Computer Vision

CSE/ECE 576
Motion and Optical Flow

Linda Shapiro
Professor of Computer Science & Engineering
Professor of Electrical & Computer Engineering
We live in a moving world

• Perceiving, understanding and predicting motion is an important part of our daily lives
Motion and perceptual organization

• Even “impoverished” motion data can evoke a strong percept

Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept

Seeing motion from a static picture?

http://www.ritsumei.ac.jp/~akitaoka/index-e.html
More examples
How is this possible?

• The true mechanism is yet to be revealed

• FMRI data suggest that illusion is related to some component of eye movements

• We don’t expect computer vision to “see” motion from these stimuli, yet
The cause of motion

• Three factors in imaging process
  – Light
  – Object
  – Camera

• Varying either of them causes motion
  – Static camera, moving objects (surveillance)
  – Moving camera, static scene (3D capture)
  – Moving camera, moving scene (sports, movie)
  – Static camera, moving objects, moving light (time lapse)
Motion scenarios (priors)

- Static camera, moving scene
- Moving camera, static scene
- Moving camera, moving scene
- Static camera, moving scene, moving light
We still don’t touch these areas
How can we recover motion?
Recovering motion

• **Feature-tracking**
  - Extract visual features (corners, textured areas) and “track” them over multiple frames

• **Optical flow**
  - Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow)

Two problems, one registration method

Feature tracking

• Challenges
  – Figure out which features can be tracked
  – Efficiently track across frames
  – Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
  – Drift: small errors can accumulate as appearance model is updated
  – Points may appear or disappear: need to be able to add/delete tracked points
What is Optical Flow?
What is Optical Flow? Movement
What is Optical Flow?

Movement
What is Optical Flow?

Movement
What is Optical Flow?

Movement
What is Optical Flow?

Movement

Object
What is Optical Flow?

Pan

Movement
What is Optical Flow?

Movement:
Forward
What is Optical Flow?

Movement
Why do we want Optical Flow?
Why do we want Optical Flow?

Motion Estimation
Why do we want Optical Flow?

Motion Estimation

Object Tracking
Why do we want Optical Flow?

Motion Estimation

Object Tracking

Visual Odometry

Estimating the position of a robot.
How do we find the flow in an image?
Feature Matching
Previously: Features!

- Highly descriptive local regions
- Ways to describe those regions
- Useful for:
  - Matching
  - Recognition
  - Detection
Feature Matching
Feature Matching
Feature Matching

SOLVED!
Feature Matching

Disadvantages:
Feature Matching

Disadvantages:

- Sparse!
Feature Matching

Disadvantages:

- Sparse!
- Feature alignment not exact
Feature Matching
Feature Matching

Disadvantages:

-Sparse!

-Feature alignment not exact

-Low accuracy
Feature Matching

Disadvantages:  
- Sparse!
- Feature alignment not exact
- Low accuracy

Advantages:
Feature Matching

Disadvantages:
- Sparse!
- Feature alignment not exact
- Low accuracy

Advantages:
- Scale/rotation invariant
- *kinda* lighting invariant
- Can handle large movements
Feature Matching

Disadvantages:
- Sparse!
- Feature alignment not exact
- Low accuracy

Advantages:
- Scale/rotation invariant * kinda * lighting invariant
- Can handle large movements

Overall: Doesn’t work very well for Optical Flow
What do we do instead?
Feature tracking

- Given two subsequent frames, estimate the point translation

- Key assumptions of Lucas-Kanade Tracker
  - **Brightness constancy**: projection of the same point looks the same in every frame
  - **Small motion**: points do not move very far
  - **Spatial coherence**: points move like their neighbors
The brightness constancy constraint

\[ I(x, y, t) = I(x + u, y + v, t + 1) \]

Take Taylor expansion of \( I(x+u, y+v, t+1) \) at \((x,y,t)\) to linearize the right side:

\[
I(x+u, y+v, t+1) \approx I(x, y, t) + I_x \cdot u + I_y \cdot v + I_t
\]

