Visual Motion
Analysis and Representation
Example

- Ullman’s concentric counter-rotating cylinder experiment
- Two concentric cylinders of different radii
- W. a random dot pattern on both surfaces (cylinder surfaces and boundaries are not displayed)
- Stationary: not able to tell them apart
- Counter-rotating: structures apparent
Motion helps in
- segmentation (two structures)
- identification (two cylinders)
Classes of Techniques

- **Feature-based methods**
  - Extract visual features (corners, textured areas) and track them
  - Sparse motion fields, but possibly robust tracking
  - Suitable especially when image motion is large (10s of pixels)

- **Direct-methods (Pixel-based methods)**
  - Directly recover image motion from spatio-temporal image brightness variations
  - Global motion parameters directly recovered without an intermediate feature motion calculation
  - Dense motion fields, but more sensitive to appearance variations
  - Suitable for video and when image motion is small (< 10 pixels)
Optical flow and motion analysis

- Now we move to considering images that vary over time – image sequences
  - Typical case is video – images captured at 30 frames/second (or 15, or 60, or ...)
  - $I(x, y, t) \rightarrow I_1(x, y) = I(x, y, t_1), I_2(x, y) = I(x, y, t_2)$, etc.
  - “Spatial-temporal space” describes $(x, y, t)$

What can change between $I_t$ and $I_{t+1}$?

What do images close in time have in common?
Spatio-temporal image data (examples)
Optical flow and motion analysis

- **Optical flow** is the *apparent motion* of brightness patterns in the image sequence
  - A 2D vector at each point – a vector field

- The **motion field** is the *true motion* (3D) at each point, mapped onto the 2D image
  - A vector field

- They are not always the same
  - E.g., white, featureless ball?

In general, we estimate the **motion field** by computing the **optical flow**

The motion field is not *directly* observed
Figure 8.6 The motion field of a pilot looking to the right in level flight. The field of expansion here is off at infinity to the left of the figure; equivalently, the field of contraction is off at infinity to the right of the figure. (From [Gibson 1950] with permission. Copyright © 1977, 1950 by Houghton Mifflin Company.)
Example

Fig. 7.7 Optical flow from feature point analyses. (a) An image. (b) Later image. (c) Optical flow found by relaxation.
Caveats

- Motion analysis a very important and popular area in computer vision
- A large body of literature exists with maybe hundreds of different formulations (At CVPR, you will find at least 2 or 3 sessions on motion)
- Many of them can be very mathematical
- Apparent motion != True motion
Rigid vs. nonrigid motion

- Camera motion is 6 DOF rigid motion
- Object motion may be rigid or nonrigid
  - Rigid: coffee mugs, silverware, baseballs, jets, ...
  - Nonrigid: humans, face, medical imagery, beach balls, scissors, grass, ...
    - Includes *articulated* motion
Nonrigid motion is complicated and difficult, especially with little prior knowledge on what is being viewed.

Typical problem: What are the parameters of the known nonrigid model of the object being viewed?

We’ll just focus on rigid motion.
The barber’s pole illusion
The aperture problem

- In local processing, we can only measure motion perpendicular to the image gradient
First steps

- Motion processing starts with estimating optical flow from frame to frame, either densely or sparsely.

- The typical approaches are:
  - Dense correspondence:
    - Differential methods, local area/correlation based
    - This could be hierarchical (coarse-to-fine approach)
  - Sparse correspondence
    - Matching methods, feature based

- Assumption: Points/features can be matched in nearby images
**Brightness constancy equation**

\[ \frac{d I}{d t} = \frac{d I(x(t), y(t), t)}{d t} = 0 \]

For a given scene point

\[ \frac{d I(x(t), y(t), t)}{d t} = \frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} \frac{dt}{dt} = 0 \]

\[ \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)^T \left( \frac{dx}{dt}, \frac{dy}{dt} \right) + \frac{\partial I}{\partial t} = 0 \]

\[ \nabla I \cdot \mathbf{v} + I_t = 0 \]

- \( \nabla I \) Image gradient
- \( \mathbf{v} \) Optical flow
- \( I_t \) Time difference
**Brightness constancy equation**

*(method #2)*

\[ I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \]

For a given scene point

\[ I(x + \delta x, y + \delta y, t + \delta t) - I(x, y, t) = 0 \]

\[ I(x, y, z) + \frac{\partial I(x, y, t)}{\partial x} dx + \frac{\partial I(x, y, t)}{\partial y} dy + \frac{\partial I(x, y, t)}{\partial t} dt \approx 0 \text{ by Taylor expansion} \]

\[ \frac{\partial I(x, y, t)}{\partial x} dx + \frac{\partial I(x, y, t)}{\partial y} dy + \frac{\partial I(x, y, t)}{\partial t} dt = 0 \]

\[ \frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \]

\[ \nabla I \cdot v + I_t = 0 \]
Brightness constancy equation
(method #2)

\[ I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \]
Back to the aperture problem

\[ \nabla I \cdot v + I_t = 0 \]
\[ \nabla I \cdot v = -I_t \]

Many vectors \( v \) satisfy this
Only the normal direction is constrained
On images…

\[ \nabla I \cdot v + I_t = 0 \]

This equation defines and constrains the optical flow \( v(x, y) \)
What is the image gradient?

Image gradient – the first derivative (slope) of the intensity variation in \((x, y)\)
What is the temporal gradient?
$$\frac{\partial I}{\partial t}$$
Brightness constancy of a point

Scene

Image sequence

$I_1(x(t_1), y(t_1), t_1)$ = $I_3(x(t_3), y(t_3), t_3)$

$I_2(x(t_2), y(t_2), t_2)$ = $I_3(x(t_3), y(t_3), t_3)$
**Difficulty**

- One equation with two unknowns
- Aperture problem
  - spatial derivatives use only a few adjacent pixels (limited aperture and visibility)
  - many combinations of \((u,v)\) will satisfy the equation

\[
I_x u + I_y v + I_t = 0
\]
- intensity gradient is zero
  no constraints on \((u, v)\)
  \((0,0) \cdot (u, v) = 0\)
  interpolated from other places

- intensity gradient is nonzero
  but is constant
  \(\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \cdot (u, v) = -\frac{\partial I}{\partial t}\)
  one constraints on \((u, v)\)
  only the component along the gradient are recoverable

- intensity gradient is nonzero
  and changing
  multiple constraints on \((u, v)\)
  motion recoverable

\[
\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \cdot (u, v) = -\frac{\partial I}{\partial t}_{(x_1, y_1)}
\]

\[
\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \cdot (u, v) = -\frac{\partial I}{\partial t}_{(x_2, y_2)}
\]
Patch Translation [Lucas-Kanade]

Assume a single velocity for all pixels within an image patch

$$E(u, v) = \sum_{x, y \in \Omega} \left( I_x(x, y)u + I_y(x, y)v + I_t \right)^2$$

Minimizing

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$\left( \sum \nabla I \nabla I^T \right) \hat{U} = -\sum \nabla I I_t$$

LHS: sum of the 2x2 outer product of the gradient vector
How do we determine correspondences?

Assume all change between frames is due to motion:

\[ J(x, y) \approx I(x + u(x, y), y + v(x, y)) \]
The Aperture Problem

Let \( M = \sum (\nabla I)(\nabla I)^T \) and \( b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix} \)

- Algorithm: At each pixel compute \( U \) by solving \( MU = b \)

- \( M \) is singular if all gradient vectors point in the same direction
  - e.g., along an edge
  - of course, trivially singular if the summation is over a single pixel or there is no texture
  - i.e., only \textit{normal flow} is available (aperture problem)

- Corners and textured areas are OK
SSD Surface – Textured area
SSD Surface -- Edge
SSD – homogeneous area
Limits of the gradient method

Fails when intensity structure in window is poor

Fails when the displacement is large (typical operating range is motion of 1 pixel)

*Linearization of brightness is suitable only for small displacements*

- Also, brightness is not strictly constant in images
  *actually less problematic than it appears, since we can pre-filter images to make them look similar*
Coarse-to-Fine Estimation

Pyramid of image J

Pyramid of image I

warp \rightarrow J^w \rightarrow \text{refine}

\Delta \bar{a}

\bar{a}

u=1.25 \text{ pixels}

u=2.5 \text{ pixels}

u=5 \text{ pixels}

u=10 \text{ pixels}

Szeliski
Iterative Refinement

- Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation
- Warp one image toward the other using the estimated flow field
  *(easier said than done)*
- Refine estimate by repeating the process
Optical Flow: Iterative Estimation

Initial guess: \( d_0 = 0 \)
Estimate: \( d_1 = d_0 + \hat{d} \)

(\text{using } d \text{ for displacement here instead of } u)
Optical Flow: Iterative Estimation

Initial guess: \( d_1 \)
Estimate: \( d_2 = d_1 + \hat{d} \)
Optical Flow: Iterative Estimation

Initial guess: $d_2$
Estimate: $d_3 = d_2 + \hat{d}$
Temporal coherency

\[ (u, v) \quad (u, v) \quad (u, v) \]

\[ \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \cdot (u, v) = - \frac{\partial I}{\partial t} \quad (u, v) \]

• Caveat:
  – \((u, v)\) must stay the same across several frames
  – scenes highly textured
  – \((u, v)\) at the same location actually refers to different object points
Spatial coherency

- neighboring pixels should have “similar” flow vector
- Q: What do you mean by “similar”
- A1: identical
- A2: change slowly

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial y} = \frac{\partial v}{\partial x} = \frac{\partial v}{\partial y} = 0$$

$$\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \cong 0$$

$$(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 \text{ small}$$
Mathematical formulation

- Based on Lagrange Multiplier
- Incorporate smoothness as an additional constraint
- Can be thought of as a weighting of two terms:
  - optical flow constraint
  - smoothness constraint

\[
\left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) \cdot (u, v) = -\frac{\partial I}{\partial t}
\]

\[
\left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2 + \left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2
\]
Optimize over all image plane:

\[ E = \int \int \left( \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right)^2 + \lambda \left[ \left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial u}{\partial y} \right)^2 + \left( \frac{\partial v}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2 \right] dx \, dy \]

Discretize the governing equation, at \((i,j)\):

\[
\begin{align*}
\frac{\partial u}{\partial x} &= u_{i+1,j} - u_{i,j} \\
\frac{\partial u}{\partial y} &= u_{i,j+1} - u_{i,j} \\
\frac{\partial v}{\partial x} &= v_{i+1,j} - v_{i,j} \\
\frac{\partial v}{\partial y} &= v_{i,j+1} - v_{i,j}
\end{align*}
\]

Discretized expression:

\[
E = \sum_i \sum_j \left( \frac{\partial I}{\partial x}_{i,j} u_{i,j} + \frac{\partial I}{\partial y}_{i,j} v_{i,j} + \frac{\partial I}{\partial t}_{i,j} \right)^2 \\
+ \lambda \left[ \left( u_{i+1,j} - u_{i,j} \right)^2 + \left( u_{i,j+1} - u_{i,j} \right)^2 + \left( v_{i+1,j} - v_{i,j} \right)^2 + \left( v_{i,j+1} - v_{i,j} \right)^2 \right]
\]
• At a pixel location \((k,l)\):

\[
\frac{\partial E}{\partial u_{k,l}} = 2\left( \frac{\partial I}{\partial x_{k,l}} u_{k,l} + \frac{\partial I}{\partial y_{k,l}} v_{k,l} + \frac{\partial I}{\partial t_{k,l}} \right) \frac{\partial I}{\partial x_{k,l}} \\
- 2 \lambda [(u_{k-1,l} - u_{k,l}) + (u_{k,l-1} - u_{k,l}) + (u_{k+1,l} - u_{k,l}) + (u_{k,l+1} - u_{k,l})] = 0
\]

\[
\frac{\partial E}{\partial v_{k,l}} = 2\left( \frac{\partial I}{\partial x_{k,l}} u_{k,l} + \frac{\partial I}{\partial y_{k,l}} v_{k,l} + \frac{\partial I}{\partial t_{k,l}} \right) \frac{\partial I}{\partial y_{k,l}} \\
- 2 \lambda [(v_{k-1,l} - v_{k,l}) + (v_{k,l-1} - v_{k,l}) + (v_{k+1,l} - v_{k,l}) + (v_{k,l+1} - v_{k,l})] = 0
\]

\[
\frac{\partial E}{\partial \lambda} = \ldots
\]
• Putting it all together:

\[
\left( \frac{\partial I}{\partial x_{k,l}} \right)^2 u_{k,l} + \frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial y_{k,l}} v_{k,l} + \frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial t_{k,l}} - 4 \lambda (\bar{u} - u_{k,l}) = 0
\]

\[
\frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial y_{k,l}} u_{k,l} + \left( \frac{\partial I}{\partial y_{k,l}} \right)^2 v_{k,l} + \frac{\partial I}{\partial y_{k,l}} \frac{\partial I}{\partial t_{k,l}} - 4 \lambda (\bar{v} - v_{k,l}) = 0
\]

\[
\bar{u} = \frac{(u_{k-1,l} + u_{k,l-1} + u_{k+1,l} + u_{k,l+1})}{4}
\]

\[
\bar{v} = \frac{(v_{k-1,l} + v_{k,l-1} + v_{k+1,l} + v_{k,l+1})}{4}
\]
Or:

\[ [4\lambda + \left( \frac{\partial I}{\partial x_{k,l}} \right)^2] u_{k,l} + \frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial y_{k,l}} v_{k,l} = 4\lambda u - \frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial t_{k,l}} \]

\[ \frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial y_{k,l}} u_{k,l} + [4\lambda + \left( \frac{\partial I}{\partial y_{k,l}} \right)^2] v_{k,l} = 4\lambda v - \frac{\partial I}{\partial y_{k,l}} \frac{\partial I}{\partial t_{k,l}} \]

\[ u_{k,l} = -u - \frac{\frac{\partial I}{\partial x_{k,l}} - u + \frac{\partial I}{\partial y_{k,l}} - v + \frac{\partial I}{\partial t_{k,l}}}{4\lambda + \left( \frac{\partial I}{\partial x_{k,l}} \right)^2 + \left( \frac{\partial I}{\partial y_{k,l}} \right)^2} \frac{\partial I}{\partial x_{k,l}} \]

\[ v_{k,l} = -v - \frac{\frac{\partial I}{\partial x_{k,l}} - u + \frac{\partial I}{\partial y_{k,l}} - v + \frac{\partial I}{\partial t_{k,l}}}{4\lambda + \left( \frac{\partial I}{\partial x_{k,l}} \right)^2 + \left( \frac{\partial I}{\partial y_{k,l}} \right)^2} \frac{\partial I}{\partial y_{k,l}} \]
\[ u_{k,l} = u \]
\[ v_{k,l} = v \]

\[ \frac{\partial I}{\partial x_{k,l}} u + \frac{\partial I}{\partial y_{k,l}} v + \frac{\partial I}{\partial t_{k,l}} \]

\[ 4 \lambda + \left( \frac{\partial I}{\partial x_{k,l}} \right)^2 + \left( \frac{\partial I}{\partial y_{k,l}} \right)^2 \]

- estimate based on smoothness
- how much does the smooth estimate violate optical flow constraint
- how much does the optical flow constraint matters
- direction for correction
Algorithms

1. Compute $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}$ from a pair of input images

2. Choose a weighting factor $\lambda$

3. Compute $(\overline{u}, \overline{v})$

4. At each pixel location $(k,l)$, do

$$u_{k,l}^{(n+1)} = \overline{u}^{(n)} - \frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial y_{k,l}} \frac{\partial I}{\partial t_{k,l}} \left( \frac{\partial I}{\partial x_{k,l}} + \frac{\partial I}{\partial y_{k,l}} \right)^2 + \frac{\partial I}{\partial t_{k,l}} \left( \frac{\partial I}{\partial x_{k,l}} + \frac{\partial I}{\partial y_{k,l}} \right)^2 \frac{\partial I}{\partial x_{k,l}}$$

$$v_{k,l}^{(n+1)} = \overline{v}^{(n)} - \frac{\partial I}{\partial x_{k,l}} \frac{\partial I}{\partial y_{k,l}} \frac{\partial I}{\partial t_{k,l}} \left( \frac{\partial I}{\partial x_{k,l}} + \frac{\partial I}{\partial y_{k,l}} \right)^2 + \frac{\partial I}{\partial t_{k,l}} \left( \frac{\partial I}{\partial x_{k,l}} + \frac{\partial I}{\partial y_{k,l}} \right)^2 \frac{\partial I}{\partial y_{k,l}}$$

5. Iterate steps 3 and 4 until no change or count exceeds
Results

Figure 12-8. Four frames of a synthetic image sequence showing a sphere slowly rotating in front of a randomly patterned background.
Motion representations

- How can we describe this scene?
Block-based motion prediction

- Break image up into square blocks
- Estimate translation for each block
- Use this to predict next frame, code difference (MPEG-2)
Layered motion

- Break image sequence up into “layers”:

- Describe each layer's motion
Layered motion

- Advantages:
  - can represent occlusions / disocclusions
  - each layer’s motion can be smooth
  - video segmentation for semantic processing

- Difficulties:
  - how do we determine the correct number?
  - how do we assign pixels?
  - how do we model the motion?
Layers for video summarization

Frame 0

Frame 50

Frame 80

Background scene (players removed)

Complete synopsis of the video
Background modeling (MPEG-4)

- Convert masked images into a background sprite for layered video coding
Optical flow summary

- Optical flow techniques:
  - Techniques that estimate the motion field from the image brightness constancy equation

- Optical flow:
  - Is best estimated (least noisy) at image points with high spatial image gradients. (Why?)
  - Works best for Lambertian surfaces (Why?)
  - Works best for very high frame rates (Why?)

- From optical flow, we can compute shape/structure/depth, motion parameters, segmentation, etc.
  - But if you primarily want to **track** an object, other methods may be preferred
Tracking

- Tracking is the process of updating an object’s position (and orientation, and articulation?) over time through a video sequence
  - Estimate the object *pose* at each time point
    - “Pose” – position and orientation

- Applications
  - Surveillance
  - Targeting
  - Motion-based recognition (e.g., gesture recognition, computation of egomotion)
  - Motion analysis (golf swing, gait, character animation)
  - ........
Tracking vs. optical flow

- In tracking, we are generally acknowledging that some sparse features are the points to track
  - Corners, lines, regions, patterns, contours....

- Rather than computing the full motion field from optical flow, let’s keep track of the time-varying position of these sparse features
  - Then compute \{egomotion, object pose, etc.\} from this

- This typically involves a loop of prediction, measurement, and correction
  - Often with presumed models of motion dynamics and measurement noise
Tracking vs. Matching

- Tracking requires videos
- Small displacement is assumed
- Simple features
- Use image constraint (similar to optical flow constraint)

- Matching can be done on discrete frames
- Displacement can be large (>10 pixels)
- Often more elaborate features
- Independent detection in each frame and then match
Examples LKT tracker

\[ F(x + h) \approx F(x) + hF'(x) \]

\[ E = \sum_x [F(x + h) - G(x)]^2 \]

\[ 0 = \frac{\partial E}{\partial h} \approx \frac{\partial}{\partial h} \sum_x [F(x) + hF'(x) - G(x)]^2 \]

\[ = \sum_x 2F'(x) [F(x) + hF'(x) - G(x)] \]

\[ \Rightarrow h \approx \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x F'(x)^2} \]
KLT tracker

- An iterative update algorithm
- The estimate is more accurate if $F$ is indeed linear
- Penalize pixels with large 2\textsuperscript{nd} derivatives ($w(x)$)

\[
h \approx \frac{G(x) - F(x)}{F'(x)}
\]

\[
F''(x) \approx \frac{G'(x) - F'(x)}{h} \quad \Rightarrow \quad w(x) = \frac{1}{|G'(x) - F'(x)|}
\]

\[
\begin{align*}
  h_0 &= 0 \\
  h_{k+1} &= h_k + \frac{\sum_x w(x) F'(x + h_k) [G(x) - F(x + h_k)]}{\sum_x w(x) F'(x + h_k)^2}
\end{align*}
\]