Lecture 17

Motion and Tracking

Administrative

A4 is out

- Due May 23th

A5 is out

- Due May 30th

Administrative

Recitation this friday

- Recognition review
- Jieyu Zhang

Today's agenda

- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

Reading: [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]
http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf

Today's agenda

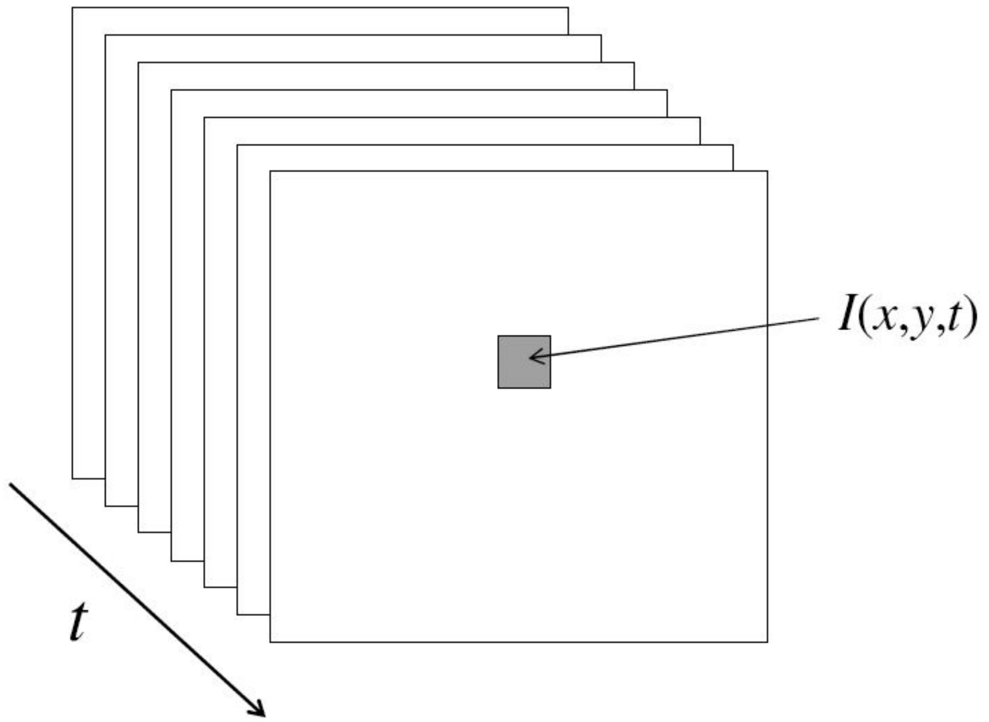
- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

Reading: [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]
http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf

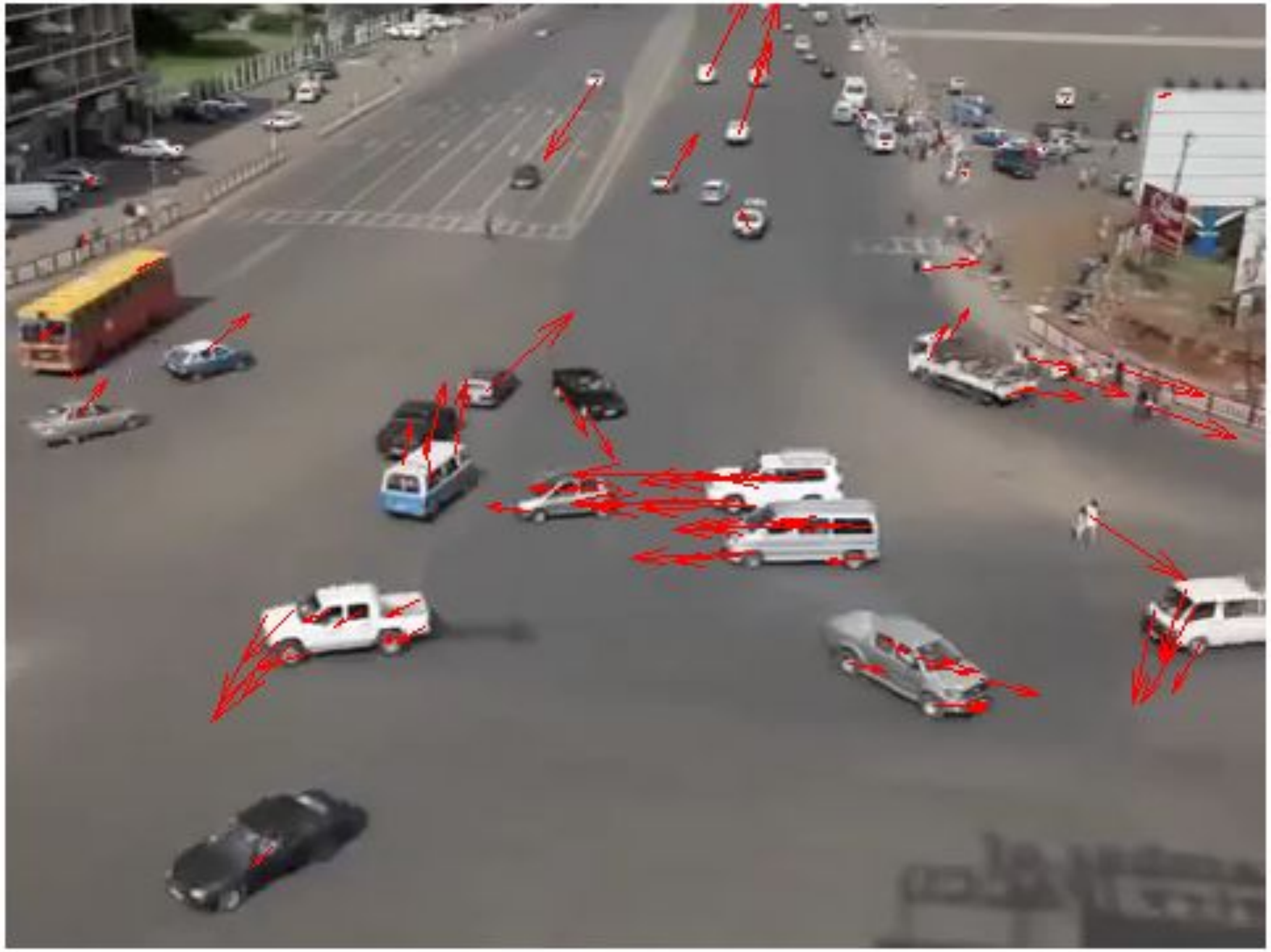
From images to videos

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



Why is motion useful?

Why is motion useful?



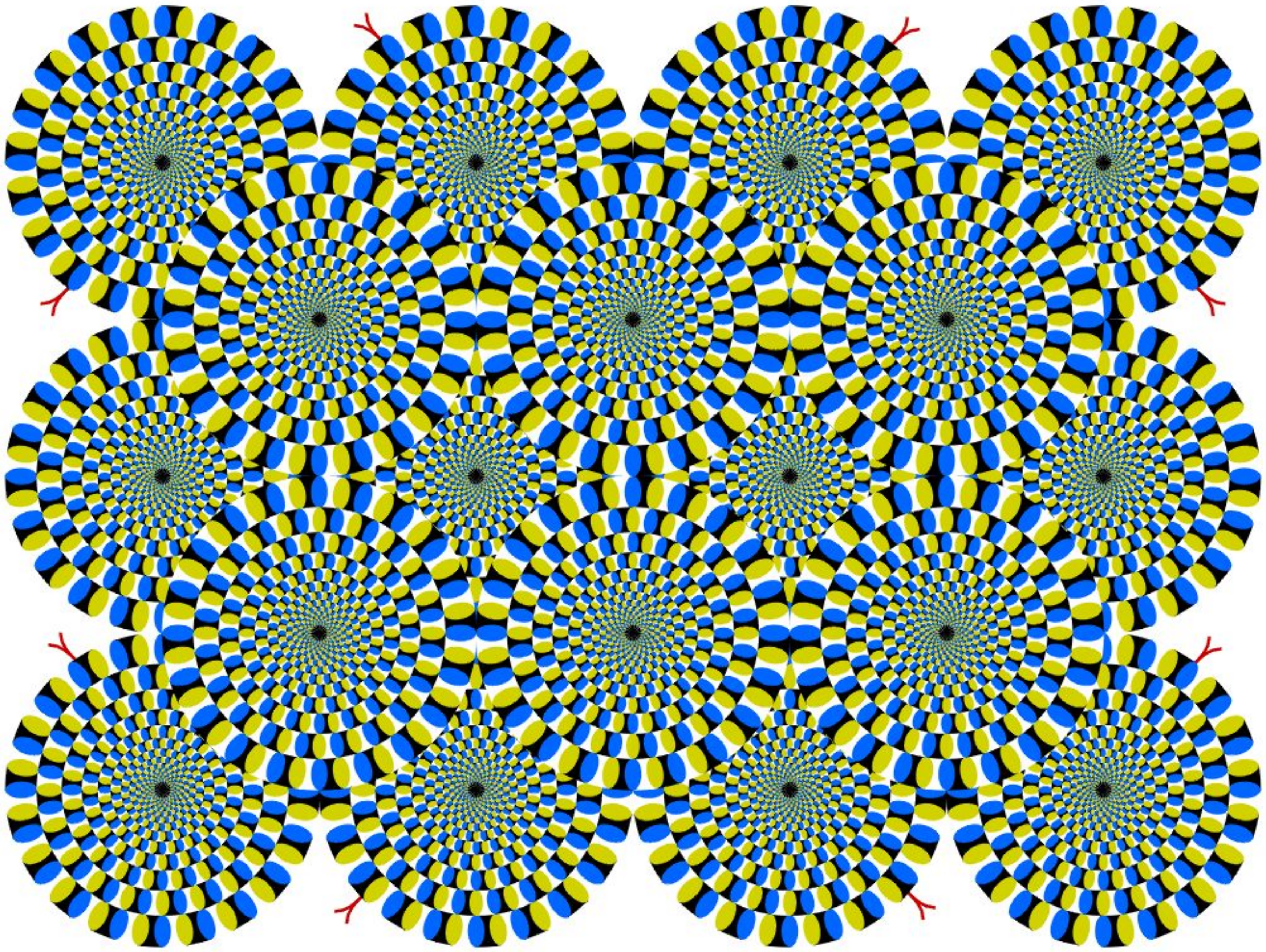
Optical flow

- Definition: optical flow is the *apparent* motion of brightness patterns in the image

- Note: apparent motion can be caused by lighting changes without any actual motion
  - Think of a uniform rotating sphere under fixed lighting (has motion but no optical flow)
  - vesus a stationary sphere under moving illumination (no motion but has optical flow)

GOAL: Recover image motion at each pixel from optical flow

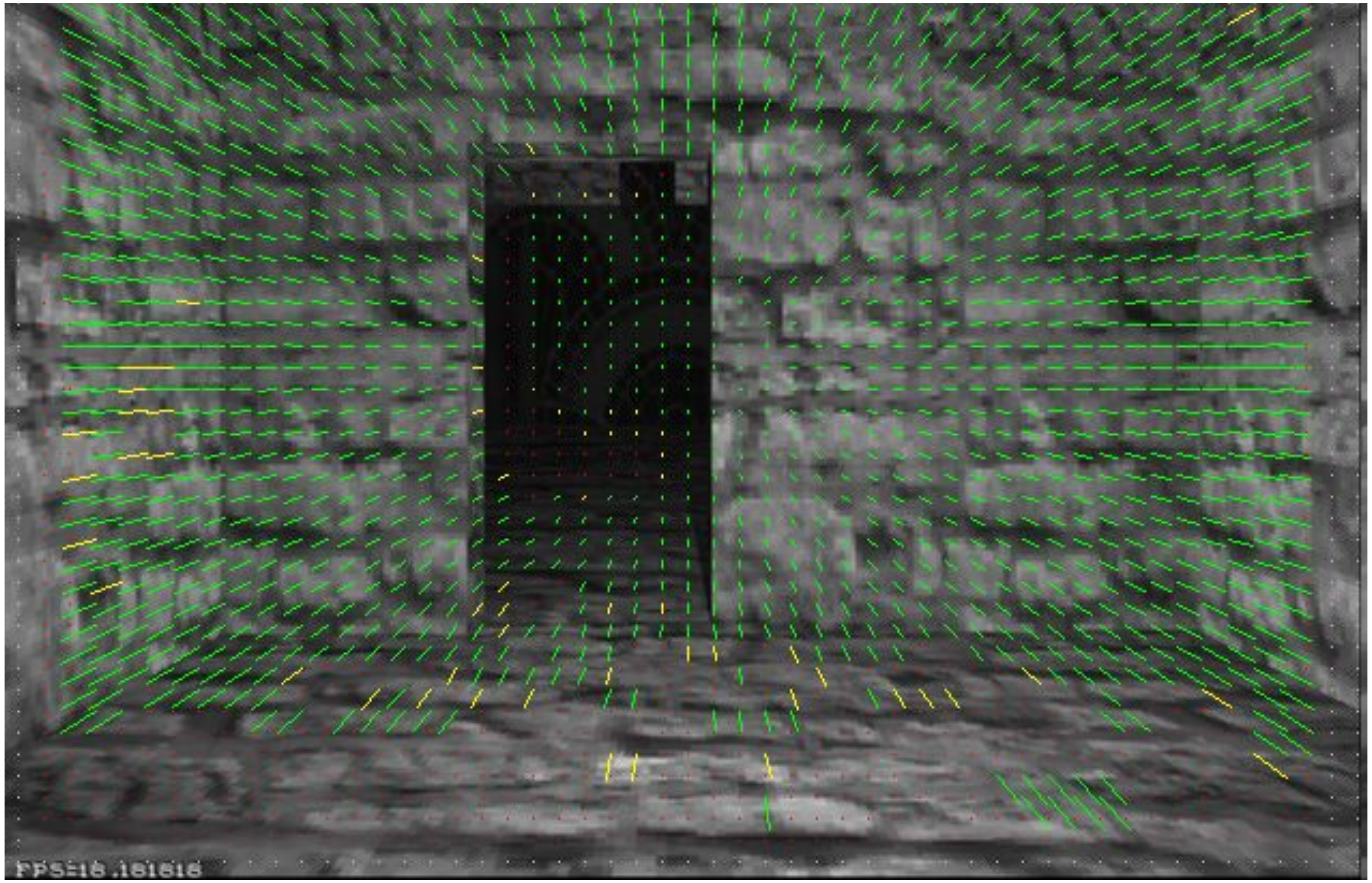
Optical flow without motion!



Optical flow

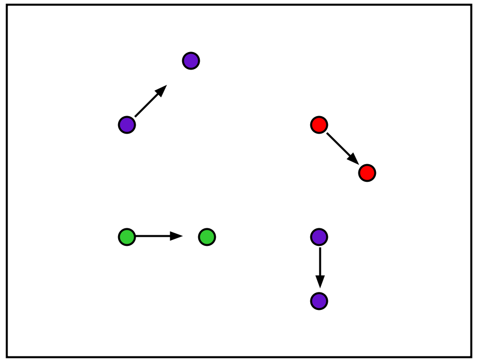
of an image gives us the apparent motion of every pixel

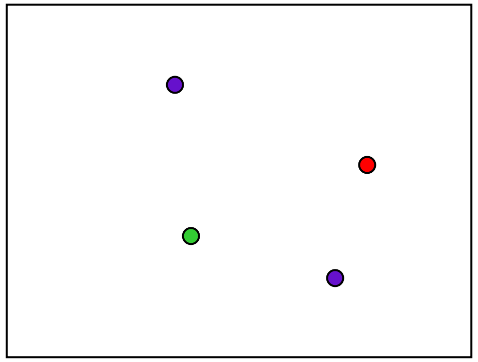
It is a function of the spatio-temporal image brightness variations



Picture courtesy of Selim Temizer - Learning and Intelligent Systems (LIS) Group, MIT

Formalizing optical flow


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


$I(x,y,t)$

- Given two subsequent frames,
- estimate the apparent motion field u(x,y), v(x,y) between them
- u(x, y) measuring the horizontal movement of the pixel at location (x, y).
- v(x, y) measures the vertical movement.

- Together, the pixel at (x, y, t-1) goes to (x+u, y+v, t)

3 assumptions when estimating optical flow

1. **small motions**: points do not move very far

2. **spatial coherence**: points move like their neighbors

3. **brightness constancy**: projection of the same point looks the same in every frame

Key Assumptions: small motions

The **small motions assumption**:
Between consecutive frames the change in pixel locations is small

* Slide from Michael Black, CS143, 2003

Key Assumptions: spatial coherence

The **spatial coherence assumption**: Neighboring pixels typically move together because they belong to the same rigid object.

* Slide from Michael Black, CS143 2003

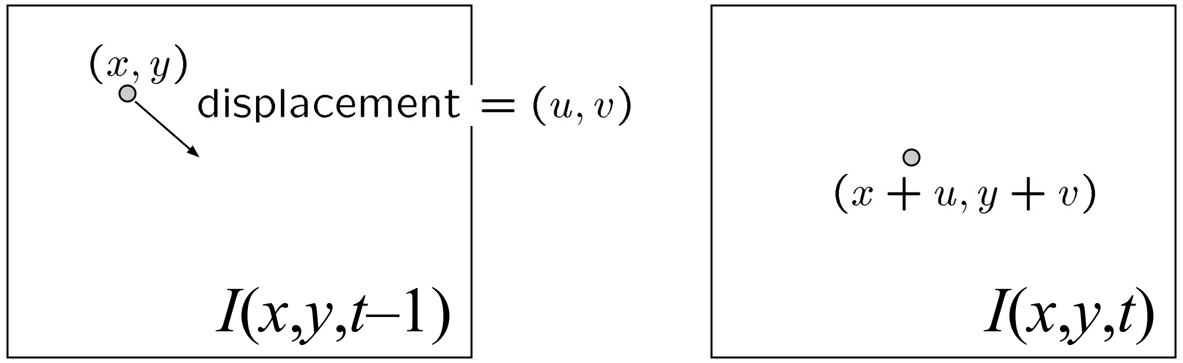
Key Assumptions: brightness Constancy

The **brightness constancy assumption**: Average brightness of pixels in a patch stays the same across consecutive frames, although their location might change

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

* Slide from Michael Black, CS143 2003

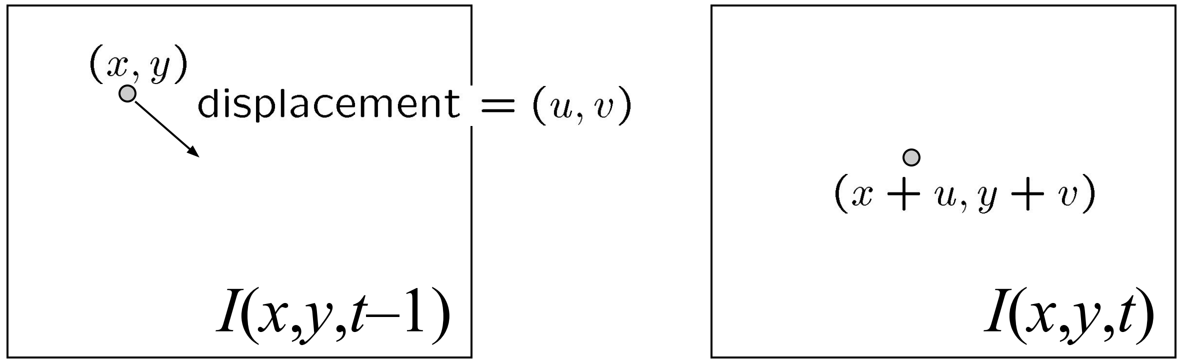
The brightness constancy constraint



- Brightness Constancy Equation:

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

The brightness constancy constraint



- Brightness Constancy Equation:

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

Linearizing the right side using Taylor expansion:

$$I(x+u,y+v,t) \approx I(x,y,t-1) + \underbrace{I_x}_{\text{Image derivative along x}} \cdot u(x,y) + I_y \cdot v(x,y) + \underbrace{I_t}_{\text{Image derivative along t}}$$

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

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

Filters used to find the derivatives

$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$ first image

$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$ second image

I_x

$\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$ first image

$\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$ second image

I_y

$\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$ first image

$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ second image

I_t

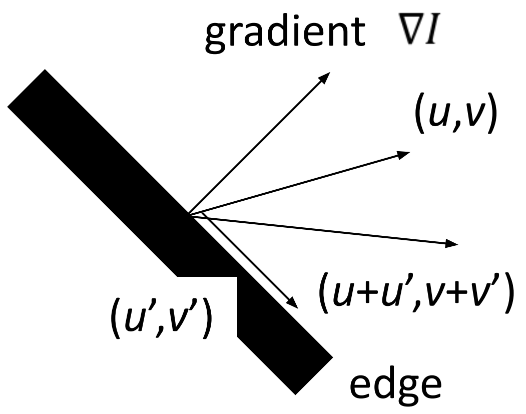
The brightness constancy constraint

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

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

- Q. How many equations and unknowns per pixel?
  - One equation, two unknowns (u,v)

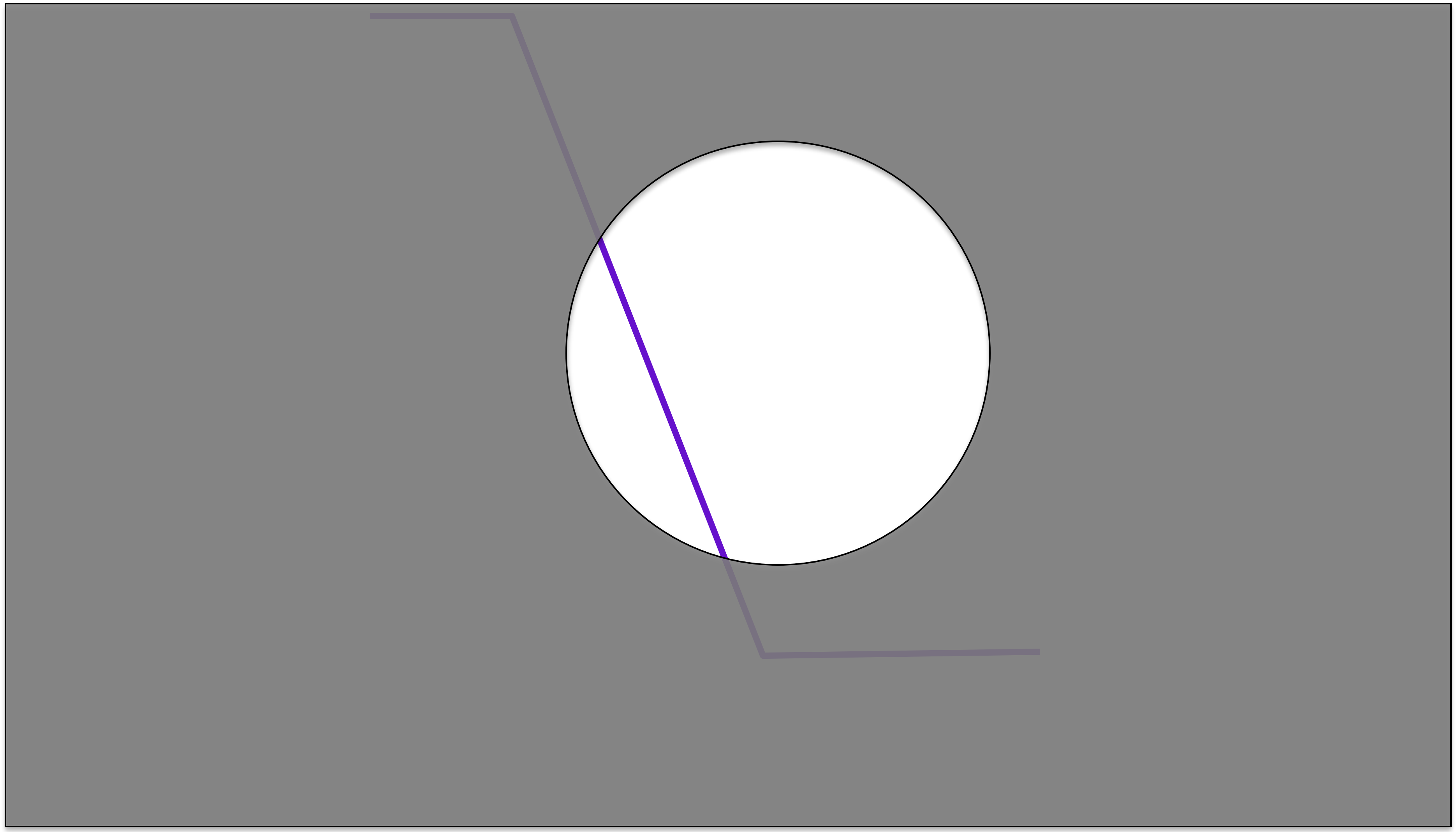
Problem: The component of the flow perpendicular to the gradient (i.e., parallel to the edge) cannot be measured



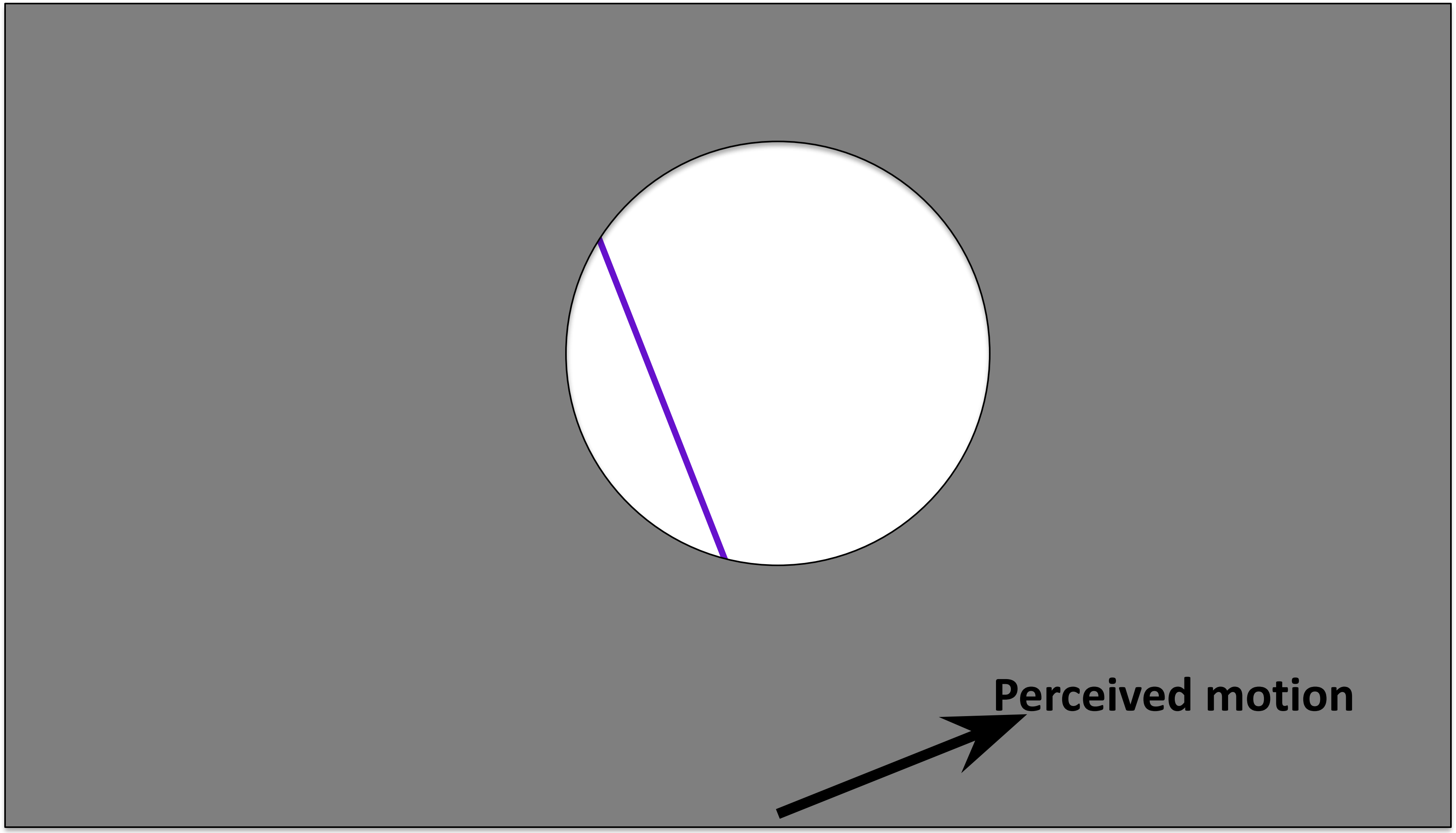
If (u, v) satisfies the equation, so does $(u+u', v+v')$ if

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

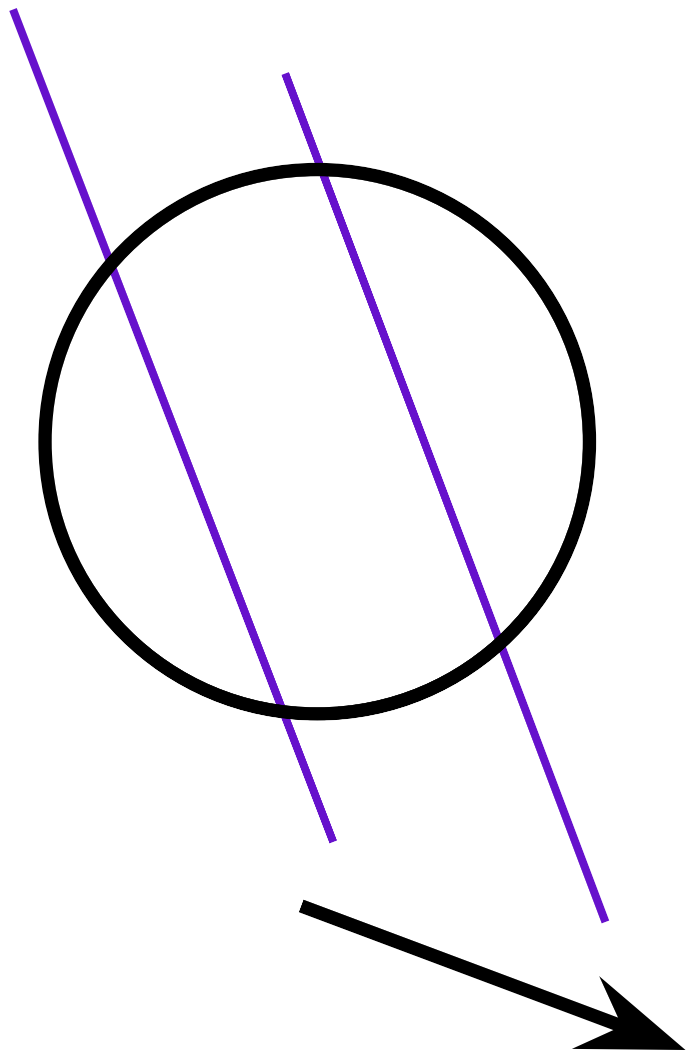
The aperture problem



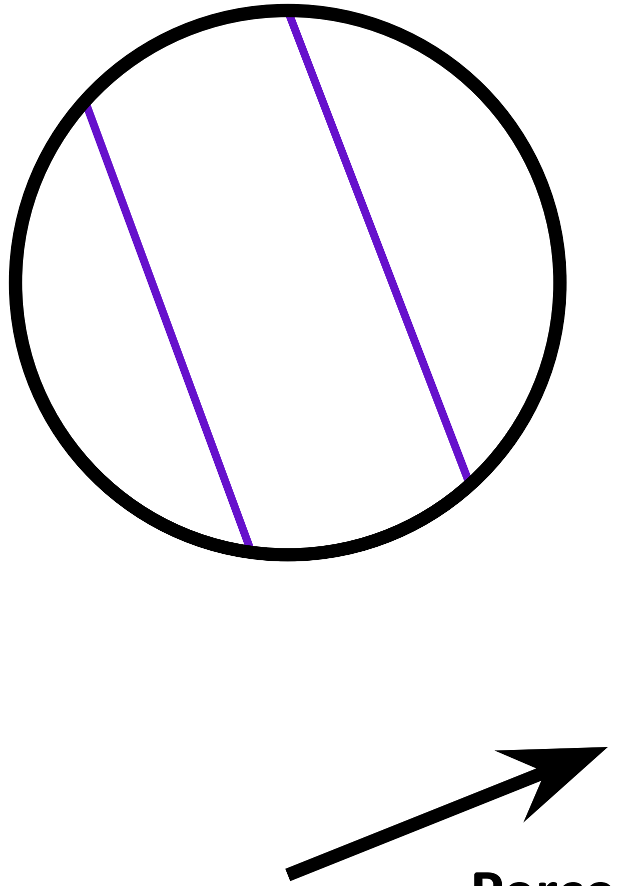
The aperture problem



The aperture problem



The aperture problem



The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Today’s agenda

- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

Reading: [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]
http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf

How to get more equations for a pixel?

- **Add in the 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

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

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

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Lucas-Kanade flow

- Overconstrained linear system:

$$\begin{bmatrix} I_x(\mathrm{p}_1) & I_y(\mathrm{p}_1) \\ I_x(\mathrm{p}_2) & I_y(\mathrm{p}_2) \\ \vdots & \vdots \\ I_x(\mathrm{p}_{25}) & I_y(\mathrm{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathrm{p}_1) \\ I_t(\mathrm{p}_2) \\ \vdots \\ I_t(\mathrm{p}_{25}) \end{bmatrix} \qquad \underset{25\text{x}2}{A} \; \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Lucas-Kanade flow

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix} \qquad \underset{25\text{x}2}{A} \ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Multiplying by A^T to solve for *d* gives us: $(A^T A)\ d = A^T b$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

The summations are over all pixels in the 5 x 5 window

Conditions for solving this Lucas-Kanade equation

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

When is This Solvable?

- **A^TA** should be invertible
- **A^TA** should not be too small, otherwise it is close to being non-invertible
  - eigenvalues λ_1 and λ_2 of **A^TA** should not be too small
- **A^TA** should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Q. Does this remind anything to you?

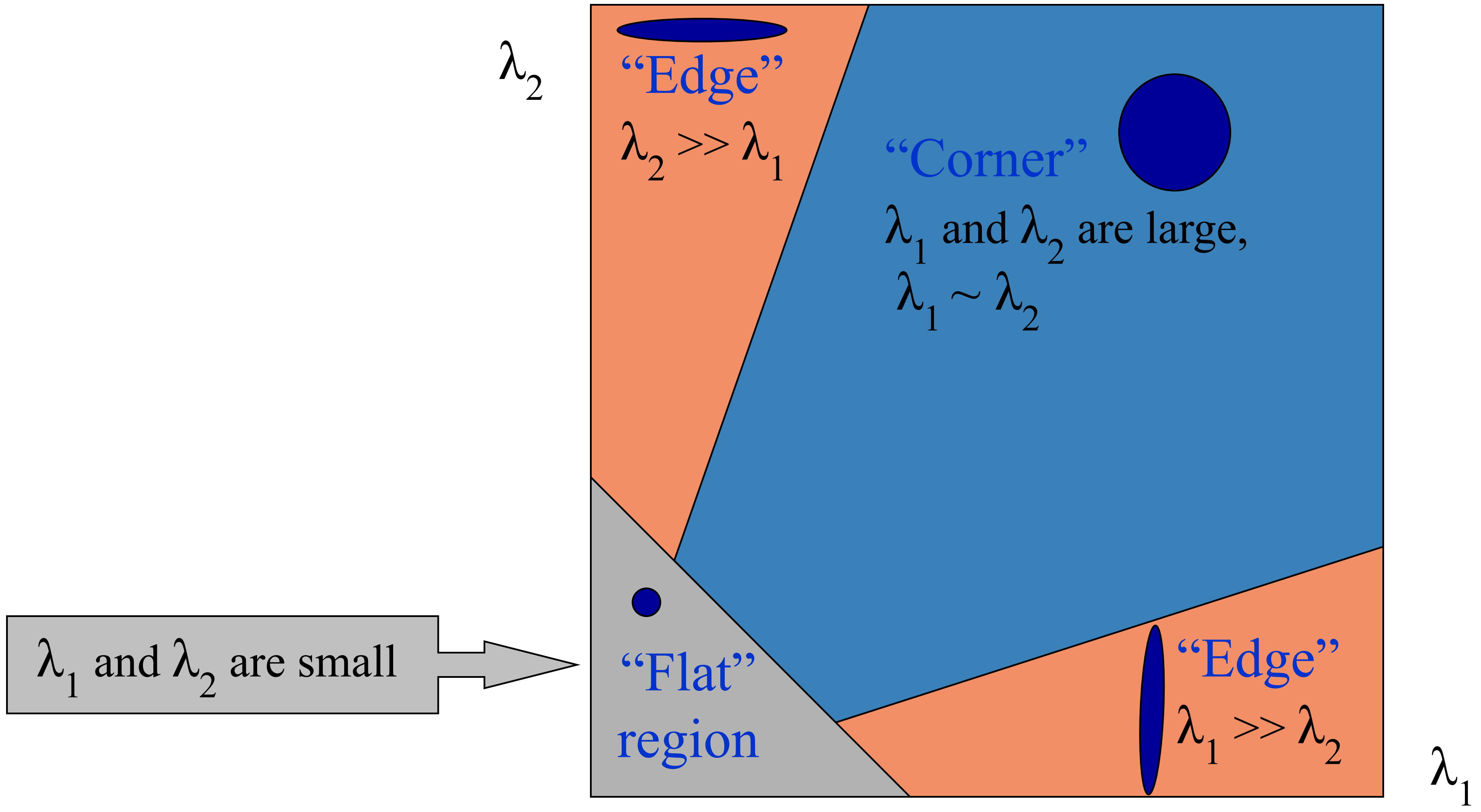
M = A^{T}A is the Harris corner detector!