\( l_t(x,y) = l(x,y,t+1) - l(x,y,t) \)

• **Difference in intensity at the same pixel between one image and the previous one.**
The brightness constancy constraint

\[
I(x+u, y+v, t+1) \approx I(x, y, t) + I_x \cdot u + I_y \cdot v + I_t
\]

\[
I(x+u, y+v, t+1) - I(x, y, t) = +I_x \cdot u + I_y \cdot v + I_t
\]

So:

\[
I_x \cdot u + I_y \cdot v + I_t \approx 0
\]

\[
\rightarrow \nabla I \cdot [u \ v]^T + I_t = 0
\]
The brightness constancy constraint

Can we use this equation to recover image motion \((u,v)\) at each pixel?

\[
\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix}^T + I_t = 0
\]

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns \((u,v)\)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured
Solving the ambiguity...


- How to get more equations for a pixel?

- **Spatial coherence constraint**
  - Assume the pixel’s neighbors have the same \((u,v)\)
    - If we use a 5x5 window, that gives us 25 equations per pixel
      
      \[
      0 = I_t(p_i) + \nabla I(p_i) \cdot [u \ v] 
      \]

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

- Least squares problem:

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

\[
A \begin{bmatrix} d \end{bmatrix} = b
\]

\[
\begin{bmatrix} 25 \times 2 & 2 \times 1 & 25 \times 1 \end{bmatrix}
\]
Matching patches across images

• Overconstrained linear system

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

Least squares solution for \( d \) given by

\[
(A^T A) d = A^T b
\]

The summations are over all pixels in the \( K \times K \) window

\[
d = (A^T A)^{-1} A^T b
\]
Conditions for solvability

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

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

\(A^T A\)

\(A^T b\)

When is this solvable? I.e., what are good points to track?

- \(A^T A\) should be invertible
- \(A^T A\) should not be too small due to noise
  - eigenvalues \(\lambda_1\) and \(\lambda_2\) of \(A^T A\) should not be too small
- \(A^T A\) should be well-conditioned
  - \(\lambda_1 / \lambda_2\) should not be too large (\(\lambda_1 =\) larger eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector
Aperture problem

Corners

Lines

Flat regions
\[ \sum \nabla I (\nabla I)^T \]

- large gradients, all the same
- large \( \lambda_1 \), small \( \lambda_2 \)
Low Texture Region

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

- gradients have small magnitude
- small $\lambda_1$, small $\lambda_2$
High Texture Region

\[ \sum \nabla I (\nabla I)^T \]

- gradients are different, large magnitudes
- large \( \lambda_1 \), large \( \lambda_2 \)
Errors in Lukas-Kanade

• What are the potential causes of errors in this procedure?
  – Suppose $A^T A$ 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?
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 1 (t)  run iterative L-K  warp & upsample  run iterative L-K  Gaussian pyramid of image 2 (t+1)
A Few Details

• **Top Level**
  – Apply L-K to get a flow field representing the flow from the first frame to the second frame.
  – Apply this flow field to warp the first frame toward the second frame.
  – Rerun L-K on the new warped image to get a flow field from it to the second frame.
  – Repeat till convergence.

• **Next Level**
  – Upsample the flow field to the next level as the first guess of the flow at that level.
  – Apply this flow field to warp the first frame toward the second frame.
  – Rerun L-K and warping till convergence as above.

• **Etc.**
Coarse-to-fine optical flow estimation

Gaussian pyramid of image 1

Gaussian pyramid of image 2
The Flower Garden Video

What should the optical flow be?
Optical Flow Results

Lucas-Kanade without pyramids
Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Optical Flow Results

Lucas-Kanade with Pyramids

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Flow quality evaluation
Flow quality evaluation
Flow quality evaluation

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

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

Lucas-Kanade flow

Ground Truth

Color encoding of flow vectors
Flow quality evaluation

- Middlebury flow page
  - [http://vision.middlebury.edu/flow/](http://vision.middlebury.edu/flow/)
Video stabilization
Video denoising
Video super resolution

Low-Res
Robust Visual Motion Analysis:
Piecewise-Smooth Optical Flow

Ming Ye
Electrical Engineering
University of Washington
Problem Statement:

Assuming only brightness conservation and piecewise-smooth motion, find the optical flow to best describe the intensity change in three frames.
Approach: Matching-Based Global Optimization

• **Step 1.** Robust local gradient-based method for high-quality initial flow estimate. Uses least median of squares instead of regular least squares.

