RWTH AACHEN UNIVERSITY

Computer Vision II – Lecture 3

Template Tracking

24.04.2014

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de

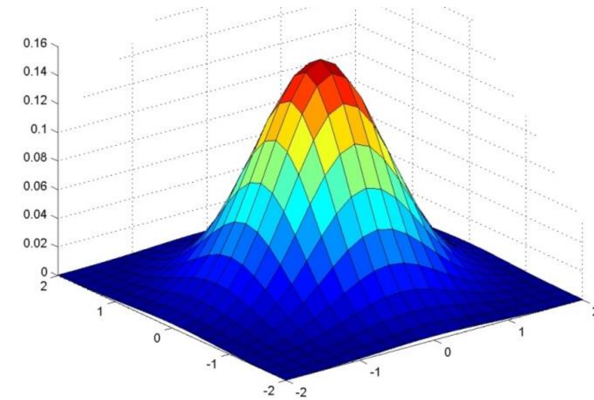
Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Articulated Tracking

Image source: Robert Collins

Recap: Gaussian Background Model

- Statistical model
  - Value of a pixel represents a measurement of the radiance of the first object intersected by the pixel's optical ray.
  - With a static background and static lighting, this value will be a constant affected by i.i.d. Gaussian noise.



- Idea
  - Model the background distribution of each pixel by a single Gaussian centered at the mean pixel value:

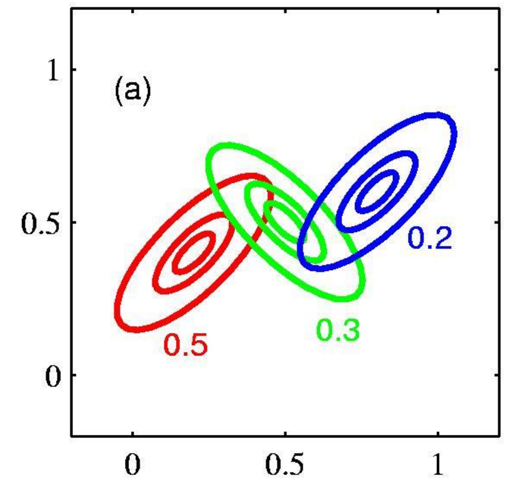
$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

  - Test if a newly observed pixel value has a high likelihood under this Gaussian model.

$\Rightarrow$ Automatic estimation of a sensitivity threshold for each pixel.

MoG Background Model

- Improved statistical model
  - Large jumps between different pixel values because different objects are projected onto the same pixel at different times.
  - While the same object is projected onto the pixel, small local intensity variations due to Gaussian noise.



- Idea
  - Model the color distribution of each pixel by a mixture of K Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  - Evaluate likelihoods of observed pixel values under this model.
  - Or let entire Gaussian components adapt to foreground objects and classify components as belonging to object or background.

Image source: Chris Bischop

Recap: Stauffer-Grimson Background Model

- **Idea**
  - Model the distribution of each pixel by a mixture of K Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \text{where} \quad \boldsymbol{\Sigma}_k = \sigma_k^2 \mathbf{I}$$

  - Check every new pixel value against the existing K components until a match is found (pixel value within $2.5\ \sigma_k$ of $\boldsymbol{\mu}_k$).
  - If a match is found, adapt the corresponding component.
  - Else, replace the least probable component by a distribution with the new value as its mean and an initially high variance and low prior weight.
  - Order the components by the value of w_k/σ_k and select the best B components as the background model, where

$$B = \arg\min_b \left(\sum_{k=1}^{b} \frac{w_k}{\sigma_k} > T \right)$$

[C. Stauffer, W.E.L. Grimson, CVPR'99]

Recap: Stauffer-Grimson Background Model

- **Online adaptation**
  - **Instead of estimating the MoG using EM, use a simpler online adaptation, assigning each new value only to the matching component.**
  - **Let $M_{k,t} = 1$ iff component k is the model that matched, else 0.**

$$\pi_k^{(t+1)} = (1-\alpha)\pi_k^{(t)} + \alpha M_{k,t}$$

  - **Adapt only the parameters for the matching component**

$$\boldsymbol{\mu}_k^{(t+1)} = (1-\rho)\boldsymbol{\mu}_k^{(t)} + \rho x^{(t+1)}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = (1-\rho)\boldsymbol{\Sigma}_k^{(t)} + \rho(x^{(t+1)} - \boldsymbol{\mu}_k^{(t+1)})(x^{(t+1)} - \boldsymbol{\mu}_k^{(t+1)})^T$$

where

$$\rho = \alpha \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

(i.e., the update is weighted by the component likelihood)

[C. Stauffer, W.E.L. Grimson, CVPR'99]

Recap: Kernel Background Modeling

- **Nonparametric density estimation**
  - **Estimate a pixel's background distribution using the kernel density estimator $K(\cdot)$ as**

$$p(\mathbf{x}^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} K(\mathbf{x}^{(t)} - \mathbf{x}^{(i)})$$

  - **Choose K to be a Gaussian $\mathcal{N}(0, \mathbf{\Sigma})$ with $\mathbf{\Sigma} = \mathrm{diag}\{\sigma_j\}$. Then**

$$p(\mathbf{x}^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\frac{(x_j^{(t)} - x_j^{(i)})^2}{\sigma_j^2}}$$

  - **A pixel is considered foreground if $p(\mathbf{x}^{(t)}) < \theta$ for a threshold θ.**
    - This can be computed very fast using lookup tables for the kernel function values, since all inputs are discrete values.
    - Additional speedup: partial evaluation of the sum usually sufficient

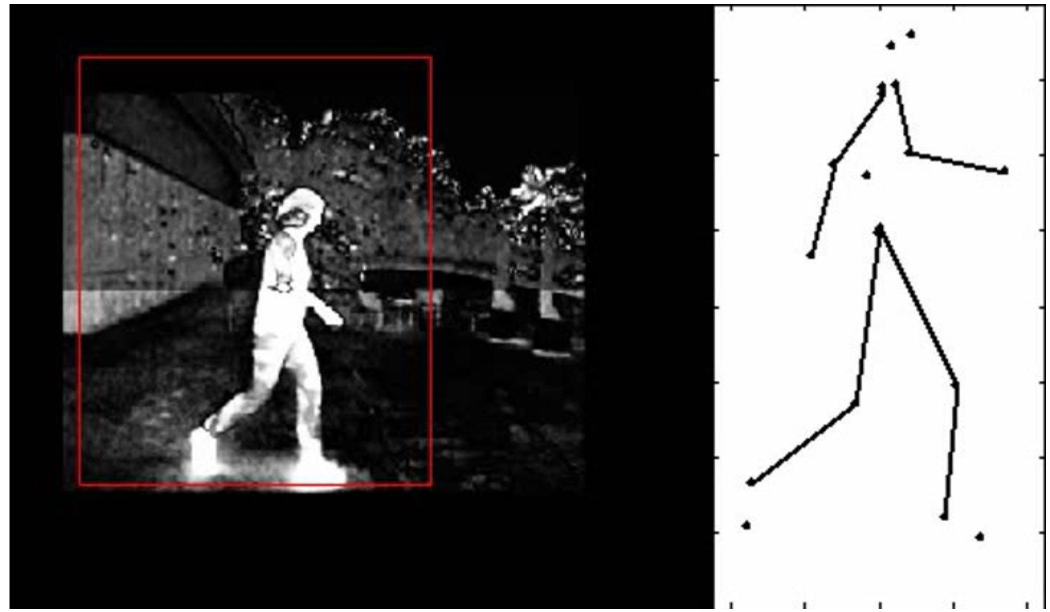
[A. Elgammal, D. Harwood, L. Davis, ECCV'00]

Applications: Visual Surveillance

- **Background modeling to detect objects for tracking**
  - **Extension: Learning a foreground model for each object.**

Video source: Ian Reid, Univ. of Oxford

Applications: Articulated Tracking



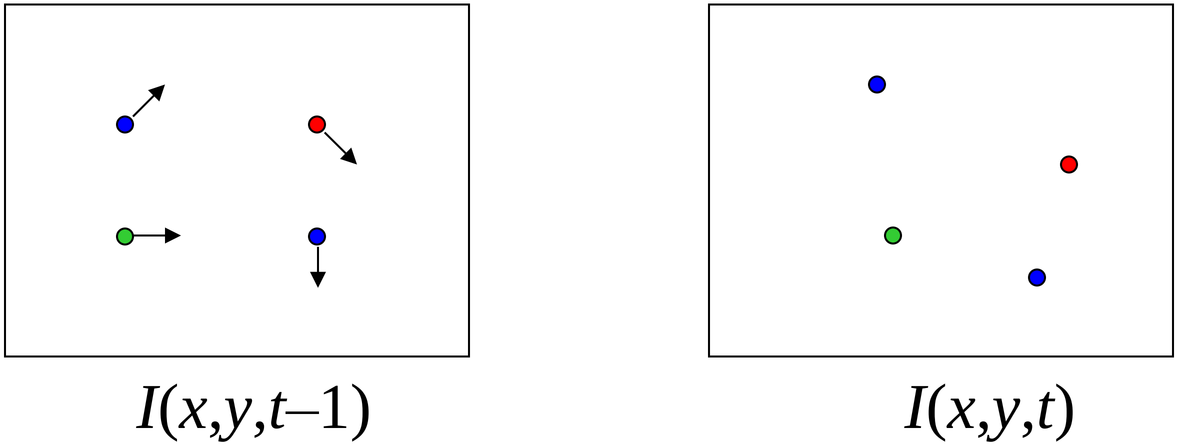
- Background modeling as preprocessing step
  - Track a person's location through the scene
  - Extract silhouette information from the foreground mask.
  - Perform body pose estimation based on this mask.

Video source: Hedvik Kjellstroem, Tobias Jaeggli

Topics of This Lecture

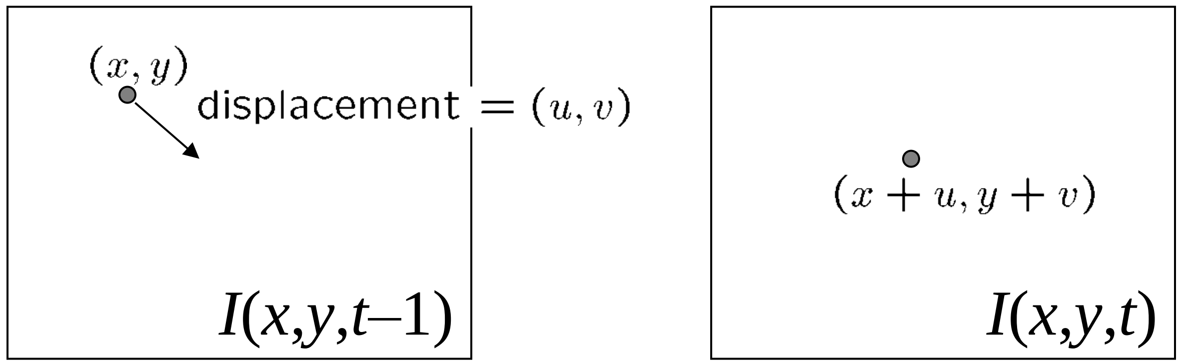
- Recap: Lucas-Kanade Optical Flow
  - Brightness Constancy constraint
  - LK flow estimation
  - Coarse-to-fine estimation
- Feature Tracking
  - KLT feature tracking
- Template Tracking
  - LK derivation for templates
  - Warping functions
  - General LK image registration
- Applications

Recap: Estimating Optical Flow



- Optical Flow
  - Given two subsequent frames, estimate the apparent motion field $u(x,y)$ and $v(x,y)$ 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.

