Lecture 20: Tracking

Tuesday, Nov 27

Paper reviews

- Thorough summary in your own words
- Main contribution
- Strengths? Weaknesses?
- How convincing are the experiments?
- Suggestions to improve them?
- Extensions?
- 4 pages max

May require reading additional references

(This is list from 8/30/07 lecture)

What to submit for the extension

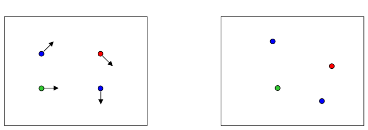
Include:
- Goal of the extension
- Summarize implementation strategy
- Analyze outcomes
- Show figures as necessary

For both, submit as hardcopy, due by the end of the day on 12/6/07.

Outline

- Last time: Motion
  - Motion field and parallax
  - Optical flow, brightness constancy
  - Aperture problem
- Today: Warping and tracking
  - Image warping for iterative flow
  - Feature tracking (vs. differential)
  - Linear models of dynamics
  - Kalman filters

Last time: Optical flow problem



$H(x, y)$ $I(x, y)$

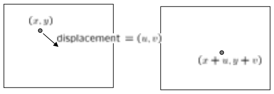
How to estimate pixel motion from image H to image I?
- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I

Adapted from Steve Seitz, UW

Last time: Motion constraints

- To recover optical flow, we need some constraints (assumptions)
  - *Brightness constancy*: in spite of motion, image measurement in small region will remain the same
  - *Spatial coherence*: assume nearby points belong to the same surface, thus have similar motions, so estimated motion should vary smoothly.
  - *Temporal smoothness*: motion of a surface patch changes gradually over time.

Last time: Brightness constancy equation



$$\frac{dI}{dt} = 0$$ Total derivative: *x* and *y* are also functions of time *t*

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

spatial gradients: **how image varies in *x* or *y* direction for fixed time**

***temporal gradient*: how image varies in time for fixed position**

temporal derivatives*, *u* and *v*: rate of change in *x* and *y

Rewritten: $I_x u + I_y v + I_t = 0.$

$$\nabla I^T \mathbf{u} + I_t = 0.$$

Last time: Aperture problem

$$\nabla I^T \mathbf{u} + I_t = 0.$$

- Brightness constancy equation: single equation, two unknowns; infinitely many solutions.
- Can only compute projection of actual flow vector [*u*,*v*] in the direction of the image gradient, that is, in the direction *normal* to the image edge.
  - Flow component in gradient direction determined
  - Flow component parallel to edge unknown.

Last time: Solving the aperture problem

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel's neighbors have the same (u,v)
    - » If we use a 5x5 window, that gives us 25 equations per pixel!

$$\nabla I^T \mathbf{u} + I_t = 0.$$

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

$\underset{25\times2}{A}$ $\underset{2\times1}{d}$ $\underset{25\times1}{b}$

Adapted from Steve Seitz, UW

Last time: Lucas-Kanade flow

Prob: we have more equations than unknowns

$\underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b}$ ⟶ minimize $\|Ad - b\|^2$

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

$$\underset{2\times2}{(A^T A)}\ \underset{2\times1}{d} = \underset{2\times1}{A^T b}$$

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

$A^T A$ $A^T b$

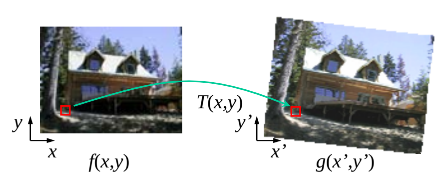
- The summations are over all pixels in the K x K window
- This technique was first proposed by Lucas & Kanade (1981)

Slide by Steve Seitz, UW

Difficulties

- When will this flow computation fail?
  - If brightness constancy is not satisfied
    - E.g., occlusions, illumination change…
  - If the motion is not small
    - derivative estimates poor
  - If points within window neighborhood do not move together
    - E.g., if window size is too large

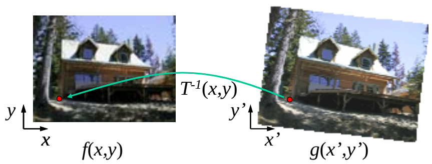
Image warping



Given a coordinate transform and a source image *f*(*x*,*y*), how do we compute a transformed image *g*(*x'*,*y'*) = *f*(*T*(*x*,*y*))?

