ECG782: MULTIDIMENSIONAL DIGITAL SIGNAL PROCESSING

MOTION
OUTLINE

- Motion Analysis Motivation
- Differential Motion
- Optical Flow

Note: most of the content comes from Sonka Chapter 16
DENSE MOTION ESTIMATION

- Motion is extremely important in vision
- Biologically: motion indicates what is food and when to run away
  - We have evolved to be very sensitive to motion cues (peripheral vision)
- Alignment of images and motion estimation is widely used in computer vision
  - Optical flow
  - Motion compensation for video compression
  - Image stabilization
  - Video summarization
Even limited motion information is perceptually meaningful

http://www.biomotionlab.ca/Demos/BMLwalker.html
MOTION ESTIMATION

- Input: sequence of images
- Output: point correspondence
- Prior knowledge: decrease problem complexity
  - E.g. camera motion (static or mobile), time interval between images, etc.

- Motion detection
  - Simple problem to recognize any motion (e.g. security)
- Moving object detection and location
  - Feature correspondence: “Feature Tracking”
  - Pixel (dense) correspondence: “Optical Flow”
- **Motion description**
  - Motion/velocity field – velocity vector associated with corresponding keypoints
  - Optical flow – dense correspondence that requires small time distance between images

- **Motion assumptions**
  - Maximum velocity – object must be located in an circle defined by max velocity
  - Small acceleration – limited acceleration
  - Common motion – all object points move similarly
  - Mutual correspondence – rigid objects with stable points

---

*Figure 16.1: Object motion assumptions. (a) Maximum velocity (shaded circle represents area of possible object location). (b) Small acceleration (shaded circle represents area of possible object location at time \( t_2 \)). (c) Common motion and mutual correspondence (rigid objects).*

© Cengage Learning 2015.
Two interrelated components:
- Localization and representation of object of interest (target)
  - Bottom-up process: deal with appearance, orientation, illumination, scale, etc.
- Trajectory filtering and data association
  - Top-down process: consider object dynamics to infer motion (motion models)
Motion Analysis Motivation
Differential Motion
Optical Flow

Note: most of the content comes from Sonka Chapter 16
Simple motion detection possible with image subtraction
- Requires a stationary camera and constant illumination
- Also known as change detection

Difference image

\[ d(i, j) = \begin{cases} 
1 & |f_1(i, j) - f_2(i, j)| > \epsilon \\
0 & \text{else} 
\end{cases} \]
- Binary image that highlights moving pixels

What are the various “detections” from this method?
- Chapter 16.1

Figure 16.2: Motion detection. (a) First frame of the image sequence. (b) Frame 2 of the sequence. (c) Last frame (frame 5). (d) Differential motion image constructed from image frames 1 and 2 (inverted to improve visualization). © M. Soska 2015.
Motion is quite important
- Indicates an object of interest

Background subtraction:
- Given an image (usually a video frame), identify the foreground objects in that image
  - Assume that foreground objects are moving
  - Typically, moving objects more interesting than the scene
  - Simplifies processing – less processing cost and less room for error
- Often used in traffic monitoring applications
  - Vehicles are objects of interest (counting vehicles)

- Human action recognition (run, walk, jump, ...)
- Human-computer interaction ("human as interface")
- Object tracking
REQUIREMENTS

- A reliable and robust background subtraction algorithm should handle:
  - Sudden or gradual illumination changes
    - Light turning on/off, cast shadows through a day
  - High frequency, repetitive motion in the background
    - Tree leaves blowing in the wind, flag, etc.
  - Long-term scene changes
    - A car parks in a parking spot
BASIC APPROACH

- Estimate the background at time $t$
- Subtract the estimated background from the current input frame
- Apply a threshold, $Th$, to the absolute difference to get the foreground mask.
  - $|I(x, y, t) - B(x, y, t)| > Th = F(x, y, t)$

How can we estimate the background?
FRAME DIFFERENCING

- Background is estimated to be the previous frame
  - \( B(x, y, t) = I(x, y, t - 1) \)
- Depending on the object structure, speed, frame rate, and global threshold, may or may not be useful
  - Usually not useful – generates impartial objects and ghosts
FRAME DIFFERENCING EXAMPLE

$Th = 25$

$Th = 50$

$Th = 100$

$Th = 200$
MEAN FILTER

- Background is the mean of the previous $N$ frames
  \[ B(x, y, t) = \frac{1}{N} \sum_{i=0}^{N-1} I(x, y, t - i) \]
  - Produces a background that is a temporal smoothing or "blur"
- $N = 10$
MEAN FILTER

