Why extract features?
• Motivation: panorama stitching
  – We have two images – how do we combine them?

Step 1: extract features
Step 2: match features

Feature Point Extraction
CSE252b
Lecture 19
6/4/2013

Why extract features?
• Motivation: panorama stitching
  – We have two images – how do we combine them?

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

Some Junctions (Corners)
T-Junction
L-junction

Introduction
Wide baseline matching
Image Filtering

Convolution

**Convolution:** \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-m}^{m} \sum_{k=-m}^{m} K(h, k) I(i-h, j-k)
\]

Linear functions

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the “convolution kernel”.

What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.

Convolution:

**Image (I)**

**Kernel (K)**

Note: Typically Kernel is relatively small in vision applications.

(From Bill Freeman)

(Freeman)
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[ R(i, j) = \sum_{h=-m/2}^{n/2} \sum_{k=-m/2}^{n/2} K(h, k) I(i-h, j-k) \]
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} K(h, k) I(i-h, j-k)
\]

Properties of convolution

Let \( f, g, h \) be images and \( \ast \) denote convolution

\[
f \ast g(x, y) = \iint f(x-u, y-v)g(u, v) \, du \, dv
\]

- Commutative: \( f \ast g = g \ast f \)
- Associative: \( f \ast (g \ast h) = (f \ast g) \ast h \)
- Linear: for scalars \( a \) & \( b \) and images \( f, g, h \)
  \[
  (af+bg) \ast h = af \ast h + bg \ast h
  \]
- Differentiation rule
  \[
  \frac{\partial}{\partial x} (f \ast g) = \frac{\partial f}{\partial x} \ast g = f \ast \frac{\partial g}{\partial x}
  \]

Averaging Filter

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.
Smoothing by Averaging

Kernel: \( \frac{e^{-\frac{x^2 + y^2}{2\sigma^2}}}{\sigma^2} \)

An Isotropic Gaussian

- The picture shows a smoothing kernel proportional to

\( \frac{e^{-\frac{x^2 + y^2}{2\sigma^2}}}{\sigma^2} \)

(which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian

Corner Detection

Edge is Where Change Occurs: 1-D

- Change is measured by derivative in 1D

  - Biggest change, derivative has maximum magnitude
  - Or 2nd derivative is zero.
Numerical Derivatives

Take Taylor series expansion of \( f(x) \) about \( x_0 \):
\[
f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2 + \ldots
\]
Consider Samples taken at increments of \( h \) and first two terms, we have
\[
f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2} f''(x_0)h^2
\]
\[
f(x_0 - h) = f(x_0) - f'(x_0)h + \frac{1}{2} f''(x_0)h^2
\]
Subtracting and adding \( f(x_0 + h) \) and \( f(x_0 - h) \) respectively yields
\[
\frac{f'(x_0)}{2h} = \frac{f(x_0 + h) - f(x_0 - h)}{2h} \quad \text{Convolve with First Derivative: } [-1 \ 0 \ 1]
\]
\[
\frac{f''(x_0)}{h^2} = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} \quad \text{Second Derivative: } [1 \ -2 \ 1]
\]

Image Gradient

- To compute gradient of image \( I \)
- Convolve \( I \) with \([-1 \ 0 \ 1]\) to produce \( I_x \)
- Convolve \( I \) with \([-1 \ 0 \ 1] \) to produce \( I_y \)
- Gradient is: \( \nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \)

Feature extraction: Corners and blobs

The Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity

Edge Detectors Tend to Fail at Corners

Finding Corners

Intuition:
- Right at corner, gradient is ill-defined.
- Near corner, gradient has two different values.
Distribution of gradients for different image patches

Formula for Finding Corners

Shi-Tomasi Detector

We look at matrix:

Why This?

General Case:

Because C is a symmetric positive definite matrix, it can be factored as follows:

Where R is a 2x2 rotation matrix and \( \lambda_i \) is non-negative.

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 \( \lambda_2 > 0 \)?

So, to detect corners

- Filter image with a Gaussian.
- Compute the gradient everywhere.
- Move window over image and construct C over the window.
- Use linear algebra to find \( \lambda_1 \) and \( \lambda_2 \).
- If they are both big, we have a corner.
  1. Let \( e(x,y) = \min(\lambda_1(x,y), \lambda_2(x,y)) \)
  2. \((x,y)\) is a corner if it’s local maximum of \( e(x,y) \) and \( e(x,y) > \tau \)

Parameters: Gaussian std. dev, window size, threshold
Variations of Criteria

- Let $\lambda_1$ and $\lambda_2$ be the two Eigenvalues of $C$ with $\lambda_1 \leq \lambda_2$
  - $\lambda_1$
  - $\lambda_1/\text{trace}(C) = \lambda_1/(\lambda_1+\lambda_2)$
  - $\text{Det}(C) + k \times \text{trace}^2(C)$ with $k=0.04$ [Harris]

Subpixel localization

See Blackboard