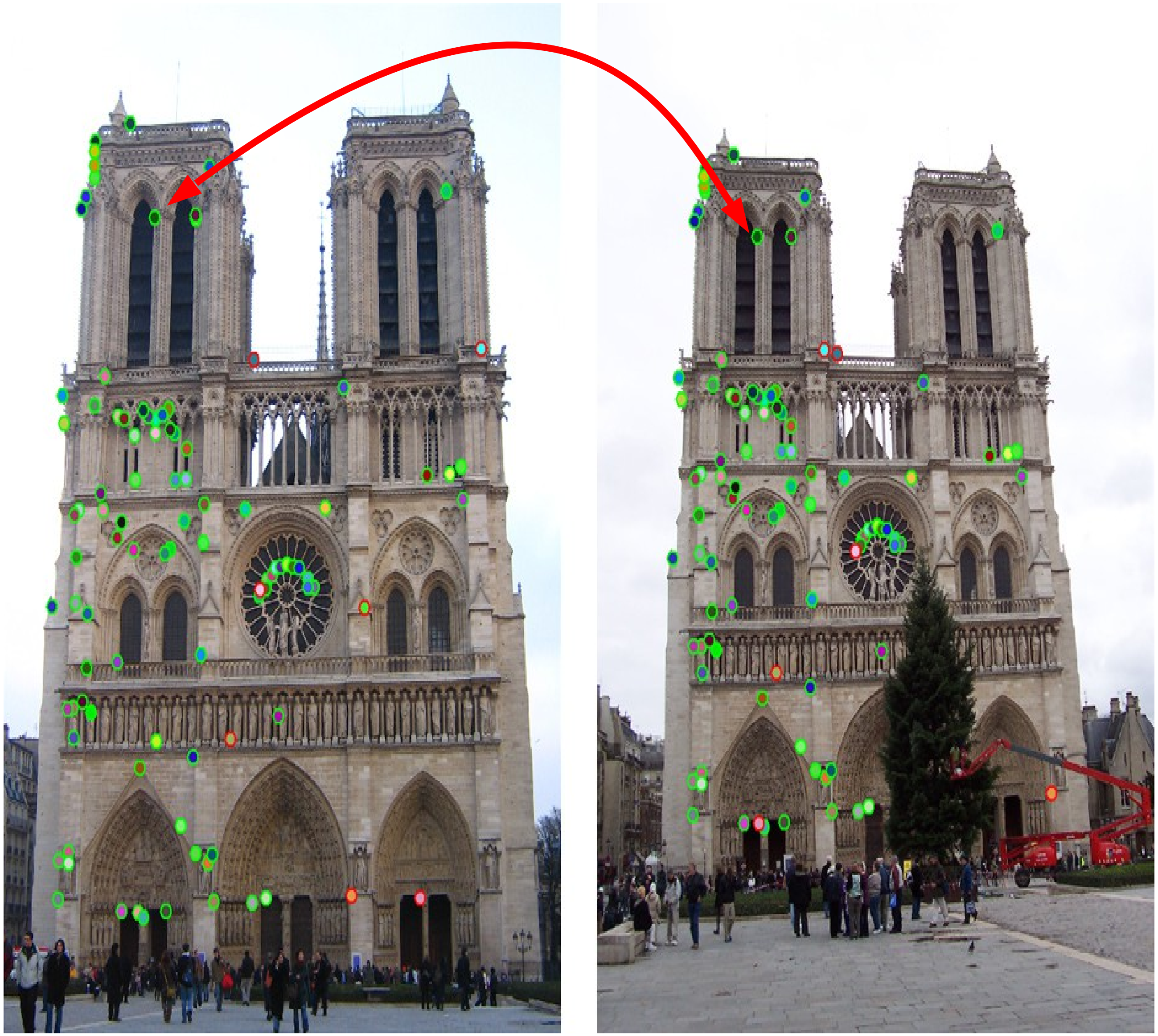
Lecture 13
Optical Flow and Tracking

COS 429: Computer Vision

PRINCETON UNIVERSITY

Slides credit:
Many slides adapted from Michael Black, James Hays, Derek Hoeim, Lana Lazebnik, Silvio Saverse, Steve Seitz, Rick Szeliski, Martial Hebert, Mark Pollefeys, and others

Recall: Feature Matching



1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

Slide Credit:

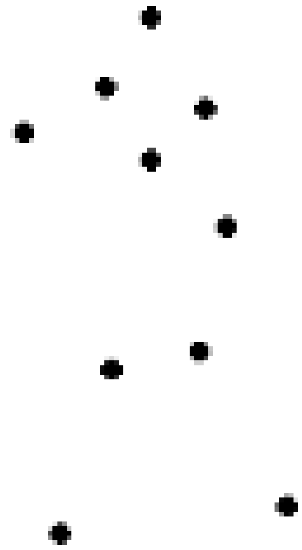
Optical Flow and Tracking

https://youtu.be/GIUDAZLfYhY?t=63

https://youtu.be/jg6Nz6BfoSQ

Slide Credit:

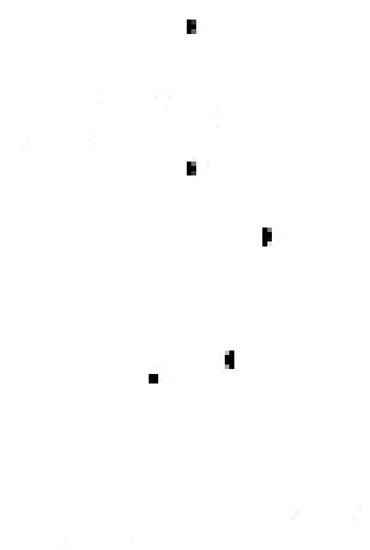
Human Motion Perception



G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

 Slide Credit:

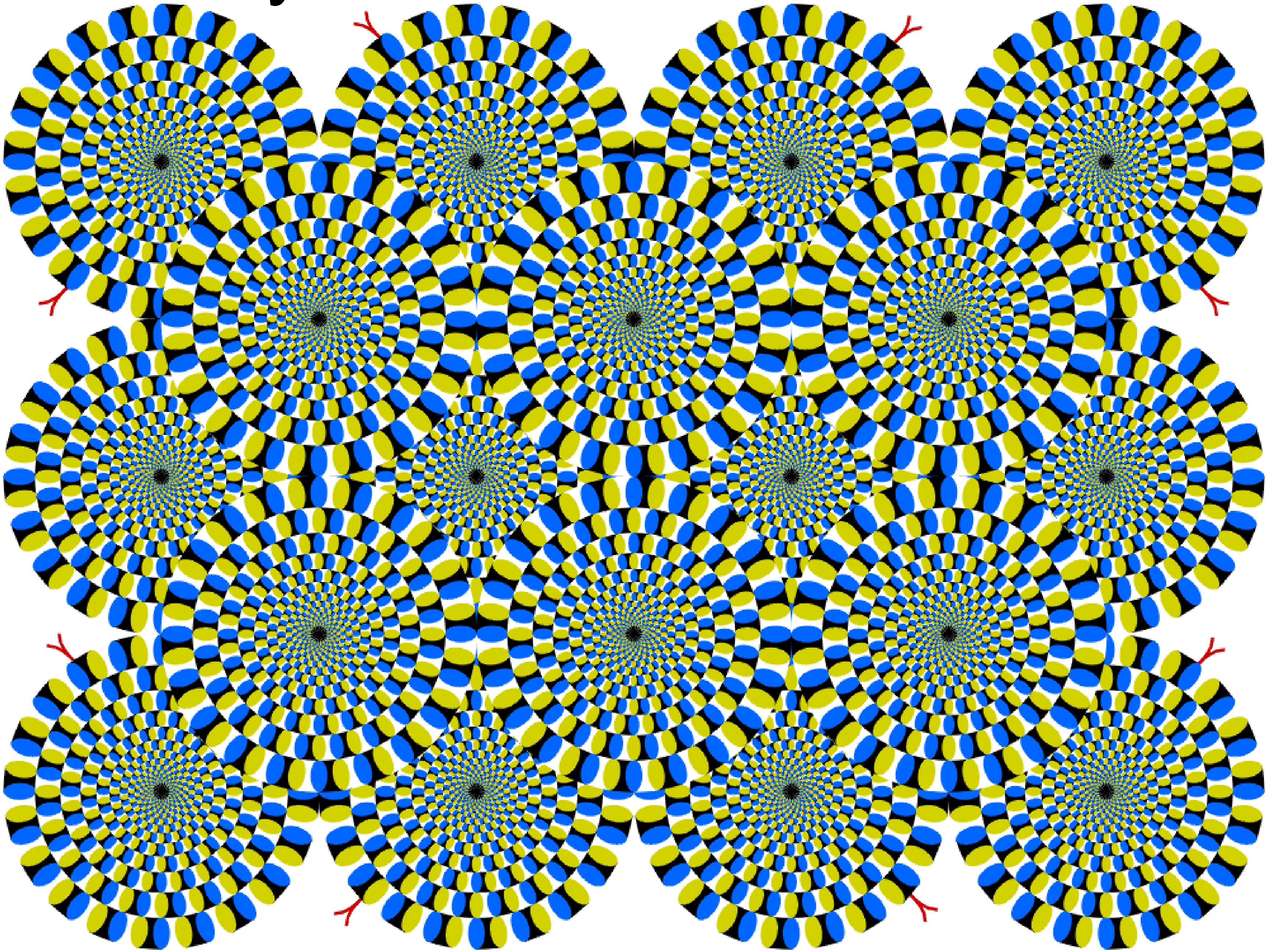
Human Motion Perception



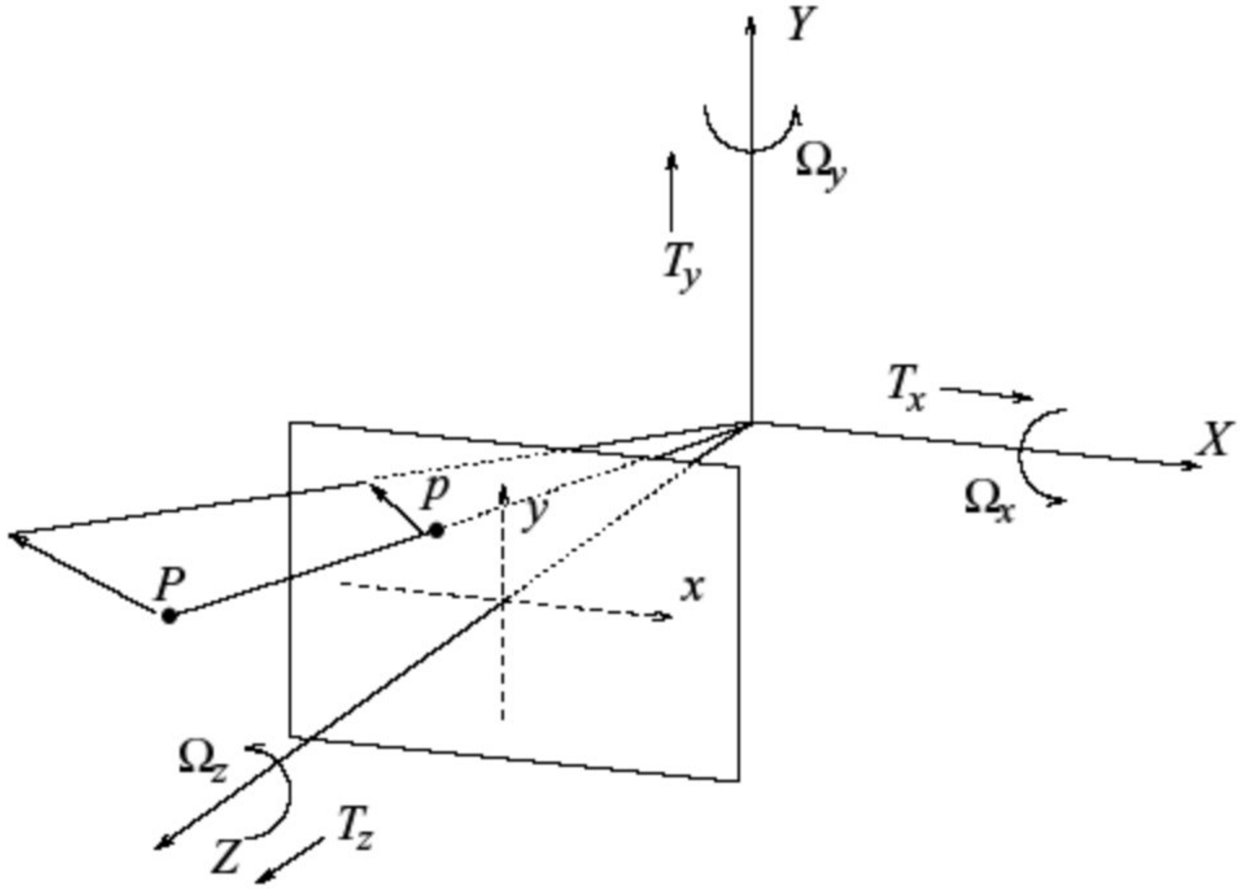
G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

 Slide Credit:

Illusory Snakes



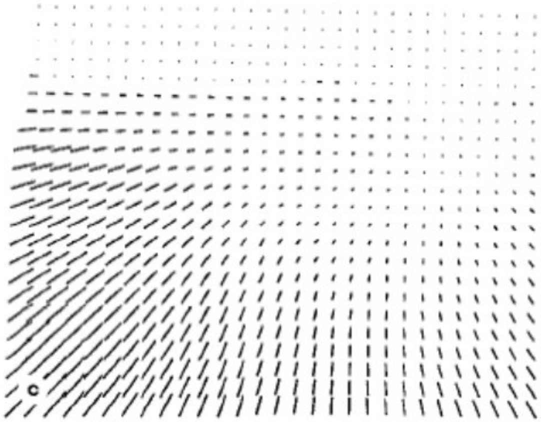
Motion Field



Motion field = 2D motion field representing the projection of the 3D motion of points in the scene onto the image plane

Slide Credit: Michael Black

Optical Flow Field



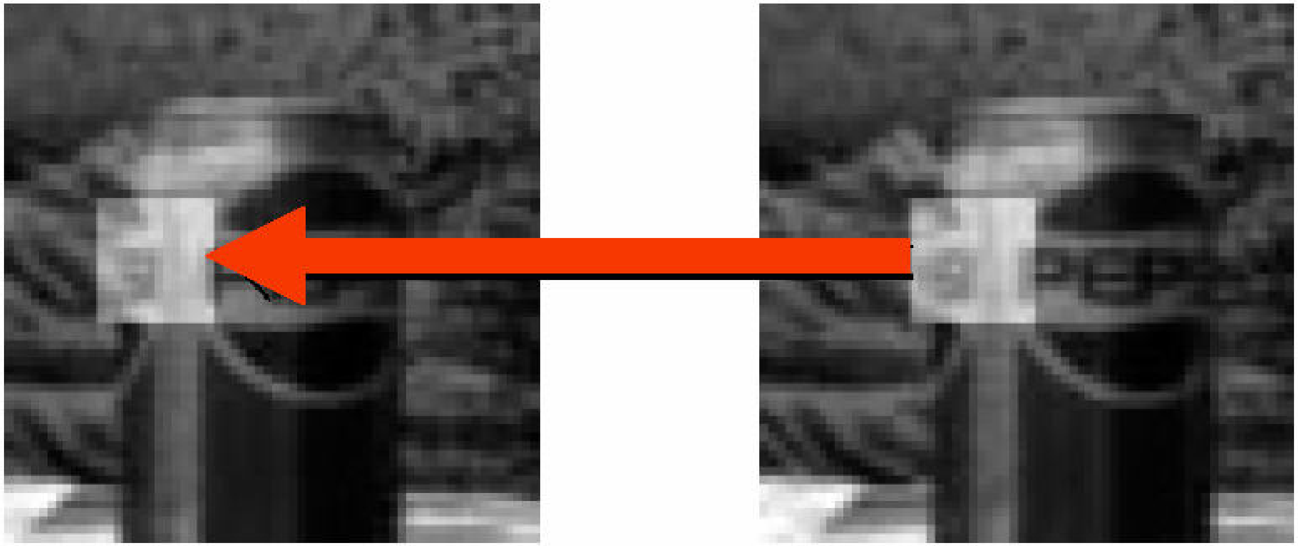
$u(x, y)$ Horizontal component

$v(x, y)$ Vertical component

Optical flow = 2D velocity field describing the apparent motion in the images.

