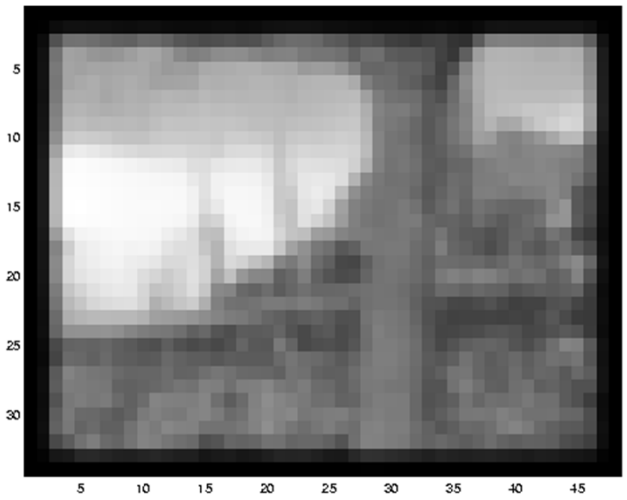
Iterative Refinement

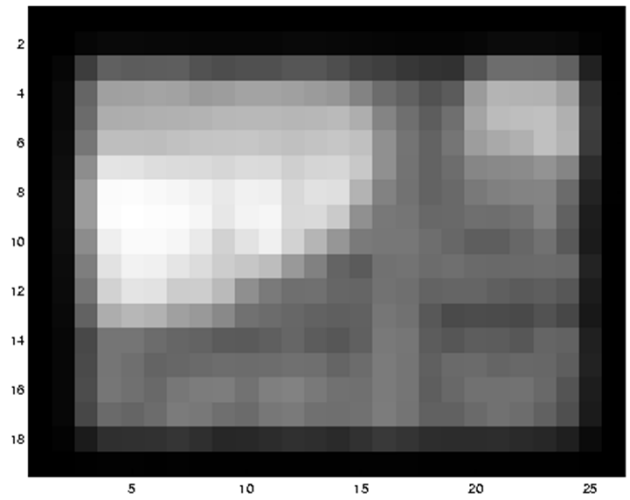
- Used in SSD/Lucas-Kanade tracking algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp H towards I using the estimated flow field
     - *use image warping techniques*
  3. Repeat until convergence

Revisiting the small motion assumption

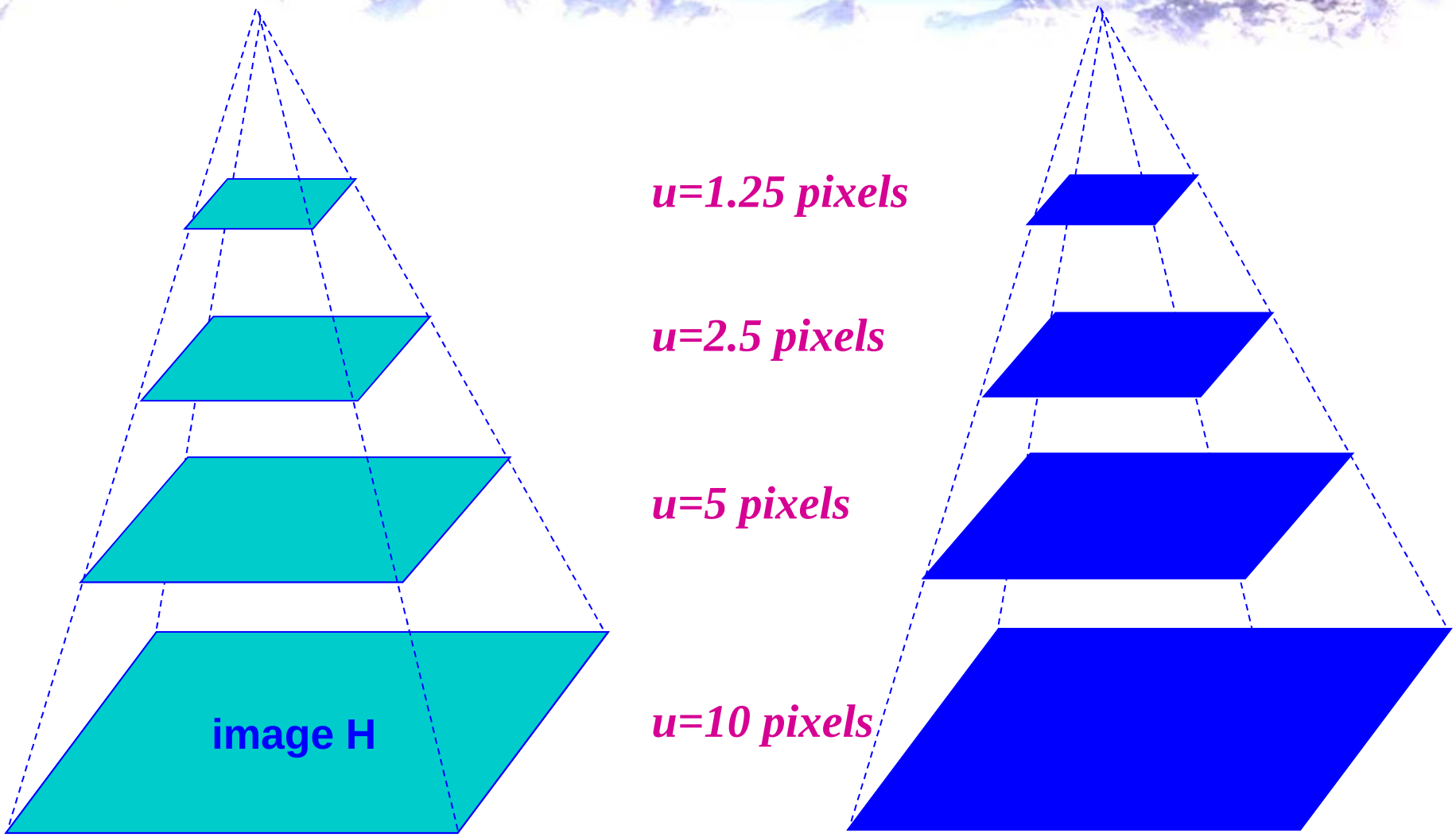
- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)
  - How might we solve this problem?

Reduce the resolution!