• **Step 2.** Global gradient-based method to improve the flow-field coherence. Minimizes a global energy function $E = \sum (E_B(V_i) + E_S(V_i))$ where $E_B$ is the brightness difference and $E_S$ is the smoothness at flow vector $V_i$.

• **Step 3.** Global matching that minimizes energy by a greedy approach. Visits each pixel and updates it to be consistent with neighbors, iteratively.
TT: Translating Tree

150x150 (Barron 94)

|     | $e_\angle (^\circ)$ | $e_{|.|}$ (pix) | $\bar{e}$ (pix) |
|-----|---------------------|------------------|-----------------|
| BA  | 2.60                | 0.128            | 0.0724          |
| S3  | 0.248               | 0.0167           | 0.00984         |

$e$: error in pixels, cdf: cumulative distribution function for all pixels
DT: Diverging Tree

150x150 (Barron 94)

<table>
<thead>
<tr>
<th></th>
<th>$e_\angle$ (°)</th>
<th>$e_|$(pix)</th>
<th>$\bar{e}$(pix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
<td>6.36</td>
<td>0.182</td>
<td>0.114</td>
</tr>
<tr>
<td>S3</td>
<td>2.60</td>
<td>0.0813</td>
<td>0.0507</td>
</tr>
</tbody>
</table>
YOS: Yosemite Fly-Through

316x252 (Barron, cloud excluded)

|     | $e_\angle (\degree)$ | $e_{|\bullet|} (\text{pix})$ | $\bar{e} (\text{pix})$ |
|-----|----------------------|-----------------------------|-------------------------|
| BA  | 2.71                 | 0.185                       | 0.118                   |
| S3  | 1.92                 | 0.120                       | 0.0776                  |
TAXI: Hamburg Taxi

256x190, (Barron 94)
max speed 3.0 pix/frame

Ours

LMS

BA

Error map

Smoothness error
Traffic

512x512 (Nagel) max speed: 6.0 pix/frame

Ours

Error map

Smoothness error
FG: Flower Garden

360x240 (Black)
Max speed: 7pix/frame

BA

LMS

Ours

Error map

Smoothness error
Representing Moving Images with Layers

J. Y. Wang and E. H. Adelson
MIT Media Lab
Goal

- Represent moving images with sets of overlapping layers
- Layers are ordered in depth and occlude each other
- Velocity maps indicate how the layers are to be warped over time
Simple Domain: Gesture Recognition
More Complex: What are the layers?
Motion Analysis Example

2 separate layers shown as 2 affine models (lines);
The gaps show the occlusion.
Motion Estimation Steps


2. From the optical flow representation, determine a set of affine motions. Segment into regions with an affine motion within each region.
Results

Figure 11: (a) The optic flow from multi-scale gradient method. (b) Segmentation obtained by clustering optic flow into affine motion regions. (c) Segmentation from consistency checking by image warping. Representing moving images with layers.

Figure 12: The layers corresponding to the tree, the flower bed, and the house shown in figures (a-c), respectively. The affine flow field for each layer is superimposed.
Results

Figure 13: Frames 0, 15, and 30 as reconstructed from the layered representation shown in figures (a-c), respectively.

Figure 14: The sequence reconstructed without the tree layer shown in figures (a-c), respectively.
Results

Figure 15: Frames 0, 15 and 30, of MPEG Calendar sequence shown in figures (a-c), respectively.

