Robert Collins
CSE486, Penn State

Lecture 30:

Video Tracking: Lucas-Kanade

Two Popular Tracking Methods

- Mean-shift color histogram tracking (last time)
- Lucas-Kanade template tracking (today)

Lucas-Kanade Tracking

Review: Lucas-Kanade

- Brightness constancy
- One equation two unknowns

unknown flow vector

$$0 = I_t(\mathrm{p_i}) + \nabla I(\mathrm{p_i}) \cdot [u \;\; v]$$

temporal gradient **spatial gradient**

- How to get more equations for a pixel?
  - Basic idea: impose additional constraints
    - 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!

$$\underset{25\times 2}{\underset{A}{\begin{bmatrix} I_x(\mathrm{p_1}) & I_y(\mathrm{p_1}) \\ I_x(\mathrm{p_2}) & I_y(\mathrm{p_2}) \\ \vdots & \vdots \\ I_x(\mathrm{p_{25}}) & I_y(\mathrm{p_{25}}) \end{bmatrix}}} \underset{2\times 1}{\underset{d}{\begin{bmatrix} u \\ v \end{bmatrix}}} = - \underset{25\times 1}{\underset{b}{\begin{bmatrix} I_t(\mathrm{p_1}) \\ I_t(\mathrm{p_2}) \\ \vdots \\ I_t(\mathrm{p_{25}}) \end{bmatrix}}}$$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Robert Collins
CSE486, Penn State

Review: Lucas-Kanade (cont)

- Now we have more equations than unknowns

$$\underset{25\text{x}2}{A}\ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

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

$$\underset{2\text{x}2}{(A^T A)}\ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Robert Collins
CSE486, Penn State

Lucas Kanade Tracking

Traditional Lucas-Kanade is typically run on small, corner-like features (e.g. 5x5) to compute optic flow.

Observation: There's no reason we can't use the same approach on a larger window around the object being tracked.

Basic LK Derivation for Templates

$$E(u,v) = \sum \left[\underline{I(x+u, y+v)} - T(x,y) \right]^2$$

current frame

template (model)

u,v = hypothesized location of template in current frame

Basic LK Derivation for Templates

$$E(u,v) = \sum [I(x+u, y+v) - T(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2$$ First order approx

$$= \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]^2$$

Take partial derivs and set to zero

$$\frac{\delta E}{du} = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_x(x,y) = 0$$

$$\frac{\delta E}{dv} = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_y(x,y) = 0$$

Form matrix equation

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

solve via least-squares

One Problem with this...

Assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable for long periods of time.

However, we can easily generalize Lucas-Kanade approach to other 2D parametric motion models (like affine or projective) by introducing a "warp" function W.

$$E(u,v) = \sum [I(x+u, y+v) - T(x,y)]^2 \xrightarrow{\text{generalize}} \sum [\, I(W([x,y];P)) - T([x,y])]^2$$

Robert Collins
CSE486, Penn State

Step-by-Step Derivation

The key to the derivation is Taylor series approximation:

$$I(W([x,y];P+\Delta P)) \approx I(W([x,y];P)) + \nabla I \frac{\partial W}{\partial P} \Delta P$$

We will derive this step-by-step. First, we need two background formula:

Chain rule

$$z = f(x,y) \quad x = g(\tau) \quad y = h(\tau)$$
$$= f(g(\tau), h(\tau))$$

Then

$$\frac{\partial z}{\partial \tau} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial \tau} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial \tau}$$

$$= \frac{\partial z}{\partial x}\frac{\partial g(\tau)}{\partial \tau} + \frac{\partial z}{\partial y}\frac{\partial h(\tau)}{\partial \tau}$$

Taylor series approximation

expand about an initial estimate $\tilde{\tau}$

$$z(\tilde{\tau}+\Delta\tau) = z(\tilde{\tau}) + \left.\frac{\partial z}{\partial \tau}\right|_{\tilde{\tau}} \Delta\tau + \text{higher order terms}$$

Robert Collins
CSE486, Penn State

Step-by-Step Derivation

First consider the expansion for a single variable p

$$I(x,y;\tilde{p}+\Delta p) \approx I(x,y;\tilde{p}) + \left.\frac{\partial I}{\partial p}\right|_{\tilde{p}} \Delta p$$

Taylor approximation

$$= I(x,y;\tilde{p}) + \underbrace{\left[\frac{\partial I}{\partial x}\frac{\partial w_x(p)}{\partial p} + \frac{\partial I}{\partial y}\frac{\partial w_y(p)}{\partial p}\right]_{\tilde{p}}}_{\text{chain rule}} \Delta p$$