Recap: 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 hand side using Taylor expansion:**

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

- **Hence,** $I_x \cdot u + I_y \cdot v + I_t \approx 0$

Spatial derivatives (I_x, I_y) **Temporal derivative** (I_t)

Recap: The Brightness Constancy Constraint

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

- How many equations and unknowns per pixel?
  - One equation, two unknowns

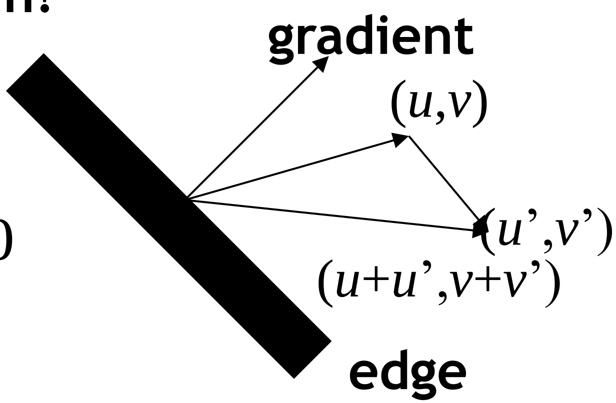
- Intuitively, what does this constraint mean?

$$\nabla I \cdot (u, v) + I_t = 0$$

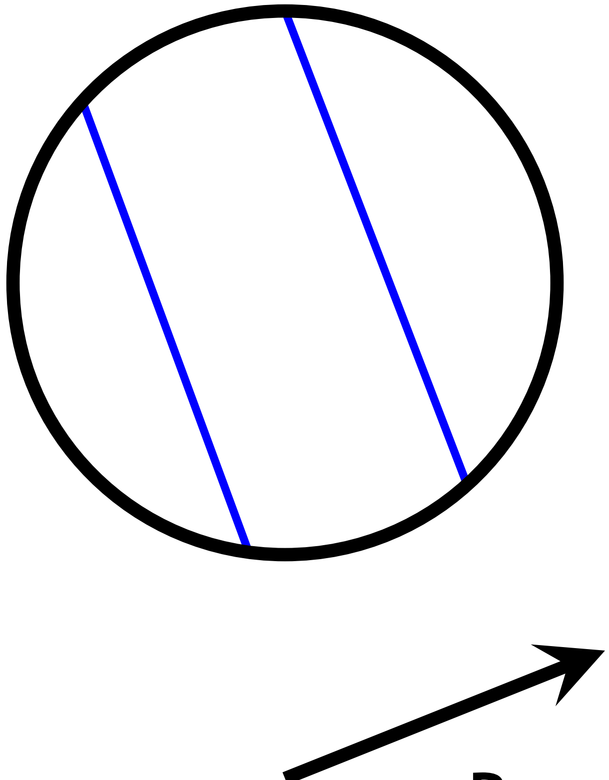
  - It gives us a constraint on the component of the flow in the direction of the gradient.

⇒ The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown!

If (u,v) satisfies the equation,
so does $(u+u', v+v')$ if $\nabla I \cdot (u', v') = 0$

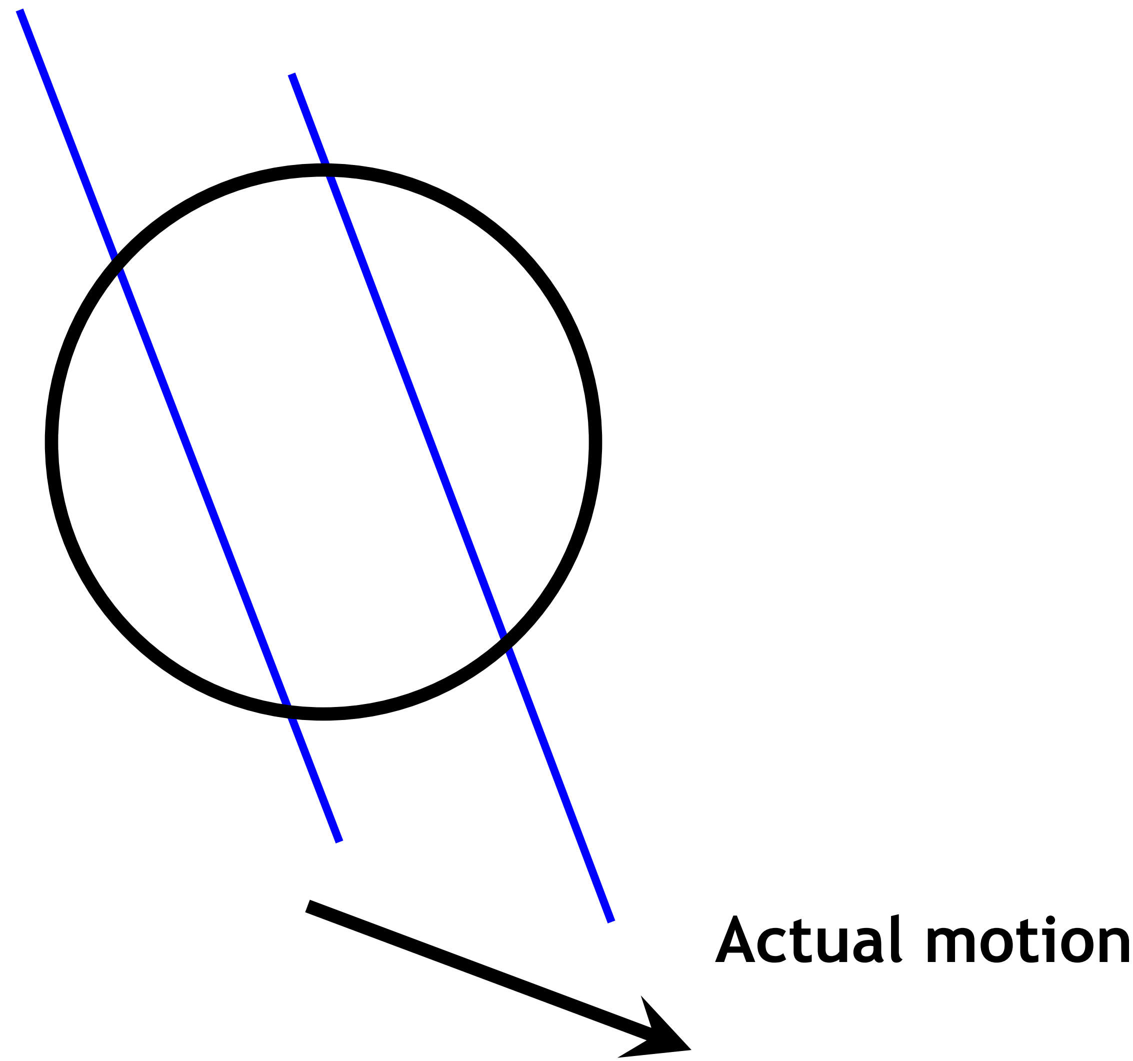


The Aperture Problem

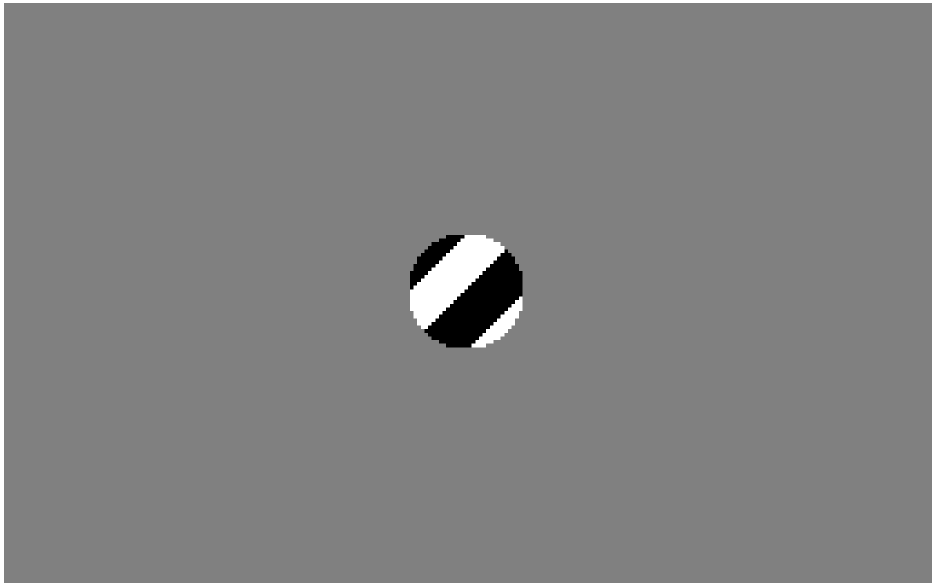


Slide credit: Svetlana Lazebnik

The Aperture Problem



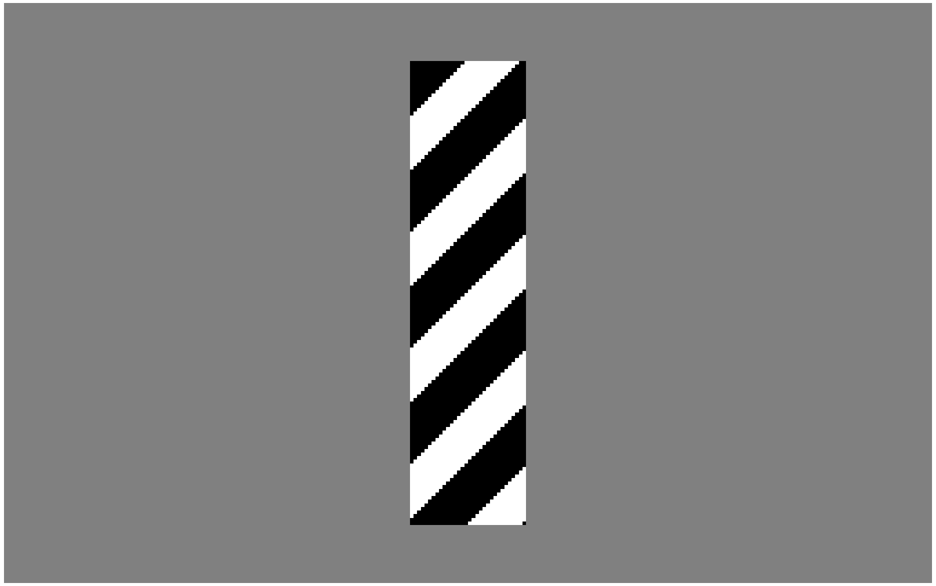
The Barber Pole Illusion



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

Slide credit: Svetlana Lazebnik

The Barber Pole Illusion



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

The Barber Pole Illusion

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

Recap: Solving the Aperture Problem

- How to get more equations for a pixel?
- Spatial coherence constraint
  - Pretend the pixel's neighbors have the same (u,v).
  - If we use a 5×5 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 *Proc. IJCAI'81*, pp. 674-679, 1981.

