Lecture 7: Features 1

COMP 590/776: Computer Vision
Instructor: Soumyadip (Roni) Sengupta
TA: Mykhailo (Misha) Shvets

Course Website: Scan Me!
Recap
Original → Linear Filtering: Cross-correlation & Convolution → Blur → Shifted left by 1 pixel → Sharpening
Convolution

• Same as cross-correlation, except that the kernel is “flipped” (horizontally and vertically)

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i - u, j - v]
\]

• Convolution is **commutative** and **associative**
Aliasing

- Images are Signals in 2D
- Signals contain low frequency (smooth regions) and high frequency (sharp changes in intensity)
- To accurately downsample a signal/image, # of samples $\geq 2 \times$ highest frequency in the signal. (Nyquist Rate!)
- If your task is to downsample by $1/4$, you do not have enough samples, thus the downsampled image is inaccurate especially in terms of high frequency components.
Partial Derivatives

Can be implemented as a convolution operation

\[
\frac{\partial f(x, y)}{\partial x} \quad \frac{\partial f(x, y)}{\partial y}
\]
Noise in 2D

Noisy Input

Ix via [-1,0,1]

Zoom

Source: D. Fouhey
Noise + Smoothing

Smoothed Input

Ix via [-1,0,1]

Zoom

Source: D. Fouhey
Canny edge detector

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   • Define two thresholds: low and high
   • Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei, J. Redmon
Fourier Transform

Teases away fast vs. slow changes in the image.
A sum of sines

- Our building block: $A \sin(\omega x + \phi)$

- Add enough of them to get any signal $f(x)$ you want!

\[ f(\text{target}) = f_1 + f_2 + f_3 + \ldots + f_n + \ldots \]
Scary Math

Fourier Transform: \[ F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-j\omega x} \, dx \]

Inverse Fourier Transform: \[ f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)e^{j\omega x} \, d\omega \]

Discrete Fourier Transform
\[ X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}. \]

The discrete Fourier transform (DFT) of an image \( f \) of size \( M \times N \) is an image \( F \) of same size defined as:

\[ F(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)e^{-j2\pi \left( \frac{um}{M} + \frac{vn}{N} \right)} \]
Amplitude Spectrum of DFT
Application: Hybrid Images


Gaussian Filter

Laplacian Filter

unit impulse

Gaussian

Laplacian of Gaussian
Today: Feature extraction—Corners and blobs
Motivation: Automatic panoramas

Credit: Matt Brown
Motivation: Automatic panoramas

GigaPan:  
http://gigapan.com/

Also see Google Zoom Views:  
https://www.google.com/culturalinstitute/beta/project/gigapixels
Steps of creating a Panorama
(For this & next week)

This is your next homework assignment!
Why extract features?

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

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

Step 1: extract features
Step 2: match features
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
Step 4: blending images

This Week
Next Week
Content: Today’s class

• Why detect features?
• What is a good feature?
• Harris Corner Detector
• Properties of Harris Corner Detector
• Blob Detector
Content: Today’s class

• Why detect features?
• What is a good feature?
• Harris Corner Detector
• Properties of Harris Corner Detector
• Blob Detector
Object recognition (David Lowe)
Application: Visual SLAM

• (aka Simultaneous Localization and Mapping)
3D Reconstruction

Internet Photos ("Colosseum")

Reconstructed 3D cameras and points
Augmented Reality
Image matching

by Diva Sian

by swashford
Harder case

by Diva Sian

by scgbt
Harder still?
Answer below (look for tiny colored squares...)

NASA Mars Rover images with SIFT feature matches
Feature matching for object search
Feature matching
More motivation...

Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking (e.g. for AR)
- Object recognition
- Image retrieval
- Robot/car navigation
- ... other
Content: Today’s class

• Why detect features?
• What is a good feature?
• Harris Corner Detector
• Properties of Harris Corner Detector
• Blob Detector
What makes a good feature?

