Motion

Computer Vision I
CS252A
Lecture 18

The Motion Field

Announcements

• HW3 due today
• HW4 to be assigned shortly, due next Friday. Easier than HW1-3.

Motion Field Equation

\[
\begin{align*}
\dot{u} &= \frac{T_{xu} - T_{xf}}{Z} - \omega_y f + \omega_y v + \frac{\omega_{xv}}{f} - \frac{\omega_x u^2}{f} \\
\dot{v} &= \frac{T_{yu} - T_{yf}}{Z} + \omega_x f - \omega_x u - \frac{\omega_{yv}}{f} - \frac{\omega_y v^2}{f}
\end{align*}
\]

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

Rigid Motion: General Case

- Position and orientation of a rigid body
- Rotation Matrix & Translation vector

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z}
\end{bmatrix} =
\begin{bmatrix}
R & T
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]

- Rotation Matrix: \( R \)
- Translation vector: \( T \)

- Velocity Vector: \( \omega \) (or \( \Omega \))

- \( \dot{p} = T + \omega \times p \)

Motion Field

Rigid Motion: General Case

- Position and orientation of a rigid body
- Rotation Matrix & Translation vector

- Velocity Vector: \( T \)
- Angular Velocity Vector: \( \omega \) (or \( \Omega \))

- \( \dot{p} = T + \omega \times p \)
Pure Translation

\[ \omega = 0 \]

\[
\begin{align*}
\dot{u} &= \frac{T_x f - T_y f}{Z} - \omega_x f + \omega_y v + \frac{\omega_{uv}}{f} + \frac{\omega_y v^2}{f} \\
\dot{v} &= \frac{T_y f - T_x f}{Z} + \omega_x f - \omega_y u - \frac{\omega_{uv}}{f} - \frac{\omega_y v^2}{f}
\end{align*}
\]

Pure Rotation: \( T=0 \)

\[
\begin{align*}
\dot{u} &= \frac{T_x f - T_y f}{Z} - \omega_x f + \omega_y v + \frac{\omega_{uv}}{f} - \frac{\omega_y u^2}{f} \\
\dot{v} &= \frac{T_y f - T_x f}{Z} + \omega_x f - \omega_y u - \frac{\omega_{uv}}{f} - \frac{\omega_x u^2}{f}
\end{align*}
\]

- Independent of \( T_x, T_y, T_z \)
- Independent of \( Z \)
- Only function of \((u,v), f\) and \( \omega \)

Motion Field Equation: Estimate Depth

\[
\begin{align*}
\dot{u} &= \frac{T_x f - T_y f}{Z} - \omega_x f + \omega_y v + \frac{\omega_{uv}}{f} - \frac{\omega_y u^2}{f} \\
\dot{v} &= \frac{T_y f - T_x f}{Z} + \omega_x f - \omega_y u - \frac{\omega_{uv}}{f} - \frac{\omega_x u^2}{f} \\
Z &= \frac{T_x f - T_y f}{\dot{u} + \omega_x f - \omega_y v - \frac{\omega_{uv}}{f} - \frac{\omega_y u^2}{f}}
\end{align*}
\]

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 \((\text{du/}dt, \text{dv/}dt)\) at \((u,v)\).

Rotational MOTION FIELD

The “instantaneous” velocity of points in an image

\[
\omega = (0,0,1)^T
\]

Optical Flow \( \neq \) Motion Field

Motion field exists but no optical flow

No motion field but shading changes
Definition of optical flow

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

Ideally, the optical flow is the motion field, i.e., the projection of the three-dimensional velocity vectors on the image.

---

**Optical Flow Constraint Equation**

\[ (x, y) \]

\[ (x + u \delta t, y + v \delta t) \]

Optical Flow: Velocities \( (u, v) \)

Displacement:

\[ (\delta x, \delta y) = (u \delta t, v \delta t) \]

- Subtracting \( I(x,y,t) \) from both sides and dividing by \( \delta t \):
  \[ \frac{\delta x}{\delta t} \frac{\partial I}{\partial x} + \frac{\delta y}{\delta t} \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0 \]

- Assume small interval, this becomes:
  \[ \frac{dx}{dt} \frac{\partial I}{\partial x} + \frac{dy}{dt} \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0 \]

---

**Optical Flow Constraint Equation**

\[ (x + u \delta t, y + v \delta t) \]

Optical Flow: Velocities \( (u, v) \)

Displacement:

\[ (\delta x, \delta y) = (u \delta t, v \delta t) \]

- Assume brightness of patch remains same in both images:
  \[ I(x + u \delta t, y + v \delta t, t + \delta t) = I(x, y, t) \]

- Assume small motion: (Taylor expansion of LHS up to first order)
  \[ I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} = I(x, y, t) \]

---

**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 \( \frac{dx}{dt}, \frac{dy}{dt} \)
- We want to solve for \( \frac{dx}{dt}, \frac{dy}{dt} \)
- One equation, two unknowns

---

**Aperture Problem and Normal Flow**

Measurements

\[ I_x = \frac{\partial I}{\partial x} \]

\[ I_y = \frac{\partial I}{\partial y} \]

\[ I_t = \frac{\partial I}{\partial t} \]

Flow vector

\[ u = \frac{dx}{dt} \]

\[ v = \frac{dy}{dt} \]

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

---

**Optical Flow Constraint**

Barber pole illusion

[Diagram showing Barber's pole with a motion field and optical flow vectors]
Normal Flow

Illusion Works Barber Pole Illusion

What is the correspondence of P & P’

Contour plots of image intensity in two images

Horn & Schunck algorithm

Additional smoothness constraint:

\[ e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2)) \, dx \, dy, \]

besides optical flow constraint equation term

\[ e_c = \iint (I_{xx}(u + I_{xy}v + I_{xt})) \, dx \, dy. \]

minimize \( e_s + \lambda e_c. \)

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

\[ E(u,v) = \sum_{x,y \in I} (I_x(x,y)u + I_y(x,y)v + I_t)^2 \]

\[
\frac{dE(u,v)}{du} = 2 \sum_{x,y \in I} I_x I_y u + I_x I_t + I_y I_t = 0
\]

\[
\frac{dE(u,v)}{dv} = 2 \sum_{x,y \in I} I_x I_y v + I_x I_t + I_y I_t = 0
\]

Solve with:

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

On the LHS: sum of the 2x2 outer product tensor of the gradient vector

\[
(\nabla I \nabla I^T) \hat{u} = -\nabla I I_t
\]
Lucas-Kanade: Singularities and the Aperture Problem

Let \( M = \sum (V_i)^T V_i \) and \( b = -\sum i_i j_i \)

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

- \( M \) 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

Low texture region

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

- 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?

Pyramid / “Coarse-to-fine”

Coarse-to-Fine Estimation

\[ I_x \cdot u + I_y \cdot v + I_z = 0 \]  
\( u, v \) small

image I

image H

Gaussian pyramid of image I

Gaussian pyramid of image H

Multi-resolution Lucas Kanade Algorithm

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

Problem: Least-squares estimators penalize deviations between data & model with quadratic error $P$ (extremely sensitive to outliers)

- Error penalty function: $\rho(c) = c^2$
- Influence function: $\psi(c) = \frac{\partial \rho(c)}{\partial c} = 2c$

Redescending error functions (e.g., Geman-McClure) help to reduce the influence of outlying measurements.

- Error penalty function: $\rho(c; s) = \frac{c^2}{s + c^2}$
- Influence function: $\psi(c; s) = \frac{2c}{(s + c^2)^2}$