 Slide Credit: Michael Black

Optical Flow Assumptions: Brightness Constancy



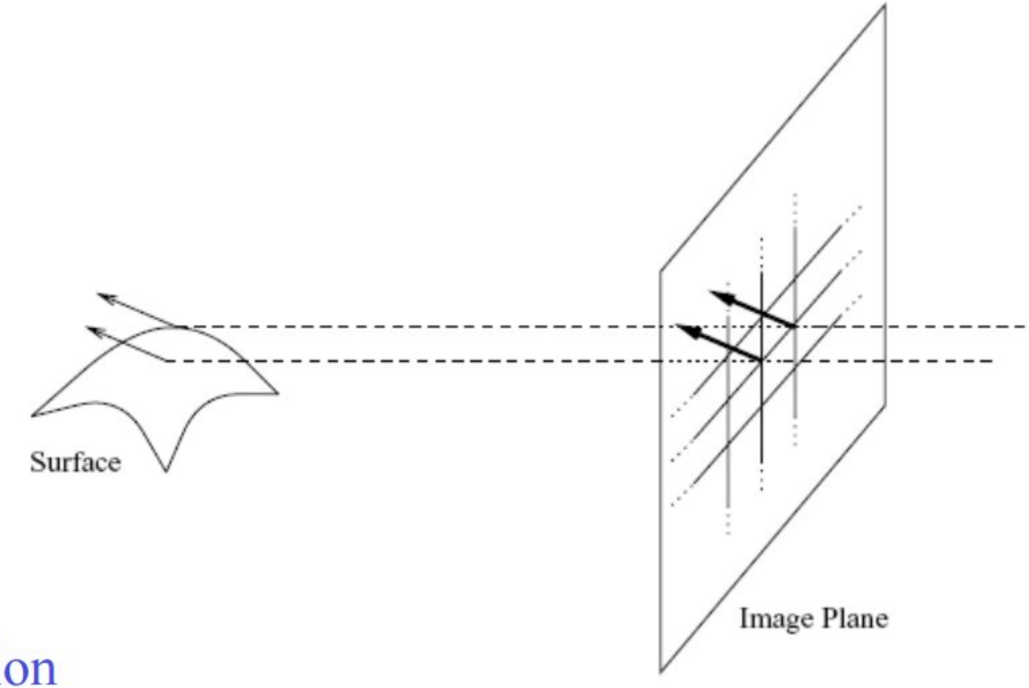
Assumption

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

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

 Slide Credit: Michael Black

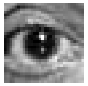
Optical Flow Assumptions: Spatial Coherence



Assumption

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

=> this allows us to assume a small patch will move together:



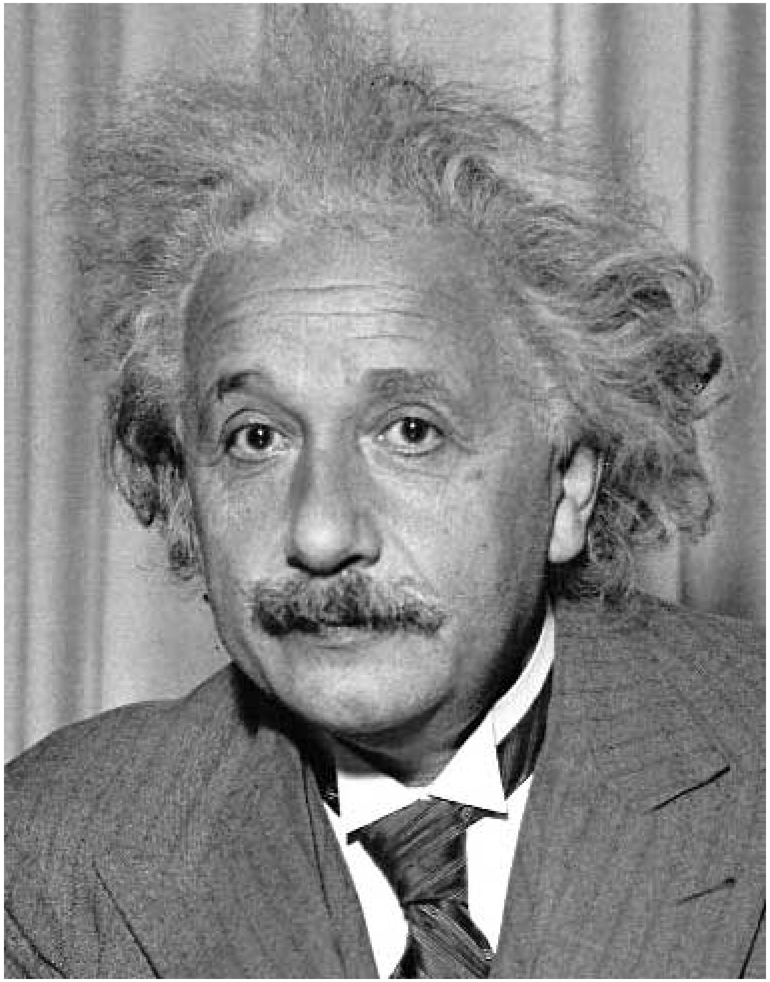
 Slide Credit: Michael Black

Distance Metric

- Goal: find  in image,
  assume translation only: no scale change or rotation,

using search (scanning the image)

- What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation



Slide Credit:

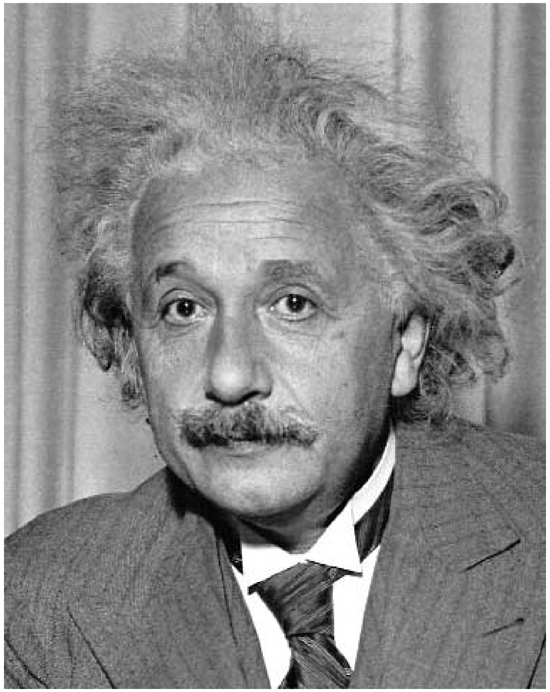
Matching with filters

Goal: find  in image

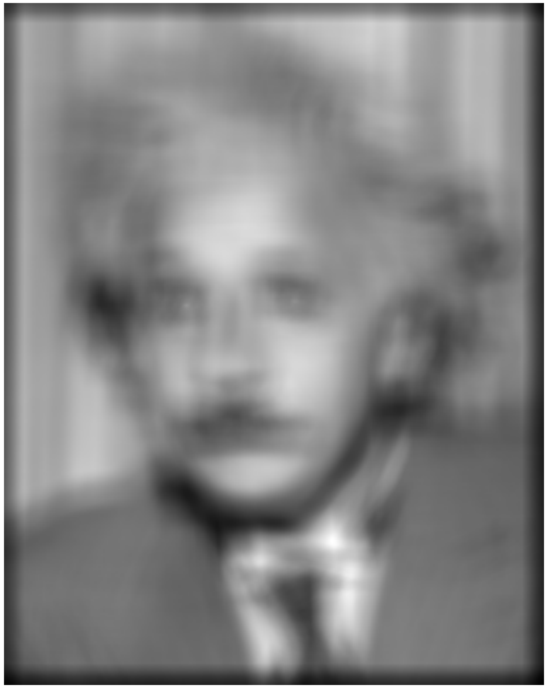
- Method 0: filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

f = image
g = filter



Input



Filtered Image

What went wrong?

response is stronger for higher intensity

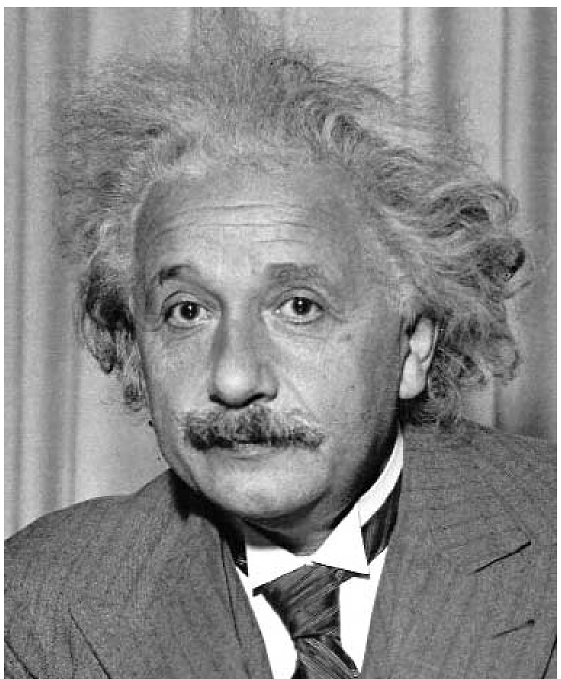
Slide Credit:

0-mean filter

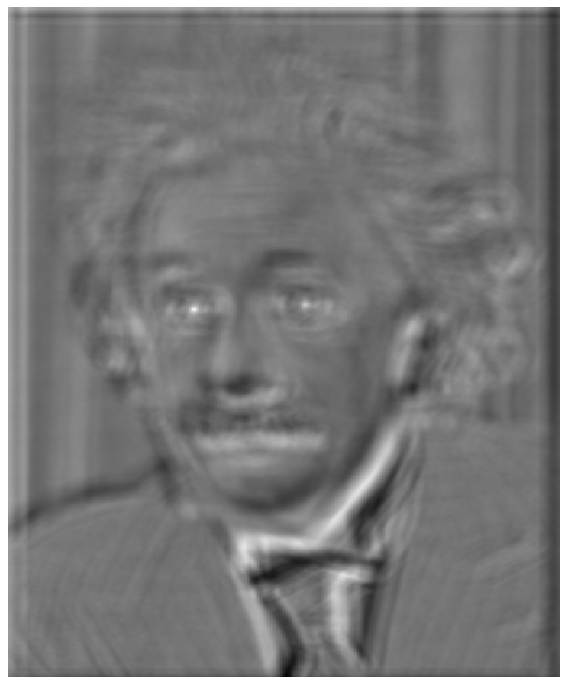
- Goal: find  image
- Method 1: filter the image with zero-mean eye

$$h[m,n] = \sum_{k,l} (f[k,l] - \bar{f})\,(g[m+k,n+l])$$

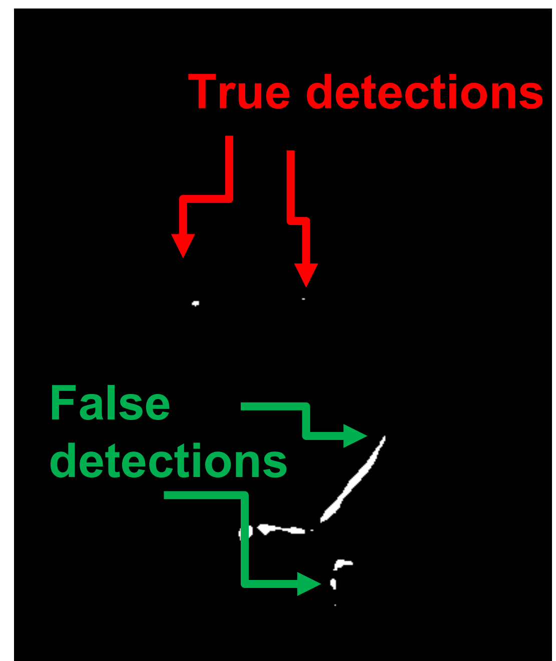
mean of f



Input



Filtered Image (scaled)



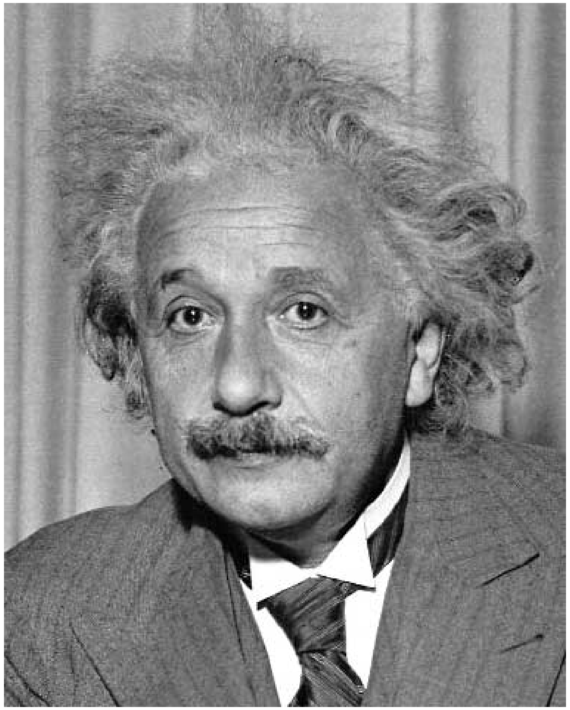
Thresholded Image

Slide Credit:

Sum of Squared error (L2)

