

# CS 558: Computer Vision

## 6<sup>th</sup> Set of Notes

Instructor: Philippos Mordohai  
Webpage: [www.cs.stevens.edu/~mordohai](http://www.cs.stevens.edu/~mordohai)  
E-mail: [Philippos.Mordohai@stevens.edu](mailto:Philippos.Mordohai@stevens.edu)  
Office: Lieb 215

# Overview

- Image Transformations
  - Based on slides by R. Szeliski
- Feature Tracking
  - Based on slides by D. Hoiem
- Dense Optical Flow
  - Based on slides by K. Grauman
- Grouping and Segmentation
  - Based on slides by D. Hoiem

# Image Transformations

Slides by R. Szeliski

# Image Transformations

image filtering: change **range** of image

$$g(x) = T(f(x))$$

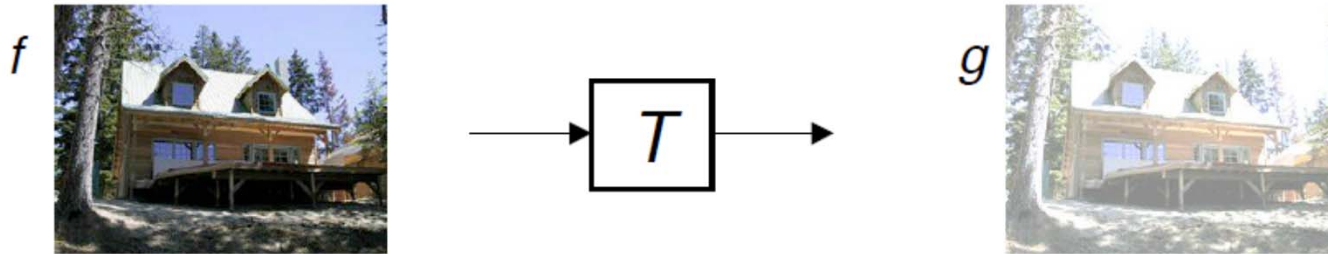
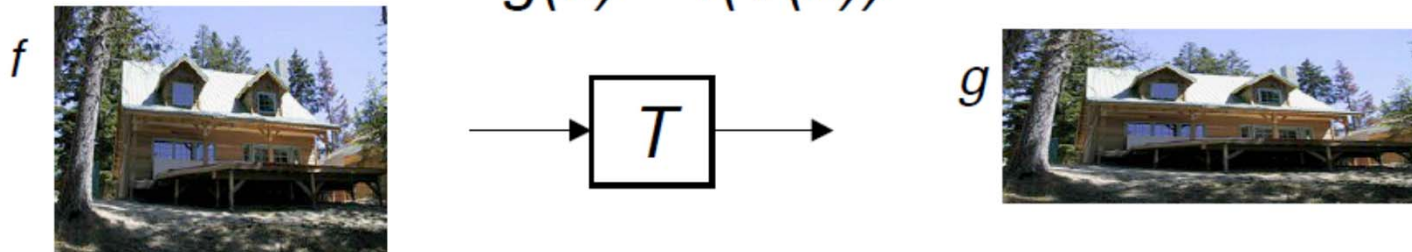


image warping: change **domain** of image

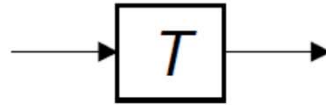
$$g(x) = f(T(x))$$



# Parametric (Global) Warping



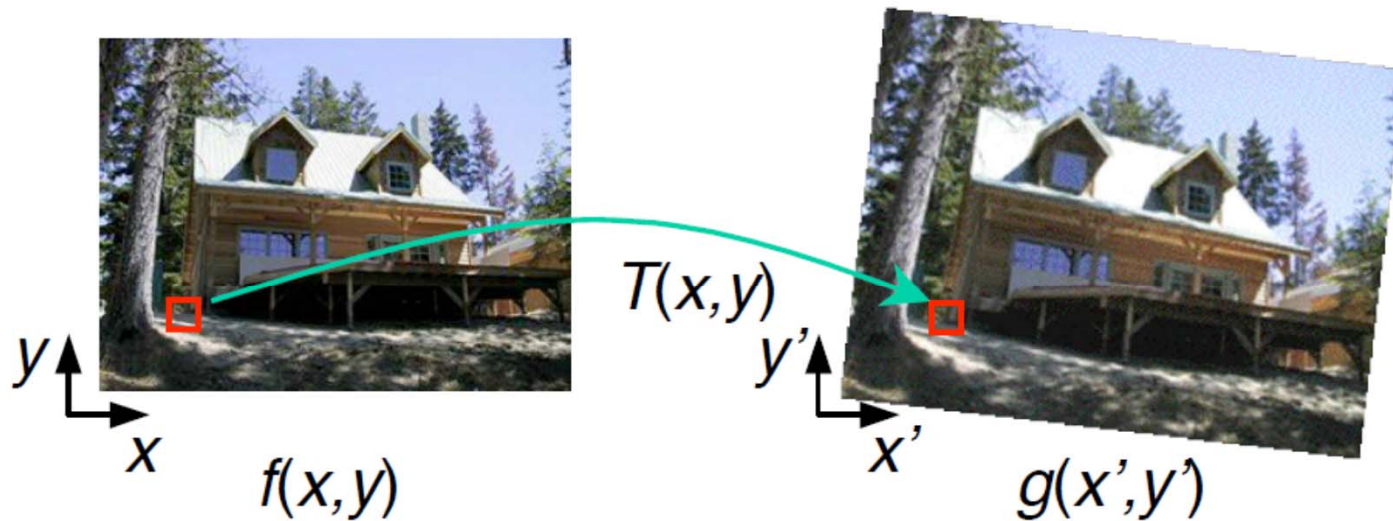
$$\mathbf{p} = (x, y)$$



$$\mathbf{p}' = (x', y')$$

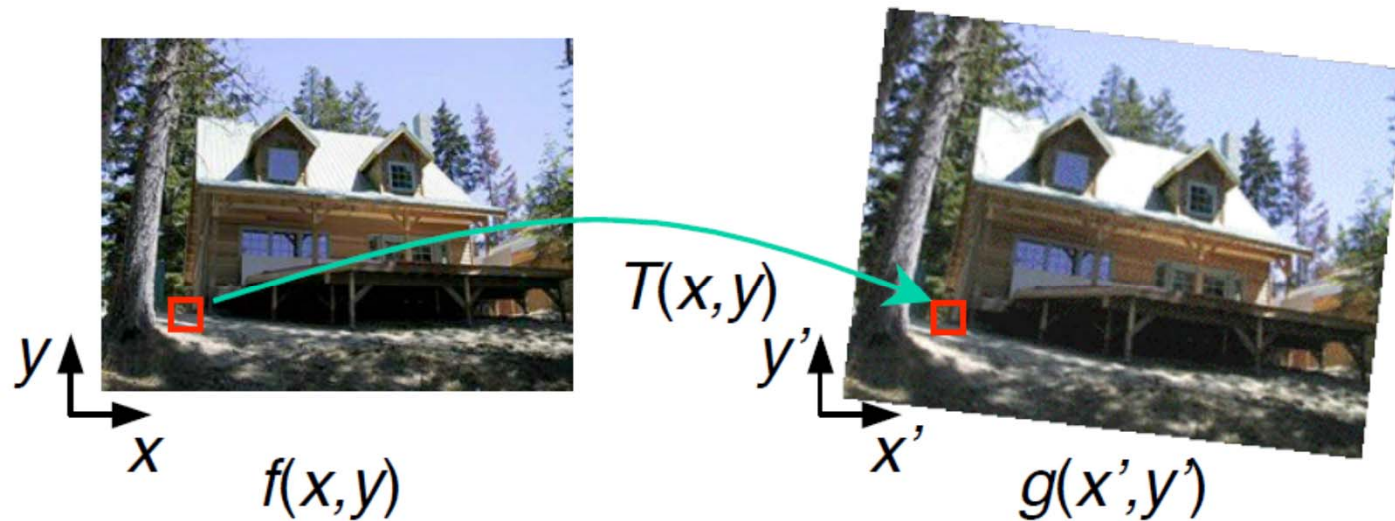
- Transformation  $T$  is a coordinate-changing machine:  
$$\mathbf{p}' = T(\mathbf{p})$$
- What does it mean that  $T$  is global?
  - It is the same for any point  $\mathbf{p}$
  - It can be described by just a few numbers (parameters)
- $T$  is represented as a matrix (see prev. slides):  
$$\mathbf{p}' = M * \mathbf{p}$$

# Image Warping



Given a coordinate transform  $(x',y') = h(x,y)$  and a source image  $f(x,y)$ , how do we compute a transformed image  $g(x',y') = f(T(x,y))$ ?

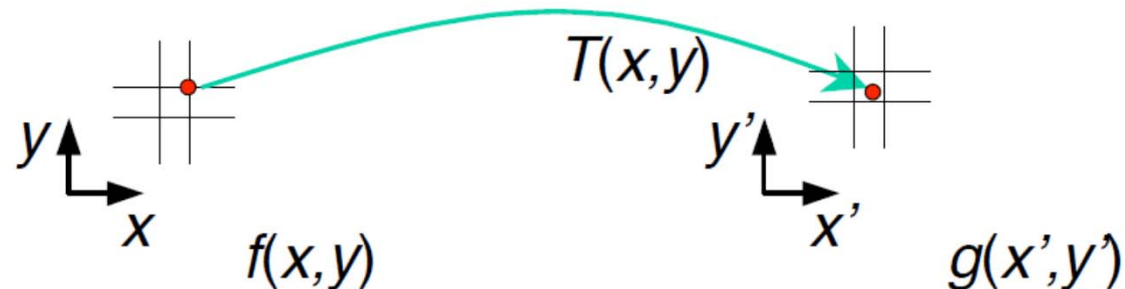
# Forward Warping



Send each pixel  $f(x,y)$  to its corresponding location  $(x',y') = T(x,y)$  in the second image

Q: what if the pixel lands “between” two pixels?

# Forward Warping



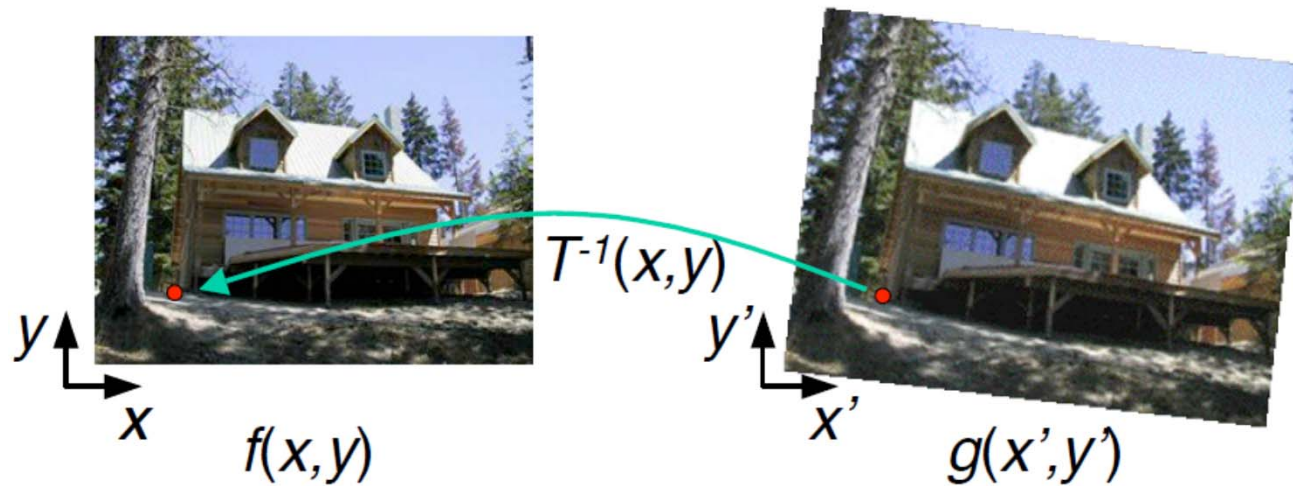
Send each pixel  $f(x,y)$  to its corresponding location  $(x',y') = T(x,y)$  in the second image

Q: what if the pixel lands “between” two pixels?

