Lecture 13:
Tracking motion features – optical flow

Professor Fei-Fei Li

Stanford Vision Lab

What we will learn today?

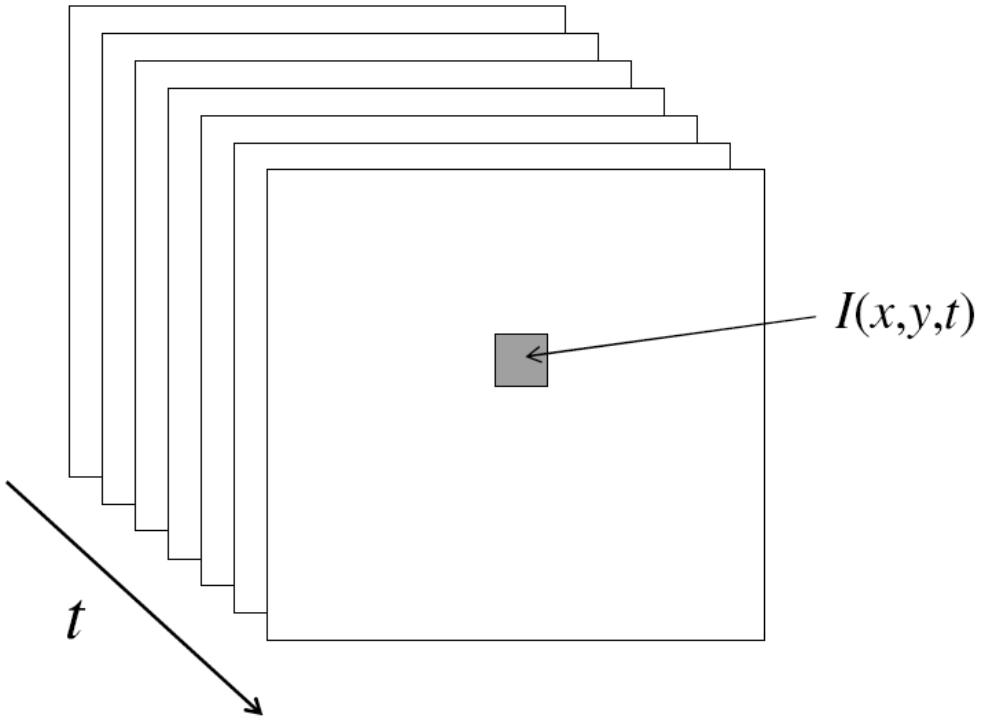
- Introduction
- Optical flow
- Feature tracking
- Applications
- (Supplementary) Technical note

Reading: [Szeliski] Chapters: 8.4, 8.5

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

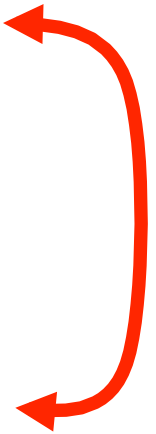
From images to videos

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

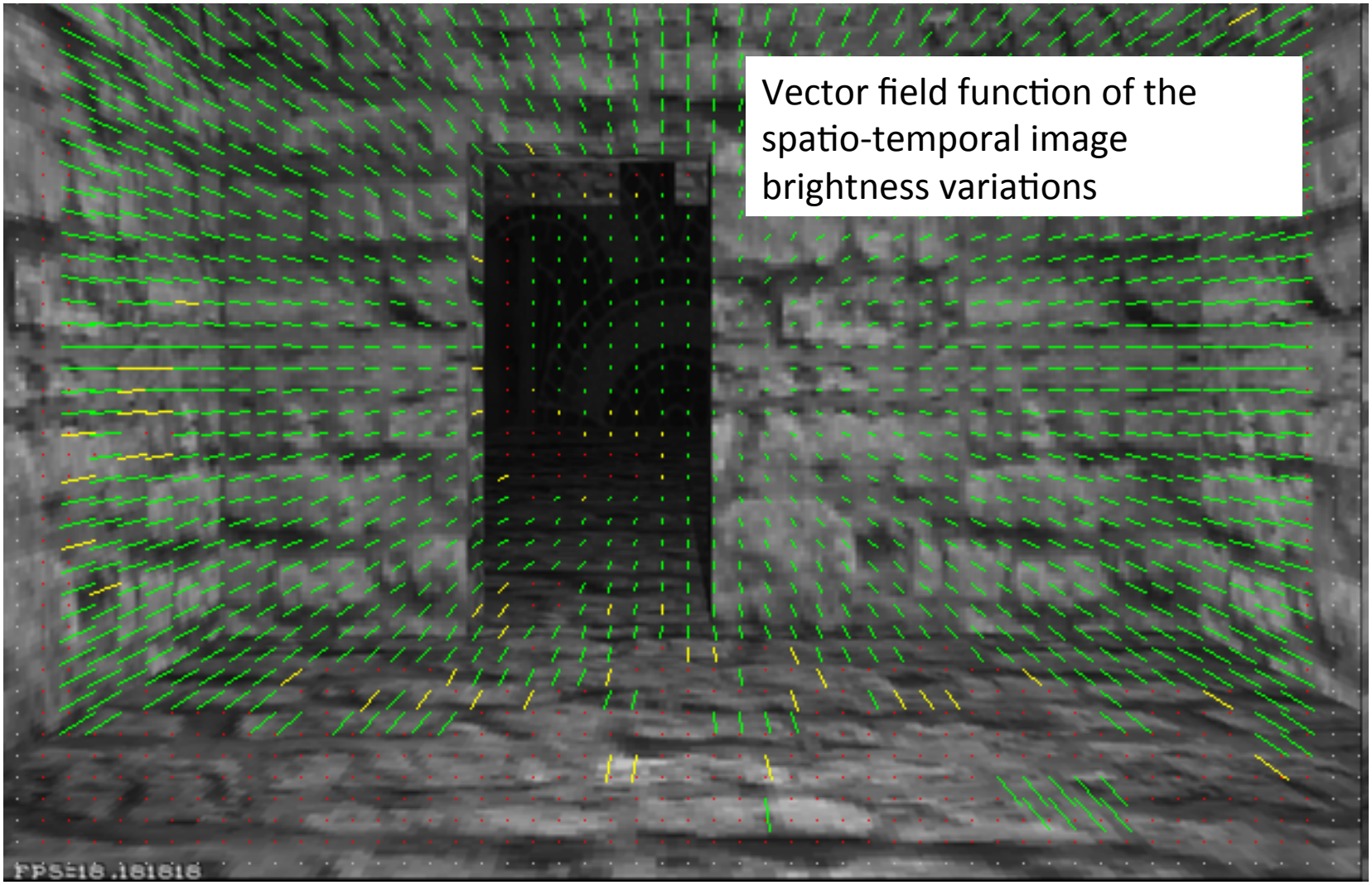


Motion estimation techniques

- Optical flow
  - Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow)
- Feature-tracking
  - Extract visual features (corners, textured areas) and "track" them over multiple frames

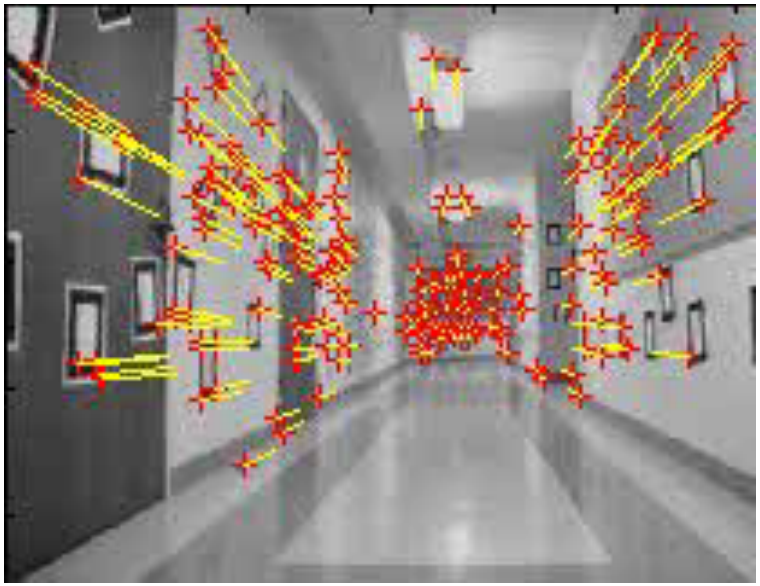


Optical flow



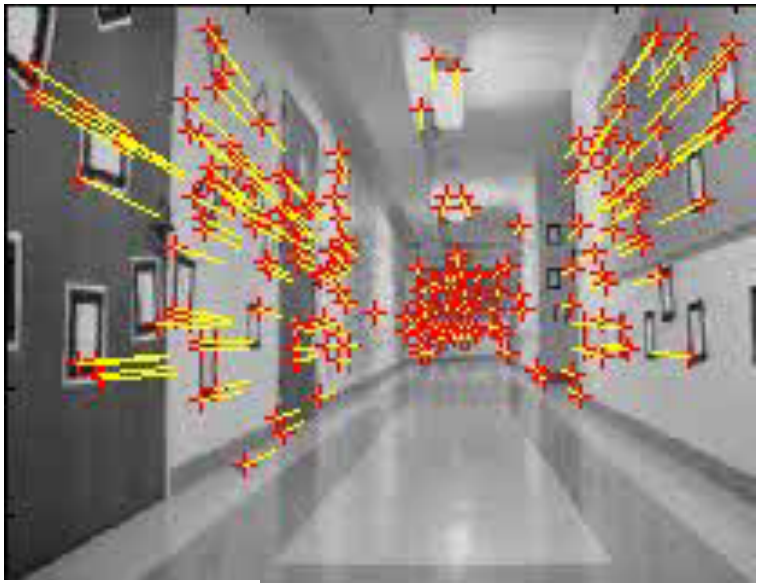
Picture courtesy of Selim Temizer - Learning and Intelligent Systems (LIS) Group, MIT

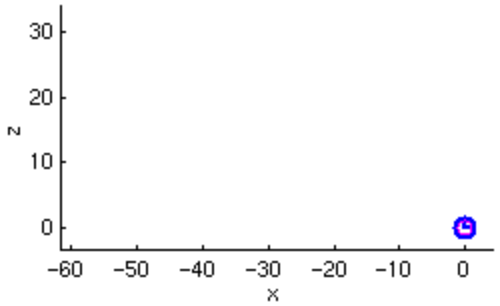
Feature-tracking



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

Feature-tracking





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

Optical flow

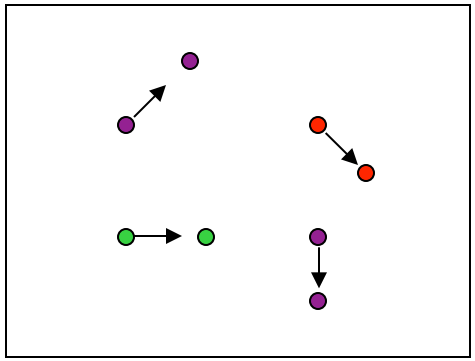
- Definition: optical flow is the *apparent* motion of brightness patterns in the image

- Note: apparent motion can be caused by lighting changes without any actual motion
  - Think of a uniform rotating sphere under fixed lighting vs. a stationary sphere under moving illumination

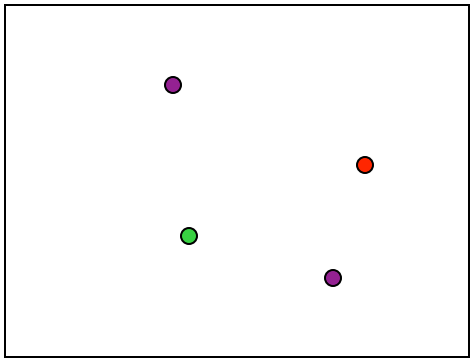
GOAL: Recover image motion at each pixel from optical flow

Source: Silvio Savarese

Estimating optical flow



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



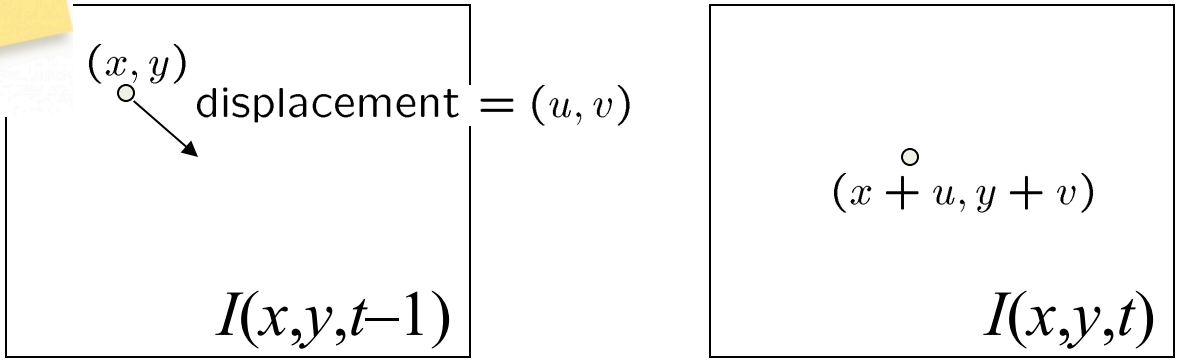
$I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field u(x,y), v(x,y) between them
- Key assumptions
  - **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

Source: Silvio Savarese

technical note

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:

Image derivative along x

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

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

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

Source: Silvio Savarese

The brightness constancy constraint

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

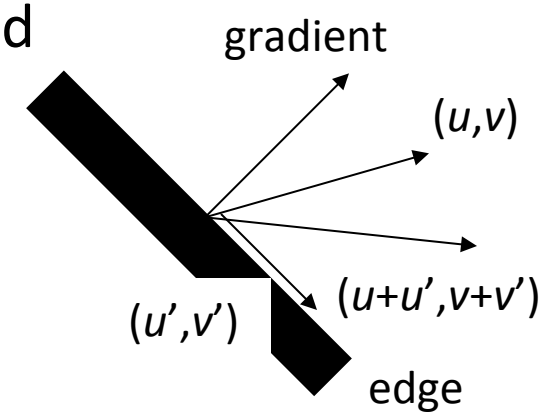
$$\nabla I \cdot \begin{bmatrix} u & v \end{bmatrix}^T + I_t = 0$$