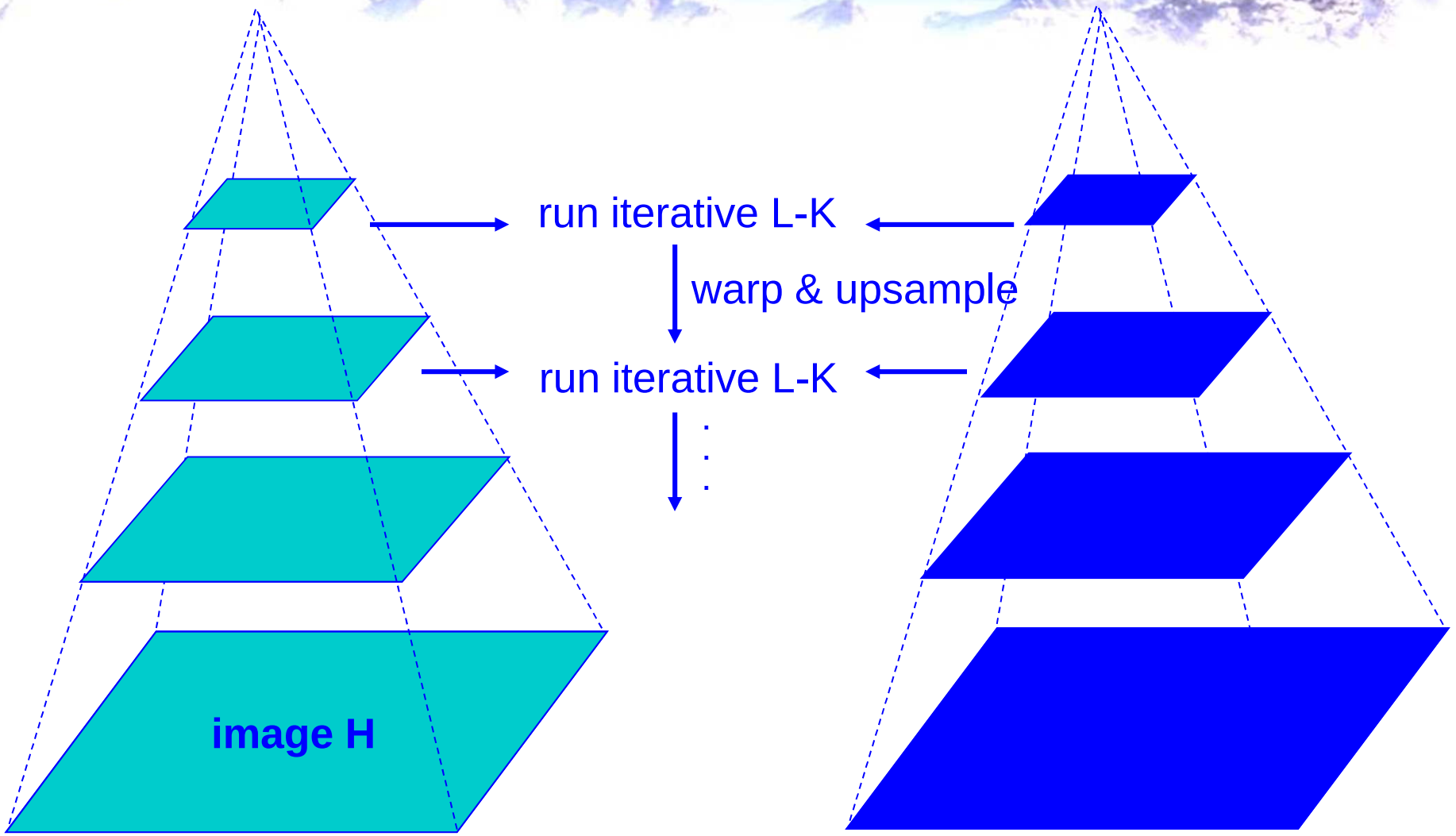




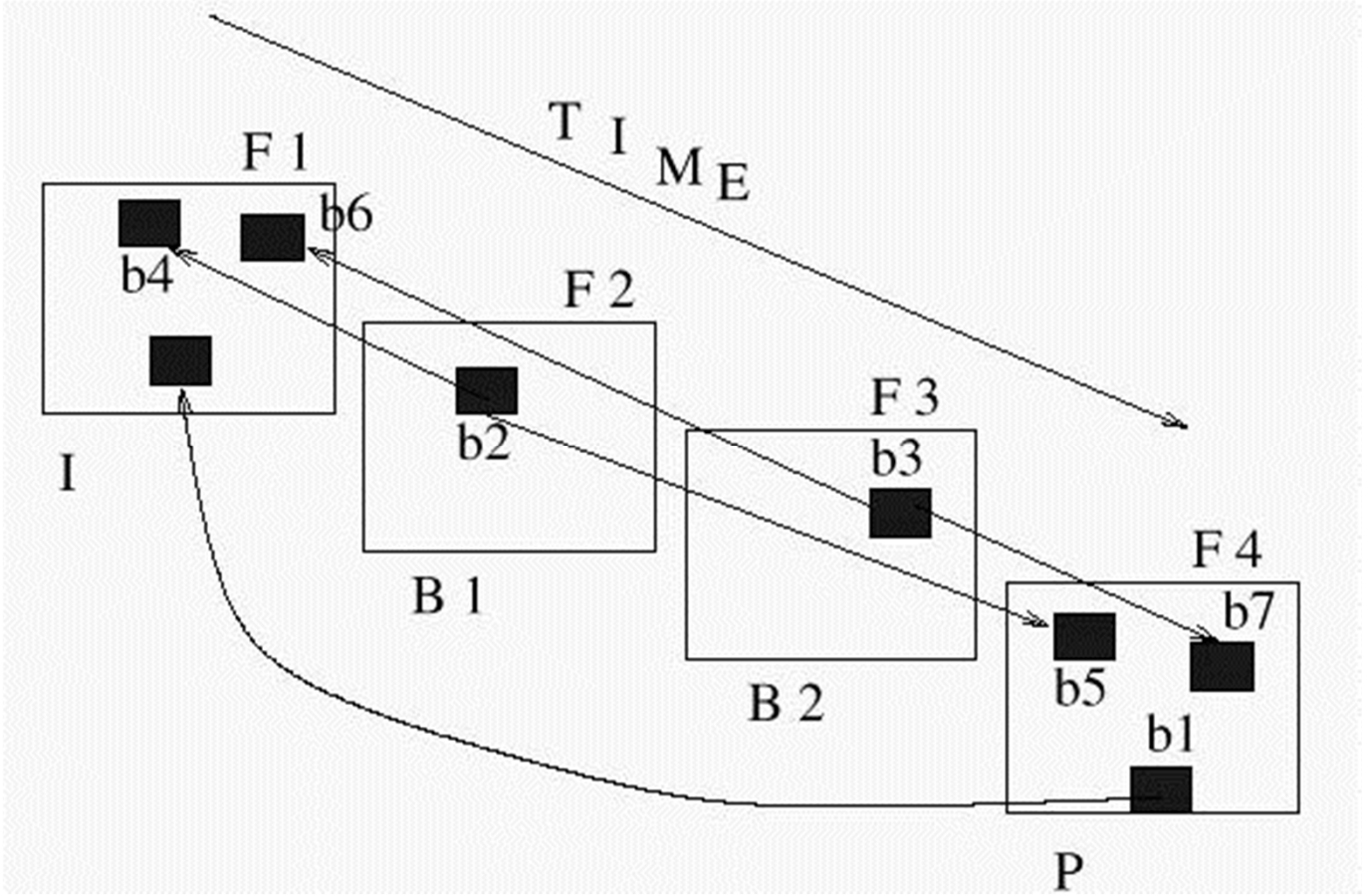
Coarse-to-fine optical flow estimation



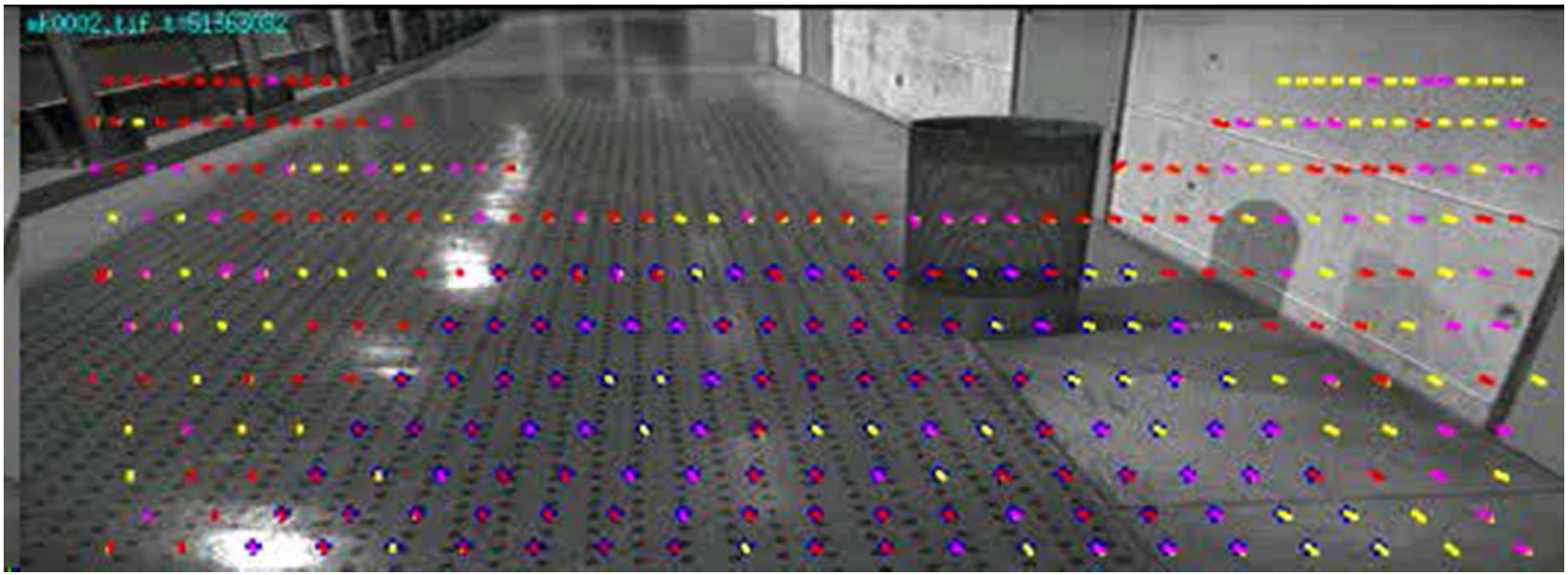
Coarse-to-fine optical flow estimation



Application: mpeg compression



HW accelerated computation of flow vectors



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

Other applications:

- Recursive depth recovery: Kostas and Jane
- Motion control (we will cover)
- Segmentation
- Tracking

Lab:

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

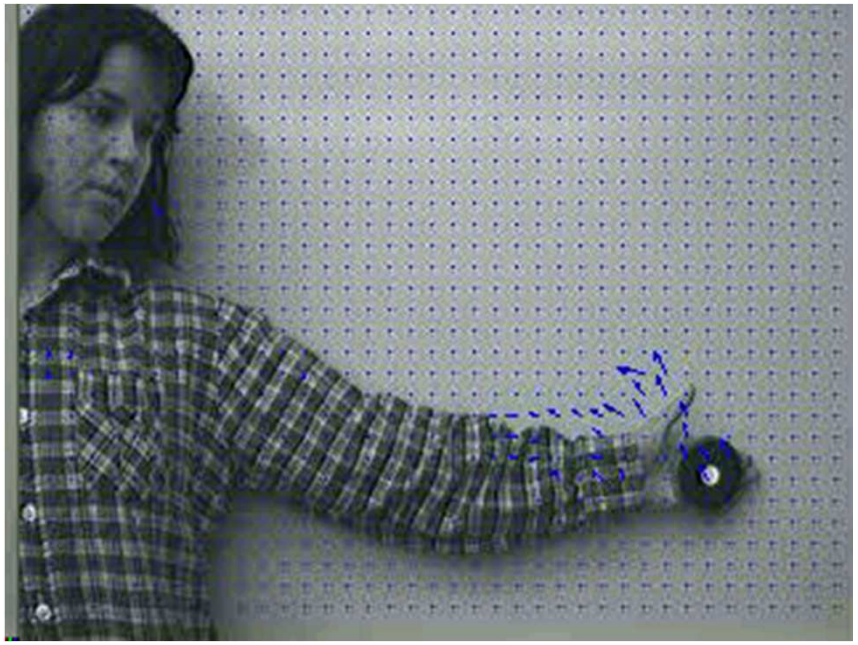
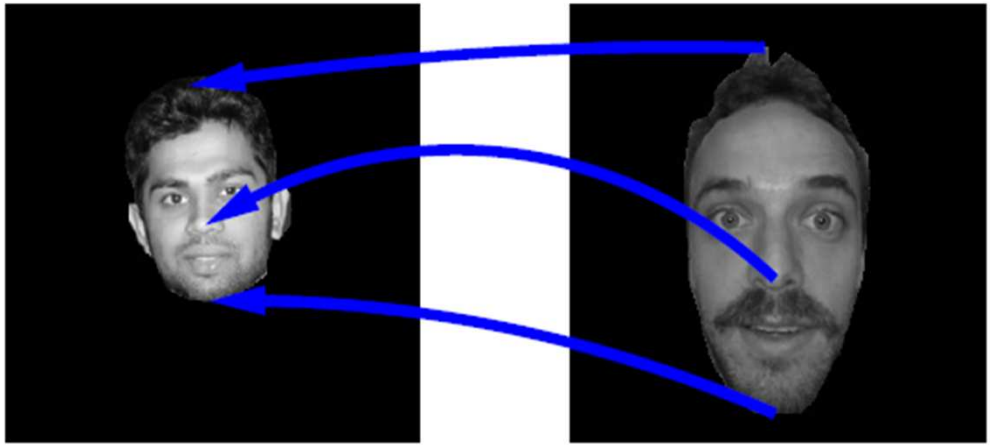


Image registration

WHAT is image registration



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

Image registration

WHAT is image registration

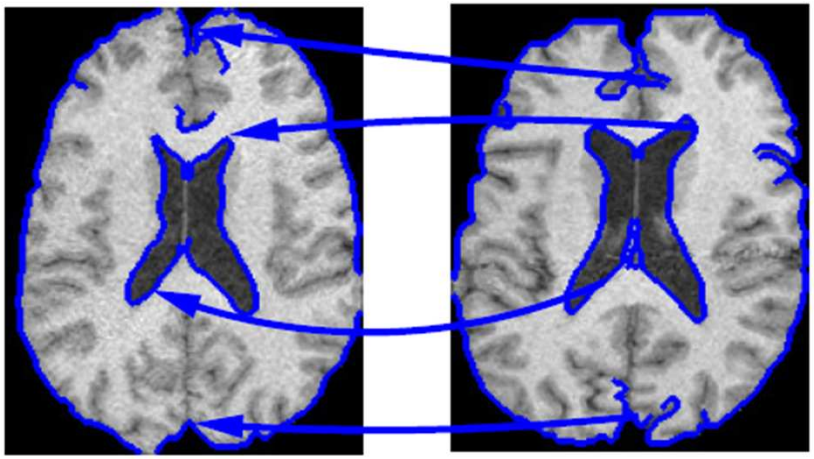
Transform a “source” image
match a “target” image

Medical image registration

WHAT is image registration

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

Medical image registration

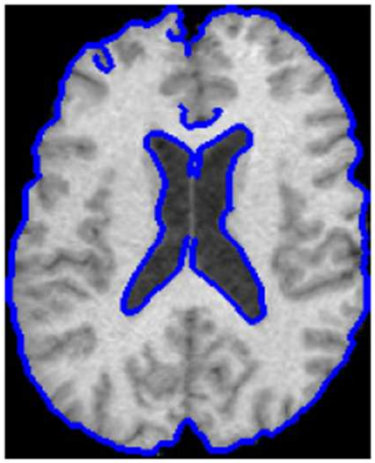


Medical image registration

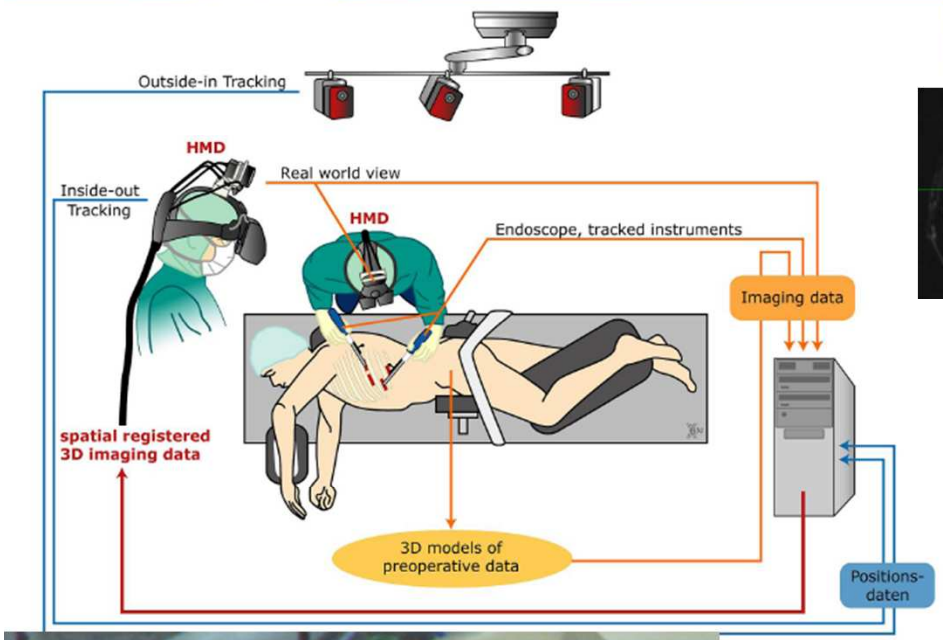
WHAT is image registration

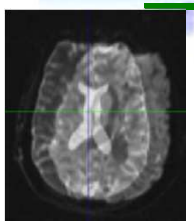
Transform a “source” image
match a “target” image