Robert Collins
CSE486, Penn State

Step-by-Step Derivation

Now consider warping functions of n variables
$p_1, p_2, \ldots, p_n$

$$w_x = w_x(p_1, p_2, \ldots, p_n)$$
$$w_y = w_y(p_1, p_2, \ldots, p_n)$$

$$I(x, y; \tilde{p}_1 + \Delta p_1, \tilde{p}_2 + \Delta p_2, \ldots, \tilde{p}_n + \Delta p_n) \approx$$

$$I(x, y; \tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_n) + \left[\frac{\partial I}{\partial x}\frac{\partial w_x}{\partial p_1} + \frac{\partial I}{\partial y}\frac{\partial w_y}{\partial p_1}\right]_{\tilde{p}_1} \Delta p_1 \; +$$

$$\left[\frac{\partial I}{\partial x}\frac{\partial w_x}{\partial p_2} + \frac{\partial I}{\partial y}\frac{\partial w_y}{\partial p_2}\right]_{\tilde{p}_2} \Delta p_2 \; +$$

$$\vdots$$

$$\left[\frac{\partial I}{\partial x}\frac{\partial w_x}{\partial p_n} + \frac{\partial I}{\partial y}\frac{\partial w_y}{\partial p_n}\right]_{\tilde{p}_n} \Delta p_n$$

Note that each variable parameter p_i contributes a term of the form

$$\left[\frac{\partial I}{\partial x}\frac{\partial w_x}{\partial p_i} + \frac{\partial I}{\partial y}\frac{\partial w_y}{\partial p_1}\right]_{\tilde{p}_i} \Delta p_i$$

Step-by-Step Derivation

Now let's rewrite the expression as a matrix equation. For each term, we can rewrite:

$$\left[\frac{\partial I}{\partial x}\frac{\partial w_x}{\partial p_i} + \frac{\partial I}{\partial y}\frac{\partial w_y}{\partial p_i}\right]_{\tilde{p}_i} \Delta p_i = \begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix}\begin{bmatrix}\partial w_x/\partial p_i \\ \partial w_y/\partial p_i\end{bmatrix}_{\tilde{p}_i} \Delta p_i$$

So that we have:

$$I(x,y;\tilde{p}_1,\ldots,\tilde{p}_n) + \begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix}\begin{bmatrix}\partial w_x/\partial p_1 \\ \partial w_y/\partial p_1\end{bmatrix}_{\tilde{p}_1} \Delta p_1 +$$

$$+ \begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix}\begin{bmatrix}\partial w_x/\partial p_2 \\ \partial w_y/\partial p_2\end{bmatrix}_{\tilde{p}_2} \Delta p_2 +$$

$$\vdots$$

Robert Collins
CSE486, Penn State

Step-by-Step Derivation

Further collecting the *dw/dp$_i$* terms into a matrix, we can write:

$$I(x,y;\tilde{p}_1,\ldots,\tilde{p}_n) +$$

$$\underbrace{\begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}}_{\substack{\text{gradient} \\ \nabla I}} \underbrace{\begin{bmatrix} \partial w_x/\partial p_1 & \partial w_x/\partial p_2 & \cdots & \partial w_x/\partial p_n \\ \partial w_y/\partial p_1 & \partial w_y/\partial p_2 & \cdots & \partial w_y/\partial p_2 \end{bmatrix}}_{\text{Jacobian } \left(\frac{\partial W}{\partial P}\right)} \underbrace{\begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_n \end{bmatrix}}_{\substack{\text{increment} \\ \text{parameters} \\ \text{to solve for} \\ \Delta P}}$$

which are the terms in the matrix equation:

$$I(W([x,y];P+\Delta P)) \approx I(W([x,y];P)) + \nabla I \frac{\partial W}{\partial P} \Delta P$$

Example: Jacobian of Affine Warp

$$Let \quad W([x,y];P) = [W_x, W_y]$$

general equation of Jacobian

$$\frac{\partial W}{\partial P} = \begin{bmatrix} \frac{\partial W_x}{\partial P_1} & \frac{\partial W_x}{\partial P_2} & \frac{\partial W_x}{\partial P_3} & \cdots & \frac{\partial W_x}{\partial P_n} \\ \frac{\partial W_y}{\partial P_1} & \frac{\partial W_y}{\partial P_2} & \frac{\partial W_y}{\partial P_3} & \cdots & \frac{\partial W_y}{\partial P_n} \end{bmatrix}$$

affine warp function (6 parameters)

$$W([x,y];P) = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \Longrightarrow \quad \frac{\partial W}{\partial P} = \frac{\partial \begin{bmatrix} x + xP_1 + yP_3 + P_5 \\ xP_2 + y + yP_4 + P_6 \end{bmatrix}}{\partial P}$$

$$= \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

Source: “Lucas-Kanade 20 years on: A unifying framework” Baker and Mathews, IJCV 04

Algorithm Summary

Iterate

Warp I to obtain $I(W([x\ y];P))$

Compute the error image $T(x) - I(W([x\ y];P))$

Warp the gradient ∇I with $W([x\ y];P)$

Evaluate $\frac{\partial W}{\partial P}$ at $([x\ y];P)$ (Jacobian)

