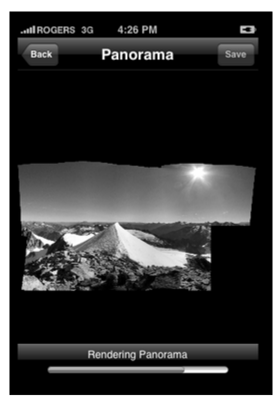
Blob Detection

Image Matching

- Matching whole images
  - For alignment, eg., mosaicing
- Matching small regions
  - Eg., for stereo
- Matching Objects



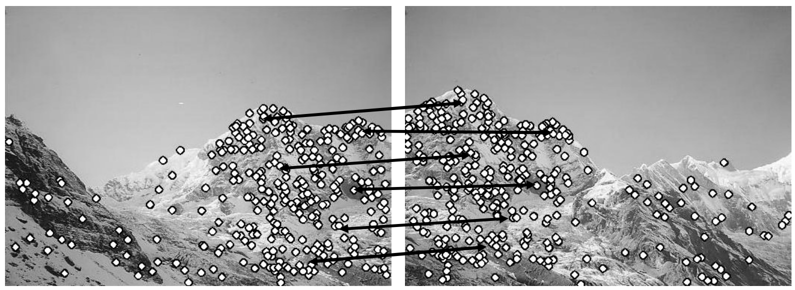
Features-based Matching

1. Find distinctive features
   - Corners, blobs, MSER…
2. Describe region around feature
   - Intensities, SIFT, …
3. Compare features to find matches
   - Local matches: Histogram comparison, normalized correlation…
   - Global matches: RANSAC
4. Use these matches
   - Find rigid alignment of images, compute disparity from each match, compute similarity score.

Example: Mosaicing

(Slides from Lazebnik)

Why extract features?



Step 1: extract features
Step 2: match features

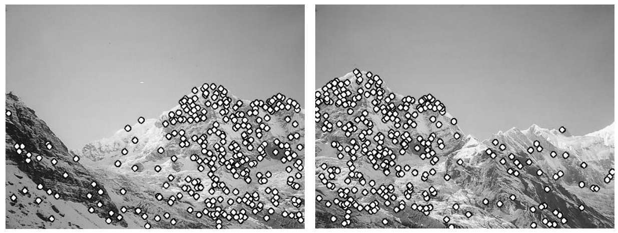
(Slides from Lazebnik)

Why extract features?

Step 1: extract features
Step 2: match features
Step 3: align images

(Slides from Lazebnik)

Characteristics of good features



- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

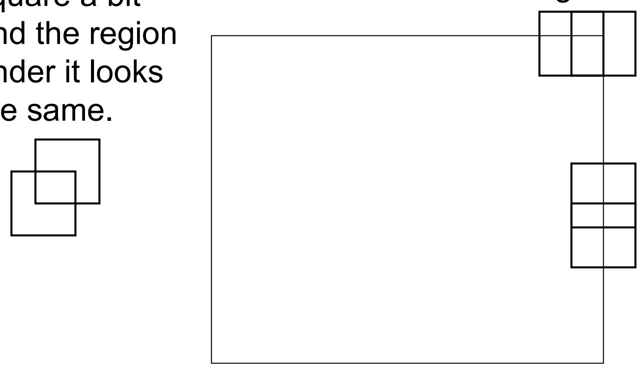
(Slides from Lazebnik)

Distinctive Features

- A point is distinctive when it looks different from its neighbors.
- A blob also looks different from neighbors at different scales.

Move this square a bit and the region under it looks the same.

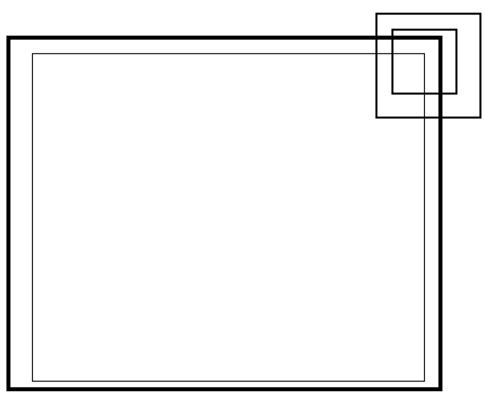
But at a corner, moving a little changes the scenery



Same here.