Medical image registration



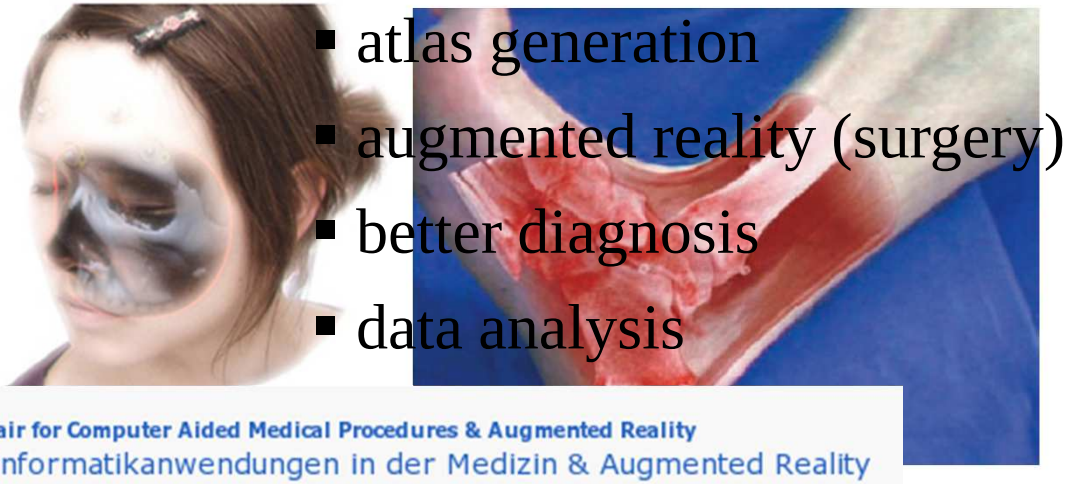
Medical applications





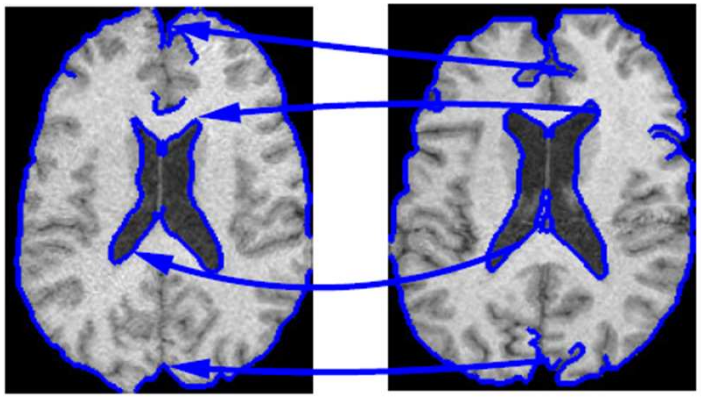
- **Data (source, target)**
  - different medical images modalities (MRI, XRay, CT…)
  - pre-acquired medical image with real-time images (video)
  - patient data with an atlas

- **For:**
  - atlas generation
  - augmented reality (surgery)
  - better diagnosis
  - data analysis



Formulation

Very similar to tracking and optic flow.



I_A I_B

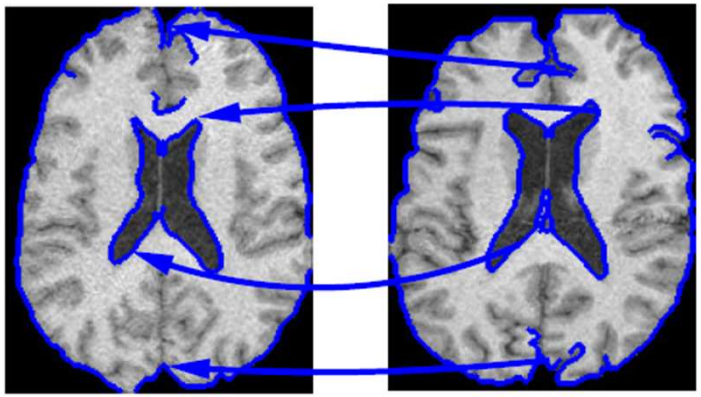
Transform a “source” image to **match** a “target” image

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

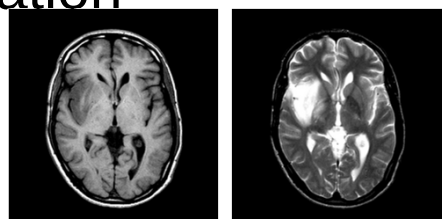
Transform a “source” image to **match** a “target” image

Find best transformation T through the minimization of an energy

Maching – similarity score : Sim

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

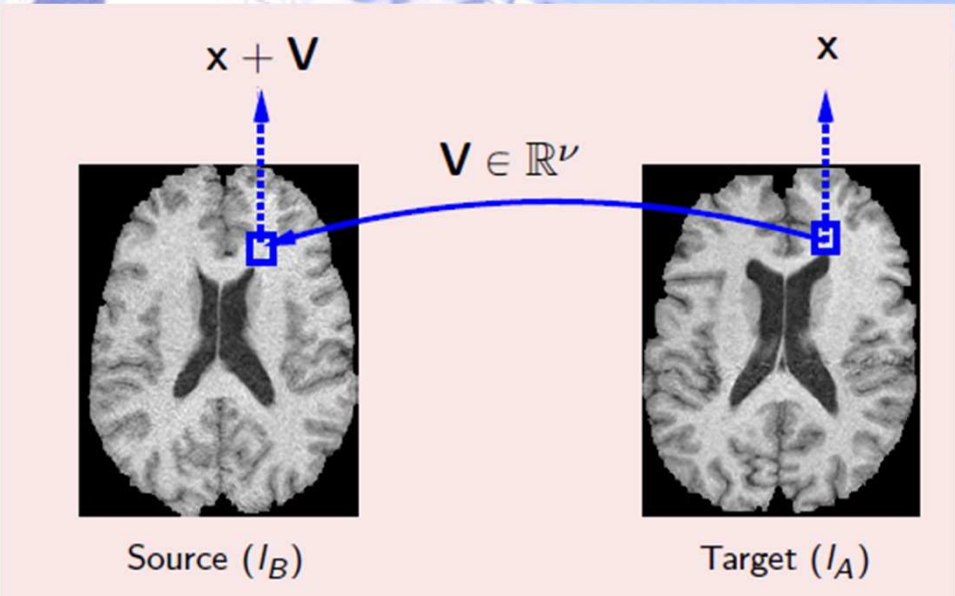
- depends on data
- simple – same type of data - SSD : sum $(I_A(x) - T(I_B(x)))^2$
- different illumination : NCC normalized cross correlation
- different imaging modalities : MI mutual information

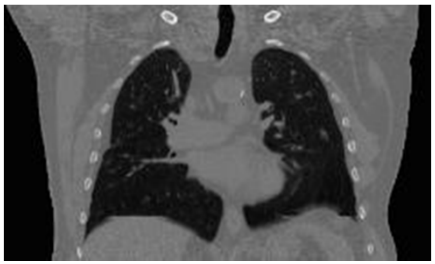


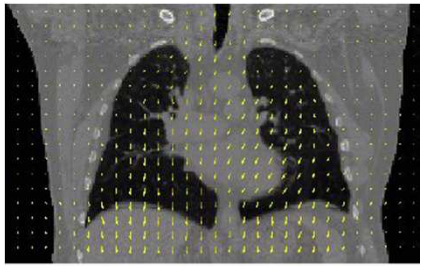
Transformation : T

- (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

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

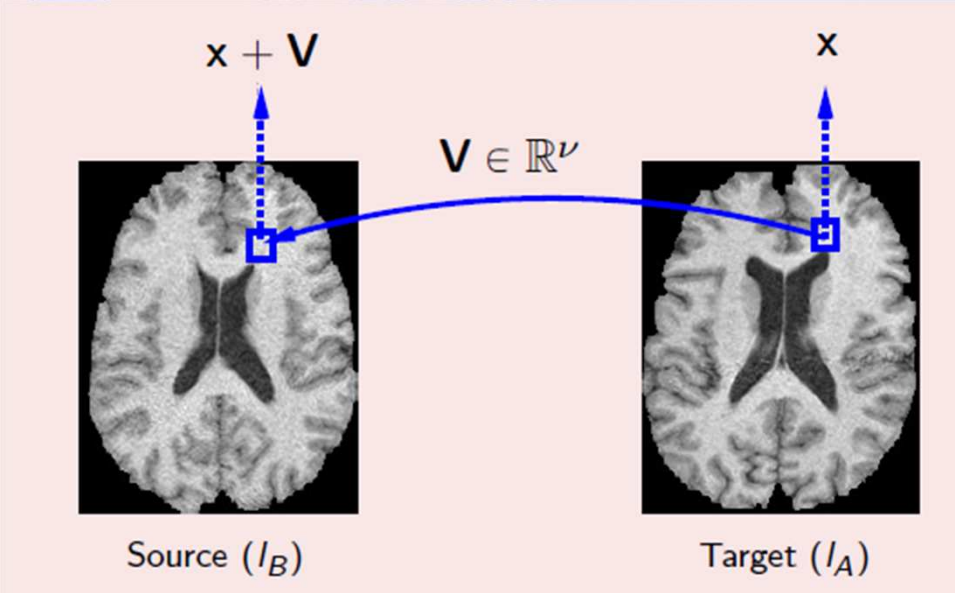
Gradient descent: solve for v iteratively adding small updates delta **δv**

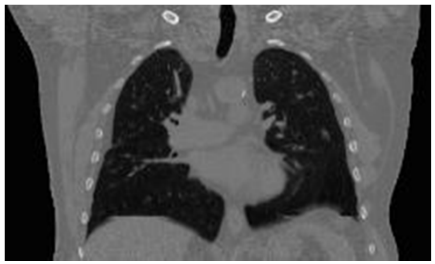
Each step is similar to an **optic flow problem**

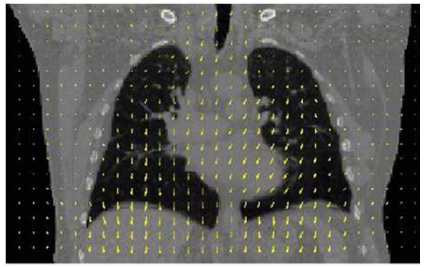
$\min_{\delta\mathbf{v}} \text{sum}_x (I_A(x) - I_B(x+\mathbf{v}+\delta\mathbf{v}))^2$

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

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

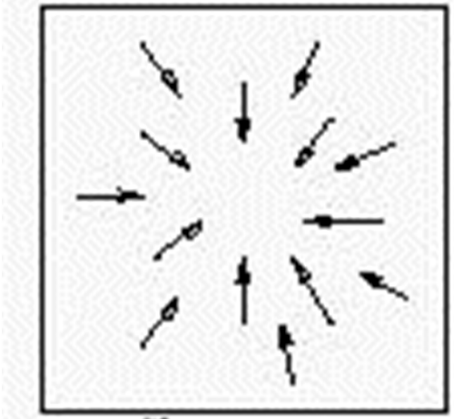
2. Mike Black, et al: Attempt to find a low dimensional subspace for complex motion

Remember:
The optic flow field

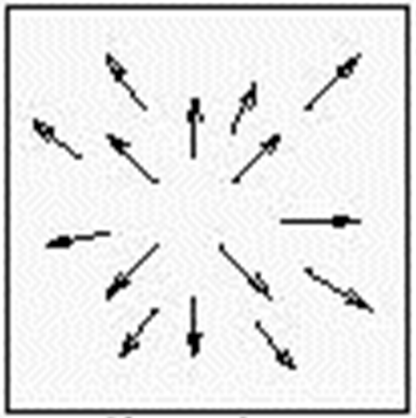
- Vector field over the image:

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

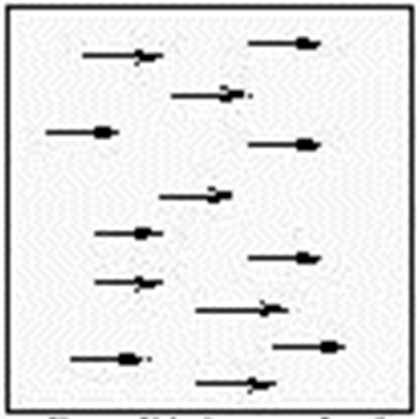
- **FOE, FOC** Focus of Expansion, Contraction



Zoom out



Zoom in

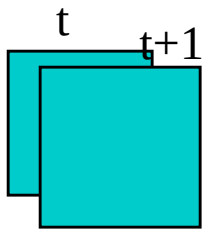


Pan Right to Left

Remember last lecture:

- Solving for the motion of a patch

Over determined equation system:



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

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

- Can be solved in e.g. least squares sense using matlab $u = M \backslash Im_t$

3-6D Optic flow

- Generalize to many freedooms (DOFs)