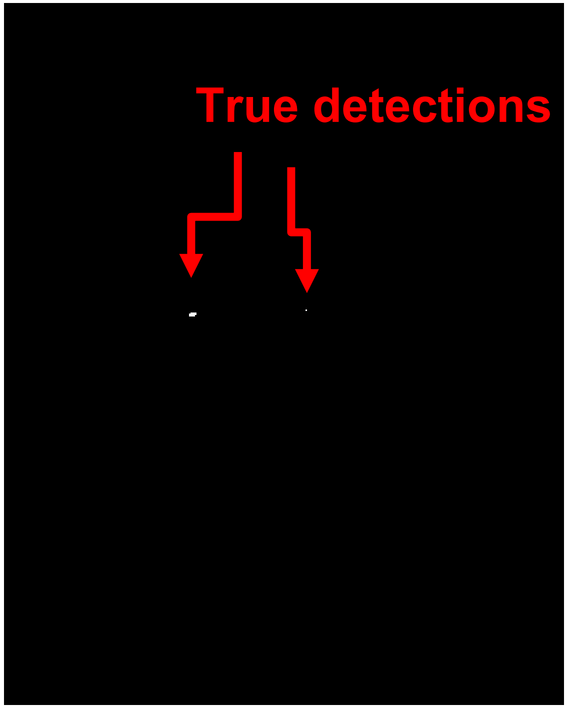
- Goal: find  in image
- Method 2: SSD

$$h[m,n]=\sum_{k,l}(g[k,l]-f[m+k,n+l])^2$$



Input

1- sqrt(SSD)



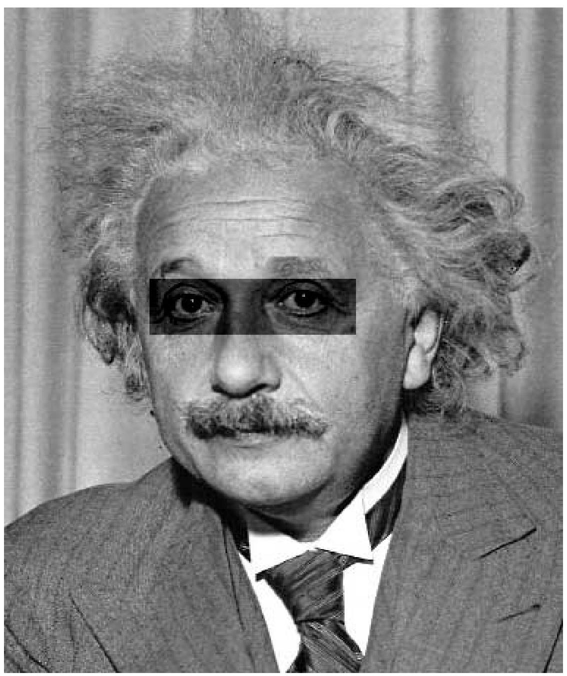
Thresholded Image

Slide Credit:

Sum of Squared error (L2)

- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$



Input

1- sqrt(SSD)

One potential downside of SSD:

Brightness Constancy Assumption

Slide Credit:

Normalized Cross-Correlation

- Goal: find in image
- Method 3: Normalized cross-correlation
(= angle between zero-mean vectors)

Template mean

image patch mean

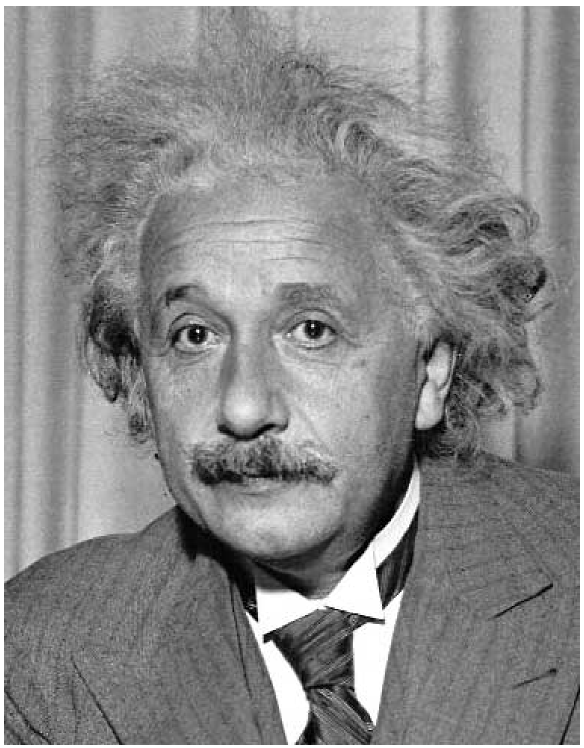
$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m-k,n-l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m-k,n-l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Matlab: `normxcorr2(template, im)`

Slide Credit:

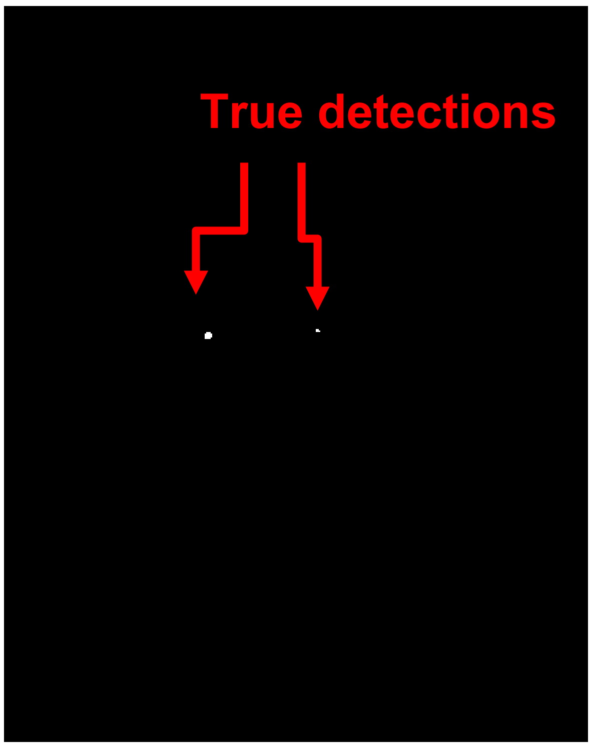
Normalized Cross-Correlation

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

Normalized X-Correlation

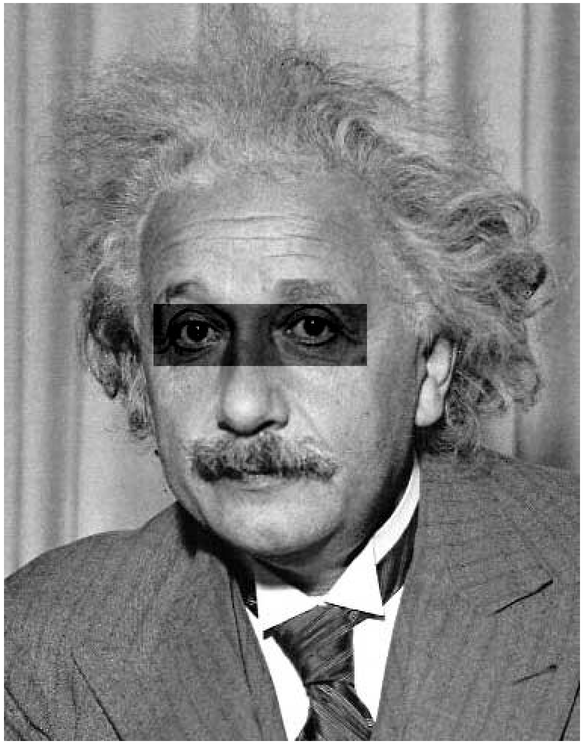


Thresholded Image

Slide Credit:

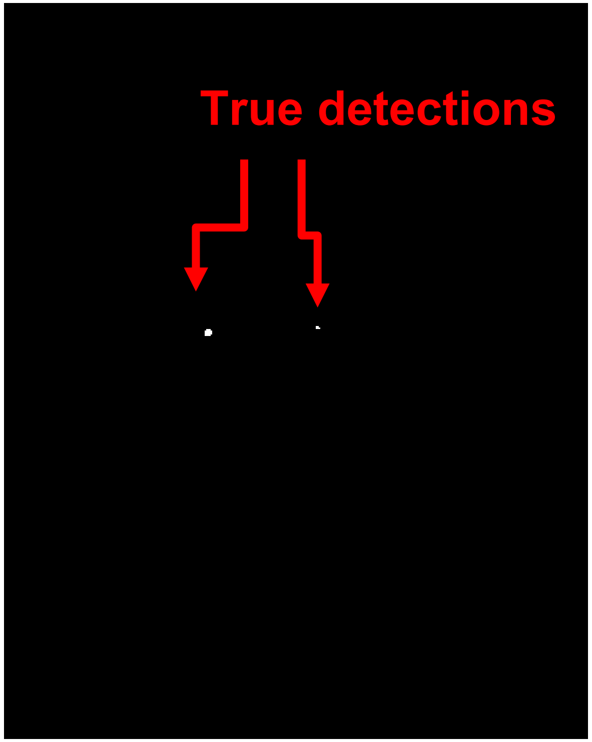
Normalized Cross-Correlation

- Goal: find in image
- Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



Thresholded Image

Slide Credit:

Search vs. Gradient Descent

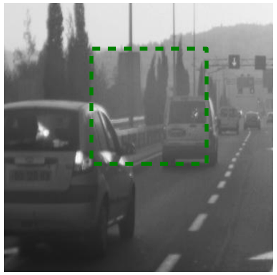
- Search:
  - Pros: Free choice of representation, distance metric; no need for good initial guess
  - Cons: expensive when searching for sub-pixel accuracy or over complex motion models (scale, rotation, affine)

- If we have a good guess, can we do something cheaper?
  - Gradient Descent

Slide Credit:

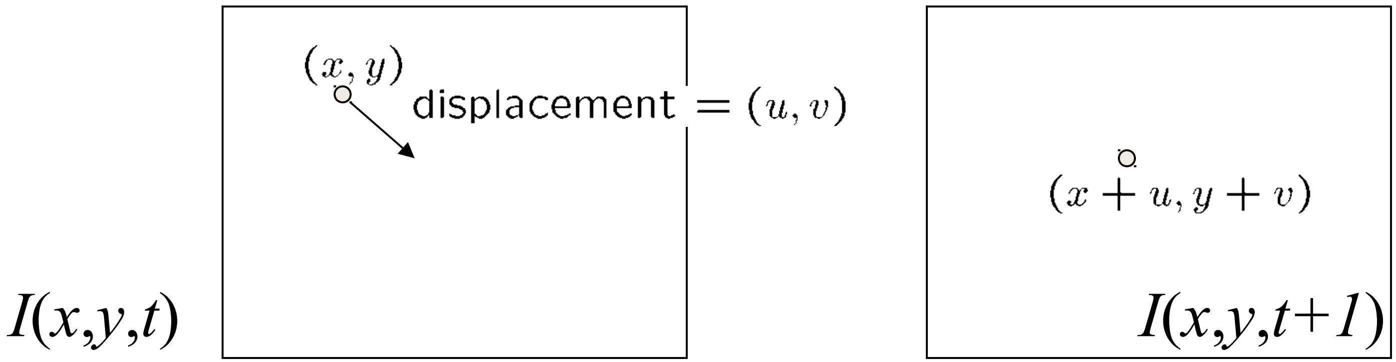
Lucas-Kanade Object Tracker

- Key assumptions:
  - **Brightness constancy:** projection of the same point looks the same in every frame (uses SSD as metric)
  - **Small motion:** points do not move very far (from guessed location)
  - **Spatial coherence:** points move in some coherent way (according to some parametric motion model)
    - For this example, assume whole object just translates in (u,v)



 Slide Credit:

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:

Image derivative along x　　Difference over frames

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

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

Hence,

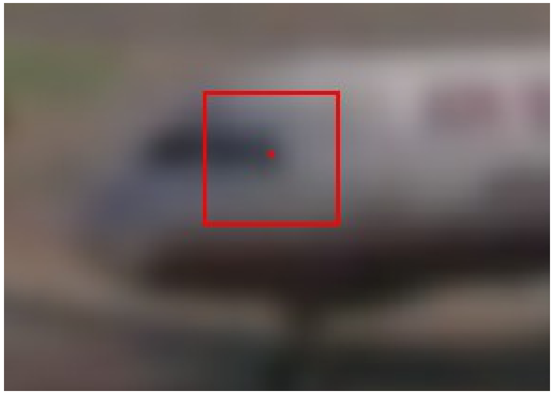
$$I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla I \cdot [u \;\; v]^T + I_t = 0$$

Slide Credit:

How does this make sense?

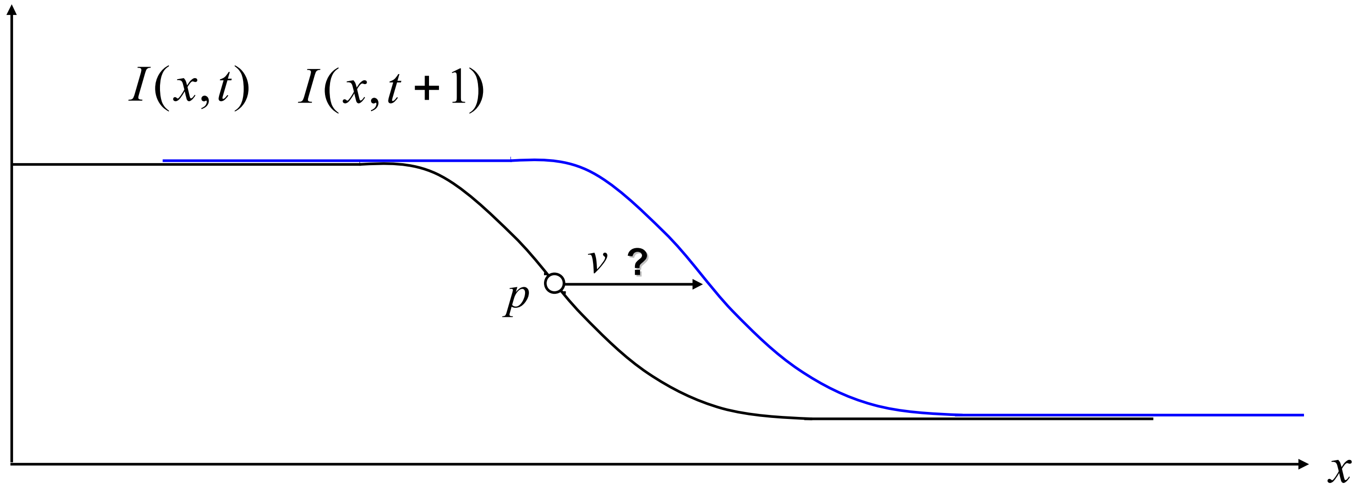
$$\nabla I \cdot [u \; v]^T + I_t = 0$$

- What do the static image gradients have to do with motion estimation?