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

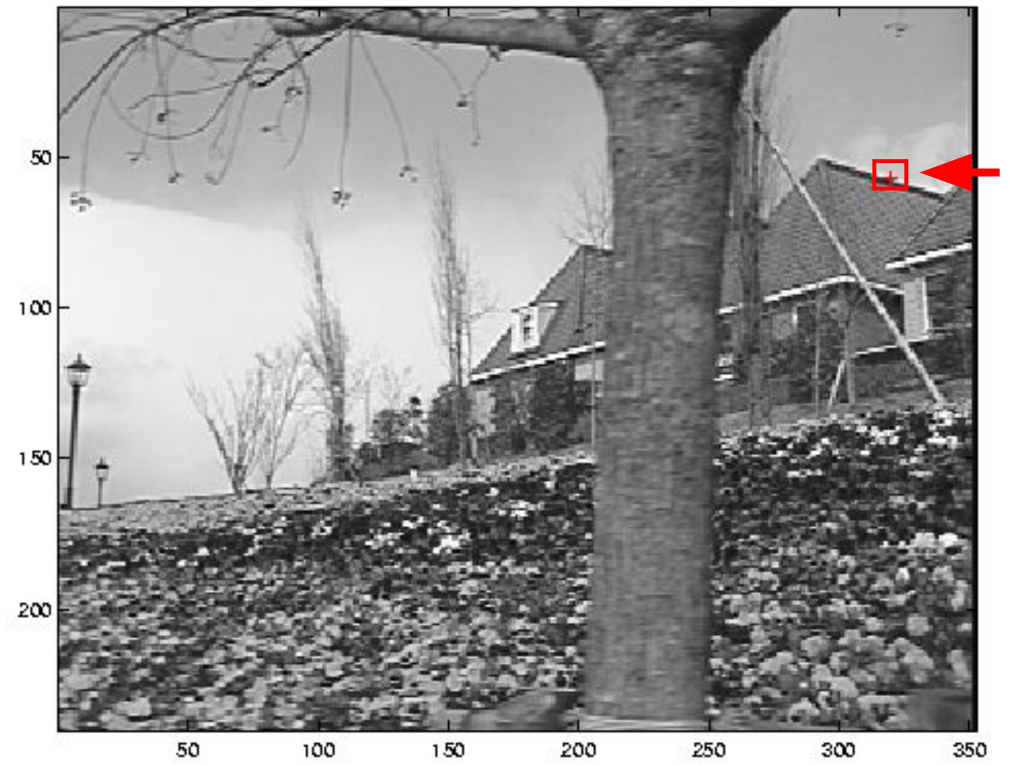
- Eigenvectors and eigenvalues of A^{T}A relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it

Interpreting the eigenvalues

Classification of image points using eigenvalues of the second moment matrix:



Edges are harder to track



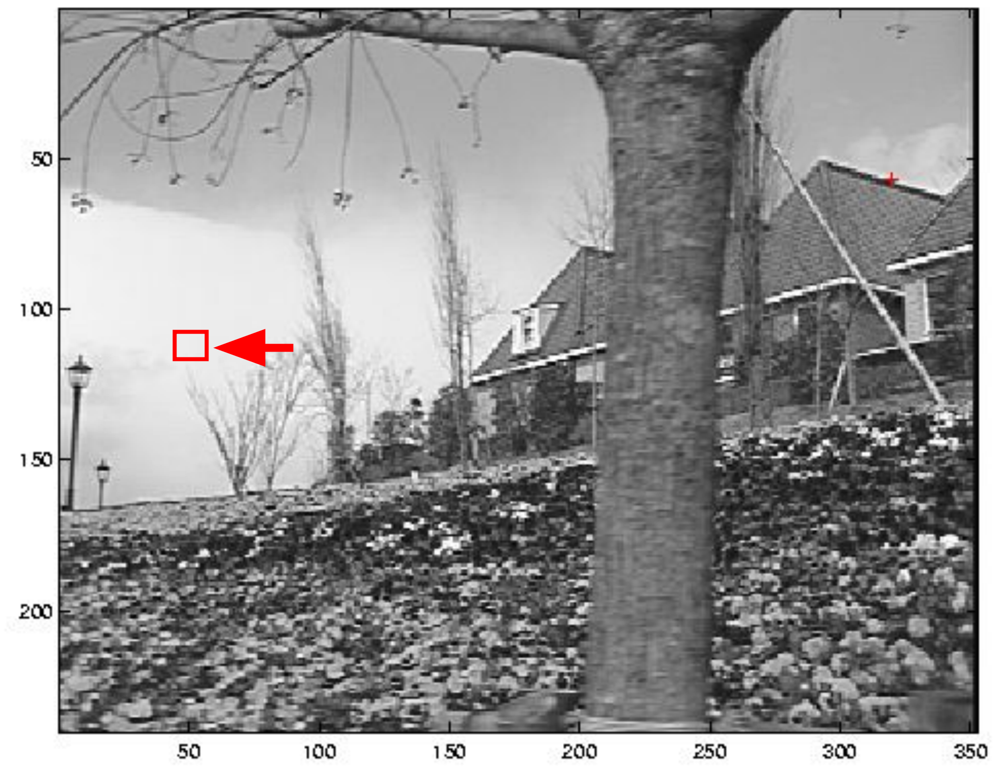
All the points on an edge look the same. It is hard to estimate where each point will move to.

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

– gradients very large or very small

– large λ_1, small λ_2

Low-texture region



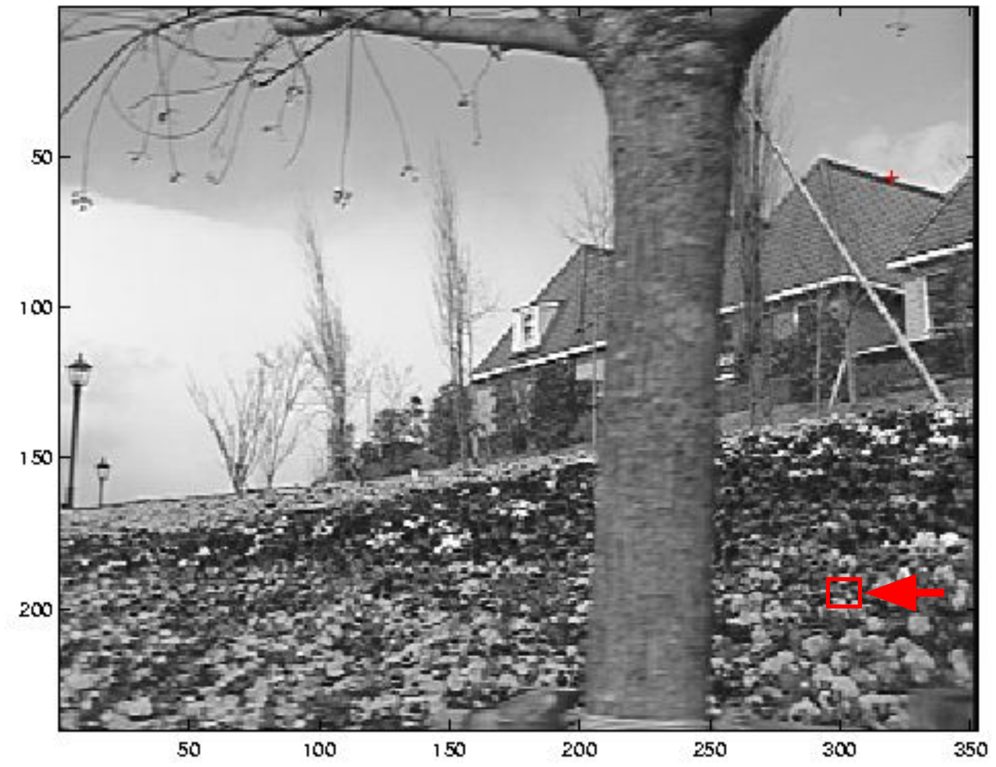
$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

Low-texture regions have small eigenvalues. The matrix is harder to invert and get accurate estimates of optical flow

High-texture region



These points are easier to estimate optical flow for.

This makes sense intuitively: You could say that corners and blobs (things that are easier to detect) are easier to track over time.

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

– gradients are different, large magnitudes

– large λ_1, large λ_2

Errors in Lucas-Kanade

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?

Improving accuracy

- Recall our small motion assumption

$$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

- This is not exact
  - To do better, we need to add higher order terms back in:

$$I_x \cdot u + I_y \cdot v + \text{higher order terms} + I_t \approx 0$$

- This is a *polynomial root finding* problem
  - Can solve using **Newton's method (which is out of scope for this class)**
  - Lukas-Kanade method does one iteration of Newton's method
    - Better results are obtained via more iterations

Iterative Lucas-Kanade Algorithm

1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp I(t-1) towards I(t) using the estimated flow field

   Calculate I(t) using the calculated optical flow
3. Repeat until convergence

When do the optical flow assumptions fail?

In other words, in what situations does the displacement of pixel patches not represent physical movement of points in space?

1. Well, television (movies) screens appear to contain objects in motion
   – Yet our TVs and monitors are actually stationary
2. Motion that doesn’t cause changes in pixels
   – e.g. A uniform rotating sphere. Nothing seems to move, yet it is rotating
3. Lighting changes can make things seem to move
   – for example, if a singular light source moves around a stationary sphere
4. Smaller motions might move in a direction opposite to motion
   – E.g. a cheetah’s muscles move opposite direction of motion.

Today’s agenda

- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

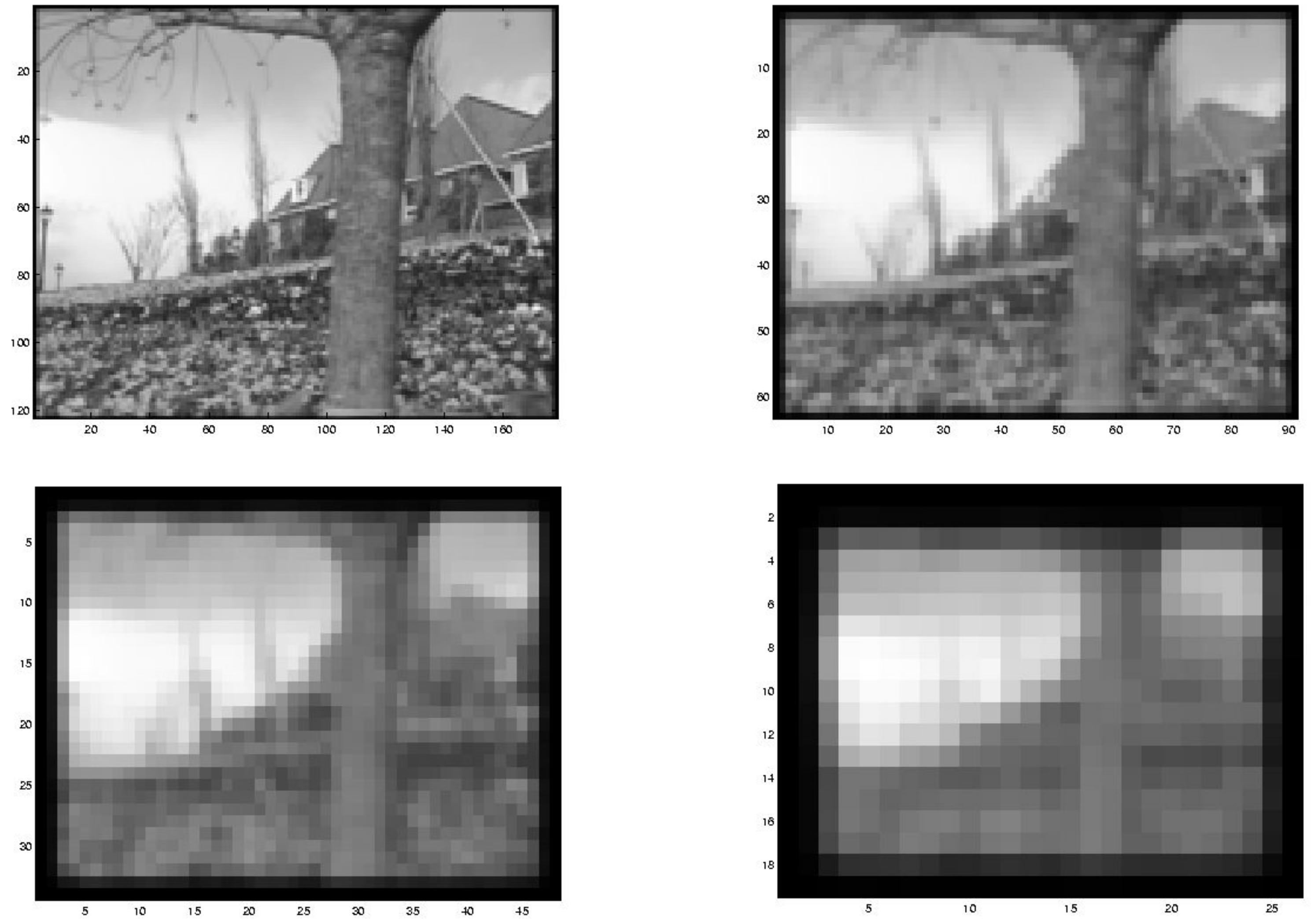
Key assumptions (Errors in Lucas-Kanade)

- **Small motion:** points do not move very far
- **Brightness constancy:** projection of the same point looks the same in every frame
- **Spatial coherence:** points move like their neighbors