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

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

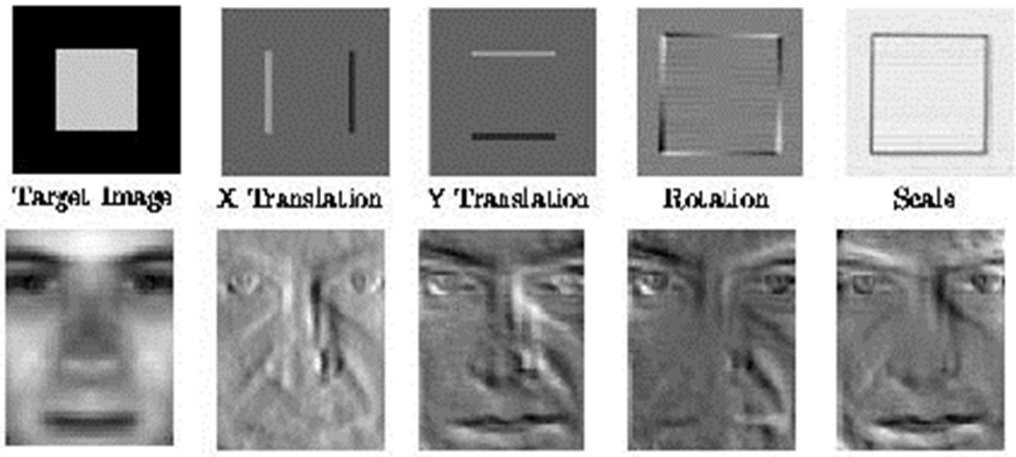
$$M = [I_x|I_y|I_r|I_s]$$

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

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

$$I_r = -yI_x + xI_y$$

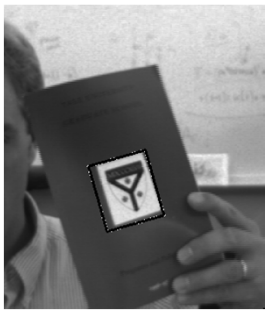
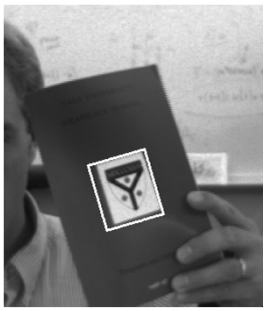
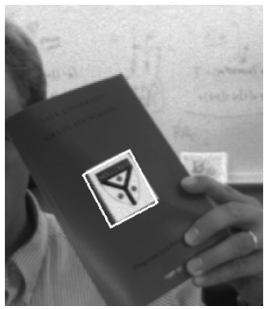
$$I_s = \frac{1}{\sqrt{x^2+y^2}}(xI_x + yI_y)$$



Im = Mu

Example:
All 6 freedoms

Template


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

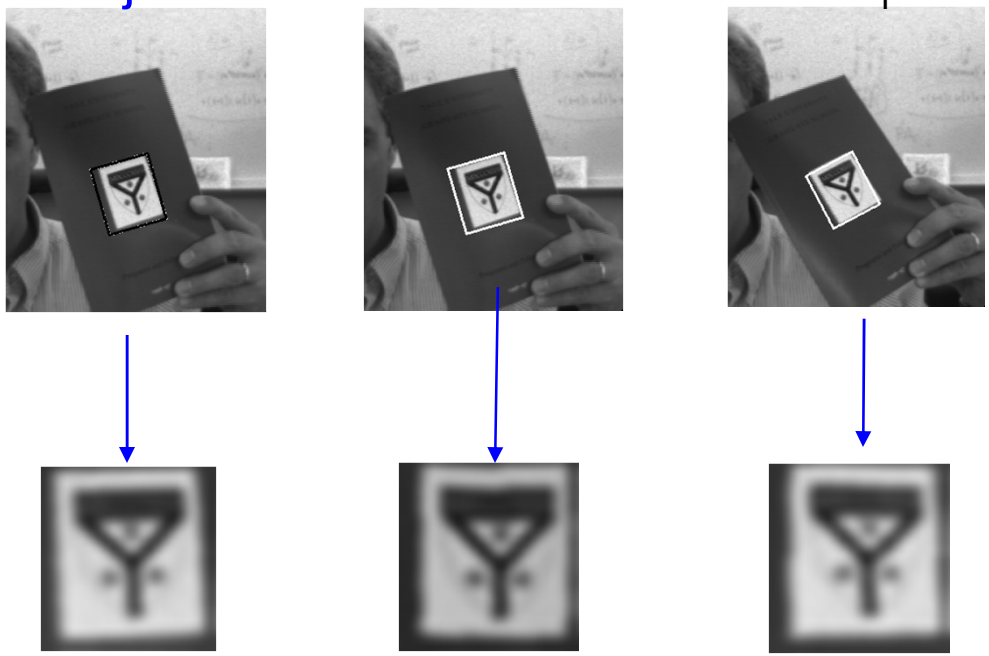





X **Y** **Rotation** **Scale** **Aspect** **Shear**

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

$$\Delta p = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \Delta I \quad \text{where} \quad \mathbf{M} = \mathbf{M}(p_t, I_t)$$

Planar 3D motion

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

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

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

2. 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 $u'_i = \mathbf{A}\, u_i + d$

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



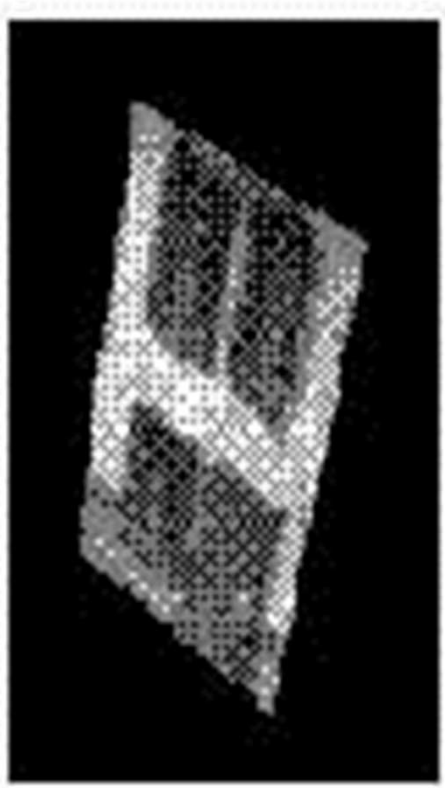


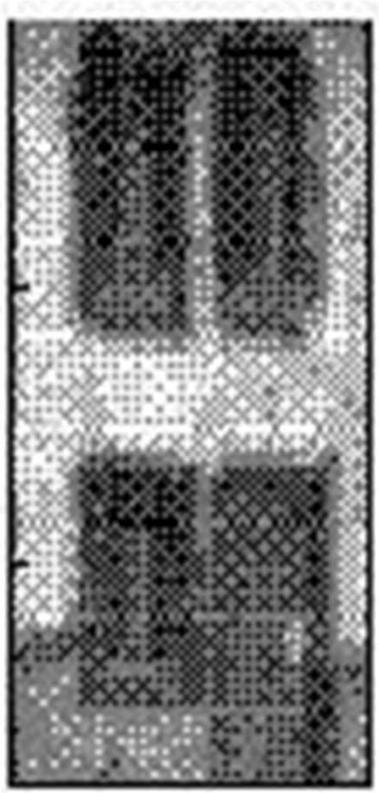

X **Y** **Rotation** **Scale** **Aspect** **Shear**

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

Planar motion under perspective projection

- Perspective plane-plane transforms defined by homographies





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

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:

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

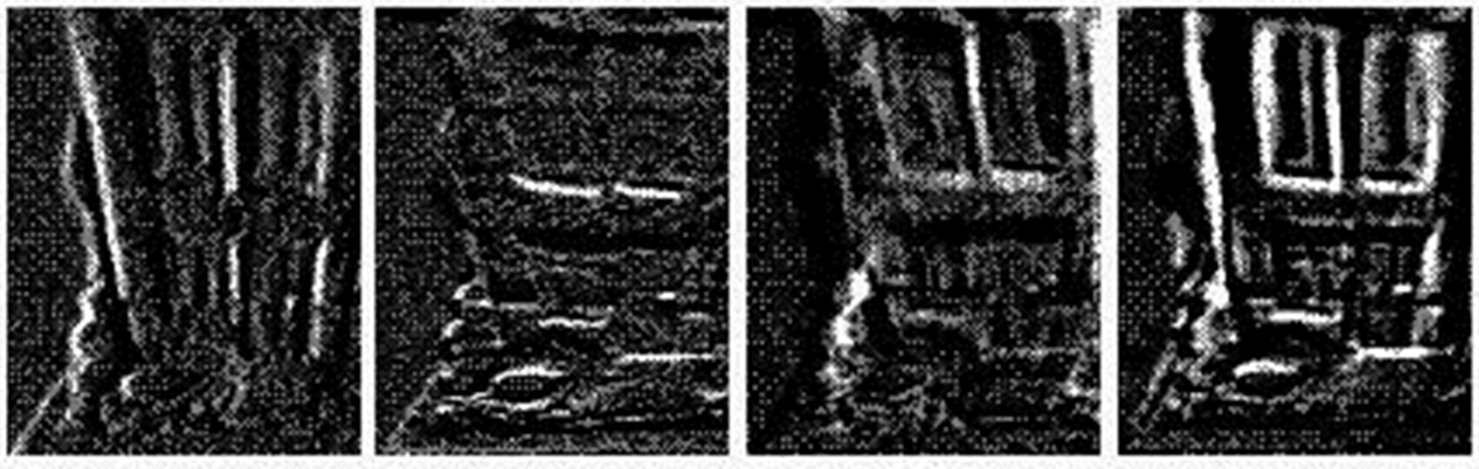
$$= [\mathbf{B}_1 \ldots \mathbf{B}_8][y_1, \ldots, y_8]^T = B_h \mathbf{y}_h$$

- 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

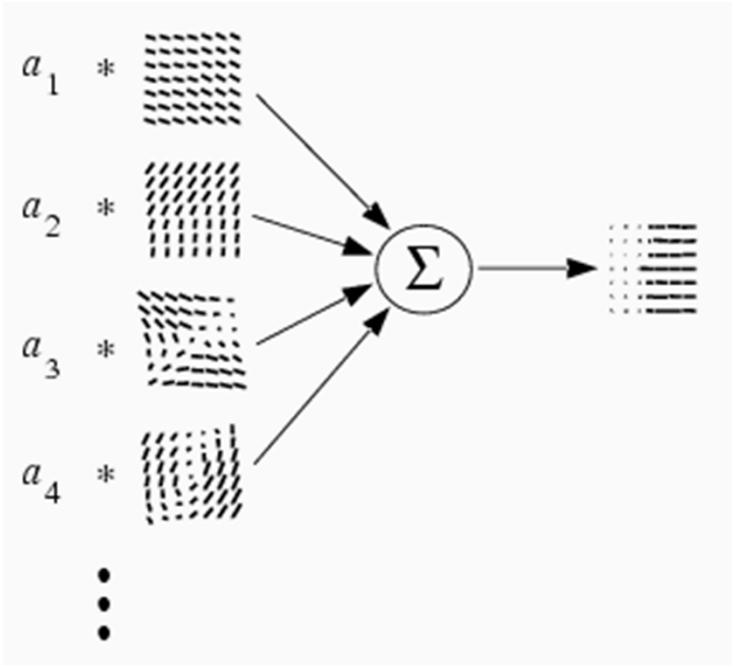


Another idea Black, Fleet) Organizing flow fields

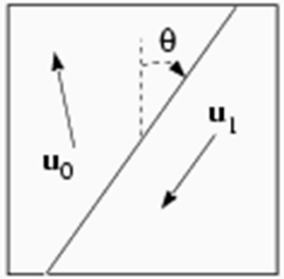
- Express flow field f in subspace basis m