Compute steepest descent images $\nabla I \frac{\partial W}{\partial P}$

Compute Hessian matrix $\sum (\nabla I \frac{\partial W}{\partial P})^T (\nabla I \frac{\partial W}{\partial P})$

Compute $\sum (\nabla I \frac{\partial W}{\partial P})^T (T(x,y) - I(W([x,y];P)))$

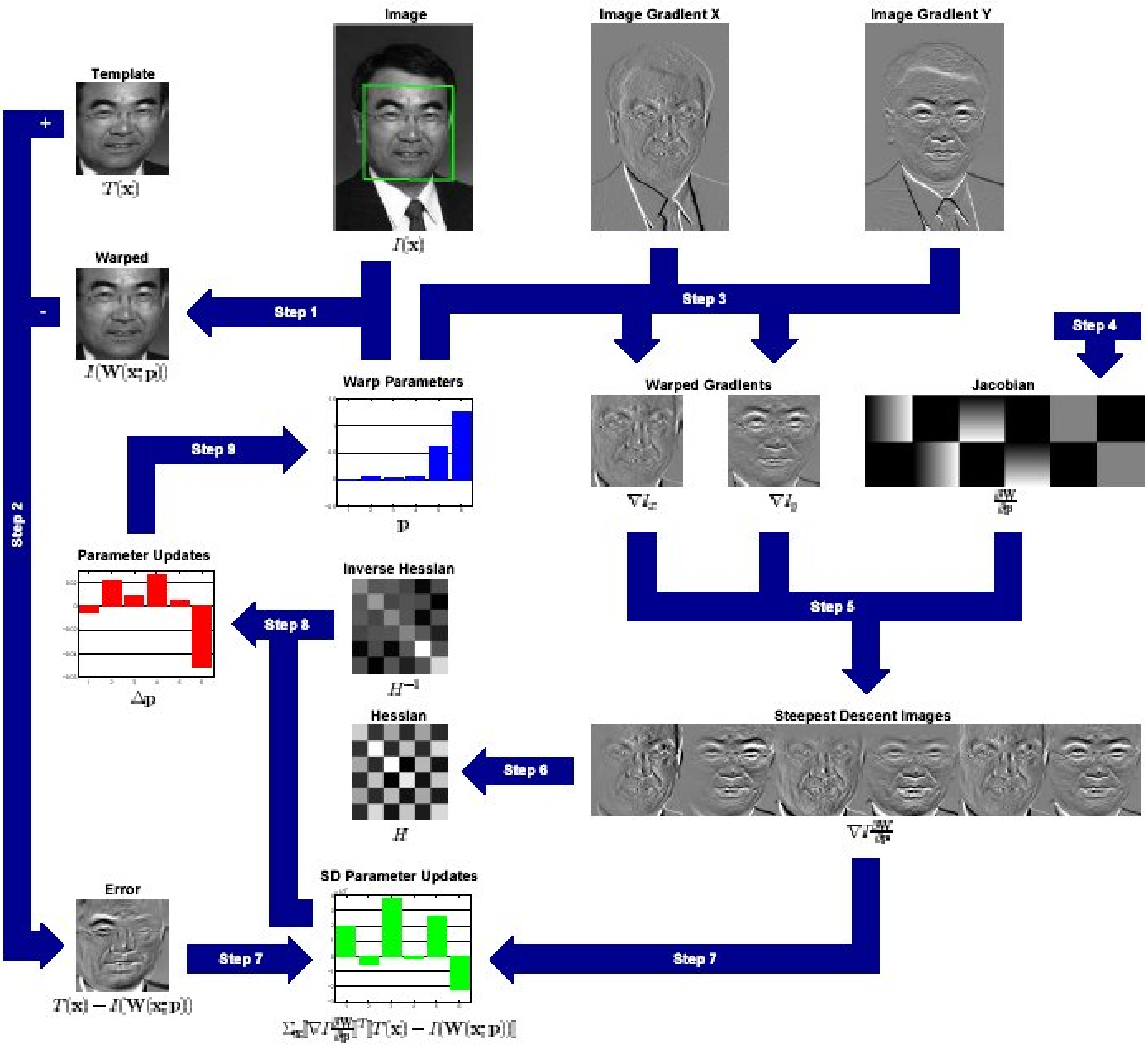
Compute ΔP

Update $P \longleftarrow P + \Delta P$

Until ΔP magnitude is negligible

Robert Collins
CSE486, Penn State

Algorithm At a Glance



Source: "Lucas-Kanade 20 years on: A unifying framework" Baker and Mathews, IJCV 04

State of the Art Lucas Kanade Tracking

Tracking facial mesh models (piecewise affine)

- 230 Frames Per Second

- Papers:
  - Original Paper [Baker and Matthews, CVPR, 2001]
  - Inverse Compositional Algorithm [Baker and Matthews, IJCV, 2004]
  - Application to AAMs [Matthews and Baker, IJCV, 2004]

Baker, Matthews, CMU