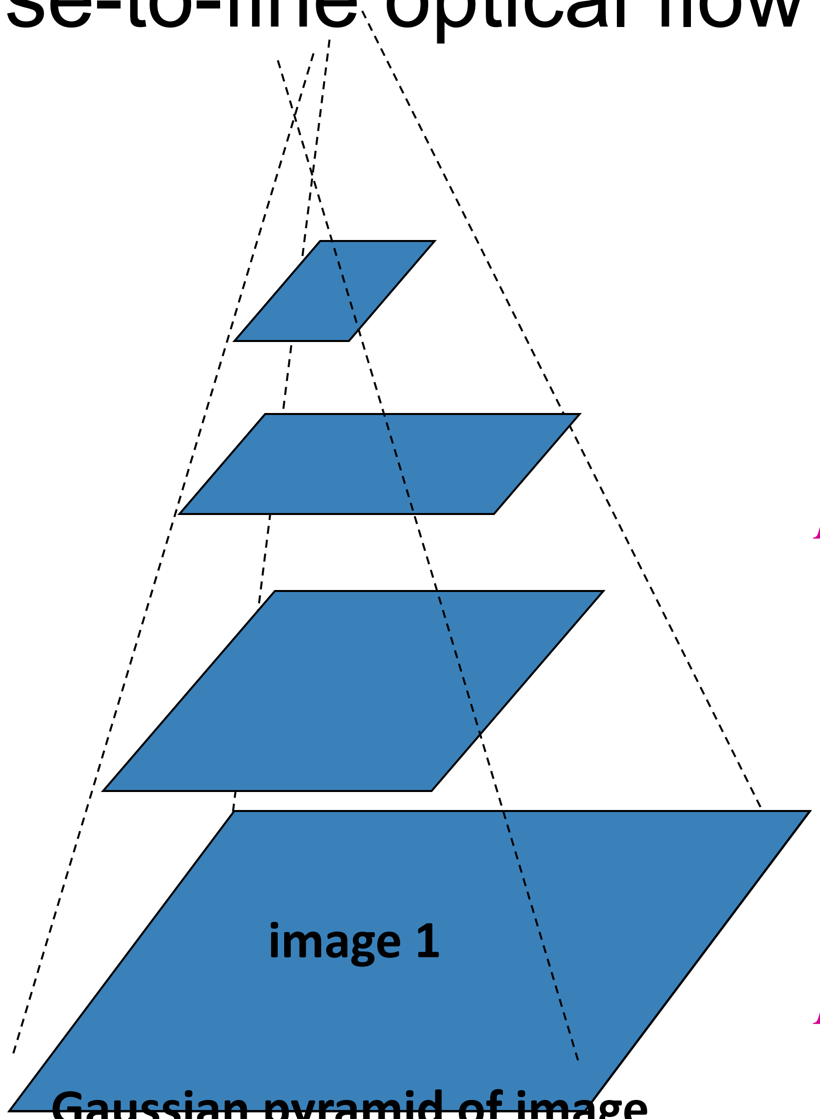
Revisiting the small motion assumption

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

Reduce the resolution so that assumption holds



Coarse-to-fine optical flow estimation

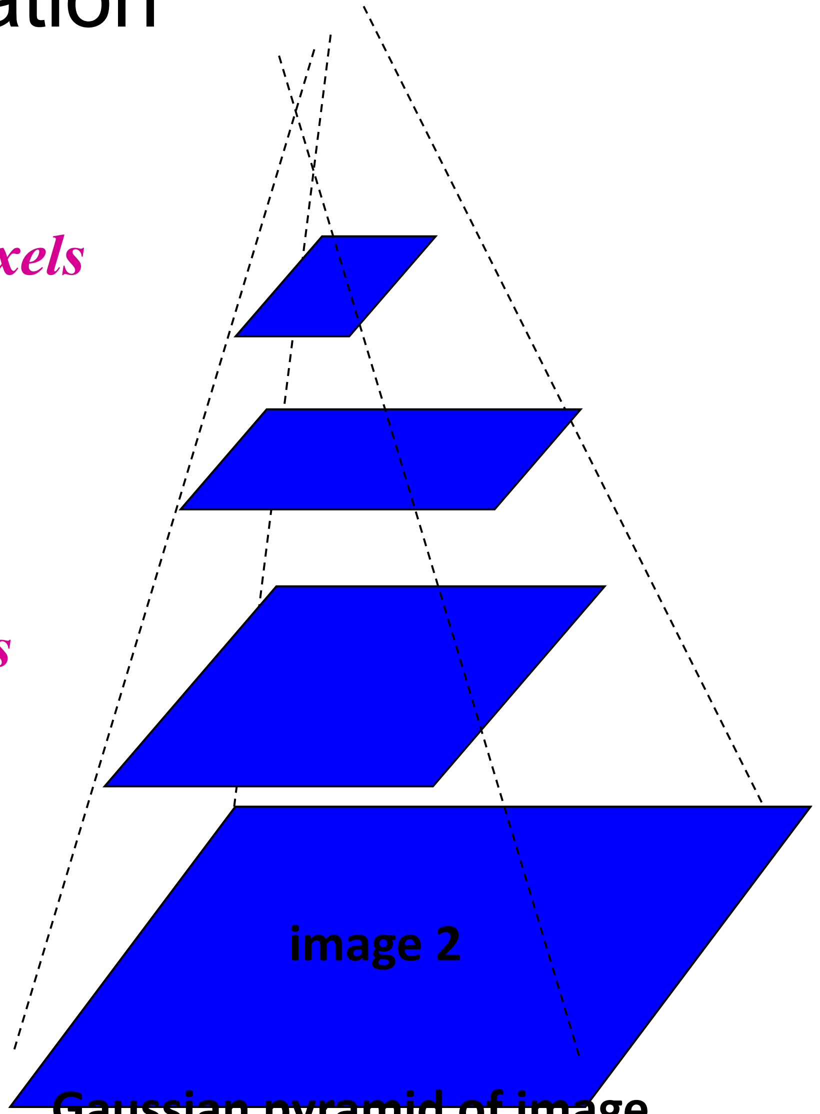


u=1.25 pixels

u=2.5 pixels

u=5 pixels

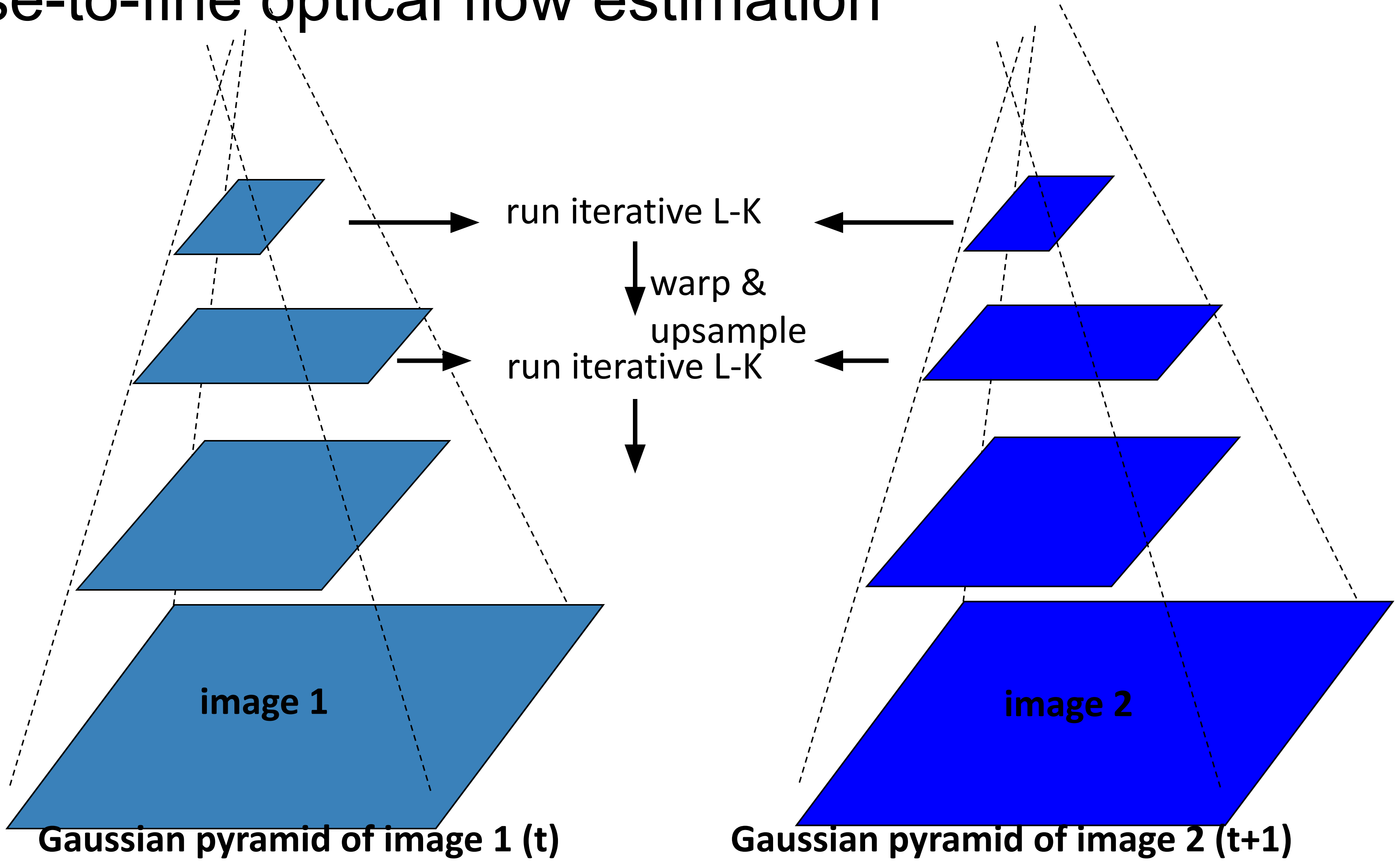
u=10 pixels



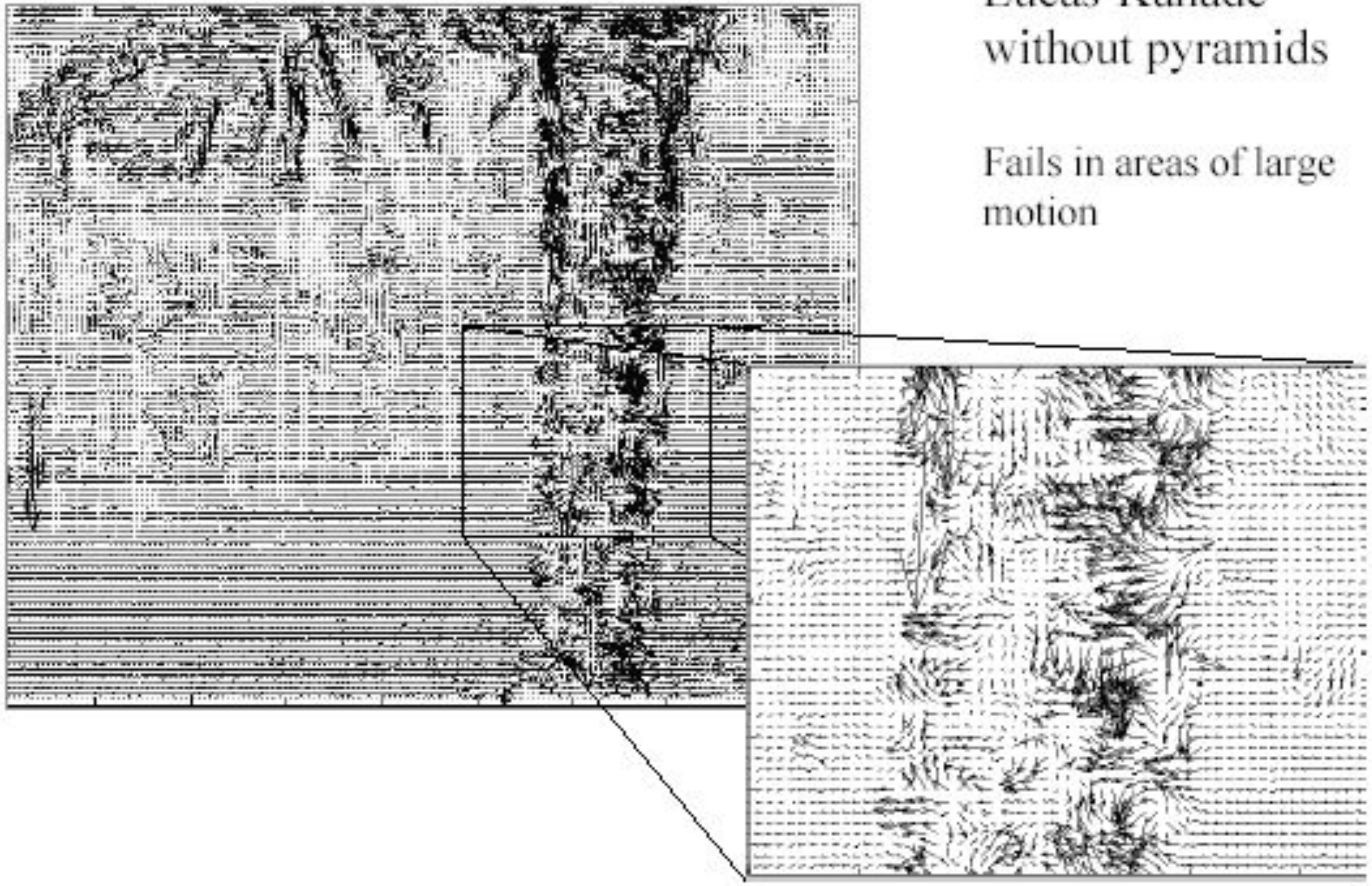
Gaussian pyramid of image 1

Gaussian pyramid of image 2

Coarse-to-fine optical flow estimation


run iterative L-K
warp &
upsample
run iterative L-K
image 1
image 2
Gaussian pyramid of image 1 (t)
Gaussian pyramid of image 2 (t+1)

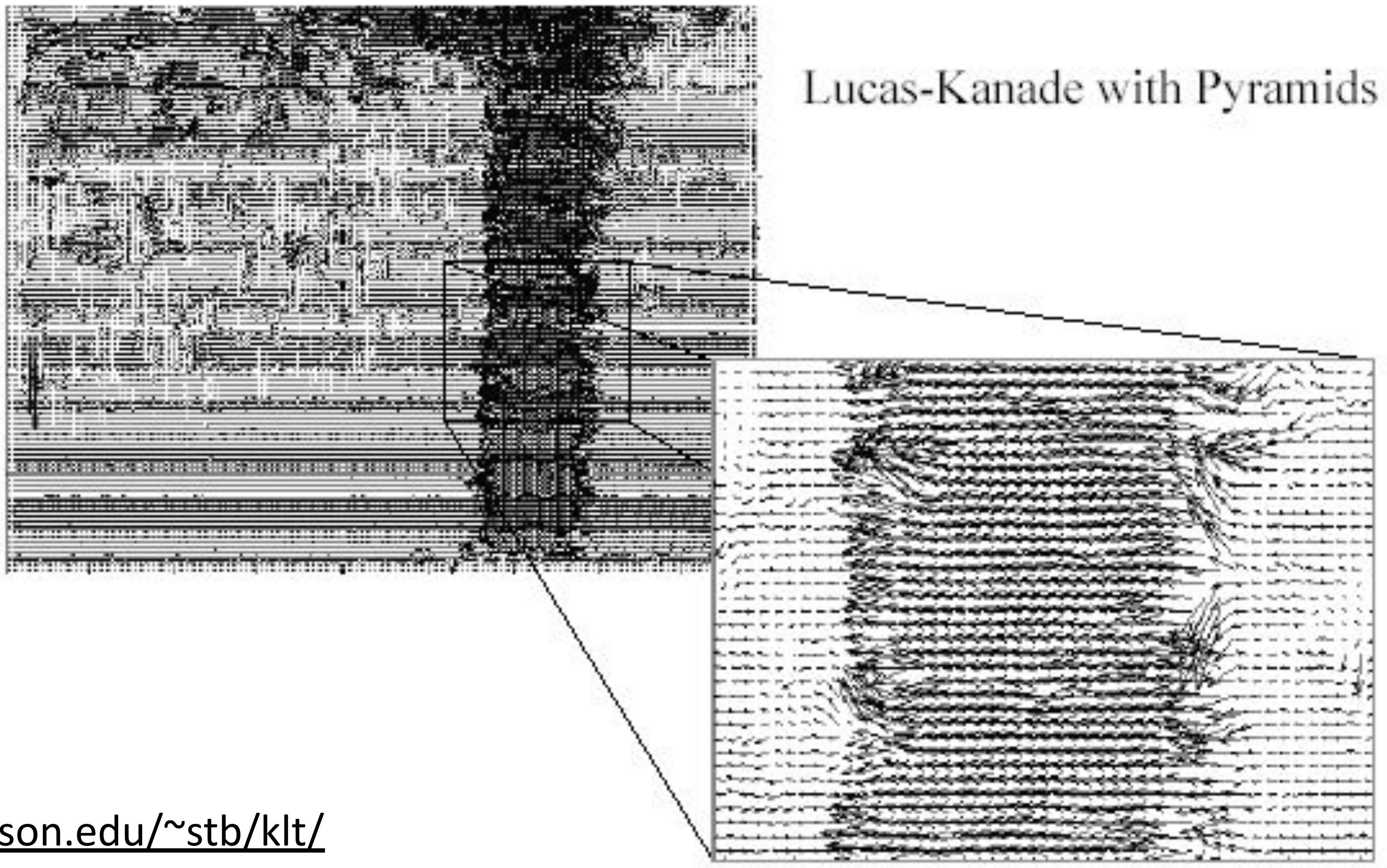
Optical Flow Results



Lucas-Kanade without pyramids

Fails in areas of large motion

Optical Flow Results



- http://www.ces.clemson.edu/~stb/klt/
- OpenCV

Today’s agenda

- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

Reading: [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]
http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf

Key assumptions (Errors in Lucas-Kanade)

- **Small motion:** points do not move very far
- **Brightness constancy:** projection of the same point looks the same in every frame
- **Spatial coherence:** points move like their neighbors

Horn-Schunk method for optical flow

- The flow is formulated as a global energy function which is should be minimized:

$$E = \iint \left[(I_x u + I_y v + I_t)^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2) \right] \mathrm{d}x\mathrm{d}y$$

Horn-Schunk method for optical flow

- The flow is formulated as a global energy function which is should be minimized:
- The first part of the function is the brightness constancy.

$$E = \iint \left[\boxed{(I_x u + I_y v + I_t)}^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2) \right] \mathrm{d}x \mathrm{d}y$$

Horn-Schunk method for optical flow

- The flow is formulated as a global energy function which is should be minimized:
- The second part is the smoothness constraint. It’s trying to make sure that the changes between pixels are small.

$$E = \iint \left[(I_x u + I_y v + I_t)^2 + \alpha^2 \left(\boxed{\|\nabla u\|^2 + \|\nabla v\|^2} \right) \right] \mathrm{d}x \mathrm{d}y$$

Horn-Schunk method for optical flow

- The flow is formulated as a global energy function which is should be minimized:
- ***α*** is a regularization constant. Larger values of ***α*** lead to smoother flows across time.

$$E = \iint \left[(I_x u + I_y v + I_t)^2 + \boxed{\alpha^2} (\|\nabla u\|^2 + \|\nabla v\|^2) \right] \mathrm{d}x \mathrm{d}y$$

Horn-Schunk method for optical flow

- The flow is formulated as a global energy function which is should be minimized:

$$E = \iint \left[(I_x u + I_y v + I_t)^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2) \right] \mathrm{d}x\mathrm{d}y$$

- This minimization can be solved by taking the derivative with respect to u and v, we get the following 2 equations:

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta u = 0$$
$$I_y(I_x u + I_y v + I_t) - \alpha^2 \Delta v = 0$$

Horn-Schunk method for optical flow

- By taking the derivative with respect to u and v, we get the following 2 equations:

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta u = 0$$
$$I_y(I_x u + I_y v + I_t) - \alpha^2 \Delta v = 0$$

- Where $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is called the Lagrange operator. It is hard to calculate. So we estimate it using $\Delta u(x,y) = \overline{u}(x,y) - u(x,y)$
  Intuition: Lagrange is the second derivative. The estimation measures the deviation from the average change.

- where $\overline{u}(x,y)$ is the weighted average of u measured at (x,y) over its neighborhood of 5 x 5 pixels

Horn-Schunk method for optical flow

- Now we substitute $\Delta u(x,y) = \overline{u}(x,y) - u(x,y)$ in:

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta u = 0$$
$$I_y(I_x u + I_y v + I_t) - \alpha^2 \Delta v = 0$$

- To get:

$$(I_x^2 + \alpha^2)u + I_x I_y v = \alpha^2 \overline{u} - I_x I_t$$
$$I_x I_y u + (I_y^2 + \alpha^2)v = \alpha^2 \overline{v} - I_y I_t$$

- Which is linear in u and v and can be solved analytically for each pixel individually.

Horn-Schunk method for optical flow

- Analytical solution for:

$$(I_x^2 + \alpha^2)u + I_x I_y v = \alpha^2 \overline{u} - I_x I_t$$
$$I_x I_y u + (I_y^2 + \alpha^2)v = \alpha^2 \overline{v} - I_y I_t$$

- is:

$$u = \bar{u} - \frac{I_x(I_x\bar{u} + I_y\bar{v} + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

$$v = \bar{v} - \frac{I_y(I_x\bar{u} + I_y\bar{v} + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

Iterative Horn-Schunk

- Similar to iterative Lucas-Kanade, there is an iterative version of Horn-Schunk algorithm.
- Since the solution depends on $\overline{u}$ and $\overline{v}$, this calculation becomes more accurate as we iteratively update the average flow.

- After each calculate, re-calculate $\overline{u}$ and $\overline{v}$

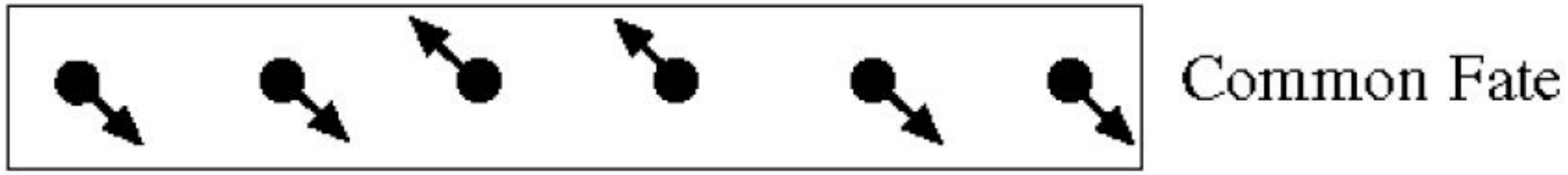
What we will learn today?

- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

Key assumptions

- **Small motion:** points do not move very far
- **Brightness constancy:** projection of the same point looks the same in every frame
- **Spatial coherence:** points move like their neighbors

Reminder: Gestalt – common fate

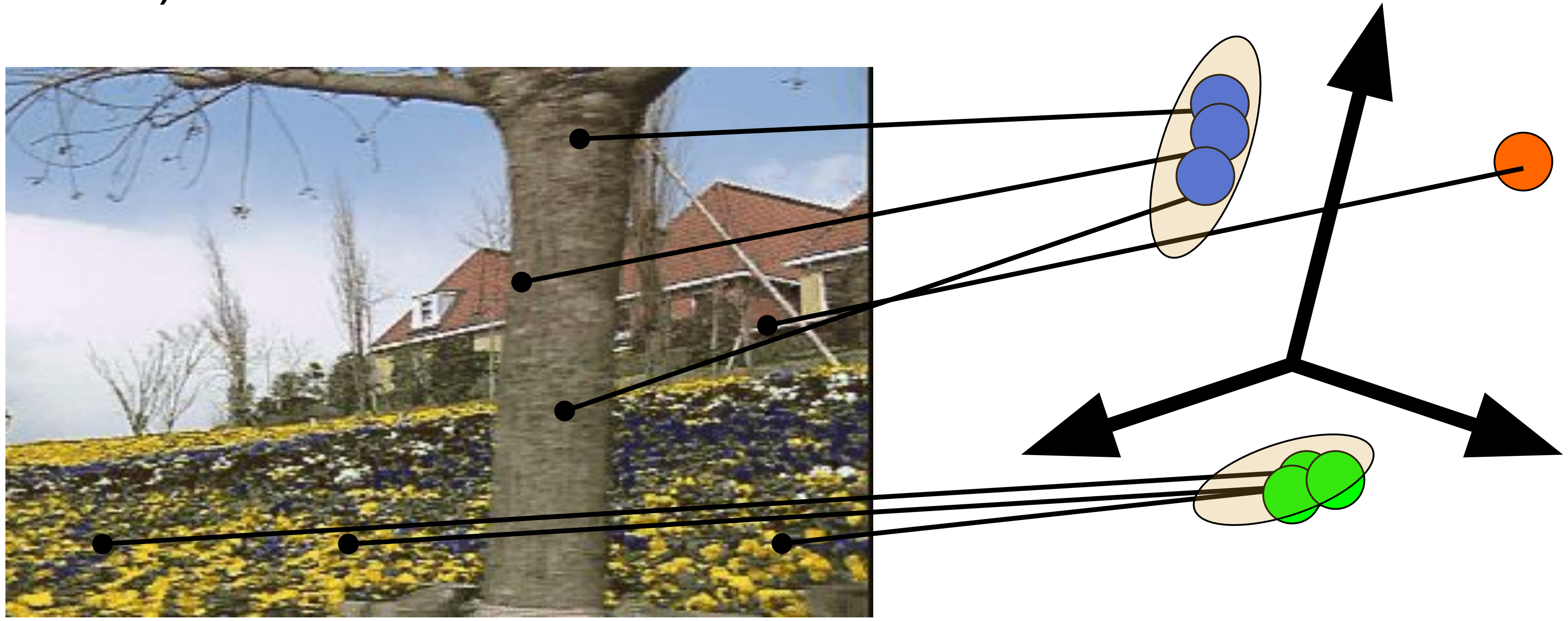


Segmentation using motion

- Use optical flow as the feature representation of each pixel

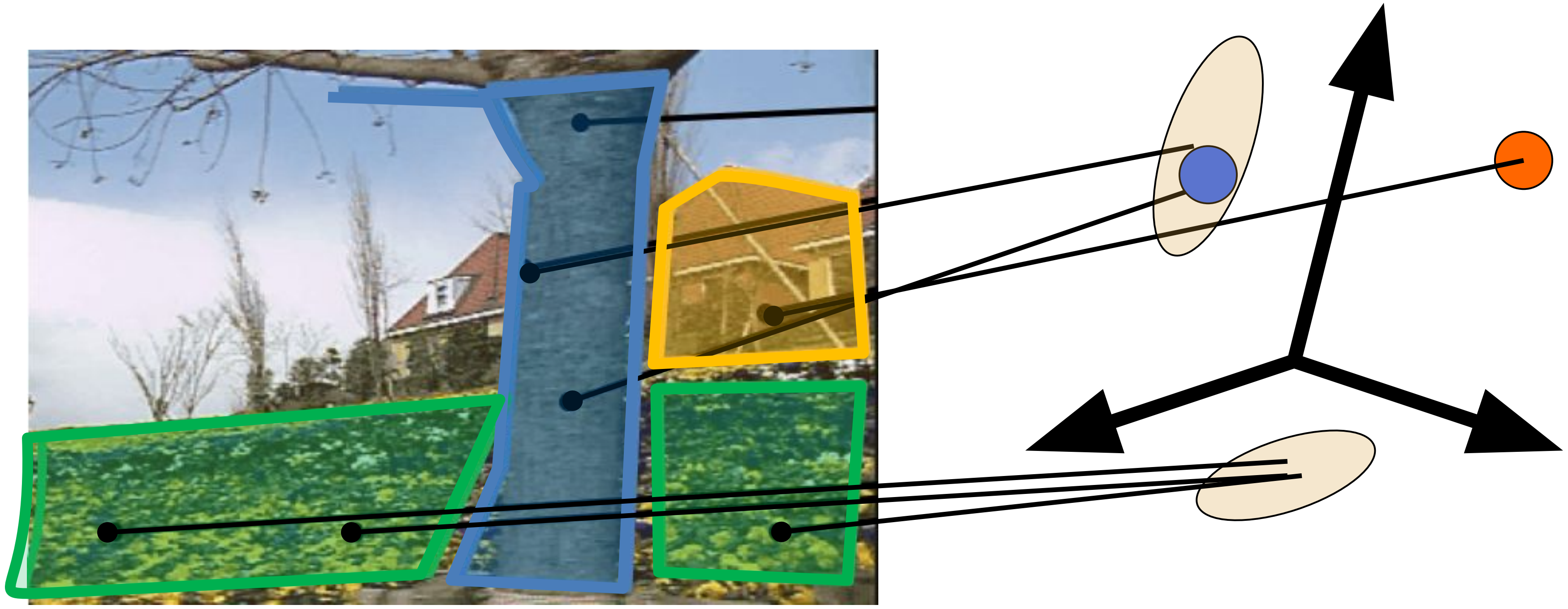
Segmentation using motion

- Use any of the segmentation algorithms: (k-means, Agglomerative clustering, mean shift)



