

Motion

Computer Vision I

CSE252A

Lecture 17

CS252A, Winter 2006 Computer Vision I

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? related to correspondence and segmentation.


Variations:

- Small motion (video),
- Wide-baseline (multi-view)

CS252A, Winter 2006 Computer Vision I

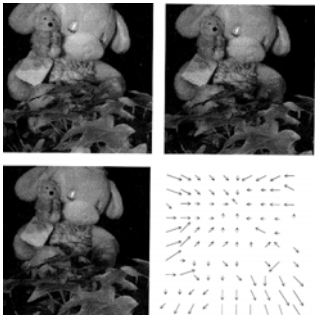
Is motion estimation inherent in humans?

Demo



CS252A, Winter 2006 Computer Vision I

The Motion Field



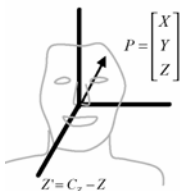
CS252A, Winter 2006 Computer Vision I

What causes a motion field?

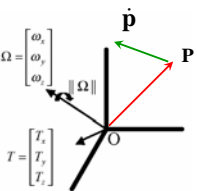
1. Camera moves (translates, rotates)
2. Object's in scene move rigidly
3. Objects articulate (pliers, humans, animals)
4. Objects bend and deform (fish)
5. Blowing smoke, clouds

CS252A, Winter 2006 Computer Vision I

Rigid Motion: General Case



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



Rigid Motion:
Velocity Vector: T
Angular Velocity Vector: ω (or Ω)

$$\dot{\mathbf{p}} = \mathbf{T} + \boldsymbol{\omega} \times \mathbf{p}$$

CS252A, Winter 2006 Computer Vision I

General Motion

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

$$\begin{aligned} \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} &= \frac{f}{z} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} - \frac{f\dot{z}}{z^2} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \frac{f}{z} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} - \frac{f\dot{z}}{z} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

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

CS252A, Winter 2006

Computer Vision I

Motion Field Equation

$$\dot{u} = \frac{T_z u - T_x f}{Z} - \omega_y f + \omega_z v + \frac{\omega_x u v}{f} - \frac{\omega_y u^2}{f}$$

$$\dot{v} = \frac{T_z v - T_y f}{Z} + \omega_x f - \omega_z u - \frac{\omega_y u v}{f} - \frac{\omega_x v^2}{f}$$

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

CS252A, Winter 2006

Computer Vision I

Pure Translation

~~$$\begin{aligned} \dot{u} &= \frac{T_z u - T_x f}{Z} - \omega_y f + \omega_z v + \frac{\omega_x u v}{f} - \frac{\omega_y u^2}{f} \\ \dot{v} &= \frac{T_z v - T_y f}{Z} + \omega_x f - \omega_z u - \frac{\omega_y u v}{f} - \frac{\omega_x v^2}{f} \end{aligned}$$~~

$$\boldsymbol{\omega} = 0$$

CS252A, Winter 2006

Computer Vision I

THE MOTION FIELD

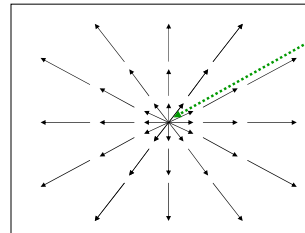
The "instantaneous" velocity of points in an image

LOOMING

The Focus of Expansion (FOE)

$$du/dt = dv/dt = 0$$

Intersection of velocity vector with image plane



With just this information it is possible to calculate:

1. Direction of motion
2. Time to collision

CS252A, Winter 2006

u 1

Pure Rotation: $\mathbf{T}=0$

~~$$\begin{aligned} \dot{u} &= \frac{T_z u - T_x f}{Z} - \omega_y f + \omega_z v + \frac{\omega_x u v}{f} - \frac{\omega_y u^2}{f} \\ \dot{v} &= \frac{T_z v - T_y f}{Z} + \omega_x f - \omega_z u - \frac{\omega_y u v}{f} - \frac{\omega_x v^2}{f} \end{aligned}$$~~

