Computer Vision

CSE 455
Corner Detection

Linda Shapiro
Professor of Computer Science & Engineering
Professor of Electrical Engineering
Review: What’s an edge?

- Image is a function
- Edges are rapid changes in this function
Image derivatives

- Recall: \[ f'(a) = \lim_{h \to 0} \frac{f(a + h) - f(a)}{h}. \]
- Want smoothing too!

\[ \frac{1}{2} \cdot \begin{pmatrix} -1 & 0 & 1 \\ 1 & 2 & 1 \\ 1 & 2 & 1 \end{pmatrix} \quad \text{Sobel Operator} \]

\[ f(x) \quad f'(x) \]
Laplacian (2nd derivative)!

- Crosses zero at extrema
- Recall:
  \[ f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}. \]
- Laplacian:
  \[ \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]
- Again, have to estimate \( f''(x) \):

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]
Laplacians also sensitive to noise

- Again, use gaussian smoothing
- Can just use one kernel since convs commute
- **Laplacian of Gaussian, LoG**
- Can get good approx. with 5x5 - 9x9 kernels
Difference of Gaussian (DoG)

- Gaussian is a low pass filter
- Strongly reduce components with frequency $f < \sigma$
- $(g*I)$ low frequency components
- $I - (g*I)$ high frequency components
- $g(\sigma_1)*I - g(\sigma_2)*I$
  - Components in between these frequencies
- $g(\sigma_1)*I - g(\sigma_2)*I = [g(\sigma_1) - g(\sigma_2)]*I$

![Diagram of Gaussian functions](image)

$\sigma = 2$  $-  $  $\sigma = 1$
DoGs
Canny Edge Detection

- Your first image processing pipeline!
  - Old-school CV is all about pipelines

Algorithm:

- Smooth image (only want “real” edges, not noise)
- Calculate gradient direction and magnitude
- Non-maximum suppression perpendicular to edge
- Threshold into strong, weak, no edge
- Connect together components

http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/
Gradient magnitude and direction

- Sobel filter
Non-maximum suppression
Non-maximum suppression
Non-maximum suppression

http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/
Threshold edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - $R > T$: strong edge
  - $R < T$ but $R > t$: weak edge
  - $R < t$: no edge
- Why two thresholds?

http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/
Connect ‘em up!

- Strong edges are edges!
- Weak edges are edges iff they connect to strong
- Look in some neighborhood (usually 8 closest)

http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/
Features!

- Highly descriptive local regions
- Ways to describe those regions
- Useful for:
  - Matching
  - Recognition
  - Detection

[Diagrams showing image gradients and keypoint descriptor]
How to create a panorama

- Say we are stitching a panorama
- Want patches in image to match to other image
- Hopefully only match one spot
Q1: How close are two patches?

- Sum squared difference
- Images I, J
- $\sum_{x,y} (I(x,y) - J(x,y))^2$
Q2: How can we find unique patches?

- Sky: bad
  - Very little variation
  - Could match any other sky
How can we find unique patches?

- **Sky: bad**
  - Very little variation
  - Could match any other sky

- **Edge: ok**
  - Variation in one direction
  - Could match other patches along same edge
How can we find unique patches?

- **Sky**: bad
  - Very little variation
  - Could match any other sky

- **Edge**: ok
  - Variation in one direction
  - Could match other patches along same edge

- **Corners**: good!
  - Only one alignment matches
What are we going to do?

• We are going to build a panorama from two (or more) images.

• We need to learn about
  – Finding interest points
  – Describing small patches about such points
  – Finding matches between pairs of such points on two images, using the descriptors
  – Selecting the best set of matches and saving them
  – Constructing homographies (transformations) from one image to the other and picking the best one
  – Stitching the images together to make the panorama
Preview: Harris detector

Interest points extracted with Harris (~ 500 points)
How can we find corresponding points?
Not always easy

NASA Mars Rover images
Answer below (look for tiny colored squares…)
Human eye movements

Yarbus eye tracking

What catches your interest?
Interest points

- Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
  - Which points would you choose?
Intuition
Corners

- 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

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions

Source: A. Efros
Let’s look at the gradient distributions
Principal Component Analysis

Principal component is the direction of highest variance.

Next, highest component is the direction with highest variance *orthogonal* to the previous components.