A: Distribute color among neighboring pixels (splatting)

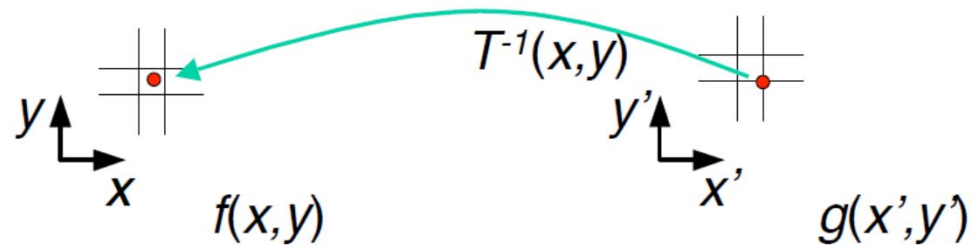


# Inverse Warping



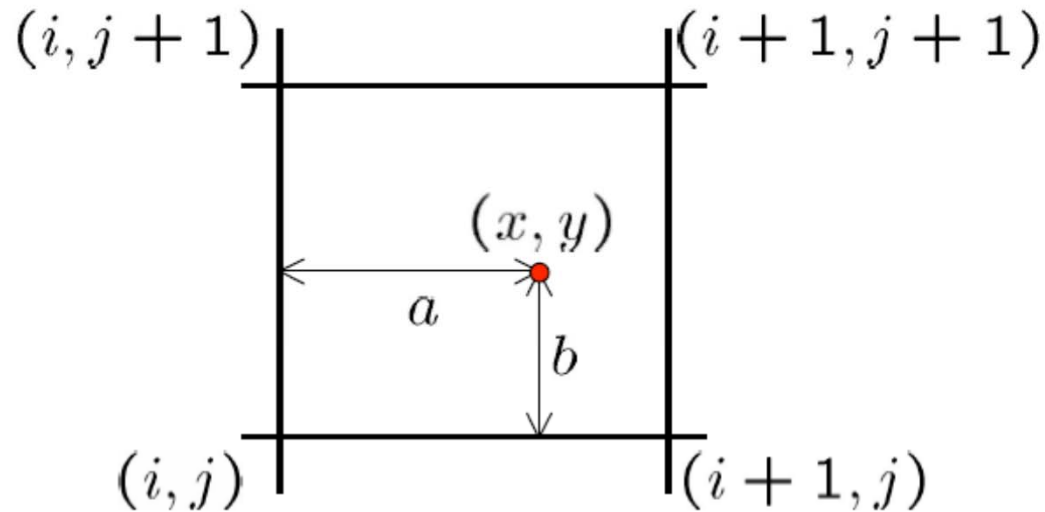
- Get each pixel  $g(x',y')$  from its corresponding location  $(x,y) = T^{-1}(x',y')$  in the first image
- Q: what if pixel comes from “between” two pixels?

# Inverse Warping



- Get each pixel  $g(x', y')$  from its corresponding location  $(x, y) = T^{-1}(x', y')$  in the first image
- Q: what if pixel comes from “between” two pixels?
- A: interpolate color value from neighbors
  - **Bilinear interpolation** typically used

# Bilinear Interpolation



$$\begin{aligned} f(x, y) = & (1 - a)(1 - b) f[i, j] \\ & + a(1 - b) f[i + 1, j] \\ & + ab f[i + 1, j + 1] \\ & + (1 - a)b f[i, j + 1] \end{aligned}$$

# Forward vs. Inverse Warping

- Which is better?
- ...

# Feature tracking

Mostly based on slides by Derek Hoiem, also partially based on sources by C. Tomasi, T. Kanade and T. Svoboda

# Feature Matching

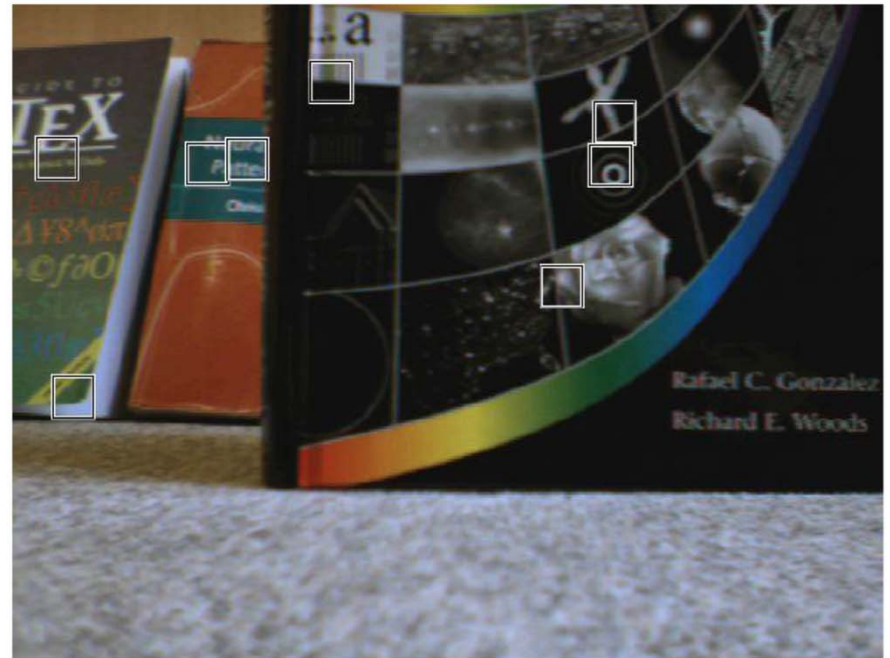
- Given a feature in one image, how to find the best match in another image?
  - Assumption images are consecutive frames in video
- So far we have searched for best match
  - By testing all possible translations by integer number of pixels (template matching)
  - By comparing rotation-invariant descriptors (SIFT)

# Camera Motion

I

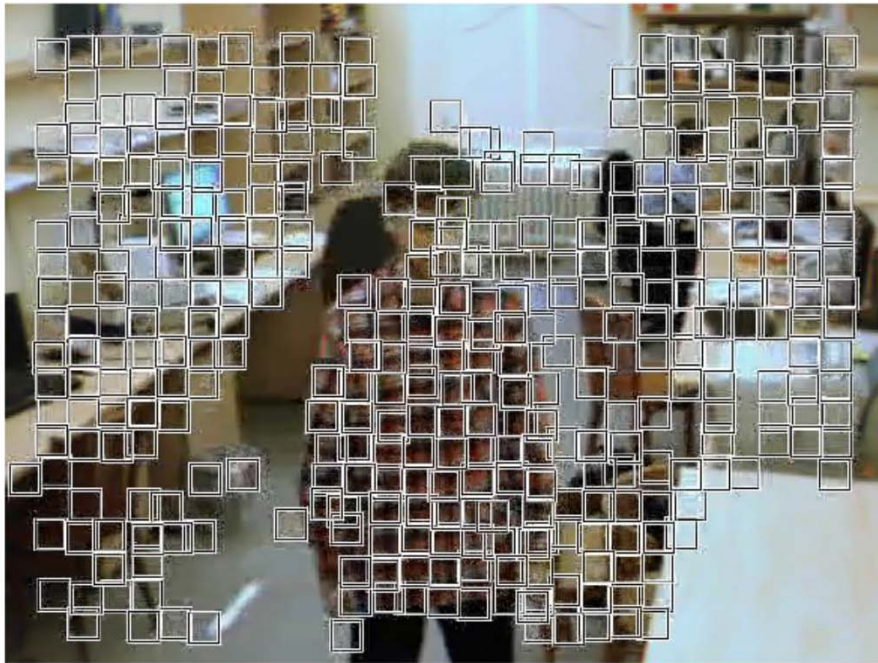


J

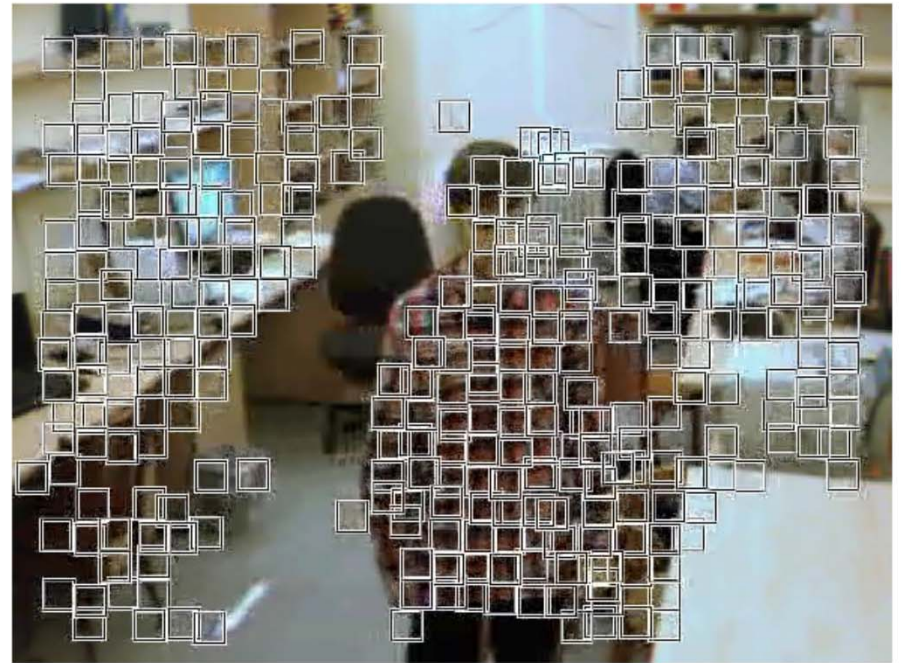


# Object Motion

I



J

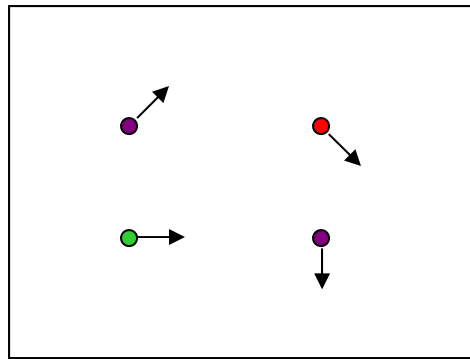




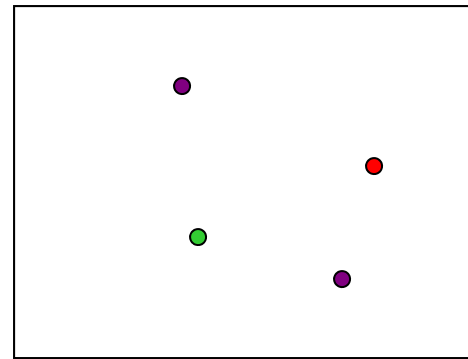
# Feature Tracking

- Challenges
  - Figure out which features can be tracked
  - Efficiently track across frames
  - Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
  - Drift: small errors can accumulate as appearance model is updated
  - Points may appear or disappear: need to be able to add/delete tracked points

# Feature Tracking



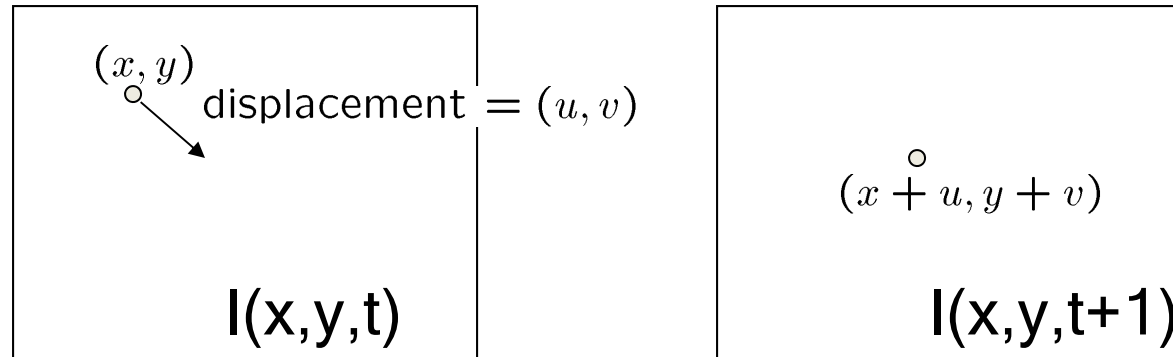
$I(x,y,t)$



$I(x,y,t+1)$

- Given two subsequent frames, estimate the point translation
- Key assumptions of Lucas-Kanade Tracker
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

# The Brightness Constancy Constraint



- Brightness Constancy Equation:

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

Take Taylor expansion of  $I(x+u, y+v, t+1)$  at  $(x,y,t)$  to linearize the right side:

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + \overset{\text{Image derivative along x}}{I_x} \cdot u + I_y \cdot v + \overset{\text{Difference over frames}}{I_t}$$

$$I(x + u, y + v, t + 1) - I(x, y, t) = I_x \cdot u + I_y \cdot v + I_t$$

$$\text{Hence, } I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \quad \nabla I \cdot [u \ v]^T + I_t = 0$$

# The Brightness Constancy Constraint

Can we use this equation to recover image motion  $(u,v)$  at each pixel?