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

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

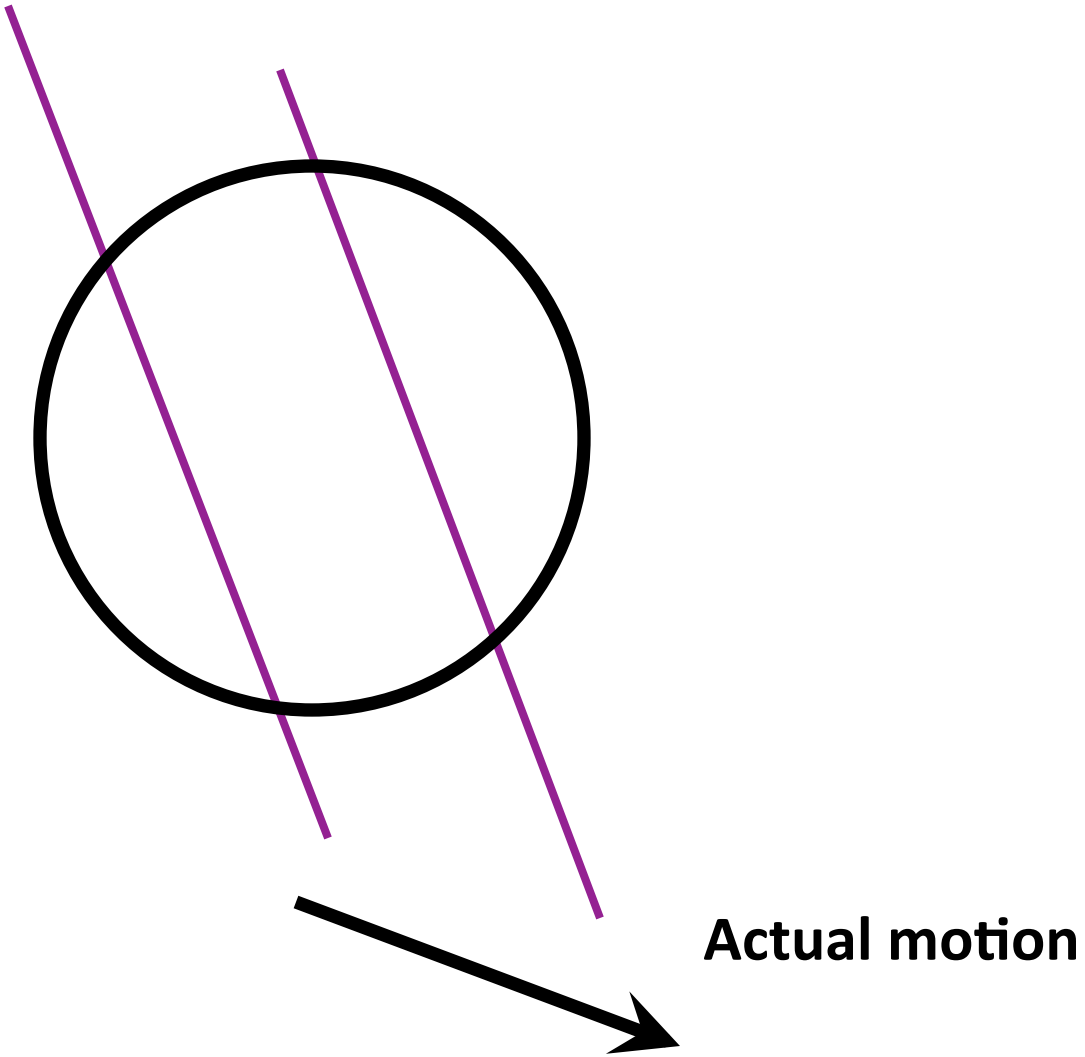
If (*u*, *v*) satisfies the equation, so does (*u+u'*, *v+v'*) if

$$\nabla I \cdot \begin{bmatrix} u' & v' \end{bmatrix}^T = 0$$



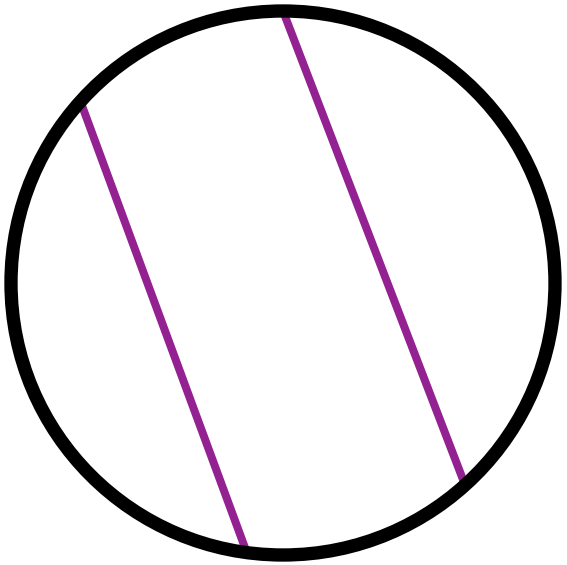
Source: Silvio Savarese

The aperture problem



Source: Silvio Savarese

The aperture problem



Perceived motion

Source: Silvio Savarese

The barber pole illusion

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

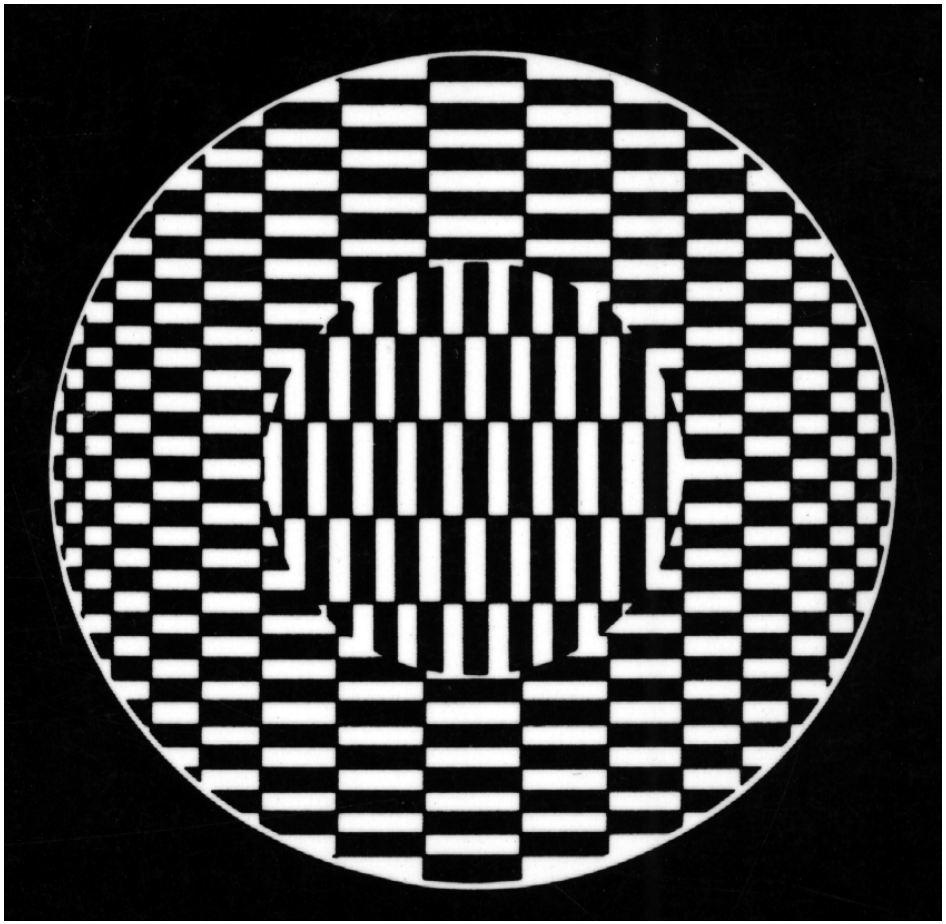
Source: Silvio Savarese

The barber pole illusion

Source: Silvio Savarese

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

Aperture problem cont'd



* From Marc Pollefeys COMP 256 2003

Solving the ambiguity…

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.

- 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

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

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

Source: Silvio Savarese

Lucas-Kanade flow

- Overconstrained linear system:

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

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Source: Silvio Savarese

Conditions for solvability

- When is this system solvable?
  - What if the window contains just a single straight edge?

Source: Silvio Savarese

Lucas-Kanade flow

- Overconstrained linear system

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

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

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

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

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

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

Source: Silvio Savarese

Conditions for solvability

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

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

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

When is This Solvable?

- **$A^T A$** should be invertible
- **$A^T A$** should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of **$A^T A$** should not be too small
- **$A^T A$** should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Does this remind anything to you?

Source: Silvio Savarese

$M = A^T A$ is the *second moment matrix* !

(Harris corner detector…)

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

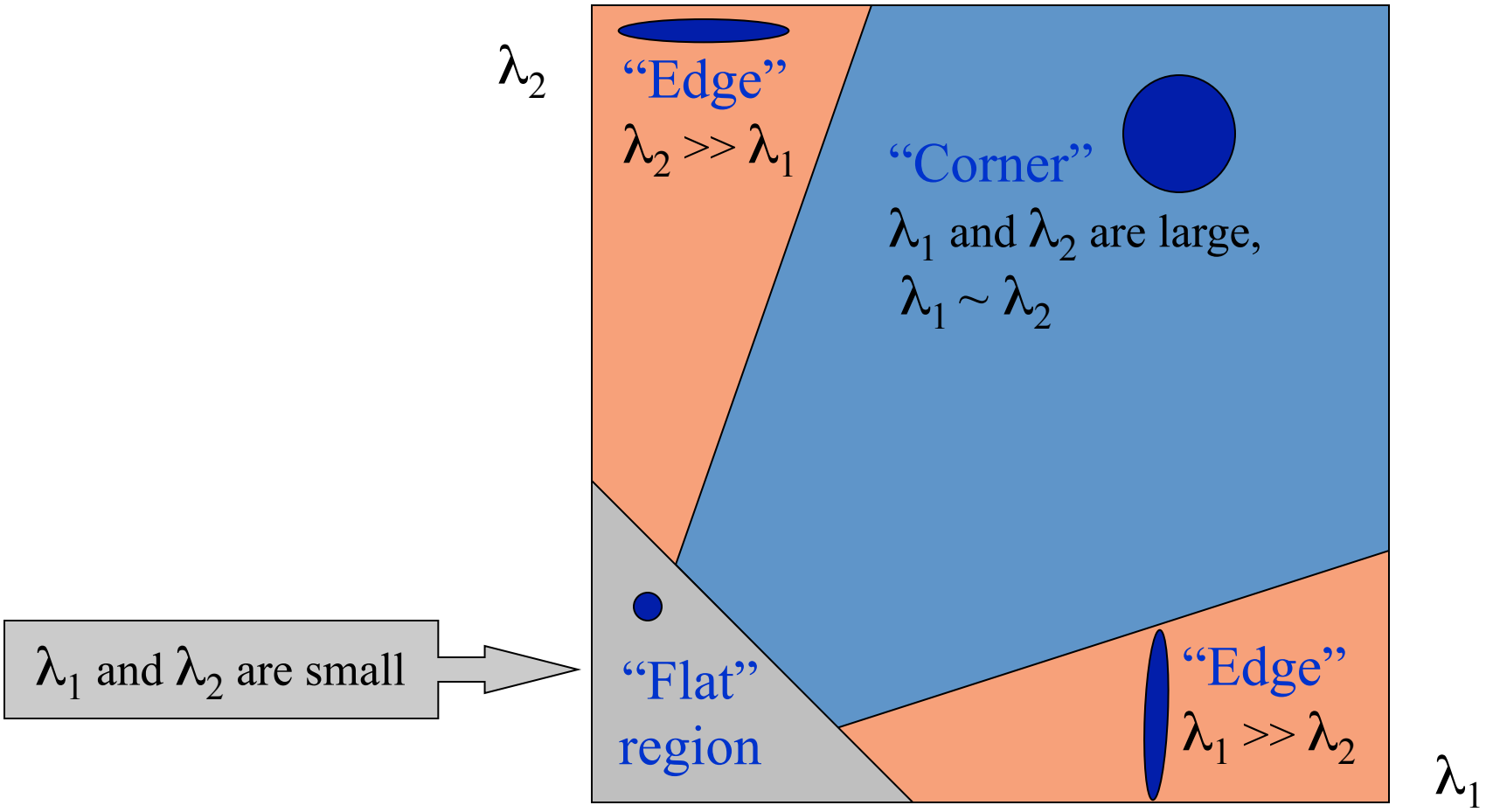
- Eigenvectors and eigenvalues of $A^T A$ relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it

Source: Silvio Savarese

technical note

Interpreting the eigenvalues

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



Source: Silvio Savarese

technical note

Edge

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

– gradients very large or very small

– large λ_1, small λ_2

Source: Silvio Savarese

technical note

Low-texture region

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

– gradients have small magnitude

– small λ_1, small λ_2

Source: Silvio Savarese

technical note

High-texture region

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

– gradients are different, large magnitudes

– large λ_1, large λ_2

Source: Silvio Savarese

What we will learn today?

- Introduction
- Optical flow
- Feature tracking
- Applications
- (Supplementary) Technical note

What are good features to track?

- Can measure "quality" of features from just a single image
- Hence: tracking Harris corners (or equivalent) guarantees small error sensitivity!