How to compute PCA components:

1. Subtract off the mean for each data point.
2. Compute the covariance matrix.
3. Compute eigenvectors and eigenvalues.
4. The components are the eigenvectors ranked by the eigenvalues.

\[
H x = \lambda x
\]

Definition: A scalar \( \lambda \) is called an eigenvalue of the \( n \times n \) matrix \( A \) if there is a nontrivial solution \( x \) of \( Ax = \lambda x \). Such \( x \) is called an eigenvector corresponding to the eigenvalue \( \lambda \).
Corners have ... Both eigenvalues are large!
Second Moment Matrix or **Harris Matrix**

\[
H = \begin{bmatrix}
\Sigma_i w_i I_x I_x & \Sigma_i w_i I_x I_y \\
\Sigma_i w_i I_x I_y & \Sigma_i w_i I_y I_y
\end{bmatrix}
\]

2 x 2 matrix of image derivatives smoothed by Gaussian weights.

Notation:

\[
I_x \leftrightarrow \frac{\partial I}{\partial x} \quad I_y \leftrightarrow \frac{\partial I}{\partial y} \quad I_x I_y \leftrightarrow \frac{\partial I}{\partial x} \frac{\partial I}{\partial y}
\]

- First compute $I_x$, $I_y$, and $I_x I_x$, $I_y I_y$, $I_x I_y$; then apply Gaussian to each.
From HW3: Structure matrix

- Weighted sum of gradient information
  - \[ \begin{vmatrix}
  \sum_i w_i x(i) x(i) & \sum_i w_i x(i) y(i) \\
  \sum_i w_i x(i) y(i) & \sum_i w_i y(i) y(i)
  \end{vmatrix} \]

- Use Gaussian weighting
- Eigen vectors/values of this matrix summarize the distribution of the gradients nearby
- \(\lambda_1\) and \(\lambda_2\) are eigenvalues
  - \(\lambda_1\) and \(\lambda_2\) both small: no gradient
  - \(\lambda_1 \gg \lambda_2\): gradient in one direction
  - \(\lambda_1\) and \(\lambda_2\) similar: multiple gradient directions, corner

We’ll tell you how to store this!
Estimating Response

\[ H = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \lambda_{\pm} = \frac{1}{2} \left( (a + d) \pm \sqrt{4bc + (a - d)^2} \right) \]

- A few methods we use to estimate:
- Calculate these directly from the 2x2 matrix
  - \( \text{det}(S) = ad - bc = \lambda_1 * \lambda_2 \)
  - \( \text{trace}(S) = a + d = \lambda_1 + \lambda_2 \)
- Estimate formula 1: \( R = \text{det}(S) - \alpha \text{trace}(S)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \)
- Estimate formula 2: \( R = \frac{\text{det}(S)}{\text{trace}(S)} = \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)} \)
- If these estimates are large, we call it a corner
Harris Corner Detector

- Calculate derivatives $I_x$ and $I_y$
- Calculate 3 measures $I_xI_x$, $I_yI_y$, $I_xI_y$
- Calculate weighted sums
  - Want a weighted sum of nearby pixels, guess what this is?
    - Gaussian!
- Estimate response
- Non-max suppression!
Harris Detector: Steps
Harris Detector: Steps

Compute corner response $R$
Harris Detector: Steps

Find points with large corner response: $R > \text{threshold}$
Harris Detector: Steps

Take only the points of local maxima of $R$
Harris Detector: Results
Properties of the Harris corner detector

- Translation invariant? Yes
- Rotation invariant? Yes
- Scale invariant? No

Corner!

All points will be classified as edges

What’s the problem?
• Can we do better than Harris?
• The Harris corner detector is not widely used except in class assignments.
• The SIFT detector/descriptor is the standard.
• Let’s take a look.
What is the “best” scale?
Scale Invariance

How can we independently select interest points in each image, such that the detections are repeatable across different scales?

K. Grauman, B. Leibe
Differences between Inside and Outside

1. We can use a Laplacian function
Scale

But we use a Gaussian.

Why Gaussian?

It is invariant to scale change, i.e., \( f * G_\sigma * G_\sigma' = f * G_\sigma'' \) and has several other nice properties. Lindeberg, 1994

In practice, the Laplacian is approximated using a Difference of Gaussian (DoG).
Difference-of-Gaussian (DoG)

\[ G_1 - G_2 = \text{DoG} \]
DoG example

Take Gaussians at multiple spreads and uses DoGs.

\[ \sigma = 1 \]

\[ \sigma = 66 \]
Scale invariant interest points

Interest points are local maxima in both position and scale.