Recap: Solving the Aperture Problem

- **Least squares problem:**

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

- **Minimum least squares solution given by solution of**

$$\underset{2\mathrm{x}2}{(A^T A)} \; \underset{2\mathrm{x}1}{d} = \underset{2\mathrm{x}1}{A^T b}$$

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

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

Recap: 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?**
  - $A^T A$ **should be invertible.**
  - $A^T A$ **entries should not be too small (noise).**
  - $A^T A$ **should be well-conditioned.**

  $\Rightarrow$ **Looking for cases where A has two large eigenvalues (i.e., corners and highly textured areas).**

Recap: Iterative LK Refinement

1. Estimate velocity at each pixel using one iteration of LK estimation.

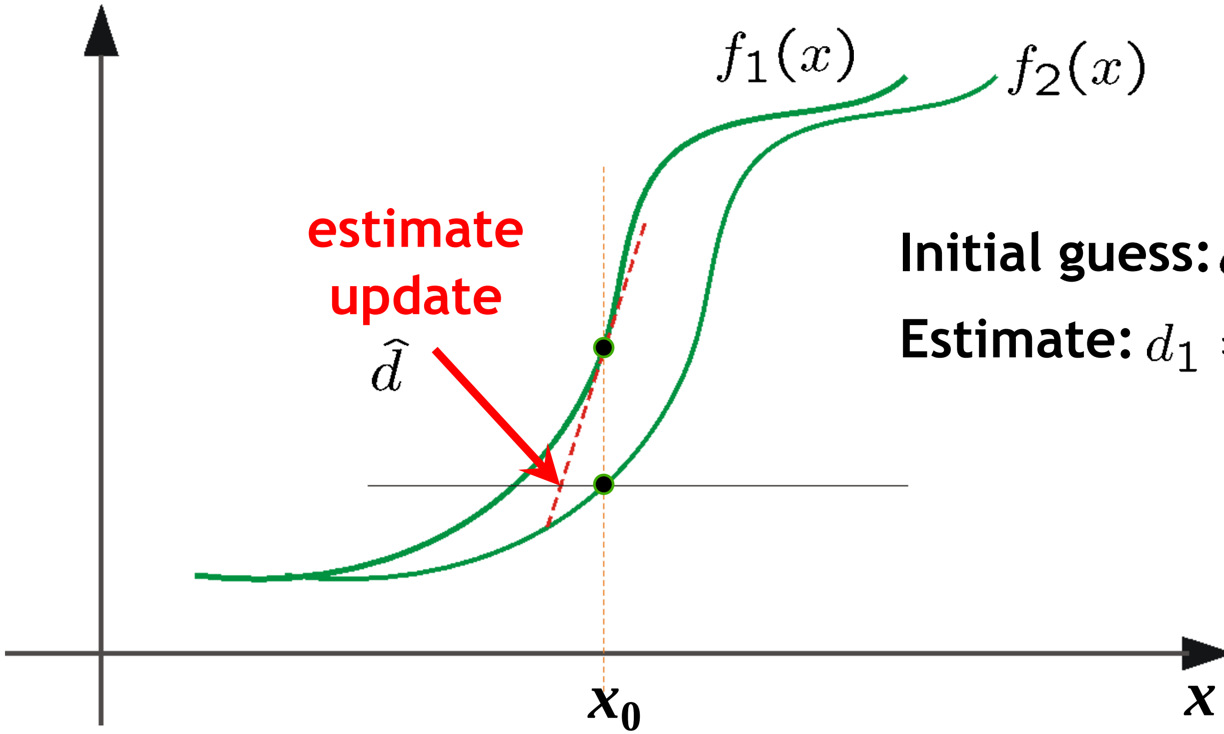
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

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

2. Warp one image toward the other using the estimated flow field.
   - *(Easier said than done)*

3. Refine estimate by repeating the process.

Recap: Iterative LK Refinement

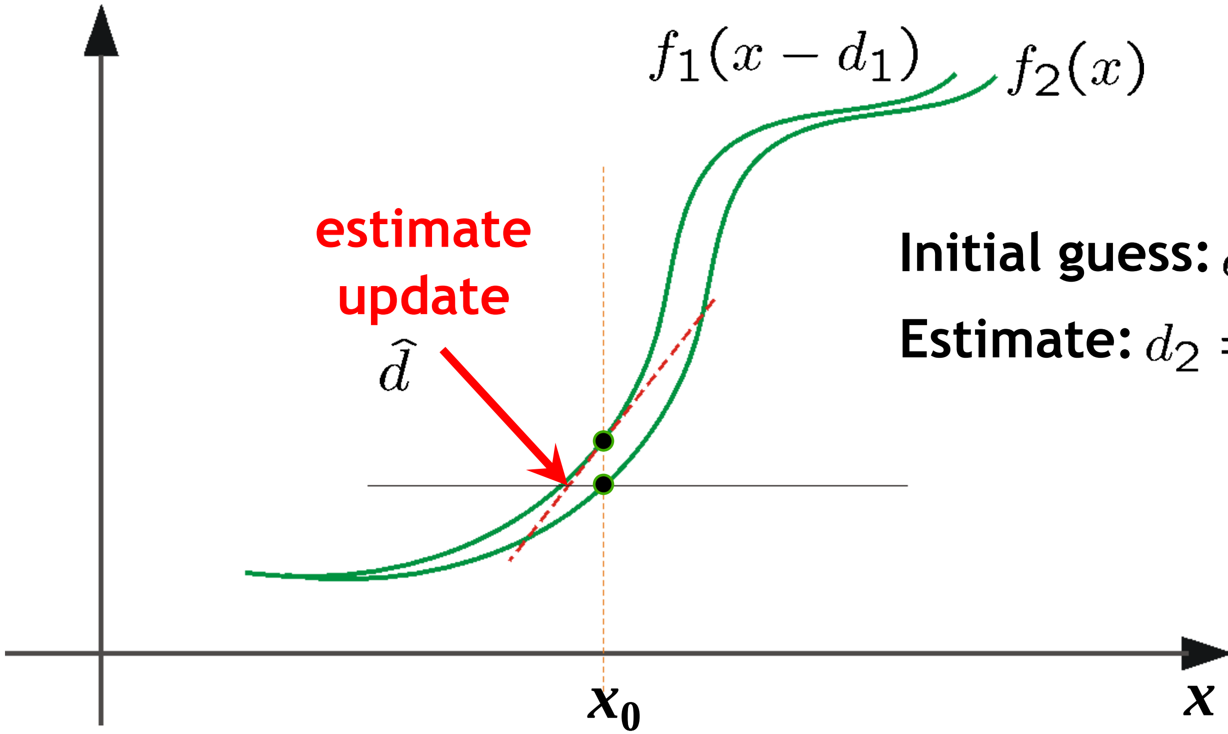


Initial guess: $d_0 = 0$

Estimate: $d_1 = d_0 + \hat{d}$

(using d for *displacement* here instead of u)

Recap: Iterative LK Refinement

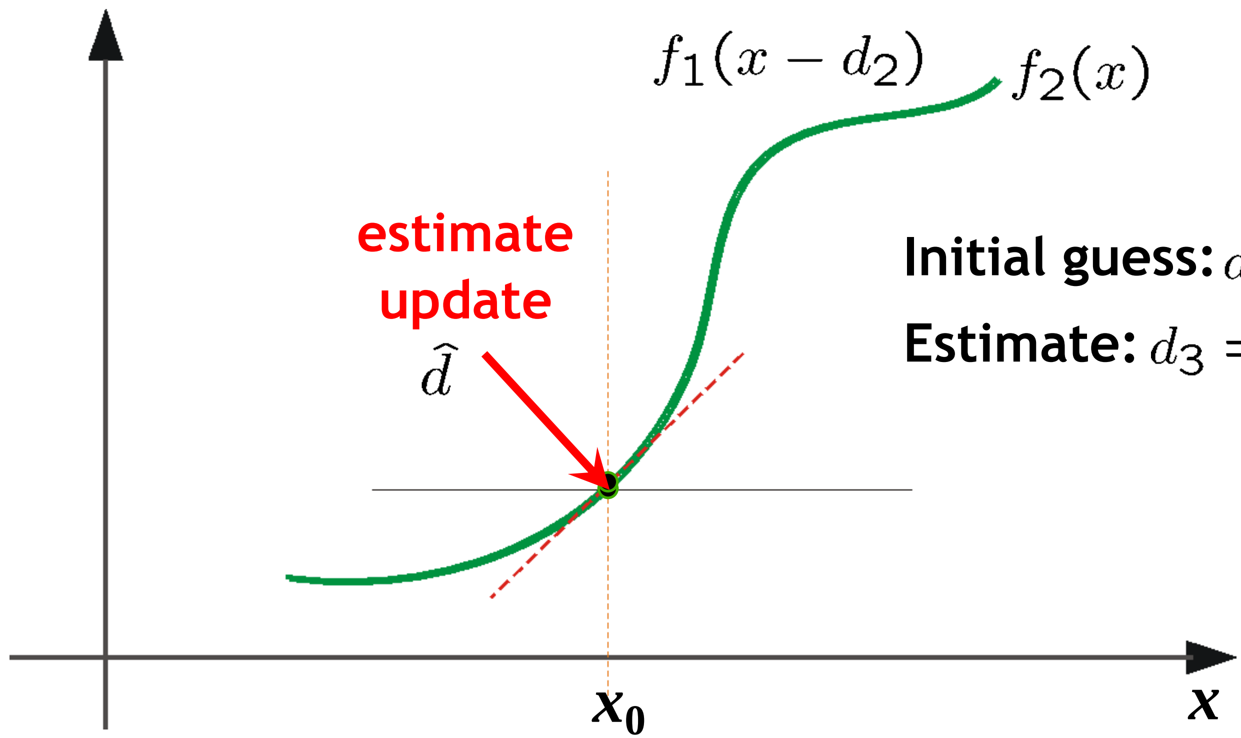


Initial guess: d_1

Estimate: $d_2 = d_1 + \hat{d}$

(using d for *displacement* here instead of u)