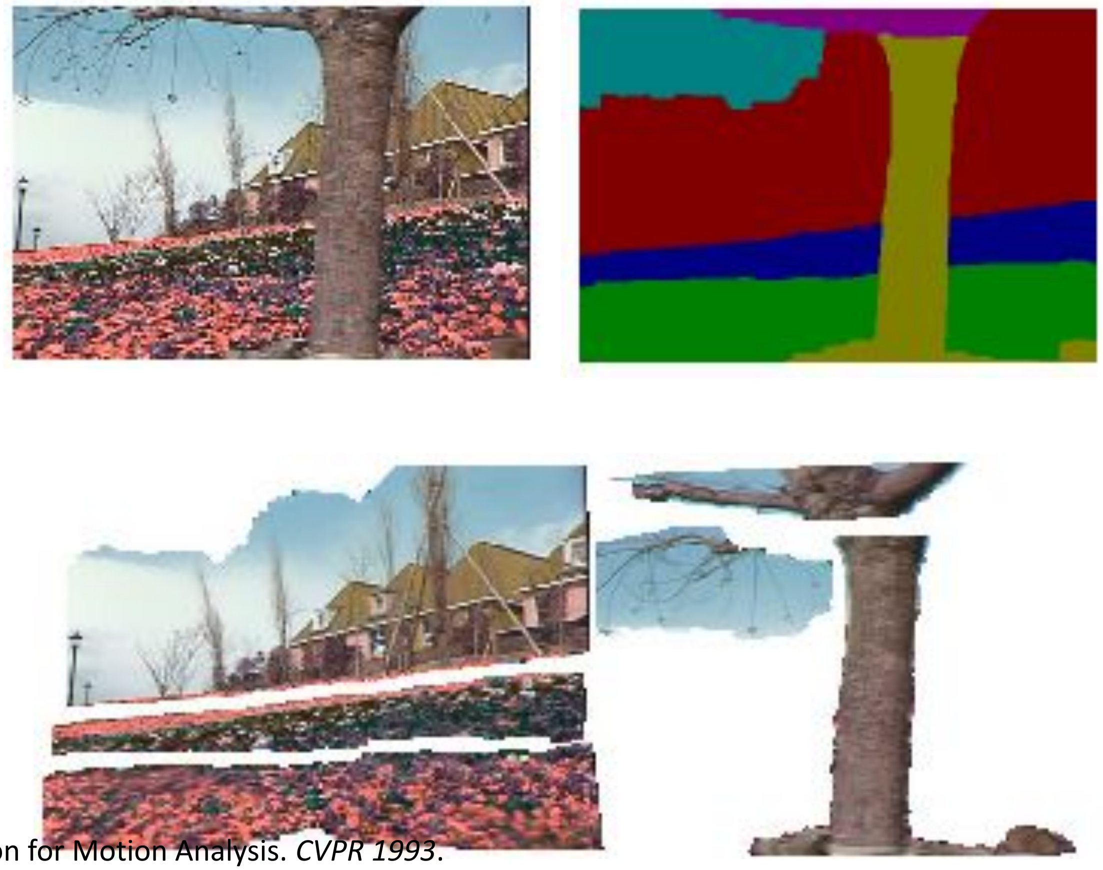
Segmentation using motion

- Use any of the segmentation algorithms: (k-means, Agglomerative clustering, mean shift)



Segmentation using motion

- Use any of the segmentation algorithms: (k-means, Agglomerative clustering, mean shift)

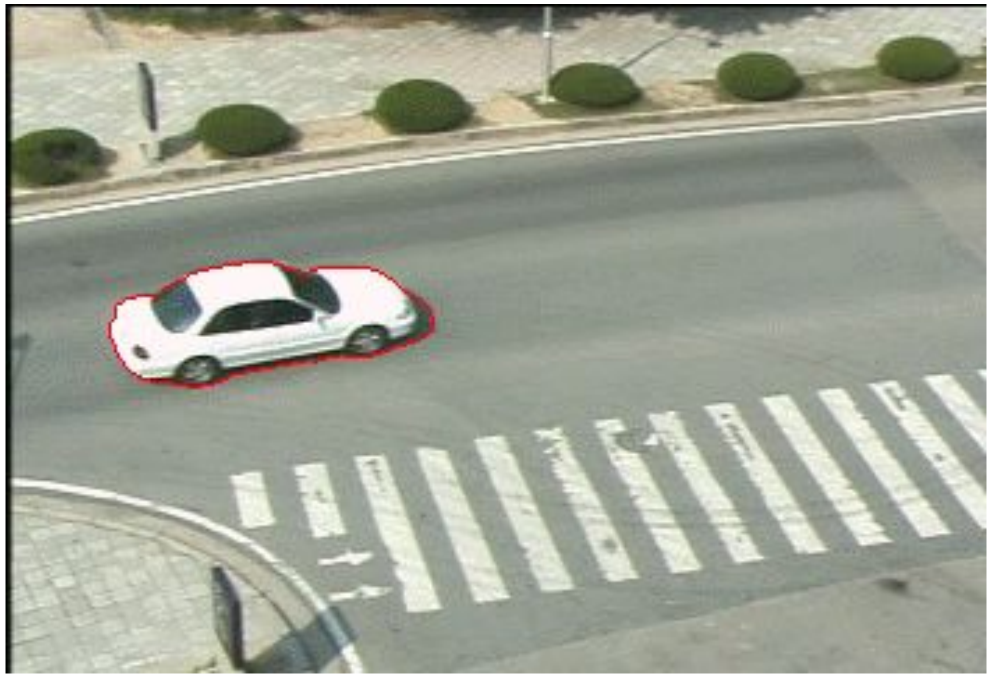


J. Wang and E. Adelson. Layered Representation for Motion Analysis. *CVPR 1993*.

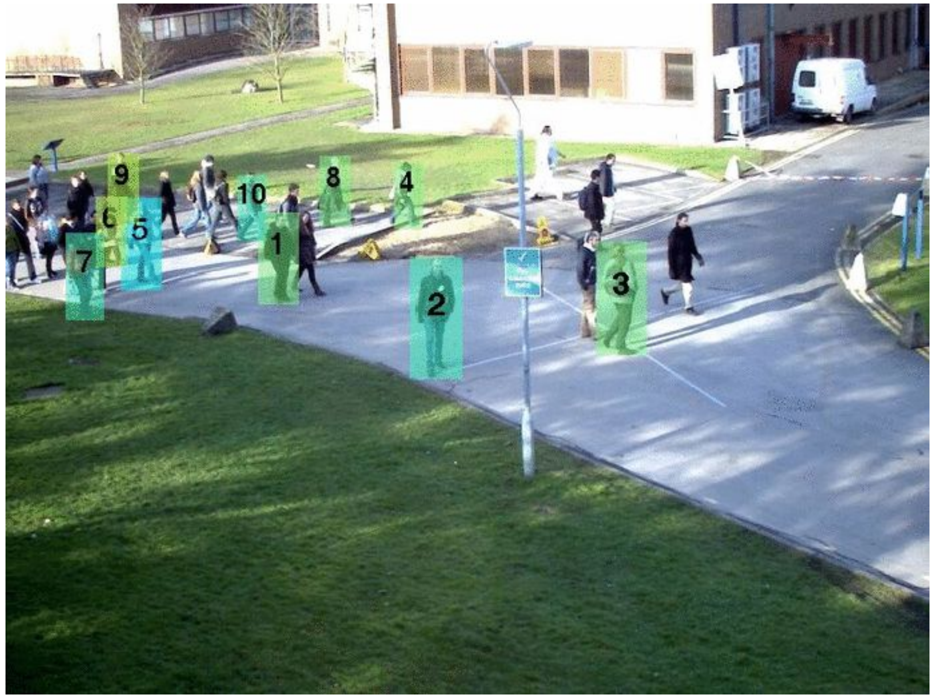
Today’s agenda

- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

Single object tracking

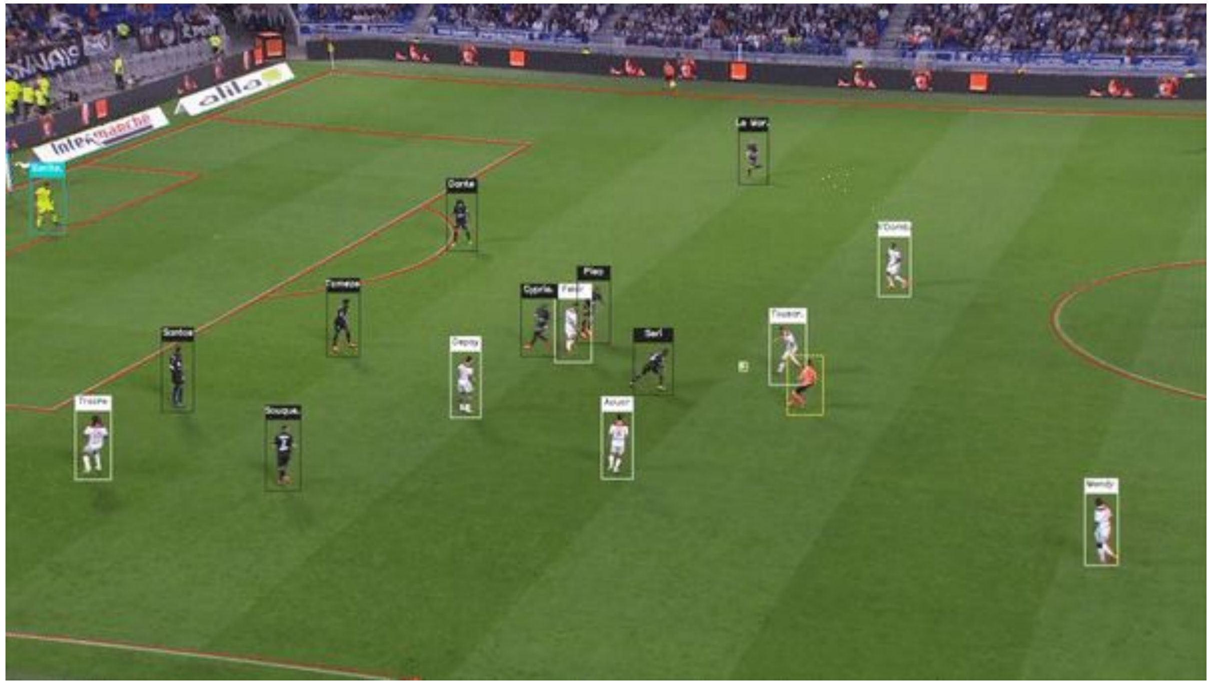


Multiple object tracking



Tracking with a fixed camera

Tracking with a fixed camera



Tracking with a moving camera

Challenges in Feature tracking

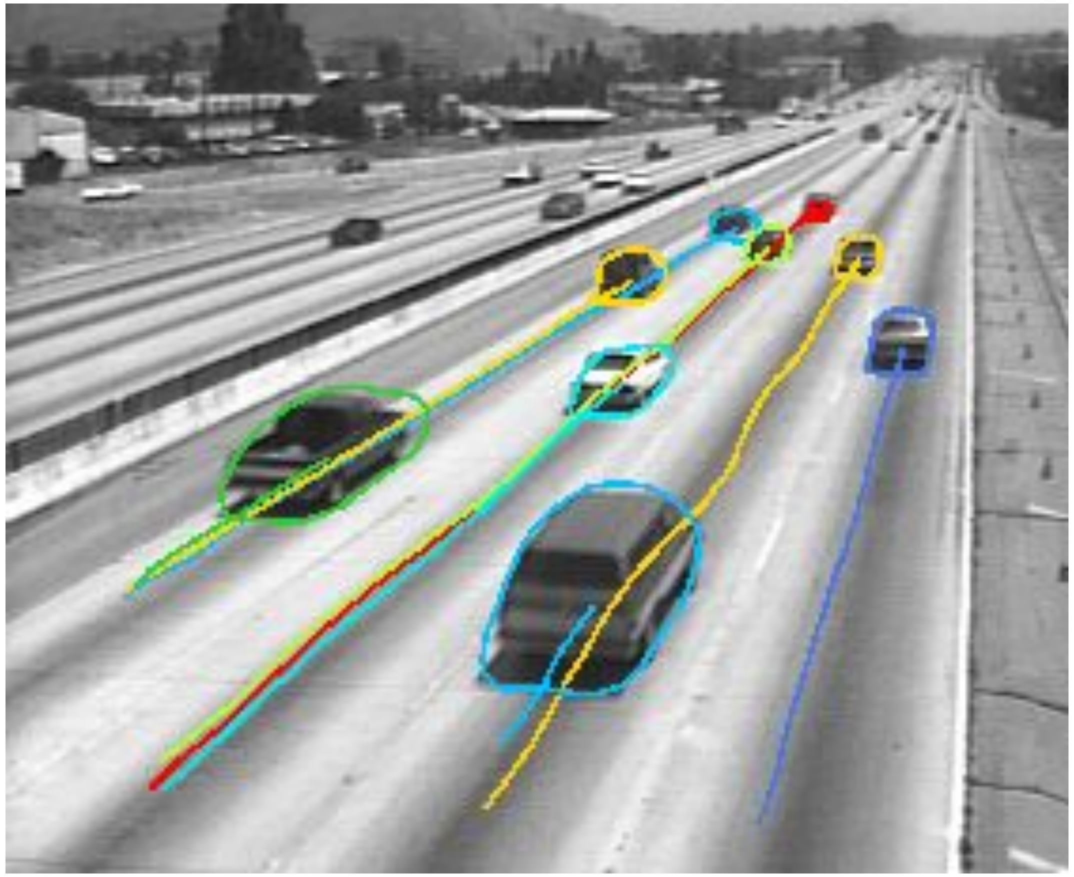
- Figure out which features can be tracked
  - Efficiently track across frames
- Some points may change appearance over time
  - e.g., due to rotation, moving into shadows, etc.
- Drift: small errors can accumulate over time
- Points may appear or disappear.
  - need to be able to add/delete tracked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

What are good features to track?

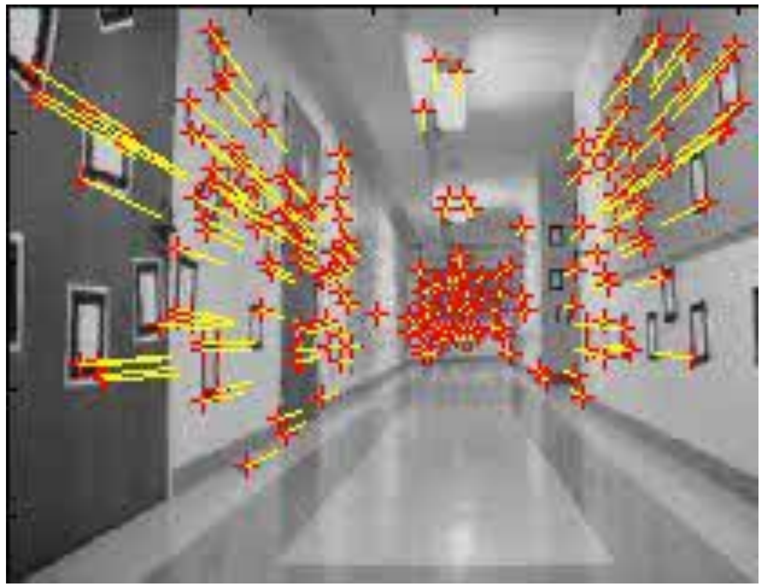
- Intuitively, we want to avoid smooth regions and edges. But is there a more is principled way to define good features?
- What kinds of image regions can we detect easily and consistently?
  - SIFT blobs!
  - Harris corners!

Optical flow can help track features

Once we have the features we want to track, lucas-kanade or other optical flow algorithm can help track those features

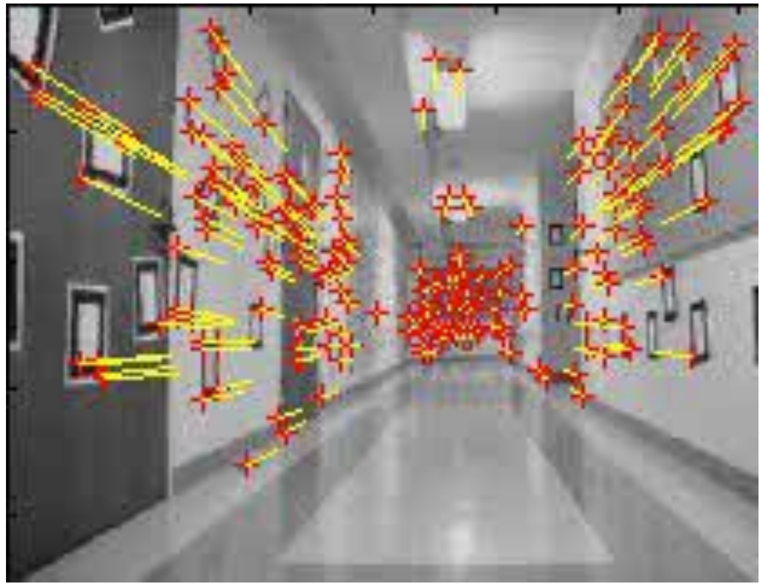


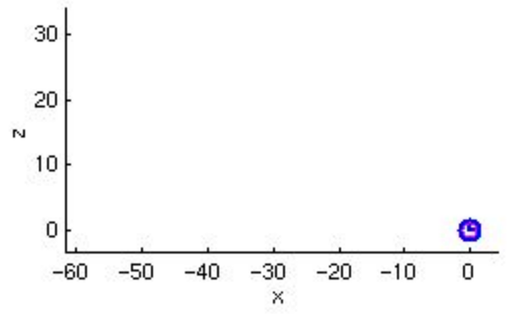
Feature-tracking



Courtesy of Jean-Yves Bouguet – Vision Lab, California Institute of Technology

Feature-tracking





Courtesy of Jean-Yves Bouguet – Vision Lab, California Institute of Technology

Simple KLT tracker

1. Find a good point to track (harris corner)
2. For each Harris corner compute optical flow (translation or affine) between consecutive frames.
3. Link motion vectors in successive frames to get a track for each Harris point
4. Introduce new Harris points by applying Harris detector at every m (10 or 15) frames
5. Track new and old Harris points using steps 1-3

KLT tracker for fish

Video credit: Kanade

Tracking cars

Video credit: Kanade

Tracking movement

Video credit: Kanade

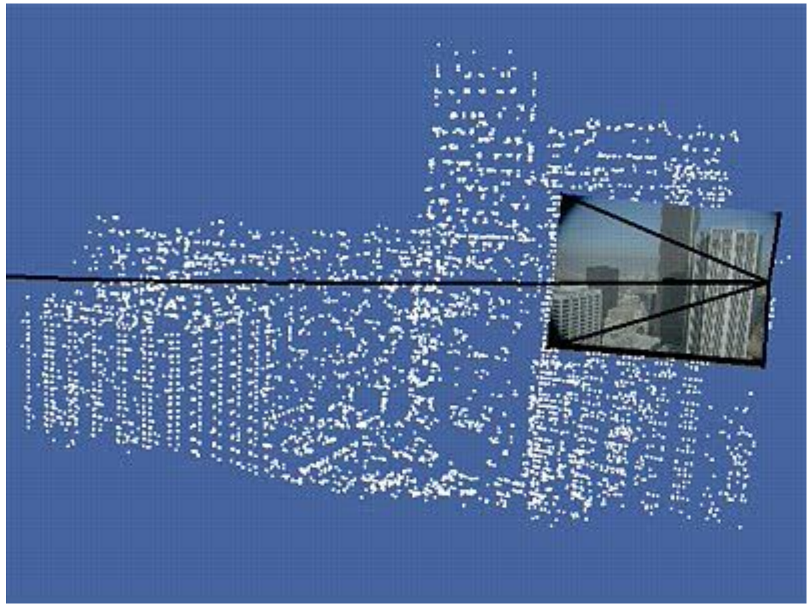
What we will learn today?

