CS448A: Experiments in Motion Capture

10/11/2000

• Review of Lucas-Kanade + Matlab examples/demo
• Analysis / Extensions

• Support Maps / Layers:
  - Robust Norm
  - Layered Motion
  - Background Subtraction
  - Color Based Tracking

• Planning: Projects, Presentation
• Recording Session

Lucas-Kanade: Function Minimization: 1D Image

\[ E(u) = \sum_x (F(x+u) - G(x))^2 \]
\[ \approx \sum_x (F_x(x) u - F_t(x))^2 \]

Linearization:

Intensity

Spatial Gradient
Temporal Gradient
Lucas-Kanade: Function Minimization: 2D Image

\[
E(u, v) = \sum_{x, y \in \text{ROI}} (F(x + u, y + v) - G(x))^2
\]

\[
\approx \sum_{x, y \in \text{ROI}} (F_x(x, y)u + F_y(x, y)v - F_t(x, y))^2
\]

Spatial Gradient  Temporal Gradient

Lucas-Kanade: Function Minimization: 2D Image

Minimize \( E(u, v) \):

\[
\frac{\partial E}{\partial u} = 0 \quad \Rightarrow \quad \begin{bmatrix} \sum F_x^2 & \sum F_x F_y \\ \sum F_x F_y & \sum F_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum F_t F_x \\ \sum F_t F_y \end{bmatrix}
\]

\[
C \begin{bmatrix} u \\ v \end{bmatrix} = D
\]

\[
\begin{bmatrix} u \\ v \end{bmatrix} = C^{-1} D
\]
Lucas-Kanade: Implementation

- Step 0: Initialization (for now by hand – more later)
- Step 1: compute: C and D and solve for (u,v):
  - How to compute gradients Fx, Fy, Ft
- Step 2: re-warp image G:
  - How to do sub-pixel image interpolation
- Step 3: Loop:
  - How to measure error / terminate

Lucas-Kanade: Implementation

- Step 1: compute: C and D and solve for (u,v):
  - How to compute gradients Fx, Fy, Ft

A) Finite pixel difference ??
(Problems with image noise, local minima)

B) Better: Convolution with special Filter
B) Better: Convolution with special Filter

- Blurr Image with Convolution of *Gaussian Kernel*
  - weighted averaging (center more strongly)
  - sigma (standard deviation) -> spread

Lucas-Kanade: Implementation

B) Better: Convolution with special Filter

- Blurr Image AND compute derivative:
Lucas-Kanade: Implementation

Why Gaussian / Gauss. Deriv.
- Mathematical convenience
- Efficiency
- Separable
- Central Limit Theorem
- Signal to noise ratio
- Localisation (for derivative)

Lucas-Kanade: Implementation

- Step 1: compute: C and D and solve for (u,v):
  - How to compute gradients Fx,Fy,Ft

- Compute with Gaussian kernel
  (coarse-to-fine strategy by decreasing sigma)
Lucas-Kanade: Implementation

- Step 0: Initialization (for now by hand – more later)
- Step 1: compute: C and D and solve for (u,v):
  - How to compute gradients F_x, F_y, F_t

- Step 2: re-warp image G:
  - How to do sub-pixel image interpolation

- Step 3: Loop:
  - How to measure error / terminate

Matlab demo of interp2 →
Lucas-Kanade: Implementation

- Step 0: Initialization (for now by hand – more later)
- Step 1: Compute: C and D and solve for (u,v):
  - How to compute gradients Fx,Fy,Ft
- Step 2: Re-warp image G:
  - How to do sub-pixel image interpolation
- Step 3: Loop:
  - How to measure error / terminate

Look at: $\| F_t \|^2$
Lucas-Kanade: Implementation
CS448A: Experiments in Motion Capture

10/11/2000

• Review of Lucas-Kanade + Matlab examples/demo
  • Analysis / Extensions

• Support Maps / Layers:
  - Robust Norm
  - Layered Motion
  - Background Subtraction
  - Color Based Tracking

• Planning: Projects, Presentation
• Recording Session

Lucas-Kanade: Graphical Analysis

White-board ->
Lucas-Kanade: Singularities

Minimize $E(u,v)$:

\[
\frac{\partial E}{\partial u} = 0 \quad \Rightarrow \quad \begin{bmatrix} \sum F_x^2 & \sum F_x F_y \\ \sum F_x F_y & \sum F_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum F_x F_t \\ \sum F_y F_t \end{bmatrix}
\]

\[
C \begin{bmatrix} u \\ v \end{bmatrix} = D
\]

\[
\begin{bmatrix} u \\ v \end{bmatrix} = C^{-1} D
\]
Lucas-Kanade: Singularities

\[ \sum F_x^2 \frac{1}{\sum F_x F_y} \sum F_x F_y \left[ \begin{array}{c} u \\ v \end{array} \right] = \sum F_y^2 \frac{1}{\sum F_x F_y} \]

\[ C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \]

\[ F_x = 0, F_y = 0 \quad F_y = 0 \]

Lucas-Kanade: Singularities

\[ \sum F_x^2 \frac{1}{\sum F_x F_y} \sum F_x F_y \left[ \begin{array}{c} u \\ v \end{array} \right] = \sum F_y^2 \frac{1}{\sum F_x F_y} \]

\[ C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \]

\[ eig(C) = (e1,0) \]
Lucas-Kanade: Aperture Problem

\[
C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}
\]

\[
eig(C) = (e_1, 0)
\]

Bergen et al.
Aperture Problem: Fix ???

\[
\begin{array}{c}
\sum F_x^2 \\
\sum F_x F_y \\
\sum F_y^2
\end{array}
\]

\[\text{eig}(C) = (e1, e2)\]

-Avoid It!