Recap: Iterative LK Refinement

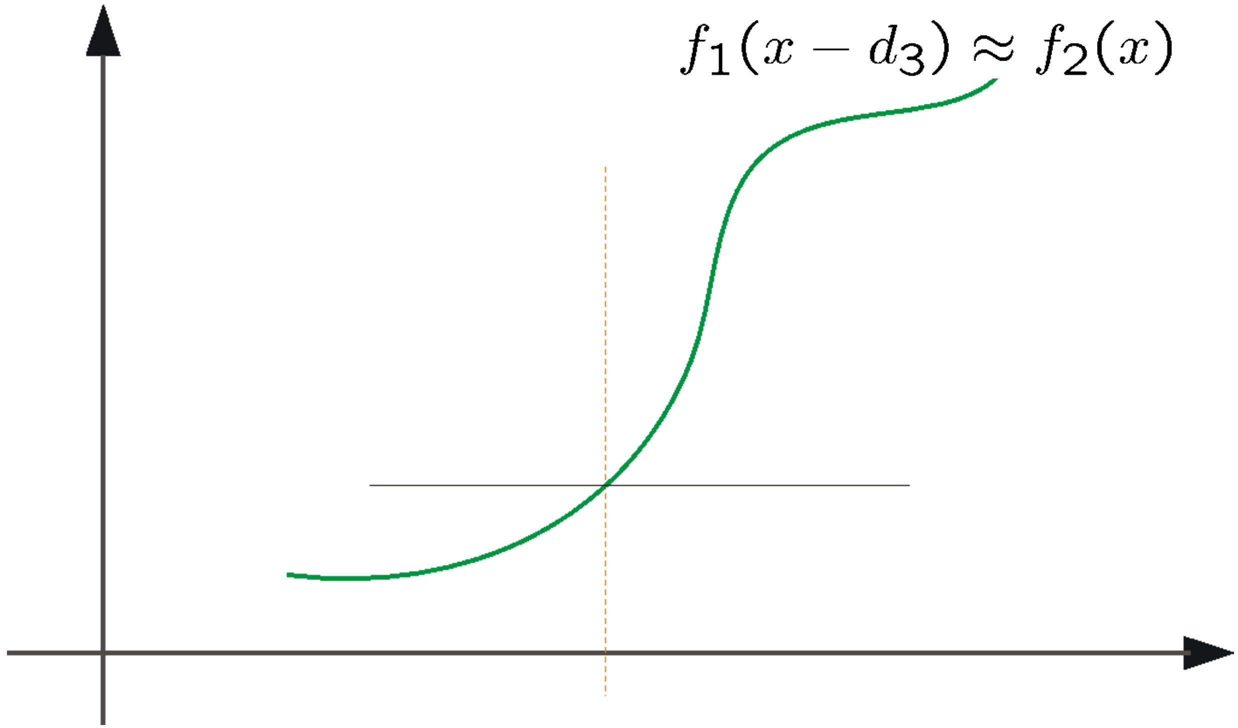


Initial guess: d_2

Estimate: $d_3 = d_2 + \hat{d}$

(using d for *displacement* here instead of u)

Recap: Iterative LK Refinement



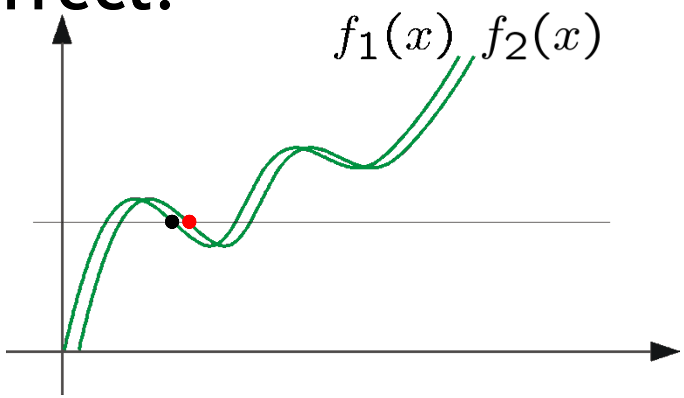
(using d for *displacement* here instead of u)

Problem Case: Large Motions

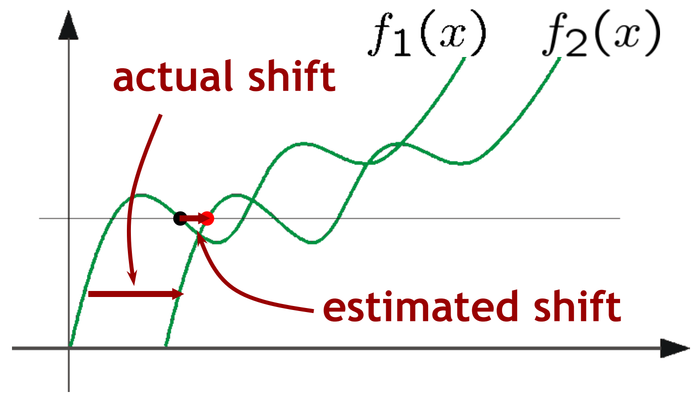
Slide credit: Svetlana Lazebnik

Temporal Aliasing

- Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity.
- I.e., how do we know which 'correspondence' is correct?



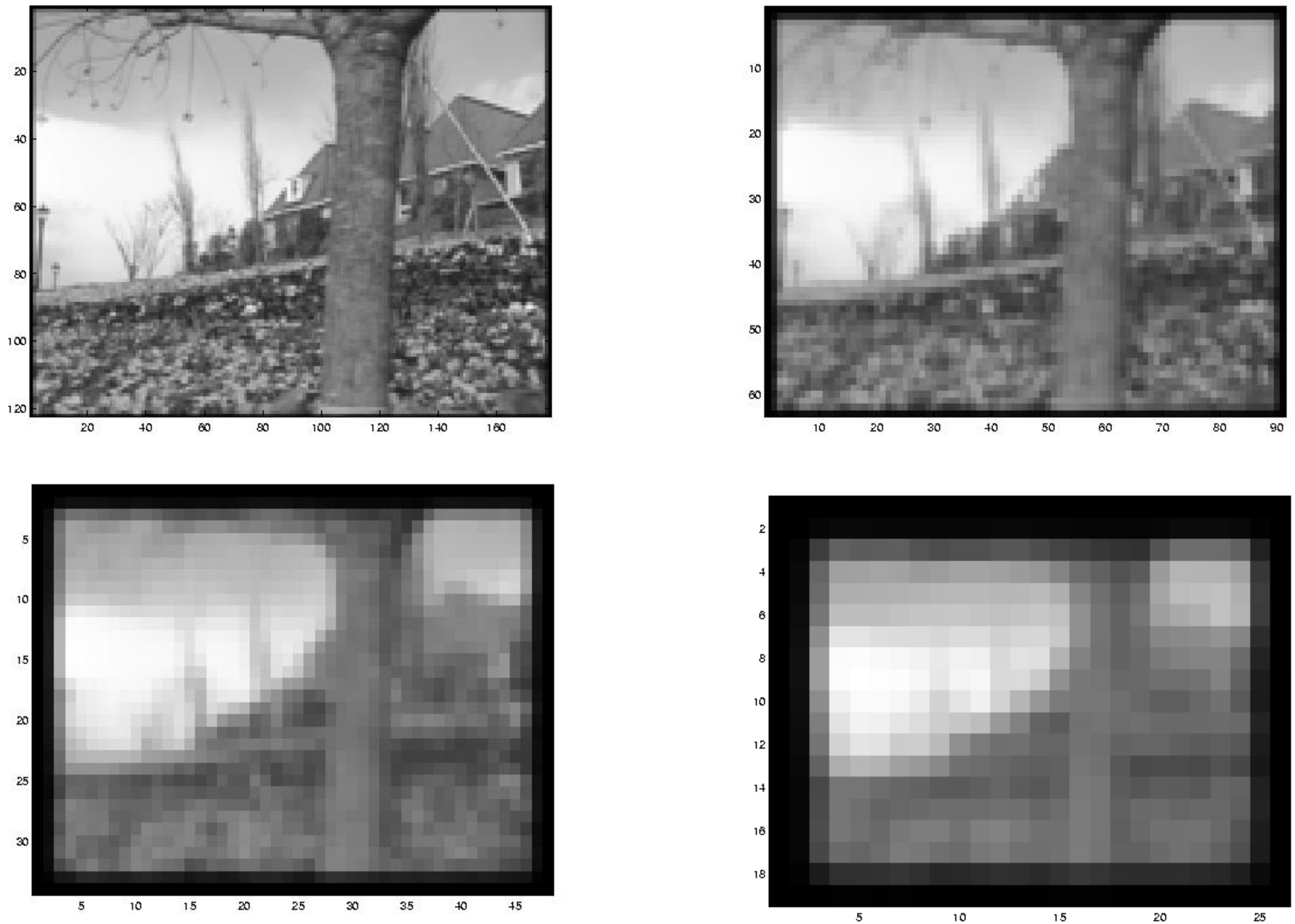
Nearest match is correct (no aliasing)