- $N = 20$
  - Estimated Background
  - Foreground Mask

- $N = 50$
  - Estimated Background
  - Foreground Mask
MEDIAN FILTER

- Assume the background is more likely to appear than foreground objects
  \[ B(x, y, t) = \text{median}(I(x, y, t - i)), \ i \in \{0, N - 1\} \]

- \( N = 10 \)
**MEDIAN FILTER**

- **$N = 20$**

- **$N = 50$**
FRAME DIFFERENCE ADVANTAGES

- Extremely easy to implement and use
- All the described variants are pretty fast
- The background models are not constant
  - Background changes over time
FRAME DIFFERENCING SHORTCOMINGS

- Accuracy depends on object speed/frame rate
- Mean and median require large memory
  - Can use a running average
    \[ B(x, y, t) = (1 - \alpha)B(x, y, t - 1) + \alpha I(x, y, t) \]
    - \( \alpha \) – is the learning rate
- Use of a global threshold
  - Same for all pixels and does not change with time
  - Will give poor results when the:
    - Background is bimodal
    - Scene has many slow moving objects (mean, median)
    - Objects are fast and low frame rate (frame diff)
    - Lighting conditions change with time
IMPROVING BACKGROUND SUBTRACTION

- Adaptive Background Mixture Models for Real-Time Tracking
  - Chris Stauffer and W.E.L. Grimson
- “The” paper on background subtraction
  - Over 10k citations since 1999

- Will read this and see more later
  - Example of paper presentation
OUTLINE

- Motion Analysis Motivation
- Differential Motion
- Optical Flow

- Note: most of the content comes from Sonka Chapter 16
OPTICAL FLOW

- Dense pixel correspondence
OPTICAL FLOW

- Dense pixel correspondence
- Hamburg Taxi Sequence
Motion estimation between images requires an error metric for comparison.

- **Sum of squared differences (SSD)**
  
  \[
  E_{SSD}(u) = \sum_i [I_1(x_i + u) - I_0(x_i)]^2 = \sum_i e_i^2
  \]

  - \(u = (u, v)\) – is a displacement vector (can be subpixel)
  - \(e_i\) - residual error

- **Brightness constancy constraint**
  
  - Assumption that corresponding pixels will retain the same value in two images
  - Objects tend to maintain the perceived brightness under varying illumination conditions [Horn 1974]

- Color images processed by channels and summed or converted to colorspace that considers only luminance
SSD IMPROVEMENTS

- As we have seen, SSD is the simplest approach and can be improved
- Robust error metrics
  - $L_1$ norm (sum absolute differences)
    - Better outlier resilience
- Spatially varying weights
  - Weighted SSD to weight contribution of each pixel during matching
    - Ignore certain parts of the image (e.g. foreground), down-weight objects during images stabilization
- Bias and gain
  - Normalize exposure between images
    - Address brightness constancy
CORRELATION

- Instead of minimizing pixel differences, maximize correlation
- Normalized cross-correlation

\[ E_{\text{NCC}}(u) = \frac{\sum_i [I_0(x_i) - \overline{I_0}] [I_1(x_i + u) - \overline{I_1}]}{\sqrt{\sum_i [I_0(x_i) - \overline{I_0}]^2} \sqrt{\sum_i [I_1(x_i + u) - \overline{I_1}]^2}}, \]

\[
\overline{I_0} = \frac{1}{N} \sum_i I_0(x_i) \quad \text{and} \\
\overline{I_1} = \frac{1}{N} \sum_i I_1(x_i + u)
\]

- Normalize by the patch intensities
- Value is between \([-1, 1]\) which makes it easy to use results (e.g. threshold to find matching pixels)
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 $I$

Key assumptions
- **Color constancy**: a point in $H$ looks the same in $I$
  - For grayscale images, this is brightness constancy
- **Small motion**: points do not move very far

This is called the optical flow problem
Let’s look at these constraints more closely

Brightness constancy:
- \( H(x, y) = I(x + u, y + v) \)

