Optical flow
Combination of slides from Rick Szeliski, Steve Seitz, Alyosha Efros and Bill Freeman

Motion without movement
Bill Freeman, Ted Adelson and David Heeger, MIT 1991

A linear combination of quadrature-phase filters can advance the local phase

Convolved with an image, the image data now modulates the local amplitude. People misattribute the phase advance to translation.

(Steerable filters allow synthesizing motion in arbitrary directions.)

http://www.cs.yorku.ca/~kosta/Motion_Without_Movement/Motion_Without_Movement.html

Konstantinos G. Derpanis
How do we align two images automatically? Two broad approaches:

- **Feature-based alignment**
  - Find a few matching features in both images
  - Compute alignment

- **Direct (pixel-based) alignment**
  - Search for alignment where most pixels agree

### Direct Alignment

The simplest approach is a brute force search (hw1)

- Need to define image matching function
  - SSD, Normalized Correlation, edge matching, etc.
- Search over all parameters within a reasonable range:

  e.g. for translation:
  
  ```
  for tx=x0:step:x1, 
  for ty=y0:step:y1, 
  compare image1(x,y) to image2(x+tx,y+ty) 
  end; 
  end; 
  ```

  Need to pick correct `x0, x1` and `step`
  - What happens if `step` is too large?

### Direct Alignment (brute force)

What if we want to search for more complicated transformation, e.g. homography?

\[
\begin{bmatrix}
wx' \\
wy' \\
w \\
\end{bmatrix}
= \begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1 \\
\end{bmatrix}
\]

for `a=a0:astep:a1`,
for `b=b0:bstep:b1`,
for `c=c0:astep:c1`,
for `d=d0:bstep:d1`,
for `e=e0:astep:e1`,
for `f=f0:astep:f1`,
for `g=g0:astep:g1`,
for `h=h0:astep:h1`,
for `i=i0:astep:i1`,

```plaintext
for a=a0:astep:a1, 
for b=b0:bstep:b1, 
for c=c0:astep:c1, 
for d=d0:bstep:d1, 
for e=e0:astep:e1, 
for f=f0:astep:f1, 
for g=g0:astep:g1, 
for h=h0:astep:h1, 
for i=i0:astep:i1, 
compare image1 to H(image2) 
end; 
end; 
end; 
end; 
end; 
end; 
end; 
end; 
```
Why estimate motion?
Lots of uses
• Track object behavior
• Correct for camera jitter (stabilization)
• Align images (mosaics)
• 3D shape reconstruction
• Special effects

Problem definition: optical flow

How to estimate pixel motion from image H to image I?
• Solve pixel correspondence problem
  - given a pixel in H, look for near 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

Optical flow constraints (grayscale images)

Optical flow equation
Combining these two equations
\[ \Omega = I(x, y) - H(x, y) \]
\[ \approx I(x, y) + I_x u + I_y v - H(x, y) \]
\[ \approx (I(x, y) - H(x, y)) + I_x u + I_y v \]
\[ \approx I_x + I_y v = I_x + \nabla I \cdot (u, v) \]

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

\[ \Omega = I_x + \nabla I \cdot [u \ v] \]

Optical flow equation

Q: how many unknowns and equations per pixel?
2 unknowns, one equation

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

http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm
http://www.liv.ac.uk/~marcob/Trieste/barberpole.html

Optical flow equation

Q: 1 equation

2 unknowns, one equation

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
http://en.wikipedia.org/wiki/Barber%27s_pole

Aperture problem
Aperture problem

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, this gives us 25 equations per pixel!

RGB version

- 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, this gives us 25 equations per pixel!

