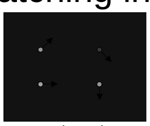
Announcements

- Quiz Thursday
- Quiz Review Tomorrow: AV Williams 4424, 4pm.
- Practice Quiz handout.

Matching

- Compare region of image to region of image.
  - We talked about this for stereo.
  - Important for motion.
    - Epipolar constraint unknown.
    - But motion small.
  - Recognition
    - Find object in image.
    - Recognize object.
- Today, simplest kind of matching. Intensities similar.

Matching in Motion: optical flow



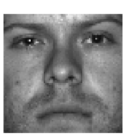
$H(x, y)$

$I(x, y)$

- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I
- How to estimate pixel motion from image H to image I?

Matching: Finding objects

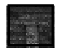
Matching: Identifying Objects



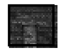
Matching: what to match

- Simplest: SSD with windows.
  - We talked about this for stereo as well:
  - Windows needed because pixels not informative enough? (More on this later).

Comparing Windows:



?
=



f g

$$SSD = \sum_{[i,j] \in R} (f(i,j) - g(i,j))^2$$

$$C_{fg} = \sum_{[i,j] \in R} f(i,j)g(i,j)$$

Most popular

(Camps)

Window size

W = 3 W = 20

- Effect of window size

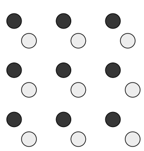
Better results with *adaptive window*

- T. Kanade and M. Okutomi, *A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment*,, Proc. International Conference on Robotics and Automation, 1991.
- D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. International Journal of Computer Vision, 28(2):155-174, July 1998

(Seitz)

Subpixel SSD

- When motion is a few pixels or less, motion of an integer no. of pixels can be insufficient.



Bilinear Interpolation

To compare pixels that are not at integer grid points, we resample the image.

Assume image is locally bilinear.

I(x,y) = ax + by + cxy + d = 0. Given the value of the image at four points: I(x,y), I(x+1,y), I(x,y+1), I(x+1,y+1) we can solve for a,b,c,d linearly. Then, for any u between x and x+1, for any v between y and y+1, we use this equation to find I(u,v).

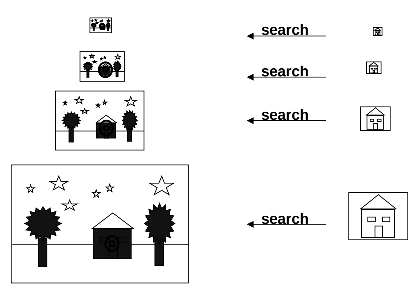
Matching: How to Match Efficiently

- Baseline approach: try everything.

$$\arg\min_{u,v} \sum \left(W(x,y) - I(x+u, y+v)\right)^2$$

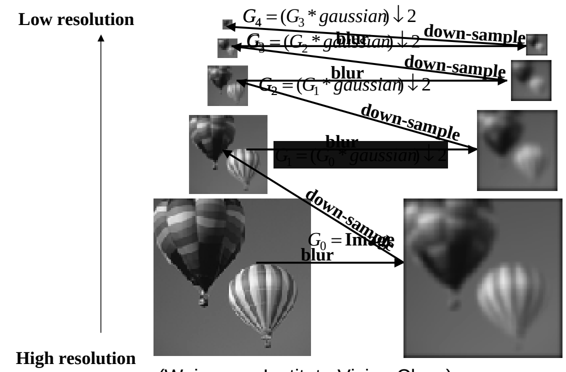
  - Could range over whole image.
  - Or only over a small displacement.

Matching: Multiscale



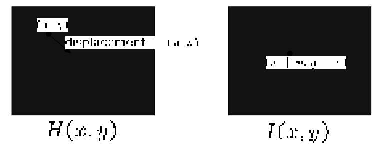
(Weizmann Institute Vision Class)

The Gaussian Pyramid



(Weizmann Institute Vision Class)

When motion is small: Optical Flow



- Small motion: (u and v are less than 1 pixel)
- Brute force not possible
  - suppose we take the Taylor series expansion of I:

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

(Seitz)

Optical flow equation

- Combining these two equations

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$\begin{aligned} 0 &= I(x+u, y+v) - H(x,y) \\ &\approx I(x,y) + I_x u + I_y v - H(x,y) \\ &\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\ &\approx I_t + I_x u + I_y v \\ &\approx I_t + \nabla I \cdot [u \; v] \end{aligned}$$

- In the limit as u and v go to zero, this becomes exact

$$0 = I_t + \nabla I \cdot [\tfrac{\partial x}{\partial t} \; \tfrac{\partial y}{\partial t}]$$

(Seitz)

Optical flow equation

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

- Q: how many unknowns and equations per pixel?
- 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

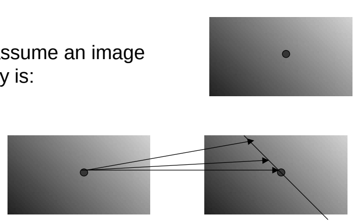
(Seitz)

This explains the Barber Pole illusion
http://www.sandlotscience.com/Ambiguous/barberpole.htm

First Order Approximation

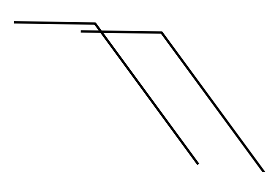
When we assume that: $\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$

We assume an image locally is:



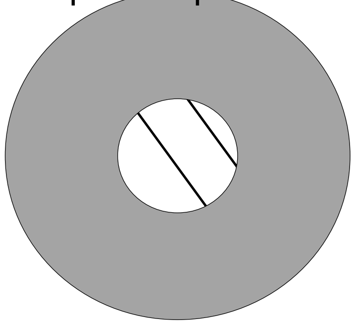
(Seitz)

Aperture problem



(Seitz)

Aperture problem



(Seitz)

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, that gives us 25 equations per pixel! $0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u\ v]$

$$\underbrace{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}}_{\substack{A \\ 25\times2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\substack{d \\ 2\times1}} = - \underbrace{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}_{\substack{b \\ 25\times1}}$$

(Seitz)

Lukas-Kanade flow

$$\underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

- We have more equations than unknowns: solve least squares problem. This is given by:

$$\underset{2\times2}{(A^TA)}\ \underset{2\times1}{d} = \underset{2\times1}{A^Tb}$$

$$\underbrace{\begin{bmatrix} \sum I_xI_x & \sum I_xI_y \\ \sum I_xI_y & \sum I_yI_y \end{bmatrix}}_{A^TA} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_xI_t \\ \sum I_yI_t \end{bmatrix}}_{A^Tb}$$

– Summations over all pixels in the KxK window

– Does A^TA look familiar?

(Seitz)

Conditions for solvability

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

$$\underbrace{\begin{bmatrix} \sum I_xI_x & \sum I_xI_y \\ \sum I_xI_y & \sum I_yI_y \end{bmatrix}}_{A^TA} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_xI_t \\ \sum I_yI_t \end{bmatrix}}_{A^Tb}$$

When is This Solvable?

- $\mathbf{A^TA}$ should be invertible
- $\mathbf{A^TA}$ should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^TA}$ should not be too small
- $\mathbf{A^TA}$ should be well-conditioned
  - λ_1/λ_2 should not be too large (λ_1 = larger eigenvalue)

(Seitz)

Does this seem familiar? Formula for Finding Corners

We look at matrix:

Sum over a small region, the hypothetical corner

Gradient with respect to x, times gradient with respect to y

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

WHY THIS?

First, consider case where:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means all gradients in neighborhood are:

(k,0) or (0, c) or (0, 0) (or off-diagonals cancel).

What is region like if:

1. $\lambda1 = 0$?
2. $\lambda2 = 0$?
3. $\lambda1 = 0$ and $\lambda2 = 0$?
4. $\lambda1 > 0$ and $\lambda2 > 0$?

General Case:

From Singular Value Decomposition it follows that since C is symmetric:

$$C = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

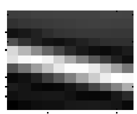
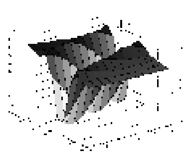
where R is a rotation matrix.

So every case is like one on last slide.

So, corners are the things we can track

- Corners are when $\lambda 1$, $\lambda 2$ are big; this is also when Lucas-Kanade works.
- Corners are regions with two different directions of gradient (at least).
- Aperture problem disappears at corners.
- At corners, 1st order approximation fails.

Edge

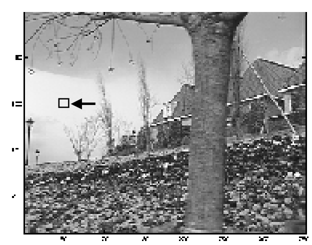
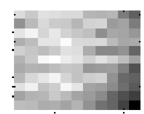
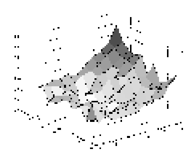




$\sum\sum \nabla I (\nabla I)^T$

- large gradients, all the same
- large λ_1, small λ_2

(Seitz)

Low texture region

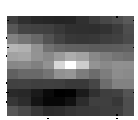
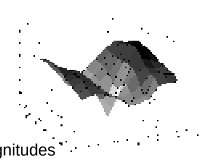




$\sum\sum \nabla I (\nabla I)^T$

- gradients have small magnitude
- small λ_1, small λ_2

(Seitz)

High textured region

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

- gradients are different, large magnitudes
- large λ_1, large λ_2

(Seitz)

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...

(Seitz)

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?

(Seitz)

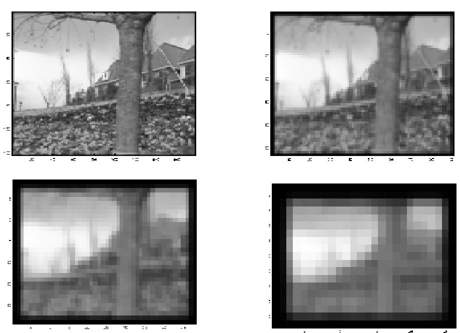
Iterative Refinement

- 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 bilinear interpolation*
     - Repeat until convergence

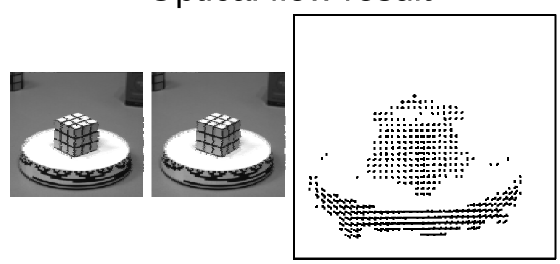
(Seitz)

If Motion Larger: Reduce the resolution

(Seitz)



Optical flow result



Dewey morph

(Seitz)

Tracking features over many Frames

- Compute optical flow for that feature for each consecutive H, I
- When will this go wrong?
  – Occlusions—feature may disappear
    - need to delete, add new features
  – Changes in shape, orientation
    - allow the feature to deform
  – Changes in color
  – Large motions
    - will pyramid techniques work for feature tracking?

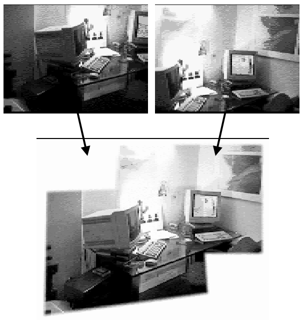
(Seitz)

Applications:

- MPEG—application of feature tracking
  – http://www.pixeltools.com/pixweb2.html

(Seitz)

Image alignment



- Goal: estimate single (u,v) translation for entire image
  - Easier subcase: solvable by pyramid-based Lukas-Kanade

(Seitz)

Summary

- Matching: find translation of region to minimize SSD.
  - Works well for small motion.
  - Works pretty well for recognition sometimes.
- Need good algorithms.
  - Brute force.
  - Lucas-Kanade for small motion.
  - Multiscale.
- Aperture problem: solve using corners.
  - Other solutions use normal flow.