Detectors and Descriptors

EECS 442 – David Fouhey
Winter 2023, University of Michigan

https://web.eecs.umich.edu/~fouhey/teaching/EECS442_W23/
Goal

How big is this image as a vector?
389x600 = 233,400 dimensions (big)
Applications To Have In Mind

Part of the same photo?

Same computer from another angle?
Applications To Have In Mind

Building a 3D Reconstruction Out Of Images

Slide Credit: N. Seitz
Applications To Have In Mind

Stitching photos taken at different angles
One Example

Given two images: how do you align them?
for y in range(-ySearch, ySearch + 1):
    for x in range(-xSearch, xSearch + 1):
        # Touches all HxW pixels!
        check_alignment_with_images()
One Motivating Example

Given these images: how do you align them?

These aren’t off by a small 2D translation but instead by a 3D rotation + translation of the camera.

Photo credit: M. Brown, D. Lowe
One Solution

for y in yRange:
    for x in xRange:
        for z in zRange:
            for xRot in xRotVals:
                for yRot in yRotVals:
                    for zRot in zRotVals:
                        # touches all HxW pixels!
                        check_alignment_with_images()

This code should make you really unhappy

Note: this actually isn’t even the full number of parameters; it’s actually 8 for loops.
An Alternate Approach

Given these images: how would you align them?

A mountain peak!

This dark spot

A mountain peak!

This dark spot
An Alternate Approach

Finding and Matching

1: find corners+features
2: match based on local image data

Slide Credit: S. Lazebnik, original figure: M. Brown, D. Lowe
What Now?

Given pairs \( p_1, p_2 \) of correspondence, how do I align?

Consider translation-only case from HW1.
An Alternate Approach

Solving for a Transformation

3: Solve for transformation $T$ (e.g. such that $p1 \equiv T \ p2$) that fits the matches well

Note the homogeneous coordinates, you’ll see them again.

Slide Credit: S. Lazebnik, original figure: M. Brown, D. Lowe
An Alternate Approach

Blend Them Together

Key insight: we don’t work with full image. We work with only parts of the image.

Photo Credit: M. Brown, D. Lowe
Today

Finding edges (part 1) and corners (part 2) in images.

Corner of the glasses

Edge next to panel
Where do Edges Come From?
Where do Edges Come From?

Depth / Distance
Discontinuity

Why?
Where do Edges Come From?

Surface Normal / Orientation Discontinuity

Why?
Where do Edges Come From?

Surface Color / Reflectance Properties Discontinuity
Where do Edges Come From?

Illumination
Discontinuity
Last Time

\[-1 \quad 0 \quad 1\]

\[-1 \quad 0 \quad 1\]^T

lx

ly
Derivatives

Remember derivatives?

Derivative: rate at which a function $f(x)$ changes at a point as well as the direction that increases the function

Gradient: all of the partial derivatives (derivatives in only one direction) stacked together.
What Should I Know?

• Gradients are simply partial derivatives per-dimension: if \( x \) in \( f(x) \) has \( n \) dimensions, \( \nabla f(x) \) has \( n \) dimensions.

• Gradients point in direction of ascent and tell the rate of ascent.

• If \( a \) is minimum of \( f(x) \) \( \rightarrow \nabla_f(a) = 0 \).

• Reverse is not true, especially in high-dimensional spaces.
Last Time

\[
\begin{align*}
\text{lx} & = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \\
\text{ly} & = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T
\end{align*}
\]
Why Does This Work?

Image is function $f(x,y)$

Remember:

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

Approximate:

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x, y)}{1}$$

Another one:

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x - 1, y)}{2}$$
Other Differentiation Operations

<table>
<thead>
<tr>
<th></th>
<th>Horizontal</th>
<th>Vertical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prewitt</td>
<td>$\begin{bmatrix} -1 &amp; 0 &amp; 1 \ -1 &amp; 0 &amp; 1 \ -1 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 1 &amp; 1 \ 0 &amp; 0 &amp; 0 \ -1 &amp; -1 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Sobel</td>
<td>$\begin{bmatrix} -1 &amp; 0 &amp; 1 \ -2 &amp; 0 &amp; 2 \ -1 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 2 &amp; 1 \ 0 &amp; 0 &amp; 0 \ -1 &amp; -2 &amp; -1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

Why might people use these compared to [-1,0,1]?
Images as Functions or Points

Key idea: can treat image as a point in $\mathbb{R}^{(H\times W)}$ or as a function of $x,y$.

$$\nabla I(x, y) = \begin{bmatrix} \frac{\partial I}{\partial x}(x, y) \\ \frac{\partial I}{\partial y}(x, y) \end{bmatrix}$$

How much the intensity of the image changes as you go horizontally at $(x,y)$

(Of ten called $I_x$)
Image Gradient Direction

Some gradients

\[ \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} , 0 \end{bmatrix} \]

\[ \nabla f = \begin{bmatrix} 0, \frac{\partial f}{\partial y} \end{bmatrix} \]

\[ \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \end{bmatrix} \]

Figure Credit: S. Seitz
Image Gradient

Gradient: direction of maximum change. What’s the relationship to edge direction?