Note that RGB is not enough to disambiguate because R, G & B are correlated
Just provides better gradient

Lukas-Kanade flow

- Prob: we have more equations than unknowns
- Solution: solve least squares problem
  - minimum least squares solution given by solution (in d) of:

\[
\begin{align*}
0 &= l_1(p_1)[0,1,2] + \nabla f(p_1)[0,1,2] \cdot [u \ v] \\
&\vdots \\
0 &= l_{25}(p_{25})[0,1,2] + \nabla f(p_{25})[0,1,2] \cdot [u \ v] \\
\end{align*}
\]

The gradient constraint:

Define a line in the (u,v) space

Combining Local Constraints

Aperture Problem and Normal Flow

The gradient constraint:

Normal Flow:

\[
u = -\frac{I_y}{\nabla I} \quad \text{and} \quad v = -\frac{I_x}{\nabla I}
\]
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}
= -\frac{1}{\sum I_x I_y}
\begin{bmatrix}
\sum I_x I_y \\
\sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
I_x \\
I_y
\end{bmatrix}
= \nabla I(\nabla I)^T

When is This Solvable?

- \(A^T A\) 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
  - \(\lambda_1 \lambda_2\) should not be too large (\(\lambda_1\) larger eigenvalue)

\(A^T A\) is solvable when there is no aperture problem

\[
A^T A = \begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
= \sum |I_x| I_x |I_y| I_y = \sum \nabla I(\nabla I)^T

Local Patch Analysis

Edge

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

Low texture region

- gradients have small magnitude
  - small \(\lambda_1\), small \(\lambda_2\)

High textured region

- gradients are different, large magnitudes
  - large \(\lambda_1\), large \(\lambda_2\)

Observation

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 later on when we do feature tracking...
Errors in Lukas-Kanade

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?

Iterative Refinement

Iterative Lukas-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

Optical Flow: Iterative Estimation

Initial guess: $d_0 = 0$
Estimate: $d_1 = d_0 + \hat{d}$

Estimate: $d_2 = d_1 + \hat{d}$

Estimate: $d_3 = d_2 + \hat{d}$

Estimate: $d_4 = d_3 + \hat{d}$
Optical Flow: Iterative Estimation

Some Implementation Issues:

- Warping is not easy (ensure that errors in warping are smaller than the estimate refinement).
- Warp one image, take derivatives of the other so you don’t need to re-compute the gradient after each iteration.
- Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity).

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?

Optical Flow: Aliasing

Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity. I.e., how do we know which correspondence is correct?

actual shift

estimated shift

nearest match is correct (no aliasing)

nearest match is incorrect (aliasing)

To overcome aliasing: coarse-to-fine estimation.

Optical Flow: Aliasing

Reduce the resolution!

Coarse-to-fine optical flow estimation

Gaussian pyramid of image H

Gaussian pyramid of image I

Coarse-to-fine optical flow estimation

image I

image H

Gaussian pyramid of image H

Gaussian pyramid of image I

Coarse-to-fine optical flow estimation

run iterative L-K

warp & upsample

run iterative L-K
Beyond Translation
So far, our patch can only translate in (u,v)
What about other motion models?
• rotation, affine, perspective

Same thing but need to add an appropriate Jacobian
See Szeliski’s survey of Panorama stitching

\[ A^T A = \sum_{i} J_i (\nabla I_i)^T J_i^T \]
\[ A^T b = -\sum_{i} J_i^T (\nabla I_i)^T \]

Block-based motion prediction
Break image up into square blocks
Estimate translation for each block
Use this to predict next frame, code difference (MPEG-2)

Recap: Classes of Techniques

Feature-based methods (e.g. SIFT+Ransac+regression)
• Extract visual features (corners, textured areas) and track them over multiple frames
• Sparse motion fields, but possibly robust tracking
• Suitable especially when image motion is large (10s of pixels)

Direct-methods (e.g. optical flow)
• Directly recover image motion from spatio-temporal image brightness variations
• Global motion parameters directly recovered without an intermediate feature motion calculation
• Dense motion fields, but more sensitive to appearance variations
• Suitable for video and when image motion is small (< 10 pixels)

Motion Magnification
(go to other slides…)

Retiming
http://www.realviz.com/retiming.htm