Announcements

- Homework 4 is due June 5, 11:59 PM
- Final exam will be a take home exam
- TA Evaluations

Reading:
- Section 7.2 Two-frame structure from motion
- Section 8.1 Translational alignment
- Section 8.2 Parametric motion
- Section 8.4 Optical flow

Motion

Structure from Motion

Some problems of motion
1. Correspondence: Where have elements of the image moved between image frames
2. Reconstruction: Given correspondence, what is 3-D geometry of scene
3. Segmentation: What are regions of image corresponding to different moving objects
4. Tracking: Where have objects moved in the image?

Variations:
- Small motion (video)
- Wide-baseline (multi-view)

Motion

Structure-from-Motion (SFM)

Goal: Take as input two or more images or video without knowledge of the camera position/motion, and estimate the camera position and 3D structure of scene.

Two Approaches
1. Discrete motion (wide baseline)
   1. Orthographic (affine) vs. Perspective
   2. Two view vs. Multi-view
   3. Calibrated vs. Uncalibrated
2. Continuous (Infinitesimal) motion

Some problems of motion
1. Correspondence: Where have elements of the image moved between image frames
2. Reconstruction: Given correspondence, what is 3-D geometry of scene
3. Segmentation: What are regions of image corresponding to different moving objects
4. Tracking: Where have objects moved in the image?

Variations:
- Small motion (video)
- Wide-baseline (multi-view)
Discrete Motion: Some Counting

Consider \( M \) images of \( N \) points, how many unknowns?

1. Affix coordinate system to location of first camera location: \((M-1)*6\) Unkowns
2. 3-D Structure: \(3*N\) Unkowns
3. Can only recover structure and motion up to scale. Why?

Total number of unknowns: \((M-1)*6+3*N-1\)
Total number of measurements: \(2*M*N\)
Solution is possible when \((M-1)*6+3*N-1 \leq 2*M*N\)

The Eight-Point Algorithm (Longuet-Higgins, 1981)

\[
p'Ep' = 0 \quad \text{with} \quad E = [t]_R.
\]

\[
\begin{bmatrix}
  E_{11} & E_{12} & E_{13} & E_{14} & E_{15} & E_{16} \\
  E_{21} & E_{22} & E_{23} & E_{24} & E_{25} & E_{26} \\
  E_{31} & E_{32} & E_{33} & E_{34} & E_{35} & E_{36} \\
  E_{41} & E_{42} & E_{43} & E_{44} & E_{45} & E_{46} \\
  E_{51} & E_{52} & E_{53} & E_{54} & E_{55} & E_{56} \\
  E_{61} & E_{62} & E_{63} & E_{64} & E_{65} & E_{66}
\end{bmatrix}\begin{bmatrix}
u' \\
v' \\
v' \\
v' \\
v' \\
v'
\end{bmatrix} = 0
\]

\[
\begin{bmatrix}
u' & 0 & 0 & 0 & 0 & 0 \\
0 & v' & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
u' \\
v' \\
v' \\
v' \\
v' \\
v'
\end{bmatrix} = 0
\]

• Set \(E_{33}\) to 1
• Use 8 points \((u,v)_i\), \(i=1..8\)

\[
\begin{bmatrix}
u'_1 & \cdots & \nu'_8 \\
v'_1 & \cdots & v'_8 \\
v'_1 & \cdots & v'_8 \\
v'_1 & \cdots & v'_8 \\
v'_1 & \cdots & v'_8 \\
v'_1 & \cdots & v'_8
\end{bmatrix}\begin{bmatrix}
E_{11} \\
E_{21} \\
E_{31} \\
E_{41} \\
E_{51} \\
E_{61}
\end{bmatrix} = 0
\]

Solve For \(E\) 
Solve For \(R\) and \(t\)

Sketch of Two View SFM Algorithm

Input: Two images
1. Detect feature points
2. Find 8 matching feature points (easier said than done)
3. Compute the Essential Matrix \(E\) using Normalized 8-point Algorithm
4. Compute \(R\) and \(T\) (recall that \(E=RS\) where \(S\) is skew symmetric matrix)
5. Perform stereo matching using recovered epipolar geometry expressed via \(E\).
6. Reconstruct 3-D geometry of corresponding points.

Feature matching
Evaluate normalized cross correlation (or sum of squared differences) for all features with similar coordinates
e.g. \((x', y')\in[x-x_{\text{min}}, x+x_{\text{max}}][y-y_{\text{min}}, y+y_{\text{max}}]\)

Keep mutual best matches
Still many wrong matches!
Comments

• **Greedy Algorithm:**
  – Given feature in one image, find best match in second image irrespective of other matches.
  – OK for small motions, little rotation, small search window
• **Otherwise**
  – Must compare descriptor over rotation
  – Can’t consider $O(n^2)$ potential pairings (way too many), so
    • Manual correspondence (e.g., façade, photogrammetry).
    • Use random sampling (RANSAC).
    • More descriptive features (line segments, SIFT, larger regions, color).
    • Use video sequence to track, but perform SFM w/ first and last image.