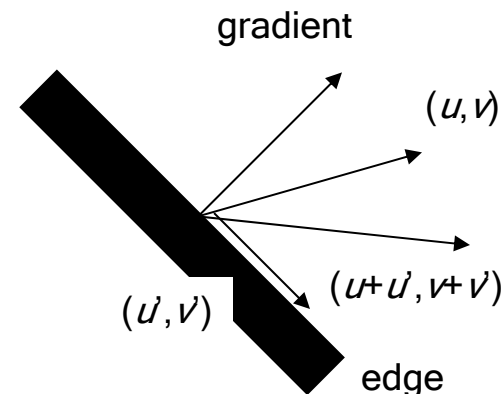
$$\nabla I \cdot [u \ v]^T + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns  $(u,v)$

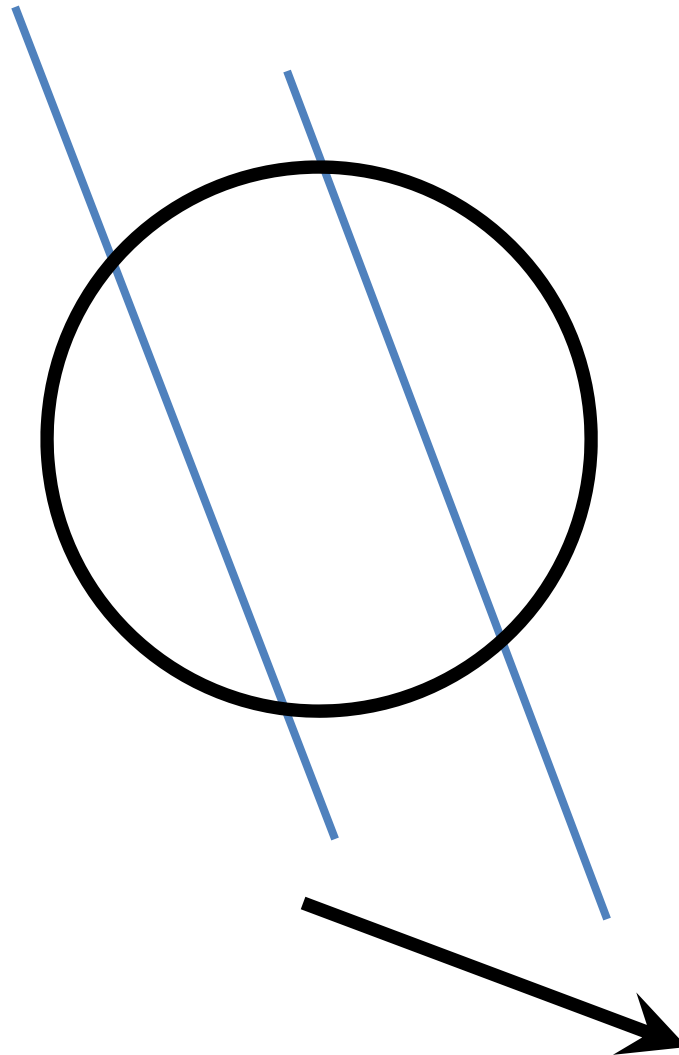
The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If  $(u, v)$  satisfies the equation,  
so does  $(u+u', v+v')$  if

$$\nabla I \cdot [u' \ v']^T = 0$$

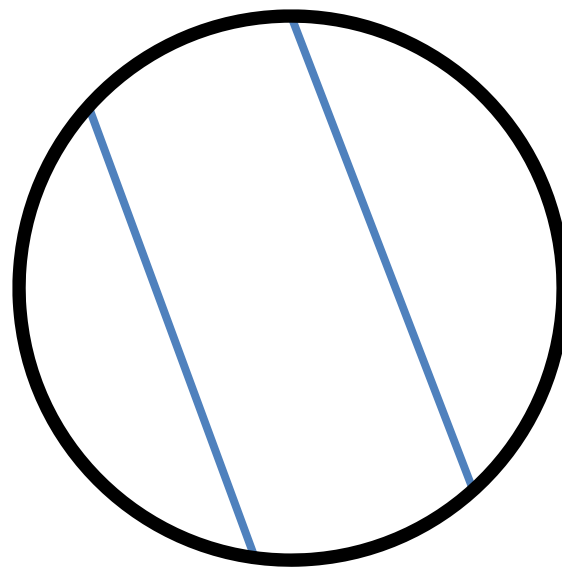


# The Aperture Problem



**Actual motion**

# The Aperture Problem



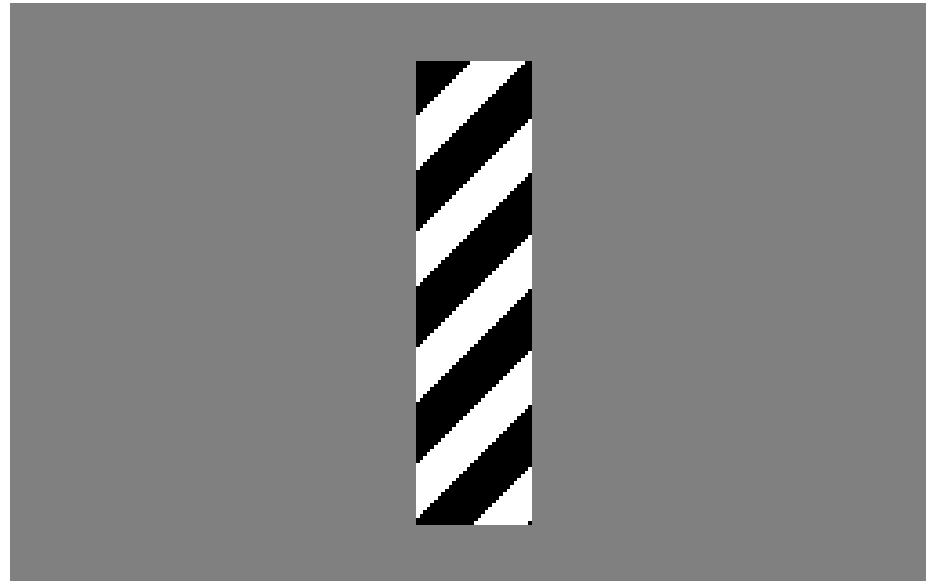
**Perceived motion**

# The Barber Pole Illusion



[http://en.wikipedia.org/wiki/Barberpole\\_illusion](http://en.wikipedia.org/wiki/Barberpole_illusion)

# The Barber Pole Illusion



[http://en.wikipedia.org/wiki/Barberpole\\_illusion](http://en.wikipedia.org/wiki/Barberpole_illusion)



# Solving the Ambiguity...

- How to get more equations for a pixel?
- **Spatial coherence constraint**
- Assume the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

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

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

# Solving the Ambiguity...

- Least squares problem:

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \quad \begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{matrix}$$

# Matching Patches across Images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \quad \begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{matrix}$$

Least squares solution for  $d$  given by  $(A^T A) d = A^T b$

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

$$\begin{matrix} A^T A & A^T b \end{matrix}$$

The summations are over all pixels in the  $K \times K$  window

# Conditions for Solvability

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

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

$A^T A$   $A^T b$

When is this solvable? I.e., what are good points to track?

- $A^T A$  should be invertible
- $A^T A$  should not be too small due to noise
  - eigenvalues  $\lambda_1$  and  $\lambda_2$  of  $A^T A$  should not be too small
- $A^T A$  should be well-conditioned
  - $\lambda_1 / \lambda_2$  should not be too large ( $\lambda_1 =$  largest eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

$M = A^T A$  is the *second moment matrix* !  
(Harris corner detector...)

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

- **Eigenvectors and eigenvalues of  $A^T A$  relate to edge direction and magnitude**
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it

# Low-texture region



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

- gradients have small magnitude
- small  $\lambda_1$ , small  $\lambda_2$

# Edge



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

- gradients very large or very small
- large  $\lambda_1$ , small  $\lambda_2$

# High-texture Region

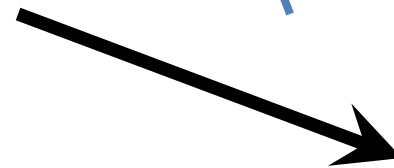
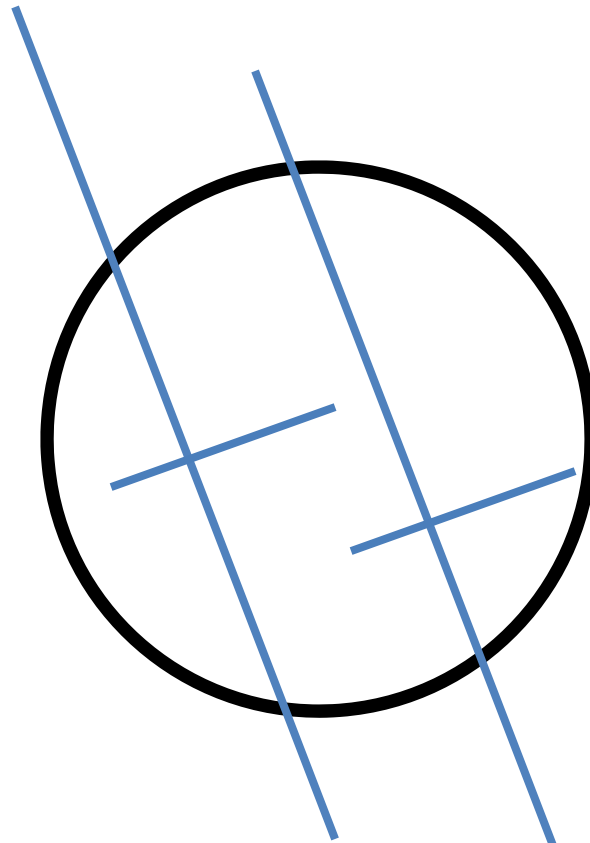


$$\sum \nabla I (\nabla I)^T$$

- gradients are different, large magnitudes
- large  $\lambda_1$ , large  $\lambda_2$

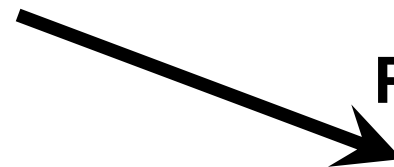
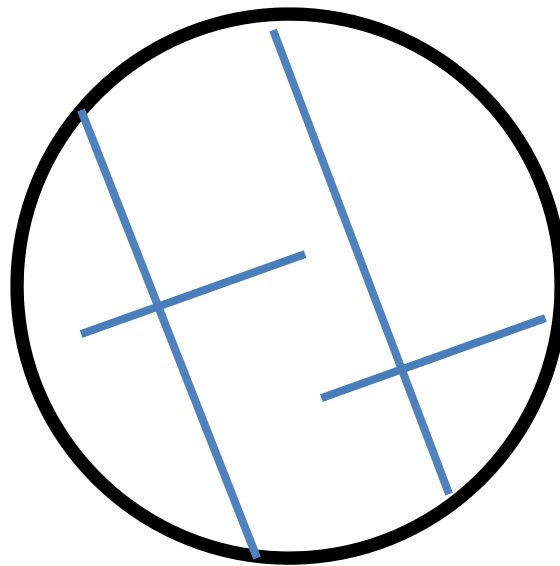