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

CS252A, Winter 2006

Computer Vision I

Rotational MOTION FIELD

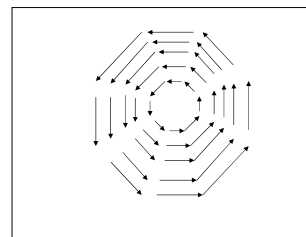
The "instantaneous" velocity of points in an image

PURE ROTATION

$$\boldsymbol{\omega} = (0,0,1)^T$$

$$\dot{u} = +\omega_z v$$

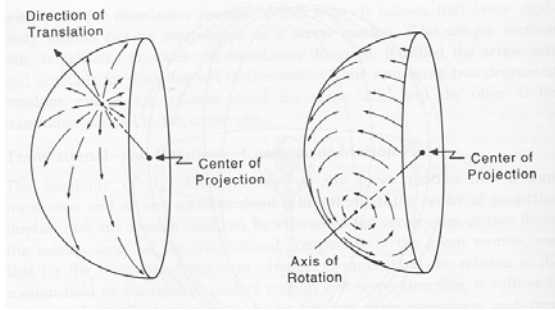
$$\dot{v} = -\omega_z u$$



CS252A, Winter 2006

Computer Vision I

Pure Rotation: Motion Field on Sphere



CS252A, Winter 2006

Computer Vision I

Mathematical formulation

$I(x, y, t)$ = brightness at image point (x, y) at time t

Consider scene (or camera) to be moving, so (x, y) is a function of time (i.e., $x(t), y(t)$), and point is moving with velocity $(dx/dt, dy/dt)$

Brightness constancy assumption:

$$I\left(x + \frac{dx}{dt} \delta t, y + \frac{dy}{dt} \delta t, t + \delta t\right) = I(x, y, t) \quad \rightarrow \quad \frac{dI}{dt} = 0$$

Optical flow constraint equation:

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

CS252A, Winter 2006

Computer Vision I

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

CS252A, Winter 2006

Computer Vision I

Aperture Problem and Normal Flow

Measurements

$$I_x = \frac{\partial I}{\partial x}$$

$$I_y = \frac{\partial I}{\partial y}$$

$$I_t = \frac{\partial I}{\partial t}$$

The gradient constraint:

$$I_x u + I_y v + I_t = 0$$

$$\nabla I \cdot \vec{U} = 0$$

Defines a line in the (u, v) space

Flow vector

$$u = \frac{dx}{dt}$$

$$v = \frac{dy}{dt}$$

Normal Flow:

$$u_{\perp} = -\frac{I_y \nabla I_x}{|\nabla I| |\nabla I|}$$

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

CS252A, Winter 2006

Computer Vision I

Normal Flow

Illusion Works Barber Pole Illusion



CS252A, Winter 2006

Computer Vision I

Lucas-Kanade: Integrate over a Patch

Assume a single velocity for all pixels within an image patch

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

$$\frac{dE(u, v)}{du} = \sum 2I_x(I_x u + I_y v + I_t) = 0$$

$$\frac{dE(u, v)}{dv} = \sum 2I_y(I_x u + I_y v + I_t) = 0$$

Solve with:

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

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

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

CS252A, Winter 2006

Computer Vision I

Lucas-Kanade: Singularities and the Aperture Problem

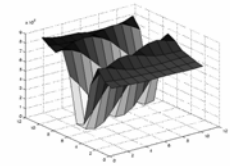
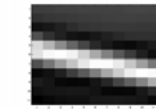
$$\text{Let } M = \sum (\nabla I)(\nabla I)^T \quad \text{and} \quad b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