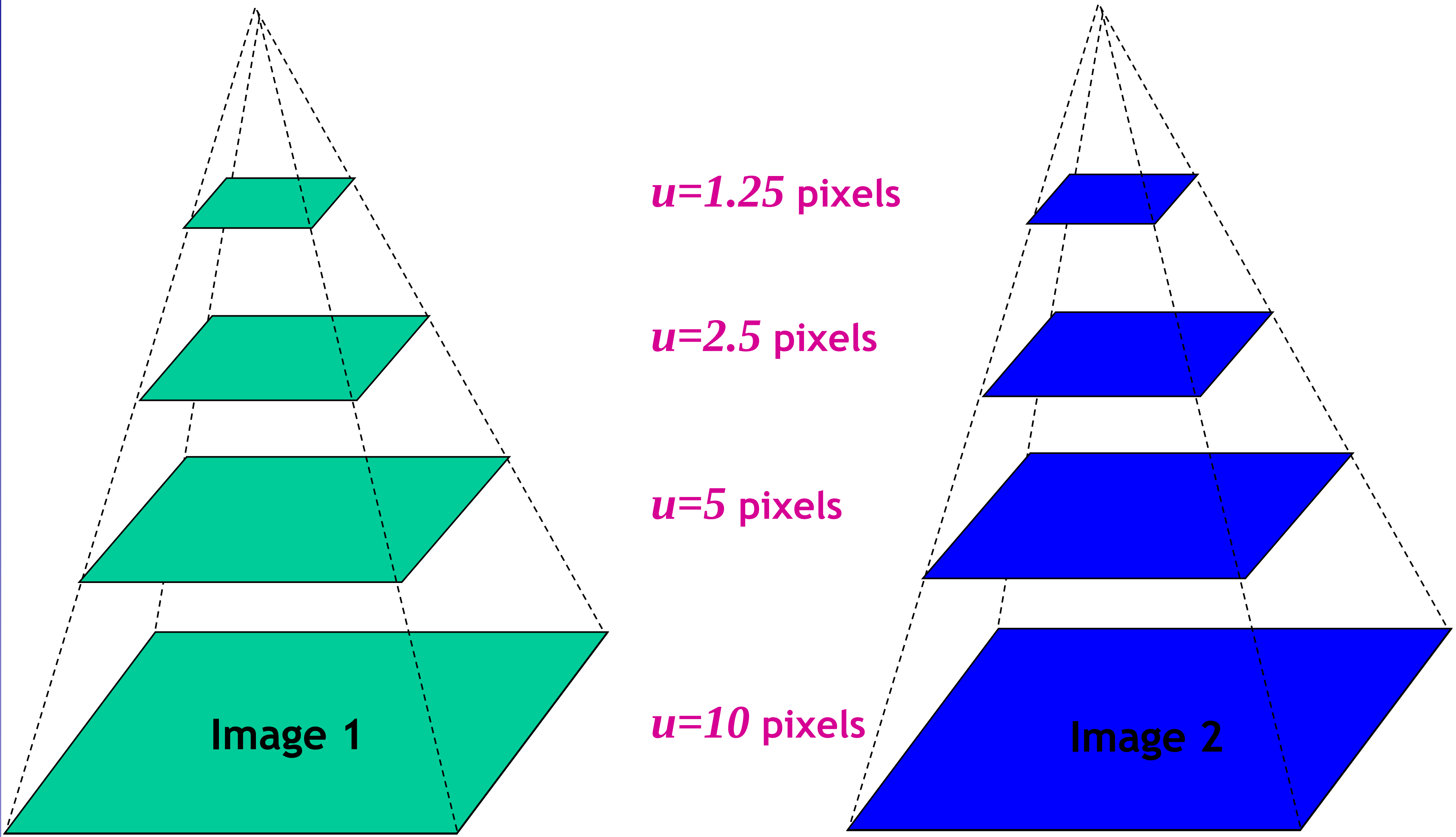
Nearest match is incorrect (aliasing)

- To overcome aliasing: **coarse-to-fine estimation**.

Idea: Reduce the Resolution!



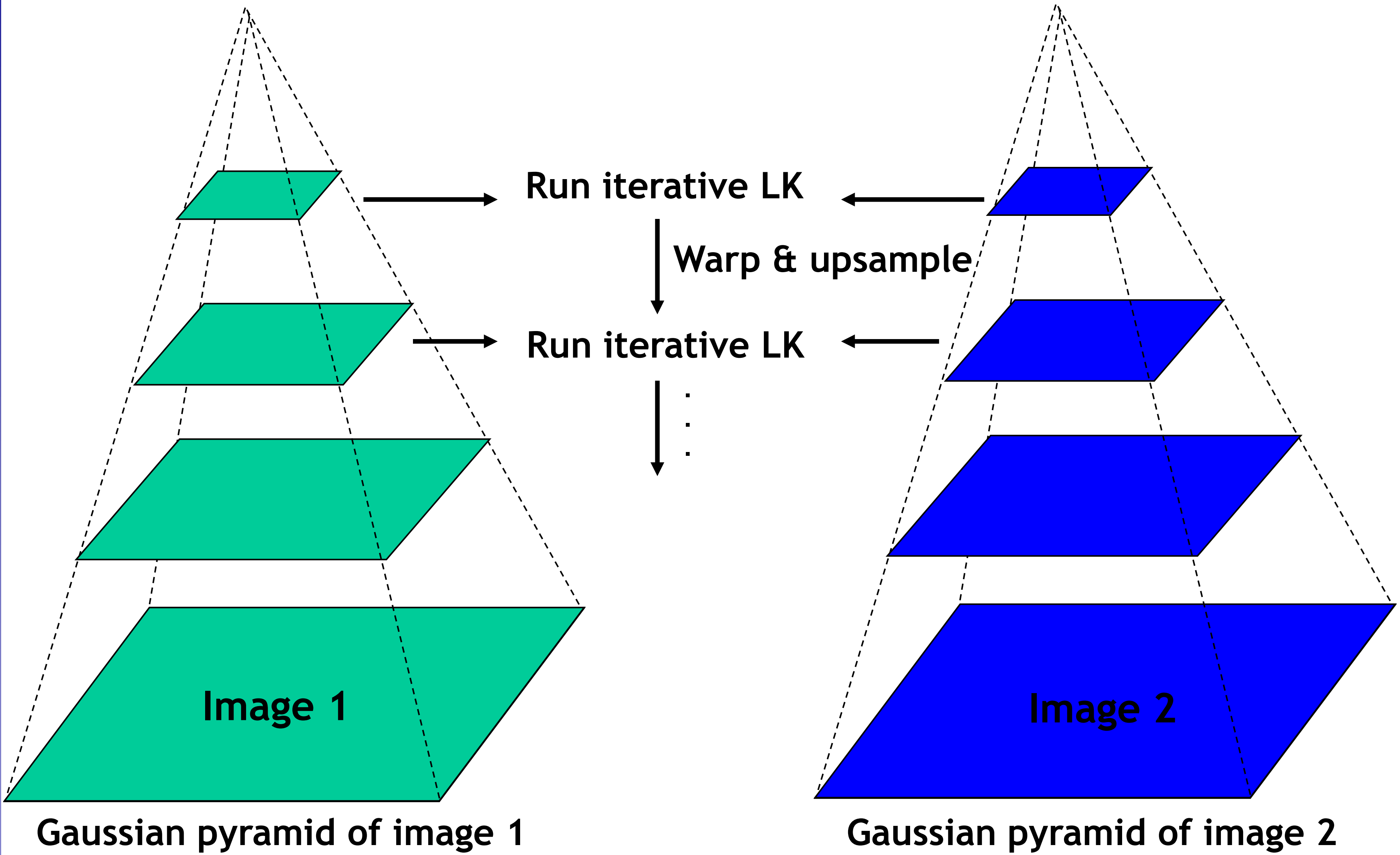
Recap: Coarse-to-fine Optical Flow Estimation



Gaussian pyramid of image 1

Gaussian pyramid of image 2

Recap: Coarse-to-fine Optical Flow Estimation

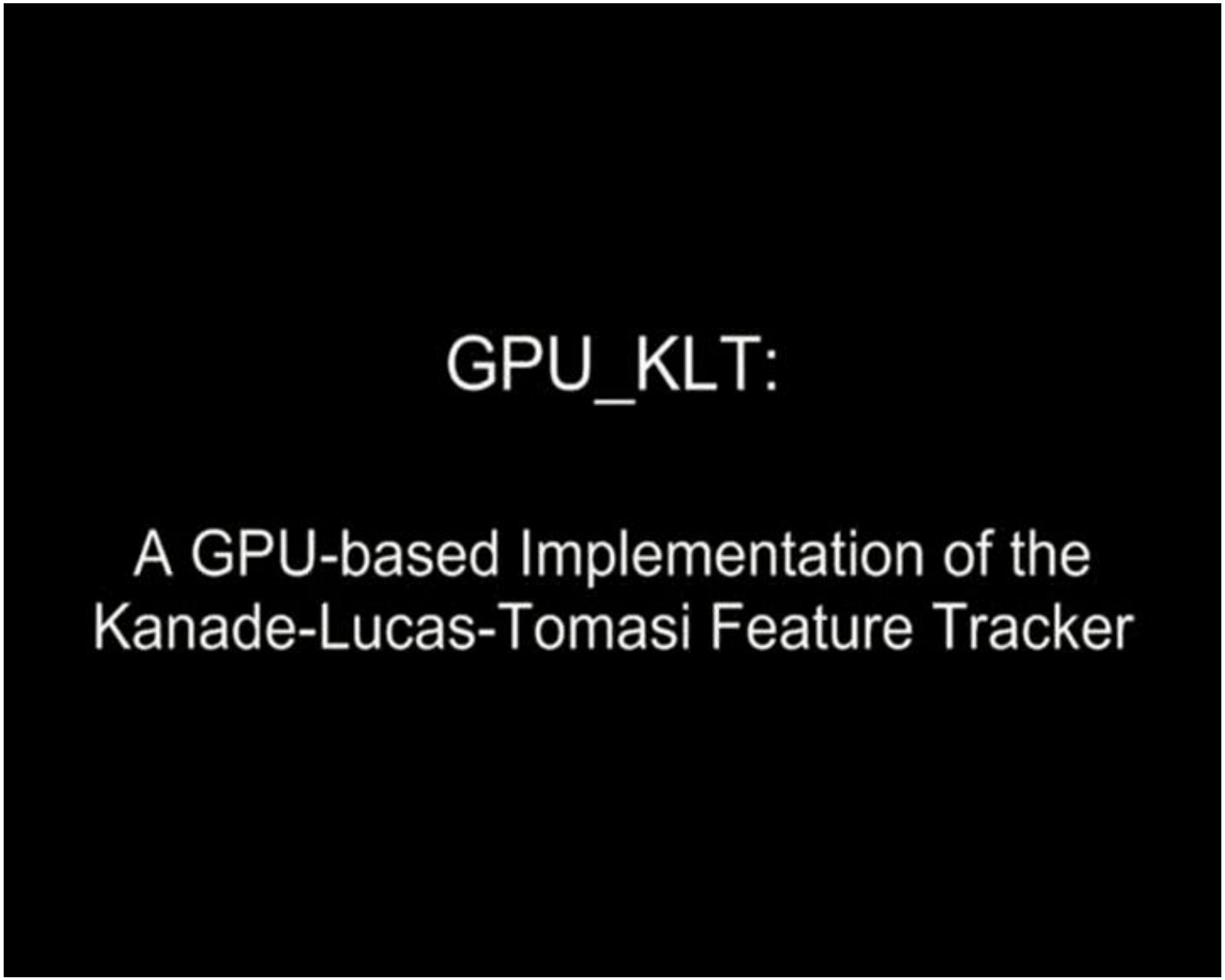


Slide credit: Steve Seitz

Topics of This Lecture

- Recap: Lucas-Kanade Optical Flow
  - Brightness Constancy constraint
  - LK flow estimation
  - Coarse-to-fine estimation
- **Feature Tracking**
  - **KLT feature tracking**
- Template Tracking
  - LK derivation for templates
  - Warping functions
  - General LK image registration
- Applications

KLT Feature Tracking



http://www.cs.unc.edu/~ssinha/Research/GPU_KLT/

Shi-Tomasi Feature Tracker

- Idea
  - Find good features using eigenvalues of second-moment matrix
  - Key idea: "good" features to track are the ones that can be tracked reliably.
- Frame-to-frame tracking
  - Track with LK and a pure *translation* motion model.
  - More robust for small displacements, can be estimated from smaller neighborhoods (e.g., 5×5 pixels).
- Checking consistency of tracks
  - *Affine* registration to the first observed feature instance.
  - Affine model is more accurate for larger displacements.
  - Comparing to the first frame helps to minimize drift.

