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
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:
    \[
    N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\}
    \]
  - 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
    $$ p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) $$
  - 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.
Recap: Stauffer-Grimson Background Model

- **Idea**
  - Model the distribution of each pixel by a mixture of $K$ Gaussians
    \[
    p(x) = \sum_{k=1}^{K} \pi_k N(x_n | \mu_k, \Sigma_k)
    \]
    where \( \Sigma_k = \sigma_k^2 I \)
  - Check every new pixel value against the existing $K$ components until a match is found (pixel value within $2.5 \sigma_k$ of $\mu_k$).
  - 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 $w_k/\sigma_k$ 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
    
    \[ \mu_k^{(t+1)} = (1 - \rho)\mu_k^{(t)} + \rho x^{(t+1)} \]
    
    \[ \Sigma_k^{(t+1)} = (1 - \rho)\Sigma_k^{(t)} + \rho(x^{(t+1)} - \mu_k^{(t+1)})(x^{(t+1)} - \mu_k^{(t+1)})^{T} \]

  where

  \[ \rho = \alpha \mathcal{N}(x_n | \mu_k, \Sigma_k) \]

  (i.e., the update is weighted by the component likelihood)
Recap: Kernel Background Modeling

- Nonparametric density estimation
  - Estimate a pixel’s background distribution using the kernel density estimator $K(\cdot)$ as
    \[
    p(x^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} K(x^{(t)} - x^{(i)})
    \]
  - Choose $K$ to be a Gaussian $\mathcal{N}(0, \Sigma)$ with $\Sigma = \text{diag}\{\sigma_j\}$. Then
    \[
    p(x^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2} \frac{(x_{i}^{(t)} - x_{i}^{(i)})^2}{\sigma_j^2}}
    \]
  - A pixel is considered foreground if $p(x^{(t)}) < \theta$ for a threshold $\theta$.
    - 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
Applications: Visual Surveillance

- Background modeling to detect objects for tracking
  - Extension: Learning a foreground model for each object.
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, \( I_x \cdot u + I_y \cdot v + I_t \approx 0 \)

Slide credit: Svetlana Lazebnik
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?
  - It gives us a constraint on the component of the flow in the direction of the gradient.

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

⇒ 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 \(\nabla I \cdot (u', v') = 0\)

Slide credit: Svetlana Lazebnik
The Aperture Problem
The Aperture Problem

Actual motion
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

Slide credit: Svetlana Lazebnik
The Barber Pole Illusion

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

Slide credit: Svetlana Lazebnik
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 \(5 \times 5\) window, that gives us 25 equations per pixel

\[
0 = I_t(p_i) + \nabla I(p_i) \cdot [u \ v]
\]

\[
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
\]

Recap: Solving the Aperture Problem

- Least squares problem:

\[
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
\]

- Minimum least squares solution given by solution of

\[
(A^T A) d = A^T b
\]

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

\[
A^T A
\]

\[
A^T b
\]

(The summations are over all pixels in the $K \times K$ window)
Recap: Conditions for Solvability

- **Optimal** \((u, v)\) satisfies Lucas-Kanade equation

\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} =
\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix} = A^T A
\]

- **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.
  - 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.
\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} = -
\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
\]

\[A^T A\]
\[A^T b\]

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 $u$)

Slide credit: Steve Seitz
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: $d_3 = d_2 + \tilde{d}$

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

Slide credit: Steve Seitz
Recap: Iterative LK Refinement

\[ f_1(x - d_3) \approx f_2(x) \]

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

Slide credit: Steve Seitz
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?

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

Slide credit: Steve Seitz
Idea: Reduce the Resolution!
Recap: Coarse-to-fine Optical Flow Estimation

Image 1

Gaussian pyramid of image 1

u=10 pixels

u=5 pixels

u=2.5 pixels

u=1.25 pixels

Gaussian pyramid of image 2

Image 2

Slide credit: Steve Seitz
Recap: Coarse-to-fine Optical Flow Estimation

Image 1
Gaussian pyramid of image 1

Image 2
Gaussian pyramid of image 2

Run iterative LK
Warp & upsample
Run iterative LK

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

GPU_KLT:

A GPU-based Implementation of the Kanade-Lucas-Tomasi Feature Tracker

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., $5 \times 5$ 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.


Slide credit: Svetlana Lazebnik
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).


Slide credit: Svetlana Lazebnik
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.,
    - 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/chezach/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., $5 \times 5$ patches) to compute optical flow (→ KLT).
  - However, there is no reason why we can’t use the same approach on a larger window around the tracked object.

Slide credit: Robert Collins
Basic LK Derivation for Templates

\[ E(u, v) = \sum_x \left[ I(x + u, y + v) - T(x, y) \right]^2 \]

Explanation:
- \( E(u, v) \) is the energy function
- \( I(x + u, y + v) \) is the intensity at the hypothesized location
- \( T(x, y) \) is the template intensity
- The goal is to minimize this energy to find the best match

Slide credit: Robert Collins
Basic LK Derivation for Templates

- **Taylor expansion**

\[
E(u, v) = \sum_x [I(x + u, y + v) - T(x, y)]^2
\]

\[
\approx \sum_x [I(x, y) + uI_x(x, y) + vI_y(x, y) - T(x, y)]^2
\]

\[
= \sum_x [uI_x(x, y) + vI_y(x, y) + D(x, y)]^2 \quad \text{with } D = I - T
\]

- **Taking partial derivatives**

\[
\frac{\partial E}{\partial u} = \sum_x [uI_x(x, y) + vI_y(x, y) + D(x, y)] I_x(x, y) \overset{!}{=} 0
\]

\[
\frac{\partial E}{\partial v} = \sum_x [uI_x(x, y) + vI_y(x, y) + D(x, y)] I_y(x, y) \overset{!}{=} 0
\]