→ Implemented in Open CV

Source: Silvio Savarese

Recap

- 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

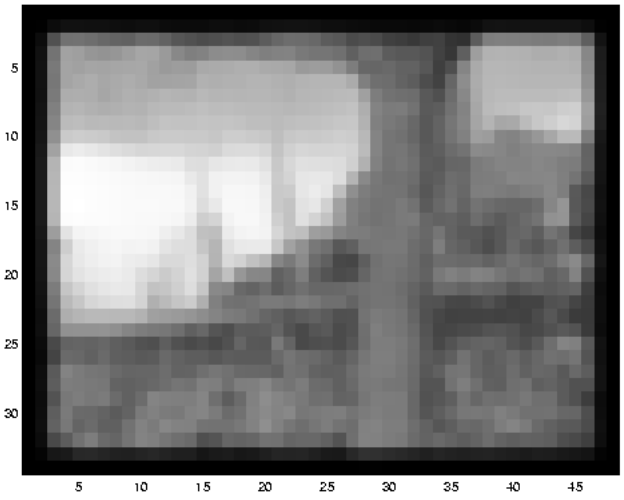
Source: Silvio Savarese

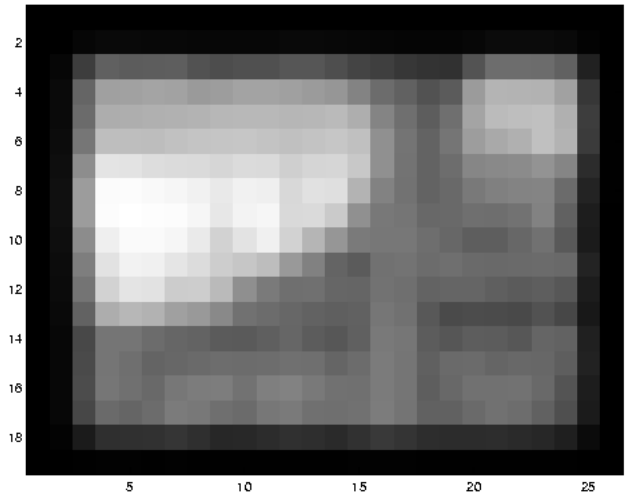
Revisiting the small motion assumption

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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

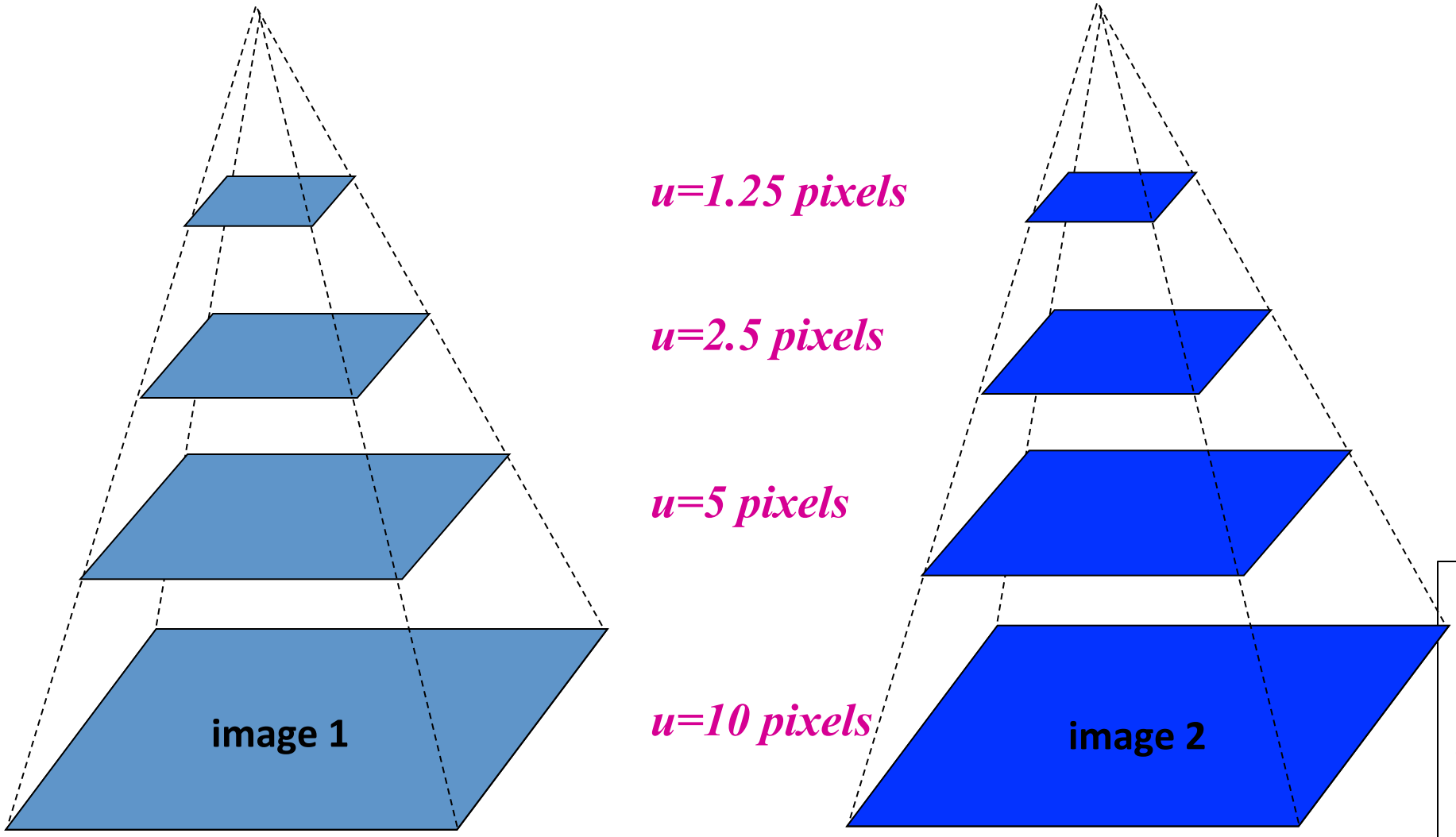
Reduce the resolution!





* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Coarse-to-fine optical flow estimation

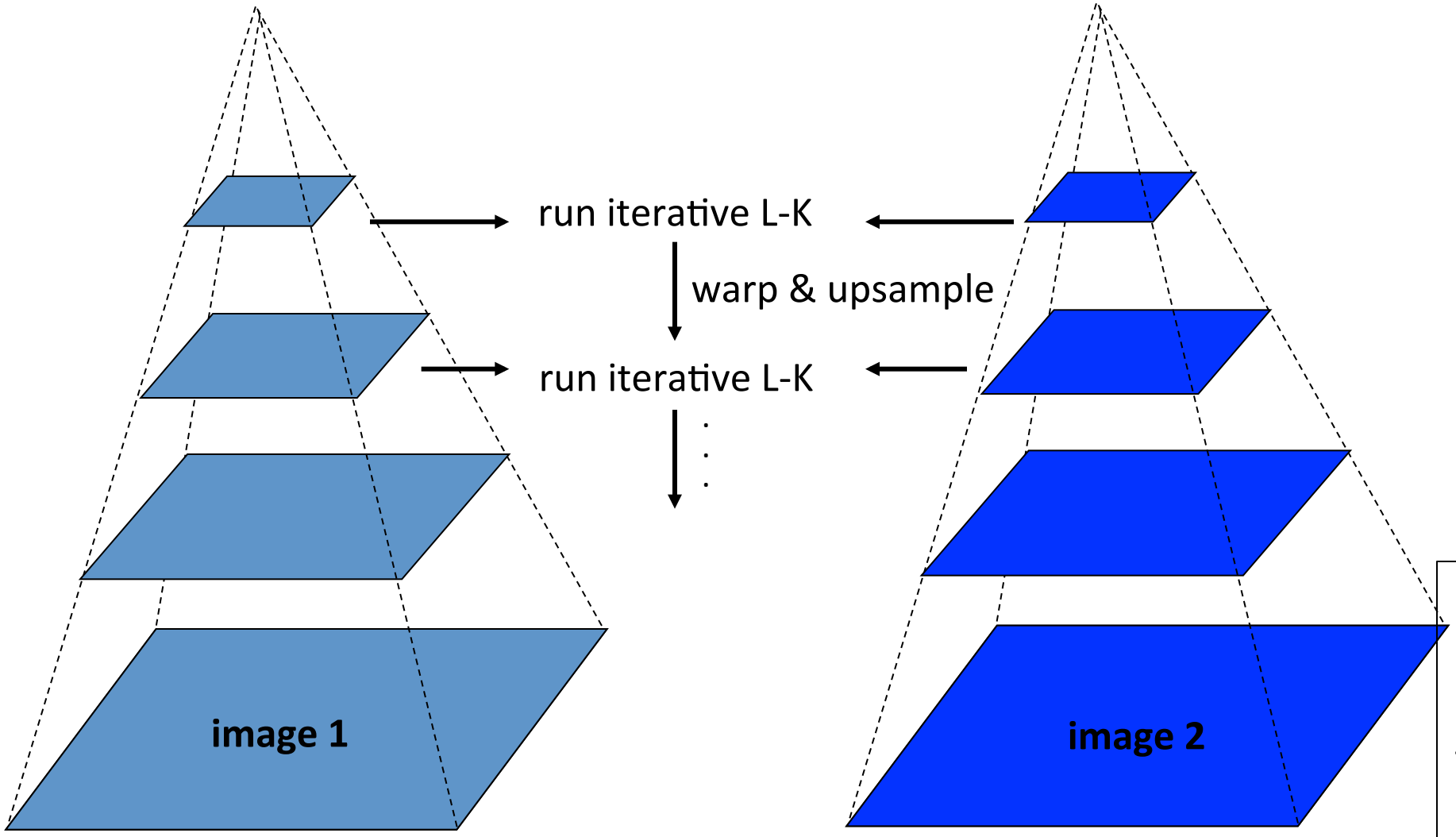


Gaussian pyramid of image 1

Gaussian pyramid of image 2

Source: Silvio Savarese

Coarse-to-fine optical flow estimation

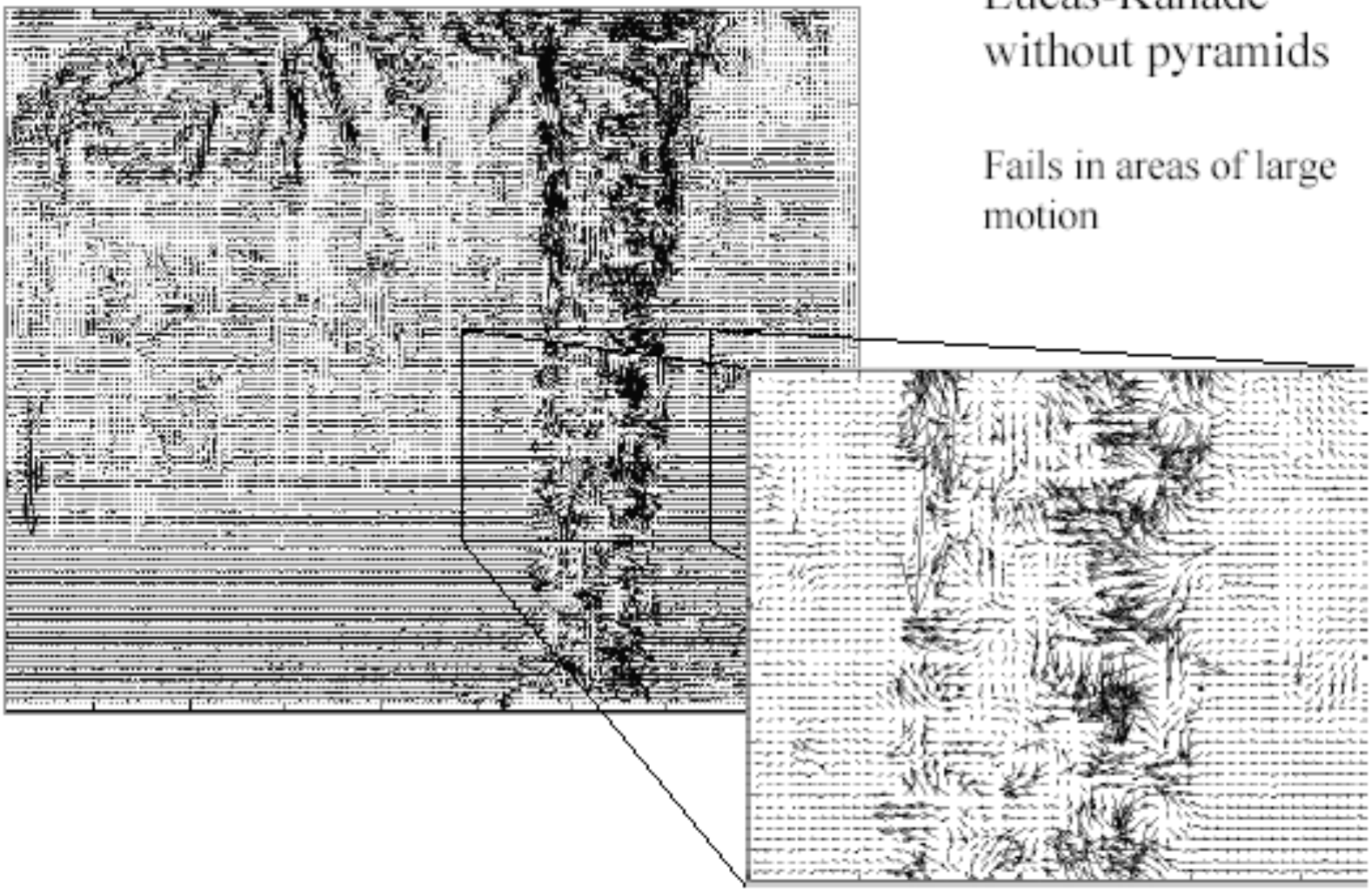


Gaussian pyramid of image 1 (t)

Gaussian pyramid of image 2 (t+1)