Features = A set of salient keypoints (pixels) in an image
Local features: main components

1) **Detection**: Identify the interest points

2) **Description**: Extract vector feature descriptor surrounding each interest point

3) **Matching**: Determine correspondence between descriptors in two views
Advantages of local features

Locality
• features are local, so robust to occlusion and clutter

Quantity
• hundreds or thousands in a single image

Distinctiveness:
• can differentiate a large database of objects

Efficiency
• real-time performance achievable
Invariant local features

Find features that are invariant to transformations

• geometric invariance: translation, rotation, scale
• photometric invariance: brightness, exposure, …
Want uniqueness

Look for image regions that are unusual
  • Lead to unambiguous matches in other images

How to define “unusual”? 
Content: Today’s class

• Why detect features?
• What is a good feature?
• **Harris Corner Detector**
• Properties of Harris Corner Detector
• Blob Detector
Harris corner detector

- C. Harris, M. Stephens. “A Combined Corner and Edge Detector”. 1988
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
Local measures of uniqueness

Suppose we only consider a small window of pixels
  • What defines whether a feature is a good or bad candidate?
Harris Detector: Basic Idea

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions
Local measures of uniqueness

- How does the window change when you shift it?
- Shifting the window in any direction causes a big change

“flat” region: no change in all directions
“edge”: no change along the edge direction
“corner”: significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov
Harris corner detection: the math

Consider shifting the window $W$ by $(u,v)$

- how do the pixels in $W$ change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” $E(u,v)$:

$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

- We are happy if this error is high
- Slow to compute exactly for each pixel and each offset $(u,v)$

Corner Detection: Mathematics

Change in appearance of window $W$ for the shift $[u,v]$:

$$E(u, v) = \sum_{(x,y) \in W} [I(x+u, y+v) - I(x, y)]^2$$
Corner Detection: Mathematics
Change in appearance of window $W$ for the shift $[u,v]$: 

$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

$I(x, y)$

$E(u, v)$

$E(3,2)$
Corner Detection: Mathematics

Change in appearance of window $W$ for the shift $[u,v]$:

$$E(u, v) = \sum_{(x,y) \in W} [I(x+u, y+v) - I(x, y)]^2$$

We want to find out how this function behaves for small shifts $E(u, v)$
Small motion assumption

Taylor Series expansion of $I$:

\[ I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms} \]

If the motion $(u, v)$ is small, then first order approximation is good:

\[ I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v \]

\[ \approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \]

shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide...

$f(x + u) \approx f(x) + f'(x)u$
Corner detection: the math

Consider shifting the window $W$ by $(u,v)$
  
  • define an SSD “error” $E(u,v)$:

\[
E(u, v) = \sum_{(x,y)\in W} [I(x + u, y + v) - I(x, y)]^2
\]

\[
\approx \sum_{(x,y)\in W} [I(x, y) + I_x u + I_y v - I(x, y)]^2
\]

\[
\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2
\]
Corner detection: the math

Consider shifting the window $W$ by $(u,v)$

- define an SSD “error” $E(u,v)$:

$$E(u, v) \approx \sum_{(x,y) \in W} [I_x u + I_y v]^2$$

$$\approx A u^2 + 2 B u v + C v^2$$

$$A = \sum_{(x,y) \in W} I_x^2 \quad B = \sum_{(x,y) \in W} I_x I_y \quad C = \sum_{(x,y) \in W} I_y^2$$

- Thus, $E(u,v)$ is locally approximated as a quadratic error function
The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u, v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

Let’s try to understand its shape.
\[ E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ A = \sum_{(x,y) \in W} I_x^2 \]

\[ B = \sum_{(x,y) \in W} I_x I_y \]

\[ C = \sum_{(x,y) \in W} I_y^2 \]

Horizontal edge: \( I_x = 0 \)

\[ H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix} \]
\[
E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\]

\[
A = \sum_{(x,y) \in W} I_x^2
\]

\[
B = \sum_{(x,y) \in W} I_x I_y
\]

\[
C = \sum_{(x,y) \in W} I_y^2
\]

Vertical edge: \( I_y = 0 \)

\[
H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}
\]

\[
E(u,v)
\]
Interpreting the second moment matrix

- Consider the axis-aligned case (gradients are either horizontal or vertical):

  \[(u \ v) \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} (u \ v) = 1\]

  \[a u^2 + b v^2 = 1\]

  \[\frac{u^2}{(a^{-1/2})^2} + \frac{v^2}{(b^{-1/2})^2} = 1\]
General case

We can visualize $H$ as an ellipse with axis lengths determined by the *eigenvalues* of $H$ and orientation determined by the *eigenvectors* of $H$.