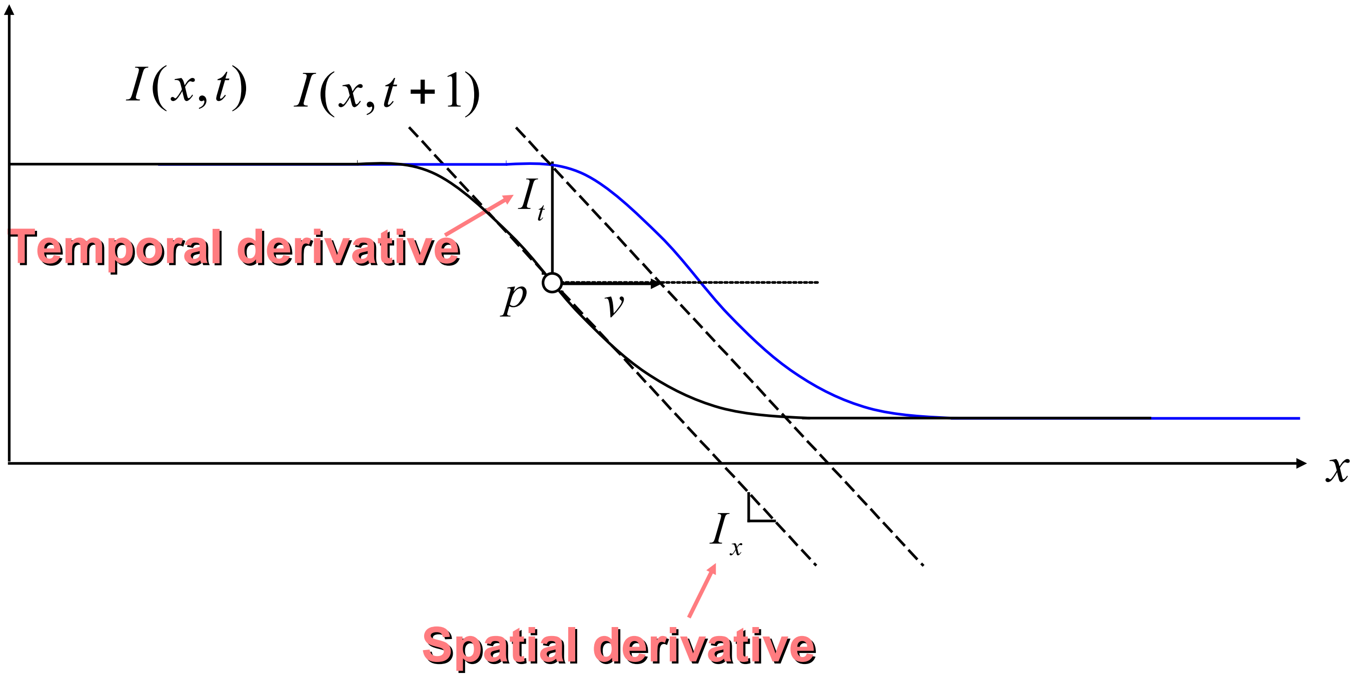
Slide Credit:

Tracking in the 1D case:



Slide Credit:

Tracking in the 1D case:



$$I_x = \left.\frac{\partial I}{\partial x}\right|_t \qquad I_t = \left.\frac{\partial I}{\partial t}\right|_{x=p}$$

$$\vec{v} \approx - \frac{I_t}{I_x}$$

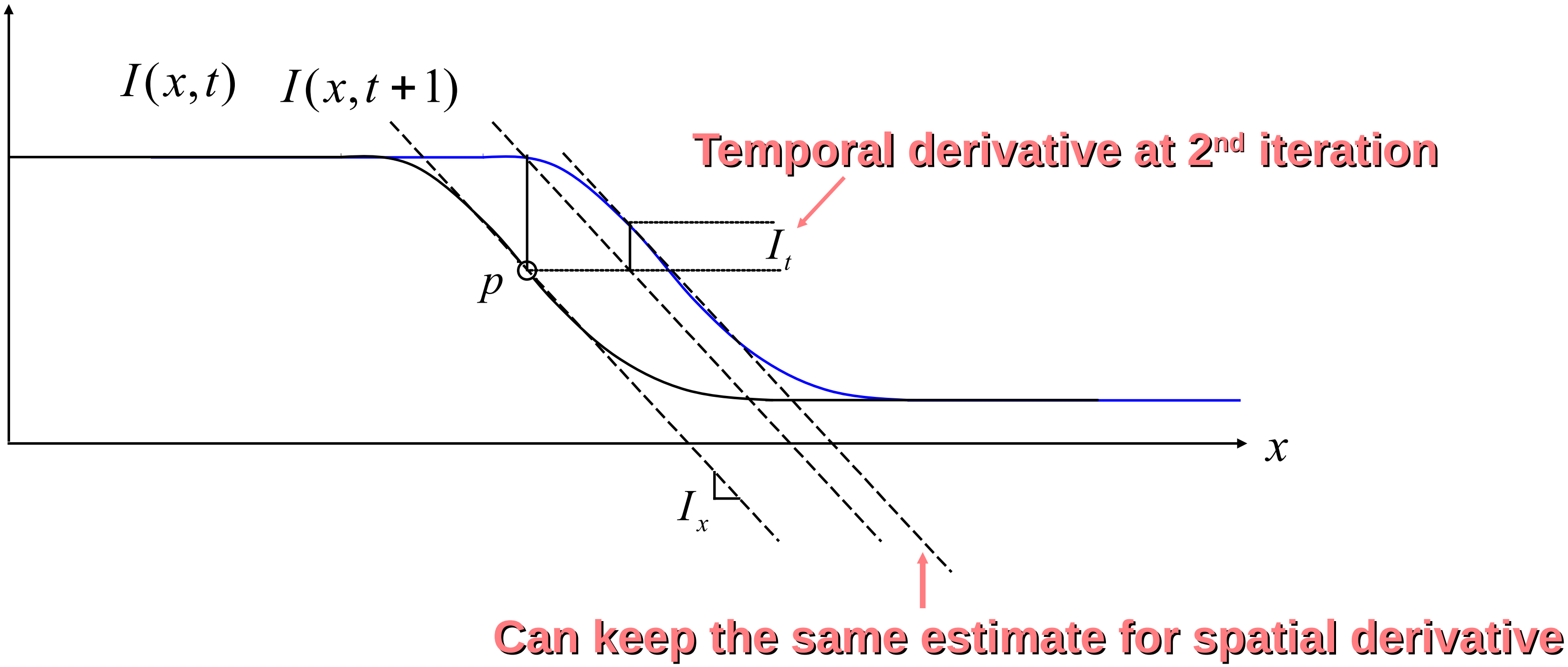
Assumptions:

- **Brightness constancy**
- **Small motion**

 Slide Credit:

Tracking in the 1D case:

Iterating helps refining the velocity vector



$$\vec{v} \leftarrow \vec{v}_{previous} - \frac{I_t}{I_x}$$

Converges in about 5 iterations

Slide Credit:

The brightness constancy constraint

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

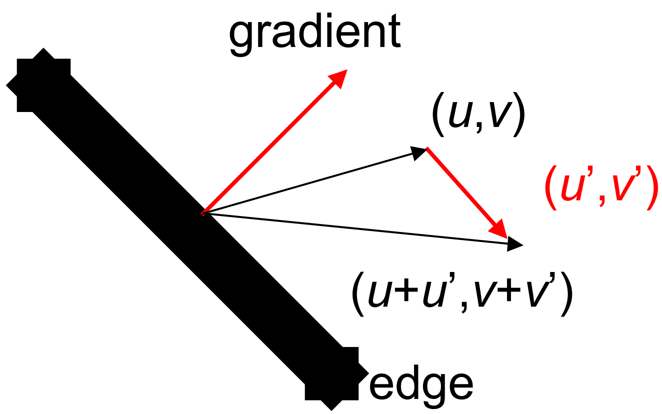
$$\nabla I \cdot [u \ v]^T + I_t = 0$$

- 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

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



Slide Credit:

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.

- Spatial coherence constraint: solve for many pixels and assume they all have the same motion
- In our case, if the object fits in a 5x5 pixel patch, this gives us 25 equations:

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

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

Slide Credit:

Solving the ambiguity...

- Least squares problem:

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

Slide Credit:

Solving for free parameters (u,v)

- Over-constrained linear system

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

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

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

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

Slide Credit:

Dealing with larger movements: Iterative refinement

Original (x,y) position

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

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

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

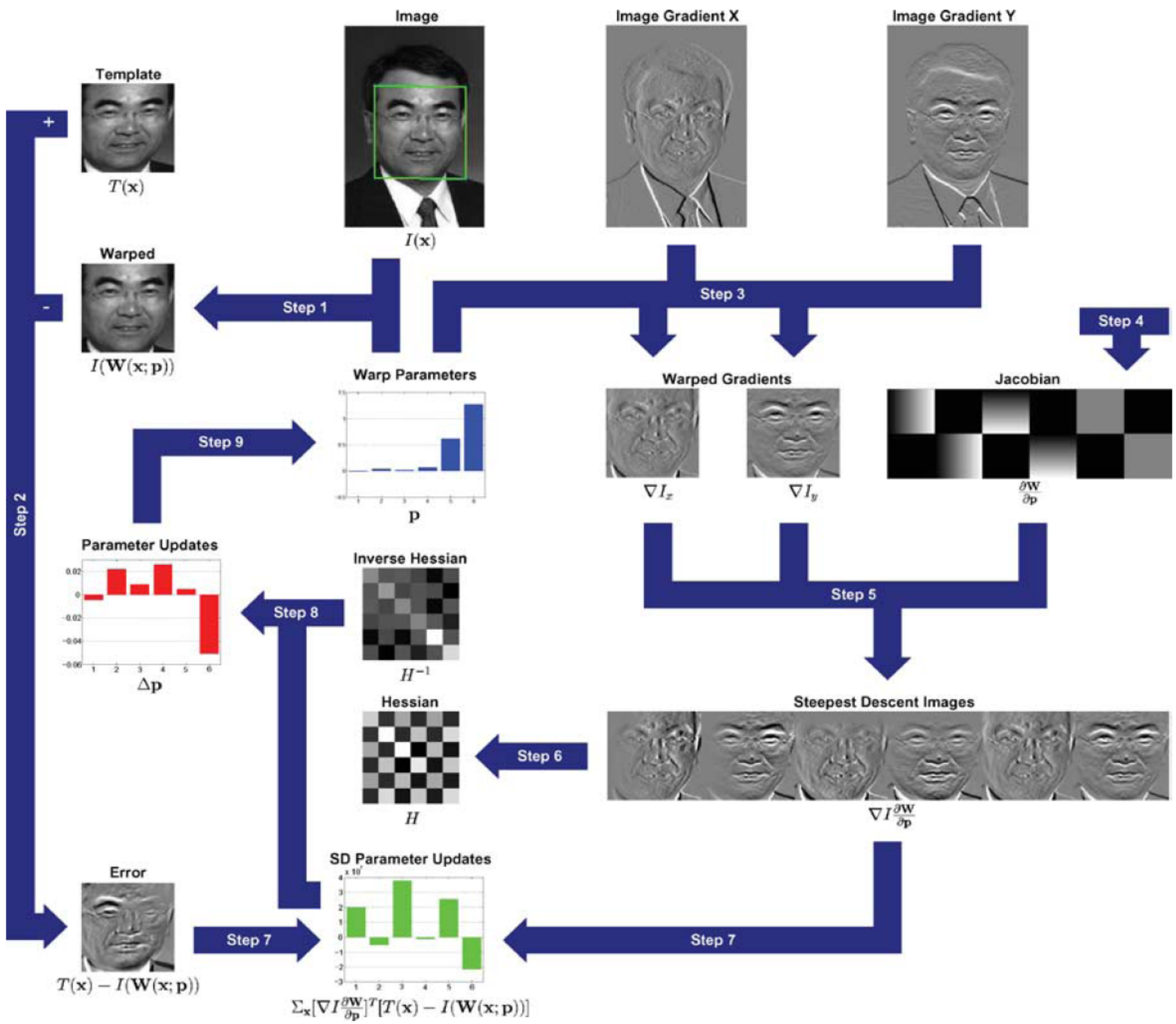
2nd moment matrix for feature patch in first image

displacement

1. Shift window by (u, v): `x'=x'+u;  y'=y'+v;`
2. Recalculate I_t
3. Repeat steps 2-4 until small change
   - Use interpolation to warp by subpixel values

Slide Credit:

Schematic of Lucas-Kanade



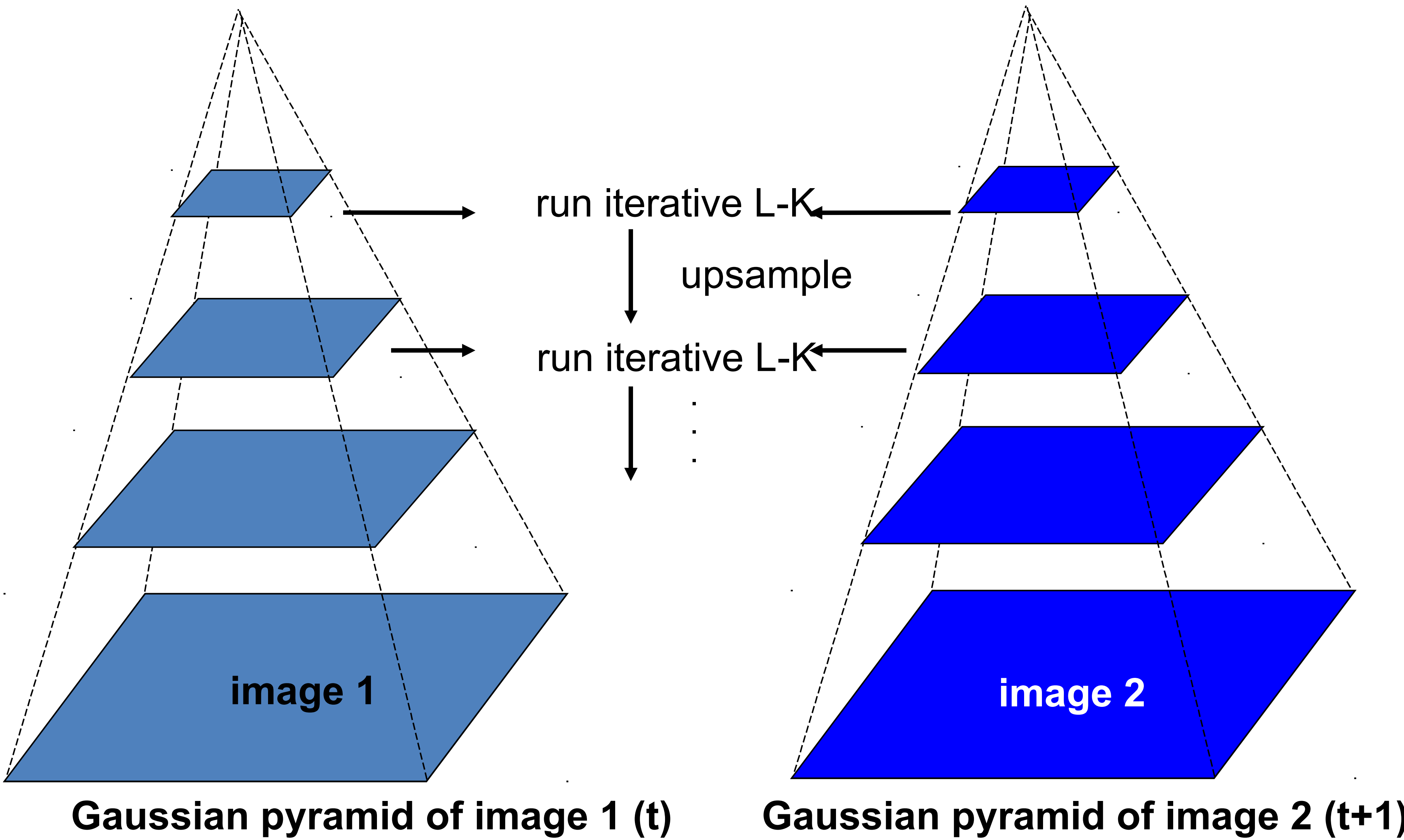
Slide Credit: [Baker & Matthews, 2003]

Dealing with larger movements

- How to deal with cases where the initial guess is not within a few pixels of the solution?