Slide from Alyosha Efros, CMU

Inverse warping

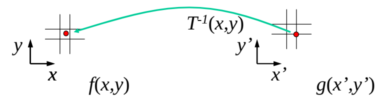


Get each pixel $g(x',y')$ from its corresponding location
$(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

Slide from Alyosha Efros, CMU

Inverse warping



Get each pixel $g(x',y')$ from its corresponding location
$(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

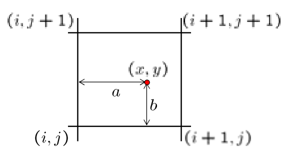
A: *Interpolate* color value from neighbors

– nearest neighbor, bilinear…

Slide from Alyosha Efros, CMU

Bilinear interpolation

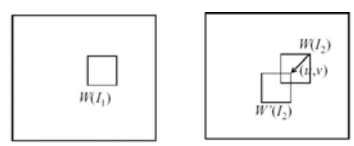
Sampling at *f(x,y):*



$$
\begin{aligned}
f(x,y) = \; & (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

Slide from Alyosha Efros, CMU

Iterative flow computation



To iteratively refine flow estimates, repeat until warped version of first image very close to second image:

- compute flow vector [u, v]
- warp image toward the other using estimated flow field

Figure from Martial Hebert, CMU

Feature Detection

Tracking features

Feature tracking

- Compute optical flow for that feature for each consecutive frame pair

When will this go wrong?

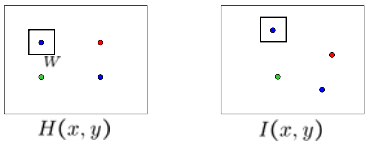
- Occlusions—feature may disappear
  - need mechanism for deleting, adding new features
- Changes in shape, orientation
  - allow the feature to deform
- Changes in color
- Large motions

Adapted from Steve Seitz, UW

Handling large motions

Derivative-based flow computation requires small motion.

- If the motion is much more than a pixel, use discrete **search** instead



- Given feature window W in H, find best matching window in I
- Minimize sum squared difference (SSD) of pixels in window

$$min_{(u,v)} \left\{ \sum_{(x,y)\in W} |I(x+u, y+v) - H(x,y)|^2 \right\}$$

- Solve by doing a search over a specified range of (u,v) values
  – this (u,v) range defines the **search window**

Adapted from Steve Seitz, UW

- For a discrete matching search, what are the tradeoffs of the chosen **search window** size?

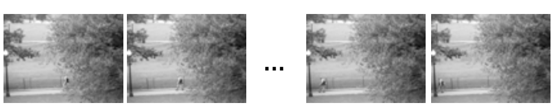
Summary: Motion field estimation

- **Differential techniques**
  – optical flow: use spatial and temporal variation of image brightness at all pixels
  – assumes we can approximate motion field by constant velocity within small region of image plane
- **Feature matching techniques**
  – estimate disparity of special points (easily tracked features) between frames
  – sparse

Think of stereo matching: same as estimating motion if we have two close views or two frames close in time.

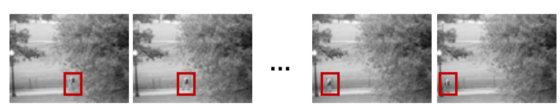
- Tracking with features: where should the search window be placed?
  – Near match at previous frame
  – More generally, according to expected *dynamics* of the object

Detection vs. tracking



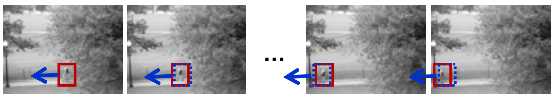
t=1 t=2 t=20 t=21

Detection vs. tracking



Detection: We detect the object independently in each frame and can record its position over time, e.g., based on blob's centroid or detection window coordinates

Detection vs. tracking



Tracking with *dynamics*: We use image measurements to estimate position of object, but also incorporate position predicted by dynamics, i.e., our expectation of object's motion pattern.