Ellipse equation:

$$\begin{bmatrix} u & v \end{bmatrix} H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

$\lambda_{\text{max}}, \lambda_{\text{min}}$ : eigenvalues of $H$
Quick eigenvalue/eigenvector review

The eigenvectors of a matrix $A$ are the vectors $x$ that satisfy:

$$Ax = \lambda x$$

The scalar $\lambda$ is the eigenvalue corresponding to $x$

- The eigenvalues are found by solving:

$$\det(A - \lambda I) = 0$$

- In our case, $A = H$ is a 2x2 matrix, so we have

$$\det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2} \left[ (h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

Once you know $\lambda$, you find $x$ by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$
Corner detection: the math

\[ E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

Define shift directions with the smallest and largest change in error

\[ x_{\text{max}} = \text{direction of largest increase in } E \]
\[ l_{\text{max}} = \text{amount of increase in direction } x_{\text{max}} \]

\[ x_{\text{min}} = \text{direction of smallest increase in } E \]
\[ l_{\text{min}} = \text{amount of increase in direction } x_{\text{min}} \]

Eigenvalues and eigenvectors of \( H \)

- Define shift directions with the smallest and largest change in error
- \( x_{\text{max}} = \text{direction of largest increase in } E \)
- \( \lambda_{\text{max}} = \text{amount of increase in direction } x_{\text{max}} \)
- \( x_{\text{min}} = \text{direction of smallest increase in } E \)
- \( \lambda_{\text{min}} = \text{amount of increase in direction } x_{\text{min}} \)

\[ H x_{\text{max}} = \lambda_{\text{max}} x_{\text{max}} \]
\[ H x_{\text{min}} = \lambda_{\text{min}} x_{\text{min}} \]
Corner detection: the math

How are $\lambda_{\text{max}}$, $x_{\text{max}}$, $\lambda_{\text{min}}$, and $x_{\text{min}}$ relevant for feature detection?

• What’s our feature scoring function?
Corner detection: the math

How are $\lambda_{\text{max}}$, $x_{\text{max}}$, $\lambda_{\text{min}}$, and $x_{\text{min}}$ relevant for feature detection?

- What’s our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue ($\lambda_{\text{min}}$) of $H$
Interpreting the eigenvalues

Classification of image points using eigenvalues of $M$:

- **“Corner”**: $\lambda_1$ and $\lambda_2$ are large, $\lambda_1 \sim \lambda_2$; $E$ increases in all directions.
- **“Edge”**: $\lambda_1 >> \lambda_2$.
- **“Flat” region**: $\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions.
Visualization of second moment matrices
Visualization of second moment matrices

Note: axes are rescaled so ellipse areas are proportional to edge energy (i.e., bigger ellipses correspond to stronger edges)
Corner detection summary

Here’s what you do:

- Compute the gradient at each point in the image
- For each pixel:
  - Create the $H$ matrix from nearby gradient values
  - Compute the eigenvalues.
  - Find points with large response ($\lambda_{\text{min}} > \text{threshold}$)
- Choose those points where $\lambda_{\text{min}}$ is a local maximum as features

$$H = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
Corner detection summary

Here’s what you do:

• Compute the gradient at each point in the image
• For each pixel:
  • Create the $H$ matrix from nearby gradient values
  • Compute the eigenvalues.
  • Find points with large response ($\lambda_{\text{min}} > \text{threshold}$)
• Choose those points where $\lambda_{\text{min}}$ is a local maximum as features
The Harris operator

\( \lambda_{\text{min}} \) is a variant of the “Harris operator” for feature detection

\[
f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\text{determinant}(H)}{\text{trace}(H)}
\]

- The \text{trace} is the sum of the diagonals, i.e., \( \text{trace}(H) = h_{11} + h_{22} \)
- Very similar to \( \lambda_{\text{min}} \) but less expensive (no square root)
- Called the \textit{Harris Corner Detector} or \textit{Harris Operator}
- Lots of other detectors, this is one of the most popular

Alternate Version of Harris Detector

\[
R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 = \text{det}(M) - k \cdot \text{tr}(M)^2
\]

\( M = H \)
Harris detector: Steps

1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix $H$ in a Gaussian window around each pixel
3. Compute corner response function $f$ or $R$
4. Threshold $f$ or $R$
5. Find local maxima of response function (nonmaximum suppression)

The Harris operator

\[ \lambda_{\text{min}} \]
Harris detector example
f value (red high, blue low)
Threshold \((f > \text{value})\)
Find local maxima of $f$ (non-max suppression)
Harris features (in red)
Feature selection

- Distribute points evenly over the image
Adaptive Non-maximal Suppression

• Desired: Fixed # of features per image
  • Want evenly distributed spatially...
  • Sort points by non-maximal suppression radius
    [Brown, Szeliski, Winder, CVPR’05]
Weighting the derivatives

• In practice, using a simple window $W$ doesn’t work too well

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

• Instead, we’ll weight each derivative value based on its distance from the center pixel

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
Harris Corners – Why so complicated?