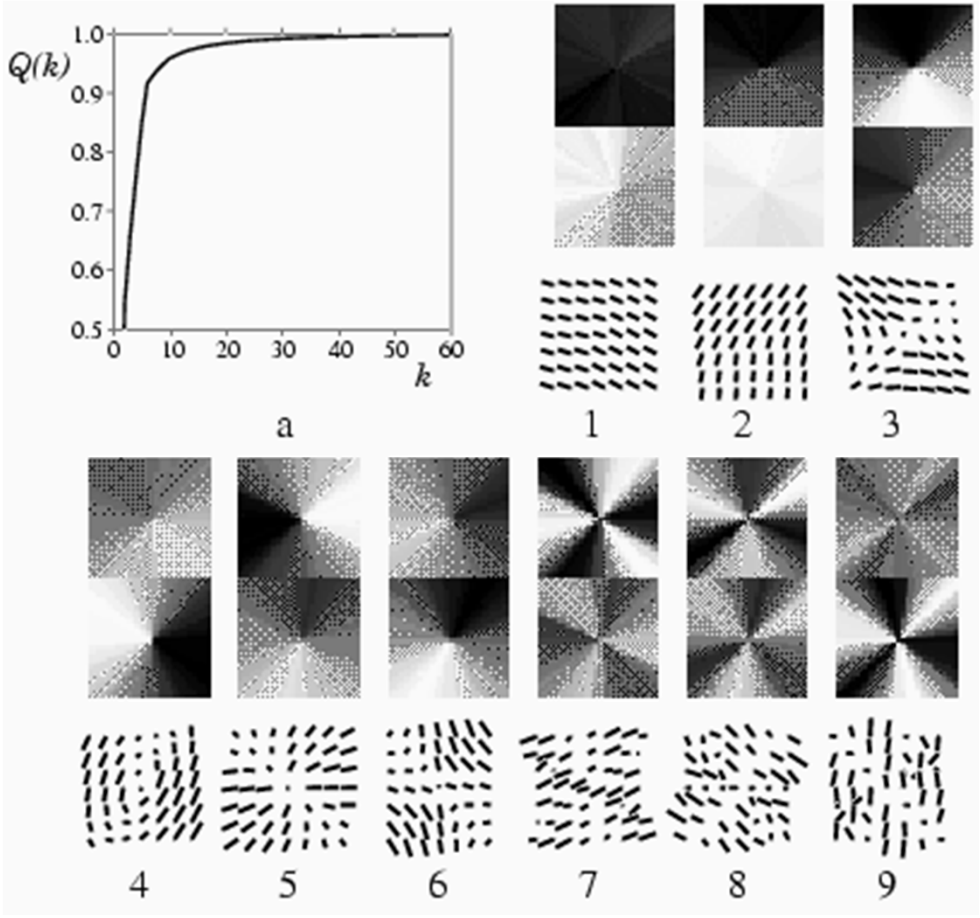
$$\vec{f}_k = \sum_{i=1}^{k} a_i \vec{m}_i$$

- 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

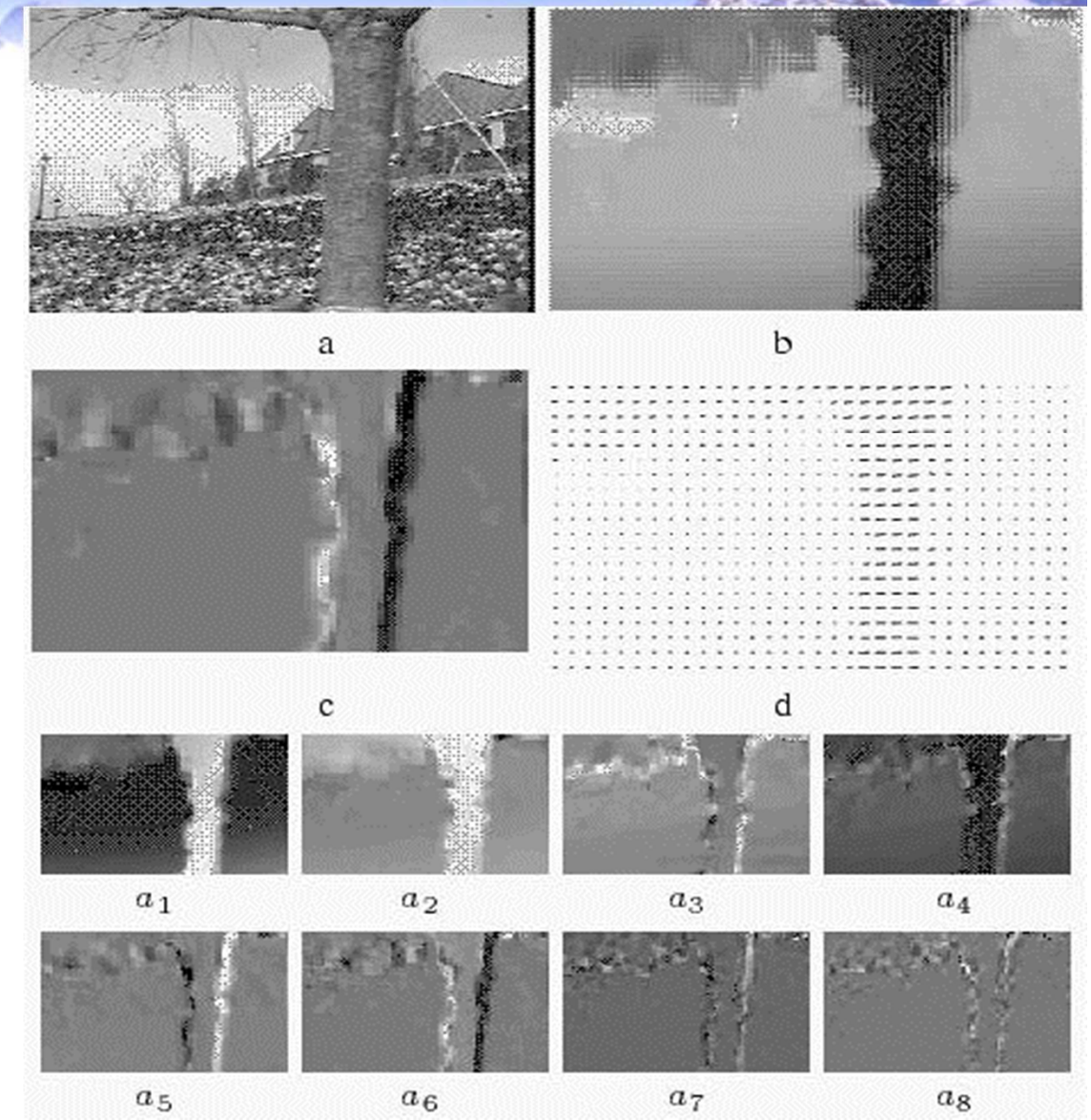
$$E(\vec{b};\vec{a}) = \sum_{\vec{x}\in R} \rho(I(\vec{x}+\vec{u}(\vec{x};\vec{a}+\vec{b}),t+1)-I(\vec{x},t),\ \sigma)$$

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

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

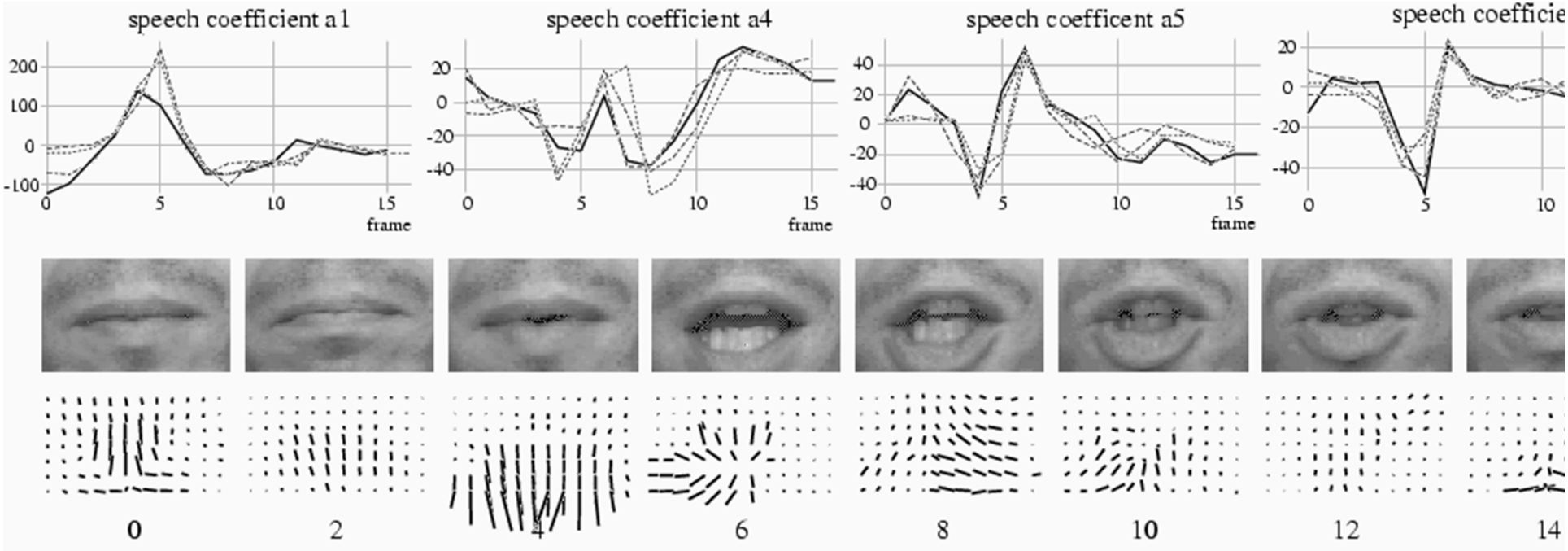
Experiment
Moving camera

- 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:

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

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

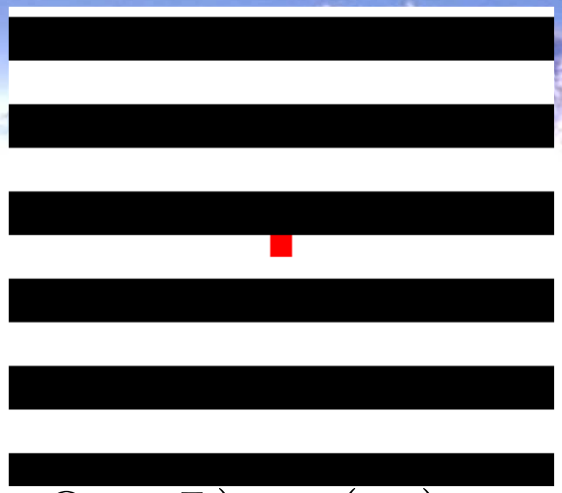
Motion due to translation: depends on depth

Motion due to rotation: independent of depth

Sensing and Perceiving Motion

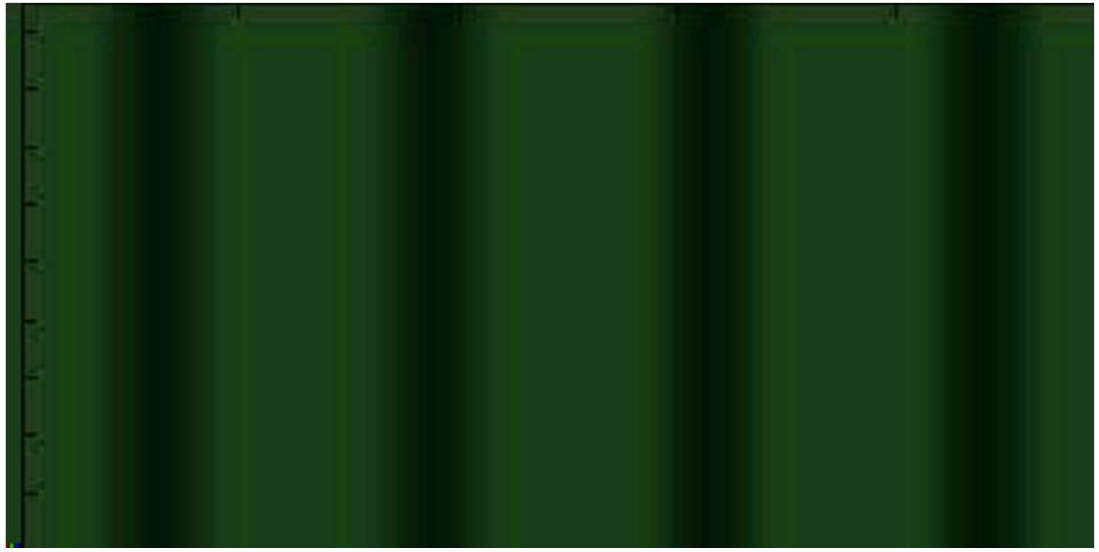
Martin
Jagersand

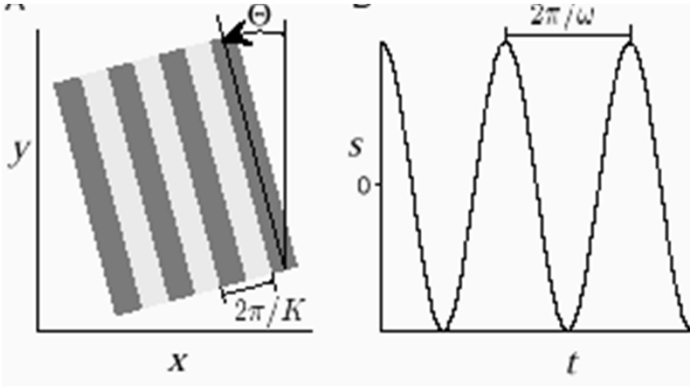
Counterphase sin grating



- 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:

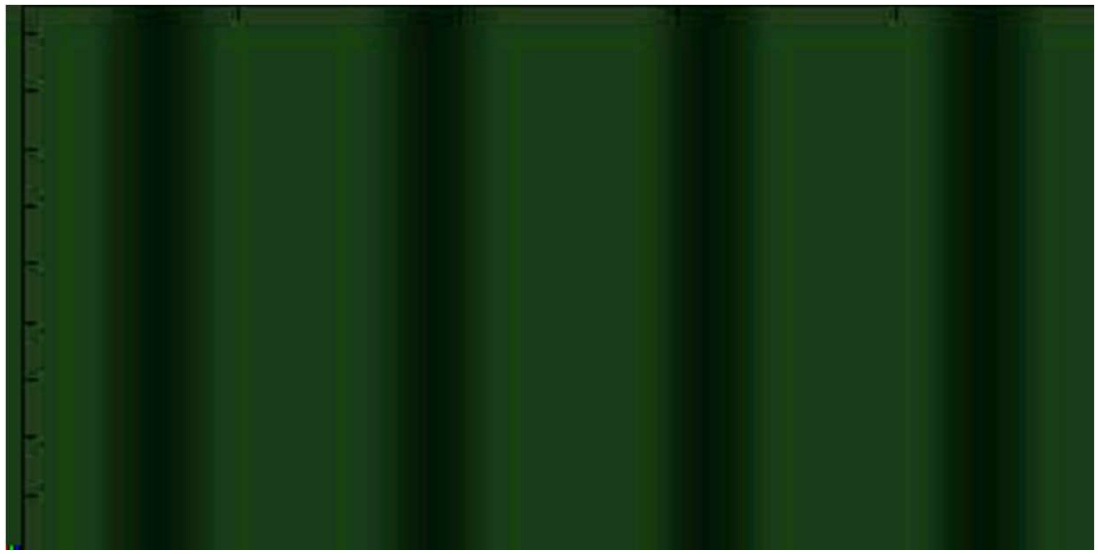
$$\tfrac{1}{2}(\cos([a, b]\begin{bmatrix} x \\ y \end{bmatrix} - \omega t) + \cos([a, b]\begin{bmatrix} x \\ y \end{bmatrix} + \omega t)$$

$=$ ⟵ ||||| $+$ ||||| ⟶

Result: Standing wave is superposition of two moving waves

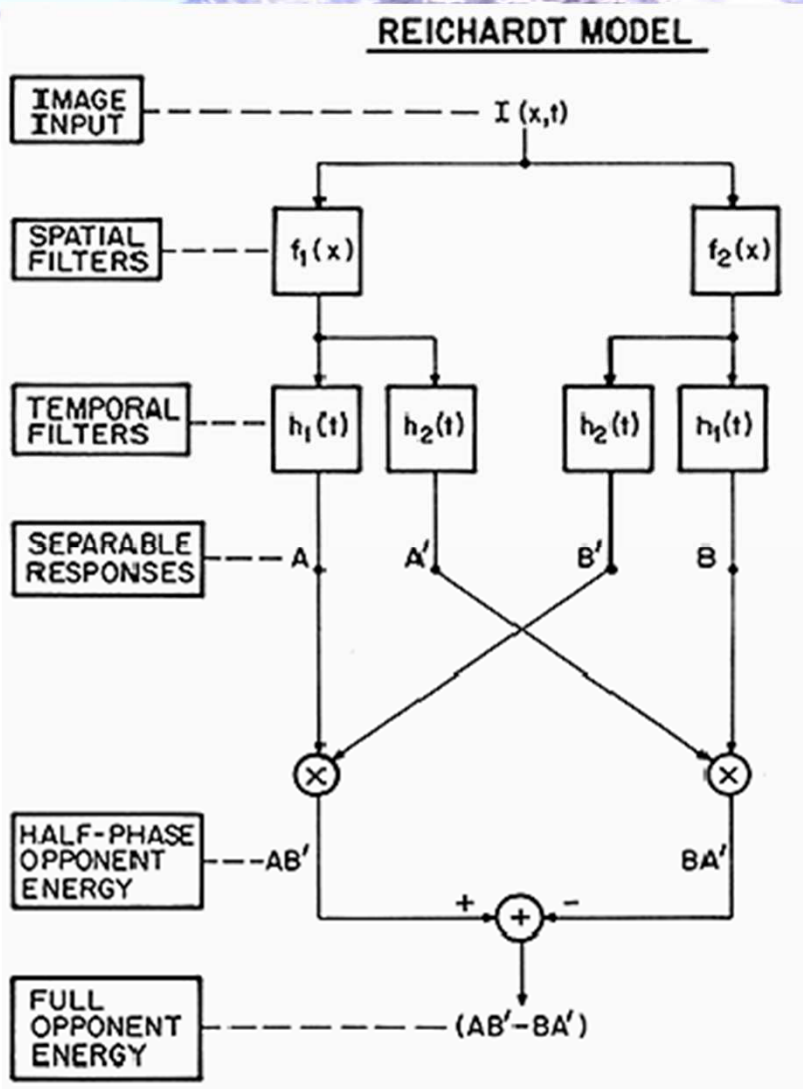
Analysis:

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



Reichardt detector

- QT movie

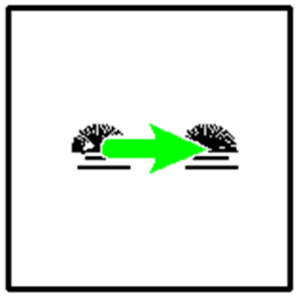


Several motion models

object based

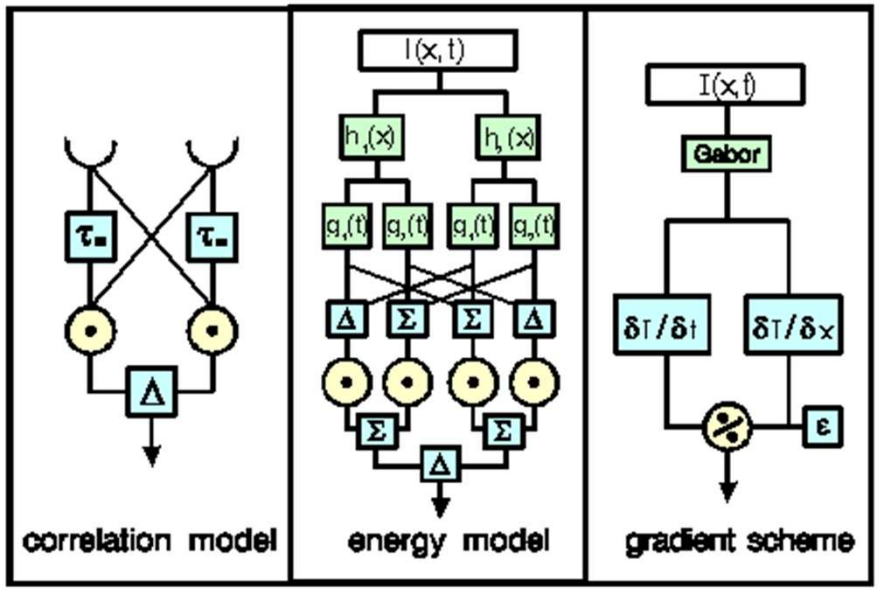
token matching

$V = \Delta x/\Delta t$



intensity based

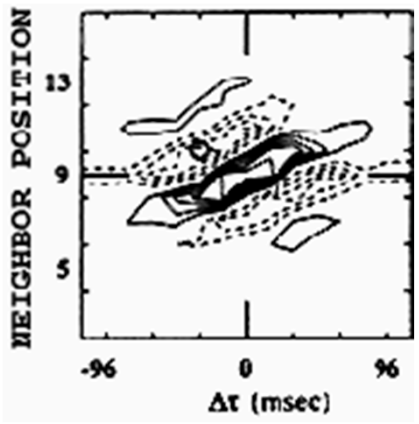
- Gradient: in Computer Vision
- Correlation: In bio vision
- Spatiotemporal filters: Unifying model

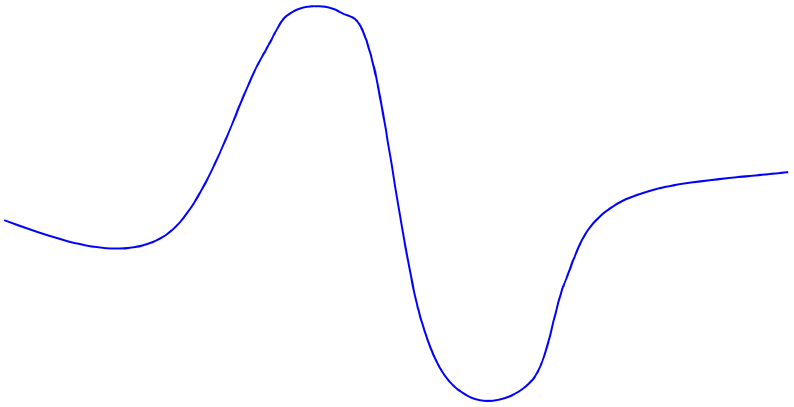


Spatial response: Gabor function

- Definition:

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





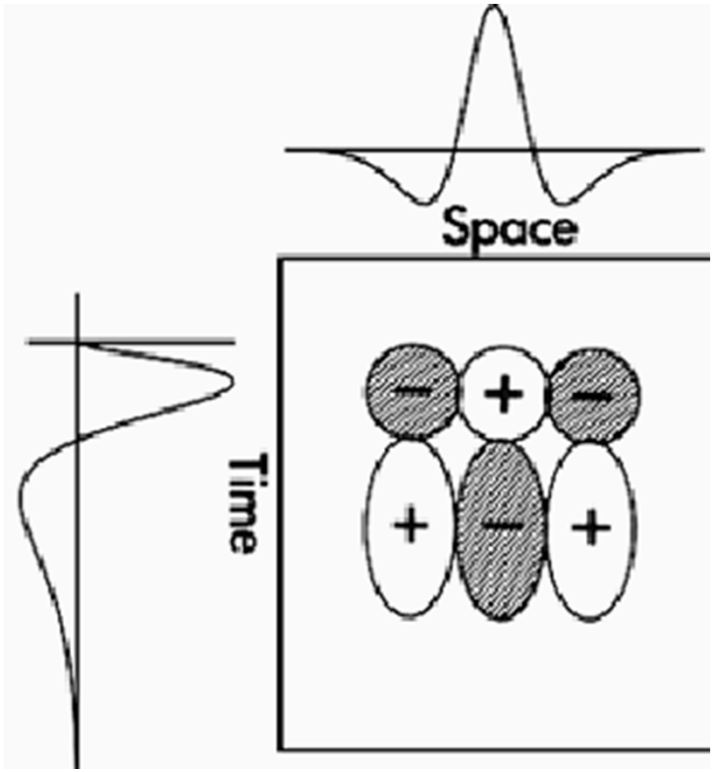
Temporal response:

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:

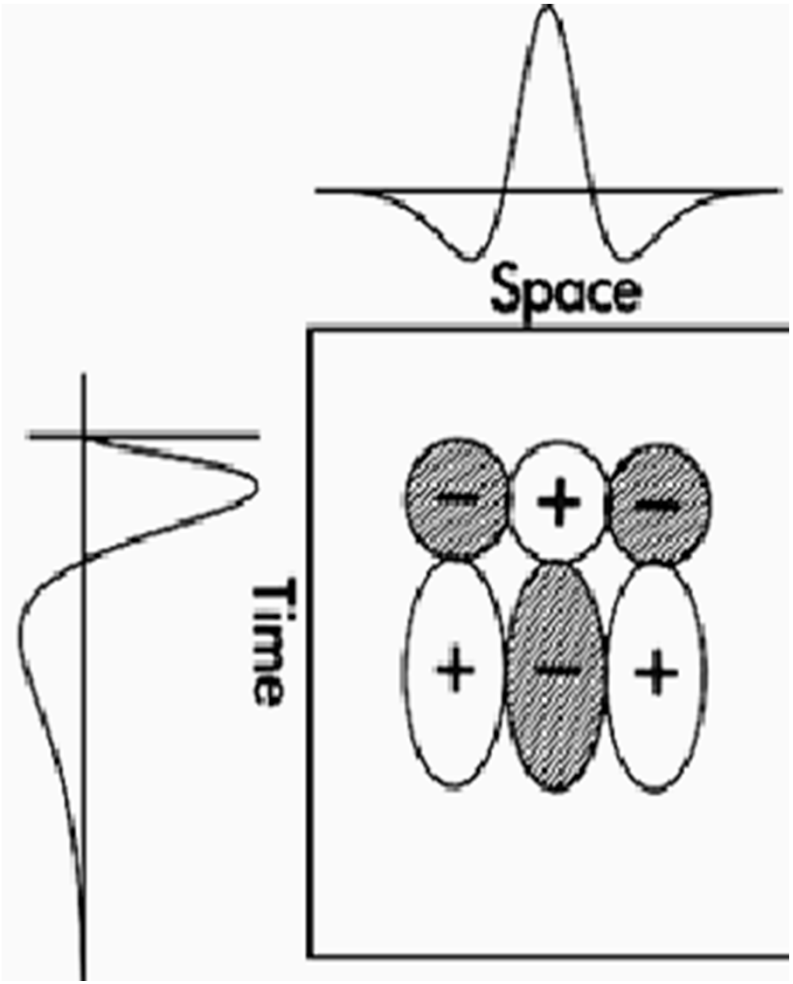
- For our grating: (Theta=0)