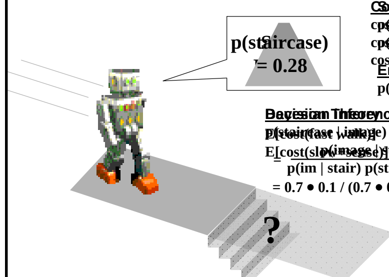
Goal of tracking

- Have a model of expected motion
- Given that, predict where objects will occur in next frame, even before seeing the image
- Intent:
  - do less work looking for the object, restrict search
  - improved estimates since measurement noise tempered by trajectory smoothness

General assumptions

- Expect motion to be continuous, so we can predict based on previous trajectories
  - Camera is not moving instantly from viewpoint to viewpoint
  - Objects do not disappear and reappear in different places in the scene
  - Gradual change in pose between camera and scene
- Able to model the motion

Example of Bayesian Inference



cost(slow+sense) = $1

Environment prior

p(staircase) = 0.1

= $1,000 • 0.28 = $280

p(im | stair) p(stair) + p(im | no stair) p(no stair)

= 0.7 • 0.1 / (0.7 • 0.1 + 0.2 • 0.9) = 0.28

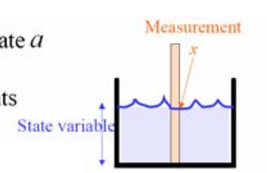
Slide by Sebastian Thrun and Jana Košecká, Stanford University

Tracking as inference: Bayes Filters

- Hidden state x_t
  - The unknown true parameters
  - E.g., actual position of the person we are tracking
- Measurement y_t
  - Our noisy observation of the state
  - E.g., detected blob's centroid
- Can we calculate $p(x_t \mid y_1, y_2, \ldots, y_t)$?
  - Want to recover the state from the observed measurements

Idea of recursive estimation

- Goal: Find estimate $\hat{a}$ of state a such that the least square error between measurements and the state is minimum

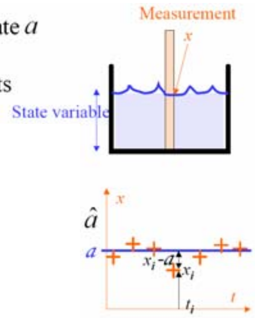


Note temporary change of notation: state is $\mathbf{a}$*, and measurement at time step i is* $\mathbf{x}_i$*.*

Adapted from Cornelia Fermüller, UMD.

Idea of recursive estimation

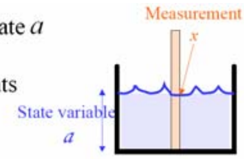
- Goal: Find estimate $\hat{a}$ of state a such that the least square error between measurements and the state is minimum



Adapted from Cornelia Fermüller, UMD.

Idea of recursive estimation

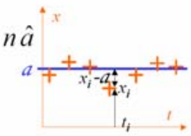
- Goal: Find estimate $\hat{a}$ of state a such that the least square error between measurements and the state is minimum



$$C = \frac{1}{2}\sum_{i=1}^{n}(x_i - a)^2$$

$$\frac{\partial C}{\partial a} = 0 = \sum_{i=1}^{n}(x_i - \hat{a}) = \sum_{i=1}^{n} x_i - n\hat{a}$$

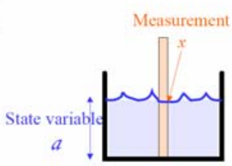
$$\hat{a} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

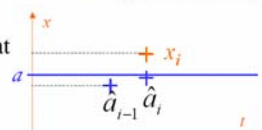


Adapted from Cornelia Fermüller, UMD.

Idea of recursive estimation

- We don't want to wait until all data have been collected to get an estimate $\hat{a}$ of the depth
- We don't want to reprocess old data when we make a new measurement
- Recursive method: data at step i are obtained from data at step $i-1$



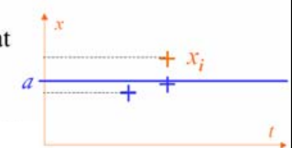


Adapted from Cornelia Fermüller, UMD.

Idea of recursive estimation

- Recursive method: data at step i are obtained from data at step $i-1$

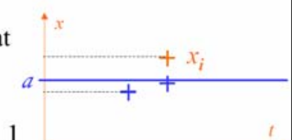
$$\hat{a}_i = \frac{1}{i}\sum_{k=1}^{i} x_k$$



Adapted from Cornelia Fermüller, UMD.

Idea of recursive estimation

- Recursive method: data at step i are obtained from data at step $i-1$

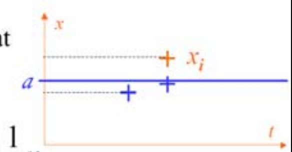


$$\hat{a}_i = \frac{1}{i}\sum_{k=1}^{i} x_k = \frac{1}{i}\sum_{k=1}^{i-1} x_k + \frac{1}{i} x_i$$

Adapted from Cornelia Fermüller, UMD.