# The Aperture Problem Resolved



**Actual motion**

# The Aperture Problem Resolved



**Perceived motion**

# Dealing with Larger Motion: Iterative Refinement

1. Initialize  $(x', y') = (x, y)$
2. Compute  $(u, v)$  by

Original  $(x, y)$  position

$$I_t = I(x', y', t+1) - I(x, y, t)$$

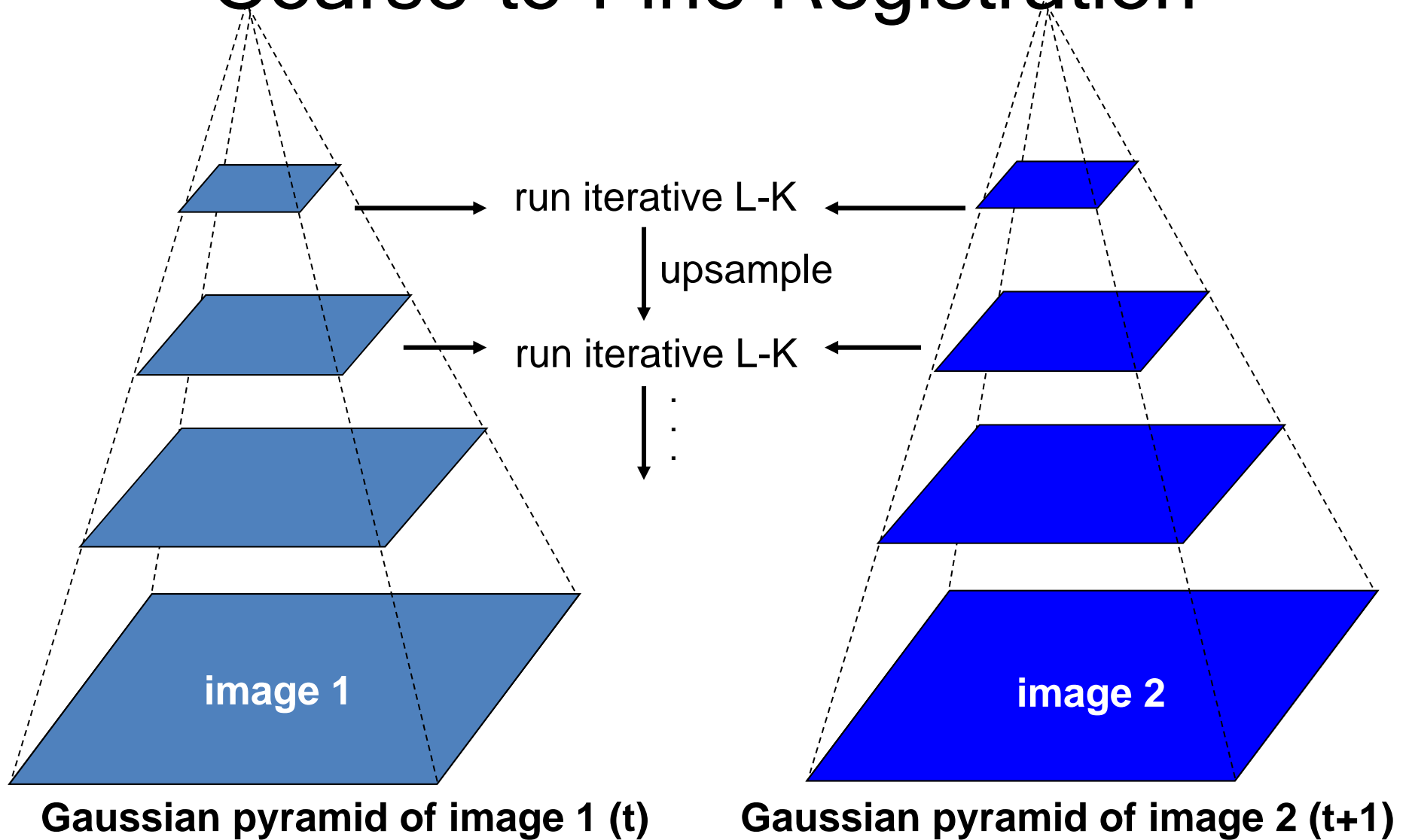
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

2<sup>nd</sup> moment matrix for feature  
patch in first image

displacement

3. Shift window by  $(u, v)$ :  $x' = x' + u$ ;  $y' = y' + v$ ;
4. Recalculate  $I_t$
5. Repeat steps 2-4 until change is small
  - Use interpolation for subpixel values

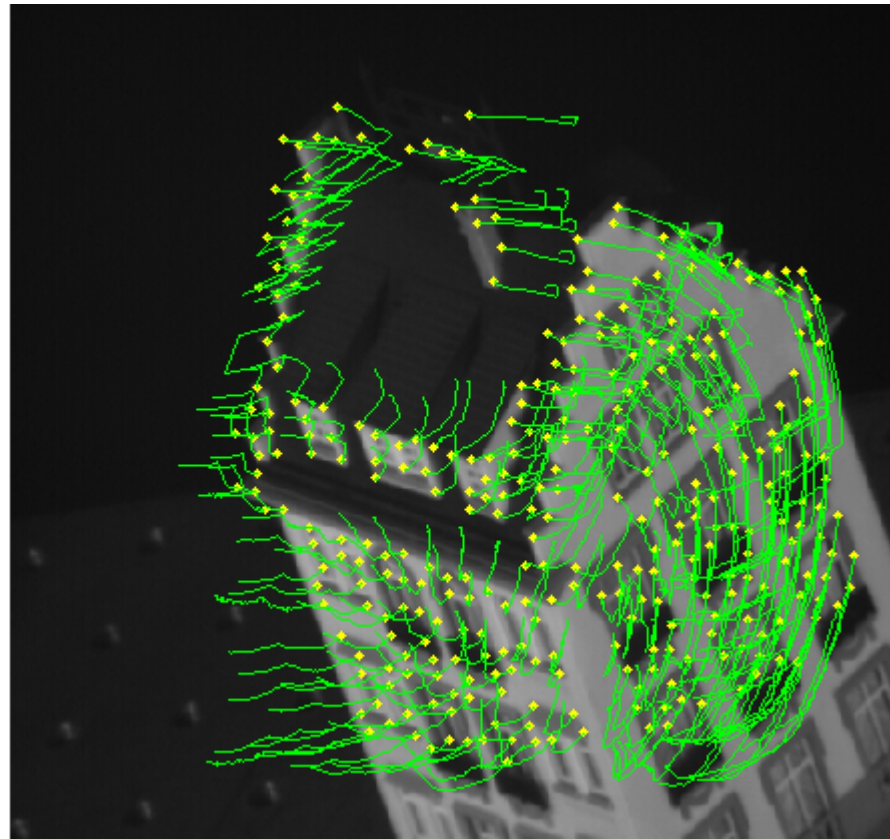
# Dealing with Larger Motion: Coarse-to-Fine Registration



# Shi-Tomasi Feature Tracker

- Find good features using eigenvalues of second-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)
  - Key idea: “good” features to track are the ones whose motion can be estimated reliably
- Track from frame to frame with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement
- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

# Feature Tracking Examples



# Feature Tracking Examples

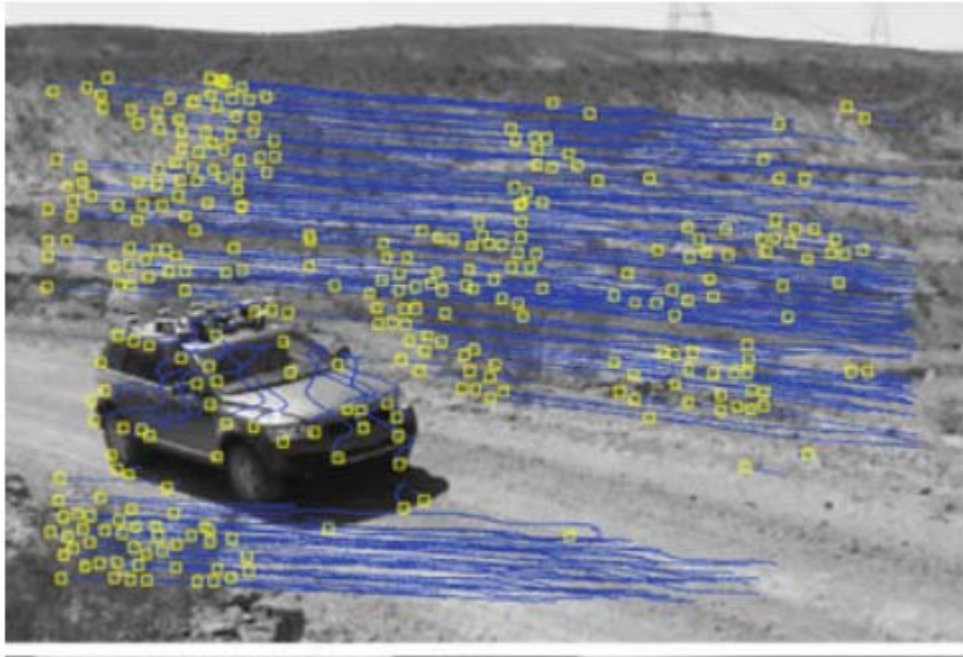
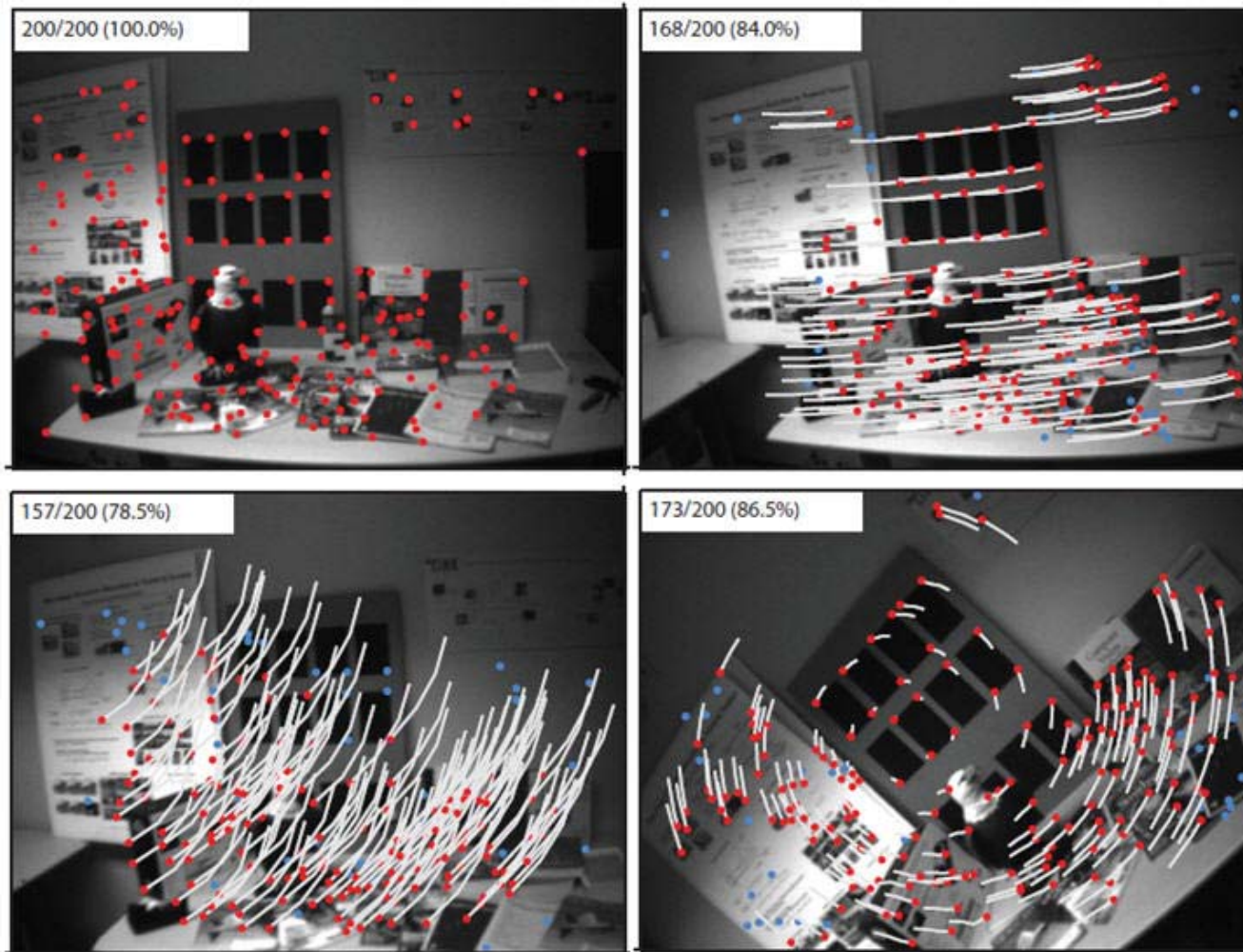


Image by Sudipta Sinha et al.