Figure 16: The layers corresponding to the ball, the train, and the background shown in figures (a-c), respectively.
Summary

• Major contributions from Lucas, Tomasi, Kanade
  – Tracking feature points
  – Optical flow
  – Stereo
  – Structure from motion

• Key ideas
  – By assuming brightness constancy, truncated Taylor expansion leads to simple and fast patch matching across frames
  – Coarse-to-fine registration
  – Global approach by former EE student Ming Ye
  – Motion layers methodology by Wang and Adelson
Back to the Homework

• For HW 6, you will implement optical flow!
• In particular, you will implement the Lucas-Kanade optical flow finder to find the optical flow between two image frames.
Homework 6
Optical Flow
Motion
Overall idea

• We’ll use Lucas-Kanade’s equation to find the optical flow.

• We'll need spatial and temporal gradient information for the flow equations.

• We'll be calculating structure matrices again, so we need to do aggregated sums over regions of the image.
  - Optical flow has to run on video, so it needs to be fast! we'll use integral images to simulate smoothing with a box filter instead of smoothing with a Gaussian filter.

• We’ll calculate velocity from spatial and temporal gradient information and use that to draw the motion lines.
1. Integral Image

- The **Integral Image** (or Summed Area Table) is used as a quick and effective way of calculating the sum of values (pixel values) or calculating the average intensity in a given image.

- When creating an Integral Image, if we go to any point \((x,y)\), the corresponding Integral Image value is the sum of all the pixel values **above**, to the **left** and of course including the **original** pixel value of \((x,y)\) itself.

\[
s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1)
\]
\[ s(x,y) = i(x,y) + s(x-1,y) + s(x, y-1) - s(x-1, y-1) \]

**Step 1**

**Image**

\[
\begin{array}{c|c}
5 & 2 \\
\hline
3 & i(x,y) \\
\hline
\end{array}
\]

**Summed Area Table**

\[
\begin{array}{c|c}
s(x-1,y-1) & s(x,y-1) \\
\hline
s(x-1,y) & s(x,y) \\
\hline
\end{array}
\]

**Step 2**

**Image**

\[
\begin{array}{c|c}
5 & 2 \\
\hline
3 & i(x,y) \\
\hline
\end{array}
\]

**Summed Area Table**

\[
\begin{array}{c|c}
s(x-1,y-1) & s(x,y-1) \\
\hline
s(x-1,y) & 7 \\
\hline
s(x,y) & s(x,y) \\
\hline
\end{array}
\]

**Step 3**

**Image**

\[
\begin{array}{c|c}
5 & 2 \\
\hline
3 & i(x,y) \\
\hline
\end{array}
\]

**Summed Area Table**

\[
\begin{array}{c|c}
s(x-1,y-1) & s(x,y-1) \\
\hline
s(x-1,y) & 7 \\
\hline
s(x,y) & 16 \\
\hline
\end{array}
\]
Calculate average intensity

How to calculate area in original image, using the corresponding integral image:

Original:
Area = 5 + 2 + 3 + 6 = 16

Integral:
Area (in original image)
= [S(D) – S(C)] – [S(B) – S(A)]
= (64 – 32) – (32 – 16) = 16
### Calculate average intensity

**Original Image**

<table>
<thead>
<tr>
<th>5</th>
<th>4</th>
<th>3</th>
<th>8</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

**Integral Image**

<table>
<thead>
<tr>
<th>5</th>
<th>9</th>
<th>12</th>
<th>20</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>21</td>
<td>25</td>
<td>35</td>
<td>44</td>
</tr>
<tr>
<td>17</td>
<td>36</td>
<td>40</td>
<td>55</td>
<td>71</td>
</tr>
<tr>
<td>24</td>
<td>46</td>
<td>56</td>
<td>76</td>
<td>101</td>
</tr>
<tr>
<td>25</td>
<td>49</td>
<td>61</td>
<td>89</td>
<td>117</td>
</tr>
</tbody>
</table>

Total of **9** operations.
- \(9 + 1 + 2 + 6 + 0 + 5 + 3 + 6 + 5 = 37\)
- \(\frac{37}{9} = 4.11\)

Total of **4** operations.
- \((76 - 20) - (24 - 5) = 37\)
- \(\frac{37}{9} = 4.11\)
TODO #1: Integral Image

• Don’t forget to git pull first. There are a couple of modified images and libraries.