- **Equation in matrix form**

\[
\sum_x \begin{bmatrix}
I_x^2 & I_x I_y \\
I_x I_y & I_y^2
\end{bmatrix} \begin{bmatrix}
u
\end{bmatrix} = \sum_x \begin{bmatrix}
I_x D \\
I_y D
\end{bmatrix}
\]

\[\Rightarrow\quad \text{Solve via least-squares}\]
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 $W$ with parameters $p$.

$$E(u, v) = \sum_x [I(x + u, y + v) - T(x, y)]^2$$

$$\downarrow$$

$$E(p) = \sum_x [I(W([x, y]; p)) - T([x, y])]^2$$
Geometric Image Warping

- The warp $W(x; p)$ describes the geometric relationship between two images.

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

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

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

- Perspective
  \[ W([x, y]; p) = \frac{1}{p_7 x + p_8 y + 1} \begin{bmatrix} x + p_1 x + p_3 y + p_5 \\ y + p_2 x + p_4 y + p_6 \end{bmatrix} \]

- Note: Other parametrizations are possible; the above ones are just particularly convenient here.
General LK Image Registration

• Goal
  - Find the warping parameters $p$ 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
    $$\arg\min_p \sum_x \left[ I(W(x; p)) - T(x) \right]^2$$
  - We assume that an initial estimate of $p$ is known and iteratively solve for increments to the parameters $\Delta p$:
    $$\arg\min_{\Delta p} \sum_x \left[ I(W(x; p + \Delta p)) - T(x) \right]^2$$
Step-by-Step Derivation

- **Key to the derivation**
  - Taylor expansion around $\Delta p$
    
    $$I(W(x; p + \Delta p)) \approx I(W(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p + \mathcal{O}(\Delta p^2)$$
  
  - Using pixel coordinates $x = [x, y]$
    
    $$I(W([x, y]; p + \Delta p)) \approx I(W([x, y]; p_1, \ldots, p_n))$$
    
    $$+ \left[ \frac{\partial I \partial W_x}{\partial x \partial p_1} + \frac{\partial I \partial W_y}{\partial y \partial p_1} \right]_{p_1} \Delta p_1$$

    $$+ \left[ \frac{\partial I \partial W_x}{\partial x \partial p_2} + \frac{\partial I \partial W_y}{\partial y \partial p_2} \right]_{p_1} \Delta p_2$$

    $$+ \cdots$$

    $$+ \left[ \frac{\partial I \partial W_x}{\partial x \partial p_n} + \frac{\partial I \partial W_y}{\partial y \partial p_n} \right]_{p_n} \Delta p_n$$

Slide credit: Robert Collins
Step-by-Step Derivation

- Rewriting this in matrix notation

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

\[
+ \begin{bmatrix}
\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial W_x}{\partial p_1} \\
\frac{\partial W_y}{\partial p_1}
\end{bmatrix}
\Delta p_1
\]

\[
+ \begin{bmatrix}
\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial W_x}{\partial p_2} \\
\frac{\partial W_y}{\partial p_2}
\end{bmatrix}
\Delta p_2
\]

\[
+ \ldots
\]

\[
+ \begin{bmatrix}
\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial W_x}{\partial p_n} \\
\frac{\partial W_y}{\partial p_n}
\end{bmatrix}
\Delta p_n
\]
Step-by-Step Derivation

- And further collecting the derivative terms

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

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

- Written in matrix form

\[ I(W(x; p + \Delta p)) \approx I(W(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p \]
Example: Jacobian of Affine Warp

- General equation of Jacobian

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

- Affine warp function (6 parameters)

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

- Result

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

\[
= \begin{bmatrix}
x & 0 & y & 0 & 1 & 0 \\
0 & x & 0 & y & 0 & 1
\end{bmatrix}
\]
Minimizing the Registration Error

- Optimization function after Taylor expansion

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

- Minimizing this function
  
  How?
Minimizing the Registration Error

- **Optimization function after Taylor expansion**
  \[
  \arg\min_{\Delta p} \sum_x \left[ I(W(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right]^2
  \]

- **Minimizing this function**
  \[
  \frac{\partial}{\partial \Delta p} = 0 \rightarrow 2 \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T \left[ I(W(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right] = 0
  \]
  
  - **Closed-form solution for \( \Delta p \) (Gauss-Newton):**
    \[
    \Delta p = H^{-1} \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T [T(x) - I(W(x; p))]
    \]

  - **where** \( H \) **is the Hessian**
    \[
    H = \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T \left[ \nabla I \frac{\partial W}{\partial p} \right]
    \]

B. Leibe
Summary: LK Algorithm

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

- Until $\Delta p$ magnitude is negligible

B. Leibe

[S. Baker, I. Matthews, IJCV’04]
LK Algorithm Visualization

- Template: T(x)
- Warped: I(W(x; p))
- Image: I(x)
- Image Gradient X: \nabla I_x
- Image Gradient Y: \nabla I_y
- Warp Parameters: p
- Warped Gradients: \nabla I^W_x, \nabla I^W_y
- Jacobian: \nabla^W W
- Parameter Updates: \Delta p
- Inverse Hessian: H^{-1}
- Hessian: H
- Error: T(x) - I(W(x; p))
- SD Parameter Updates: \Sigma [\nabla I^W_x]^T [T(x) - I(W(x; p))]
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.
  ⇒ Good initialization necessary
Side Note

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

- 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
  - $W(x) = W_{obj} P W_t W_\alpha Q x$
  - 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

• A more recent paper giving a better explanation

• The original KLT paper by Shi & Tomasi