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Tracking Example

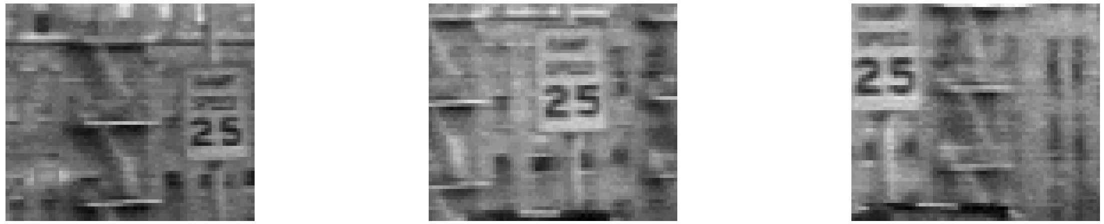


Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

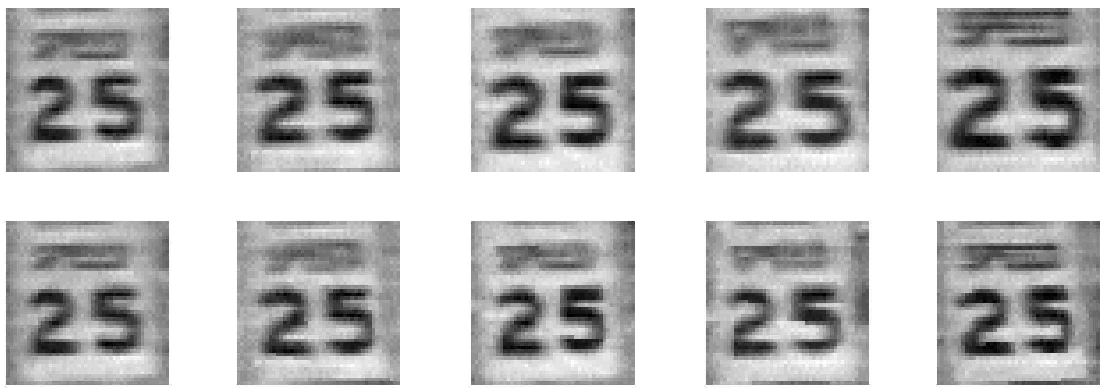


Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Real-Time GPU Implementations

- This basic feature tracking framework (Lucas-Kanade + Shi-Tomasi) is commonly referred to as “KLT tracking”.
  - Used as preprocessing step for many applications
  - Lends itself to easy parallelization

- Very fast GPU implementations available, e.g.,
  - C. Zach, D. Gallup, J.-M. Frahm,
    Fast Gain-Adaptive KLT tracking on the GPU.
    In CVGPU’08 Workshop, Anchorage, USA, 2008
  - 216 fps with automatic gain adaptation
  - 260 fps without gain adaptation

http://www.cs.unc.edu/~ssinha/Research/GPU_KLT/

http://www.inf.ethz.ch/personal/chzach/opensource.html

Topics of This Lecture

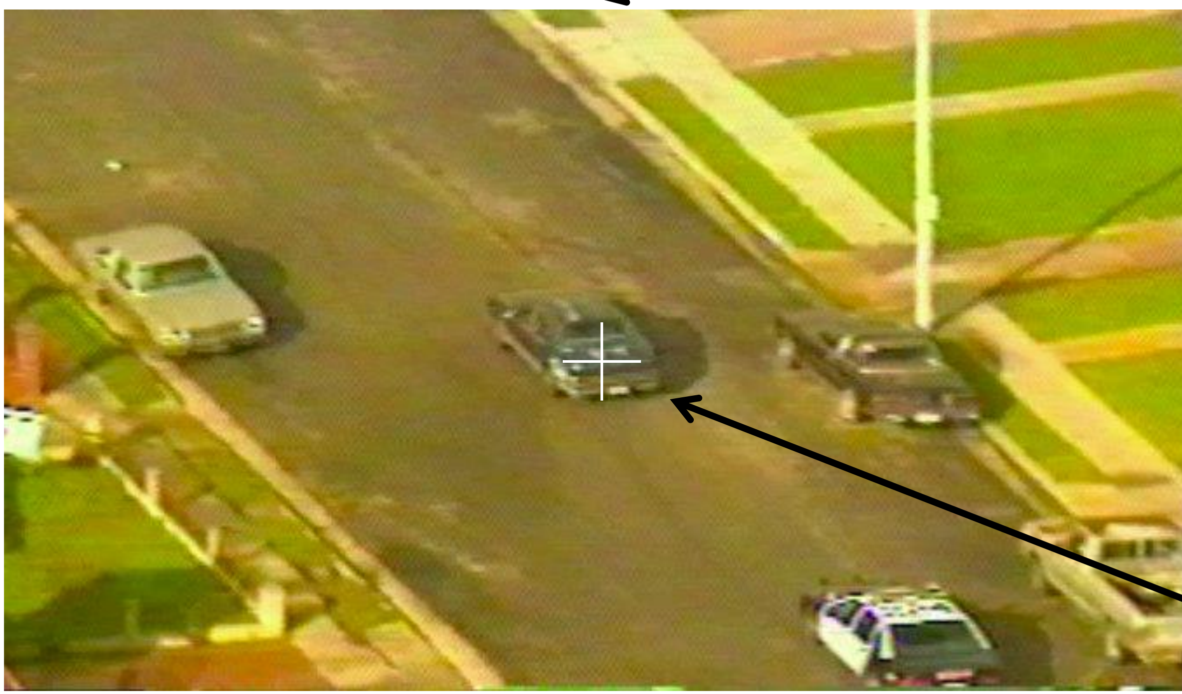
- Recap: Lucas-Kanade Optical Flow
  - Brightness Constancy constraint
  - LK flow estimation
  - Coarse-to-fine estimation
- Feature Tracking
  - KLT feature tracking
- Template Tracking
  - LK derivation for templates
  - Warping functions
  - General LK image registration
- Applications

Lucas-Kanade Template Tracking

- Traditional LK
  - Typically run on small, corner-like features (e.g., 5×5 patches) to compute optical flow ($\rightarrow$ KLT).
  - However, there is no reason why we can't use the same approach on a larger window around the tracked object.

Basic LK Derivation for Templates

$$E(u,v) = \sum_{\mathbf{x}} \left[I(x+u, y+v) - T(x,y) \right]^2$$



Current frame

Template model

(u,v) = hypothesized location of template in current frame

Basic LK Derivation for Templates

- **Taylor expansion**

$$E(u,v) = \sum_{\mathbf{x}} \left[I(x+u, y+v) - T(x,y)\right]^2$$

$$\approx \sum_{\mathbf{x}} \left[I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)\right]^2$$

$$= \sum_{\mathbf{x}} \left[uI_x(x,y) + vI_y(x,y) + D(x,y)\right]^2 \quad \text{with } D = I - T$$

- **Taking partial derivatives**

$$\frac{\partial E}{\partial u} = \sum_{\mathbf{x}} \left[uI_x(x,y) + vI_y(x,y) + D(x,y)\right] I_x(x,y) \stackrel{!}{=} 0$$

$$\frac{\partial E}{\partial v} = \sum_{\mathbf{x}} \left[uI_x(x,y) + vI_y(x,y) + D(x,y)\right] I_y(x,y) \stackrel{!}{=} 0$$

- **Equation in matrix form**

$$\sum_{\mathbf{x}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \sum_{\mathbf{x}} \begin{bmatrix} I_x D \\ I_y D \end{bmatrix} \quad \Rightarrow$$

Solve via least-squares

One Problem With This...

- Problematic Assumption
  - Assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable for long periods of time.

- However...
  - We can easily generalize the LK approach to other 2D parametric motion models (like affine or projective) by introducing a "warp" function $\mathbf{W}$ with parameters $\mathbf{p}$.

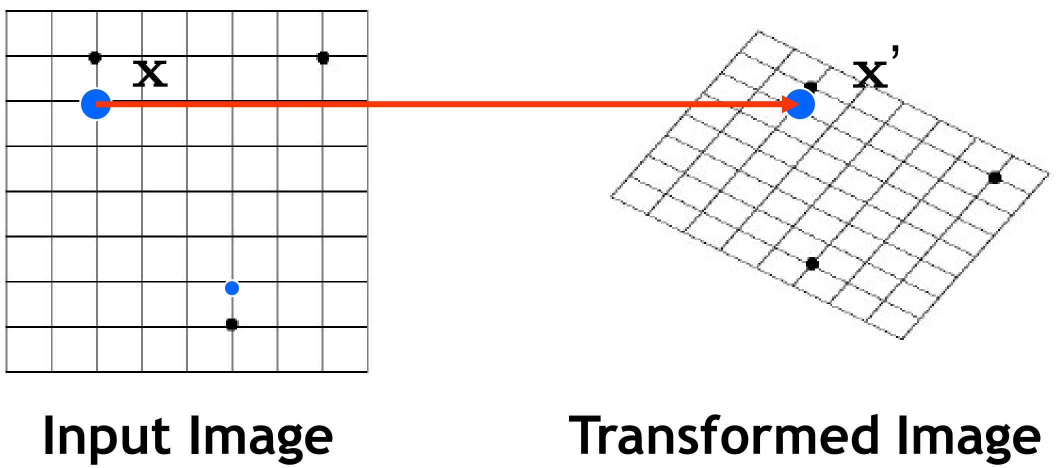
$$E(u,v) = \sum_{\mathbf{x}} \left[I(x+u, y+v) - T(x,y) \right]^2$$

$$\downarrow$$

$$E(\mathbf{p}) = \sum_{\mathbf{x}} \left[I(\mathbf{W}([x,y];\mathbf{p})) - T([x,y]) \right]^2$$