Small motion
- \( u \) and \( v \) are less than 1 pixel
- Take a Taylor series expansion of \( I(x + u, y + v) \)

\[
I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms}
\]

\[
\approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v
\]
Combining these two equations

\[ 0 = I(x + u, y + v) - H(x, y) \]

\[ \approx I(x, y) + I_xu + I_yv - H(x, y) \]

\[ \approx (I(x, y) - H(x, y)) + I_xu + I_yv \]

\[ \approx I_t + I_xu + I_yv \]

\[ \approx I_t + \nabla I \cdot [u \ v] \]

In the limit as \( u \) and \( v \) go to zero, this becomes exact

\[ 0 = I_t + \nabla I \cdot \left[ \frac{\partial x}{\partial t} \frac{\partial y}{\partial t} \right] \]
OPTICAL FLOW EQUATION

\[ 0 = I_t + \nabla I \cdot [u \ v] \]

- How many unknowns and equations per pixel?
  - \( u \) and \( v \) are unknown - 1 equation, 2 unknowns

- Intuitively, what does this constraint mean?
  - The component of the flow in the gradient direction is determined
  - The component of the flow parallel to an edge is unknown

- This explains the Barber Pole illusion
  - [Link to Barber Pole Illusion](http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm)

- If \((u, v)\) satisfies the equation, so does \((u + u', v + v')\) if \(\nabla I \cdot [u' \ v'] = 0\)
APERTURE PROBLEM

Actual Motion
Basic idea: assume motion field is smooth

Horn & Schunk: add smoothness term

$$\int \int (I_t + \nabla I \cdot [u \ v])^2 + \lambda^2(||\nabla u||^2 + ||\nabla v||^2) \ dx \ dy$$

Lucas & Kanade: assume locally constant motion
- Pretend the pixel’s neighbors have the same (u,v)

Many other methods exist. Here’s an overview:
- [http://vision.middlebury.edu/flow/](http://vision.middlebury.edu/flow/)
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!

\[
0 = I_t(p_i) + \nabla I(p_i) \cdot [u \ v]
\]

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

\[
A_{25 \times 2}, \quad d_{2 \times 1}, \quad b_{25 \times 1}
\]
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!

\[
0 = I_t(p_i)[0, 1, 2] + \nabla I(p_i)[0, 1, 2] \cdot [u \ v]
\]

\[
\begin{bmatrix}
I_x(p_1)[0] & I_y(p_1)[0] \\
I_x(p_1)[1] & I_y(p_1)[1] \\
I_x(p_1)[2] & I_y(p_1)[2] \\
\vdots & \vdots \\
I_x(p_{25})[0] & I_y(p_{25})[0] \\
I_x(p_{25})[1] & I_y(p_{25})[1] \\
I_x(p_{25})[2] & I_y(p_{25})[2]
\end{bmatrix}
\begin{bmatrix}
u \\v
\end{bmatrix}
\begin{bmatrix}
I_t(p_1)[0] \\
I_t(p_1)[1] \\
I_t(p_1)[2] \\
\vdots \\
I_t(p_{25})[0] \\
I_t(p_{25})[1] \\
I_t(p_{25})[2]
\end{bmatrix}
\]

\[
A_{75 \times 2} d_{2 \times 1} b_{75 \times 1}
\]
**LUCAS-KANADE FLOW**

- **Problem:** More equations than unknowns
  \[
  \begin{bmatrix}
  A_{25x2} & d_{2x1} & b_{25x1}
  \end{bmatrix}
  \rightarrow \text{minimize } ||Ad - b||^2
  \]

- **Solution:** Solve least squares problem
  - Minimum LS solution by finding \(d\)
    \[
    \begin{bmatrix}
    \sum I_x I_x & \sum I_x I_y \\
    \sum I_x I_y & \sum I_y I_y
    \end{bmatrix}
    \begin{bmatrix}
    u \\
    v
    \end{bmatrix}
    = - \begin{bmatrix}
    \sum I_x I_t \\
    \sum I_y I_t
    \end{bmatrix}
    \]
  - The summations are over all pixels in the K x K window
  - This technique was first proposed by Lucas & Kanade (1981)
CONDITIONS FOR SOLVABILITY

- Optimal \((u, v)\) satisfies Lucas-Kanade equation

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

- When is This Solvable?
  - \(A^T A\) should be invertible
  - \(A^T A\) should not be too small due to noise
    - Eigenvalues \(l_1\) and \(l_2\) of \(A^T A\) should not be too small
  - \(A^T A\) should be well-conditioned
    - \(l_1/l_2\) should not be too large \((l_1 = \text{larger eigenvalue})\)

- \(A^T A\) is the Harris matrix (see Interest Points)
  - Finds “corners” (areas of gradient in orthogonal directions)
This is a two image problem BUT

- Can measure sensitivity by just looking at one of the images!
- This tells us which pixels are easy to track, which are hard
  - Very useful for feature tracking...
APERTURE PROBLEM

Actual Motion
APERTURE PROBLEM

Perceived Motion
What are the potential causes of errors in this procedure?