- Use Eigenvalues to initialize “Good Features” (Shi-Tomasi)

- Good Feature Location: \(\text{min}(\text{eig1, eig2}) > a\)

Aperture Problem: Fix ???

“Good Features” (Shi-Tomasi)

Figure 10: The first frame of a 26 frame sequence taken with a forward moving camera.

Figure 11: The features selected according to the texturedness criterion of section 4.
Aperture Problem: Fix ???

\[
\sum F_x^2 \sum F_y^2 \\
\sum F_x F_y \sum F_y^2
\]

\[eig(C) = (e1, e2)\]

- Hack It!

- regularize C:

\[
C_{reg} = C + \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}
\]

Aperture Problem: Fix ???

\[
\sum F_x^2 \sum F_y^2 \\
\sum F_x F_y \sum F_y^2
\]

\[eig(C) = (e1, e2)\]

- Justify Hack!

- Simoncelli et al 1991:

\[
\lambda_x = \left[ \frac{M}{(\sigma_1 ||f_x||^2 + \sigma_2)} + \lambda_x^0 \right]^{-1}
\]

\[
\mu_x = -\lambda_x \cdot \frac{\delta}{(\sigma_1 ||f_x||^2 + \sigma_2)}
\]
Aperture Problem: Fix ???

\[
\begin{align*}
\sum F_x^2 & \quad \sum F_x F_y \\
\sum F_x F_y & \quad \sum F_y^2
\end{align*}
\]

\[\text{eig}(C) = (e_1, e_2)\]

- Increase Aperture!

- Coarse-to-fine Pyramids (Bergen et al, Simoncelli)

Aperture Problem: Fix ???

\[
\begin{align*}
\sum F_x^2 & \quad \sum F_x F_y \\
\sum F_x F_y & \quad \sum F_y^2
\end{align*}
\]

\[\text{eig}(C) = (e_1, e_2)\]

- Increase Aperture!

- Larger Integration ROI -> more complex motion model
Affine Extension

Affine Motion Model:

\[
\begin{bmatrix}
    x + u \\
y + v
\end{bmatrix}
= \begin{bmatrix}
    a_1 & a_2 \\
a_3 & a_4
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
+ \begin{bmatrix}
a_5 \\
a_6
\end{bmatrix}
\]

- 2D Translation
- 2D Rotation
- Scale in X / Y
- Shear

Matlab demo ->

Affine Extension

Affine Motion Model -> Lucas-Kanade:

\[
E(u, v) = \sum_{x, y \in ROI} (F(x + u, y + v) - G(x, y))^2
\]

\[
\begin{bmatrix}
a_1 x + a_2 y + a_3 \\
a_4 x + a_5 y + a_6
\end{bmatrix}
\rightarrow
\begin{bmatrix}
da_1 \\
C ... \\
da_6
\end{bmatrix}
= D
\]

Matlab demo ->
CS448A: Experiments in Motion Capture

10/11/2000

- Review of Lucas-Kanade + Matlab examples/demo
- Analysis / Extensions

**Support Maps / Layers:**
- Robust Norm
- Layered Motion
- Background Subtraction
- Color Based Tracking

- Planning: Projects, Presentation
- Recording Session

Support Maps / Layers

- L2 Norm vs **Robust Norm**

- **Dangers** of least square fitting:

  White-board -> Outliers are bad
Support Maps / Layers

- L2 Norm vs Robust Norm

- Dangers of least square fitting:
Support Maps / Layers

- Robust Norm -- good for outliers
- nonlinear optimization

![Diagram of robust function](image)

Support Maps / Layers

- Black-Jepson-95

![Images of Outliers, Robust 2, Outliers 2](image)

*Fig. 6. Robust matching with shadow (see text).*
Support Maps / Layers

- Iterative Technique

Add weights to each pixel eq (white board)

Support Maps / Layers

- how to compute weights?
  
  -> previous iteration: *how good does G-warp matches F?*

  -> probabilistic distance: *Gaussian:*
Support Maps / Layers

- More General: *Layered Motion (Jepson/Black, Weiss/Adelson, …)*

Figure 11: Segmentation achieved when $\sigma$ is lowered even more. The algorithm finds four segments – the flower bed and the house are segregated.

Support Maps / Layers

- Special Cases of *Layered Motion:*
  - Background substraction
  - Outlier rejection (== robust norm)
  - Simplest Case: Each Layer has *uniform color*
Support Maps / Layers

- Color Layers:

\[ P(\text{skin} \mid F(x,y)) \]

Support Maps / Layers

- more details for homework:

  - Record Images
  - Compute RGB, or HSV, or HSV-on-a-cone
  - Label Skin region
  - Compute mean and covariance in color space
  - For each new video frame:
    - Compute for each pixel \( p1 = P(F(x,y) \mid \text{skin-color}) \) (Gaussian)
    - Compute for each pixel \( p2 = P(F(x,y) \mid \text{don’t know}) \) (uniform)
    - Compute \( P(\text{skin-color} \mid F(x,y) ) \cdot p1/(p1+p2) \)
CS448A: Experiments in Motion Capture

10/11/2000

- Review of Lucas-Kanade + Matlab examples/demo
- Analysis / Extensions

- Support Maps / Layers:
  - Robust Norm
  - Layered Motion
  - Background Subtraction
  - Color Based Tracking

- Planning: Projects, Presentation
- Recording Session