Lecture 12
Tracking

COS 429: Computer Vision

Slides credit:
Many slides adapted from James Hays, Derek Hoeim, Lana Lazebnik, Silvio Saverese, who in turn adapted slides from Steve Seitz, Rick Szeliski, Martial Hebert, Mark Pollefeys, and others
Motivation: Mobileye
Camera-based Driver Assistance System

Safety Application based on single forward looking camera:
• Lane Detection
  - Lane Departure Warning (LDW)
  - Lane Keeping and Support
• Vehicle Detection
  - Forward Collision Warning (FCW)
  - Headway Monitoring and Warning
  - Adaptive Cruise Control (ACC)
  - Traffic Jam Assistant
  - Emergency Braking (AEB)
• Pedestrian Detection
  - Pedestrian Collision Warning (PCW)
  - Pedestrian Emergency Braking

For Videos, visit www.mobileye.com
Detect... Detect ... Detect...
Or 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.
Approaches to Object Tracking

- Motion model (translation, translation+scale, affine, non-rigid, ...)
- Image representation (gray/color pixel, edge image, histogram, 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
Distance Metric

• Goal: find \(d\) 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
Matching with filters

Goal: find \[\text{in image} \]

• Method 0: filter the image with eye patch

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

What went wrong?

response is stronger
for higher intensity
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 \( f \)
Sum of Squared error (L2)

- **Goal**: find \(\text{in image}\)
- **Method 2: SSD**
  \[
  h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2
  \]
Sum of Squared error (L2)

- Goal: find in image
- Method 2: SSD

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

One potential downside of SSD:
Brightness Constancy Assumption
Normalized Cross-Correlation

- **Goal**: find \( \text{} \) in image
- **Method 3**: Normalized cross-correlation (= angle between zero-mean vectors)

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

**Matlab**: `normxcorr2(template, im)`
Normalized Cross-Correlation

- Goal: find 🕵️‍♂️ in image
- Method 3: Normalized cross-correlation
Normalized Cross-Correlation

- Goal: find \( \begin{array}{c} \text{\textbullet} \end{array} \) in image
- Method 3: Normalized cross-correlation

Input

Normalized X-Correlation

Thresholded Image

True detections
Search vs. Gradient Descent

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

• If we have a good guess, can we do something cheaper?
  – Gradient Descent
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)
The brightness constancy constraint

\[ I(x, y, t) = I(x+u, y+v, t+1) \]

Take Taylor expansion of \( I(x+u, y+v, t+1) \) at \( (x, y, t) \) to linearize the right side:

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

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

Hence,

\[ I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \Rightarrow \nabla I \cdot [u \ v]^T + I_t = 0 \]
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?
Intuition in 1-D

Solve for $u$ in: \[ I_x \cdot u + I_t \approx 0 \]
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

\[ \nabla I \cdot [u' \ v']^T = 0 \]
The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion
The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion
The aperture problem

Perceived motion
The aperture problem

Actual motion
Solving the ambiguity...


- 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:

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

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

- Least squares problem:

\[
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
= 
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix} = b
\]
Matching patches across images

- Over-constrained linear system

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

\[
A \ d = b
\]

Least squares solution for \( d \) given by

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

\[
A^T A 
\]

\[
A^T b
\]

The summations are over all pixels in the K x K window
Dealing with larger movements: Iterative refinement

1. Initialize \((x', y') = (x, y)\)

2. Compute \((u, v)\) by

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

- Use interpolation to warp by subpixel values

1. Shift window by \((u, v)\): 
\(x' = x' + u; \quad y' = y' + v;\)

2. Recalculate \(I_t\)

3. Repeat steps 2-4 until small change

- Use interpolation to warp by subpixel values
Schematic of Lucas-Kanade

[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?
Dealing with larger movements: coarse-to-fine registration

- Gaussian pyramid of image 1 (t)
- Gaussian pyramid of image 2 (t+1)

run iterative L-K

upsample

run iterative L-K
Coarse-to-fine optical flow estimation

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 1

image 2
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
  – ...

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 affected by brightness
  - Any filter that is zero-mean
  - Example: vertical, horizontal edge filters
  - Example: Non-parametric filters (Rank & Census Transforms)
More Problems

• Outliers: bright strong features that are wrong

• Complex, high dimensional, or non-rigid motion
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)
Conditions for solvability

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

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

\(A^T A\)

\(A^T b\)

When is this solvable? I.e., what are good points to track?

- \(A^T A\) should be invertible
- \(A^T A\) should not be too small due to noise
  - eigenvalues \(\lambda_1\) and \(\lambda_2\) of \(A^T A\) should not be too small
- \(A^T A\) should be well-conditioned
  - \(\lambda_1 / \lambda_2\) should not be too large (\(\lambda_1 = \text{larger eigenvalue}\))

Recall: This is the Harris Corner Detector!
Low-texture region

\[ \sum \nabla I (\nabla I)^T \]
- gradients have small magnitude
- small $\lambda_1$, small $\lambda_2$
\[ \sum \nabla I (\nabla I)^T \]

- gradients very large or very small
- large \( \lambda_1 \), small \( \lambda_2 \)
High-texture region

\[ \sum \nabla I (\nabla I)^T \]

- gradients are different, large magnitudes
- large \( \lambda_1 \), large \( \lambda_2 \)
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)
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.

What would the motion field of a non-rotating ball moving towards the camera look like?
Optical flow

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

• Ideally, optical flow would be the same as the motion field

• Have to be careful: 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
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
Example
Multi-resolution registration

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Optical Flow Results

Lucas-Kanade without pyramids
Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Optical Flow Results

Lucas-Kanade with Pyramids

* From Khurram Shafique
CAP5415 Computer Vision 2003
Errors in Lucas-Kanade

• The motion is large
  – Possible Fix: Keypoint matching, coarse search, multiresolution

• A point does not move like its neighbors
  – Possible Fix: Region-based matching

• Brightness constancy does not hold
  – Possible Fix: Gradient constancy