Feature-Based Recognition

- Till now we've been looking at methods that require dense correspondences.
  - May be assumed, eg., Eigenfaces, LLE…
  - May be searched for, eg., Shape Context
  - Correspondence comes from smooth transformation
- Feature-based applied to wider range of objects
  - Correspondences may be sparse
  - Objects may not be related by smooth transformation (eg., bag of words).

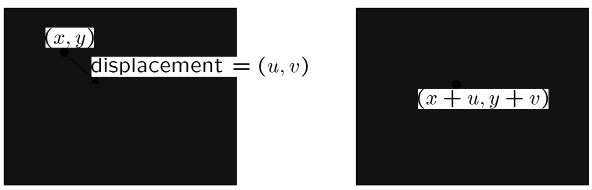
Features

- Choice of features
  - Distinctive points
  - Stability over transformations
  - Corners
    - Stable over Euclidean transformations
  - Scale space
- Descriptors
  - Largely based on gradient direction
  - Histograms popular

Corners

- Intuitively, should be locally unique
- One way to get at that is through motion.
  – A point is different from its neighborhood iff we can accurately track it with small motion

When motion is small: Optical Flow



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

- Small motion: (u and v are less than 1 pixel)
  – *H(x,y) = I(x+u,y+v)*

Brightness Change Constraint Equation

- 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

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

(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

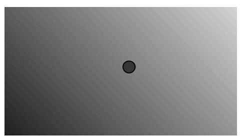
-2222222222222

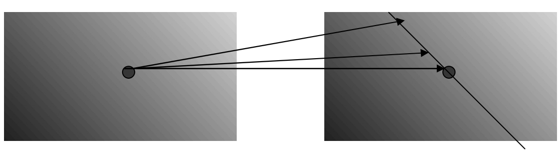
I(3,3) = 2. H(3,3) = 3. So $I_t(3,3) = -1$. GRAD I(3,3) = (0,1). So our constraint equation will be: 0 = -1 + <(0,1), (u,v)>, which is 1 = v. We recover the v component of the optical flow, but not the u component. This is the aperture problem.

First Order Approximation

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

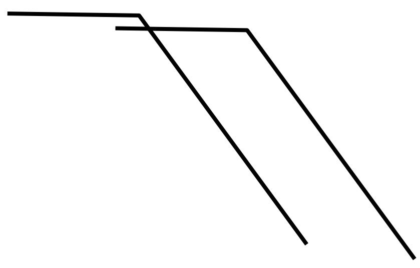
We assume an image locally is:





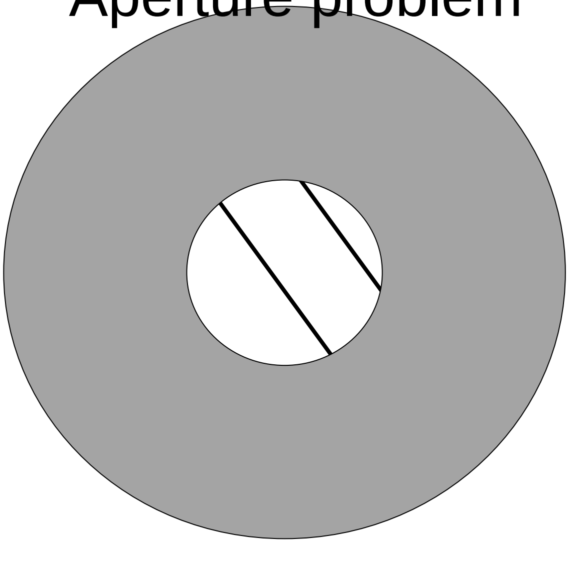
(Seitz)

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

$\underset{25\times2}{A} \ \underset{2\times1}{d} = \underset{25\times1}{b}$ $\longrightarrow$ 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

(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?

- $\mathbf{A^TA}$ 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)

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:

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

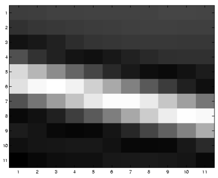
where R is a rotation matrix.

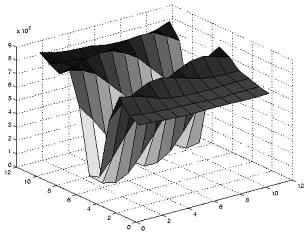
So every case is like one on last slide.

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, 1^{st} order approximation fails.