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

- Write as sum of components:

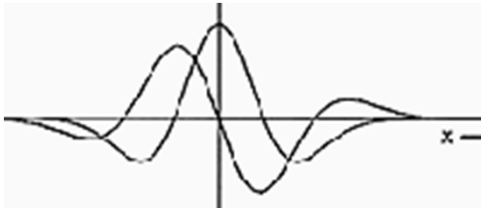
= exp(…)*(acos… + bsin…)

Space-time receptive field

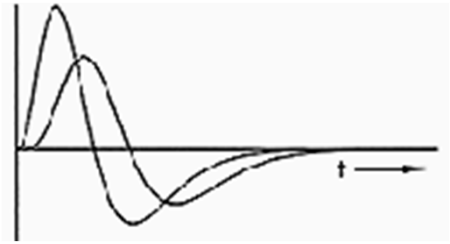


Combined cells

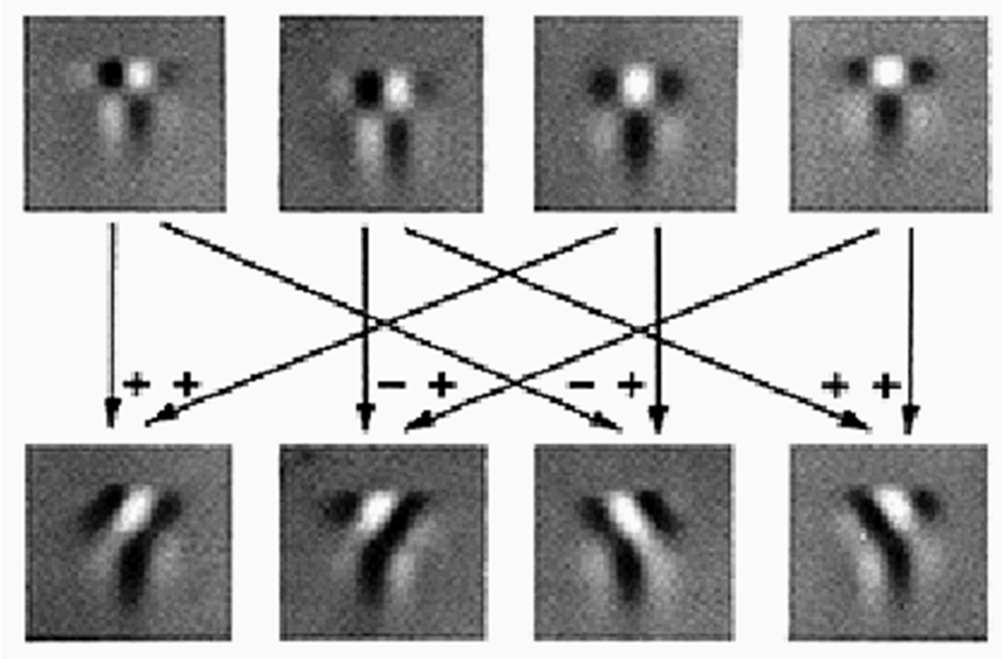
- Spat:



Temp:



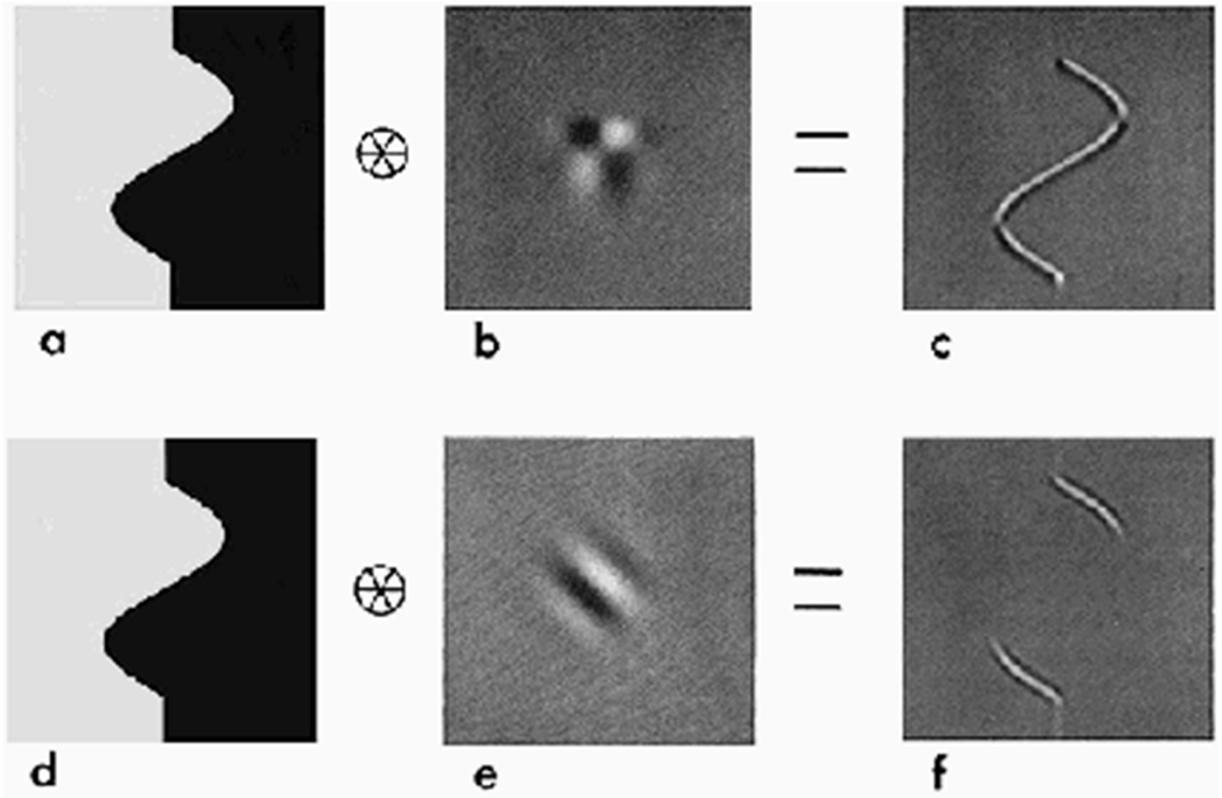
- Both:



- Comb:

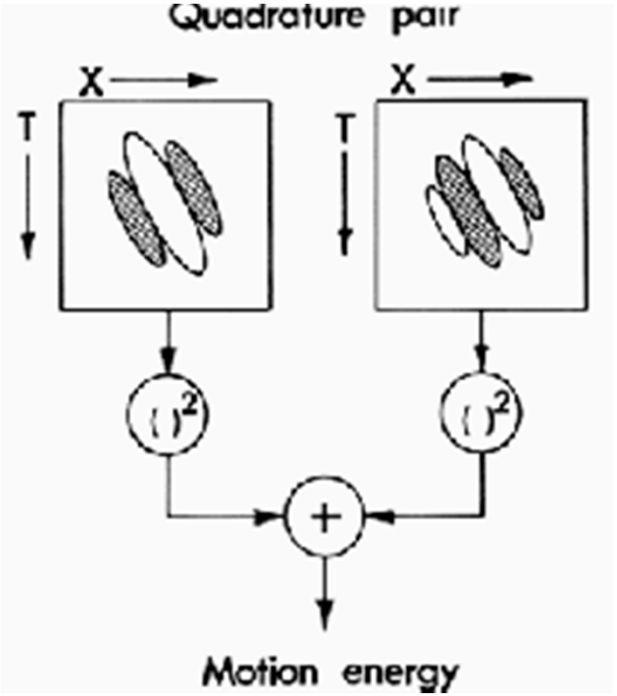
Result:

- More directionally specific response

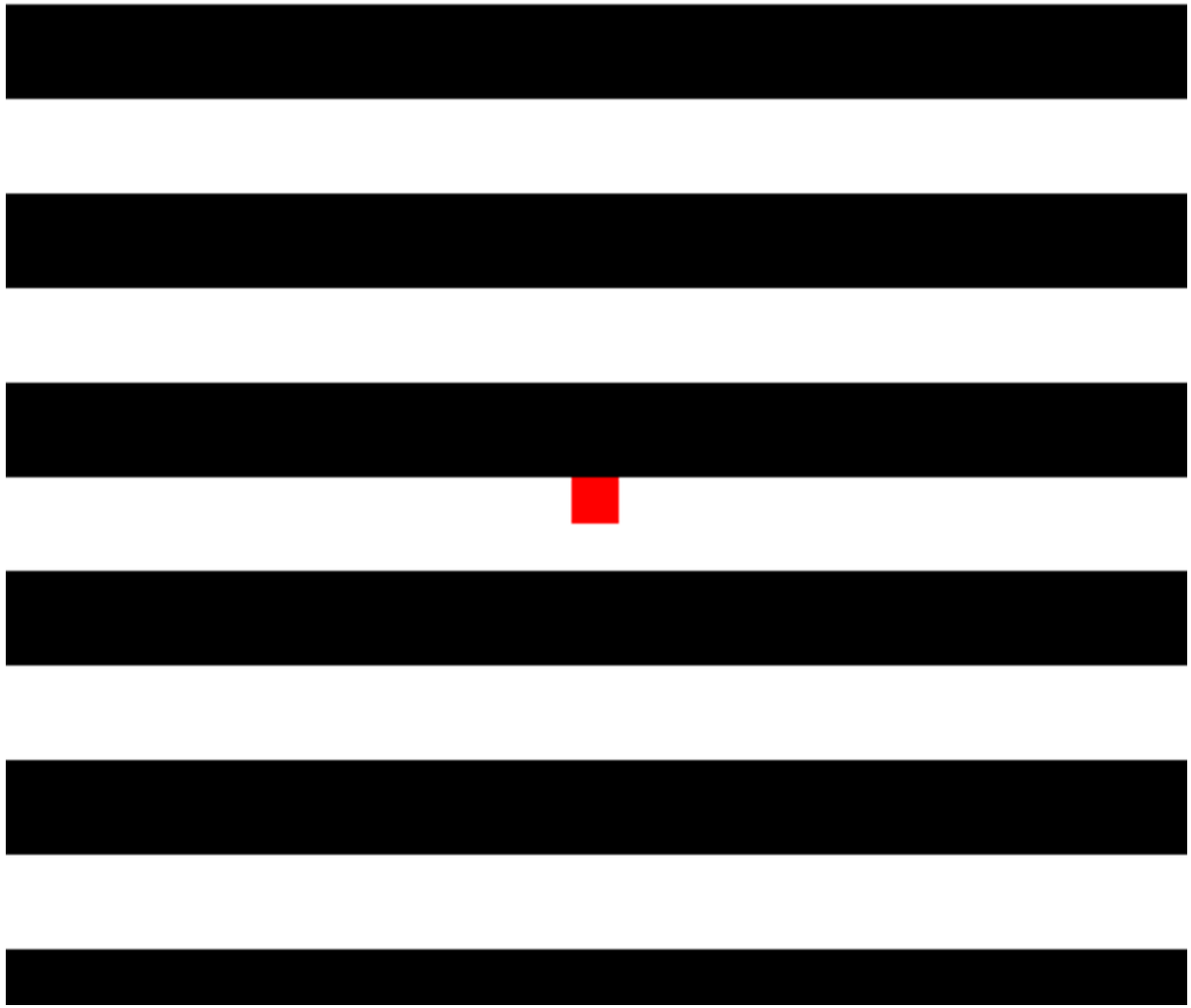


Energy model:

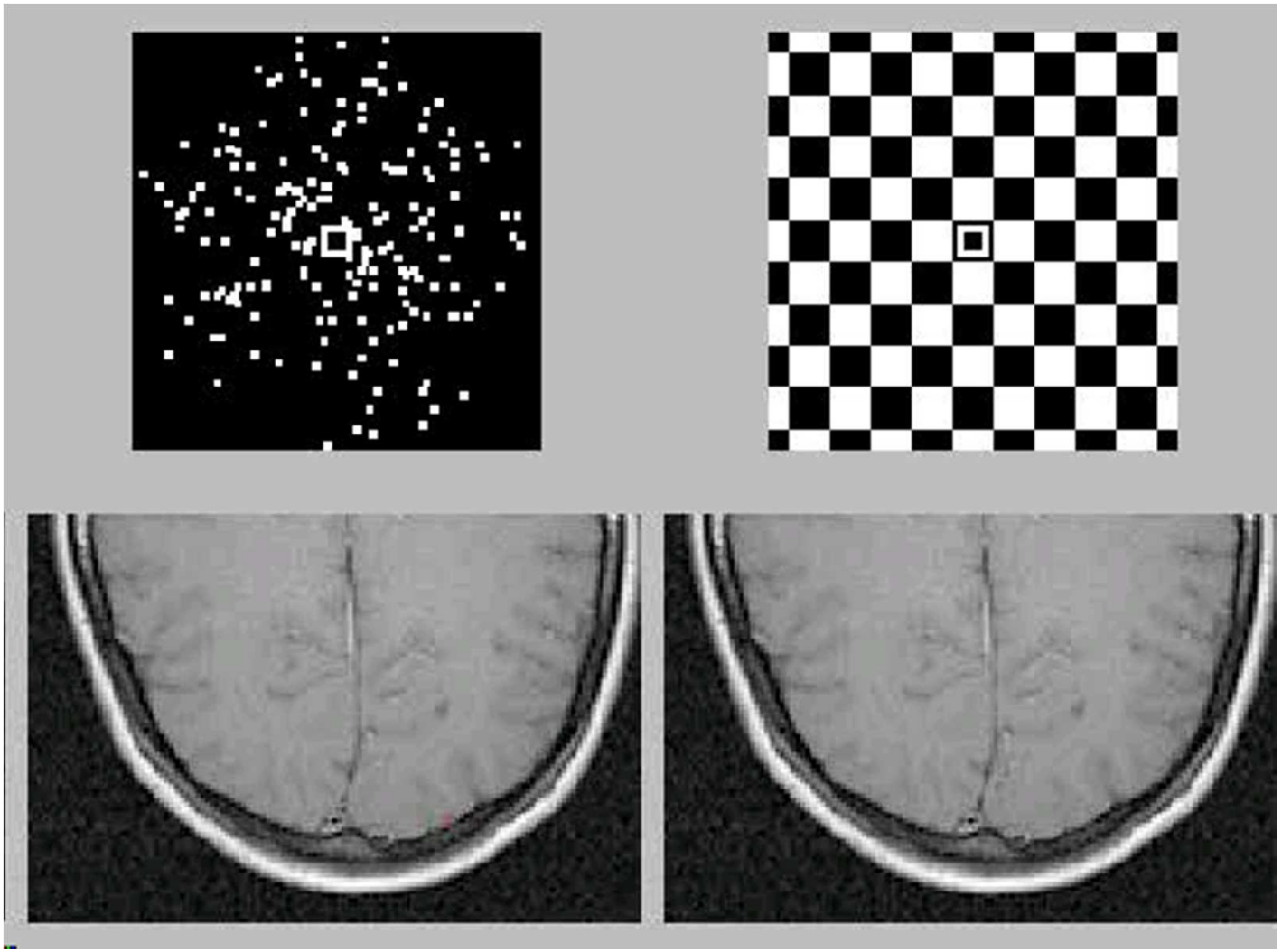
- Sum odd and even phase components
- Quadrature rectifier



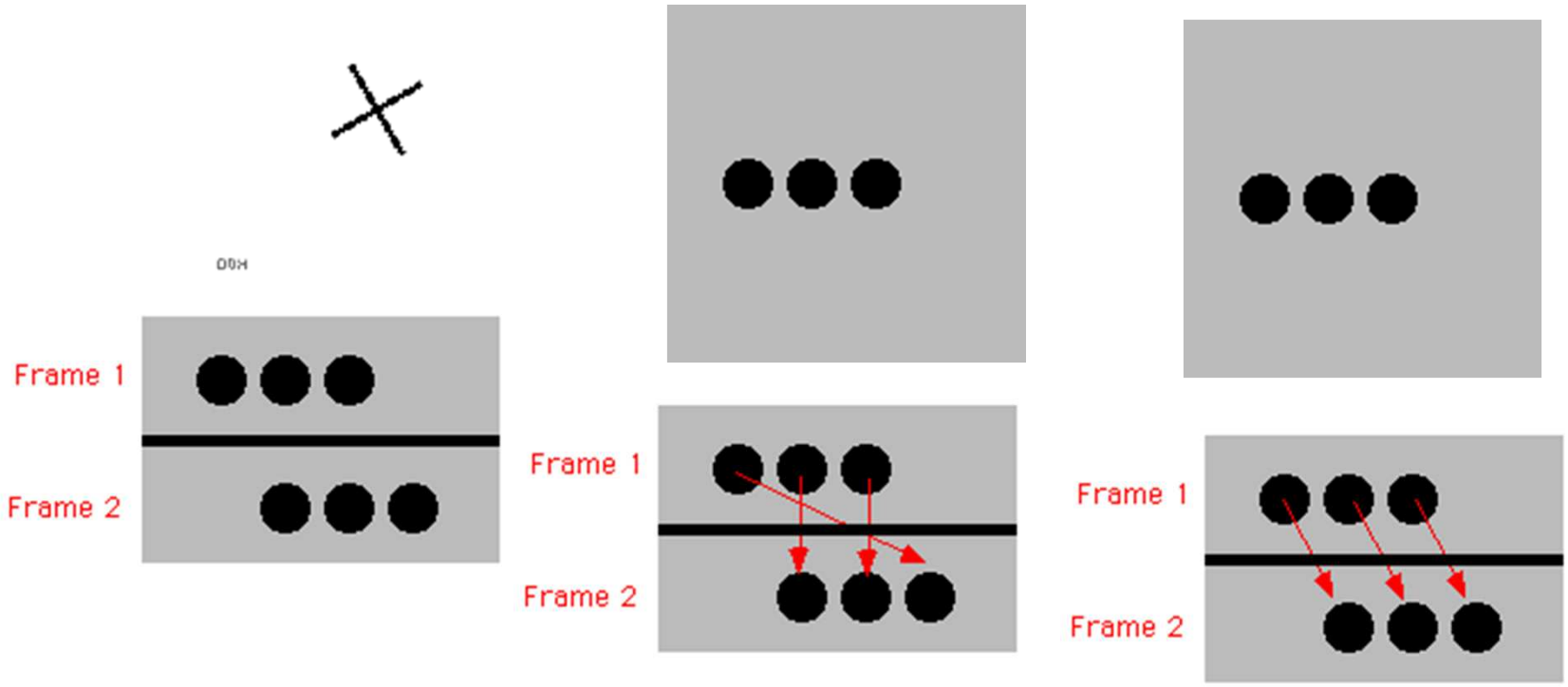
Adaption:
Motion aftereffect



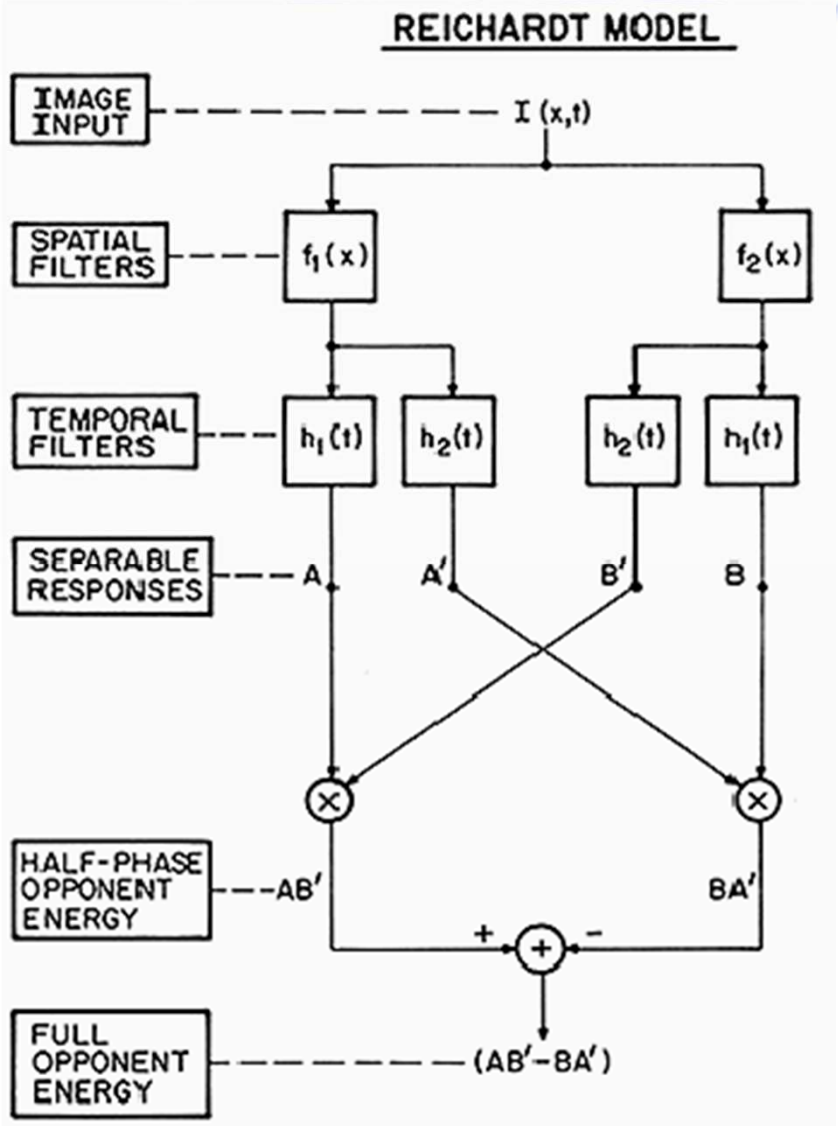
Where is motion processed?

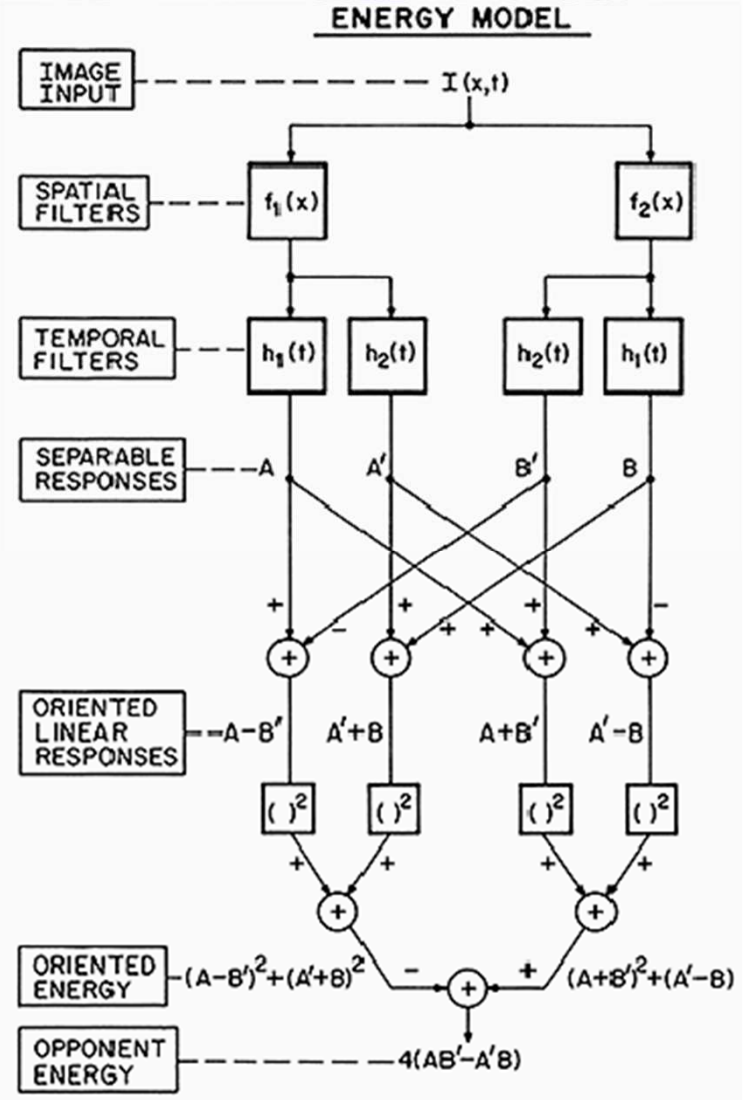


Higher effects:



Equivalence: Reich and Spat





Conclusion

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