A corner (in computer vision) is technically defined as a region where the intensity pattern changes a lot with any small translatiion.

However, a corner isn’t well localized in scale.

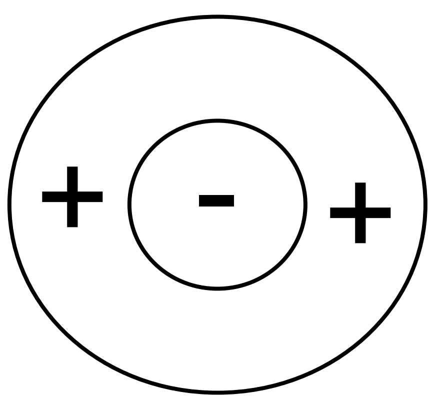


For that we need blobs

What is a good blob detector?

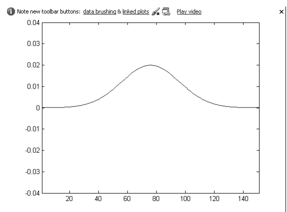
- A filter that has versions at multiple scales.
- The biggest response should be when the filter has the same location and scale as the blob.

Center-Surround Filter

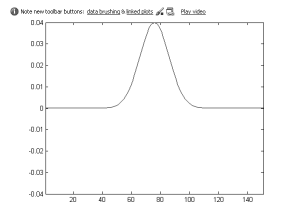


- When does this have biggest response?
  - When inside is as dark as possible
  - And outside is as light as possible.
  - Ie, a dark spot.
  - Note, this locates position and scale.
- Similar filters are in animals (eg., frog).

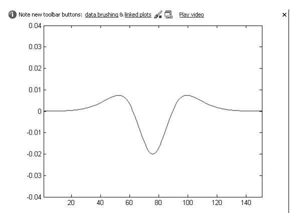
Difference of Gaussian



-



=



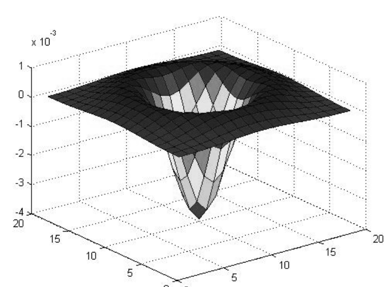
Basic Algorithm

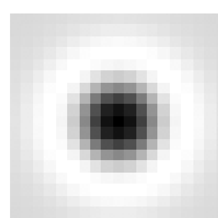
- Filter with Gaussian at different scales
  - This is done by just repeatedly filtering with the same Gaussian.
- Subtract image filtered at one scale with image filtered at previous scale.
- Look for local extrema
  - A pixel is bigger (smaller) than all eight neighbors, and all nine neighboring pixels at neighboring scales.

Which Scales

- More scales can produce greater accuracy.
- But also more expense.
- We are taking a derivative, so need to be careful about denominator.
  – It turns out that we should increase scale multiplicatively. Sigma, k*sigma, k*k*sigma….
  – Sigma = 1.6 produces reasonable results.
  – k = cuberoot(2). (These values are heuristic).

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D





$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

(Slides from Lazebnik)

Efficient implementation

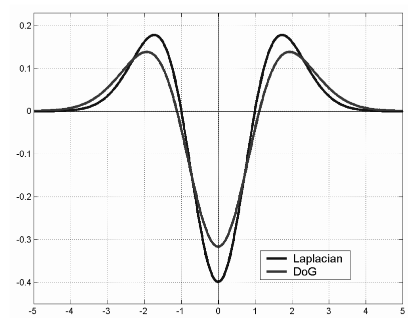
- Approximating the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



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

Formula for Finding Corners

We look at matrix:

Sum over a small region, the hypothetical corner

Derivative with respect to x, times deriv. 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 $\lambda 1$, $\lambda 2$ 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, 1^{st} order approximation fails.