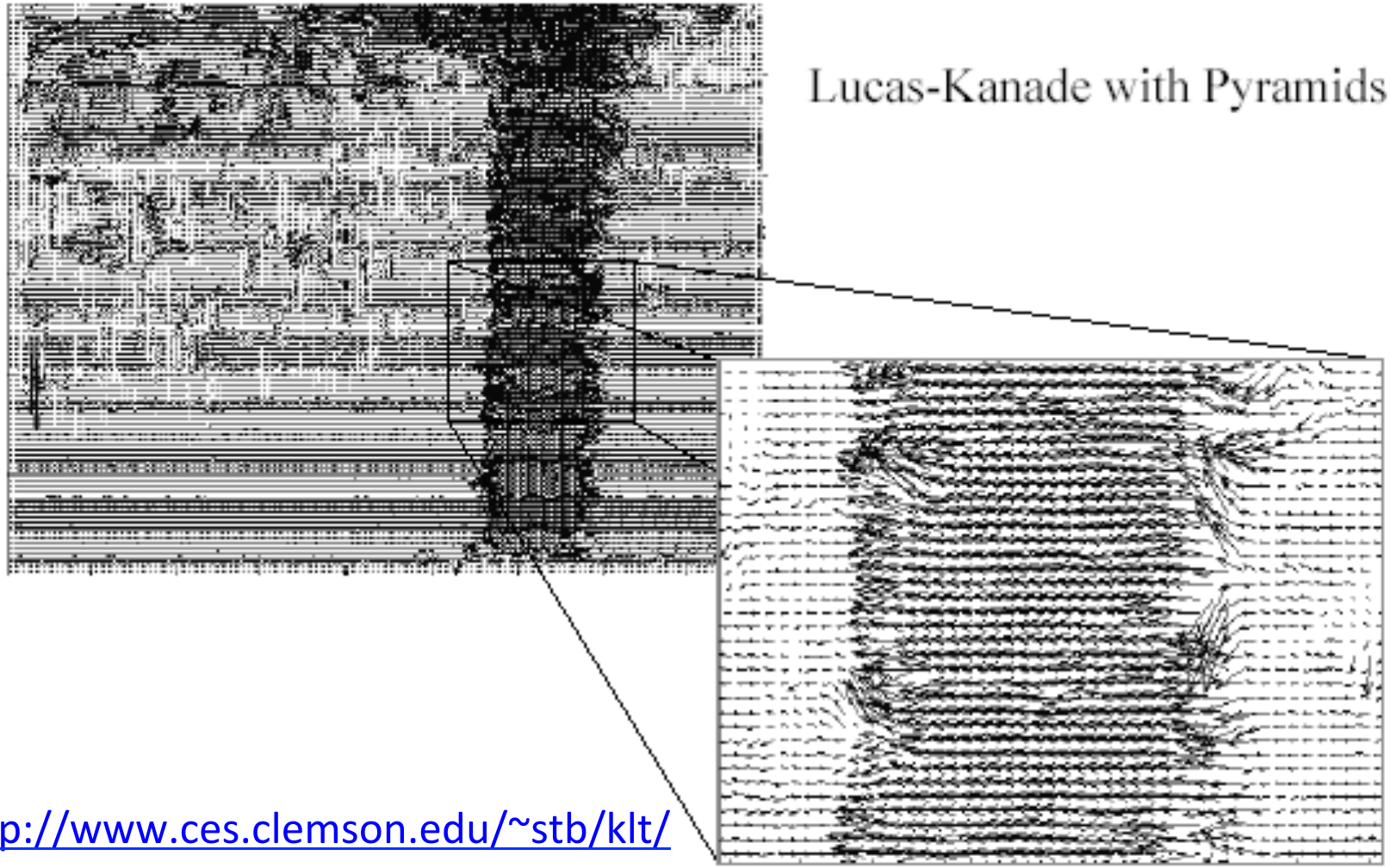
Source: Silvio Savarese

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Recap

- 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

Source: Silvio Savarese

Reminder: Gestalt – common fate

Common Fate

Motion segmentation

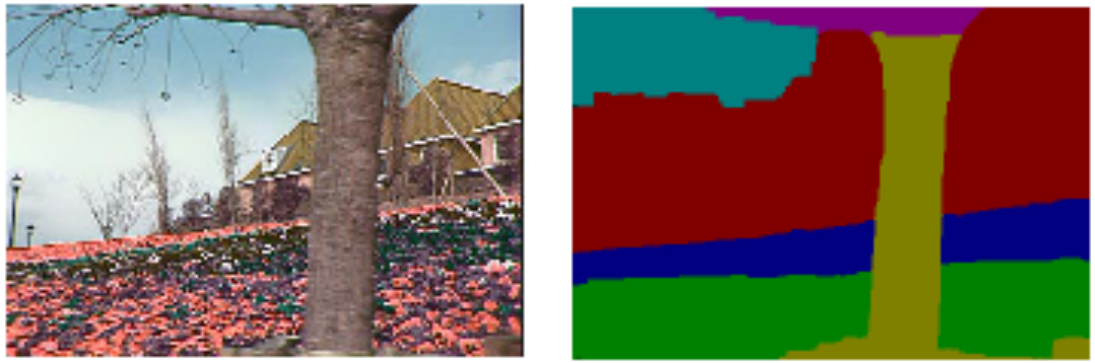
- How do we represent the motion in this scene?

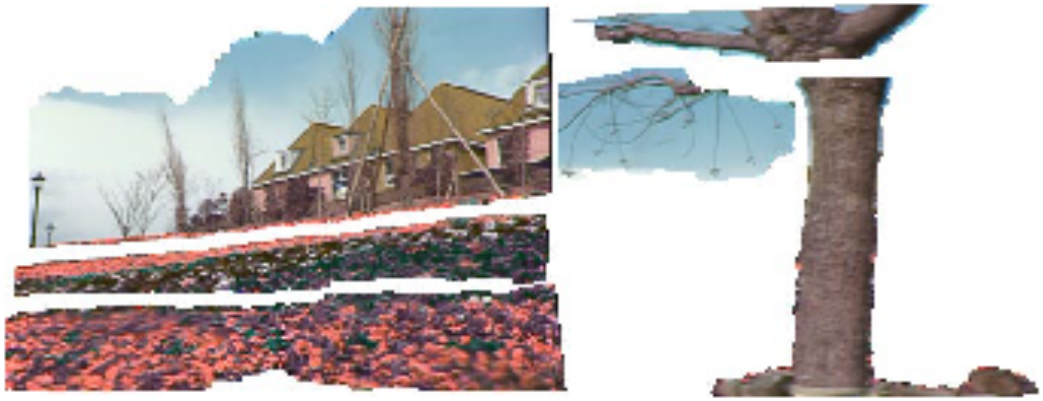
Source: Silvio Savarese

Motion segmentation

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

- Break image sequence into "layers" each of which has a coherent (affine) motion





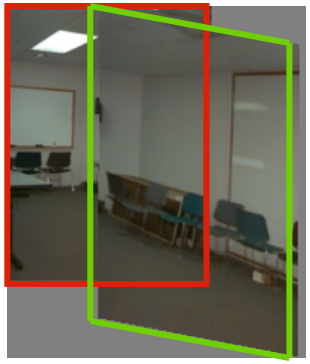
Source: Silvio Savarese

technical note

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:

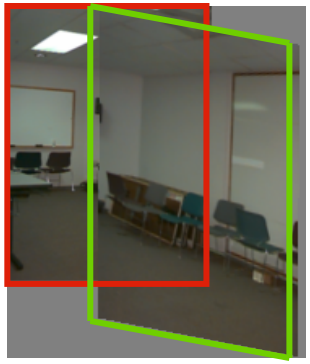
$$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

Source: Silvio Savarese

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:

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

- Each pixel provides 1 linear constraint in 6 unknowns
- Least squares minimization:

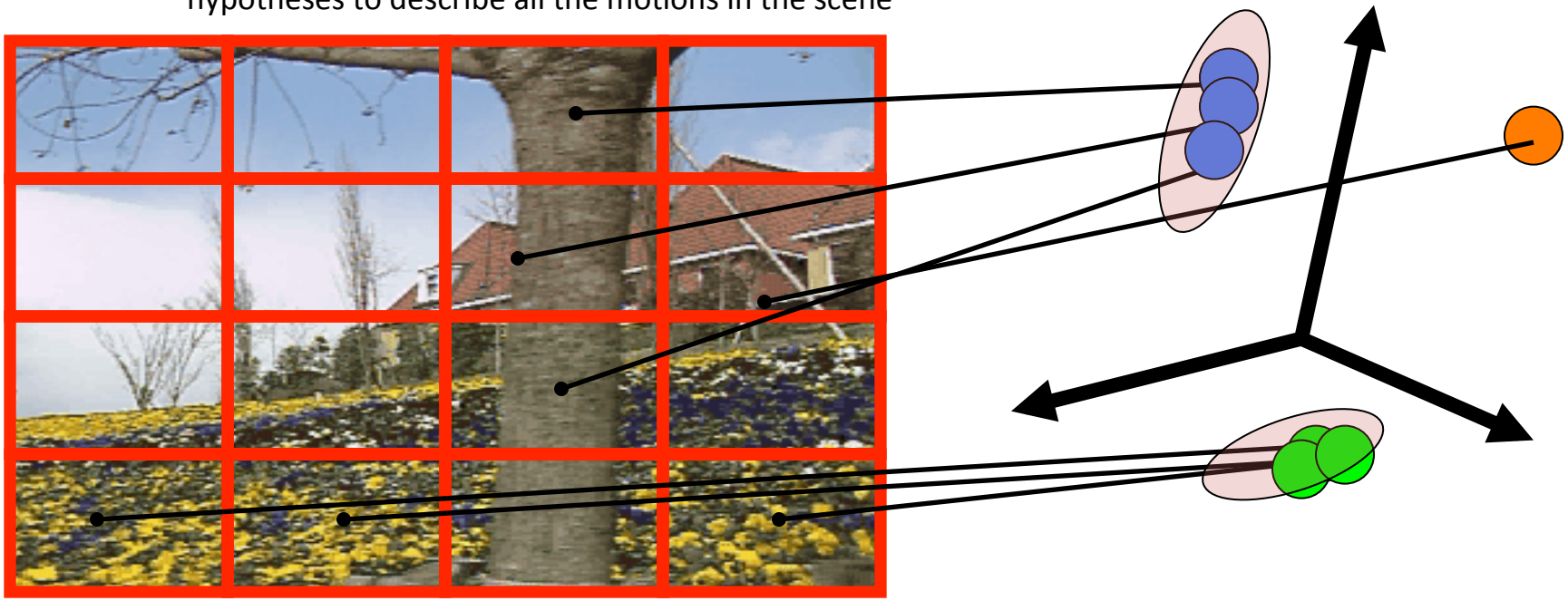
$$Err(\vec{a}) = \sum \left[I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \right]^2$$

Source: Silvio Savarese

ow do we estimate the layers?

technical note

- 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

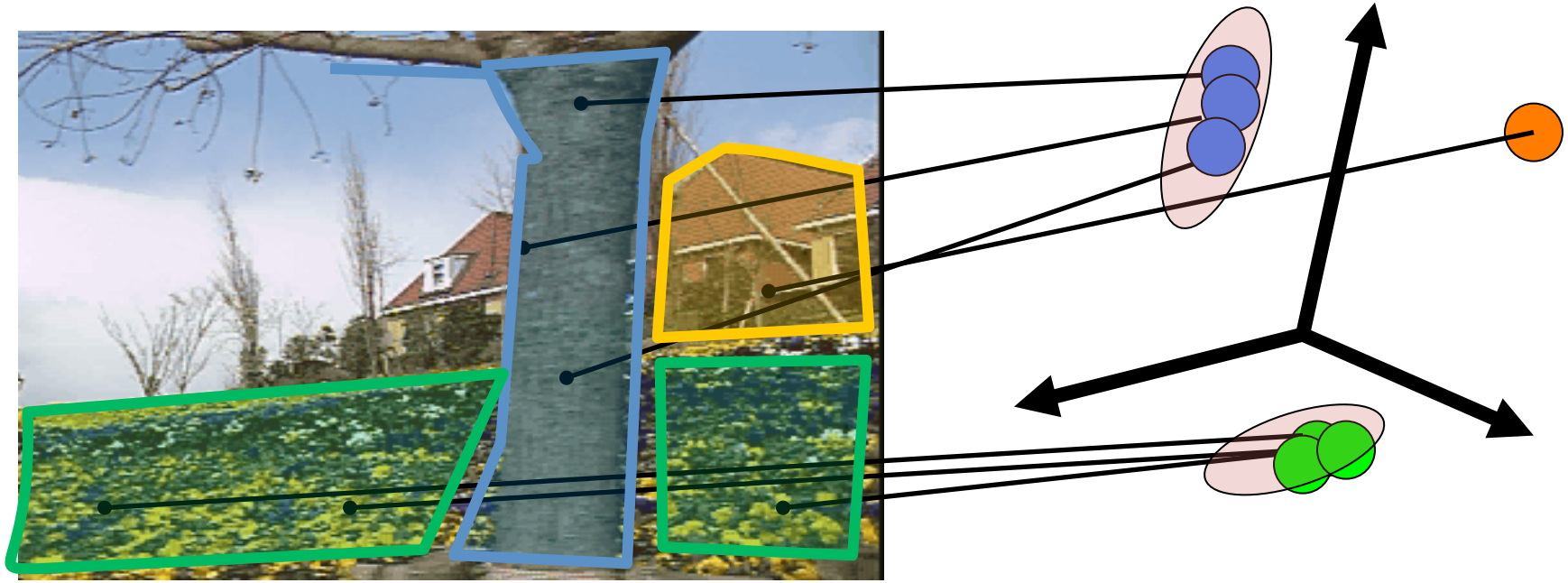


Source: Silvio Savarese

technical note

ow 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



Source: Silvio Savarese

ow do we estimate the layers?

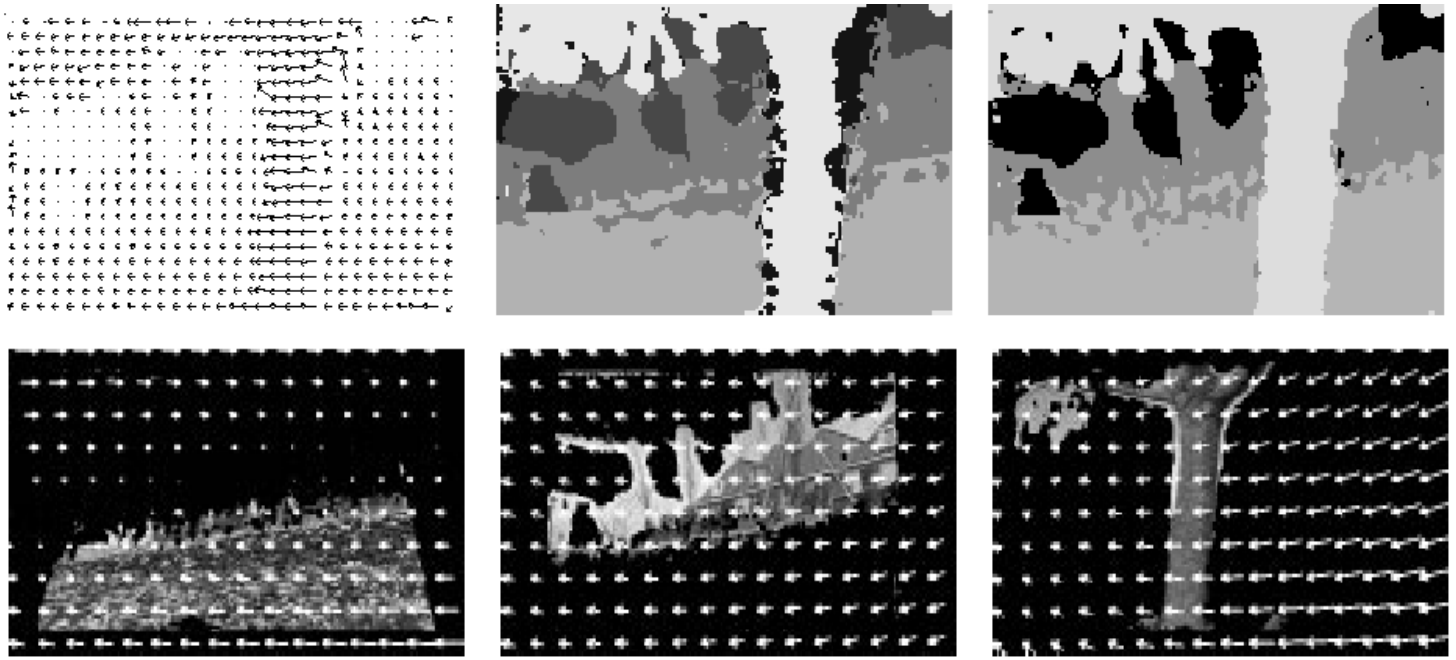
- 1. Obtain a set of initial affine motion hypotheses
  - Divide the image into blocks and estimate affine motion parameters in each block by least squares
    - Eliminate hypotheses with high residual error
  - Map into motion parameter space
  - Perform k-means clustering on affine motion parameters
    - 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

