Today

- Motion
- Optical Flow
- Motion Segmentation
- Feature Tracking

From images to videos

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

• Sometimes, motion is the only cue


Motion and perceptual organization

• Sometimes, motion is foremost cue

Motion and perceptual organization

• Even “impoverished” motion data can evoke a strong percept

Uses of motion

• Estimating 3D structure
• Segmenting objects based on motion cues
• Learning and tracking dynamical models
• Recognizing events and activities

Motion estimation techniques

• Optical flow
  – Recover image motion at each pixel from spatio–temporal image brightness variations (optical flow)

• Feature-tracking
  – Extract visual features (corners, textured areas) and “track” them over multiple frames

Today

• Motion
• Optical Flow
• Motion Segmentation
• Feature Tracking

Optical flow

Vector field function of the spatio-temporal image brightness variations

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

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

Motion field and parallax

- $X(t)$ is a moving 3D point
- Velocity of scene point: $V = \frac{dX}{dt}$
- $x(t) = (x(t), y(t))$ is the projection of $X$ in the image
- Apparent velocity $v$ in the image: given by components $v_x = \frac{dx}{dt}$ and $v_y = \frac{dy}{dt}$
- These components are known as the motion field of the image

To find image velocity $v$, differentiate $x=(x,y)$ with respect to $t$ (using quotient rule):

$$x = f \frac{X}{Z} \quad v_x = f \frac{ZV_z - V_x X}{Z^2} = \frac{fV_z - V_x X}{Z}$$

$$y = f \frac{Y}{Z} \quad v_y = f \frac{V_y - V_z y}{Z}$$

Image motion is a function of both the 3D motion $(V)$ and the depth of the 3D point $(Z)$.

Motion field and parallax

- Pure translation: $V$ is constant everywhere
  
  $$v_x = \frac{fV_x - V_x x}{Z}, \quad v_y = \frac{1}{Z} (v_0 - V_z x), \quad v_0 = (fV_x, fV_y)$$
Motion field and parallax

- Pure translation: $V$ is constant everywhere
  \[
  \mathbf{v} = \frac{1}{Z} (\mathbf{v}_0 - V_z \mathbf{x}), \quad \mathbf{v}_0 = (fV_z, fV_y)
  \]
- The length of the motion vectors is inversely proportional to the depth $Z$
- $V_z$ is nonzero:
  - Every motion vector points toward (or away from) the vanishing point of the translation direction

\[\text{Slide credit: A. Efros}\]

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

\[\text{Figure from Jahne et al, 1999}\]

Estimating optical flow

- Given two subsequent frames, estimate the apparent motion field $u(x,y), v(x,y)$ between them
- Key assumptions
  - **Brightness constancy**: projection of the same point looks the same in every frame
  - **Small motion**: points do not move very far
  - **Spatial coherence**: points move like their neighbors
The brightness constancy constraint

\[
\begin{align*}
\nabla \cdot \begin{bmatrix} u \\ v \\ I_t \end{bmatrix} + I_t &= 0 \\
\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t &= 0
\end{align*}
\]

• How does this make sense?

How many equations and unknowns per pixel?

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

The aperture problem

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

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

\[
\nabla I \cdot \begin{bmatrix} u' \\ v' \end{bmatrix} = 0
\]
The aperture problem

Perceived motion

The barber pole illusion

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

Slide credit: J. Hays

Solving the ambiguity...

- How to get more equations for a pixel?
- Spatial coherence constraint
  - Assume the pixel’s neighbors have the same \((u,v)\)
    - If we use a 5x5 window, that gives us 25 equations per pixel
      \[ 0 = I_t(p) + \nabla I(p) \cdot [u \ v] \]
      \[
      \begin{bmatrix}
      I_x(p_1) & I_y(p_1) \\
      I_x(p_2) & I_y(p_2) \\
      \vdots & \vdots \\
      I_x(p_{25}) & I_y(p_{25}) \\
      \end{bmatrix}
      \begin{bmatrix}
      u \\
      v \\
      \end{bmatrix}
      = -
      \begin{bmatrix}
      I_t(p_1) \\
      I_t(p_2) \\
      \vdots \\
      I_t(p_{25}) \\
      \end{bmatrix}
      \]


Slide credit: S. Savarese
Lucas-Kanade flow

- Overconstrained linear system:
\[
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
\begin{bmatrix}A & d \end{bmatrix}
\]

Conditions for solvability

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

Lucas-Kanade flow

- Overconstrained linear system:
\[
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
\begin{bmatrix}A & d \end{bmatrix}
\]

Least squares solution for \(d\) given by
\[
(\begin{bmatrix}A^T \end{bmatrix}) \begin{bmatrix}d \end{bmatrix} = A^T b
\]

Conditions for solvability

Optimal \((u, v)\) satisfies Lucas-Kanade equation
\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -
\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
\begin{bmatrix}A^T \end{bmatrix}
\]

\[
A^T A
\]

When is this solvable? i.e., what are good points to track?
- \(\lambda_1\) should be invertible
- \(\lambda_1\) should not be too small due to noise
  - eigenvalues \(\lambda_1\) and \(\lambda_2\) of \(A^T A\) should not be too small
- \(\lambda_1\) should be well-conditioned
  - \(\lambda_1 / \lambda_2\) should not be too large (\(\lambda_1\) = larger eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

The summations are over all pixels in the \(K \times K\) window.
M = \( A^T A \) is the second moment matrix!

(Harris corner detector...)

\[ A^T A = \left[ \sum I_x I_x \sum I_y I_y \right] = \sum \left[ \begin{array}{c} I_x \\ I_y \end{array} \right] = \nabla I(\nabla I)^T \]

• Eigenvectors and eigenvalues of \( A^T A \) relate to edge direction and magnitude
  • The eigenvector associated with the largest eigenvalue points in the direction of fastest intensity change
  • The other eigenvector is orthogonal to it

Interpreting the eigenvalues

Classification of image points using eigenvalues of M:

λ₁ and λ₂ are small; E is almost constant in all directions

λ₁ >> λ₂

E increases in all directions

λ₁ >> λ₂

“Corner”

“Edge”

“Edge”

“Flat” region

Low-texture region

- gradients have small magnitude
  - small \( l_1 \), small \( l_2 \)

Edge

- gradients very large or very small
  - large \( l_1 \), small \( l_2 \)
High-texture region

\[ \sum \nabla I(\nabla I)^T \]
- gradients are different, large magnitudes
- large \( l_1 \), large \( l_2 \)

What are good features to track?
- Can measure “quality” of features from just a single image
- Hence: tracking Harris corners (or equivalent) guarantees small error sensitivity!

Recap
- Key assumptions (Errors in Lucas–Kanade)
  - **Small motion**: points do not move very far
  - **Brightness constancy**: projection of the same point looks the same in every frame
  - **Spatial coherence**: points move like their neighbors

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

*From Khurram Hassan-Shafique CAP5415 Computer Vision 2003*
Reduce the resolution!

Multi-resolution Lucas Kanade Algorithm

- Compute ‘simple’ LK at highest level
- At level $i$
  - Take flow $u_{i-1}, v_{i-1}$ from level $i-1$
  - Bilinear interpolate it to create $u_i^*, v_i^*$ matrices of twice resolution for level $i$
  - Multiply $u_i^*, v_i^*$ by 2
  - Compute $f_i$ from a block displaced by $u_i^*(x,y), v_i^*(x,y)$
  - Apply LK to get $u_i^*(x, y), v_i^*(x, y)$ (the correction in flow)
  - Add corrections $u_i^* v_i^*$, i.e. $u_i = u_i^* + u_i^*$, $v_i = v_i^* + v_i^*$.

Coarse-to-fine optical flow estimation

Iterative Refinement

- Iterative Lukas-Kanade Algorithm
  1. Estimate displacement at each pixel by solving Lucas-Kanade equations
  2. Warp $I(t)$ towards $I(t+1)$ using the estimated flow field
     - Basically, just interpolation
  3. Repeat until convergence
Coarse-to-fine optical flow estimation

- Run iterative L-K
- Warp & upsample

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

Optical Flow Results

- Lucas-Kanade without pyramids
- Fails in areas of large motion

Recap

- Key assumptions (Errors in Lucas-Kanade)
  - **Small motion**: points do not move very far
  - **Brightness constancy**: projection of the same point looks the same in every frame
  - **Spatial coherence**: points move like their neighbors
Today

- Motion
- Optical Flow
- Motion Segmentation
- Feature Tracking

Motion segmentation

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

What are layers?

- Each layer is defined by an alpha mask and an affine motion model

- How do we represent the motion in this scene?
Affine motion

\[ u(x, y) = a_1 + a_2 x + a_3 y \]
\[ v(x, y) = a_4 + a_5 x + a_6 y \]

- Substituting into the brightness constancy equation:

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

- Each pixel provides 1 linear constraint in 6 unknowns

Least squares minimization:

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

How do we estimate the layers?

1. Obtain a set of initial affine motion hypotheses
   - Divide the image into blocks and estimate affine motion parameters in each block by least squares
     - Eliminate hypotheses with high residual error
   - Map into motion parameter space
   - Perform k-means clustering on affine motion parameters
     - Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene

2. Iterate until convergence:
   - Assign each pixel to best hypothesis
     - Pixels with high residual error remain unassigned

How do we estimate the layers?
How do we estimate the layers?

1. Obtain a set of initial affine motion hypotheses
   - Divide the image into blocks and estimate affine motion parameters in each block by least squares
     • Eliminate hypotheses with high residual error
   - Map into motion parameter space
   - Perform k-means clustering on affine motion parameters
     • Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene

2. Iterate until convergence:
   - Assign each pixel to best hypothesis
     • Pixels with high residual error remain unassigned
   - Perform region filtering to enforce spatial constraints
   - Re-estimate affine motions in each region

Today

• Motion
• Optical Flow
• Motion Segmentation
• Feature Tracking

Example result


Tracking

Fei-Fei Li, K. Grauman, D. Ramanan.
Why is tracking challenging?

- Small, few visual features
- Erratic movements, moving very quickly
- Occlusions, leaving and coming back
- Surrounding similar-looking objects

Strategies for tracking

- Tracking by repeated detection
  - Works well if object is easily detectable (e.g., face or colored glove) and there is only one
  - Need some way to link up detections
  - Best you can do, if you can’t predict motion

Tracking with dynamics

- Key idea: Based on a model of expected motion, predict where objects will occur in next frame, before even seeing the image
  - Restrict search for the object
  - Measurement noise is reduced by trajectory smoothness
  - Robustness to missing or weak observations

Optical flow for tracking?

If we have more than just a pair of frames, we could compute flow from one to the next:

But flow only reliable for small motions, and we may have occlusions, textureless regions that yield bad estimates anyway...
Motion estimation techniques

• Optical flow
  – Directly recover image motion at each pixel from spatio-temporal image brightness variations
  – Dense motion fields, but sensitive to appearance variations
  – Suitable for video and when image motion is small

• Feature-based methods
  – Extract visual features (corners, textured areas) and track them over multiple frames
  – Sparse motion fields, but more robust tracking
  – Suitable when image motion is large (10s of pixels)

Feature-based matching for motion

• For a discrete matching search, what are the tradeoffs of the chosen search window size?

• Which patches to track?
  – Select interest points – e.g. corners

• Where should the search window be placed?
  – Near match at previous frame
  – More generally, taking into account the expected dynamics of the object

Example: A Camera Mouse

Video interface: use feature tracking as mouse replacement

• User clicks on the feature to be tracked
• Take the 15x15 pixel square of the feature
• In the next image do a search to find the 15x15 region with the highest correlation
• Move the mouse pointer accordingly
• Repeat in the background every 1/30th of a second

James Gips and Margrit Betke
http://www.bc.edu/schools/csom/eagleeyes/
Feature-based matching for motion

- For a discrete matching search, what are the tradeoffs of the chosen search window size?
- Which patches to track?
  - Select interest points – e.g., corners
- Where should the search window be placed?
  - Near match at previous frame
  - More generally, taking into account the expected dynamics of the object

Detection vs. tracking

Detection: We detect the object independently in each frame and can record its position over time, e.g., based on blob’s centroid or detection window coordinates

Tracking with dynamics: We use image measurements to estimate position of object, but also incorporate position predicted by dynamics, i.e., our expectation of object’s motion pattern.
Detection vs. tracking

Tracking with dynamics: We use image measurements to estimate position of object, but also incorporate position predicted by dynamics, i.e., our expectation of object’s motion pattern.

Tracking with dynamics
• Use model of expected motion to predict where objects will occur in next frame, even before seeing the image.
• Intent:
  – Do less work looking for the object, restrict the search.
  – Get improved estimates since measurement noise is tempered by smoothness, dynamics priors.
• Assumption: continuous motion patterns:
  – Camera is not moving instantly to new viewpoint
  – Objects do not disappear and reappear in different places in the scene
  – Gradual change in pose between camera and scene

Tracking as inference
• The hidden state consists of the true parameters we care about, denoted $X$.
• The measurement is our noisy observation that results from the underlying state, denoted $Y$.
• At each time step, state changes (from $X_{t-1}$ to $X_t$) and we get a new observation $Y_t$.

State vs. observation
• Hidden state: parameters of interest
• Measurement: what we get to directly observe

Slide credit: K. Grauman
Tracking as inference

- The hidden state consists of the true parameters we care about, denoted $X$.
- The measurement is our noisy observation that results from the underlying state, denoted $Y$.
- At each time step, state changes (from $X_{t-1}$ to $X_t$) and we get a new observation $Y_t$.
- Our goal: recover most likely state $X_t$ given
  - All observations seen so far.
  - Knowledge about dynamics of state transitions.

Independence assumptions

- Only immediate past state influences current state
  $$P(X_t | X_0, \ldots, X_{t-1}) = P(X_t | X_{t-1})$$
  dynamics model
- Measurement at time $t$ depends on current state
  $$P(Y_t | X_0, \ldots, X_{t-1}, Y_{t-1}, X_t) = P(Y_t | X_t)$$
  observation model
Tracking as inference

- **Prediction:**
  
  Given the measurements we have seen up to this point, what state should we predict?

  \[ P(\mathbf{X}_t | y_0, \ldots, y_{t-1}) \]

- **Correction:**
  
  Now given the current measurement, what state should we predict?

  \[ P(\mathbf{X}_t | y_0, \ldots, y_t) \]

Questions

- How to represent the known dynamics that govern the changes in the states?
- How to represent relationship between state and measurements, plus our uncertainty in the measurements?
- How to compute each cycle of updates?

**Representation:** We’ll consider the class of linear dynamic models, with associated Gaussian pdfs.

**Updates:** via the Kalman filter.

Notation reminder

\[ \mathbf{x} \sim N(\mu, \Sigma) \]

- Random variable with Gaussian probability distribution that has the mean vector \( \mu \) and covariance matrix \( \Sigma \).
- \( \mathbf{x} \) and \( \mu \) are d-dimensional, \( \Sigma \) is d x d.

If \( \mathbf{x} \) is 1-d, we just have one \( \Sigma \) parameter \( \sigma^2 \).

Linear dynamic model

- Describe the *a priori* knowledge about
  
  - System dynamics model: represents evolution of state over time.
    \[ \mathbf{x}_t \sim N(D\mathbf{x}_{t-1}; \Sigma_d) \]

  - Measurement model: at every time step we get a noisy measurement of the state.
    \[ \mathbf{y}_t \sim N(M\mathbf{x}_t; \Sigma_m) \]
Example: randomly drifting points
\[ x_t \sim N(Dx_{t-1}; \Sigma_d) \]
- Consider a stationary object, with state as position
- Position is constant, only motion due to random noise term.
- State evolution is described by identity matrix \( D=I \)

Example: Constant velocity (1D points)
\[ x_t \sim N(Dx_{t-1}; \Sigma_d) \quad y_t \sim N(Mx_t; \Sigma_m) \]
- State vector: position \( p \) and velocity \( v \)
  \[ x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \quad p_t = x_t \quad v_t = \frac{d}{dt} x_t \]
  \[ x_t = D x_{t-1} + \text{noise} = \]
- Measurement is position only
  \[ y_t = M x_t + \text{noise} = \]

Questions
- How to represent the known dynamics that govern the changes in the states?
- How to represent relationship between state and measurements, plus our uncertainty in the measurements?
- How to compute each cycle of updates?

Representation: We’ll consider the class of linear dynamic models, with associated Gaussian pdfs.

Updates: via the Kalman filter.
The Kalman filter

- Method for tracking linear dynamical models in Gaussian noise
- The predicted/corrected state distributions are Gaussian
  - Only need to maintain the mean and covariance
  - The calculations are easy

1D Kalman filter: **Prediction**

- Have linear dynamic model defining predicted state evolution, with noise
  \[ X_t \sim N(dx_{t-1}, \sigma_d^2) \]
- Want to estimate predicted distribution for next state
  \[ P(X_t|y_0, \ldots, y_{t-1}) = N(\mu_t^-, (\sigma_t^-)^2) \]
- Update the mean:
  \[ \mu_t^- = d\mu_{t-1} \]
- Update the variance:
  \[ (\sigma_t^-)^2 = \sigma_d^2 + (d\sigma_{t-1})^2 \]

1D Kalman filter: **Correction**

- Have linear model defining the mapping of state to measurements:
  \[ Y_t \sim N(mx_t, \sigma_m^2) \]
- Want to estimate corrected distribution given latest meas.:
  \[ P(X_t|y_0, \ldots, y_t) = N(\mu_t^+, (\sigma_t^+)^2) \]
- Update the mean:
  \[ \mu_t^+ = \frac{\mu_t^- \sigma_m^2 + m y_t (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2} \]
- Update the variance:
  \[ (\sigma_t^+)^2 = \frac{\sigma_m^2 (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2} \]
Prediction vs. correction

\[ \mu_t^+ = \frac{\mu_t^- \sigma_m^2 + m \nu_t^- (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2} \quad (\sigma_t^+)^2 = \frac{\sigma_m^2 (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2} \]

- What if there is no prediction uncertainty \((\sigma_t^- = 0)\)?
  \[ \mu_t^+ = \mu_t^- \quad (\sigma_t^+)^2 = 0 \]
  The measurement is ignored!

- What if there is no measurement uncertainty \((\sigma_m = 0)\)?
  \[ \mu_t^+ = \frac{y_t^-}{m} \quad (\sigma_t^+)^2 = 0 \]
  The prediction is ignored!

Slide credit: L. Lazebnik

Constant velocity model

\[ \sigma_{\text{state}} = \begin{bmatrix} \sigma_{x_{\text{state}}} & \sigma_{y_{\text{state}}} & \sigma_{z_{\text{state}}} \end{bmatrix}^T, \quad \sigma_{\text{measurement}} = \begin{bmatrix} \sigma_{x_{\text{measurement}}} & \sigma_{y_{\text{measurement}}} & \sigma_{z_{\text{measurement}}} \end{bmatrix}^T \]

\[ \sigma_{\text{predicted estimate}} = \begin{bmatrix} \sigma_{x_{\text{predicted estimate}}} & \sigma_{y_{\text{predicted estimate}}} & \sigma_{z_{\text{predicted estimate}}} \end{bmatrix}^T, \quad \sigma_{\text{corrected estimate}} = \begin{bmatrix} \sigma_{x_{\text{corrected estimate}}} & \sigma_{y_{\text{corrected estimate}}} & \sigma_{z_{\text{corrected estimate}}} \end{bmatrix}^T \]

Bars: variance estimates before and after measurements

Slide credit: K. Grauman
Kalman filter processing

- state
- measurement
- predicted mean estimate
- corrected mean estimate
- bars: variance estimates before and after measurements

Constant velocity model

Example: Kalman Filter

Prediction

Correction

Update Location, Velocity, etc.

Comparison

Ground Truth
Observation
Correction

http://www.cs.bu.edu/~betke/research/bats/
A bat census

http://www.cs.bu.edu/~betke/research/bats/

Tracking: issues

• Initialization
  – Often done manually
  – Background subtraction, detection can also be used
• Data association, multiple tracked objects
  – Occlusions, clutter
  – Which measurements go with which tracks?

• Deformable and articulated objects
• Constructing accurate models of dynamics
  – E.g., Fitting parameters for a linear dynamics model
• Drift
  – Accumulation of errors over time


Drift
Tracking Summary

• Tracking as inference
  – Goal: estimate posterior of object position given measurement
• Linear models of dynamics
  – Represent state evolution and measurement models
• Kalman filters
  – Recursive prediction/correction updates to refine measurement
• General tracking challenges