- Optical flow
- Lucas-Kanade method
- Pyramids for large motion
- Horn-Schunk method
- Segmentation from motion
- Tracking
- Applications

Uses of motion

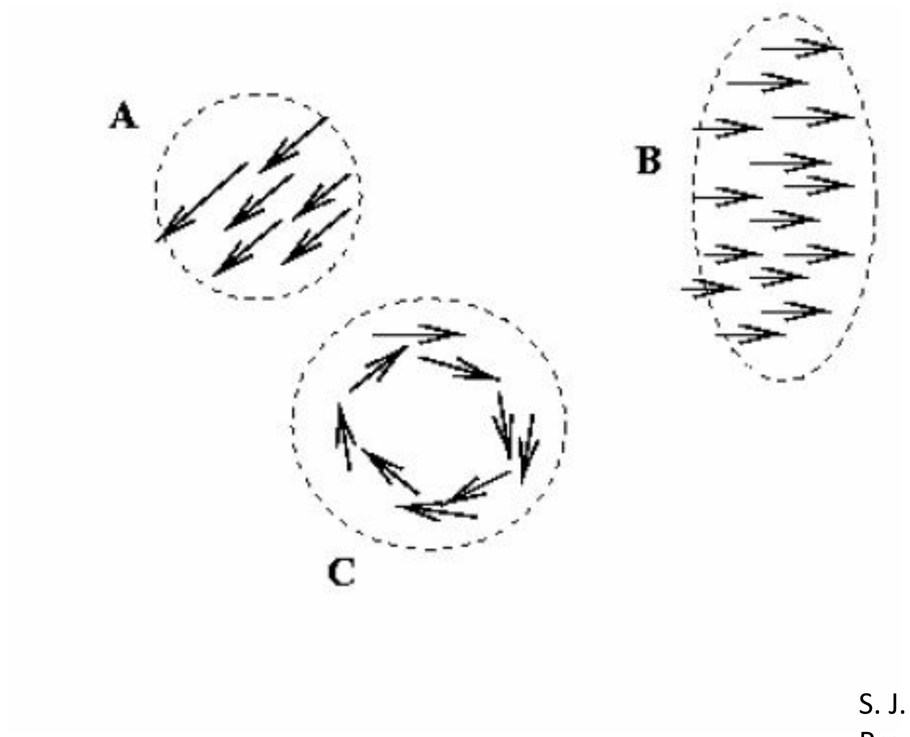
- Segmenting objects based on motion cues
- Learning dynamical models
- Improving video quality
  - Motion stabilization
  - Super resolution
- Tracking objects
- Recognizing events and activities

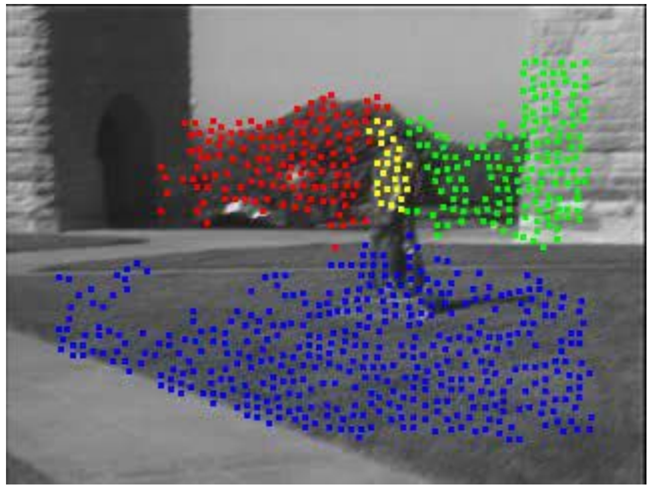
Estimating 3D structure



Segmenting objects based on motion cues

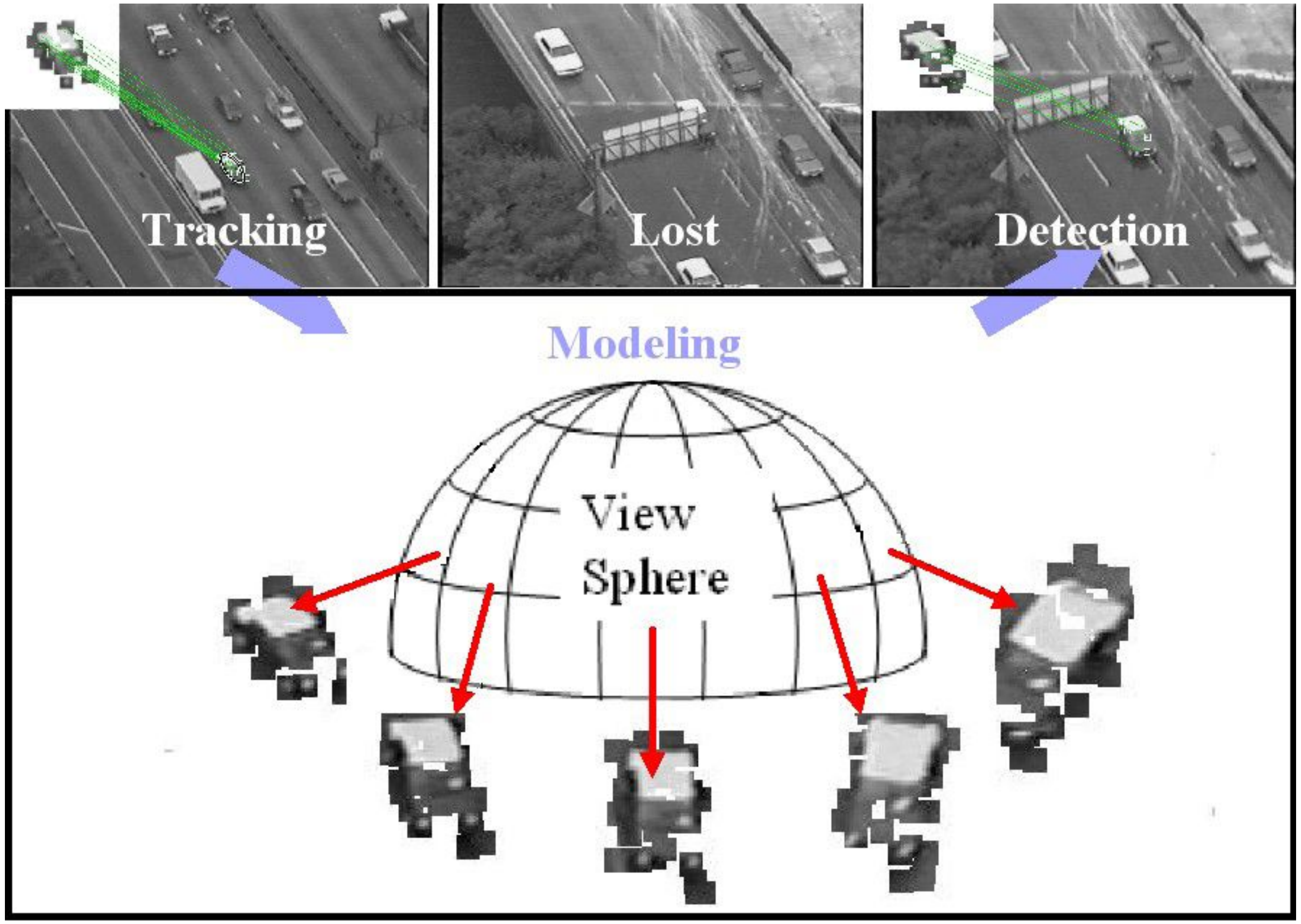
- Motion segmentation
  - Segment the video into multiple *coherently* moving objects





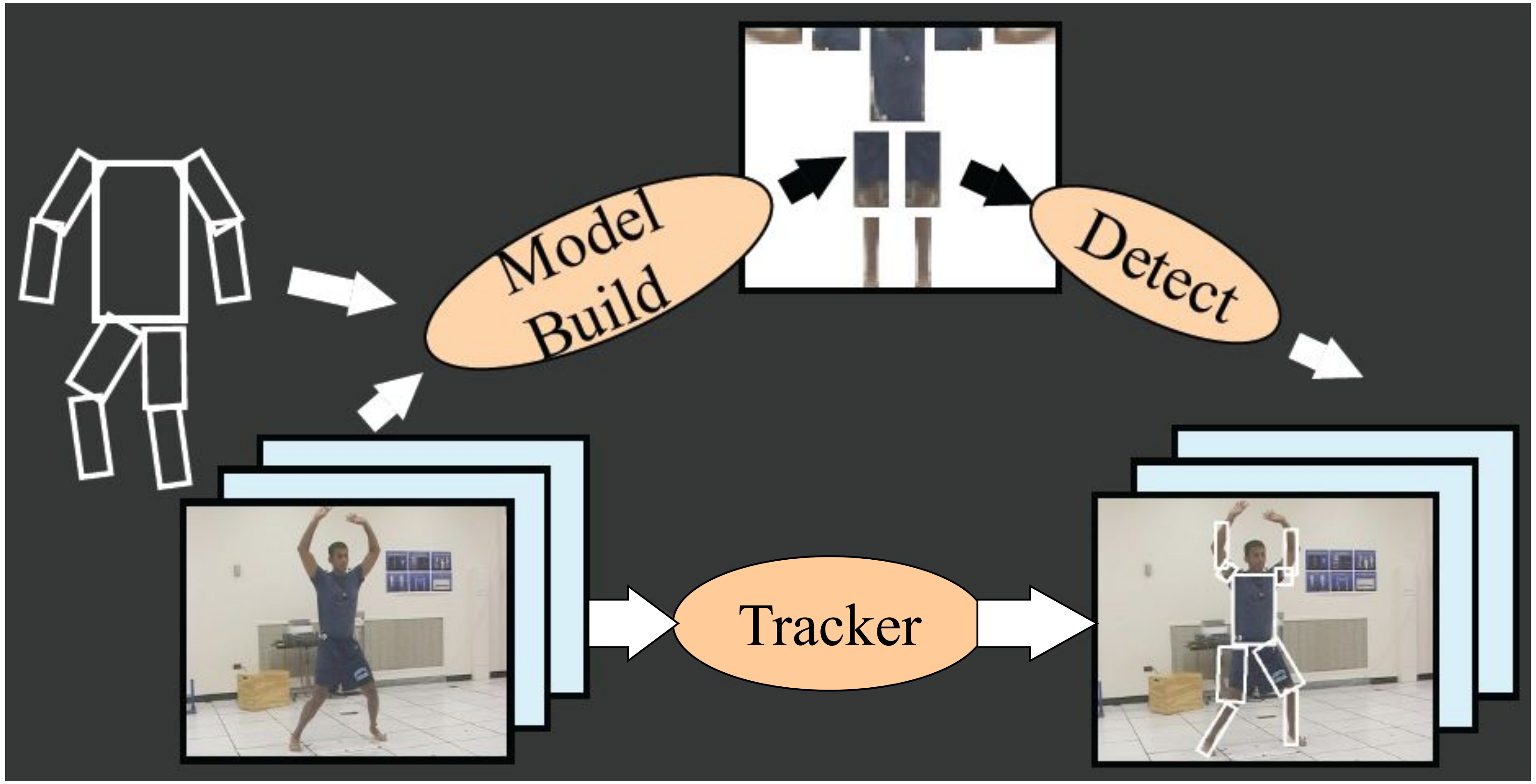
S. J. Pundlik and S. T. Birchfield, Motion Segmentation at Any Speed, Proceedings of the British Machine Vision Conference (BMVC) 2006

Tracking objects



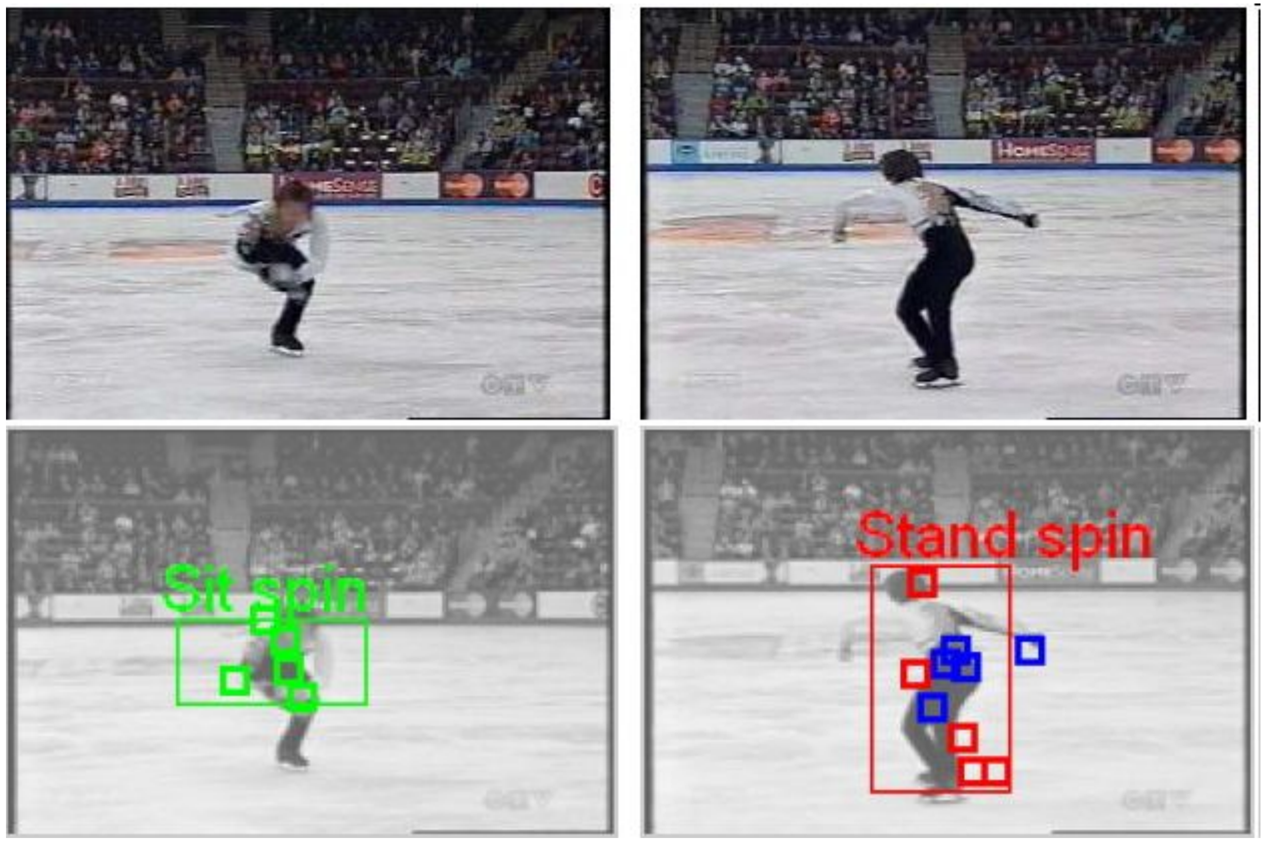
Z.Yin and R.Collins, "On-the-fly Object Modeling while Tracking," *IEEE Computer Vision and Pattern Recognition (CVPR '07)*, Minneapolis, MN, June 2007.