2. Iterate until convergence:

- Assign each pixel to best hypothesis
  - Pixels with high residual error remain unassigned
- Perform region filtering to enforce spatial constraints
- Re-estimate affine motions in each region

Source: Silvio Savarese

Example result



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

Source: Silvio Savarese

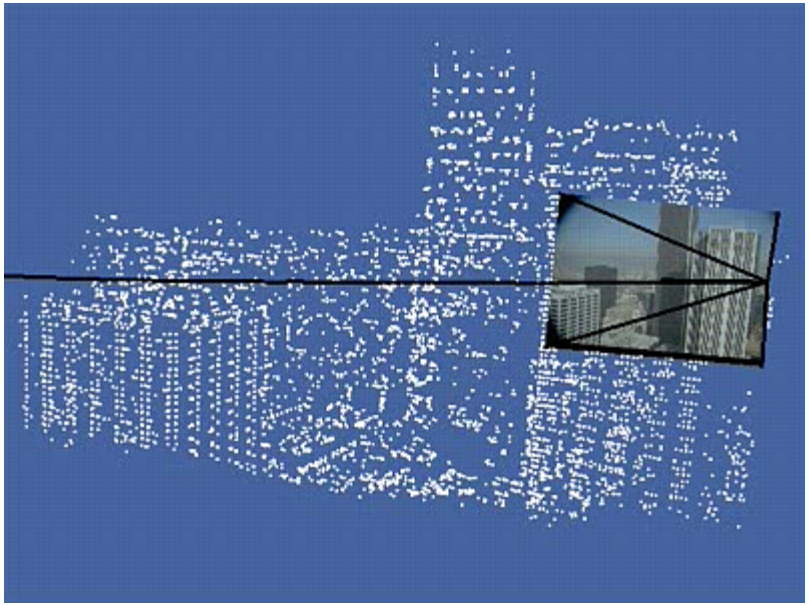
What we will learn today?

- Introduction
- Optical flow
- Feature tracking
- Applications
- (Supplementary) Technical note

Uses of motion

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

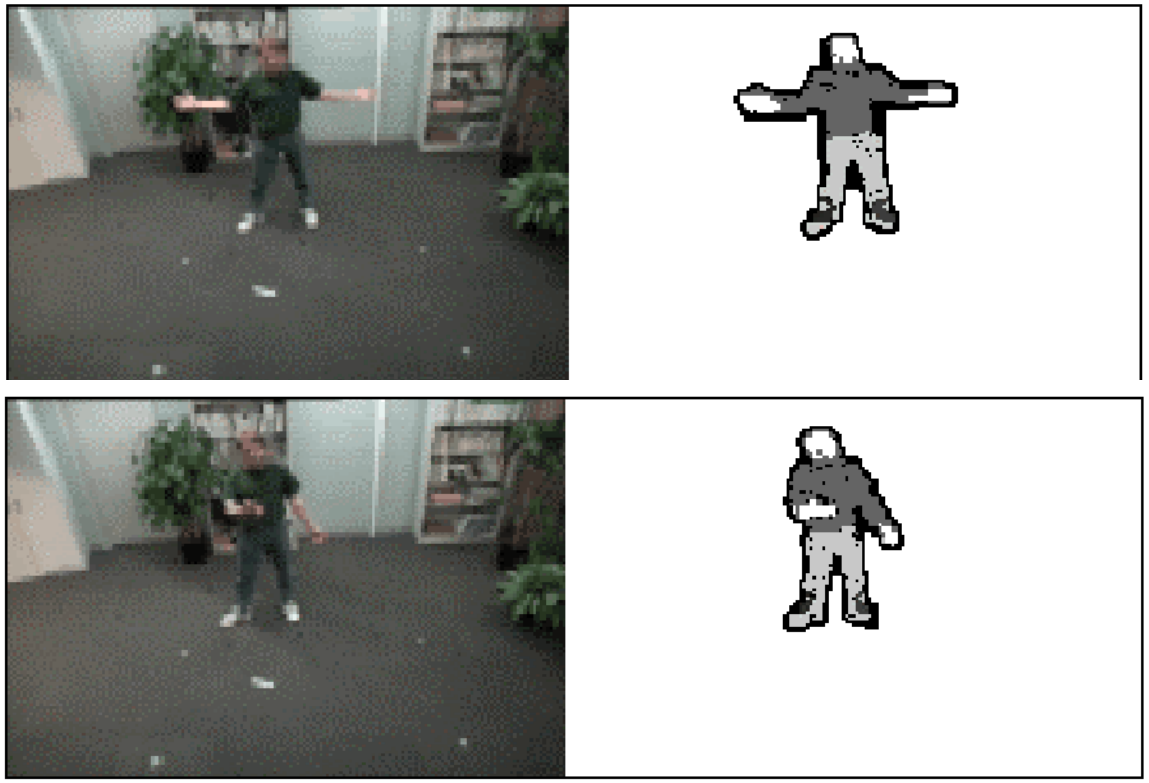
Estimating 3D structure



Source: Silvio Savarese

Segmenting objects based on motion cues

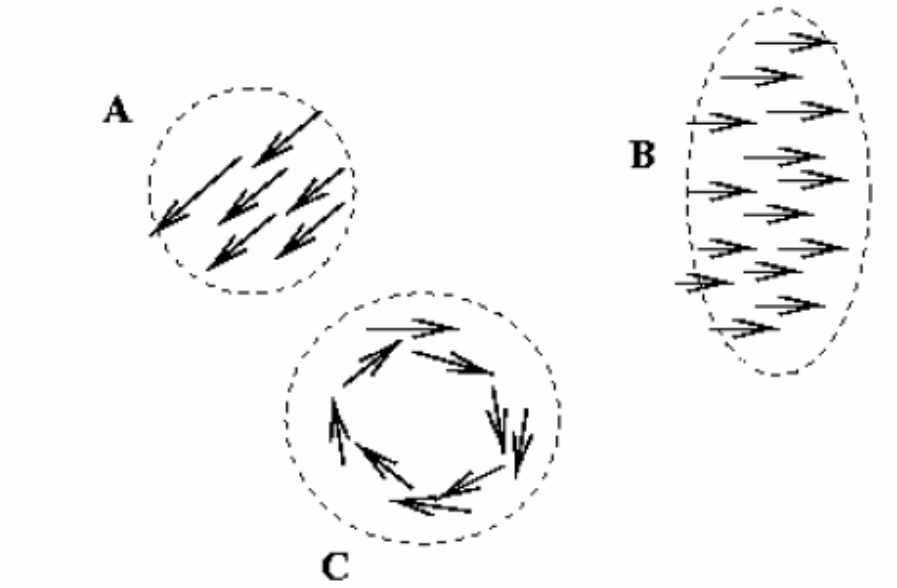
- Background subtraction
  - A static camera is observing a scene
  - Goal: separate the static *background* from the moving *foreground*

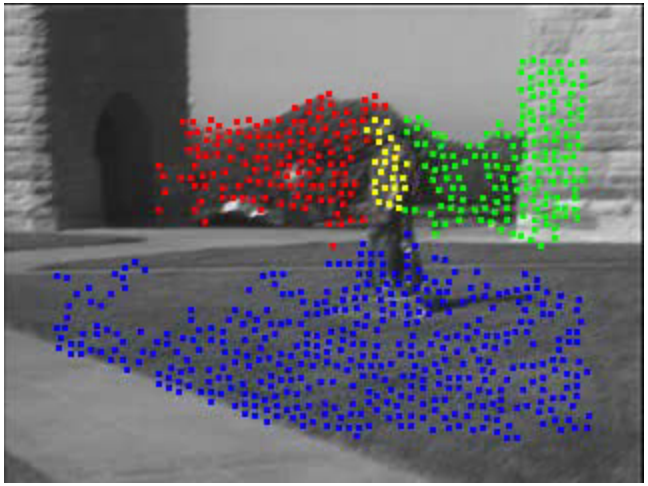


Source: Silvio Savarese

Segmenting objects based on motion cues

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

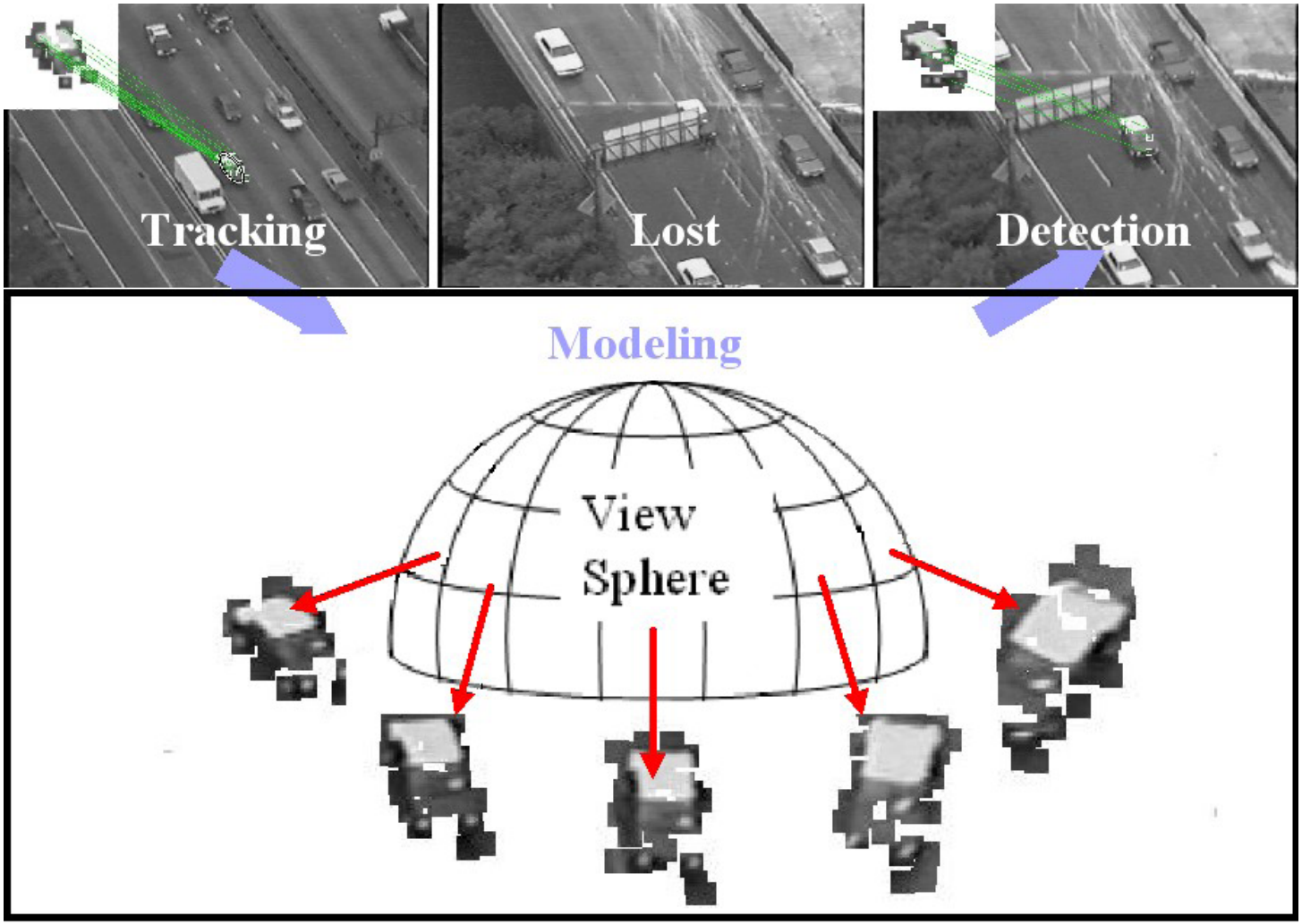




S. J. Pundlik and S. T. Birchfield, Motion Segmentation at Any Speed, Proceedings of the British Machine Vision Conference (BMVC) 2006

Source: Silvio Savarese

Tracking objects



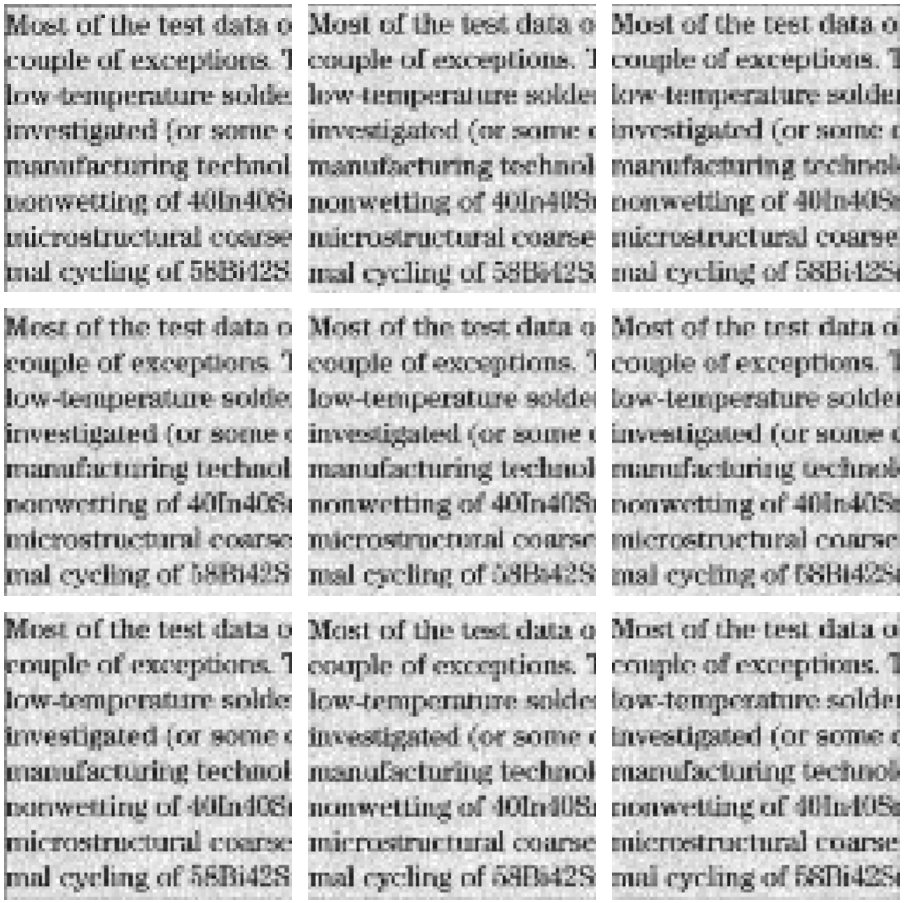
Z.Yin and R.Collins, "On-the-fly Object Modeling while Tracking," *IEEE Computer Vision and Pattern Recognition (CVPR '07),* Minneapolis, MN, June 2007.