Apply Gaussians with different $\sigma$’s.

Look for extrema in difference of Gaussians.

$\Rightarrow$ List of $(x, y, \sigma)$
Scale (Lowe uses multiple scales)

In practice the image is downsampled for larger sigmas.

Lowe’s Pyramid Scheme

The parameter $s$ determines the number of images per octave.

$s+2$ filters

\[
\sigma_{s+1} = 2^{(s+1)/s} \sigma_0 \\
\sigma_s = 2^{s/2} \sigma_0 \\
\sigma_i = 2^{i/s} \sigma_0 \\
\sigma_2 = 2^{2/s} \sigma_0 \\
\sigma_1 = 2^{1/s} \sigma_0 \\
\sigma_0
\]

$s+3$ images including original

Difference of Gaussian (DOG)

$s+2$ difference images
Key point localization

• Detect maxima and minima of difference-of-Gaussian in scale space

• Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below

For each max or min found, output is the location and the scale.
Scale-space extrema detection: experimental results over 32 images that were synthetically transformed and noise added.

- **Sampling in scale for efficiency**
  - How many scales should be used per octave? $S = ?$
    - More scales evaluated, more keypoints found
    - $S < 3$, stable keypoints increased too
    - $S > 3$, stable keypoints decreased
    - $S = 3$, maximum stable keypoints found
Results: Difference-of-Gaussian

K. Grauman, B. Leibe
How can we find correspondences?

Similarity transform
Rotation invariance

- Rotate patch according to its **dominant gradient orientation**
- This puts the patches into a canonical orientation.
Orientation Normalization

- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation

[Lowe, SIFT, 1999]
What’s next?

Once we have found the keypoints and a dominant orientation for each, we need to describe the (rotated and scaled) neighborhood about each.

128-dimensional vector
Important Point

• People just say “SIFT”.

• But there are TWO parts to SIFT.
  1. an interest point detector
  2. a region descriptor

• They are independent. Many people use the region descriptor without looking for the points.
Homework 2

filter_image.c

Due: April 28
2.0 Copy two functions from previous homework

Note that you wrote two functions (i.e. l1_normalize and make_box_filter) in homework 1, which will be useful for this assignment.

Simply copy your solution from your previous submission to the filter_image.c file.
2.1 Write a convolution function

image convolve_image(image im, image filter, int preserve) {}

Note the “preserve” parameter: if preserve is 1, the output should have the same number of channels as the input; if preserve is 0, the output should only have one channel.

Read the instructions in the Readme file carefully!
2.2 Make some more filters and try them out!

Create 3 useful filters (kernel size = 3 for all of them):

- image make_highpass_filter()
- image make_sharpen_filter()
- image make_emboss_filter()

Answer the Questions (in your source file):

1. With which of these filters should we use preserve when we run our convolution and which ones should we not? Why?
2. Do we have to do any post-processing for the above filters? Which ones and why?
2.3 Implement a Gaussian kernel

image make_gaussian_filter(float sigma)

Create a Gaussian filter with given sigma.

Note that the kernel size is the next highest odd integer from 6x sigma. E.g.

- if sigma is 0.6, then the size of the Gaussian filter is 5 x 5;
- if sigma is 2, then the size of Gaussian filter is 13 x 13.

\[ G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]
2.4 Hybrid images

- image add_image(image a, image b)
- image sub_image(image a, image b)

The input images a and b have the same height, width, and channels.

Sum or subtract the give two images (a + b or a - b) and return the result. The result image should also have the same height, width, and channels as the inputs.
2.5 Sobel Filters

- image make_gx_filter()
- image make_gy_filter()
- image *sobel_image(image im)

We apply Sobel filters (kernel size = 3) to the input image. The sobel_image function should return two images: the gradient magnitude and direction.

```c
image *rst = calloc(2, sizeof(image));
image magnitude = .... TODO
rst[0] = magnitude;
rst[1] = direction;
```
2.6 [Extra Credit] Median Filter

Fill in the function `image apply_median_filter(image im, int k)`.

We assume a median filter is a square, with the same height and width. The kernel size is always a positive odd number. We use "clamp" padding for borders and corners. The output image should have the same width, height, and channels as the input image. You should apply the median filter to each channel of the input image `im`.

![Image 1](image1.png)
![Image 2](image2.png)