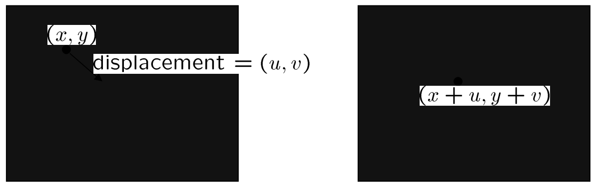
When motion is small: Optical Flow



$H(x,y)$ $\qquad$ $I(x,y)$

- Small motion: (u and v are less than 1 pixel)
  - *H(x,y) = I(x+u,y+v)*
- 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: $I_x = \frac{\partial I}{\partial x}$

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

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

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

(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

Let's look at an example of this. Suppose we have an image in which H(x,y) = y. That is, the image will look like:

11111111111111

22222222222222

33333333333333

And suppose there is optical flow of (1,1). The new image will look like:

-1111111111111

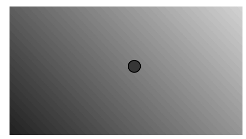
-2222222222222

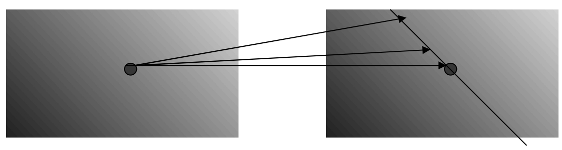
I(3,3) = 2. H(3,3) = 3. So I_t(3,3) = -1. GRAD I(3,3) = (0,1). So our constraint equation will be: 0 = -1 + <(0,1), (u,v)>, which is 1 = v. We recover the v component of the optical flow, but not the u component. This is the aperture problem.

First Order Approximation

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

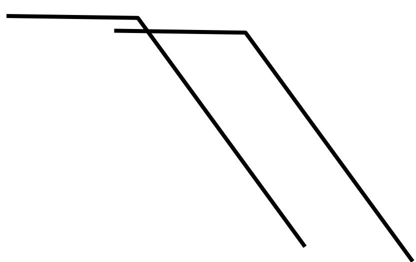
We assume an image locally is:





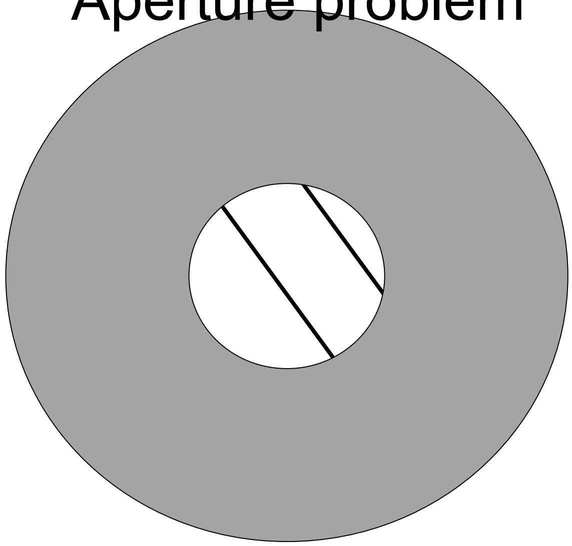
(Seitz)

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! $0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \ v]$

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

(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:

$$\underset{2\times2}{(A^T A)}\ \underset{2\times1}{d} = \underset{2\times1}{A^T b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = -\underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

  – Summations over all pixels in the KxK window
  – Does $A^T A$ look familiar?

(Seitz)

Let's look at an example of this. Suppose we have an image with a corner.

1111111111		-----------------
1222222222	And this translates down and to the right:	-1111111111
1233333333		-1222222222
1234444444		-1233333333

Let's compute I_t for the whole second image:

----------	Ix = ----------	Iy = -------------
0-1-1-1-1-1	--00000	--------------
-1-1-1-1-1-1	--.50000	-0-.5-1-1-1-1-1-1
-1-1-1-1-1-1-	--1.5000	-00-.5-1-1-1-1-1

Then the equations we get have the form:

(.5,-.5)*(u,v) = 1, (1,0)*(u,v) = 1, (0,-1)(u,v) = 1.

Together, these lead to a solution that u = 1, v = -1.

Conditions for solvability

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

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

When is This Solvable?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be too small
- $\mathbf{A^T A}$ 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

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

Matrix is symmetric

WHY THIS?

First, consider case where:

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

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

General Case:

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

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

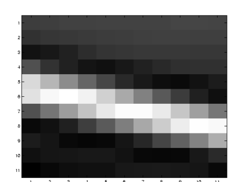
where R is a rotation matrix.

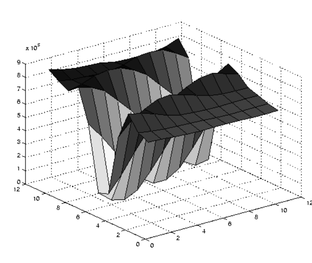
So every case is like one on last slide.

So, corners are the things we can track

- Corners are when λ1, λ2 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



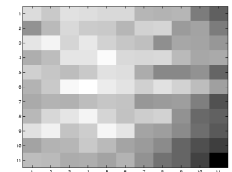


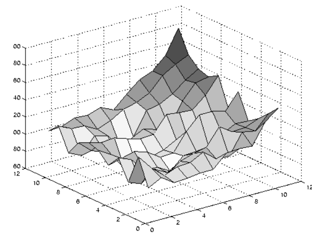
$$\sum \nabla I (\nabla I)^T$$

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

(Seitz)

Low texture region





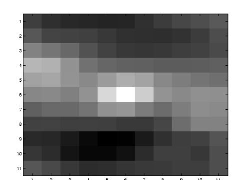
$$\sum \nabla I (\nabla I)^T$$

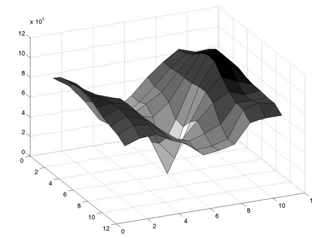
– gradients have small magnitude

– small λ_1, small λ_2

(Seitz)

High textured region





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

– 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)

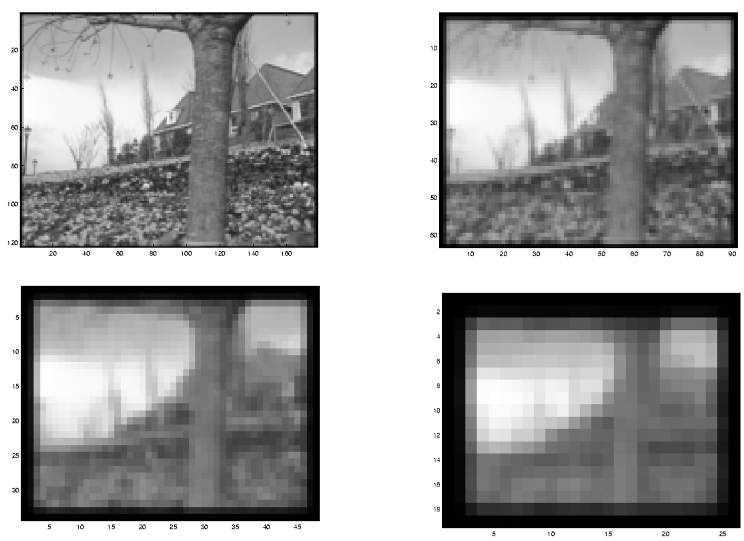
Iterative Refinement

- 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

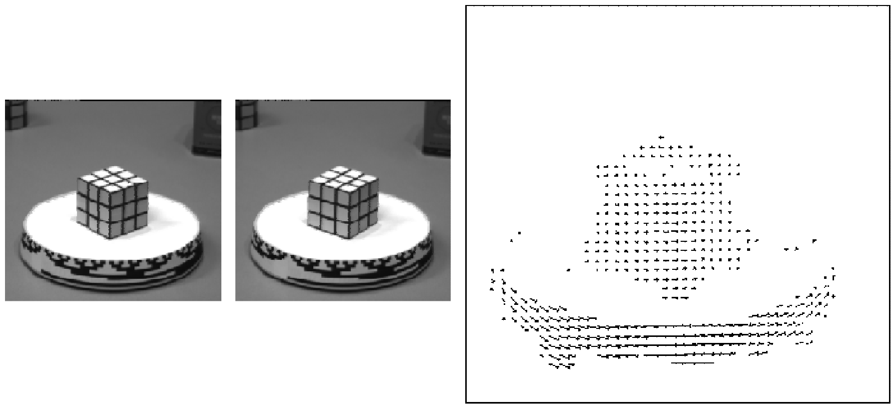
(Seitz)

If Motion Larger: Reduce the resolution

(Seitz)



Optical flow result



Dewey morph

(Seitz)

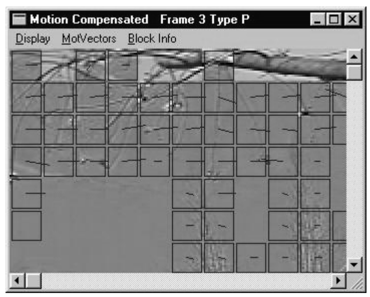
Tracking features over many Frames

- 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)

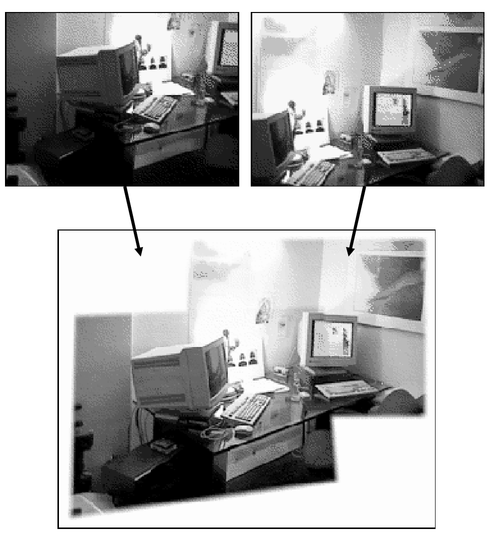
Applications:

- 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.