Source: Silvio Savarese

Synthesizing dynamic textures

Super-resolution

Example: A set of low quality images



Source: Silvio Savarese

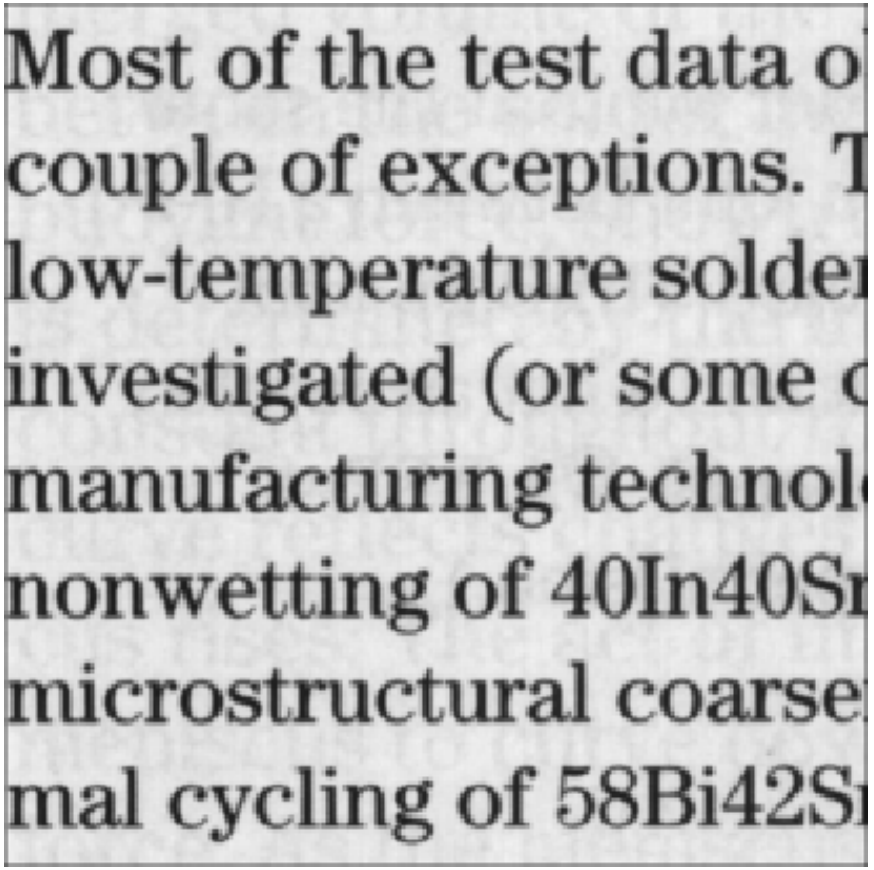
Super-resolution

Each of these images looks like this:

Source: Silvio Savarese

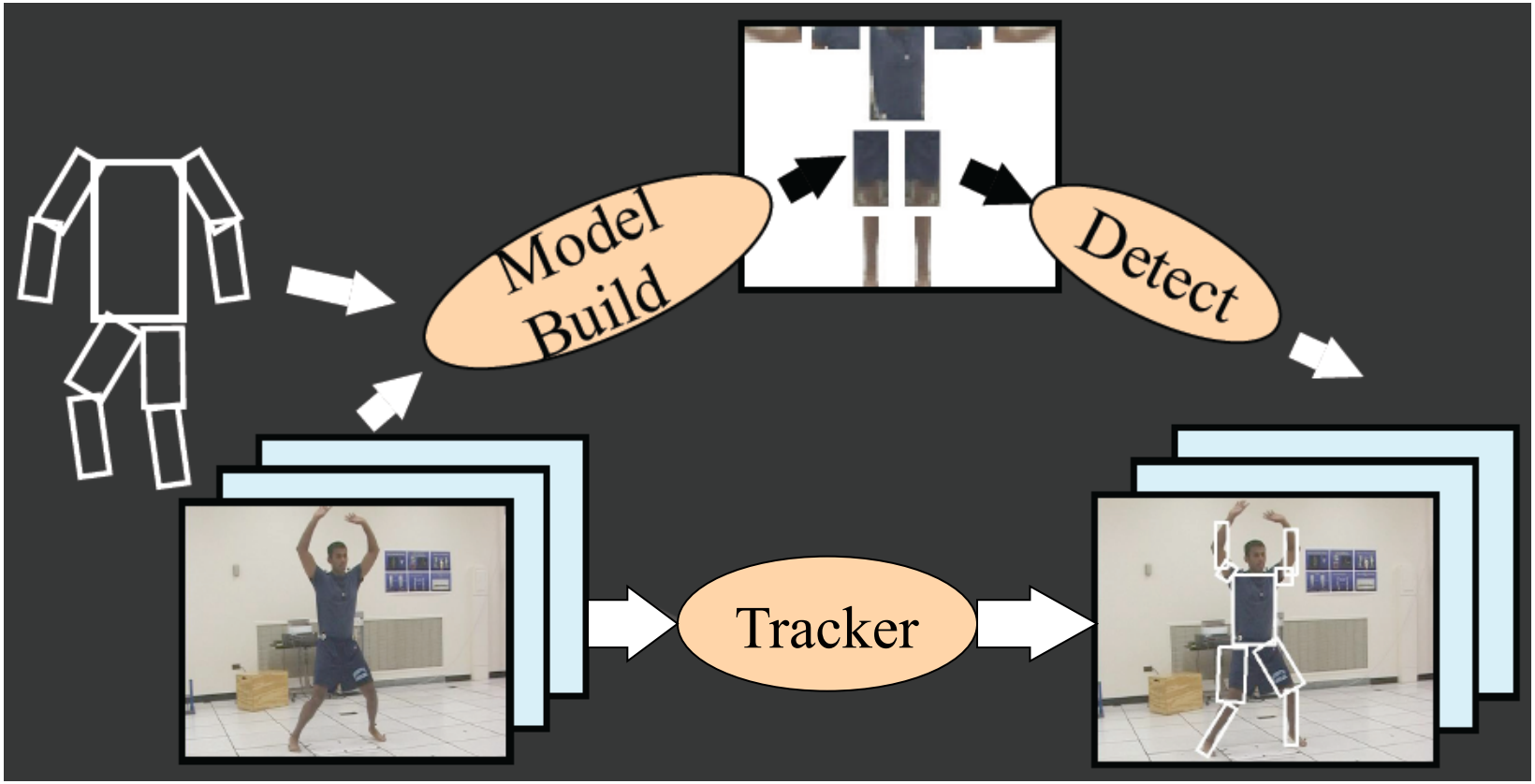
Super-resolution

The recovery result:



Source: Silvio Savarese

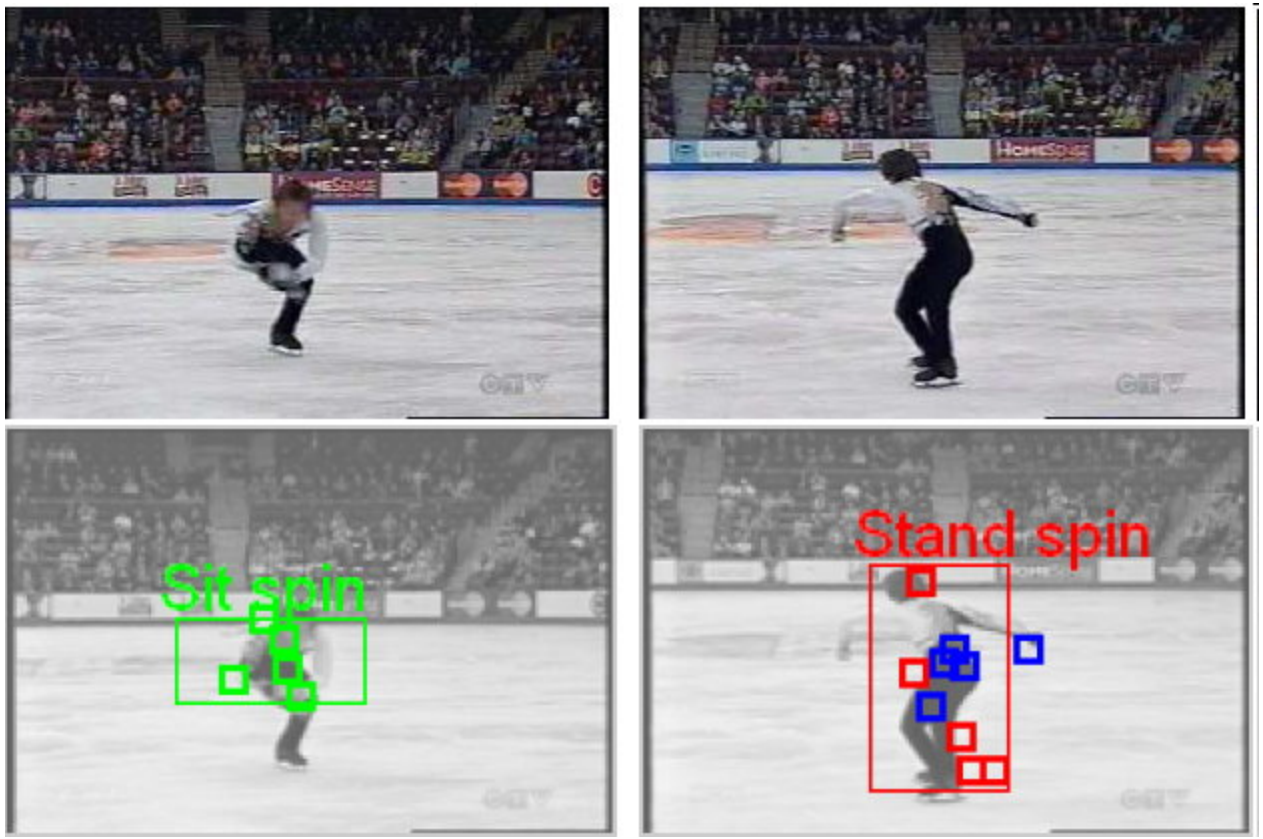
Recognizing events and activities



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

Source: Silvio Savarese

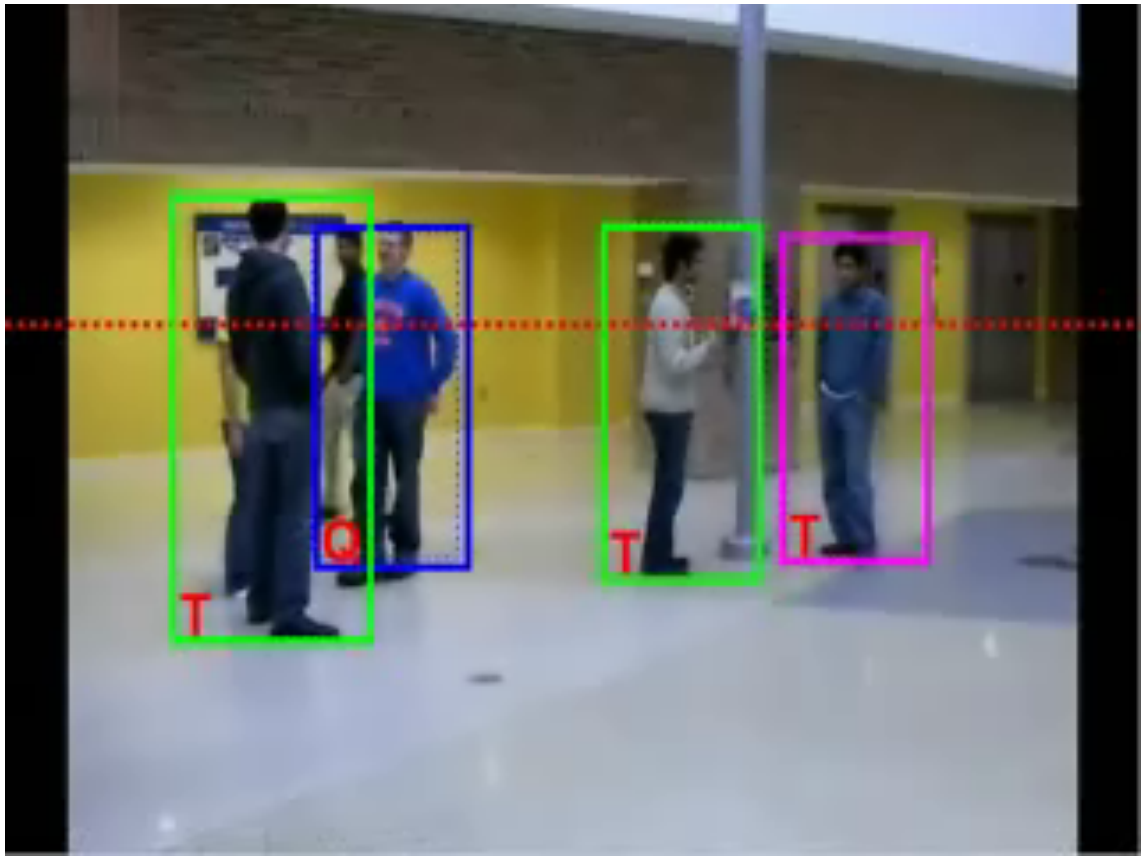
Recognizing events and activities



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

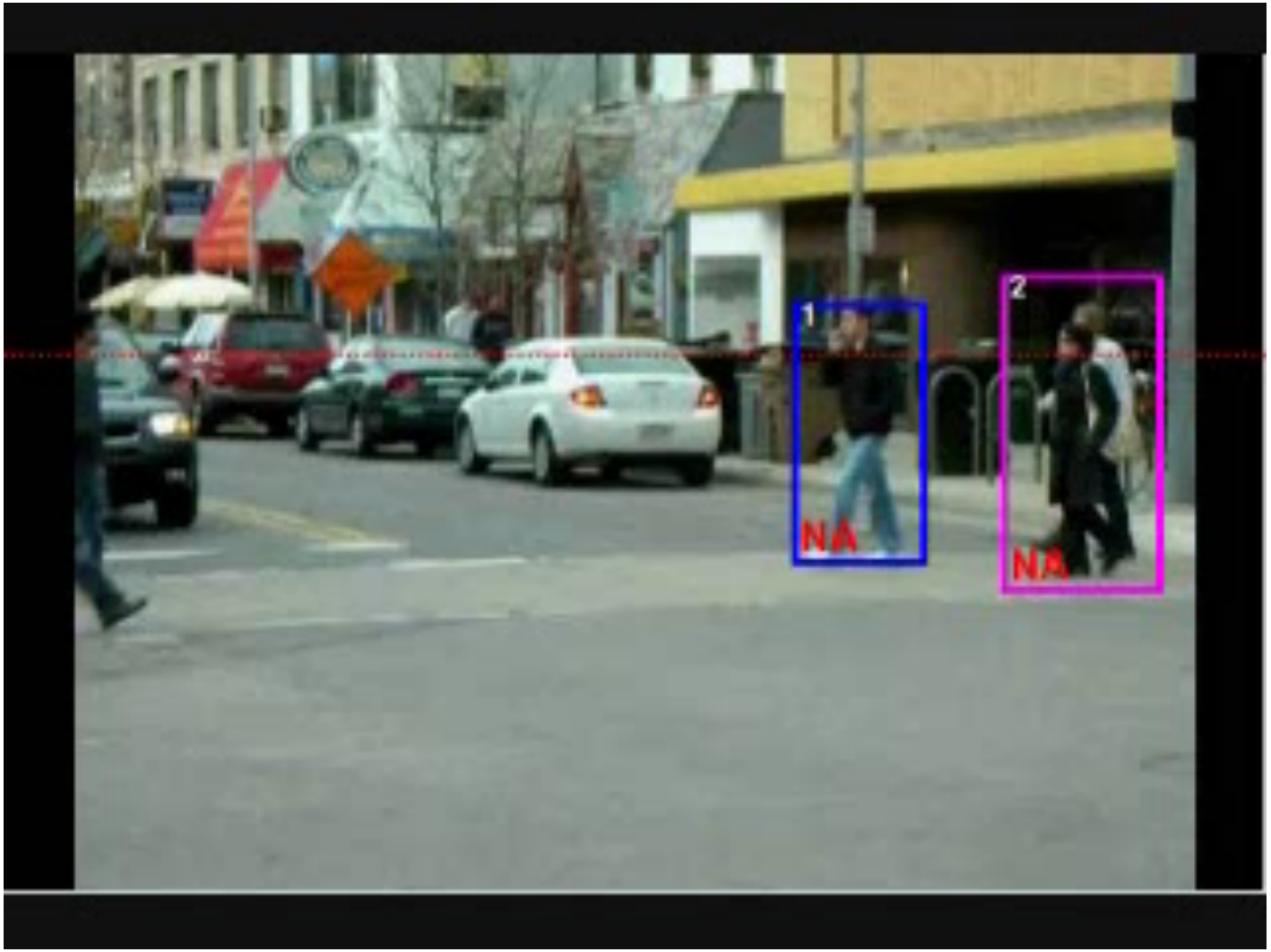
Recognizing events and activities

Crossing – Talking – Queuing – Dancing – jogging



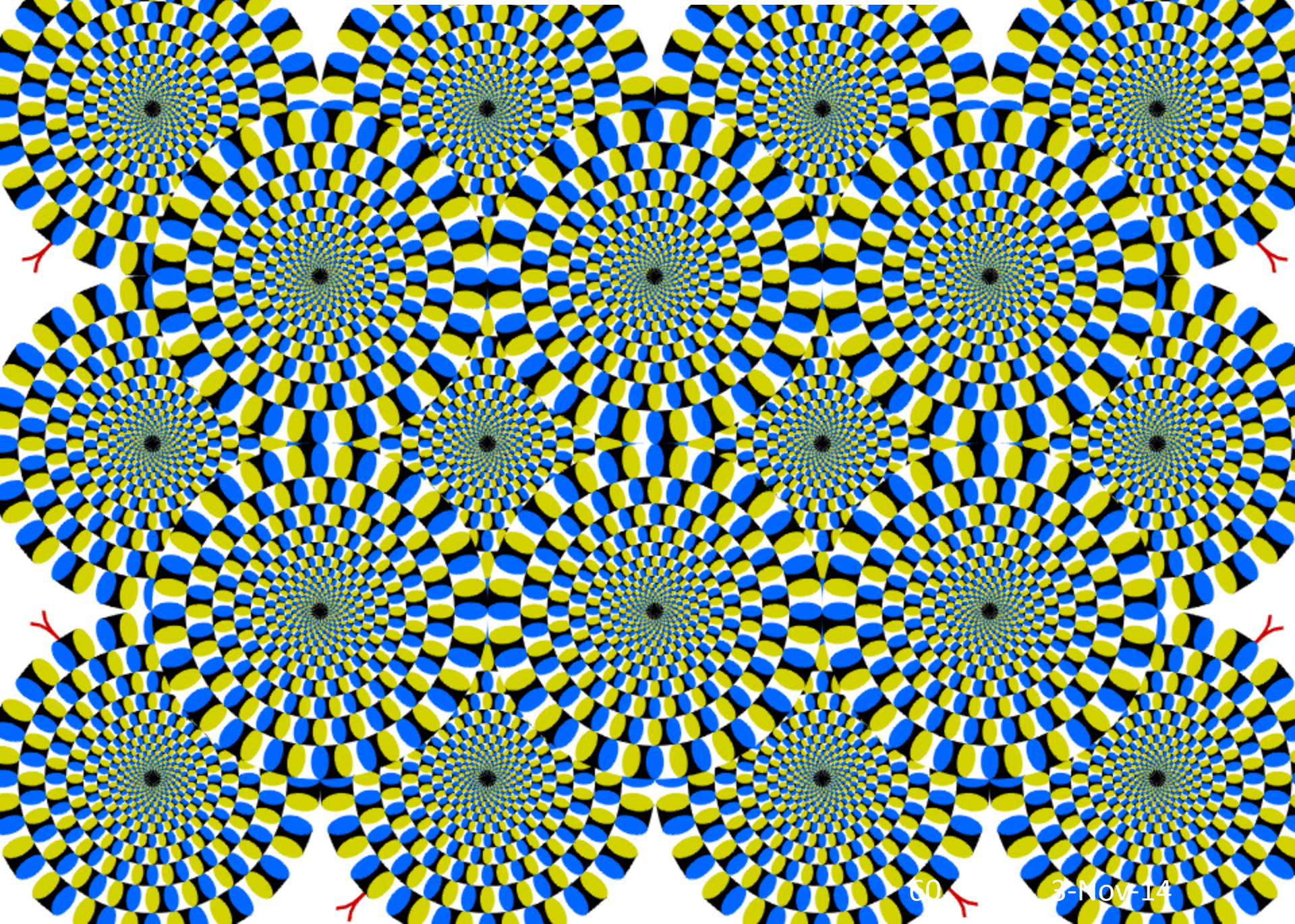
W. Choi & K. Shahid & S. Savarese WMC 2010

Source: Silvio Savarese



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

Optical flow without motion!



What we have learned today?

- Introduction
- Optical flow
- Feature tracking
- Applications
- (Supplementary) Technical note

Reading: [Szeliski] Chapters: 8.4, 8.5

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

Optical Flow Estimation

Goal: Introduction to image motion and 2D optical flow estimation.

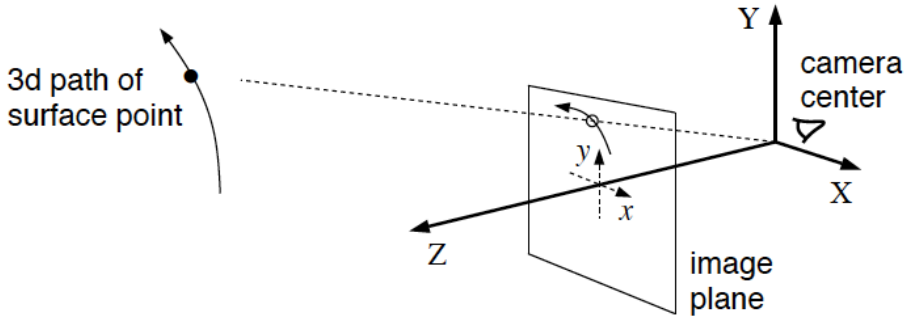
Motivation:

- Motion is a rich source of information about the world:
  - segmentation
  - surface structure from parallax
  - self-motion
  - recognition
  - understanding behavior
  - understanding scene dynamics
- Other correspondence / registration problems:
  - stereo disparity (short and wide baseline)
  - computer-assisted surgery
  - multiview alignment for mosaicing or stop-frame animation

Fleet & Weiss, 2005

technical note

Introduction to Image Motion



A 3D point $\vec{\mathbf{X}}$ follows a space-time path $\vec{\mathbf{X}}(t)$. Its velocity is $\vec{\mathbf{V}}$ is

$$\vec{\mathbf{V}} = \frac{d\vec{\mathbf{X}}(t)}{dt} = \left(\frac{dX(t)}{dt}, \frac{dY(t)}{dt}, \frac{dZ(t)}{dt}\right)^T .$$

Perspective projection (for nodal length f) of the 3D path onto the image plane produces a 2D path,

$$\vec{\mathbf{x}}(t) = (x(t), y(t)) = \left(\frac{fX(t)}{Z(t)}, \frac{fY(t)}{Z(t)}\right) ,$$

the instantaneous 2D velocity of which is

$$\begin{aligned}\vec{\mathbf{u}} &= \left(\frac{dx(t)}{dt}, \frac{dy(t)}{dt}\right)^T \\ &= \frac{f}{Z(t)}\left(\frac{dX(t)}{dt}, \frac{dY(t)}{dt}\right)^T - \frac{f}{Z^2(t)}\frac{dZ(t)}{dt}(X(t), Y(t))^T .\end{aligned}$$

Fleet & Weiss, 2005

Definition: **2D Motion Field** – 2D velocities for all visible points.
Optical Flow Field – Estimate of the 2D motion field.

Optical Flow

technical note

Two Key Problems:

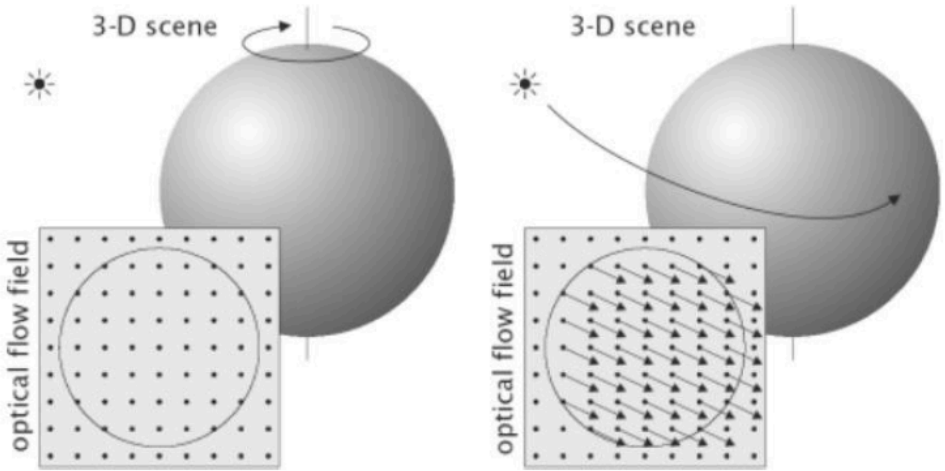
1. Determine what image property to track.
2. Determine how to track it

Brightness constancy: More precisely, let's track points of constant brightness, assuming that surface radiance is constant over time:

$$I(x, y, t+1) \;=\; I(x - u_1, y - u_2, t)\;.$$

Brightness constancy is often assumed by researchers, and often violated by Mother Nature; so the resulting optical flow field is sometimes a very poor approximation to the 2D motion field.

For example, a rotating Lambertian sphere with a static light source produces a static image. But a stationary sphere with a moving light source produces drifting intensities (figure from Jahne et al, 1999).



Fleet & Weiss, 2005

technical note

Gradient-Based Motion Estimation

Let $f(x)$ denote a 1D greylevel function of spatial position, and assume $f(x)$ is just translated by d between time 1 and time 2:

$$f_2(x) = f_1(x - d) \; .$$

We can express the shifted signal as a Taylor expansion of f_1 about x:

$$f_1(x - d) \;=\; f_1(x) - d\, f_1'(x) + O(d^2 f_1'') \; ,$$

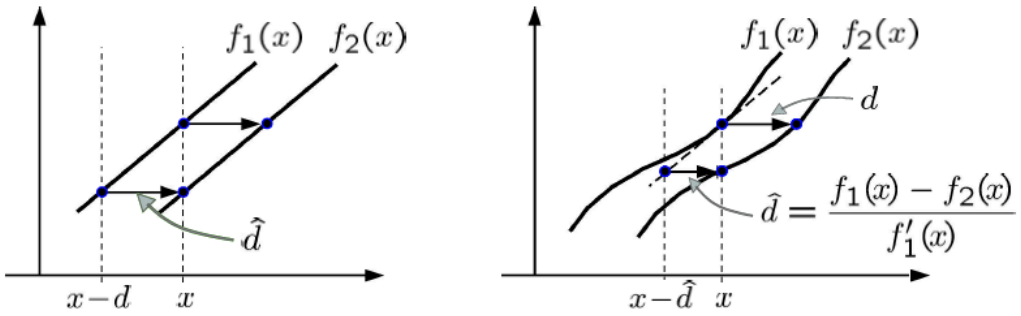
in which case the difference between the two signals is given by

$$f_2(x) - f_1(x) \;=\; -\, d\, f_1'(x) + O(d^2 f_1'') \; .$$

Then, a first-order approximation to the displacement is

$$\hat{d} \;=\; \frac{f_1(x) - f_2(x)}{f_1'(x)} \; .$$

For linear signals the first-order estimate is exact.



For nonlinear signals, the accuracy of the approximation depends on the displacement magnitude and the higher-order signal structure.