• Can’t we just check for regions with lots of gradients in the x and y directions?
  • No! A diagonal line would satisfy that criteria
Content: Today’s class

• Why detect features?
• What is a good feature?
• Harris Corner Detector

• Properties of Harris Corner Detector
• Blob Detector
Image transformations

- Geometric
  - Rotation
  - Scale

- Photometric
  - Intensity change
Invariance and equivariance

• We want corner locations to be *invariant* to photometric transformations and *equivariant* to geometric transformations
  • **Invariance:** image is transformed and corner locations do not change
  • **Equivariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations
  • (Sometimes “invariant” and “equivariant” are both referred to as “invariant”)
  • (Sometimes “equivariant” is called “covariant”)

![Example image of a cow with transformations](image-url)
Harris detector invariance properties: image translation

- Derivatives and window function are equivariant

Corner location is equivariant w.r.t. translation
Harris detector invariance properties: image rotation

Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is equivariant w.r.t. image rotation
Harris detector invariance properties: Affine intensity change

- Only derivatives are used \( \rightarrow \) invariance to intensity shift \( I \rightarrow I + b \)
- Intensity scaling: \( I \rightarrow a I \)

Partially invariant to affine intensity change
Harris detector invariance properties: scaling

Neither invariant nor equivariant to scaling
Scale invariant detection

Suppose you’re looking for corners

Key idea: find scale that gives local maximum of $f$
  - in both position and scale
  - One definition of $f$: the Harris operator
Scale Invariant Detection

• Consider regions (e.g. circles) of different sizes around a point
• Regions of corresponding sizes will look the same in both images
Scale Invariant Detection

- The problem: how do we choose corresponding circles independently in each image?

- Choose the scale of the “best” corner
Keypoint detection with scale selection

- We want to extract keypoints with *characteristic scales* that are *equivariant* (or *covariant*) w.r.t. to scaling of the image


Keypoint detection with scale selection

• We want to extract keypoints with characteristic scales that are equivariant (or covariant) w.r.t. to scaling of the image

• Approach: compute a scale-invariant response function over neighborhoods centered at each location \((x, y)\) and a range of scales \((\sigma)\), find scale-space locations \((x, y, \sigma)\) where this function reaches a local maximum

• A particularly convenient response function is given by the scale-normalized Laplacian of Gaussian (LoG) filter:

\[
\nabla^2_{\text{norm}} = \sigma^2 \left( \frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial y^2} g \right)
\]
Content: Today’s class

• Why detect features?
• What is a good feature?
• Harris Corner Detector
• Properties of Harris Corner Detector
• Blob Detector
Laplacian of Gaussian

\[ \frac{\partial^2}{\partial^2 x} g + \frac{\partial^2}{\partial^2 y} g \]

Source: J. Johnson and D. Fouhey
Scale-normalized Laplacian

- You need to multiply the LoG by $\sigma^2$ to make responses comparable across scales

$$\nabla^2_{\text{norm}} = \sigma^2 \left( \frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial^2 y} g \right)$$
Laplacian of Gaussian

- “Blob” detector

Find maxima and minima of LoG operator in space and scale
Scale selection: Characteristic Scale

- We can find the characteristic scale of the blob by convolving it with scale-normalized Laplacians at several scales ($\sigma$) and looking for the maximum response.
Scale-space blob detector: Example
Scale-space blob detector: Example

sigma = 11.9912
Scale-space blob detector: Example
Find local maxima in 3D position-scale space

\[ L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow \sigma^2 \]

\[ \sigma^2 \]

\[ \sigma \]

\[ \Rightarrow \text{List of } (x, y, s) \]

K. Grauman, B. Leibe
Local features: main components

This Class

1) **Detection:** Identify the interest points

2) **Description:** Extract vector feature descriptor surrounding each interest point.

\[ \mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}] \]

\[ \mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}] \]

3) **Matching:** Determine correspondence between descriptors in two views

Next Class:
We will learn about what is SIFT feature! The most famous feature in Computer Vision!!

Kristen Grauman
Slide Credits

• [CS5670, Introduction to Computer Vision](https://example.com), Cornell Tech, by Noah Snavely.

• [CS 194-26/294-26: Intro to Computer Vision and Computational Photography](https://example.com), UC Berkeley, by Alyosha Efros.

• [Fall 2022 CS 543/ECE 549: Computer Vision](https://example.com), UIUC, by Svetlana Lazebnik.