$I_x$  $I_y$
Image Gradient

$$(\mathbf{l}_{x}^2 + \mathbf{l}_{y}^2)^{1/2} : \text{magnitude}$$
Image Gradient

\[ \text{atan2}(I_y, I_x): \text{orientation} \]

I’m making the lightness equal to gradient magnitude
Image Gradient

atan2(ly, lx): orientation

Now I’m showing all the gradients
Image Gradient

atan2(ly, lx): orientation

Why is there structure at 1 and not at 2?
Noise

Consider a row of $f(x,y)$ (i.e., fix $y$)
Noise

Conv. image + per-pixel noise with

\[ I_{i,j} = \text{True image} \quad \epsilon_{i,j} \sim N(0, \sigma^2) \]

\[ D_{i,j} = (I_{i,j+1} + \epsilon_{i,j+1}) - (I_{i,j-1} + \epsilon_{i,j-1}) \]

\[ D_{i,j} = (I_{i,j+1} - I_{i,j-1}) + \epsilon_{i,j+1} - \epsilon_{i,j-1} \]

True difference Sum of 2 Gaussians

\[ \epsilon_{i,j} - \epsilon_{k,l} \sim N(0, 2\sigma^2) \rightarrow \text{Variance doubles!} \]
Noise

Consider a row of $f(x,y)$ (i.e., make $y$ constant)

$\frac{d}{dx} f(x)$

How can we use the last class to fix this?

Slide Credit: S. Seitz
Handling Noise

Sigma = 50

\( f \)

\( g \)

\( f \ast g \)

\( \frac{d}{dx}(f \ast g) \)

Slide Credit: S. Seitz
Noise in 2D

Noisy Input

Ix via [-1,01]

Zoom
Noise + Smoothing

Smoothed Input  Ix via [-1,01]  Zoom
Let’s Make It One Pass (1D)

\[
\frac{d}{dx} (f * g) = f * \frac{d}{dx} g
\]

Slide Credit: S. Seitz
Let’s Make It One Pass (2D)
Gaussian Derivative Filter

Which one finds the X direction?

Slide Credit: L. Lazebnik
Applying the Gaussian Derivative

1 pixel  
3 pixels  
7 pixels

Removes noise, but blurs edge

Slide Credit: D. Forsyth
Compared with the Past

Gaussian Derivative

Sobel Filter

Why would anybody use the bottom filter?
## Filters We’ve Seen

<table>
<thead>
<tr>
<th>Example</th>
<th>Smoothing</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>Remove noise</td>
<td>Find edges</td>
</tr>
<tr>
<td>Yes</td>
<td>1</td>
<td>No</td>
</tr>
</tbody>
</table>

**Goal**
- Remove noise
- Find edges

**Only +?**
- Yes
- No

**Sums to**
- 1
- 0

*Why sum to 1 or 0, intuitively?*
Problems

Still an unsolved problem
Localizing Reliably

• Suppose you need to meet someone but you can’t use your cell phone to coordinate
• Where do you agree to meet?
  A: Along the Huron river
  B: Along State Street
  C: At Liberty and State Street
  D: On North Campus
Desirables

• Repeatable: should find same things even with distortion
• Saliency: each feature should be distinctive
• Compactness: shouldn’t just be all the pixels
• Locality: should only depend on local image data
Example

Can you find the correspondences?

Slide credit: N. Snavely
Example Matches

Look for the colored squares

Slide credit: N. Snavely
Basic Idea

Should see where we are based on small window, or any shift → big intensity change.