Fleet & Weiss, 2005

technical note

Gradient Constraint Equation

In two spatial dimensions, the first-order approximation is:

$$f(x+u_1, y+u_2, t+1) \approx$$
$$f(x,y,t) + u_1 f_x(x,y,t) + u_2 f_y(x,y,t) + f_t(x,y,t) \quad (1)$$

Taking the image difference with times t and $t+1$, then yields:

$$u_1\, f_x(x,y,t) + u_2\, f_y(x,y,t) + f_t(x,y,t) \;=\; 0\,.$$

Written in vector form, with $\vec{\nabla} f \equiv (f_x,\ f_y)^T$:

$$\vec{\mathbf{u}}^T \vec{\nabla} f(x,y,t) + f_t(x,y,t) \;=\; 0\,.$$

When the duration between frames is large, it is sometimes more appropriate to use only spatial derivatives in the Taylor series approximation in (1). Then one obtains a different approximation

$$\vec{\mathbf{u}}^T \vec{\nabla} f(x,y,t) + \Delta f(x,y,t) \;=\; 0$$

where $\Delta f(x,y,t) \;=\; f(x,y,t+1) - f(x,y,t)$.

Fleet & Weiss, 2005

Brightness Conservation

technical note

One can also derive the gradient constraint equation directly from brightness conservation.

Let $(x(t), y(t))$ denote a space-time path along which the image intensity remains is constant; i.e., the time-varying image f satisfies

$$f(x(t), y(t), t) = c$$

Taking the total derivative of both sides gives

$$\frac{d}{dt} f(x(t), y(t), t) = 0$$

The total derivative is given in terms of its partial derivatives

$$\begin{aligned} \frac{d}{dt} f(x(t), y(t), t) &= \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial t}\frac{dt}{dt} \\ &= f_x u_1 + f_y u_2 + f_t \\ &= \vec{\mathbf{u}}^T \vec{\nabla} f + f_t \\ &= 0 \end{aligned}$$

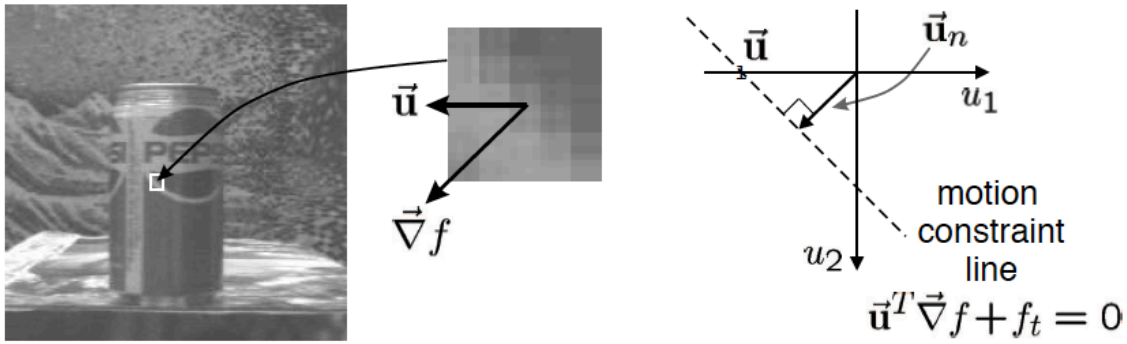
This derivation assumes that there is no aliasing, so that one can in principle reconstruct the continuous underlying signal and its derivatives in space and time. There are many situations in which this is a reasonable assumption. But with common video cameras temporal aliasing is often a problem with many video sequences, as we discuss later in these notes.

Fleet & Weiss, 2005

technical note

Normal Velocity

The gradient constraint provides one constraint in two unknowns. It defines a line in velocity space:



The gradient constrains the velocity in the direction normal to the local image orientation, but does not constrain the tangential velocity. That is, it uniquely determines only the normal velocity:

$$\vec{\mathbf{u}}_n = \frac{-f_t}{||\vec{\nabla} f||} \frac{\vec{\nabla} f}{||\vec{\nabla} f||}$$

When the gradient magnitude is zero, we get no constraint!

In any case, further constraints are required to estimate both elements of the 2D velocity $\vec{\mathbf{u}} = (u_1, u_2)^T$.

Fleet & Weiss, 2005

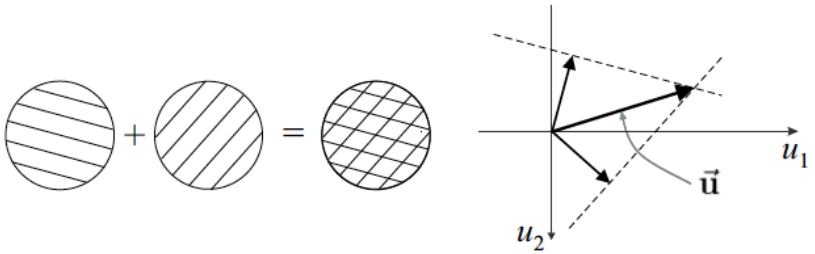
technical note

Area-Based Regression

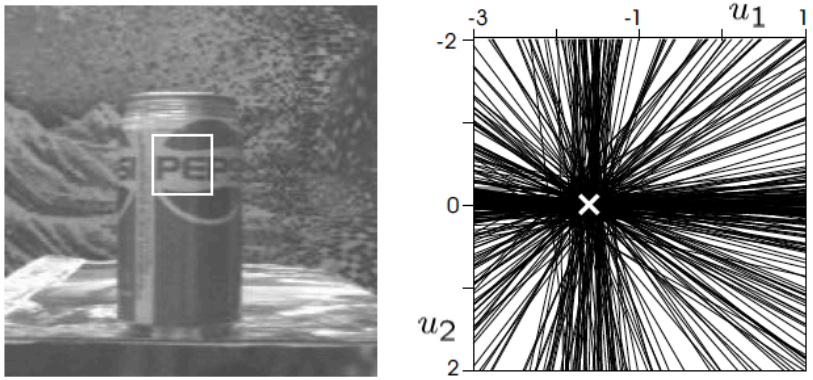
Smoothness Assumption: 2D motion is smooth in the local image neighborhood of the point at which we are estimating optical flow.

E.g., Let's say we have gradient constraints at two adjacent pixels, $\vec{\mathbf{x}}_1$ and $\vec{\mathbf{x}}_2$, which have the same velocity, $\vec{\mathbf{u}} = (u_1,\ u_2)^T$:

$$\begin{bmatrix} f_x(x_1, y_1, t) & f_y(x_1, y_1, t) \\ f_x(x_2, y_2, t) & f_y(x_2, y_2, t) \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \begin{bmatrix} f_t(x_1, y_1, t) \\ f_t(x_2, y_2, t) \end{bmatrix} = \vec{0}\ ,$$



More generally, we can use many constraints from within a local region about the point at which we wish to estimate the optical flow. Here is an example from the Pepsi sequence:



Fleet & Weiss, 2005

technical note

Area-Based Regression (Linear LS)

We will not satisfy all constraints exactly. Rather we seek the velocity that minimizes the squared error in each constraint (called the least-squares (LS) velocity estimate):

$$E(u_1, u_2) = \sum_{x,y} g(x,y)\,[u_1 f_x(x,y,t) + u_2 f_y(x,y,t) + f_t(x,y,t)]^2$$

where $g(x,y)$ is a low-pass window that helps give more weight to constraints at the center of the region.

Solution: Differentiate E with respect to (u_1, u_2) and set to zero:

$$\frac{\partial E(u_1,u_2)}{\partial u_1} = \sum_{xy} g(x,y)\,\left[u_1 {f_x}^2 + u_2 f_x f_y + f_x f_t\right] = 0$$

$$\frac{\partial E(u_1,u_2)}{\partial u_2} = \sum_{xy} g(x,y)\,\left[u_2 {f_y}^2 + u_1 f_x f_y + f_y f_t\right] = 0$$

The constraints thus yield two linear equations for u_1 and u_2.

In matrix notation, these *normal equations* and their solution are

$$\mathbf{M}\,\vec{\mathbf{u}} + \vec{\mathbf{b}} = \vec{0} \quad , \qquad \hat{\mathbf{u}} = -\mathbf{M}^{-1}\vec{\mathbf{b}}$$

(assuming $\mathbf{M}^{-1}$ exists), where

$$\mathbf{M} = \sum g \begin{pmatrix} f_x \\ f_y \end{pmatrix} (f_x,\ f_y) = \begin{bmatrix} \sum g\,{f_x}^2 & \sum g\,f_x f_y \\ \sum g\,f_x f_y & \sum g\,{f_y}^2 \end{bmatrix}$$

$$\vec{\mathbf{b}} = \sum g\,f_t \begin{pmatrix} f_x \\ f_y \end{pmatrix} = \begin{pmatrix} \sum g\,f_x f_t \\ \sum g\,f_y f_t \end{pmatrix}$$

Fleet & Weiss, 2005

technical note

Area-Based Regression (Implementation)

Since we want the optical flow at each pixel, we can compute the components of the normal equations with a set of image operators. This is much prefered to looping over image pixels.

1. First, compute 3 image gradient images at time t, corresponding to the gradient measurements:

$$f_x(\vec{\mathbf{x}}) \,, \quad f_y(\vec{\mathbf{x}}) \,, \quad f_t(\vec{\mathbf{x}})$$

2. Point-wise, we compute the quadratic functions of the derivative images. This produces five images equal to:

$$f_x^2(\vec{\mathbf{x}}) \,, \quad f_y^2(\vec{\mathbf{x}}) \,, \quad f_x(\vec{\mathbf{x}}) f_y(\vec{\mathbf{x}}) \,, \quad f_t(\vec{\mathbf{x}}) f_y(\vec{\mathbf{x}}) \,, \quad f_t(\vec{\mathbf{x}}) f_x(\vec{\mathbf{x}})$$

3. Blurring the quadratic images corresponds to the accumulating of local constraints under the spatial (or spatiotemporal) support window g above. This produces five images, each of which contains an element of the normal equations at a specific image locations:

$$g(\vec{\mathbf{x}}) * f_x^2(\vec{\mathbf{x}}) \,, \quad g(\vec{\mathbf{x}}) * f_y^2(\vec{\mathbf{x}}) \,,$$
$$g(\vec{\mathbf{x}}) * [f_x(\vec{\mathbf{x}}) \, f_y(\vec{\mathbf{x}})] \,, \quad g(\vec{\mathbf{x}}) * [f_t(\vec{\mathbf{x}}) \, f_y(\vec{\mathbf{x}})] \,, \quad g(\vec{\mathbf{x}}) * [f_t(\vec{\mathbf{x}}) \, f_x(\vec{\mathbf{x}})]$$

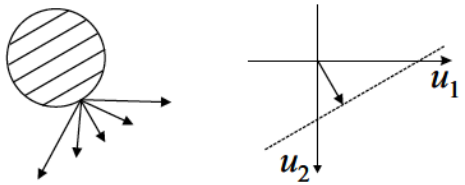
4. Compute the two images containing the components of optical flow at each pixel. This is given in closed form since the inverse of the normal matrix (i.e., M above) is easily expressed in closed form.

Fleet & Weiss, 2005

technical note

Aperture Problem

Even with the collection of constraints from within a region, the estimation will be undetermined when the matrix M is singular:



When all image gradients are parallel, then the normal matrix for the least-squares solution becomes singular (rank 1). E.g., for gradients $m(x,y)\vec{\mathbf{n}}$, where $m(x,y)$ is the gradient magnitude at pixel (x,y)

$$\mathbf{M} = \left(\sum_{x,y} g(x,y)\, m^2(x,y) \right) \vec{\mathbf{n}}\, \vec{\mathbf{n}}^T$$

Aperture Problem: For sufficiently small spatial apertures the normal matrix, M, will be singular (rank deficient).

Generalized Aperture Problem: For small apertures M becomes singular, but for sufficiently large apertures the 2D motion field may deviate significantly from the assumed motion model.

Heuristic Diagnostics: Use singular values, $s_1 \geq s_2 \geq s_3$, of $\sum_{x,y} g\, \vec{\mathbf{h}}\, \vec{\mathbf{h}}^T$, for $\vec{\mathbf{h}} = (f_x, f_y, f_t)^T$, to assess the rank of the local image structure:

- if s_2 is "too small" then only normal velocity is available
- if s_3 is "large" then the gradient constraints do not lie in a plane so a single velocity will not fit the data well.

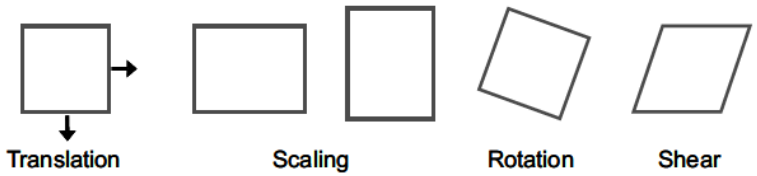
Such "bounds" will depend on the LS region size and the noise.

Fleet & Weiss, 2005

technical note

Higher-Order Motion Models

Constant flow model within an image neighborhood is often a poor model, especially when the regions become larger. Affine models often provide a more suitable model of local image deformation.



Affine flow for an image region centered at location $\vec{\mathbf{x}}_0$ is given by

$$\vec{\mathbf{u}}(\vec{\mathbf{x}}) = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} (\vec{\mathbf{x}} - \vec{\mathbf{x}}_0) + \begin{pmatrix} a_5 \\ a_6 \end{pmatrix} = \mathbf{A}(\vec{\mathbf{x}}; \vec{\mathbf{x}}_0)\, \vec{\mathbf{a}}$$

where $\vec{\mathbf{a}} = (a_1, a_2, \cdots, a_6)^T$ and

$$\mathbf{A}(\vec{\mathbf{x}}; \vec{\mathbf{x}}_0) = \begin{bmatrix} x-x_0 & y-y_0 & 0 & 0 & 1 & 0 \\ 0 & 0 & x-x_0 & y-y_0 & 0 & 1 \end{bmatrix}$$

Then the gradient constraint becomes:

$$\begin{aligned} 0 &= \vec{\mathbf{u}}(x,y)^T \vec{\nabla} f(x,y,t) + f_t(x,y,t) \\ &= \vec{\mathbf{a}}^T \mathbf{A}(x,y)^T \vec{\nabla} f(x,y,t) + f_t(x,y,t)\ , \end{aligned}$$

so, as above, the weighted least-squares solution for $\vec{\mathbf{a}}$ is given by:

$$\hat{\mathbf{a}} = \mathbf{M}^{-1} \vec{\mathbf{b}}$$

where, for weighting with the spatial window g,

$$\mathbf{M} = \sum_{x,y} g(\vec{\mathbf{x}})\, \mathbf{A}^T\, \vec{\nabla} f\, \vec{\nabla} f^T \mathbf{A}\ , \qquad \vec{\mathbf{b}} = -\sum_{x,y} g(\vec{\mathbf{x}})\, \mathbf{A}^T\, \vec{\nabla} f\, f_t$$

Fleet & Weiss, 2005