Geometric Image Warping

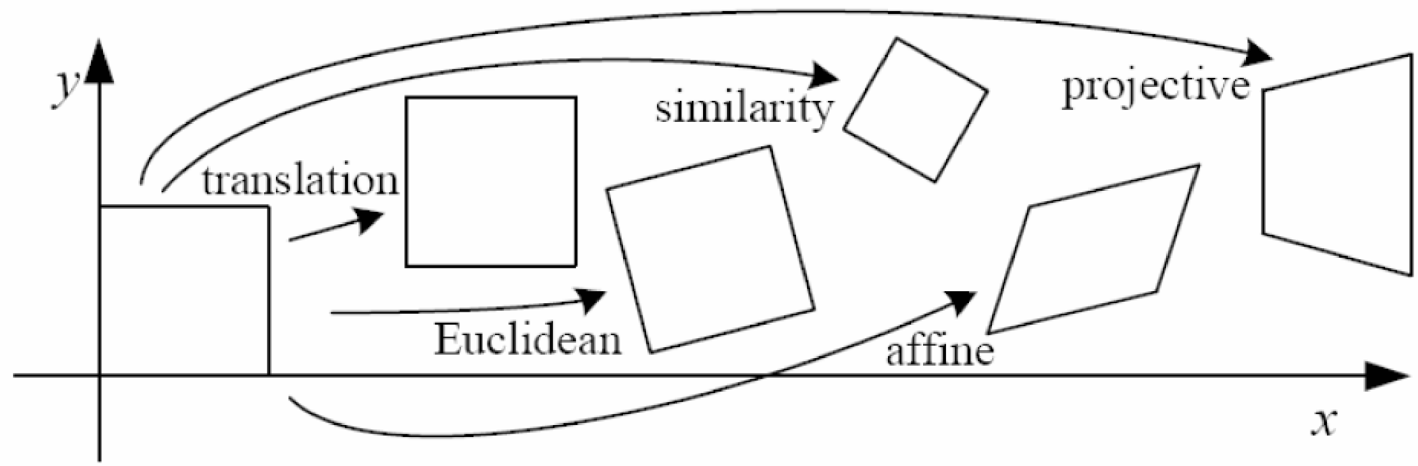
- The warp $\mathbf{W}(\mathbf{x};\mathbf{p})$ describes the geometric relationship between two images



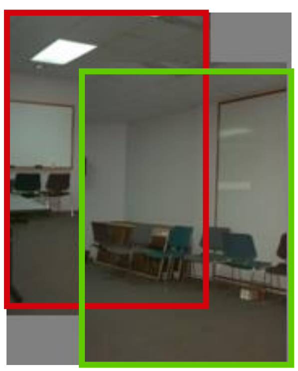
$$\mathbf{x}' = \mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix}$$

Parameters of the warp

Example Warping Functions

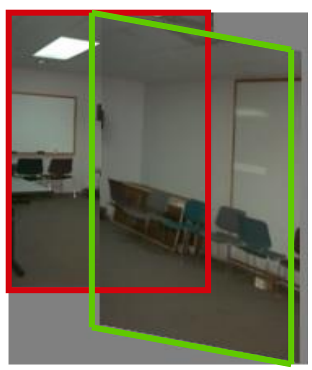


Translation



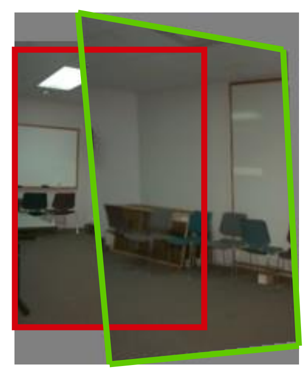
2 unknowns

Affine



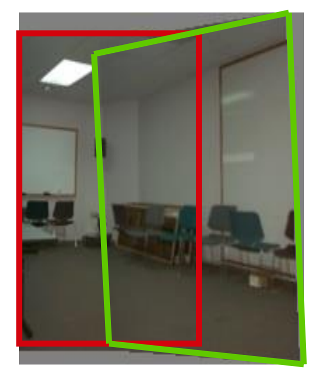
6 unknowns

Perspective



8 unknowns

3D rotation



3 unknowns

Slide credit: Steve Seitz

Example Warping Functions

- **Translation**

$$\mathbf{W}([x,y];\mathbf{p}) = \begin{bmatrix} x+p_1 \\ y+p_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Affine**

$$\mathbf{W}([x,y];\mathbf{p}) = \begin{bmatrix} x+p_1x+p_3y+p_5 \\ y+p_2x+p_4y+p_6 \end{bmatrix} = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Perspective**

$$\mathbf{W}([x,y];\mathbf{p}) = \frac{1}{p_7x+p_8y+1} \begin{bmatrix} x+p_1x+p_3y+p_5 \\ y+p_2x+p_4y+p_6 \end{bmatrix}$$

  - **Note: Other parametrizations are possible; the above ones are just particularly convenient here.**

General LK Image Registration

- **Goal**
  - Find the warping parameters $\mathbf{p}$ that minimize the sum-of-squares intensity difference between the template image and the warped input image.

- **LK formulation**
  - Formulate this as an optimization problem

$$\arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x}) \right]^2$$

  - We assume that an initial estimate of $\mathbf{p}$ is known and iteratively solve for increments to the parameters $\Delta\mathbf{p}$:

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x}) \right]^2$$

Step-by-Step Derivation

- **Key to the derivation**
  - Taylor expansion around $\Delta\mathbf{p}$

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + \mathcal{O}(\Delta\mathbf{p}^2)$$

  - Using pixel coordinates $\mathbf{x} = [x,y]$

$$\begin{aligned} I(\mathbf{W}([x,y];\mathbf{p}+\Delta\mathbf{p})) \approx\; & I(\mathbf{W}([x,y];p_1,\ldots,p_n)) \\ & + \left[\frac{\partial I}{\partial x}\frac{\partial W_x}{\partial p_1} + \frac{\partial I}{\partial y}\frac{\partial W_y}{\partial p_1}\right]_{p_1} \Delta p_1 \\ & + \left[\frac{\partial I}{\partial x}\frac{\partial W_x}{\partial p_2} + \frac{\partial I}{\partial y}\frac{\partial W_y}{\partial p_2}\right]_{p_1} \Delta p_2 \\ & + \;\ldots \\ & + \left[\frac{\partial I}{\partial x}\frac{\partial W_x}{\partial p_n} + \frac{\partial I}{\partial y}\frac{\partial W_y}{\partial p_n}\right]_{p_n} \Delta p_n \end{aligned}$$

Step-by-Step Derivation

- **Rewriting this in matrix notation**

$$
\begin{aligned}
I(\mathbf{W}([x,y];\mathbf{p}+\Delta\mathbf{p})) \approx\ & I(\mathbf{W}([x,y];p_1,\ldots,p_n)) \\
&+ \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial W_x}{\partial p_1} \\ \frac{\partial W_y}{\partial p_1} \end{bmatrix}_{p_1} \Delta p_1 \\
&+ \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial W_x}{\partial p_2} \\ \frac{\partial W_y}{\partial p_2} \end{bmatrix}_{p_2} \Delta p_2 \\
&+ \ \ldots \\
&+ \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_n} \end{bmatrix}_{p_n} \Delta p_n
\end{aligned}
$$

Step-by-Step Derivation

- **And further collecting the derivative terms**

$$I(\mathbf{W}([x,y];\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}([x,y];p_1,\ldots,p_n))$$

$$+\begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix}\begin{bmatrix}\frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n}\end{bmatrix}\begin{bmatrix}\Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_n\end{bmatrix}$$

Gradient	Jacobian	Increment parameters to solve for
∇I	$\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$	$\Delta \mathbf{p}$

- **Written in matrix form**

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p}$$

Example: Jacobian of Affine Warp

- **General equation of Jacobian**

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{bmatrix}$$

- **Affine warp function (6 parameters)**

$$\mathbf{W}([x, y]; \mathbf{p}) = \begin{bmatrix} 1 + p_1 & p_3 & p_5 \\ p_2 & 1 + p_4 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Result**

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \frac{\partial \begin{bmatrix} x + p_1 x + p_3 y + p_5 \\ p_2 x + y + p_4 y + p_6 \end{bmatrix}}{\partial \mathbf{p}}$$

$$= \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

Minimizing the Registration Error

- **Optimization function after Taylor expansion**

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

- **Minimizing this function**
  - *How?*

Minimizing the Registration Error

- **Optimization function after Taylor expansion**

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

- **Minimizing this function**
  - Taking the partial derivative and setting it to zero

$$\frac{\partial}{\partial \Delta\mathbf{p}} \stackrel{!}{=} 0 \rightarrow 2 \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right] \stackrel{!}{=} 0$$

  - Closed-form solution for $\Delta\mathbf{p}$ (Gauss-Newton):

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

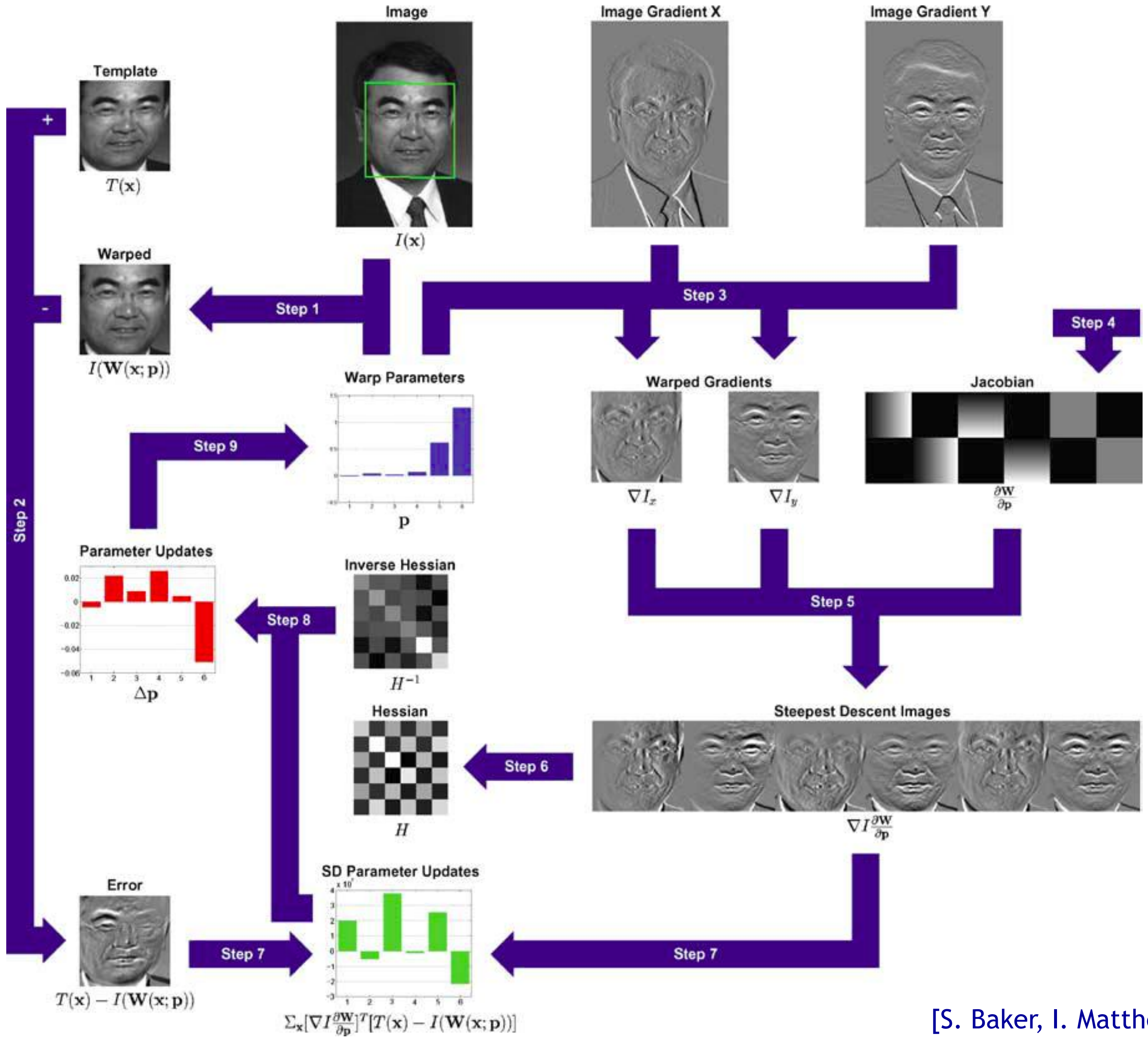
  - where $\mathbf{H}$ is the Hessian $\mathbf{H} = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$