# Feature Tracking Examples



Gyro-assisted KLT tracking. Image by Myung Hwangbo



# Summary of KLT tracking

- Find a good point to track (Harris corners)
- Use intensity second moment matrix and difference across frames to find displacement
- Iterate and use coarse-to-fine search to deal with larger movements
- When creating long tracks, check appearance of registered patch against appearance of initial patch to find points that have drifted

# Implementation issues

- Window size
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)

# Why not just do local template matching?

- Slow (need to check more locations)
- Does not give subpixel alignment (or becomes much slower)
  - Even pixel alignment may not be good enough to prevent drift
- May be useful as a step in tracking if there are large motions

# Kanade-Lucas-Tomasi Tracking

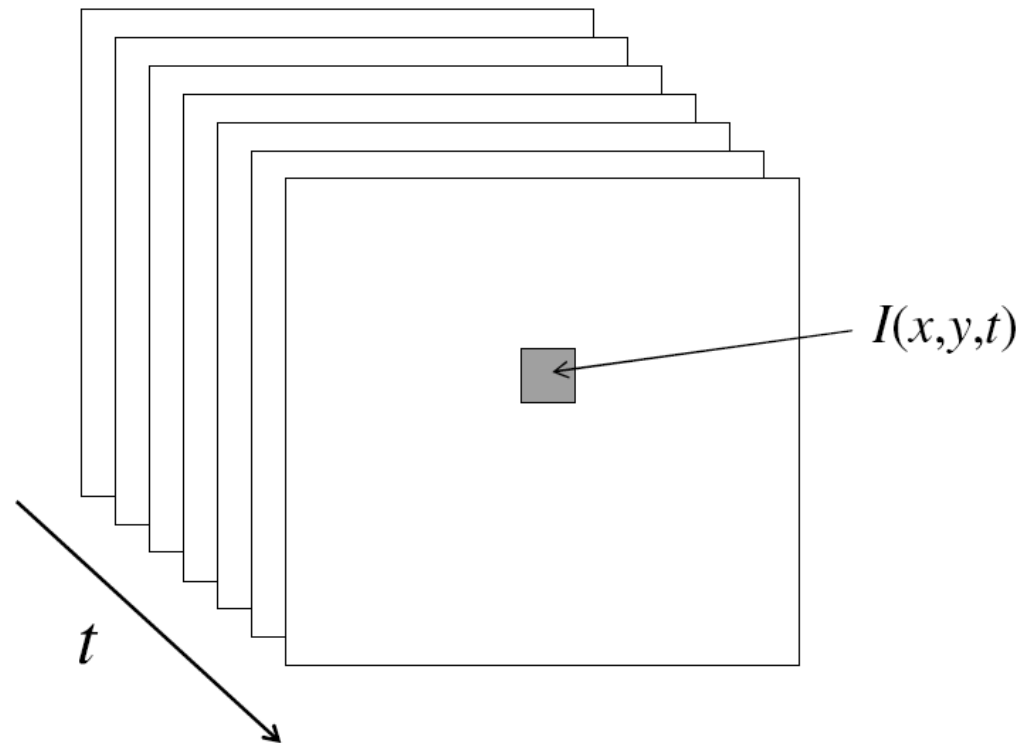
- Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Conference on Artificial Intelligence, pages 674-679, August 1981.
- Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.
- Jianbo Shi and Carlo Tomasi. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 593-600, 1994.
- Code: <http://www.ces.clemson.edu/~stb/klt/>

# Dense Motion Estimation

Based on slides by K. Grauman

# Video

- A video is a sequence of frames captured over time
- Now our image data is a function of space  $(x, y)$  and time  $(t)$

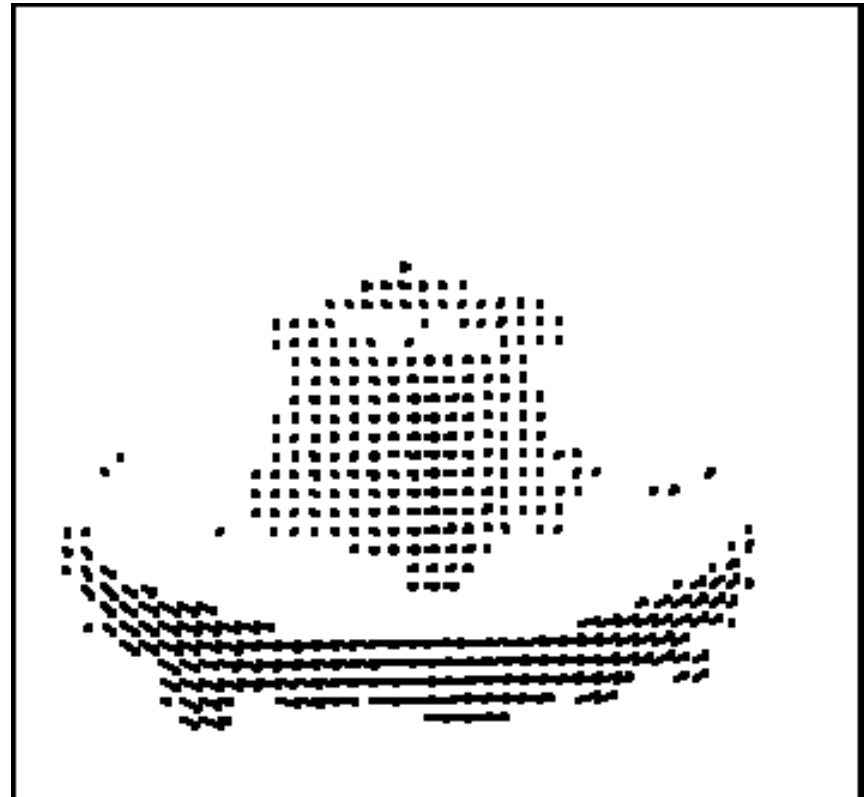
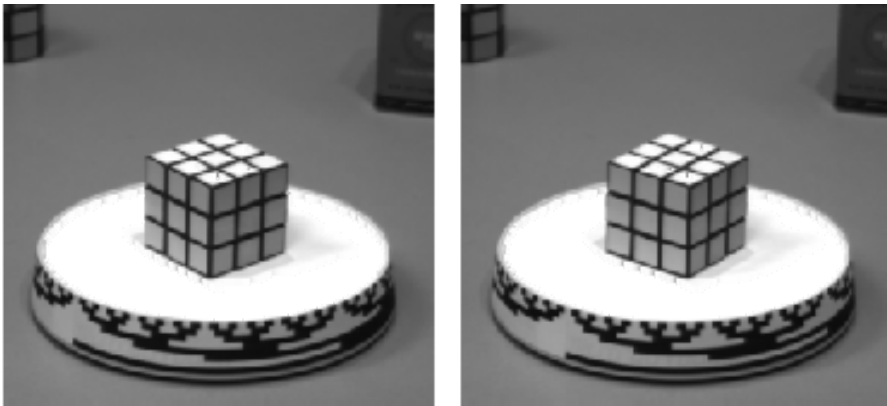


# Uses of motion

- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning dynamical models
- Recognizing events and activities
- Improving video quality (motion stabilization)

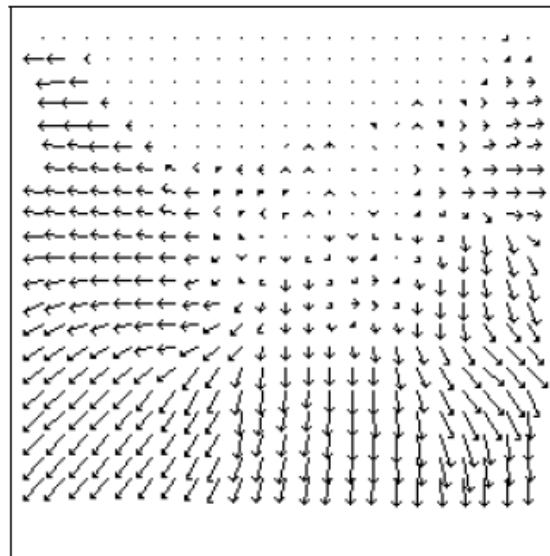
# Motion field

- The motion field is the projection of the 3D scene motion into the image

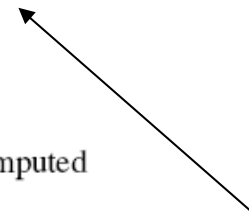




# Motion field + camera motion



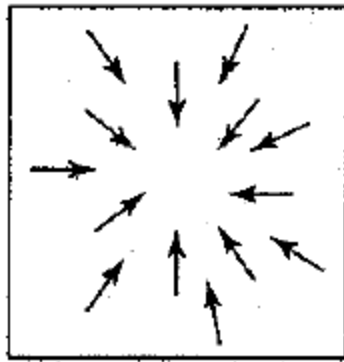
Length of flow vectors inversely proportional to depth  $Z$  of 3d point



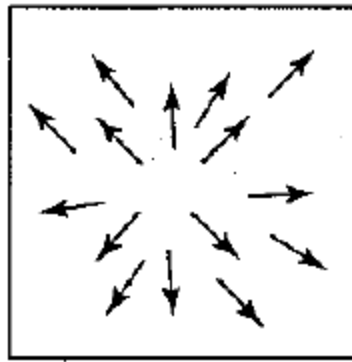
points closer to the camera move more quickly across the image plane

Figure 1.2: Two images taken from a helicopter flying through a canyon and the computed optical flow field.

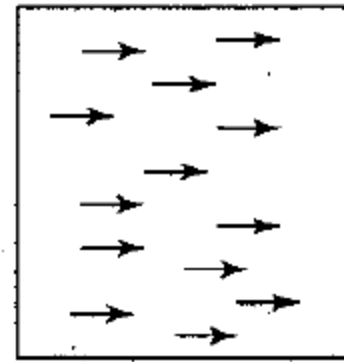
# Motion field + camera motion



**Zoom out**



**Zoom in**



**Pan right to left**

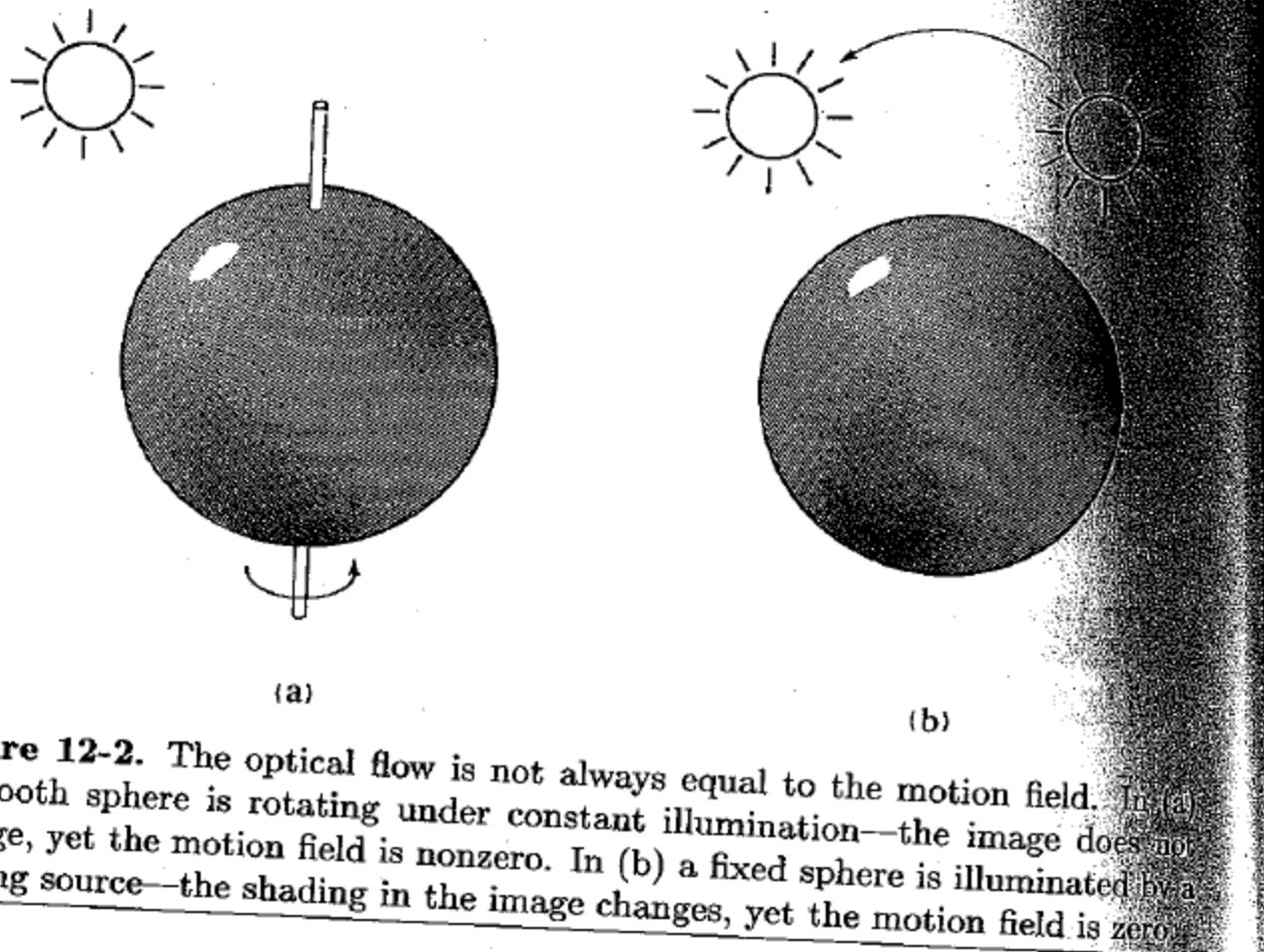
# Motion estimation techniques

- Direct methods
  - Directly recover image motion at each pixel from spatio-temporal image brightness variations
  - Dense motion fields, but sensitive to appearance variations
  - Suitable for video and when image motion is small
- Feature-based methods
  - Extract visual features (corners, textured areas) and track them over multiple frames
  - Sparse motion fields, but more robust tracking
  - Suitable when image motion is large (10s of pixels)