• Fill in image make_integral_image(image im)

  o This function makes an integral image or summed area table from an image.
  o image im: image to process
  o returns: image I such that $I[x, y] = \sum_{i \leq x, j \leq y} im[i, j]$
TODO #2: Smoothing using integral images

• Fill in `image box_filter_image(image im, int s)` so that every pixel in the output is the average of pixels in a given window size `s`.

• Note that you must call your `make_integral_image()` in this function.

• Be careful, this is not the your old `make_box_filter()` from your other homework. It is using the integral image, and a smooth window size.
TODO #3: Lucas-Kanade optical flow

- We'll be implementing optical flow. We'll use a structure matrix but this time with temporal information as well. The equation we'll use is:

\[
\begin{bmatrix}
  V_x \\
  V_y
\end{bmatrix} = 
\begin{bmatrix}
  \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\
  \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2
\end{bmatrix}^{-1} 
\begin{bmatrix}
  -\sum_i I_x(q_i)I_t(q_i) \\
  -\sum_i I_y(q_i)I_t(q_i)
\end{bmatrix}
\]

Velocity Structure Matrix Time Matrix
TODO #3.1: Time-structure matrix

• We'll need spatial and temporal gradient information for the flow equations.

• Calculate a time-structure matrix.
  – Spatial gradients can be calculated as normal.
  – The time gradient can be calculated as the difference between the previous image and the next image in a sequence.
    • \( l_t = [\text{current image}] - [\text{previous image}] \)
TODO #3.1: Time-structure matrix

Calculate the time-structure matrix of an image pair:

- Fill in `image time_structure_matrix(image im, image prev, int s)`.

  - `image im`: the input image.
  - `image prev`: the previous image in sequence.
  - `int s`: window size for smoothing.
  - `im` and `prev` to grayscale (given in the code).
  - Hint: use `sub_image` to subtract `im` and `prev`.
  - Calculate gradients and structure matrix and smooth (hint: use your `gx` and `gy` functions from HW2)

  - ...next slide: return
**TODO #3.1: Time-structure matrix**

Calculate the time-structure matrix of an image pair:

- Fill in `image time_structure_matrix(image im, image prev, int s)`.

  - returns: structure matrix which has 5 channels:
    - 1\textsuperscript{st} channel is $l_x l_x$
    - 2\textsuperscript{nd} channel is $l_y l_y$
    - 3\textsuperscript{rd} channel is $l_x l_y$
    - 4\textsuperscript{th} channel is $l_x l_t$
    - 5\textsuperscript{th} channel is $l_y l_t$

  - Each channel is a vector with the structure of an image.
  - Use `make_box_filter()` to smooth.
**TODO #3.2: Calculating velocity from the time-structure matrix**

Calculate the velocity given a time-structure image

- Fill in `image_velocity_image(image S, int stride)`
  - Image S is the output of `time_structure_matrix` which you already summed and smooth.

- For each pixel, fill in the matrix `M`, invert it, and use it to calculate the velocity.

\[
M = \begin{bmatrix}
I_x(q_i)^2 & I_x(q_i)I_y(q_i) \\
I_y(q_i)I_x(q_i) & I_y(q_i)^2
\end{bmatrix}
\]

\[
\begin{pmatrix}
v_x \\
v_y
\end{pmatrix} = -M^{-1} \ast \begin{pmatrix} I_x(t) \\ I_y(t) \end{pmatrix}
\]
Draw motion with optical flow

optical_flow_images() will call your time_structure_matrix() and velocity_image(). Then draw_flow() will draw lines of motion on the image.

Try calculating the optical flow between two dog images using tryhw6.py.

```python
a = load_image("data/dog_a.jpg")
b = load_image("data/dog_b.jpg")
flow = optical_flow_images(b, a, 15, 8)
draw_flow(a, flow, 8)
save_image(a, "lines")
```
Have fun!