Edge



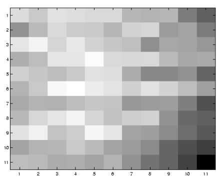


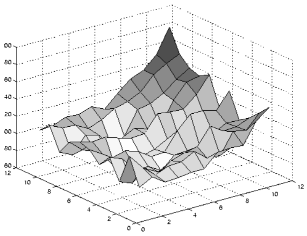
$$\sum \nabla I (\nabla I)^T$$

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

(Seitz)

Low texture region



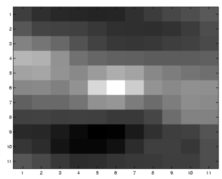


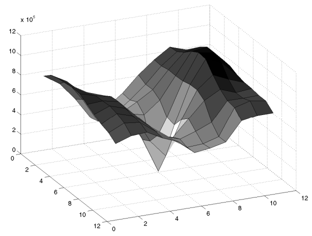
$$\sum \nabla I (\nabla I)^T$$

- 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)

Scale-Invariant Features

- Notice that we will detect same corners even as image translates and rotates.
- Scaling can effect corners
  - Region of support is scale dependent.
  - Need regions that are scale independent.

Scale-Invariant Blobs: Lindeberg

- Gaussian scale space
  - Repeated smoothing with Gaussian.
  - Linear, shift invariant
  - Doesn't introduce new features (eg., extrema).
- Blob defined as scale-normalized Laplacian of Gaussian (Mexican hat)
- Select local extrema in space and scale

$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$ (Gaussian, with scale σ)

$\mathrm{L}(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ (Image at scale σ)

$\nabla^2 L = L_{xx} + L_{yy}$ (Laplacian of Gaussian)

$\nabla^2 L_{norm}(x, y, \sigma) = \sigma^2 (L_{xx} + L_{yy})$ (Scale normalized LoG)

Then find locations where this is a local extrema with respect to x, y, σ.

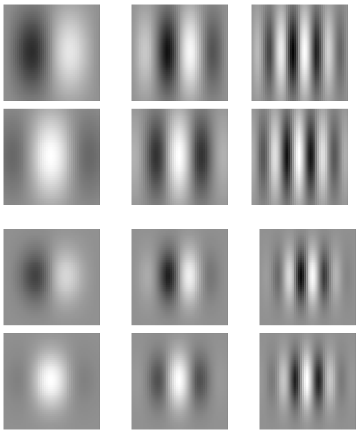
Maximally Stable Extremal Regions

- Based on isoluminant contours
  - Ie, boundaries of regions that we get by thresholding image with a fixed threshold.
  - These are *extremal* regions.
  - Notice that direction of image gradient is orthogonal to isoluminant contour.
    - So, if direction of image gradient doesn't change, isoluminant contours don't change.
  - Topology of contours doesn't change with any continuous transformation, so their structure is preserved under projective transformations.
- *Maximally Stable* regions are the ones that change least when the threshold changes
- This is related to corner detection. Good regions have high gradients in all directions.

Descriptors: Gabor Jets

- 1D Gabor filter is sine/cosine times Gaussian
  - Precursor to wavelet; localized in frequency and space.
  - Scale can vary with Gaussian, frequency with harmonic.
  - Add constant so they integrate to 0.
- 2D Gabors are oriented. 1D harmonic times Gaussian.
- Gabor Jet
  - Apply Gabors at different orientations and scales (varying frequency with scale).
  - Normalize the vector that contains all the outputs.
- Invariant to additive and multiplicative intensity changes
  - Filters integrate to zero, so invariant to additive changes.
  - Normalization removes effects of multiplicative changes.

Gabor Filters



Gabor filters at different scales and spatial frequencies

top row shows anti-symmetric (or odd) filters, bottom row the symmetric (or even) filters.

$$\cos(k_x x + k_y y) \exp - \left\{ \frac{x^2 + y^2}{2\sigma^2} \right\}$$

SIFT

- Divide region into windows
- Compute histogram of gradient orientations within each window.
  – Weight by distance to keypoint using Gaussian
  – Weight by gradient magnitude
- Many subtle optimizations
  – Eg., antialiasing when building histogram
  – Parameters carefully optimized
- Insensitive to small image shifts.

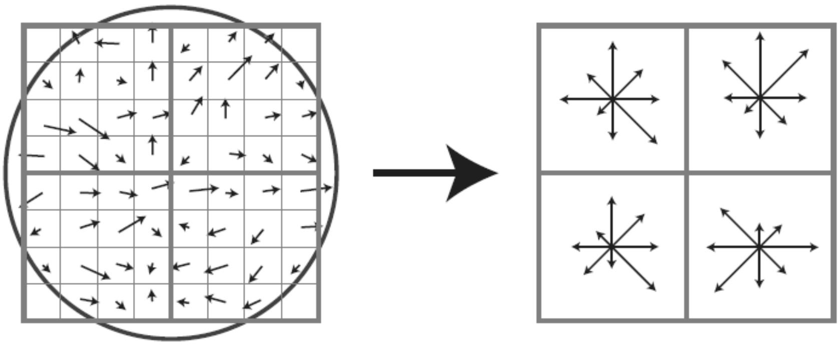


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

(from Lowe, IJCV)

HOG (Histograms of Oriented Gradients)

- Similar in spirit to SIFT
- Histograms computed on a dense, overlapping grid.
- Contrast normalization is performed within regions.