# Optical Flow

- Definition: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion

# Apparent motion $\neq$ motion field



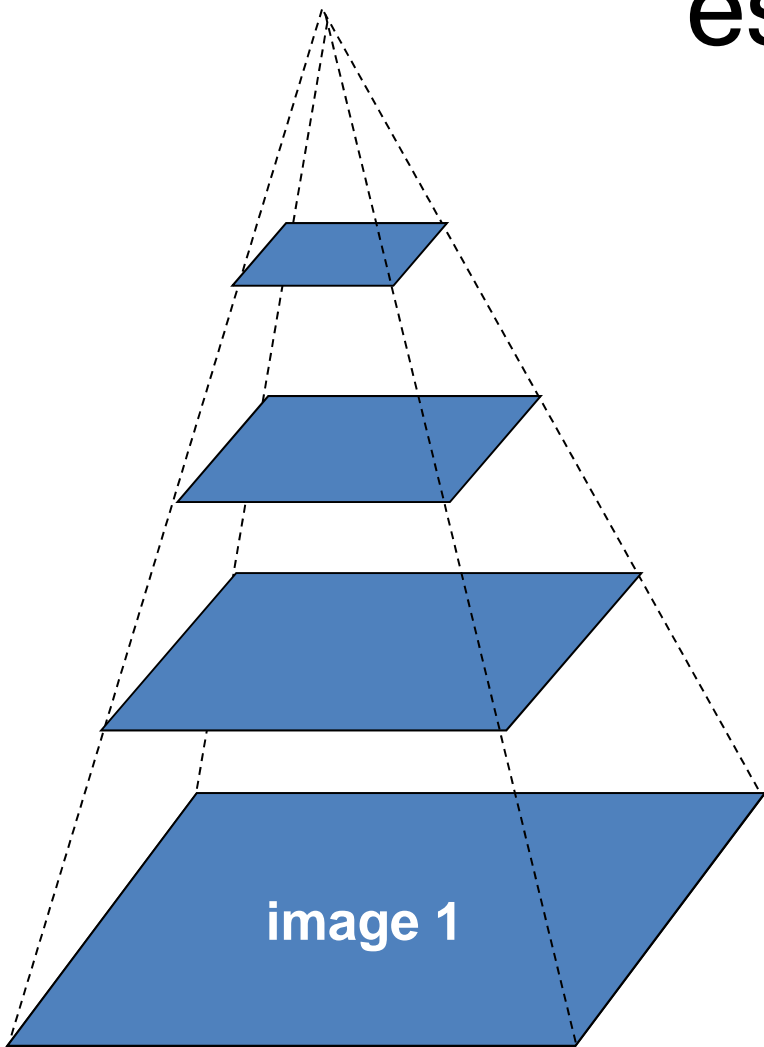
# Lucas-Kanade Optical Flow

- Same as Lucas-Kanade feature tracking, but for each pixel
  - As we saw, works better for textured pixels
- Operations can be done one frame at a time, rather than pixel by pixel
  - Efficient

# Iterative Refinement

- Iterative Lukas-Kanade Algorithm
  1. Estimate displacement at each pixel by solving Lucas-Kanade equations
  2. Warp  $I(t)$  towards  $I(t+1)$  using the estimated flow field
    - Basically, just interpolation
  3. Repeat until convergence

# Coarse-to-fine optical flow estimation



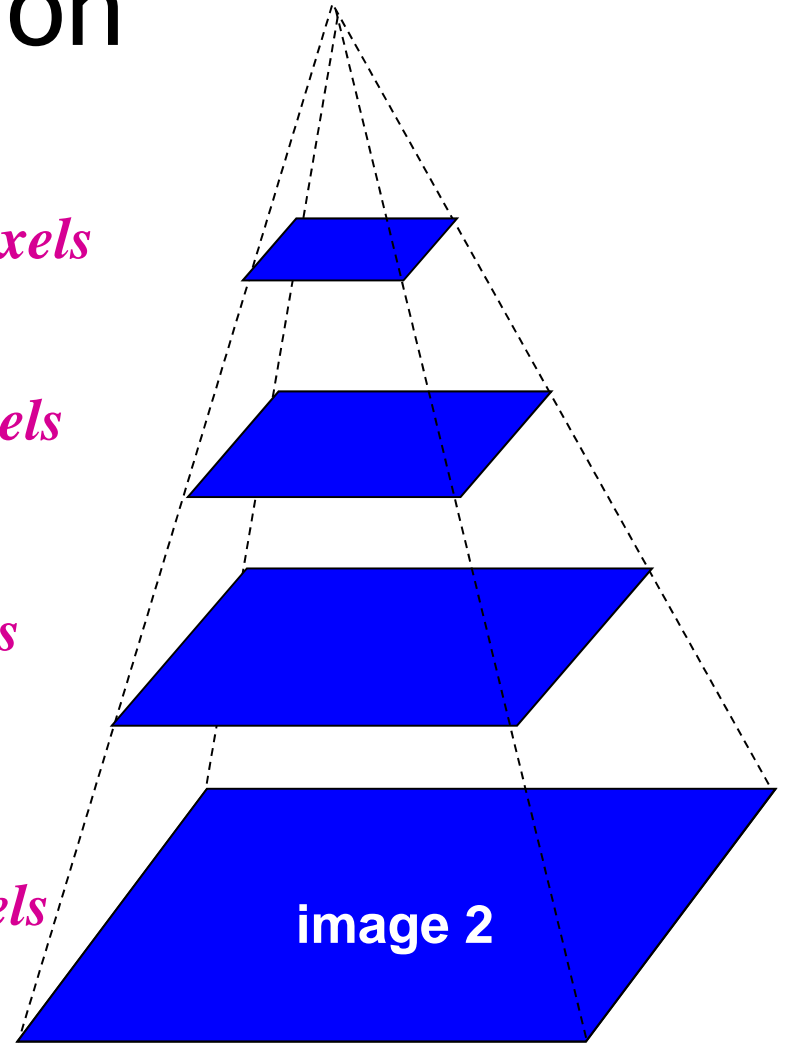
**Gaussian pyramid of image 1**

*$u=1.25$  pixels*

*$u=2.5$  pixels*

*$u=5$  pixels*

*$u=10$  pixels*



**Gaussian pyramid of image 2**



# Errors in Lucas-Kanade

- The motion is large
  - Possible Fix: Keypoint matching
- A point does not move like its neighbors
  - Possible Fix: Region-based matching
- Brightness constancy does not hold
  - Possible Fix: Gradient constancy

# State-of-the-art optical flow

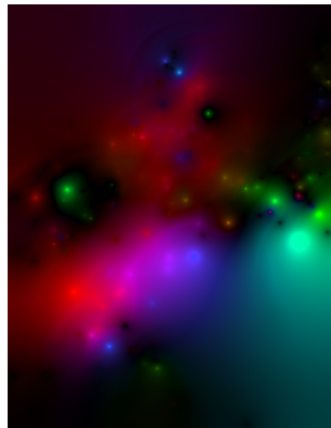
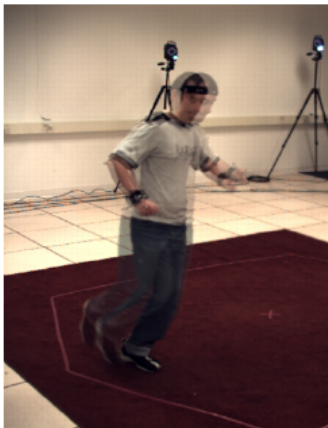
Start with something similar to Lucas-Kanade

+ gradient constancy

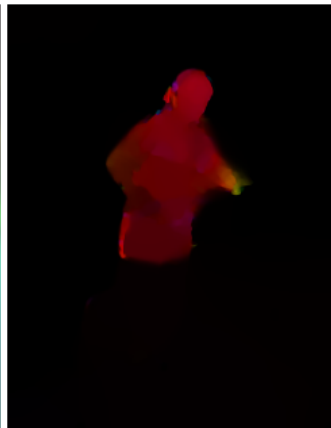
+ energy minimization with smoothing term

+ region matching

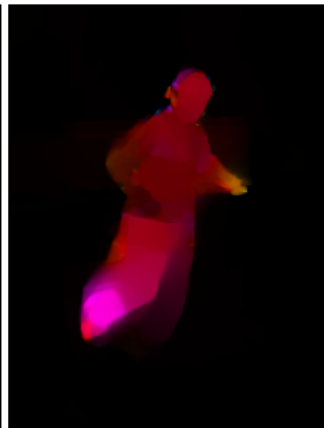
+ keypoint matching (long-range)



Region-based



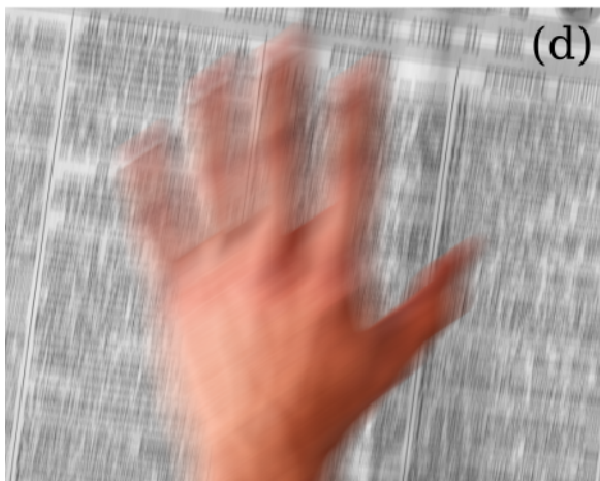
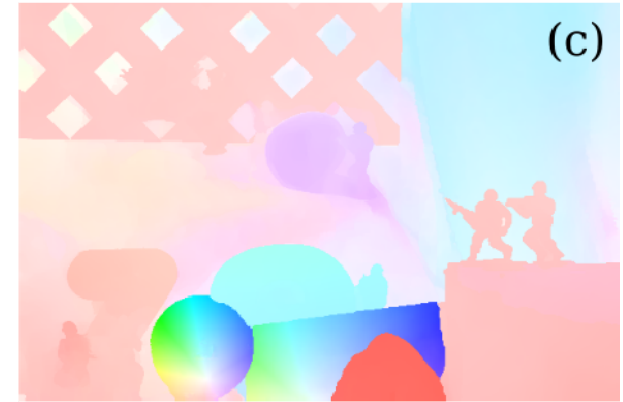
+Pixel-based



+Keypoint-based



# State-of-the-art optical flow



Two examples of applications of layered models. Top row: From a sequence of images (a), a layered model can extract the layer assignments (b) and compute highly accurate flow (c), especially at motion boundaries. Bottom row: Using a layered model, a motion blurred sequence (d) can be decomposed into foreground (e) and background (f), which can then be separately deblurred.

# Grouping and Segmentation

Based on slides by D. Hoiem

# Gestalt psychology or gestaltism

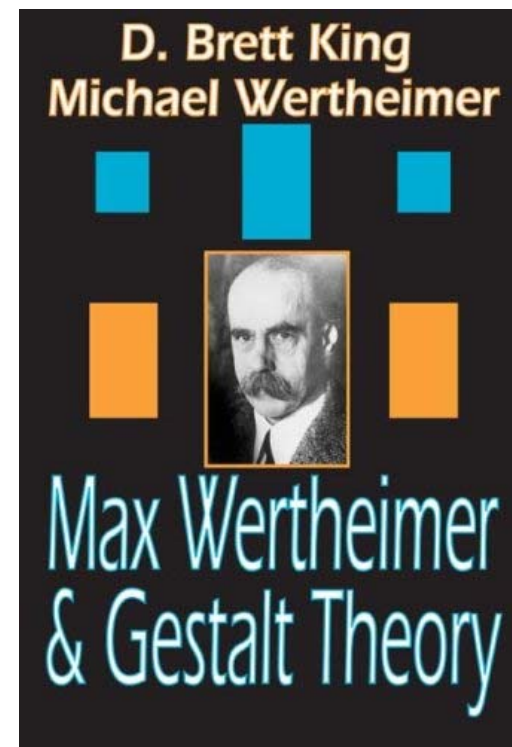
German: *Gestalt* - "form" or "whole"

Berlin School, early 20th century

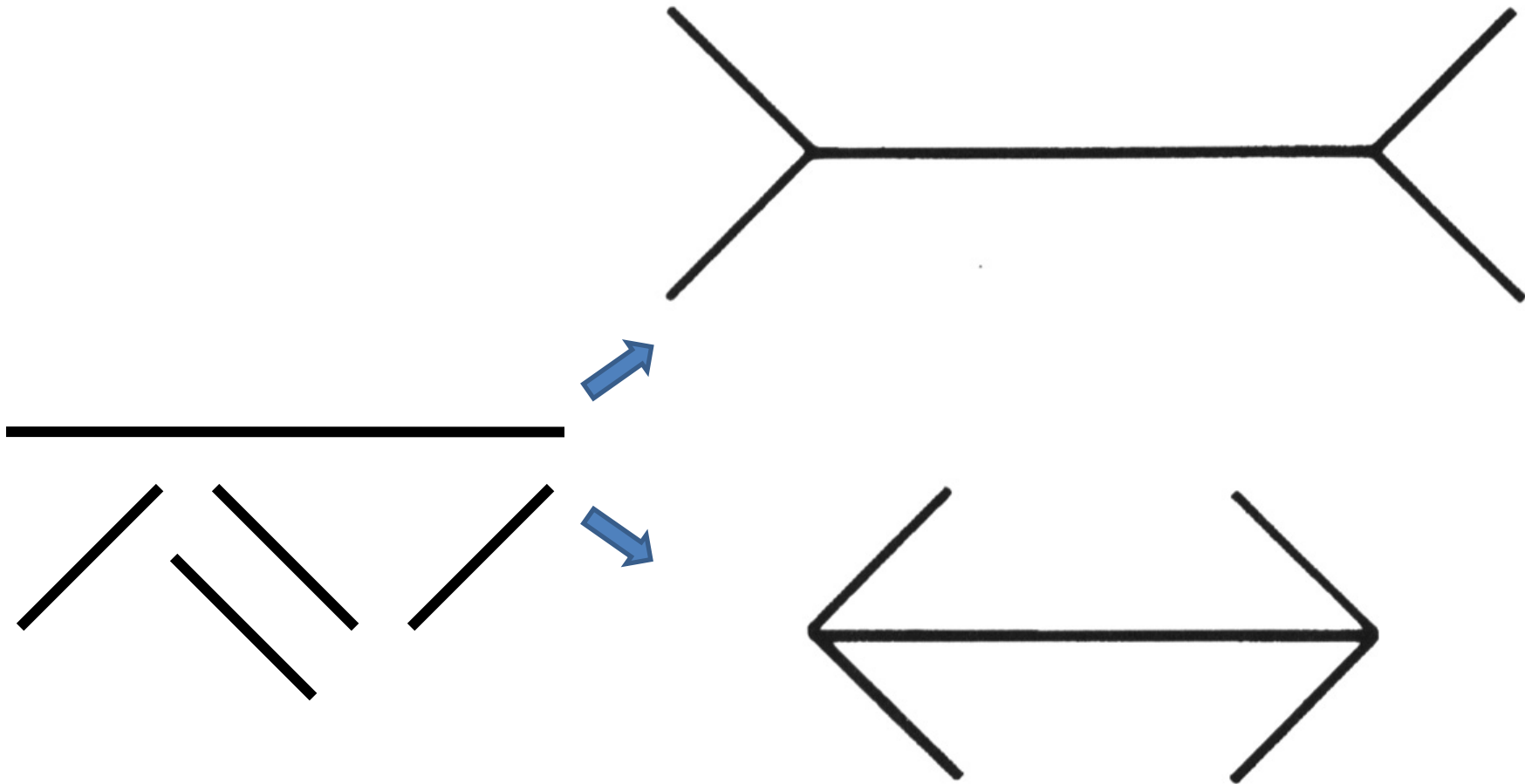
Kurt Koffka, Max Wertheimer, and Wolfgang Köhler

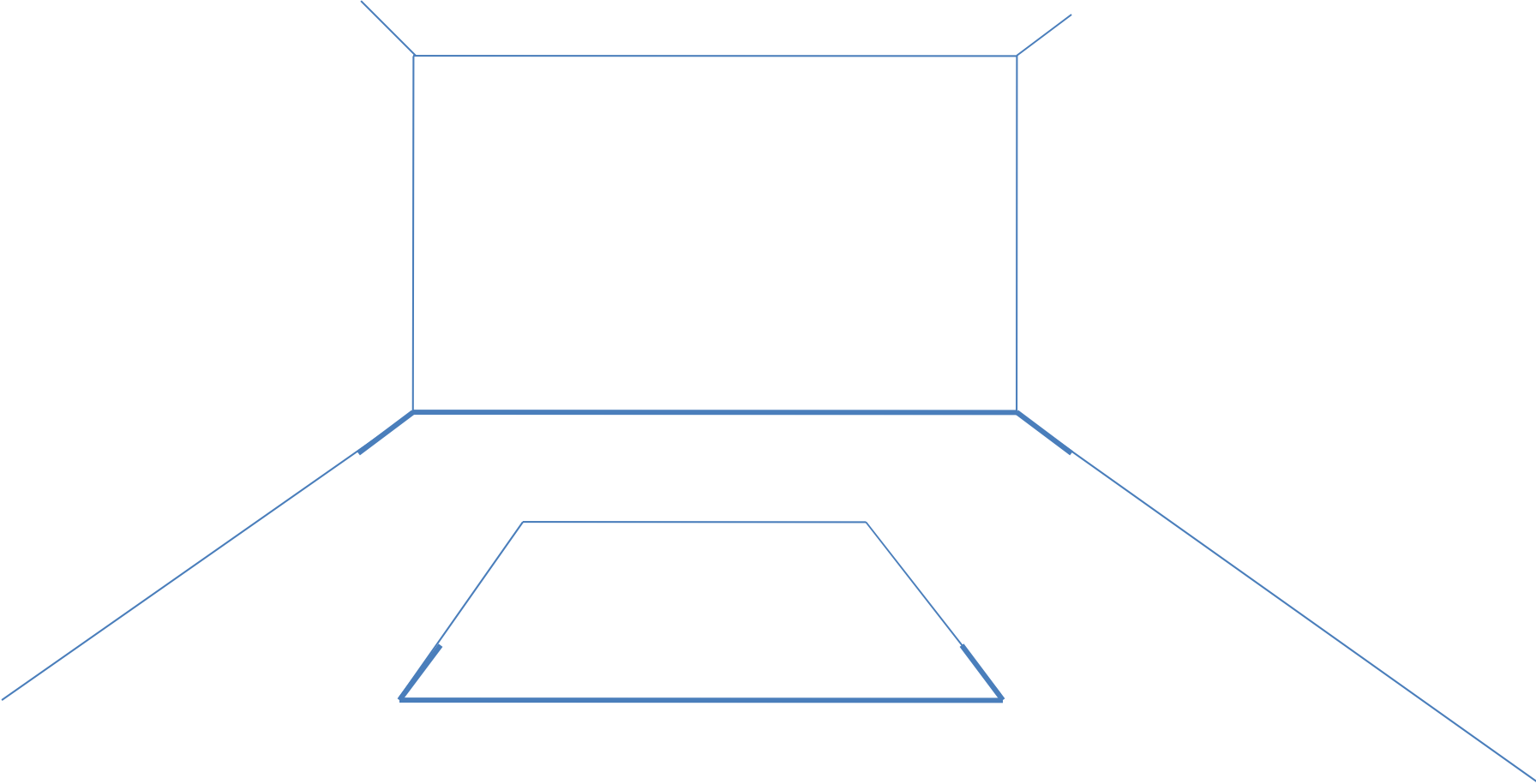
View of brain:

- whole is more than the sum of its parts
- holistic
- parallel
- analog
- self-organizing tendencies

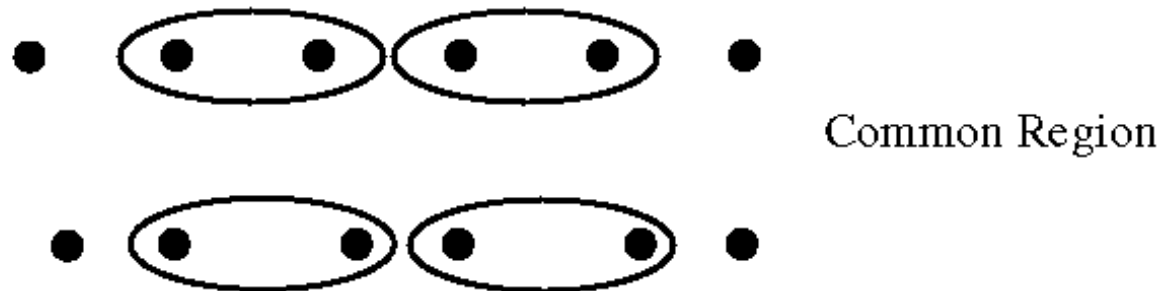
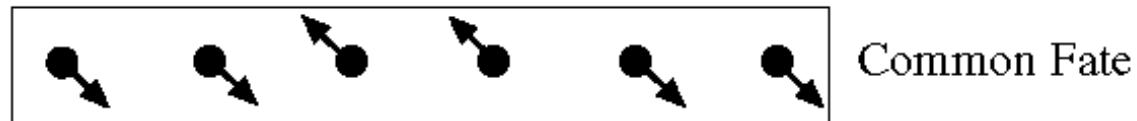
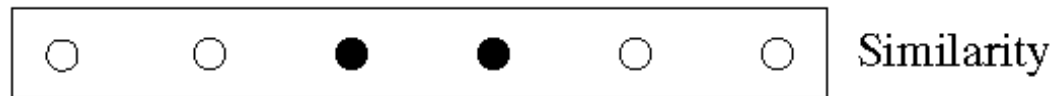


# Muller-Lyer Illusion



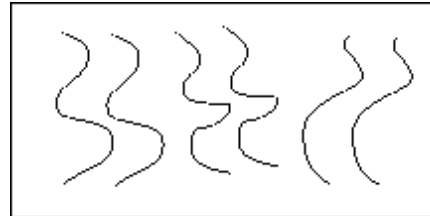


# Principles of perceptual organization

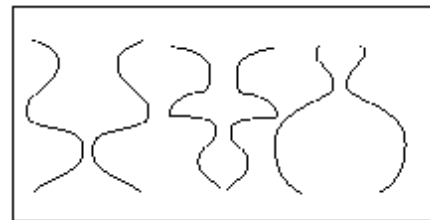




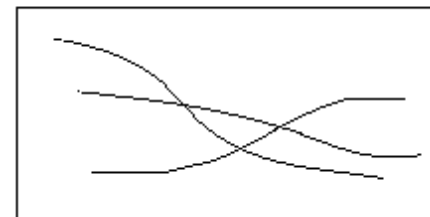
# Principles of perceptual organization



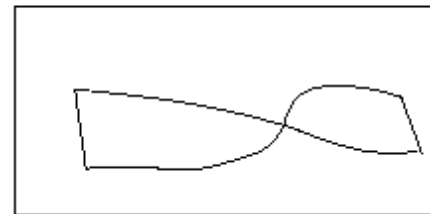
Parallelism



Symmetry

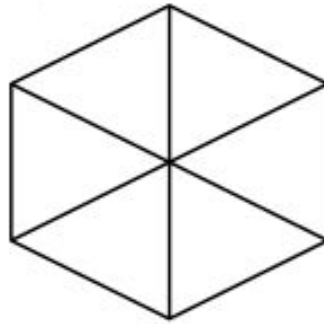


Continuity

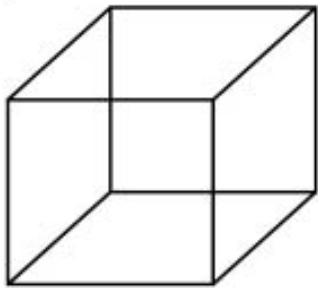


Closure

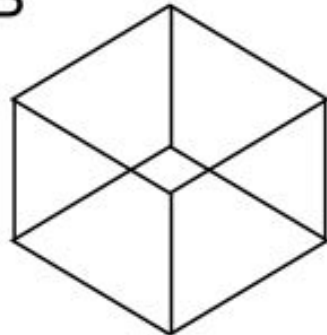
# Gestaltists do not believe in coincidence



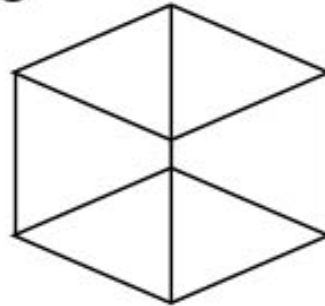
A



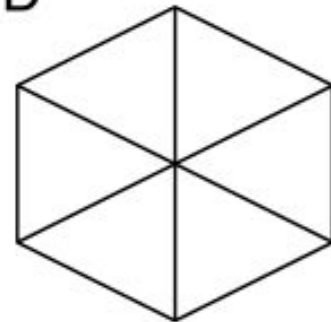
B



C



D

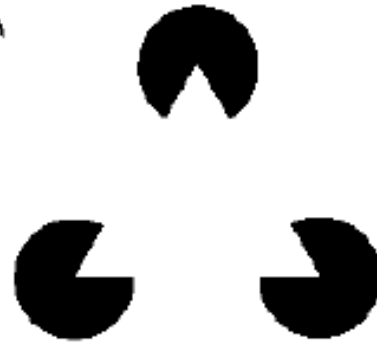


# Emergence



# Grouping by invisible completion

A



B



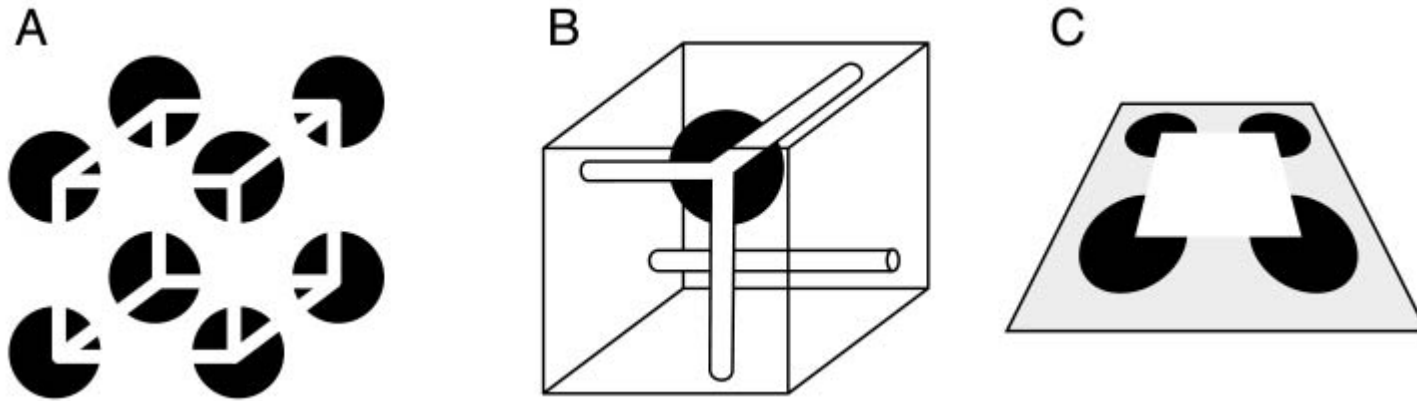
C



D



# Grouping involves global interpretation



# Gestalt cues

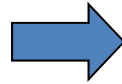
- Good intuition and basic principles for grouping
- Basis for many ideas in segmentation and occlusion reasoning
- Some (e.g., symmetry) are difficult to implement in practice

# Image segmentation

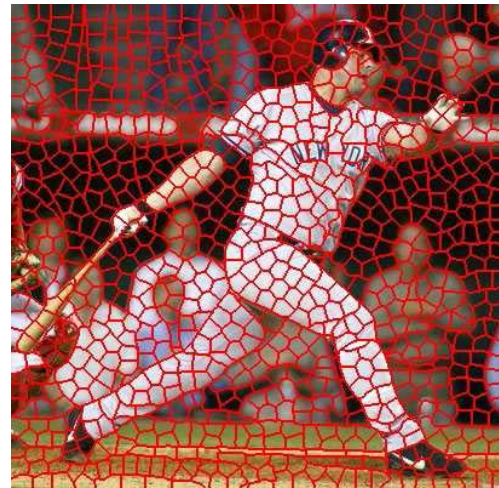
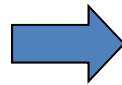
Goal: Group pixels into meaningful or perceptually similar regions



# Segmentation for efficiency: “superpixels”



[Felzenszwalb and Huttenlocher 2004]

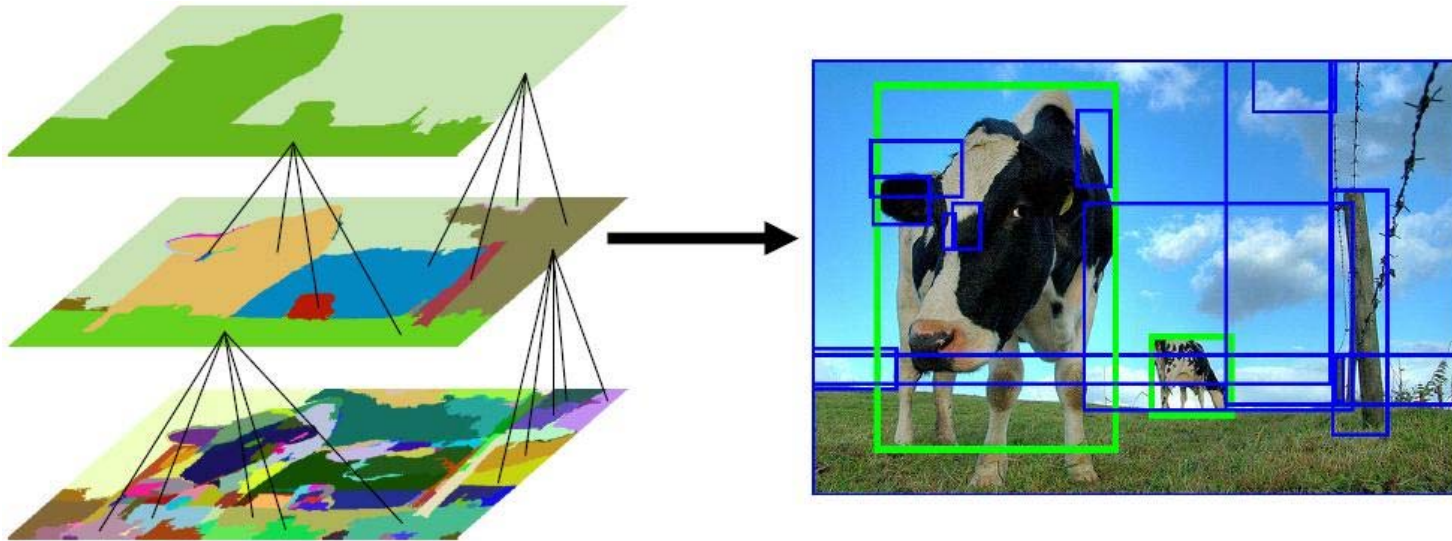




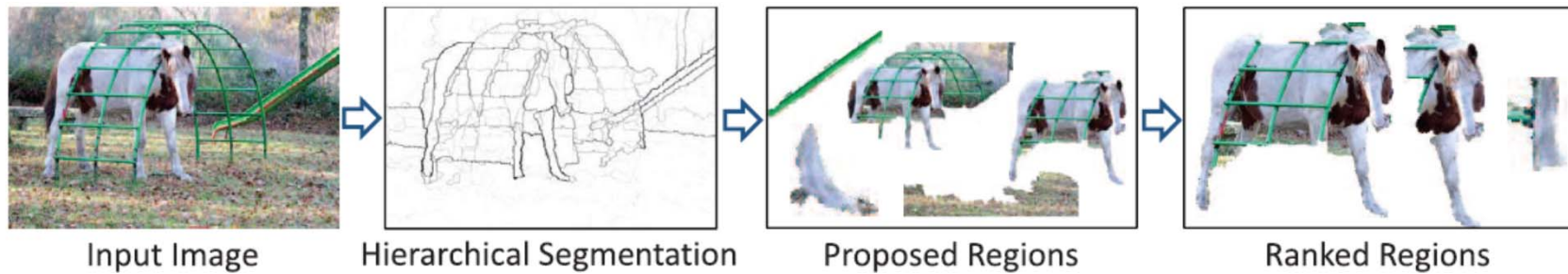
# Segmentation for feature support



# Segmentation for object proposals



“Selective Search” [Sande, Uijlings et al. ICCV 2011, IJCV 2013]



[Endres Hoiem ECCV 2010, IJCV 2014]

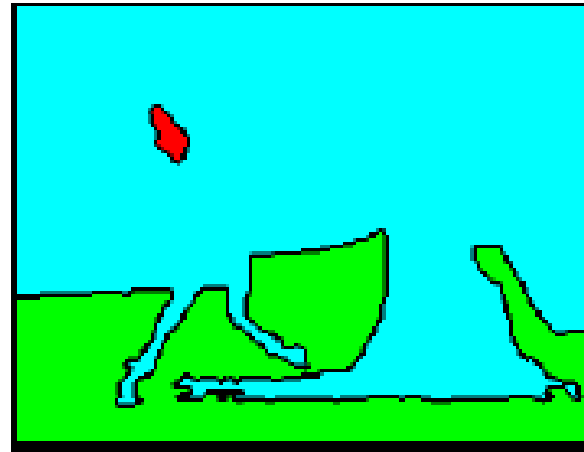
# Segmentation as the Endgoal



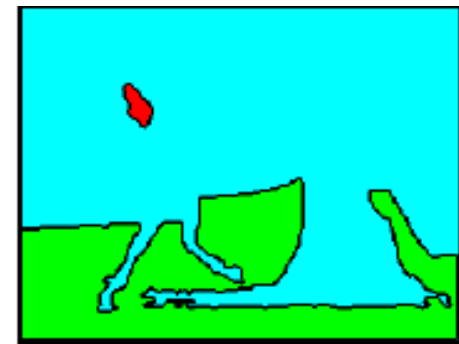
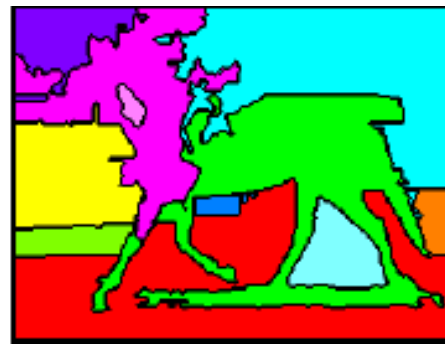
# Types of segmentations



Oversegmentation



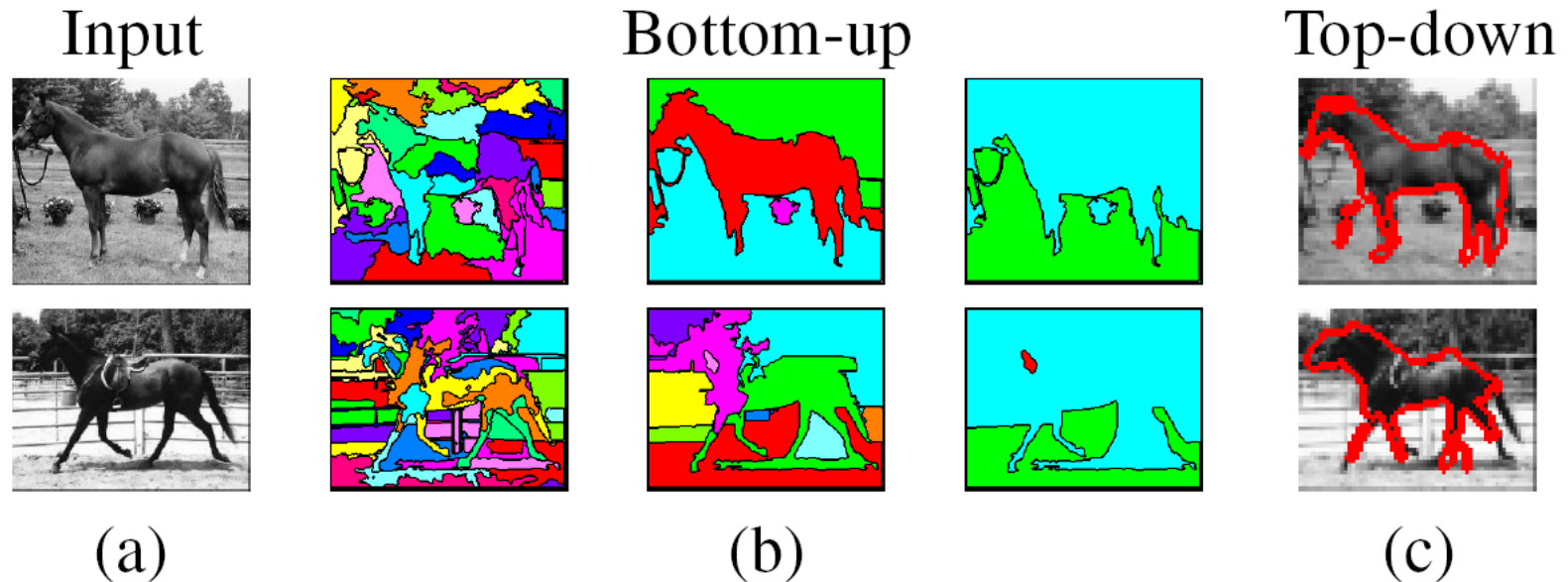
Undersegmentation



Multiple Segmentations

# Major processes for segmentation

- Bottom-up: group tokens with similar features
- Top-down: group tokens that likely belong to the same object



# Slide Credits

- This set of slides contains contributions kindly made available by the following authors
  - Derek Hoiem
  - Kristen Grauman
  - Michael Black
  - Takeo Kanade
  - Steve Lehar
  - Steve Seitz
  - Tomas Svoboda
  - Richard Szeliski
  - Carlo Tomasi