- Suppose $A^TA$ is easily invertible
- Suppose there is not much noise in the image

When our assumptions are violated

- Brightness constancy is not satisfied
- The motion is not small
- A point does not move like its neighbors
  - Window size is too large
  - What is the ideal window size?
Recall our small motion assumption

\[ 0 = I(x + u, y + v) - H(x, y) \]
\[ \approx I(x, y) + I_x u + I_y v - H(x, y) \]

Not exact, need higher order terms to do better

\[ = I(x, y) + I_x u + I_y v + \text{higher order terms} - H(x, y) \]

Results in polynomial root finding problem

- Can be solved using Newton’s method (also known as Newton-Raphson)

Lucas-Kanade method does a single iteration of Newton’s method

- Better results are obtained with more iterations
ITERATIVE REFINEMENT

- Iterative Lucas-Kanade Algorithm

1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp H towards I using the estimated flow field
   - Use image warping techniques
3. Repeat until convergence
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?
REDUCE THE RESOLUTION!
COARSE-TO-FINE OPTICAL FLOW ESTIMATION

Gaussian pyramid of image H

Gaussian pyramid of image I

- $u=10 \text{ pixels}$
- $u=5 \text{ pixels}$
- $u=2.5 \text{ pixels}$
- $u=1.25 \text{ pixels}$
COARSE-TO-FINE OPTICAL FLOW ESTIMATION

Gaussian pyramid of image H

run iterative L-K
warp & upsample
run iterative L-K

Gaussian pyramid of image I
OPTICAL FLOW RESULTS

Lucas-Kanade without pyramids

Fails in areas of large motion

Lucas-Kanade with Pyramids
ROBUST METHODS

- L-K minimizes a sum-of-squares error metric
- Least squares techniques overly sensitive to outliers

Error metrics

- quadratic: \( \rho(x) = x^2 \)
- truncated quadratic: \( \rho_{\alpha,\lambda}(x) = \begin{cases} \lambda x^2 & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}} \\ \alpha & \text{otherwise} \end{cases} \)
- lorentzian: \( \rho_\sigma(x) = \log \left( 1 + \frac{1}{2} \left( \frac{x}{\sigma} \right)^2 \right) \)
**ROBUST OPTICAL FLOW**

- Robust Horn & Schunk
  \[ \int \int \rho(I_t + \nabla I \cdot [u \ v]) + \lambda^2 \rho(||\nabla u||^2 + ||\nabla v||^2) \, dx \, dy \]

- Robust Lucas-Kanade
  \[ \sum_{(x,y) \in W} \rho(I_t + \nabla I \cdot [u \ v]) \]


BENCHMARKING OPTICAL FLOW ALGORITHMS

- Middlebury flow page
  - http://vision.middlebury.edu/flow/
FLOW QUALITY EVALUATION
FLOW QUALITY EVALUATION
FLOW QUALITY EVALUATION

- Middlebury flow page
  - http://vision.middlebury.edu/flow/

Ground Truth
FLOW QUALITY EVALUATION

- Middlebury flow page
  - http://vision.middlebury.edu/flow/

Lucas-Kanade flow

Ground Truth
FLOW QUALITY EVALUATION

- Middlebury flow page
  - [http://vision.middlebury.edu/flow/](http://vision.middlebury.edu/flow/)

DISCUSSION: FEATURES VS. FLOW?

- Features are better for:

- Flow is better for:
ADVANCED TOPICS

- **Particles:** combining features and flow

- **State-of-the-art feature tracking/SLAM**
  - Georg Klein et al. [http://www.robots.ox.ac.uk/~gk/](http://www.robots.ox.ac.uk/~gk/)

- **Deep Motion**
  - [FlowNet2.0](http://FlowNet2.0) – CNN architecture to learn flow directly
  - [DeepFlow](http://DeepFlow) – Deep matching
  - [Gladh ICPR2016](http://Gladh ICPR2016) – combined deep + hand crafted
  - [Deep Motion](http://Deep Motion) – flow + segmentation