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

CS252A, Winter 2006

Computer Vision I

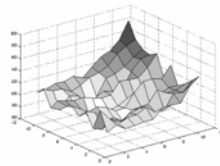
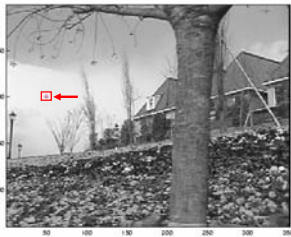
Edge



$$\sum \nabla I(\nabla I)^T$$

- large gradients, all the same
- large λ_1 , small λ_2

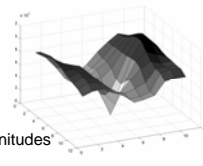
Low texture region



$$\sum \nabla I(\nabla I)^T$$

- gradients have small magnitude
- small λ_1 , small λ_2

High textured region



$$\sum \nabla I(\nabla I)^T$$

- gradients are different, large magnitudes
- large λ_1 , large λ_2

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

CS252A, Winter 2006

Computer Vision I

Some variants

- Coarse to fine (image pyramids)
- Local/global motion models
- Robust estimation

CS252A, Winter 2006

Computer Vision I

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

CS252A, Winter 2006 Computer Vision I

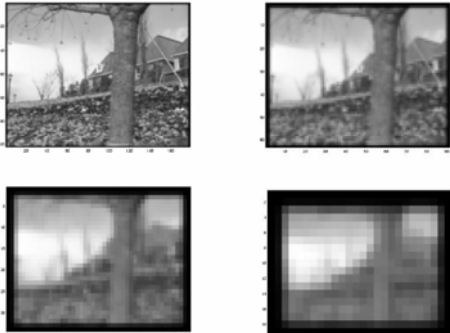
Revisiting the small motion assumption



- Is this motion small enough?
 - Probably not—it's much larger than one pixel (2nd order terms dominate)

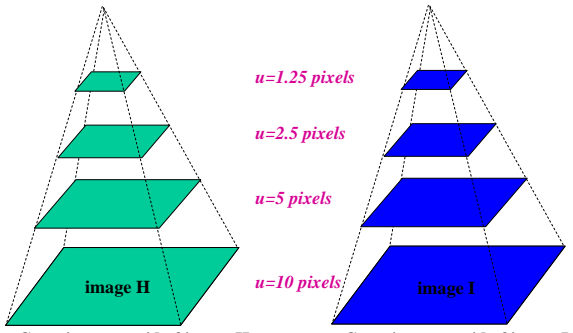
CS252A, Winter 2006 Computer Vision I

Pyramid / "Coarse-to-fine"



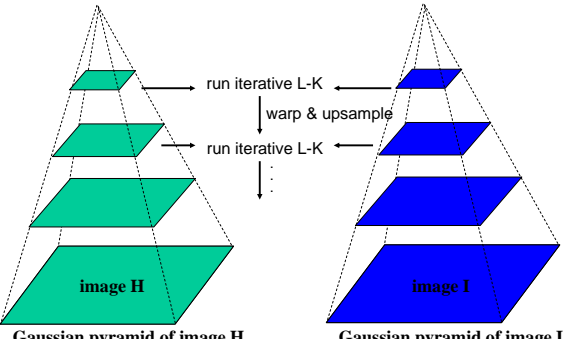
CS252A, Winter 20 Vision I

Coarse-to-fine optical flow estimation



CS252A, Winter 2006 Computer Vision I

Coarse-to-fine optical flow estimation



CS252A, Winter 2006 Computer Vision I

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. $u_i = u_i^* + u_i', v_i = v_i^* + v_i'$

CS252A, Winter 2006 Computer Vision I

Parametric (Global) Motion Models

2D Models:

(Translation)

Affine

Quadratic

Planar projective transform (Homography)

3D Models:

Instantaneous camera motion models

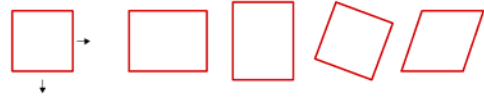
Homography+epipole

Plane+Parallax

CS252A, Winter 2006

Computer Vision I

Motion Model Example: Affine Motion



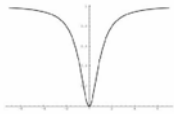
Affine: $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\mathbf{h} = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$

CS252A, Winter 2006

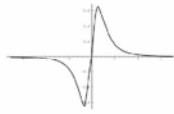
Computer Vision I

Robust Estimation

Quadratic θ function gives too much weight to outliers.



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



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

CS252A, Winter 2006

Computer Vision I