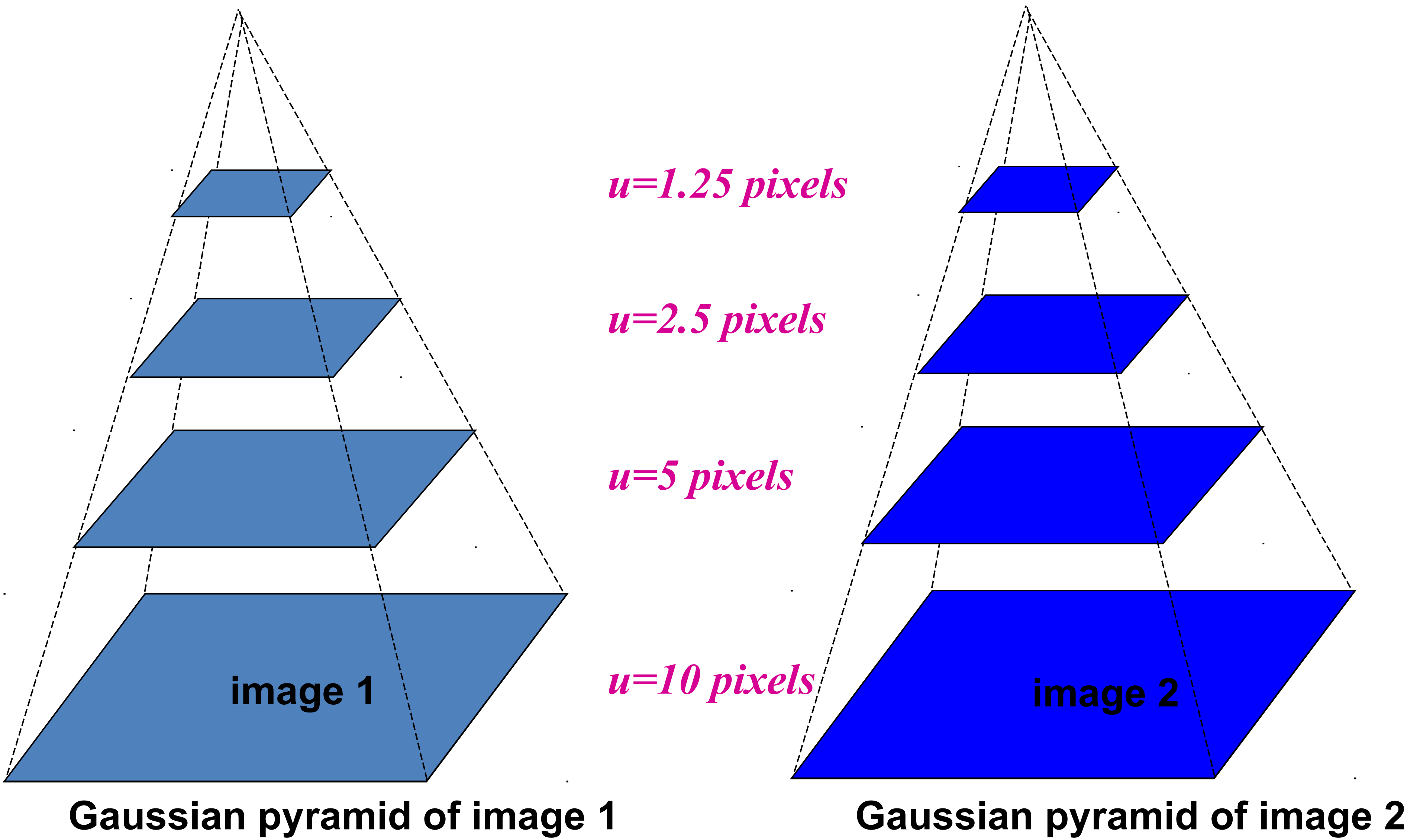
Slide Credit:

Dealing with larger movements: coarse-to-fine registration



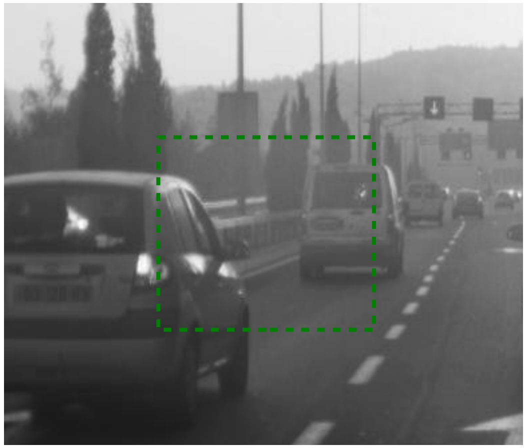
Slide Credit:

Coarse-to-fine optical flow estimation



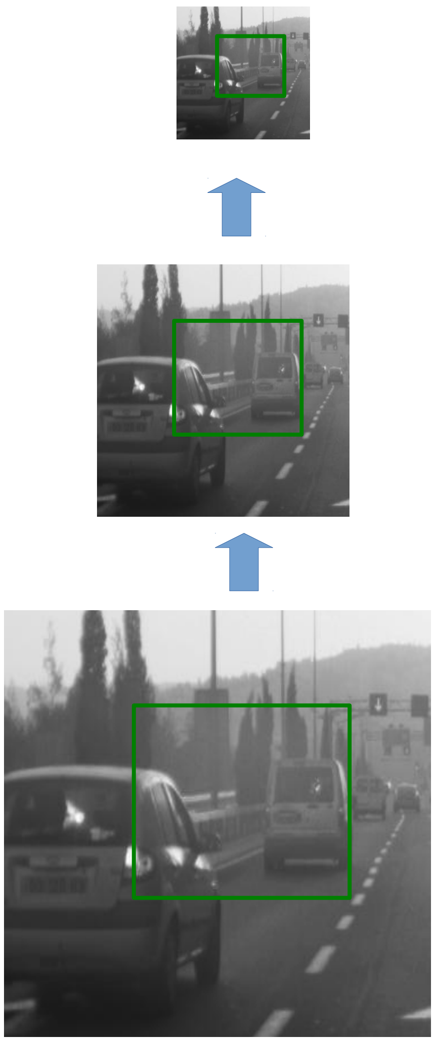
Slide Credit:

Coarse-to-fine optical flow estimation



Slide Credit:

Coarse-to-fine optical flow estimation



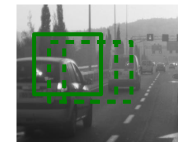
Downscale image and rectangle

Downscale image and rectangle

Slide Credit:

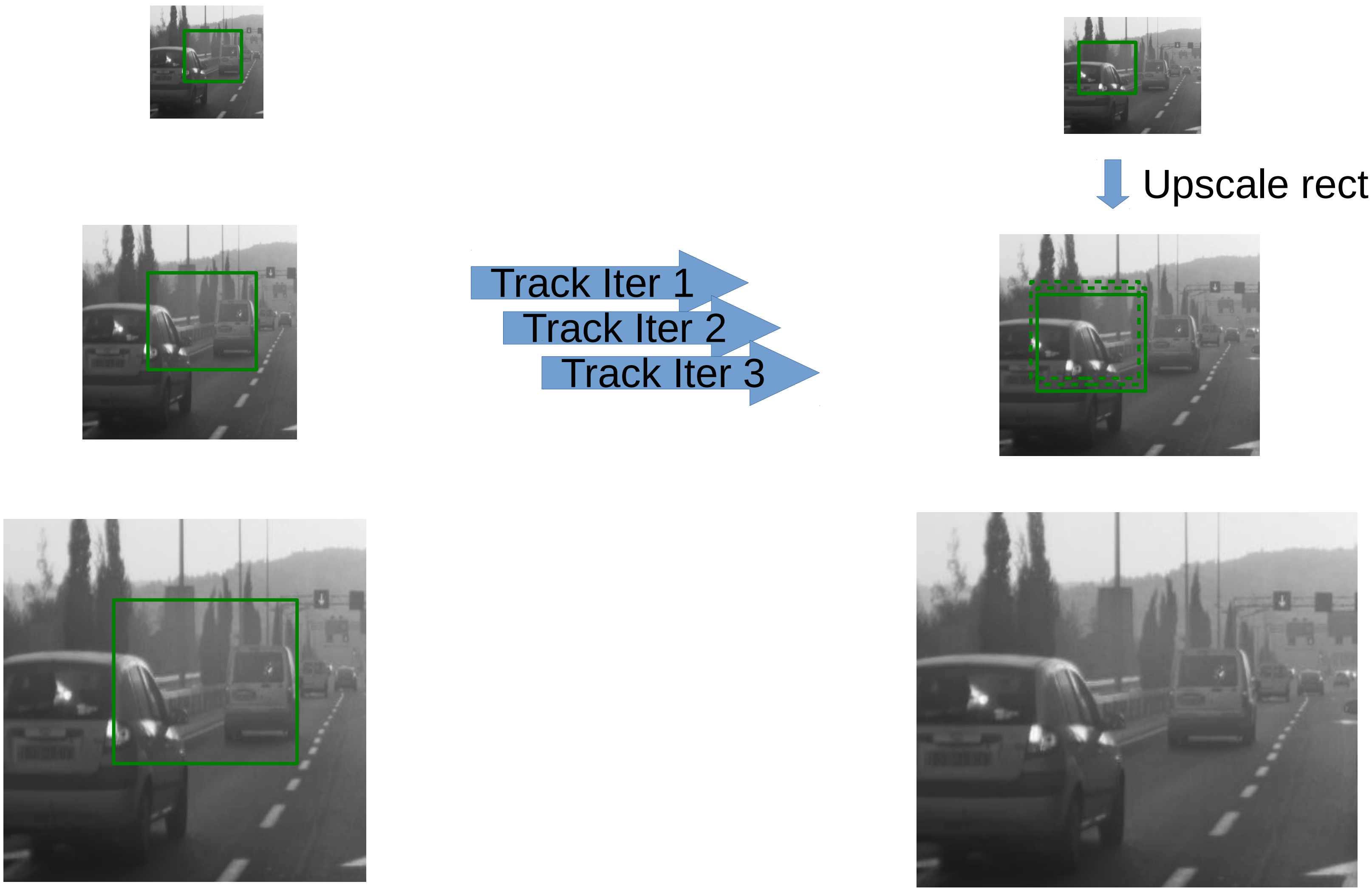
Coarse-to-fine optical flow estimation

Track Iter 1
Track Iter 2
Track Iter 3



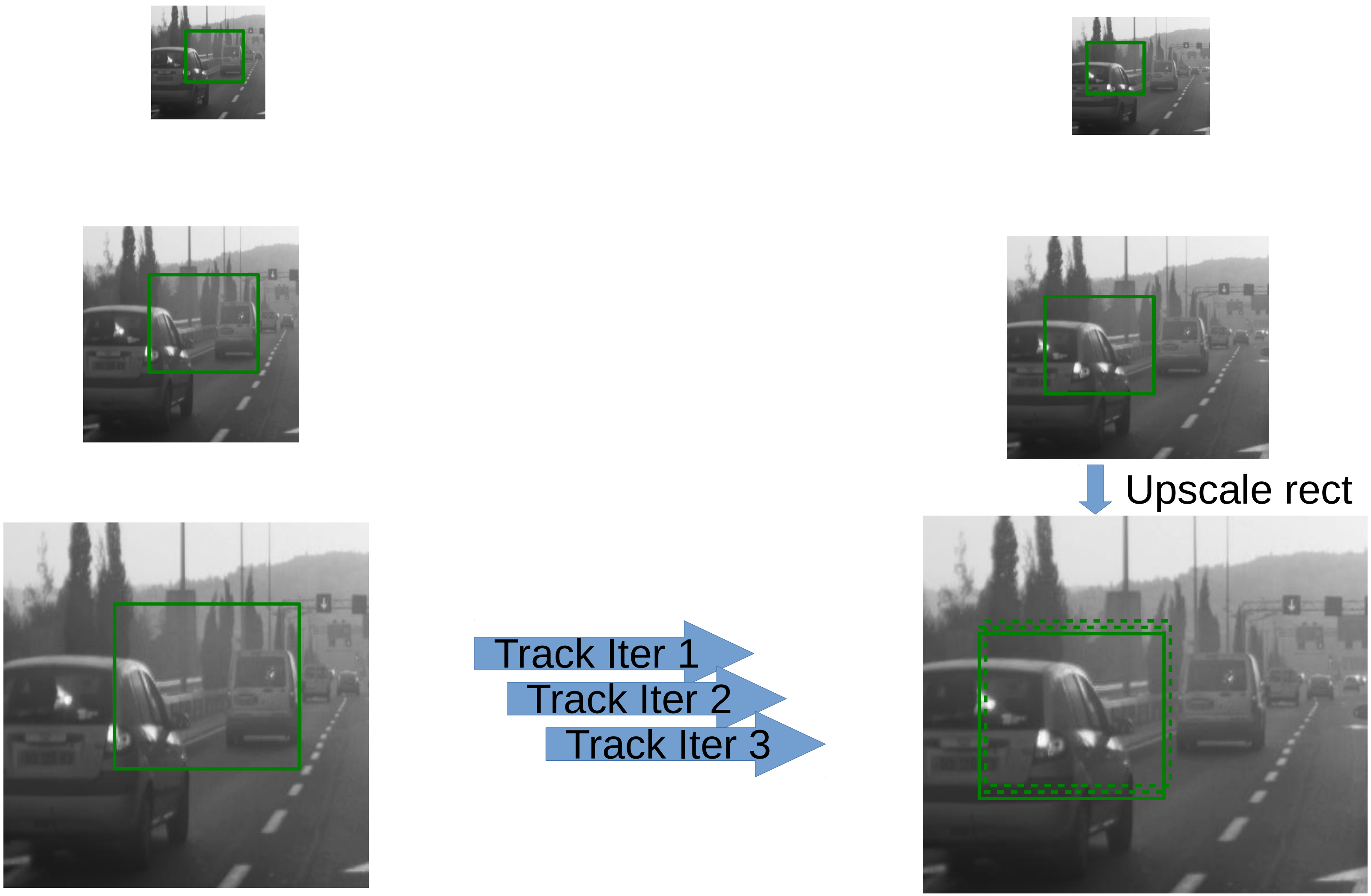
Slide Credit:

Coarse-to-fine optical flow estimation



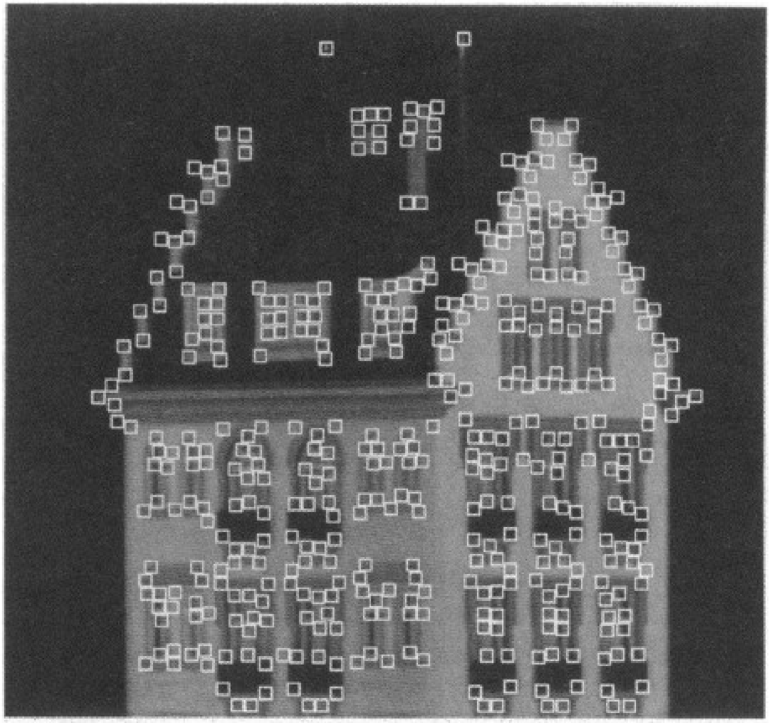
Slide Credit:

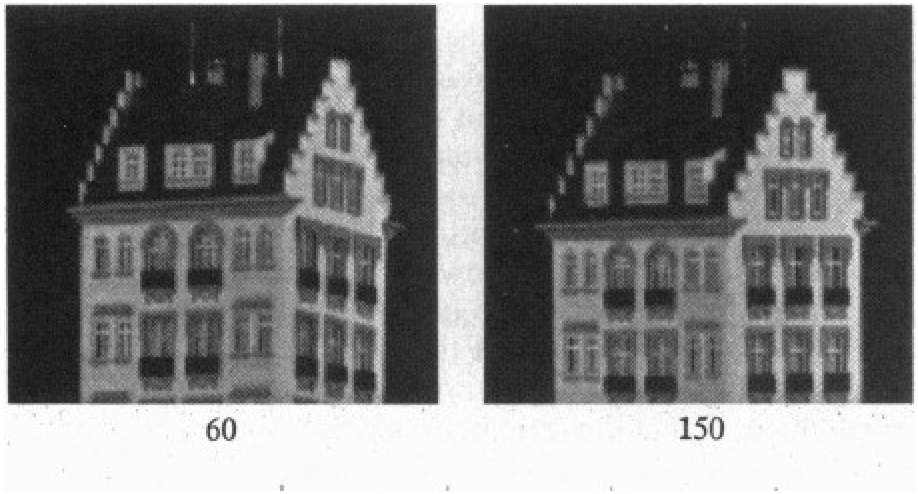
Coarse-to-fine optical flow estimation



Feature Tracking

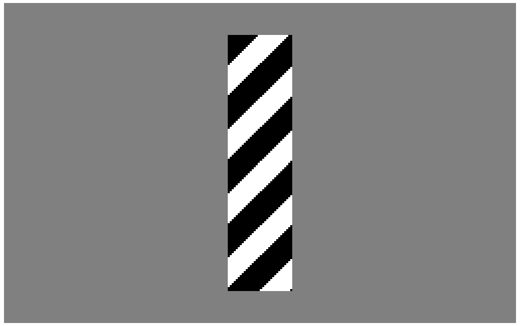
- Similar to feature matching, but track instead of match:
  - Track small, good features using translation only (u,v)
  - Use RANSAC to solve more complex motion model (Scale, Rotation, Similarity, Affine, Homography, ... Articulated, non-rigid)





Slide Credit:

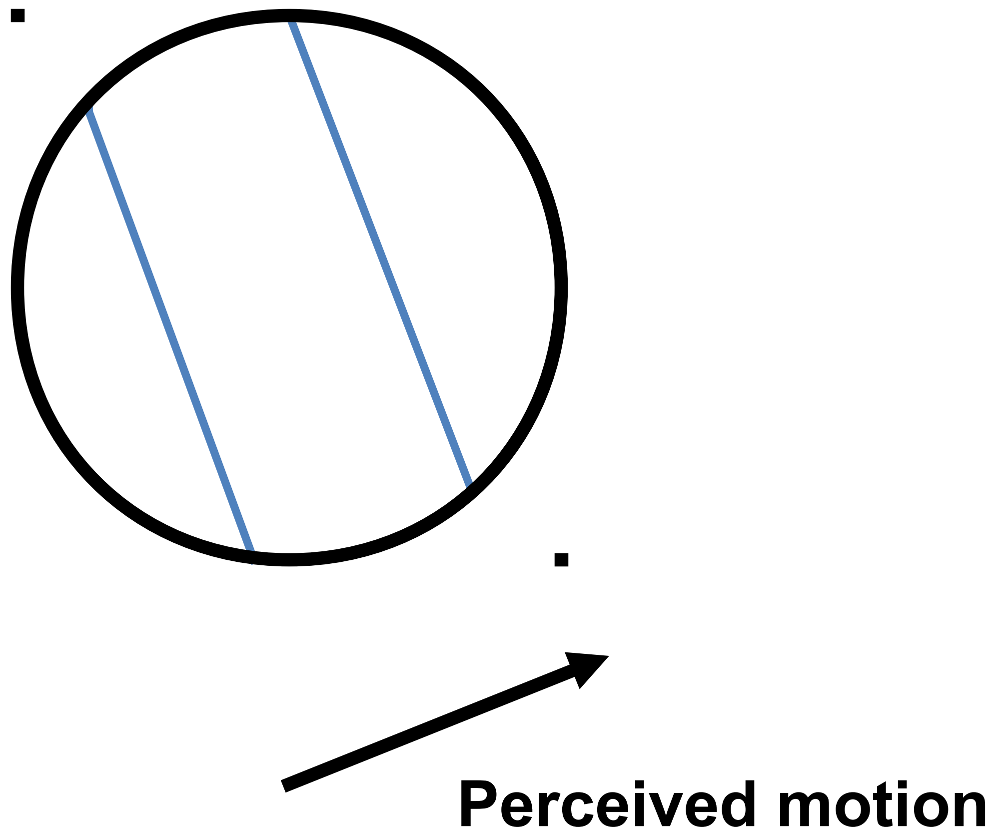
The barber pole illusion



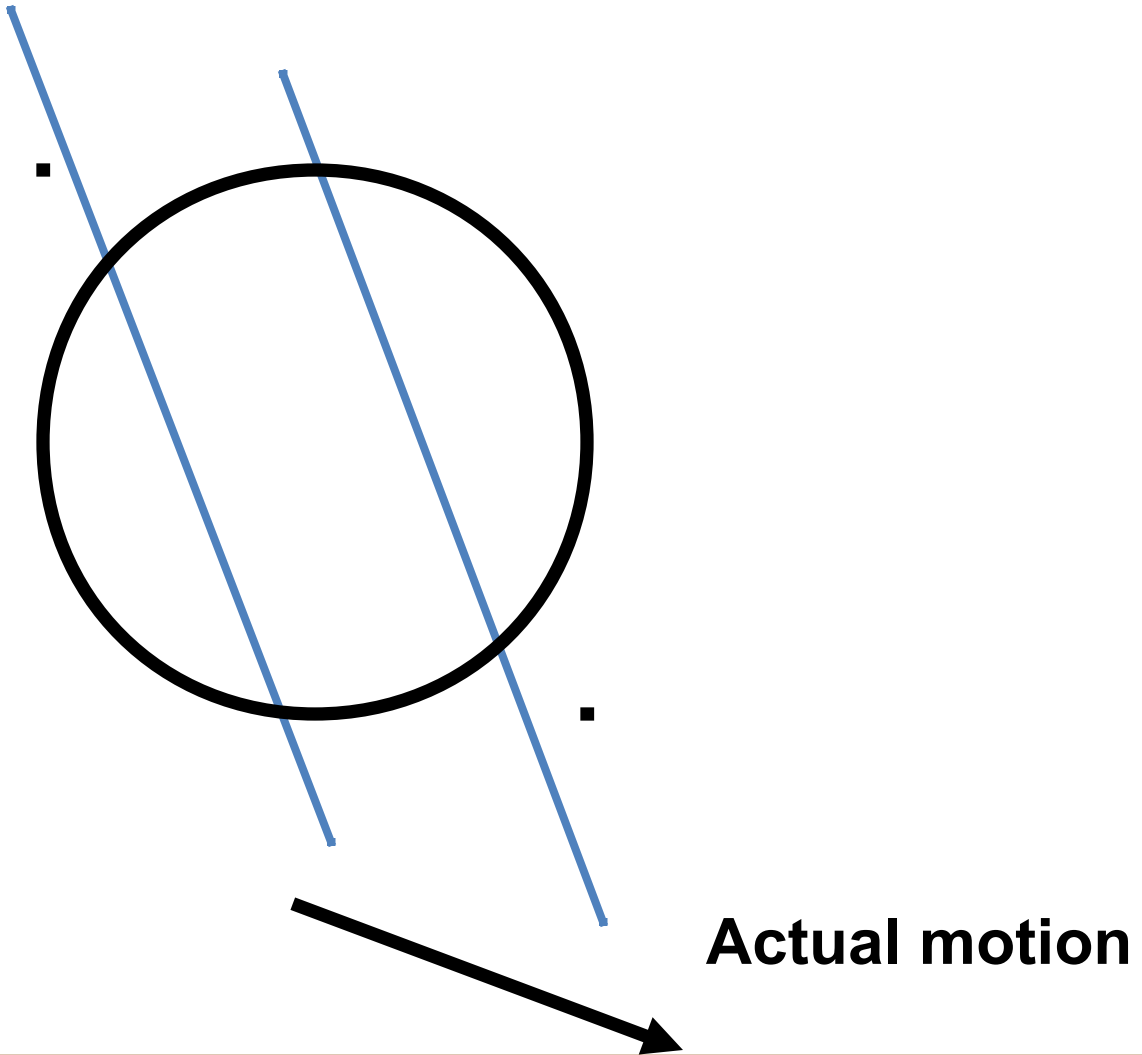
http://en.wikipedia.org/wiki/Barberpole_illusion

Slide Credit:

The aperture problem

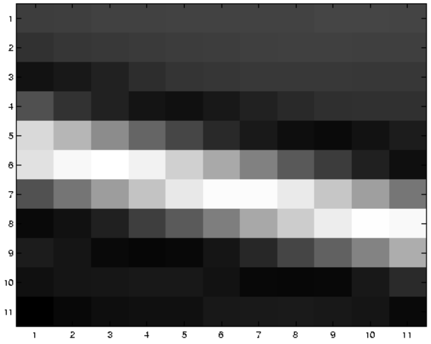


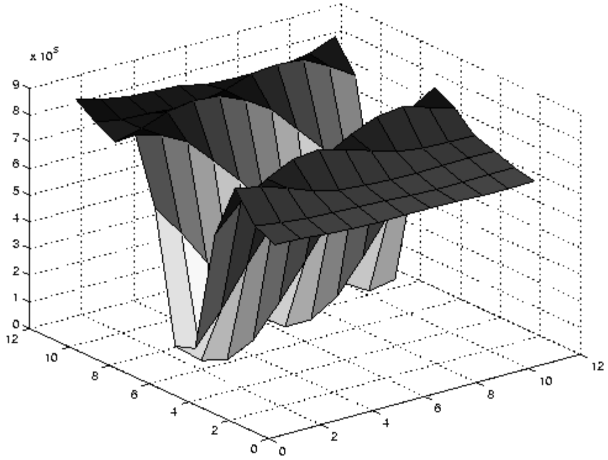
The aperture problem



Slide Credit:

Edges cause problems





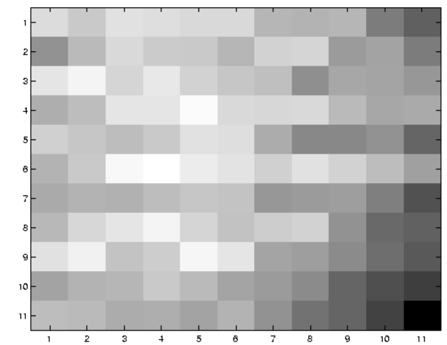
$$\sum \nabla I (\nabla I)^T$$

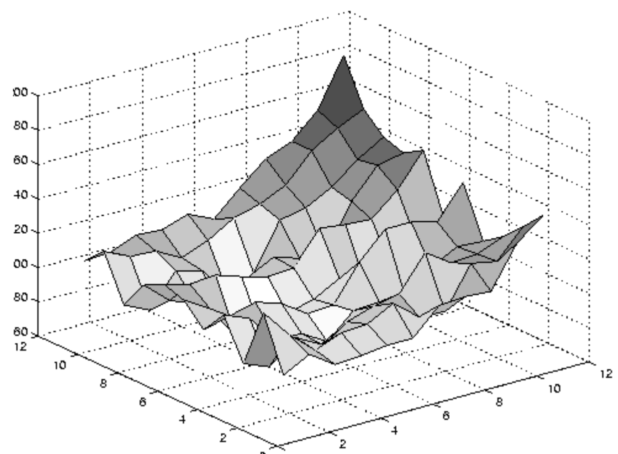
– large gradients, all the same

– large λ_1, small λ_2

Slide Credit:

Low texture regions don't work





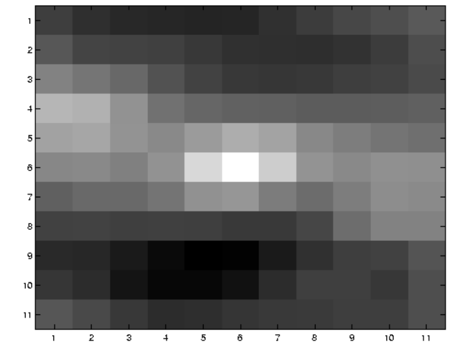
$$\sum \nabla I (\nabla I)^T$$

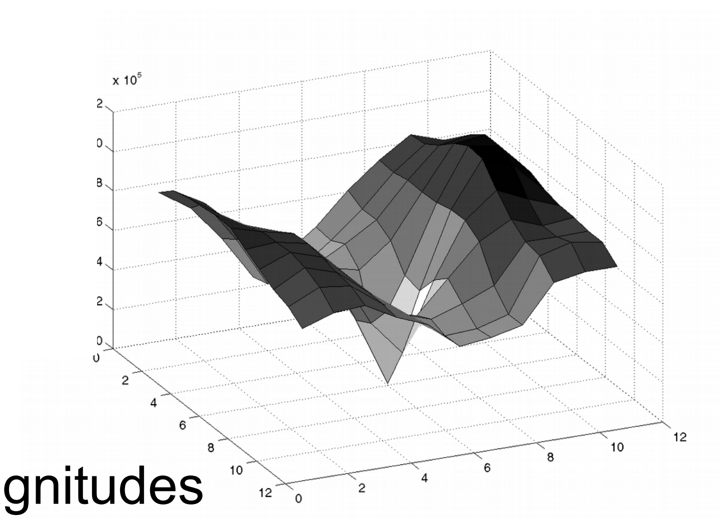
– gradients have small magnitude

– small λ_1, small λ_2

Slide Credit:

High textured region work best





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

– gradients are different, large magnitudes

– large λ_1, large λ_2

Conditions for solvability

Optimal (u, v) satisfies 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? I.e., what are good points to track?