Summary: LK Algorithm

- Iterate
  - Warp I to obtain $I(\mathbf{W}([x, y]; \mathbf{p}))$
  - Compute the error image $T([x, y]) - I(\mathbf{W}([x, y]; \mathbf{p}))$
  - Warp the gradient ∇I with $\mathbf{W}([x, y]; \mathbf{p})$
  - Evaluate $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $([x, y]; \mathbf{p})$ (**Jacobian**)
  - Compute steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
  - Compute Hessian matrix $\mathbf{H} = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$
  - Compute $\sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [T([x, y]) - I(\mathbf{W}([x, y]; \mathbf{p}))]$
  - Compute $\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [T([x, y]) - I(\mathbf{W}([x, y]; \mathbf{p}))]$
  - Update the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$
- Until $\Delta \mathbf{p}$ magnitude is negligible

[S. Baker, I. Matthews, IJCV'04]

LK Algorithm Visualization



[S. Baker, I. Matthews, IJCV'04]

Discussion LK Alignment

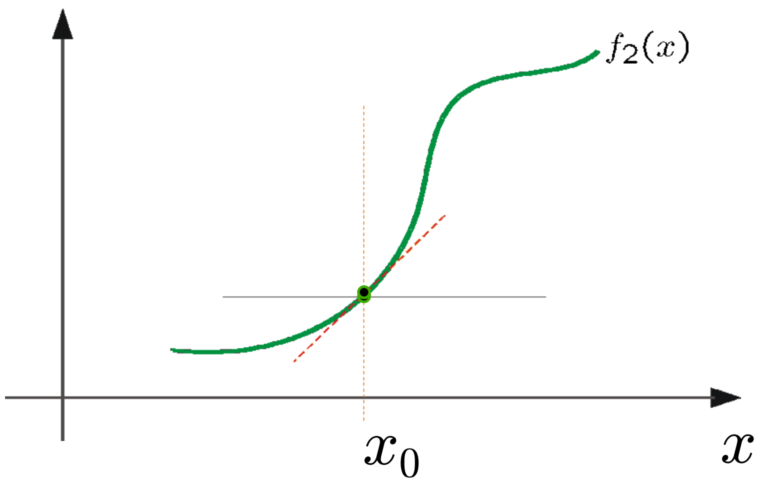
- Pros
  - All pixels get used in matching
  - Can get sub-pixel accuracy (important for good mosaicking)
  - Fast and simple algorithm
  - Applicable to Optical Flow estimation, stereo disparity estimation, parametric motion tracking, etc.

- Cons
  - Prone to local minima.
  - Relatively small movement.

  $\Rightarrow$ Good initialization necessary

Side Note

- LK Registration needs a good initialization
  - Taylor expansion corresponds to a linearization around the initial position $\mathbf{p}$.
  - This linearization is only valid in a small neighborhood around $\mathbf{p}$.



- When tracking templates…
  - We typically use the previous frame's result as initialization.

$\Rightarrow$ The higher the frame rate, the smaller the warp will be.

$\Rightarrow$ This means we get better results and need fewer LK iterations.

$\Rightarrow$ *Tracking becomes easier (and faster!) with higher frame rates.*

Discussion

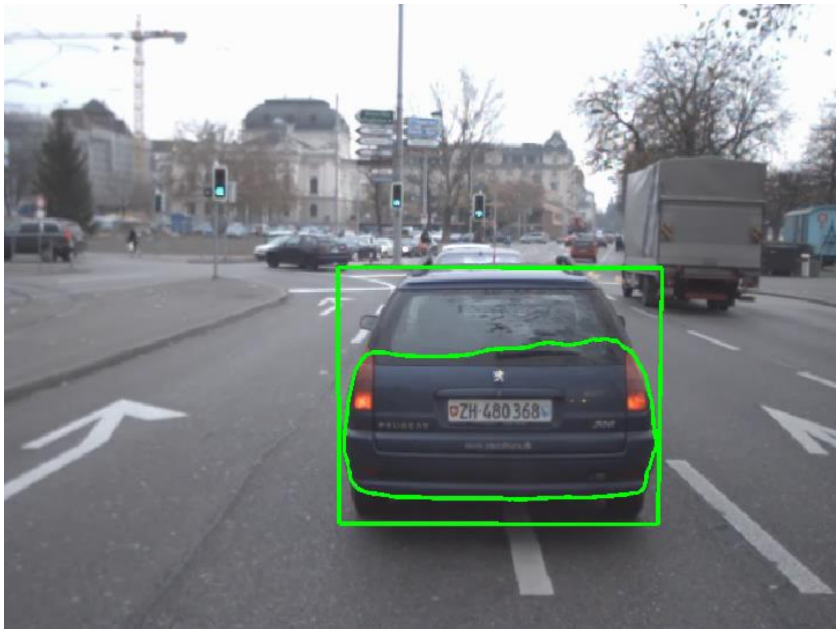
- **Beyond 2D Tracking/Registration**
  - So far, we focused on registration between 2D images.
  - The same ideas can be used when performing registration between a 3D model and the 2D image (model-based tracking).
  - The approach can also be extended for dealing with articulated objects and for tracking in subspaces.

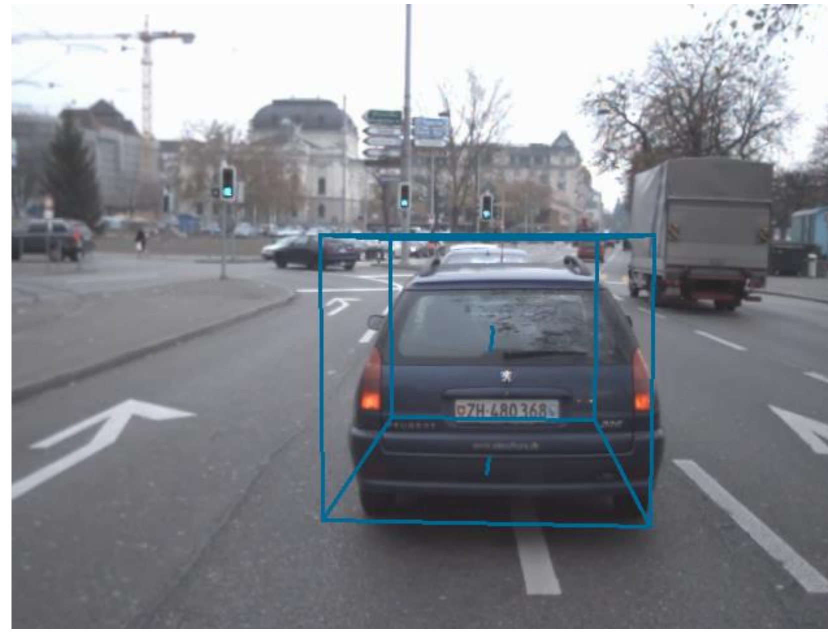
  ⇒ We will come back to this in later lectures when we talk about model-based 3D tracking...

Topics of This Lecture

- Recap: Lucas-Kanade Optical Flow
  - Brightness Constancy constraint
  - LK flow estimation
  - Coarse-to-fine estimation
- Feature Tracking
  - KLT feature tracking
- Template Tracking
  - LK derivation for templates
  - Warping functions
  - General LK image registration
- **Applications**

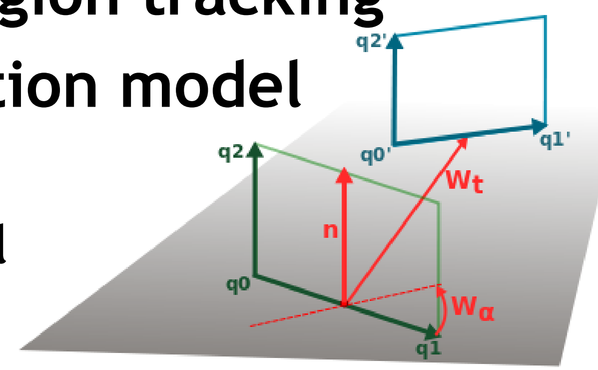
Example of a More Complex Warping Function





- **Encode geometric constraints into region tracking**
- **Constrained homography transformation model**
  - Translation parallel to the ground plane
  - Rotation around the ground plane normal
  - $\mathbf{W}(\mathbf{x}) = \mathbf{W}_{obj}\ \mathbf{P}\ \mathbf{W}_t\ \mathbf{W}_\alpha\ \mathbf{Q}\ \mathbf{x}$

$\Rightarrow$ **Input for high-level tracker with car steering model.**



[E. Horbert, D. Mitzel, B. Leibe, DAGM'10]

References and Further Reading

- The original paper by Lucas & Kanade
  - B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. IJCAI*, pp. 674-679, 1981.
- A more recent paper giving a better explanation
  - S. Baker, I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. In IJCV, Vol. 56(3), pp. 221-255, 2004.
- The original KLT paper by Shi & Tomasi
  - J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.