Recognizing events and activities



D. Ramanan, D. Forsyth, and A. Zisserman. Tracking People by Learning their Appearance. PAMI 2007.

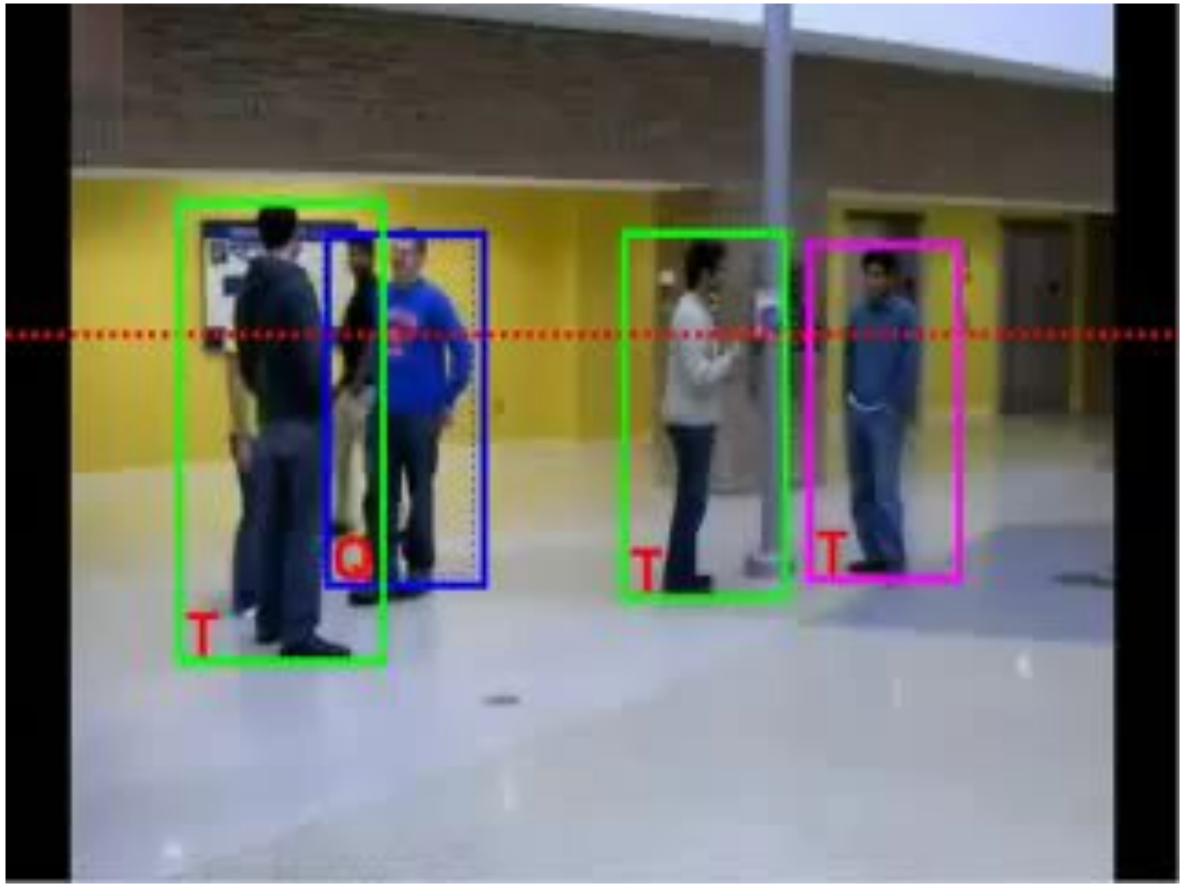
Recognizing events and activities



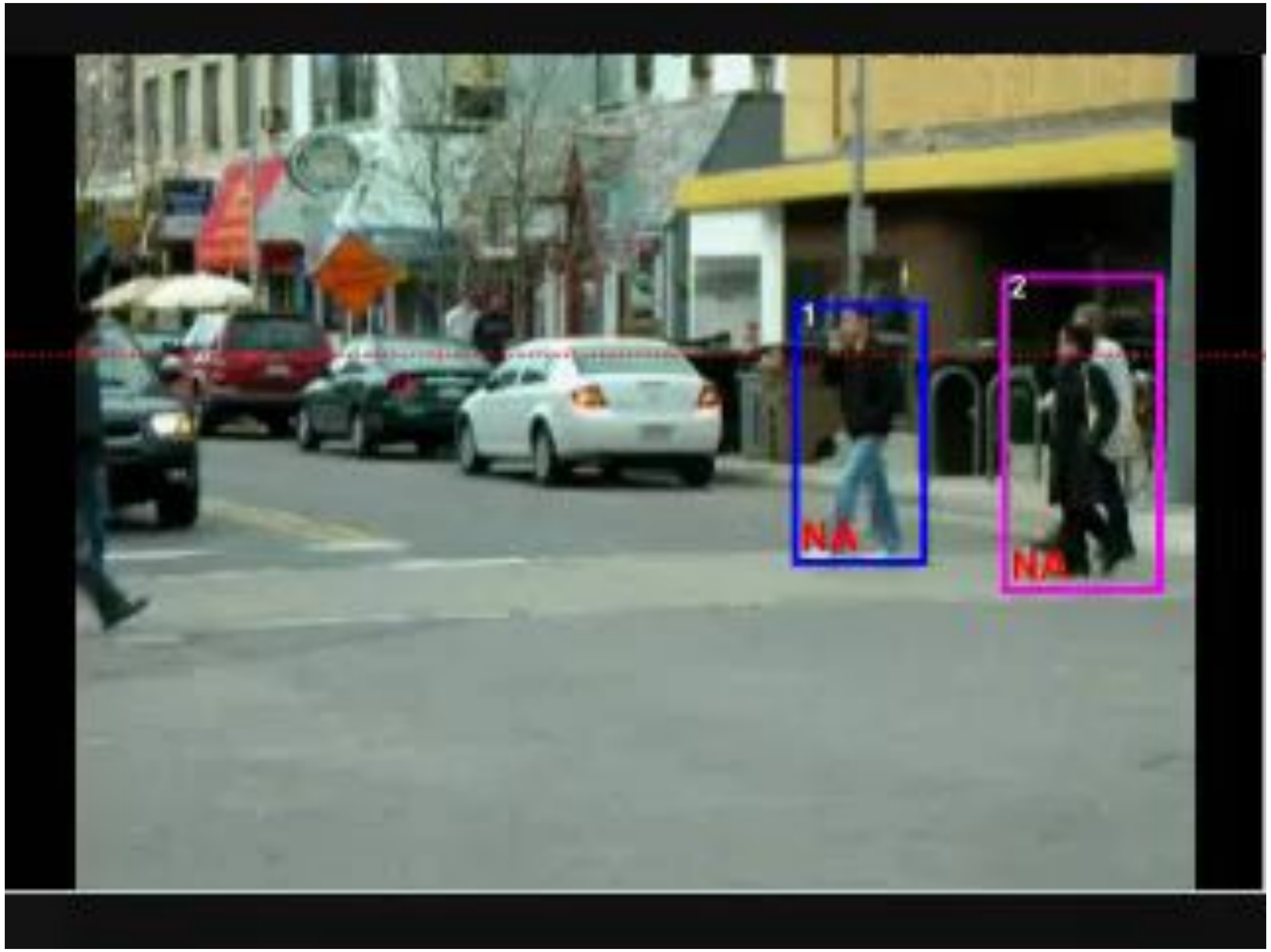
Juan Carlos Niebles, Hongcheng Wang and Li Fei-Fei, **Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words**, *(**BMVC**)*, Edinburgh, 2006.

Recognizing events and activities

Crossing – Talking – Queuing – Dancing – jogging



W. Choi & K. Shahid & S. Savarese WMC 2010



W. Choi, K. Shahid, S. Savarese, "What are they doing? : Collective Activity Classification Using Spatio-Temporal Relationship Among People", 9th International Workshop on Visual Surveillance (VSWS09) in conjuction with ICCV 09

Today's agenda

- Optical flow
- Lucas-Kanade method
- Horn-Schunk method
- Pyramids for large motion
- Segmentation from motion
- Tracking
- Applications

Reading: [Szeliski] Chapters: 8.4, 8.5

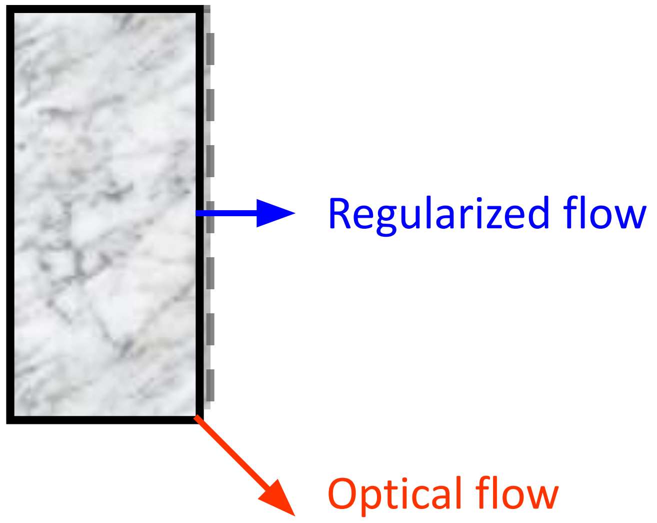
[Fleet & Weiss, 2005]
http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf

Lecture 18

Learning systems of filters

What does the smoothness regularization doing?

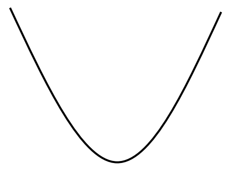
- It's a sum of squared terms (a Euclidian distance measure).
- We're putting it in the expression to be minimized.
- => In texture free regions, *there is no optical flow*
- => On edges, points will flow to nearest points, solving the aperture problem.



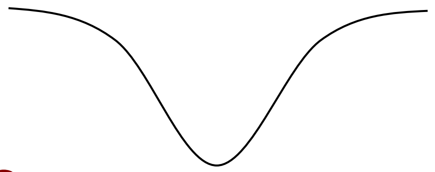
Dense Optical Flow with Michael Black's method

- Michael Black took Horn-Schunk's method one step further, starting from the regularization constant:
- Which looks like a quadratic:

$$\|\nabla u\|^2 + \|\nabla v\|^2$$



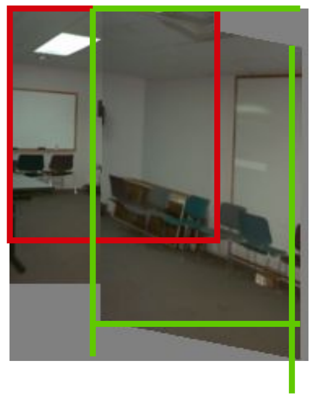
- And replaced it with this:



- Why does this regularization work better?

Affine motion

$$u(x,y) = a_1 + a_2 x + a_3 y$$
$$v(x,y) = a_4 + a_5 x + a_6 y$$



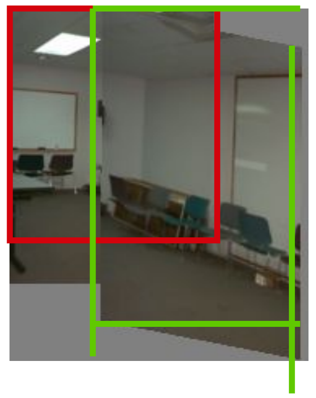
- Substituting into the brightness constancy equation:

$$\boxed{I_x \cdot u + I_y \cdot v + I_t \approx 0}$$

Affine motion

$$u(x, y) = a_1 + a_2 x + a_3 y$$

$$v(x, y) = a_4 + a_5 x + a_6 y$$



- Substituting into the brightness constancy equation:

$$\boxed{I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \approx 0}$$

- Each pixel provides 1 linear constraint in 6 unknowns

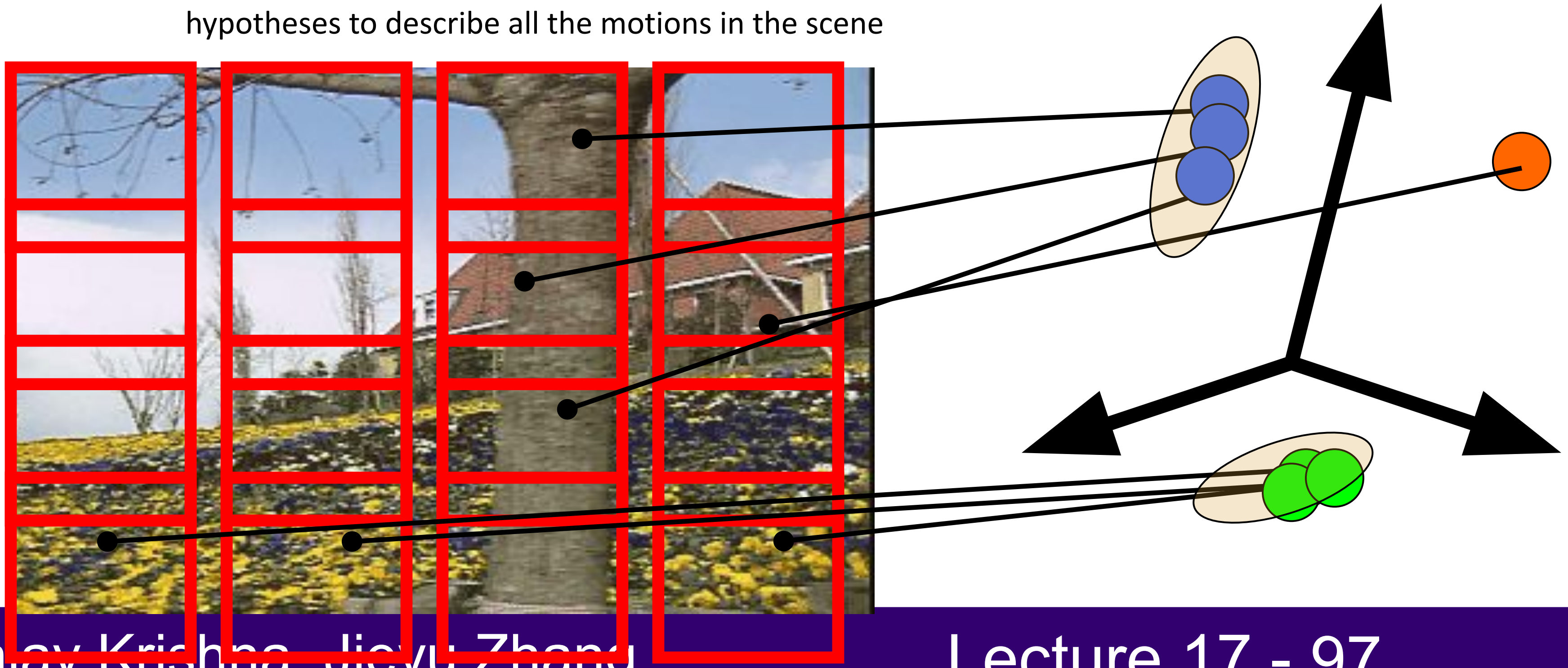
- Least squares minimization:

$$\boxed{Err(\vec{a}) = \sum \left[I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \right]^2}$$

How do we estimate the layers?

- 1. Obtain a set of initial affine motion hypotheses
  - Divide the image into blocks and estimate affine motion parameters in each block by least squares
    - Eliminate hypotheses with high residual error

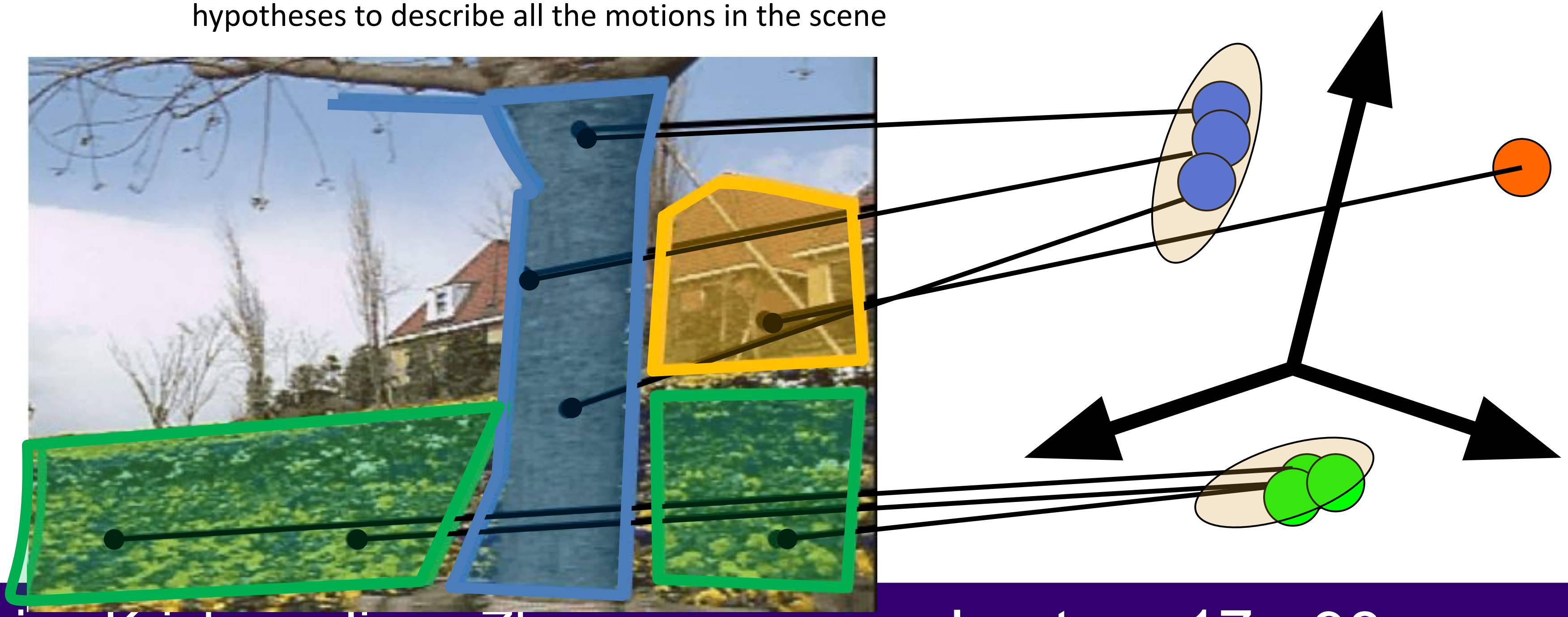
- Map into motion parameter space
- Perform k-means clustering on affine motion parameters
  - –Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene



How do we estimate the layers?

- 1. Obtain a set of initial affine motion hypotheses
  - Divide the image into blocks and estimate affine motion parameters in each block by least squares
    - Eliminate hypotheses with high residual error

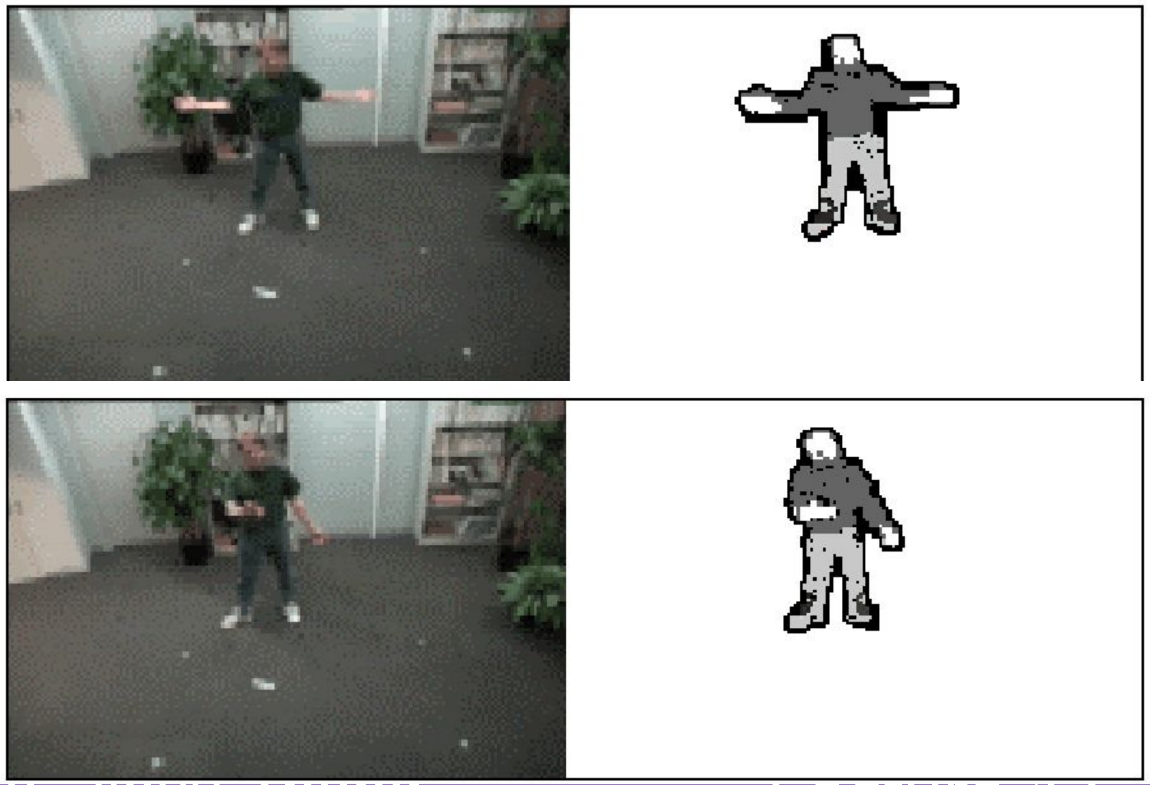
- Map into motion parameter space
- Perform k-means clustering on affine motion parameters
  - –Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene



Synthesizing dynamic textures

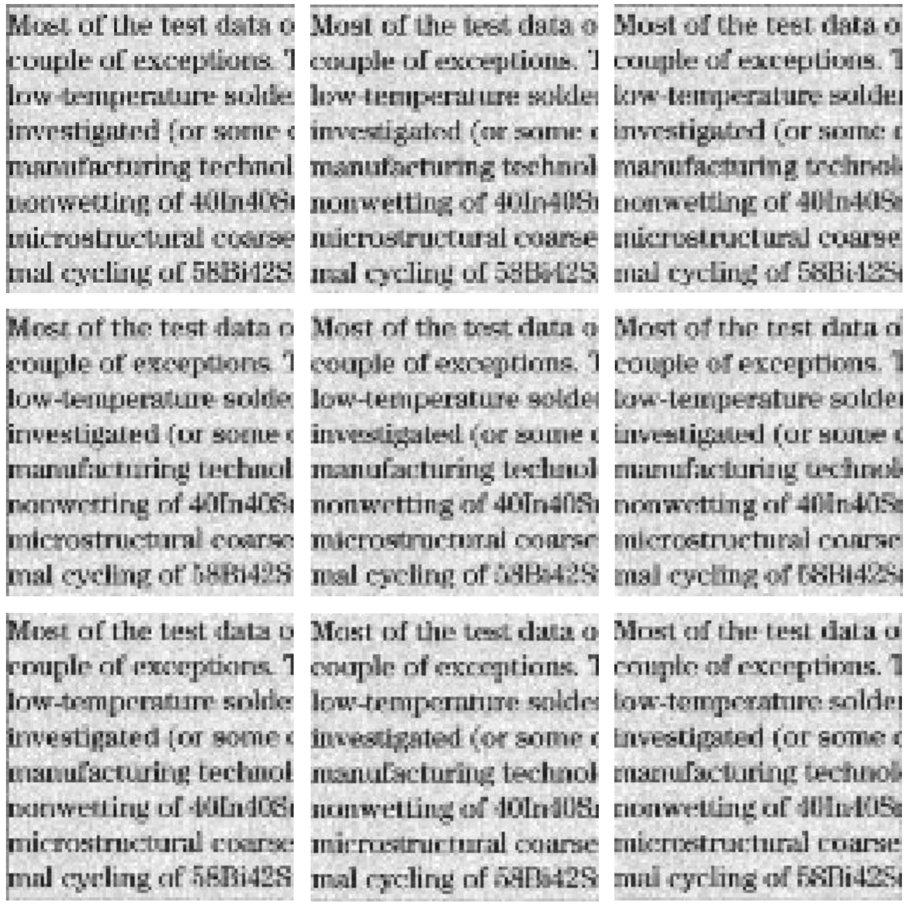
Segmenting objects based on motion cues

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



Super-resolution

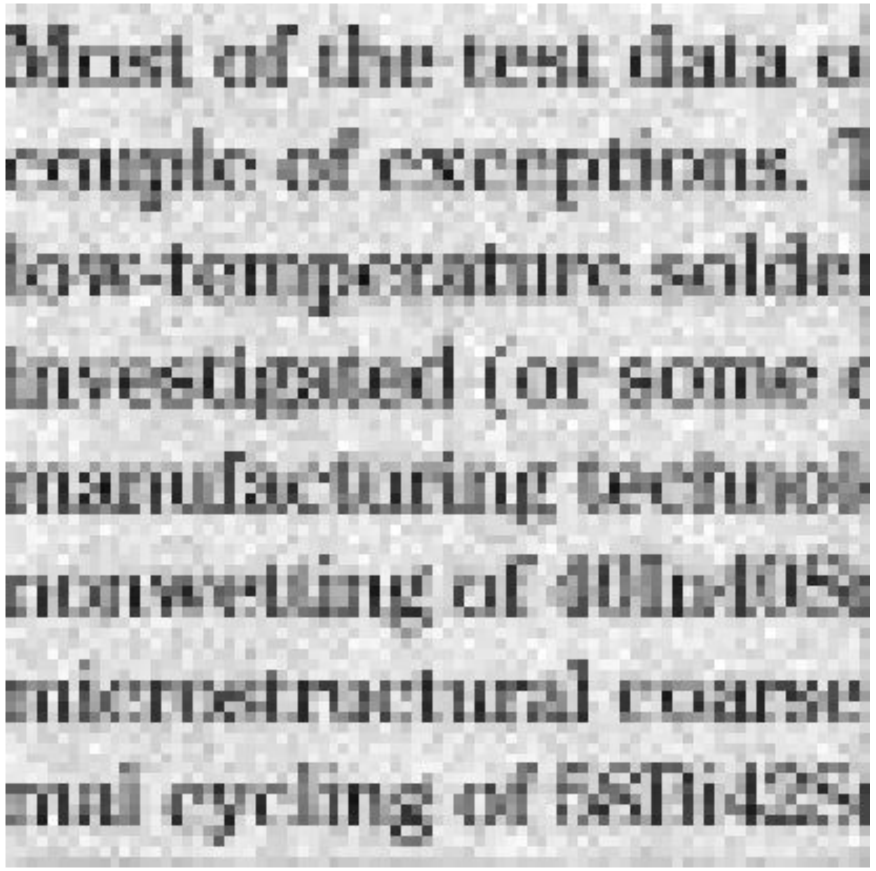
Example: A set of low quality images



Most of the test data o
couple of exceptions. T
low-temperature solde
investigated (or some c
manufacturing technol
nonwetting of 40In40S
microstructural coarse
mal cycling of 58Bi42S

Super-resolution

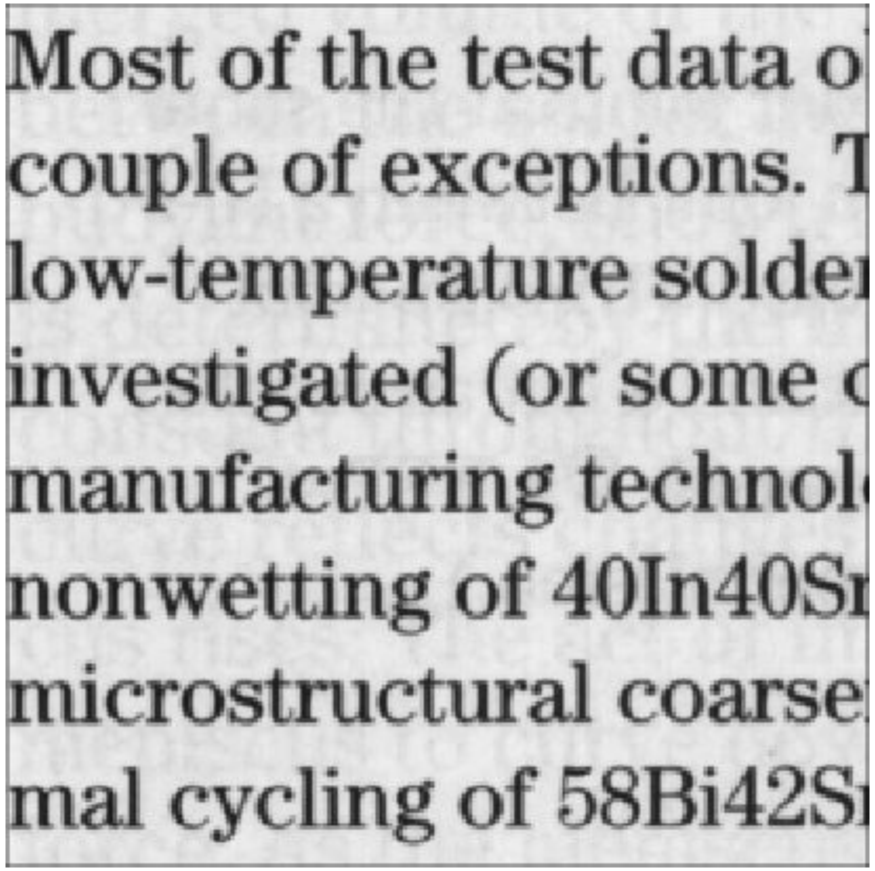
Each of these images looks like this:



Most of the test data o
couple of exceptions. T
low-temperature solder
investigated (or some c
manufacturing technol
nonwetting of 40In40S
microstructural coarse
mal cycling of 58Bi42S

Super-resolution

The recovery result:


Most of the test data o
couple of exceptions. T
low-temperature solde
investigated (or some c
manufacturing technol
nonwetting of 40In40S
microstructural coarse
mal cycling of 58Bi42S

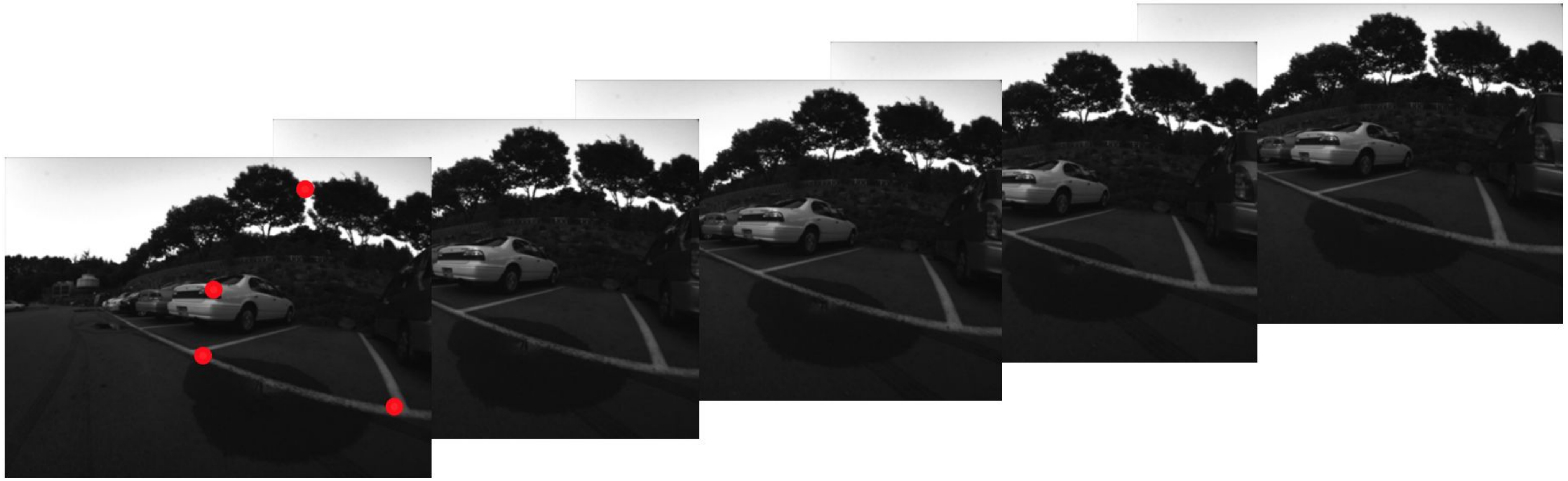
Problem statement

Image sequence

Slide credit: Yonsei Univ.

Problem statement

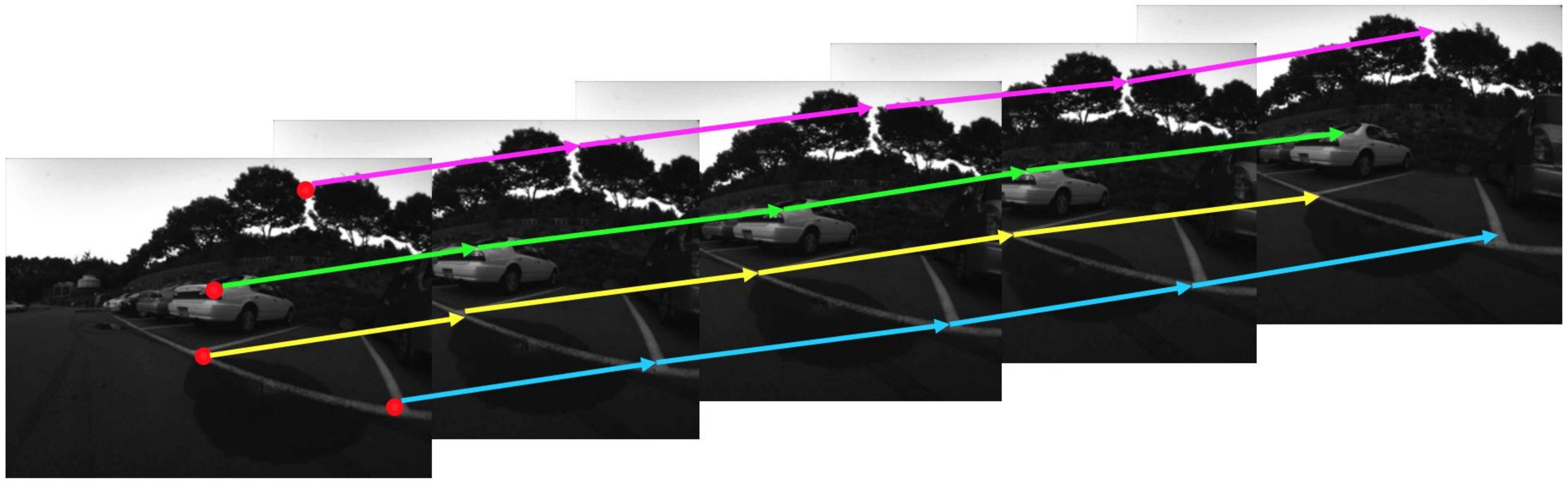
Feature point detection



Slide credit: Yonsei Univ.

Problem statement

Feature point tracking



Slide credit: Yonsei Univ.

What we will learn today?

- Feature Tracking
- Simple KLT tracker
- 2D transformations
- Iterative KLT tracker

Reading: [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]
http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf

Problem setting

- Given a video sequence, find all the features and track them across the video.
- First, use Harris corner detection to find features and their location $\boldsymbol{x}$.
- For each feature at location $\boldsymbol{x} = [x \;\; y]^T$:
  - Choose a descriptor create an initial template for that feature: $T(\boldsymbol{x})$.

KLT objective

- Our aim is to find the $\boldsymbol{p}$ that minimizes the difference between the template $T(\boldsymbol{x})$ and the description of the new location of $\boldsymbol{x}$ after undergoing the transformation.

$$\sum_{x} [I(W(\boldsymbol{x};\boldsymbol{p})) - T(x)]^2$$

- For all the features $\boldsymbol{x}$ in the image $\boldsymbol{I}$,
  - $I(W(\boldsymbol{x};\boldsymbol{p}))$ is the estimate of where the features move to in the next frame after the transformation defined by $W(\boldsymbol{x};\boldsymbol{p})$. Recall that $\boldsymbol{p}$ is our vector of parameters.
  - Sum is over an image patch around $\boldsymbol{x}$.

KLT objective

- Since $\boldsymbol{p}$ may be large, minimizing this function may be difficult:

$$\sum_{x} [I(W(\boldsymbol{x};\boldsymbol{p})) - T(x)]^2$$

- We will instead break down $\boldsymbol{p} = \boldsymbol{p_0} + \Delta\boldsymbol{p}$
  - Large + small/residual motion
  - Where $\boldsymbol{p_0}$ is going to be fixed and we will solve for $\Delta\boldsymbol{p}$, which is a small value.
  - We can initialize $\boldsymbol{p_0}$ with our best guess of what the motion is and initialize $\Delta\boldsymbol{p}$ as zero.

A little bit of math: Taylor series

- Taylor series is defined as:

$$f(x+\ \Delta x)=f(x)+\Delta x\frac{\partial f}{\partial x}+\Delta x^2\frac{\partial^2 f}{\partial x^2}+\ldots$$

- Assuming that Δx is small.
- We can apply this expansion to the KLT tracker and only use the first two terms:

Expanded KLT objective

- $$\sum_{x} [I(W(\boldsymbol{x}; \boldsymbol{p}_0 + \Delta\boldsymbol{p})) - T(x)]^2$$
  $$\approx \sum_{x} \left[I(W(\boldsymbol{x}; \boldsymbol{p}_0)) + \nabla I \frac{\partial W}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(x) \right]^2$$

It's a good thing we have already calculated what $\frac{\partial W}{\partial \boldsymbol{p}}$ would look like for affine, translations and other transformations!

Expanded KLT objective

- So our aim is to find the $\Delta \boldsymbol{p}$ that minimizes the following:

$$\underset{\Delta \boldsymbol{p}}{\operatorname{argmin}} \sum_{x} \left[I(W(\boldsymbol{x}; \boldsymbol{p}_0)) + \nabla I \frac{\partial W}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(x) \right]^2$$

- Where $\nabla I = \begin{bmatrix} I_x & I_y \end{bmatrix}$
- Differentiate wrt $\Delta \boldsymbol{p}$ and setting it to zero:

$$\sum_{x} \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}} \right]^T \left[I(W(\boldsymbol{x}; \boldsymbol{p}_0)) + \nabla I \frac{\partial W}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(x) \right] = 0$$

Solving for $\Delta \boldsymbol{p}$

- Solving for $\Delta \boldsymbol{p}$ in:

$$\sum_x \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}}\right]^T \left[I(W(\boldsymbol{x};\boldsymbol{p_0})) + \nabla I \frac{\partial W}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(x)\right] = 0$$

- we get:

$$\Delta \boldsymbol{p} = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}}\right]^T [T(x) - I(W(\boldsymbol{x};\boldsymbol{p_0}))]$$

where $H = \sum_x \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}}\right]^T \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}}\right]$

Interpreting the H matrix for translation transformations

- $$H = \sum_x \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}}\right]^T \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}}\right]$$

Recall that

1. $\nabla I = [I_x \quad I_y]$ and
2. for translation motion, $\frac{\partial W}{\partial \boldsymbol{p}}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Therefore,

$$H = \sum_x \left[[I_x \quad I_y]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right]^T \left[[I_x \quad I_y]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right]$$

$$= \sum_x \boxed{\begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix}}$$

That's the Harris corner detector we learnt in class!!!

Interpreting the H matrix for affine transformations

$$H = \sum_{\mathbf{x}} \begin{bmatrix} I_x^2 & I_x I_y & x I_x^2 & y I_x I_y & x I_x I_y & y I_x I_y \\ I_x I_y & I_y^2 & x I_x I_y & y I_y^2 & x I_y^2 & y I_y^2 \\ x I_x^2 & y I_x I_y & x^2 I_x^2 & y^2 I_x I_y & xy I_x I_y & y^2 I_x I_y \\ y I_x I_y & y I_y^2 & xy I_x I_y & y^2 I_y^2 & xy I_y^2 & y^2 I_y^2 \\ x I_x I_y & x I_y^2 & x^2 I_x I_y & xy I_y^2 & x^2 I_y^2 & xy I_y^2 \\ y I_x I_y & y I_y^2 & xy I_x I_y & y^2 I_y^2 & xy I_y^2 & y^2 I_y^2 \end{bmatrix}$$

Can you derive this yourself similarly to how we derived the translation transformation?

Overall KLT tracker algorithm

$$\Delta \boldsymbol{p} = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial \boldsymbol{p}}\right]^T [T(x) - I(W(\boldsymbol{x}; \boldsymbol{p_0}))]$$

- Given the features from Harris detector:

1. Initialize $\boldsymbol{p_0}$ and $\Delta \boldsymbol{p}$.
2. Compute the initial templates $T(x)$ for each feature.
3. Transform the features in the image I with $W(\boldsymbol{x}; \boldsymbol{p_0})$.
4. Measure the error: $I(W(\boldsymbol{x}; \boldsymbol{p_0})) - T(x)$.
5. Compute the image gradients $\nabla I = [I_x \quad I_y]$.
6. Evaluate the Jacobian $\frac{\partial W}{\partial \boldsymbol{p}}$.
7. Compute steepest descent $\nabla I \frac{\partial W}{\partial \boldsymbol{p}}$.
8. Compute Inverse Hessian H^{-1}
9. Calculate the change in parameters $\Delta \boldsymbol{p}$
10. Update parameters $\boldsymbol{p_0} = \boldsymbol{p_0} + \Delta \boldsymbol{p}$
11. Repeat 2 to 10 until $\Delta \boldsymbol{p}$ is small.

KLT over multiple frames

- Once you find a transformation for two frames, you will repeat this process for every couple of frames.
- Run Harris detector every 15-20 frames to find new features.

Challenges to consider

- 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)
  - 15x15 to 31x31 seems typical
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)