Idea of recursive estimation

- Recursive method: data at step i are obtained from data at step $i-1$



$$\hat{a}_i = \frac{1}{i}\sum_{k=1}^{i} x_k = \frac{1}{i}\sum_{k=1}^{i-1} x_k + \frac{1}{i} x_i$$

$$\hat{a}_{i-1} = \frac{1}{i-1}\sum_{k=1}^{i-1} x_k$$

Adapted from Cornelia Fermüller, UMD.

Idea of recursive estimation

- Recursive method: data at step i are obtained from data at step $i-1$

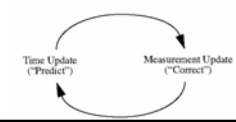
$$\hat{a}_i = \frac{1}{i}\sum_{k=1}^{i} x_k = \frac{1}{i}\sum_{k=1}^{i-1} x_k + \frac{1}{i} x_i \qquad \hat{a}_{i-1} = \frac{1}{i-1}\sum_{k=1}^{i-1} x_k$$

$$\hat{a}_i = \frac{i-1}{i}\hat{a}_{i-1} + \frac{1}{i} x_i$$

Adapted from Cornelia Fermüller, UMD.

Inference for tracking

- Recursive process:
  - Assume we have initial prior that predicts state in absence of any evidence: $P(\mathbf{X}_0)$
  - At the first frame, *correct* this given the value of $\mathbf{Y}_0 = \mathbf{y}_0$
  - Given corrected estimate for frame t
    - Predict for frame t+1
    - Correct for frame t+1



Tracking as inference

- Prediction:
  - Given the measurements we have seen up to this point, what state should we predict?

$$P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1}).$$

- Correction:
  - Now given the current measurement, what state should we predict?

$$P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_i = \boldsymbol{y}_i)$$

Assume independences to simplify

- Only immediate past state influences current state

$$P(\boldsymbol{X}_i | \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}) = P(\boldsymbol{X}_i | \boldsymbol{X}_{i-1})$$

- Measurements at time t only depend on the current state

$$P(\boldsymbol{Y}_i, \boldsymbol{Y}_j, \ldots \boldsymbol{Y}_k | \boldsymbol{X}_i) = P(\boldsymbol{Y}_i | \boldsymbol{X}_i) P(\boldsymbol{Y}_j, \ldots, \boldsymbol{Y}_k | \boldsymbol{X}_i)$$

Base case

$$P(\boldsymbol{X}_0 | \boldsymbol{Y}_0 = \boldsymbol{y}_0) = \frac{P(\boldsymbol{y}_0 | \boldsymbol{X}_0) P(\boldsymbol{X}_0)}{P(\boldsymbol{y}_0)}$$

$$\propto P(\boldsymbol{y}_0 | \boldsymbol{X}_0) P(\boldsymbol{X}_0)$$

Induction step: prediction

Prediction

Prediction involves representing

$$P(\boldsymbol{X}_i | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})$$

given

$$P(\boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}).$$

Our independence assumptions make it possible to write

$$\begin{aligned} P(\boldsymbol{X}_i | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) &= \int P(\boldsymbol{X}_i, \boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) d\boldsymbol{X}_{i-1} \\ &= \int P(\boldsymbol{X}_i | \boldsymbol{X}_{i-1}, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) P(\boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) d\boldsymbol{X}_{i-1} \\ &= \int P(\boldsymbol{X}_i | \boldsymbol{X}_{i-1}) P(\boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) d\boldsymbol{X}_{i-1} \end{aligned}$$

Induction step: correction

Correction

Correction involves obtaining a representation of

$$P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)$$

given

$$P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})$$

Our independence assumptions make it possible to write

$$\begin{aligned} P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i) &= \frac{P(\boldsymbol{X}_i,\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)}{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)} \\ &= \frac{P(\boldsymbol{y}_i|\boldsymbol{X}_i,\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)} \\ &= P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})\frac{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)} \\ &= \frac{P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})}{\int P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})d\boldsymbol{X}_i} \end{aligned}$$

Inference for tracking

- Goal is then to
  - choose good model for the prediction and correction distributions
  - use the updates to compute best estimate of state
    - Prior to seeing measurement
    - After seeing the measurement

- We stopped here on Tuesday, to be continued on Thursday.