- **A^TA** should be invertible
- **A^TA** should not be too small due to noise
  - 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)

Recall: This is the Harris Corner Detector!

Slide Credit:

Feature Point tracking

- Find a good point to track (harris corner)
- Track small patches (5x5 to 31x31) (e.g. using Lucas-Kanade)
- For rigid objects with affine motion: solve motion model parameters by robust estimation (RANSAC)

[Kanade, Lucas, Tamasi]
Slide Credit:

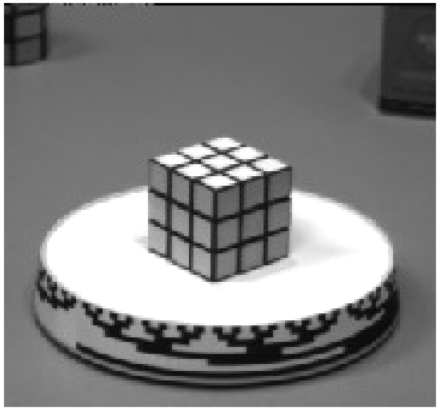
Implementation issues

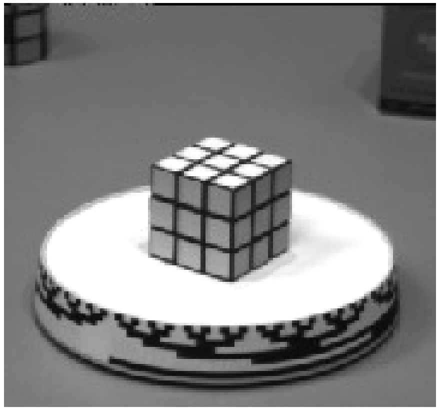
- Window size
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - 15x15 to 31x31 seems typical

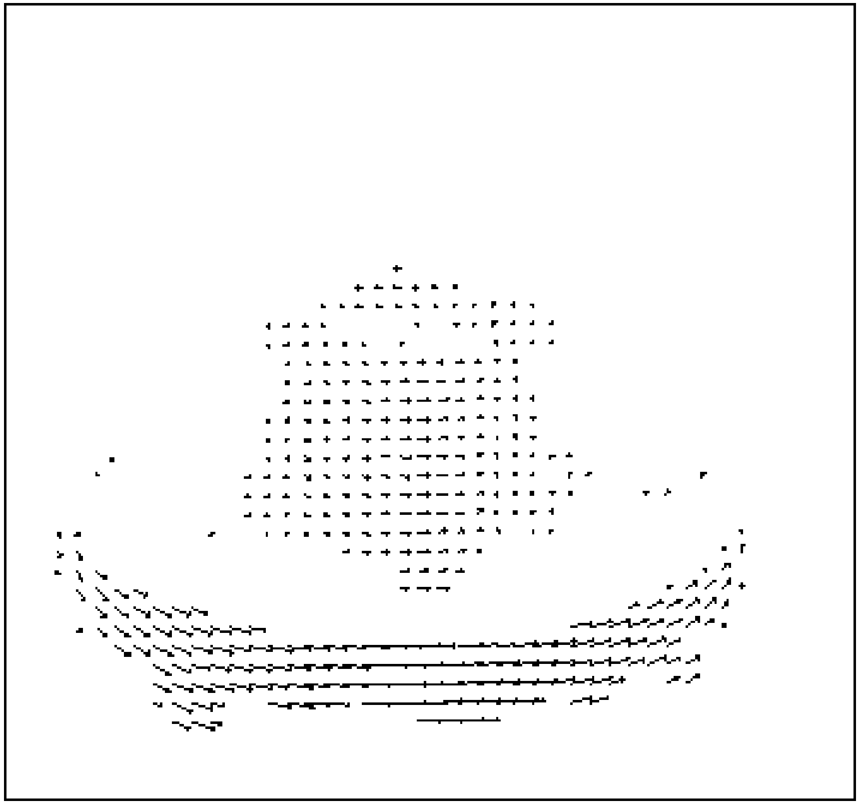
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)

Dense Motion field

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







Slide Credit:

Lucas-Kanade Optical Flow

- Same as Lucas-Kanade feature tracking, but densely 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

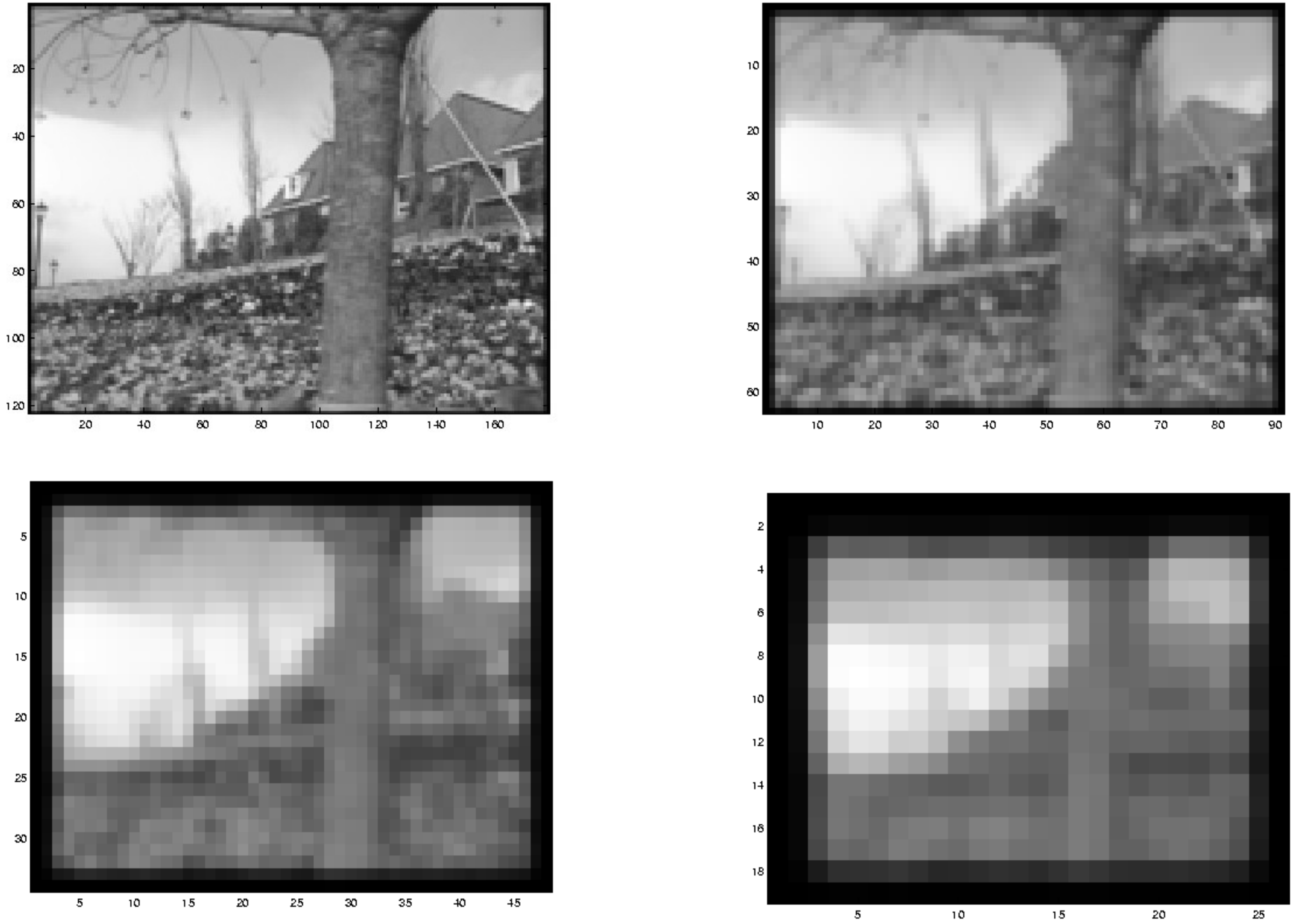
Slide Credit:

Example

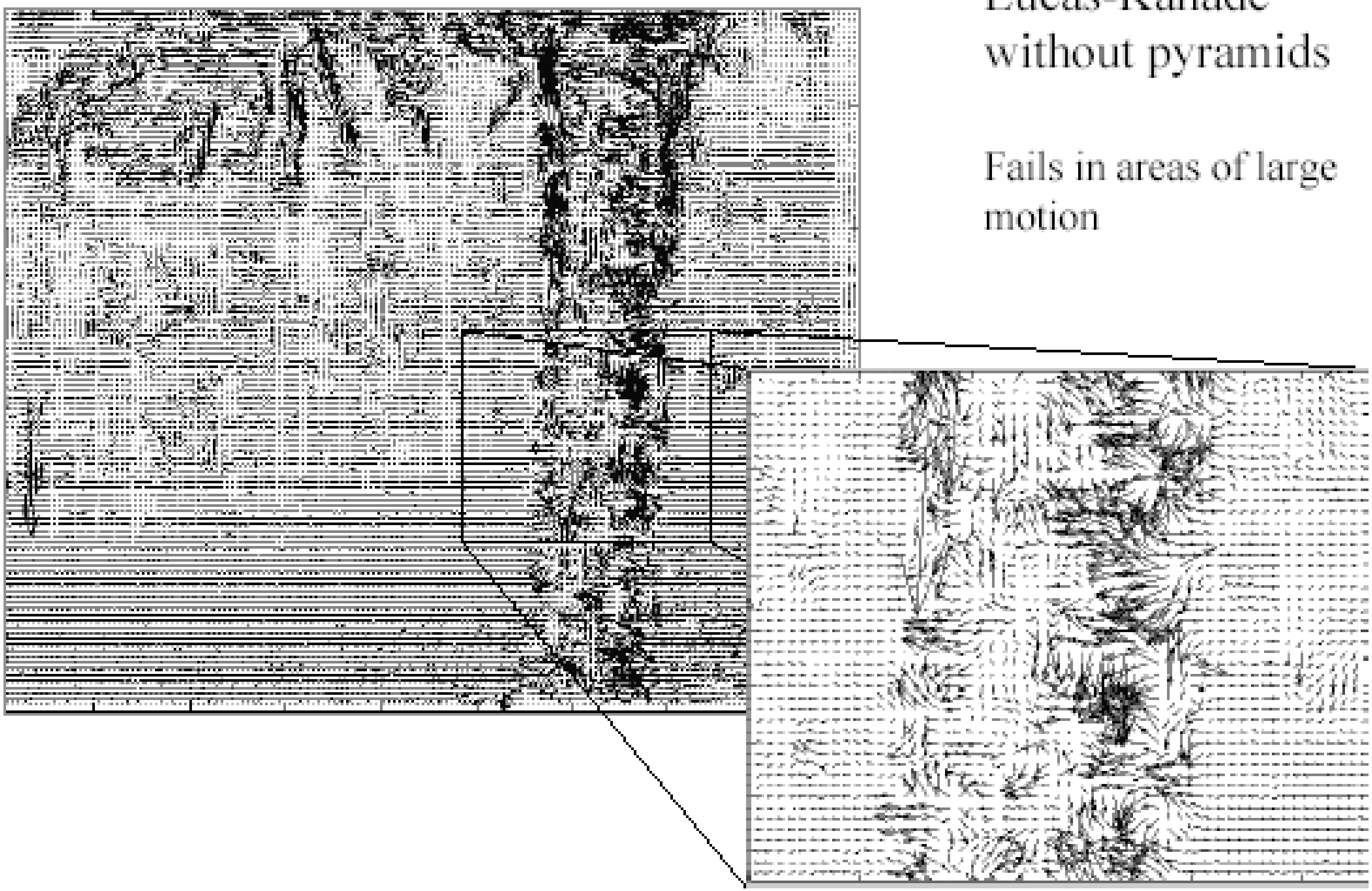
Slide Credit:

* From Khurram Hassan-Shafique CAP5415 Computer Vision 200

Multi-resolution registration



Optical Flow Results

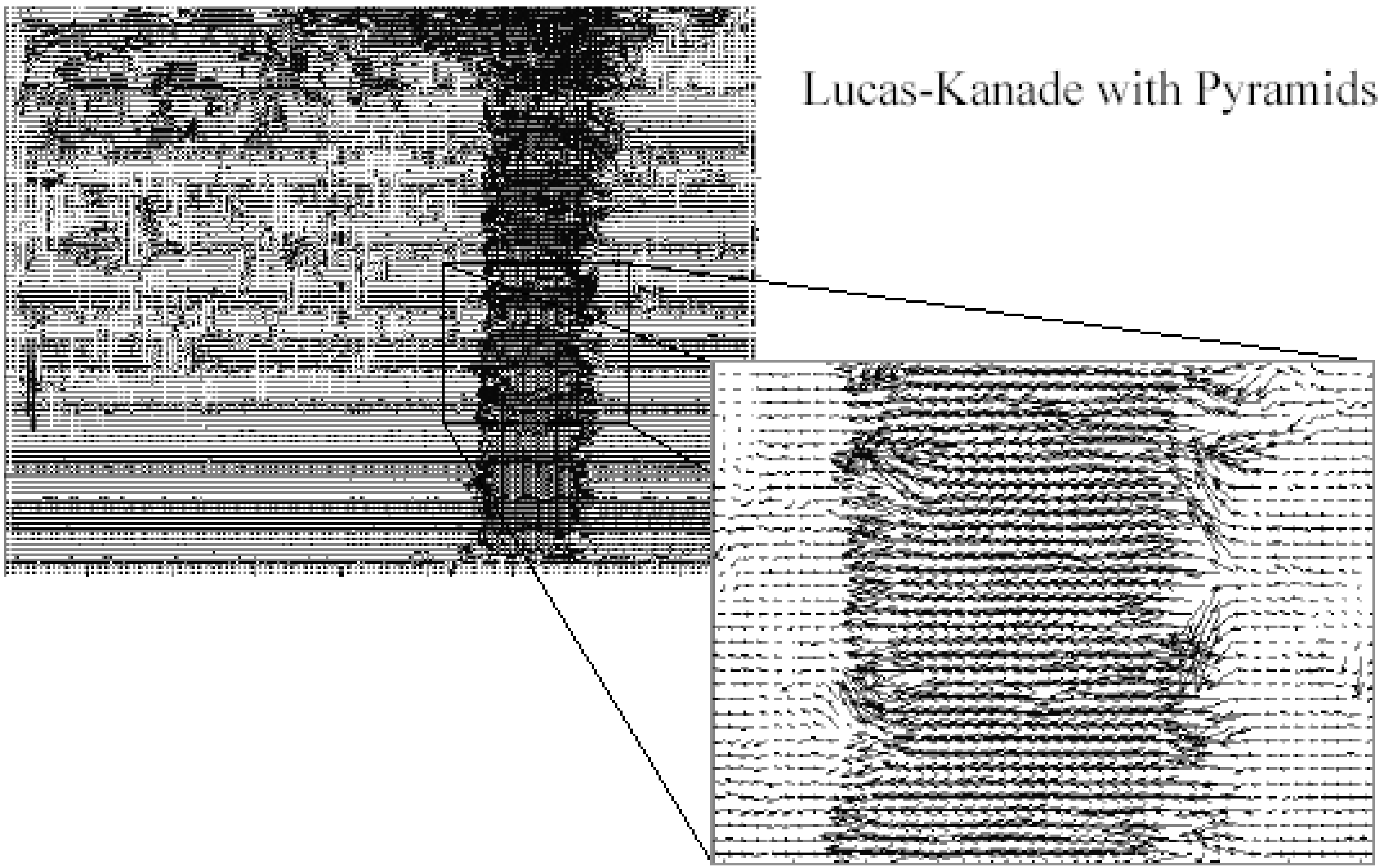


Lucas-Kanade
without pyramids

Fails in areas of large
motion

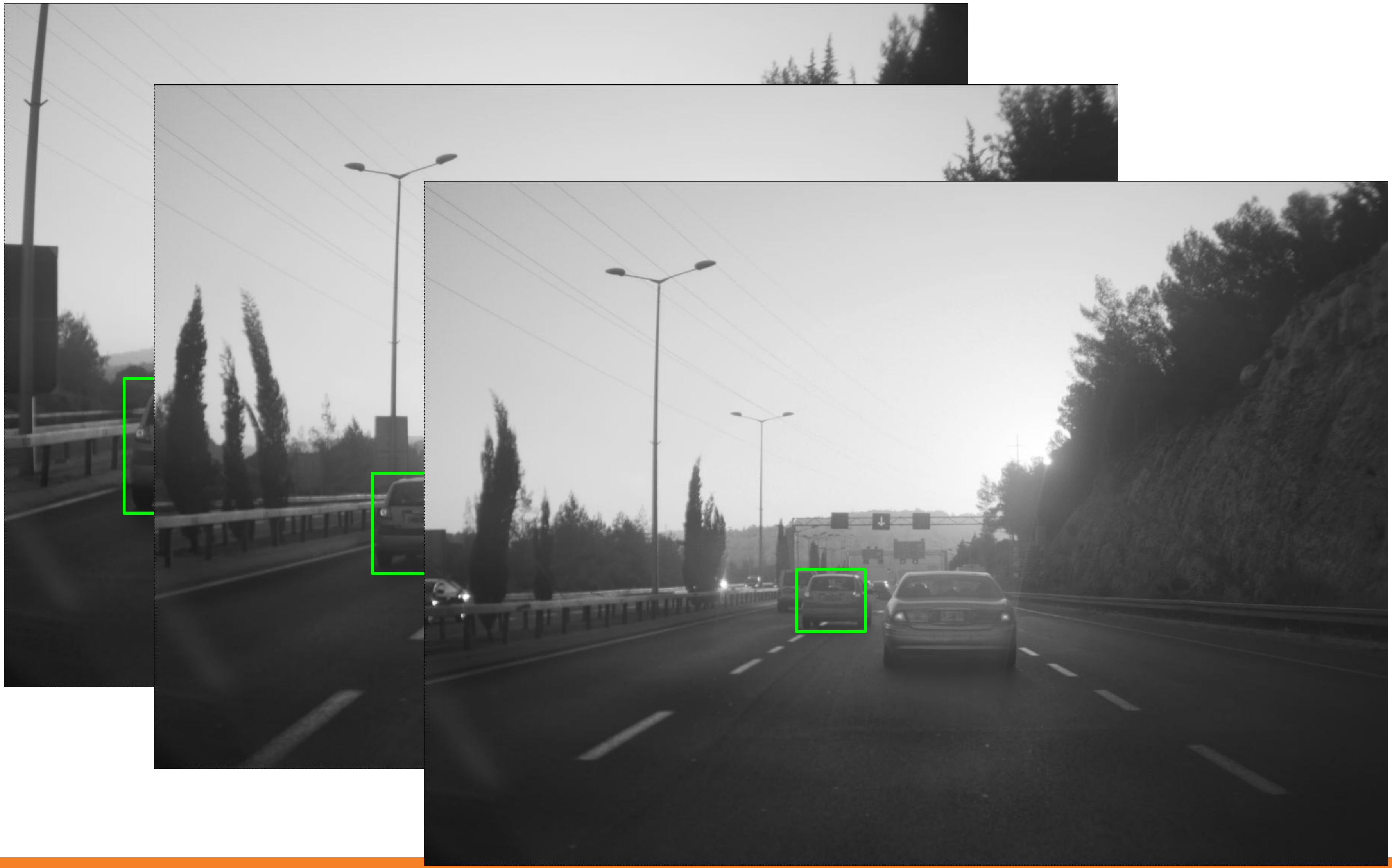
Slide Credit: * From Khurram Hassan-Shafique CAP5415 Computer Vision 20

Optical Flow Results



Slide Credit: * From Khurram Hassan-Shafique CAP5415 Computer Vision 20

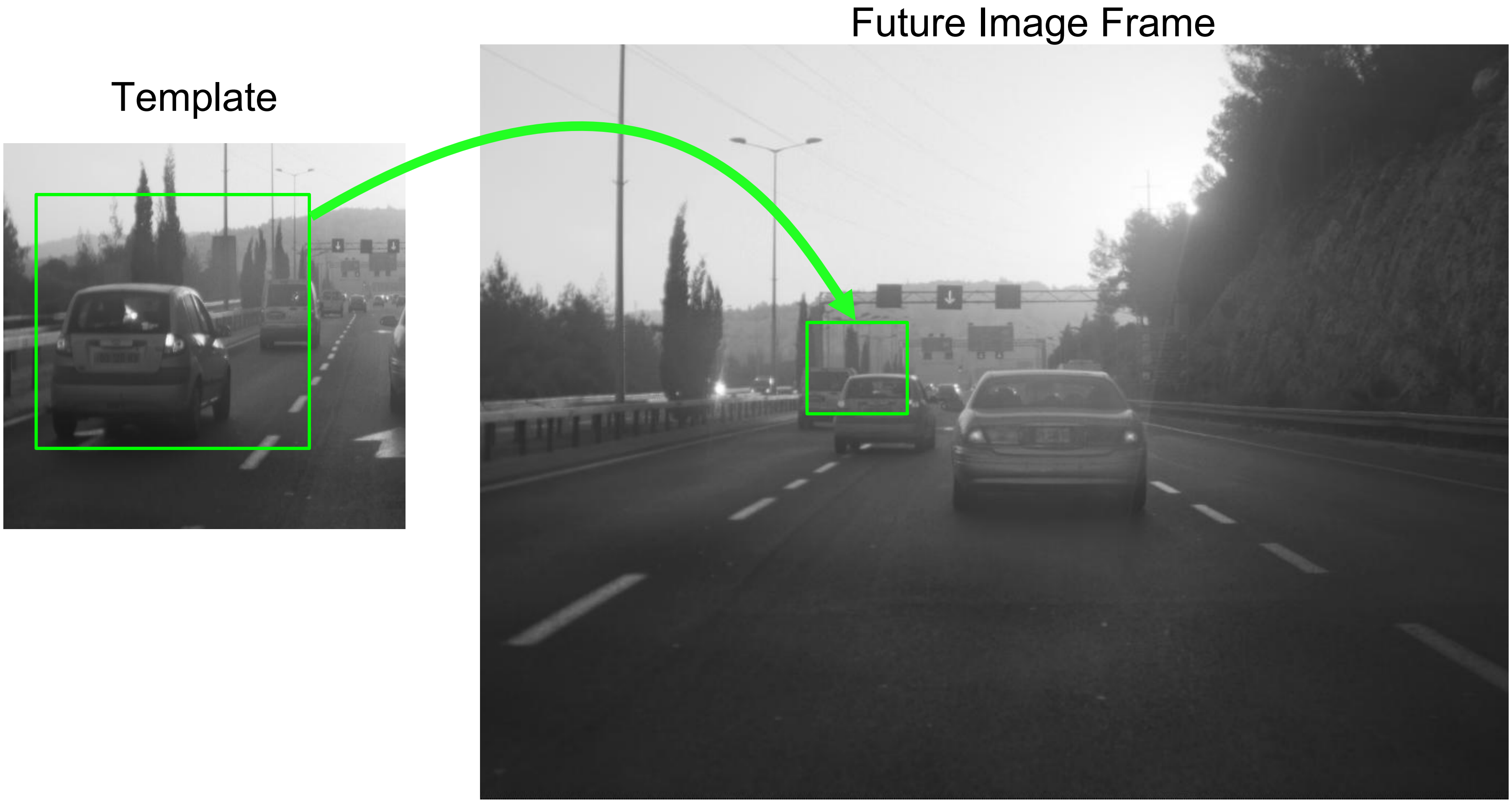
Object Detection and Tracking



Slide Credit:

Track

Once target has been located, and we “learn” what it looks like, should be easier to find in later frames... this is object tracking.



 Slide Credit:

Approaches to Object Tracking

- Motion model (translation, translation+scale, affine, non-rigid, …)
- Image representation (gray/color pixel, edge image, SIFT, HOG, wavelet...)
- Distance metric (L1, L2, normalized correlation, Chi-Squared, …)
- Method of optimization (gradient descent, naive search, combinatoric search...)
- What is tracked: whole object or selected features

Template

 Slide Credit:

Approaches to Object Tracking

Can we use Lukas-Kanade?

Yes, but need to deal with more than translation (u,v)

=> Affine Motion

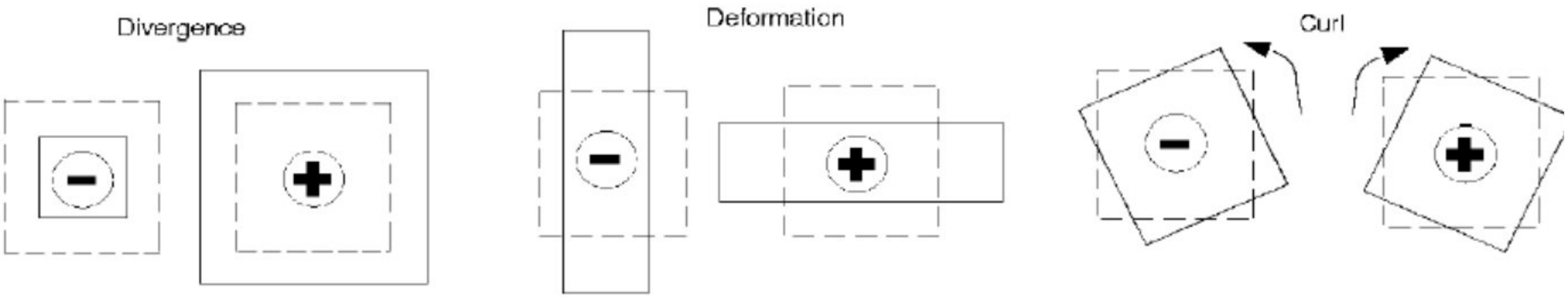
Template

Slide Credit:

Affine Motion

$$E(\mathbf{a}) = \sum_{x,y \in R} (\nabla I^T \mathbf{u}(\mathbf{x};\mathbf{a}) + I_t)^2$$

$$\mathbf{u}(\mathbf{x};\ \mathbf{a}) = \begin{bmatrix} u(\mathbf{x};\mathbf{a}) \\ v(\mathbf{x};\mathbf{a}) \end{bmatrix} = \begin{bmatrix} a_1 + a_2 x + a_3 y \\ a_4 + a_5 x + a_6 y \end{bmatrix}$$



Slide Credit: Michael Black

Affine Motion

$$\begin{bmatrix} x \\ y \end{bmatrix}^* = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_6 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^* = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous coordinates

Linear transformation

Translation

Slide Credit: Michael Black

Affine Motion Optimization

$$E(\mathbf{a}) = \sum_{x,y \in R} (I_x u + I_y v + I_t)^2$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$E(\mathbf{a}) = \sum_{x,y \in R} (I_x a_1 x + I_x a_2 y + I_x a_3 + I_y a_4 x + I_y a_5 y + I_y a_6 + I_t)^2$$

Slide Credit: Michael Black

Recall: Solving for free parameters (u,v)

- Over-constrained linear system

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

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

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

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

Slide Credit:

Affine Motion Optimization

$$E(\mathbf{a}) = \sum_{x,y \in R} (I_x a_1 x + I_x a_2 y + I_x a_3 + I_y a_4 x + I_y a_5 y + I_y a_6 + I_t)^2$$

Differentiate wrt the a_i and set equal to zero.

$$\begin{bmatrix} \Sigma I_x^2 x^2 & \Sigma I_x^2 xy & \Sigma I_x^2 x & \Sigma I_x I_y x^2 & \Sigma I_x I_y xy & \Sigma I_x I_y x \\ \Sigma I_x^2 xy & \Sigma I_x^2 y^2 & \Sigma I_x^2 y & \Sigma I_x I_y xy & \Sigma I_x I_y y^2 & \Sigma I_x I_y y \\ & & & \vdots & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} -\Sigma I_x I_t x \\ -\Sigma I_x I_t y \\ -\Sigma I_x I_t \\ -\Sigma I_y I_t x \\ -\Sigma I_y I_t y \\ -\Sigma I_y I_t \end{bmatrix}$$

Slide Credit: Michael Black

Summary

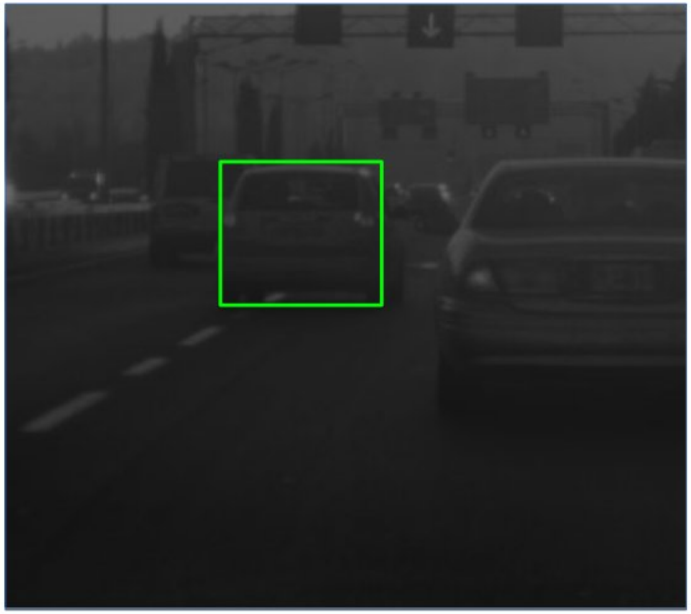
- L-K works well when:
  - Have a good initial guess
  - L2 (SSD) is a good metric
  - Can handle more degrees of freedom in motion model (scale, rotation, affine, etc.), which are too expensive for search
- But has problems with:
  - Changes in brightness
  - ...

Slide Credit:

LK Problem: Change in Brightness

Possible Solutions:

- Subtract mean intensity (based on current estimate before iteration)
- Transform gray values into some features that are not effected by brightness
  - Any filter that is zero-mean
  - Example: vertical, horizontal edge filters
  - Example: Non-parametric filters (Rank, Census Transforms)



Slide Credit:

More Problems

- Outliers: bright strong features that are wrong

- Complex, high dimensional, or non-rigid motion

Slide Credit: