Motion

Computer Vision I
CSE 252A
Lecture 13

Announcements
• Homework 2 is due Sun, Nov 8, 11:59 PM
• Reading:
  - Section 10.6.1: Optical Flow and Motion
  - Section 10.6.2: Flow Models
  - Introductory Techniques for 3-D Computer Vision, Trucco and Verri
    • Chapter 8: Motion

Continuous Motion
• Consider a video camera moving continuously along a trajectory (rotating & translating).
• How do points in the image move?
• What does that tell us about the 3-D motion & scene structure?

Motion
“When objects move at equal speed, those more remote seem to move more slowly.”
- Euclid, 300 BC

Simplest Idea for video processing
• Given image I(u,v,t) and I(u,v, t+δt), compute I(u,v, t+δt) - I(u,v,t).
• This is partial derivative: $\frac{\partial I}{\partial t}$
• At object boundaries, $\frac{\partial I}{\partial t}$ is large and is a cue for segmentation
• Does not indicate which way objects are moving

Background Subtraction
• Gather image I(x,y,t₀) of background without objects of interest (perhaps computed over average over many images).
• At time t, pixels where |I(x,y,t) - I(x,y,t₀)| > τ are labeled as coming from foreground objects
The Motion Field
Where in the image did a point move?
Down and left

What causes a motion field?
1. Camera moves (translates, rotates)
2. Objects in scene move rigidly
3. Objects articulate (pliers, humans, animals)
4. Objects bend and deform (fish)
5. Blowing smoke, clouds

Is motion estimation inherent in humans?
Demo
http://michaelbach.de/ot/cog-hiddenBird/index.html

Rigid Motion and the Motion Field

Rigid Motion: General Case
\[ \dot{p} = T + \omega \times p \]
Position and orientation of a rigid body
Rotation Matrix & Translation vector
Rigid Motion:
Velocity Vector: \( T \)
Angular Velocity Vector: \( \omega \) (or \( \Omega \))
General Motion

\[
\begin{bmatrix}
    u \\
v
\end{bmatrix} = \frac{f}{z} \begin{bmatrix}
    x \\
y
\end{bmatrix}
\]

\[
\begin{bmatrix}
    \dot{u} \\
\dot{v}
\end{bmatrix} = \frac{f}{z} \begin{bmatrix}
    \dot{x} \\
\dot{y}
\end{bmatrix} - \frac{z}{z} \begin{bmatrix}
    \dot{x} \\
\dot{y}
\end{bmatrix}
\]

\[
= \frac{f}{z} \begin{bmatrix}
    \dot{x} \\
\dot{y}
\end{bmatrix} - \frac{z}{z} \begin{bmatrix}
    \dot{u} \\
\dot{v}
\end{bmatrix}
\]

Substitute \( \dot{p} = T + \omega \times p \) where \( p = (x, y, z)^T \)

Motion Field Equation

\[
\begin{aligned}
\dot{u} &= \frac{T_u - T f}{Z} - \omega_f x + \omega_v y + \frac{\omega u v}{f} - \frac{\omega u^2}{f} \\
\dot{v} &= \frac{T_v - T f}{Z} + \omega_f y - \omega_v x - \frac{\omega u v}{f} - \frac{\omega v^2}{f}
\end{aligned}
\]

- \( T \): Components of 3-D linear motion
- \( \omega \): Angular velocity vector
- \( (u, v) \): Image point coordinates
- \( Z \): depth
- \( f \): focal length

Pure Translation

\( \omega = 0 \)

\[
\begin{aligned}
\dot{u} &= \frac{T_u - T f}{Z} + \frac{\omega u v}{f} - \frac{\omega u^2}{f} \\
\dot{v} &= \frac{T_v - T f}{Z} - \frac{\omega u v}{f} - \frac{\omega v^2}{f}
\end{aligned}
\]

Forward Translation & Focus of Expansion

[ Gibbon, 1950 ]

Motion Field Yields 3-D Motion Information

The "instantaneous" velocity of points in an image

The Focus of Expansion (FOE)

Intersection of velocity vector with image plane

With just this information it is possible to calculate:

1. Direction of motion
2. Time to collision

Pure Translation

Parallel (FOE point at infinity)

Radial about FOE

\( T_z = 0 \)

Motion parallel to image plane
Pure Rotation: $T=0$
\[
\begin{align*}
\dot{u} &= \frac{x-T_x f}{Z} - \omega_y \frac{y^2 f}{f} - \omega_z u^2 \frac{f}{f} \\
\dot{v} &= \frac{y-T_y f}{Z} \omega_x - \omega_z v^2 \frac{f}{f} - \omega_y u \frac{f}{f} \\
\end{align*}
\]
- Independent of $T_x, T_y, T_z$
- Independent of $Z$
- Only function of $(u,v), f$ and $\omega$

Rotational MOTION FIELD
The “instantaneous” velocity of points in an image.

Pure Rotation
$\omega = (0,0,1)^T$

Motion Field Equation: Estimate Depth
\[
\begin{align*}
\dot{u} &= \frac{T_x u - T_x f}{Z} - \omega_y \frac{f}{f} v - \omega_z u^2 \frac{f}{f} \\
\dot{v} &= \frac{T_y v - T_y f}{Z} \omega_x - \omega_z v^2 \frac{f}{f} - \omega_y u \frac{f}{f} \\
\end{align*}
\]
If $T$, $\omega$, and $f$ are known or measured, then for each image point $(u,v)$, one can solve for the depth $Z$ given measured motion $(du/dt, dv/dt)$ at $(u,v)$.

Optical Flow
Where do pixels move to?
Estimating the motion field from images

1. Feature-based (Sect. 8.4.2 of Trucco & Verri)
   1. Detect Features (corners) in an image
   2. Search for the same features nearby (Feature tracking).

2. Differential techniques (Sect. 8.4.1)

Problem Definition: Optical Flow

- How to estimate pixel motion from image \( H \) to image \( I \)?
  - Find pixel correspondences
    - Given a pixel in \( H \), look for nearby pixels of the same color in \( I \)
  - Key assumptions
    - color constancy: a point in \( H \) looks “the same” in image \( I \)
    - For grayscale images, this is brightness constancy
    - small motion: points do not move very far

Definition of optical flow

**OPTICAL FLOW** = apparent motion of brightness patterns

Ideally, the optical flow is the projection of the three-dimensional velocity vectors on the image

Optical Flow Constraint Equation

\[
\begin{align*}
\frac{dx}{dt} + \frac{\partial I}{\partial x} + \frac{dy}{dt} + \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} &= 0 \\
\frac{dx}{dt} + \frac{dy}{dt} + \frac{dz}{dt} &= 0
\end{align*}
\]
Mathematical formulation

[Note change of notation: image coordinates now (x,y), not (u,v)]

\[ I(x, y, t) = \text{brightness at image point } (x, y) \text{ at time } t \]

Consider scene (or camera) to be moving, so \(x(t), y(t)\)

Brightness constancy assumption:

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

Optical flow constraint equation:

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

Solving for flow

Optical flow constraint equation:

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

- We can measure \( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t} \)
- We want to solve for \( \frac{dx}{dt}, \frac{dy}{dt} \)
- One equation, two unknowns

Aperture Problem and Normal Flow

Measurements

\[ I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y}, \quad I_t = \frac{\partial I}{\partial t} \]

Flow vector

\[ u = \frac{dx}{dt}, \quad v = \frac{dy}{dt} \]

Normal Flow

The component of the optical flow in the direction of the image gradient.

Normal Flow

Illusion Works Barber Pole Illusion

Apparently an aperture problem
What is the correspondence of \( P \) & \( P' \)

Contour plots of image intensity in two images

Two ways to get flow

1. Think globally, and regularize over image
2. Look over window and assume constant motion in the window

Lucas-Kanade: Integrate over a Patch

Assume a single velocity for all pixels within an image patch: 
\[
E(u, v) = \sum_{i,j} (I_i(x, y)u + I_j(x, y)v + I_{i,j})
\]

1. \[
\frac{dE(u, v)}{du} = \sum_{i,j} 2I_i(x, y)u = 0
\]
2. \[
\frac{dE(u, v)}{dv} = \sum_{i,j} 2I_j(x, y)v = 0
\]

Solve with:
\[
\begin{pmatrix}
\sum I_i^2 & \sum I_i I_j \\
\sum I_i I_j & \sum I_j^2
\end{pmatrix}
\begin{pmatrix}
u \\
v
\end{pmatrix} = \begin{pmatrix}
-\sum I_i I_{i,j} \\
-\sum I_j I_{i,j}
\end{pmatrix}
\]

On the LHS: sum of the \( 2 \times 2 \) outer product tensor of the gradient vector
\[
(\sum \nabla I \nabla I^T)v = -\sum \nabla I I_{i,j}
\]

Lucas-Kanade: Singularities and the Aperture Problem

Let \( M = \sum (\nabla I \nabla I)^T \) and \( b = \frac{-\sum I_i I_{i,j}}{\sum I_j I_{i,j}} \)

- 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
  - i.e., only normal flow is available (aperture problem)
- Corners and textured areas are OK

Edge

- large gradients, all the same
- large \( \lambda_2 \), small \( \lambda_3 \)

Low texture region

- gradients have small magnitude
- small \( \lambda_1 \), small \( \lambda_2 \)
High textured region

\[ \sum \nabla I(\nabla I)^T \]
- gradients are different, large magnitudes
- large \( \lambda_1 \), large \( \lambda_2 \)

Some variants
- Iterative refinement
- Coarse to fine (image pyramids)
- Local/global motion models
- Robust estimation

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

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?

Limits of the (local) gradient method
1. Fails when intensity structure within window is poor
2. Fails when the displacement is large (typical operating range is motion of 1 pixel per iteration!)
   - 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

Pyramid / “Coarse-to-fine”
Coarse-to-fine optical flow estimation

Multi-resolution Lucas Kanade Algorithm
- Compute ‘simple’ LK at highest level
- At level i:
  - Take flow $u_i, v_i$ from level i-1
  - Bilinear interpolate 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. $u_i = u_i^* + u_i^*, v_i = v_i^* + v_i^*$.

Motion Model Example: Affine Motion
Affine: $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, $h = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$

Parametric (Global) Motion Models
2D Models:
- Translation
- Affine
- Quadratic
- Planar projective transform (Homography)
3D Models:
- Instantaneous camera motion models
- Homography-epipole
- Plane+Parallax

Robust Estimation
Quadratic $\theta$ function gives too much weight to outliers.

$$\rho(r, \sigma) = \frac{r^2}{\sigma^2 + r^2}$$
$$\psi(r, \sigma) = \frac{2r\sigma^2}{(\sigma^2 + r^2)^2}$$
Next Lecture

• Tracking
• Reading:
  – Chapter 11: Tracking