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

Background Subtraction

• Gather image $I(x,y,t_0)$ of background without objects of interest (perhaps computed over average over many images).
• At time $t$, pixels where $|I(x,y,t) - I(x,y,t_0)| > \tau$ are labeled as coming from foreground objects

Simplest Idea for video processing

Image Differences

• Given image $I(u,v,t)$ and $I(u,v, t+\delta t)$, compute $I(u,v, t+\delta t) - I(u,v,t)$.
• 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
• Doesn’t tell which way stuff is moving

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. Object’s 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
Position and orientation of a rigid body
Rotation Matrix & Translation vector
Rigid Motion:
Velocity Vector: $T$
Angular Velocity Vector: $\omega$ (or $\Omega$)

\[ \dot{p} = T + \omega \times p \]

General Motion
\[
\begin{bmatrix}
\dot{u} \\
\dot{v} \\
\dot{w}
\end{bmatrix} = \begin{bmatrix}
f_x \\
f_y \\
f_z
\end{bmatrix} + \frac{\dot{\omega}}{\rho} \begin{bmatrix}
\hat{z} \\
\hat{x} \\
\hat{y}
\end{bmatrix} = \begin{bmatrix}
f_x \\
f_y \\
f_z
\end{bmatrix} + \frac{\dot{\omega}}{\rho} \begin{bmatrix}
\dot{y} \\
\dot{x} \\
\dot{z}
\end{bmatrix}
\]
Substitute $\dot{p} = T + \omega \times p$ where $p=(x,y,z)^T$
**Motion Field Equation**

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

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

**Pure Translation**

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

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

**Pure Rotation: T=0**

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

**Rotational MOTION FIELD**

The “instantaneous” velocity of points in an image

**Pure Rotation**

\[\omega = (0,0,1)^T\]
Motion Field Equation: Estimate Depth

\[
\begin{align*}
\dot{u} &= \frac{T_u - T_f f}{Z} - \omega_x f + \omega_y y - \frac{\omega_y u}{f} - \frac{\omega_x u^2}{f} \\
\dot{v} &= \frac{T_v - T_f f}{Z} + \omega_x f - \omega_y y - \frac{\omega_y v}{f} - \frac{\omega_x v^2}{f}
\end{align*}
\]

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

Optical Flow

Optical Flow: Where do pixels move to?

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

1. Assume brightness of patch remains same in both images:
2. Assume small motion: (Taylor expansion of LHS up to first order)

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

Optical Flow: Velocities \((u, v)\)
Displacement: \((\delta x, \delta y) = (u \delta t, v \delta t)\)

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

Optical Flow Constraint

Barber pole illusion

\[ \text{Barber's pole} \quad \text{Motion field} \quad \text{Optical flow} \]
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

\[ E(u, v) = \sum_{i} \left( I_{x}(x, y)u + I_{y}(x, y)v + I_{0} \right) \]

\[ \frac{dE(u, v)}{du} = \sum 2I_{x}(x, y)u + I_{0} = 0 \]

\[ \frac{dE(u, v)}{dv} = \sum 2I_{y}(x, y)v + I_{0} = 0 \]

Solve with:

\[ \begin{bmatrix} \sum I_{x}^{2} & \sum I_{x}I_{y} \n \sum I_{x}I_{y} & \sum I_{y}^{2} \end{bmatrix} \begin{bmatrix} u \\
v \end{bmatrix} = \begin{bmatrix} -\sum I_{x}I_{0} \\
-\sum I_{y}I_{0} \end{bmatrix} \]

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

\[ \sum I(vI)^{T} = -\sum \nabla I_{0} \]

Lucas-Kanade: Singularities and the Aperture Problem

Let \( M = \sum (vI)^{T} \) and \( b = \left[ -\sum I_{0} \right] \)

- 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

\[ \sum \nabla I(vI)^{T} \]

- large gradients, all the same
  - large \( \lambda_{1} \), small \( \lambda_{2} \)

Low texture region

\[ \sum \nabla I(vI)^{T} \]

- 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?

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

\[ I_x \cdot u + I_y \cdot v + I_z = 0 \]

\Rightarrow \text{small } u \text{ and } v \ldots

Parametric (Global) Motion Models

2D Models:
- Translation
- Affine
- Quadratic
- Planar projective transform (Homography)

3D Models:
- Instantaneous camera motion models
- Homography+epipole
- Plane+parallax

Motion Model Example: Affine Motion

\[
\Lambda = \begin{bmatrix} a & b \\ c & d \\ \end{bmatrix}, \quad h = \begin{bmatrix} \delta x \\ \delta y \\ \end{bmatrix}
\]