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions
Formalizing Corner Detection
Formalizing Corner Detection

Zoom-In at x,y

Original Image
Formalizing Corner Detection

How might we measure similarity?
Formalizing Corner Detection

Zoom-In at x,y

Error (Sum Sqs) for u,v offset

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

Zoom-In at \( x, y \)

Error (Sum Sqs) for \( u, v \) offset

\[
\left( \sum \text{Squares} \right)^2
\]

between \( @u=-2, v=-3 \)

and unshifted

\[
\begin{array}{c}
\text{u} = -10 \\
\text{u} = 0 \\
\text{u} = 10 \\
\text{v} = -10 \end{array}
\]

\[
\begin{array}{c}
\text{v} = 0 \\
\text{v} = 10 \end{array}
\]
Formalizing Corner Detection

Zoom-In at \(x,y\)

Error at \(u=0,v=0\) is always 0. \textbf{Why?}

Error (Sum Sq) for \(u,v\) offset
Match The Location and Plot

Original Image and Zoom-In

Error Options

A

B
Match The Location and Plot

Original Image and Zoom-In

Error Options

A

B
Match The Location and Plot

Original Image and Zoom-In

Error Options

A

B
Match The Location and Plot

Original Image and Zoom-In

Error Options

A

B
Ok But Back To Math

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

Shifting windows around is expensive! We’ll find a trick to approximate this.

Note: only need to get the gist
Aside: Taylor Series for Images

Recall Taylor Series – way of *linearizing* a function:

\[ f(x + d) \approx f(x) + \frac{\partial f}{\partial x} d \]

Do the same with images, treating them as function of \( x, y \)

\[ I(x + u, y + v) \approx I(x, y) + I_x u + I_y v \]

For brevity: \( I_x = I_x \) at point \( (x,y) \), \( I_y = I_y \) at point \( (x,y) \)
Formalizing Corner Detection

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

Taylor series expansion for \( I \) at every single point in window

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

Cancel

\[ = \sum_{(x,y) \in W} \left( I_x u + I_y v \right)^2 \]

Expand

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

For brevity: \( I_x = I_x \) at point \((x,y)\), \( I_y = I_y \) at point \((x,y)\)
By linearizing image, we can approximate $E(u,v)$ with quadratic function of $u$ and $v$

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


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

$M$ is called the second moment matrix
Intuitively what is $M$?

Pretend gradients are either vertical or horizontal at a pixel (so $I_x I_y = 0$)

$$M = \begin{bmatrix} \sum_{x, y \in W} I_x^2 & \sum_{x, y \in W} I_x I_y \\ \sum_{x, y \in W} I_x I_y & \sum_{x, y \in W} I_y^2 \end{bmatrix} \approx \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

- $a, b$ both small: flat
- One big, other small: edge
- $a, b$ both big: corner

Obviously Wrong!
Intuitively what is $M$?

Pretend gradients are *either* vertical or horizontal at a pixel (so $|I_x I_y| = 0$)

$$M = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} \approx ? \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

$a, b$ both small: flat

One big, other small: edge

$a, b$ both big: corner

Image might be rotated by rotation $\theta$!
Intuitively what is $M$?

Pretend gradients are *either* vertical or horizontal at a pixel (so $I_x I_y = 0$)

$$M = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = V^{-1} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} V$$

- $a, b$ both small: flat
- One big, other small: edge
- $a, b$ both big: corner

If image rotated by rotation $\theta$ / matrix $V$

$M$ will look like

$$V^{-1} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} V$$
So What Now?

Can calculate $M$ at pixel, by summing nearby gradients, but need access to $a$ and $b$.

$$M = \begin{bmatrix}
\sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\
\sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2
\end{bmatrix} = V^{-1} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} V$$

Given $M$, can decompose it into eigenvectors $V$ and eigenvalues $\lambda_1, \lambda_2$ with $M = V^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} V$.

Really slow. Why?
So What Now?

Can calculate $M$ at pixel, by summing nearby gradients, but need access to $a$ and $b$.

$$M = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = V^{-1} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} V$$

Instead: compute quantity $R$ from $M$

$$R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

**Easy fast formula for 2x2**

**Fast – sum the diagonal**

**Empirical value, usually 0.04-0.06**
So What Now?

R tells us whether we’re at a corner, edge, or flat

\[ R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2 \]

Remake of standard diagram from S. Lazebnik from original Harris paper.
What Do I Need To Know?

• Need to be able to take derivatives of image
• Need to be able to compute the entries of $M$ at every pixel.
• Should know that some properties of $M$ indicate whether a pixel is a corner or not.

$$M = \begin{bmatrix}
\sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\
\sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2
\end{bmatrix}$$
In Practice

1. Compute partial derivatives $I_x$, $I_y$ per pixel
2. Compute $M$ at each pixel, using Gaussian weighting $w$

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


Slide credit: S. Lazebnik
In Practice

1. Compute partial derivatives $I_x$, $I_y$ per pixel
2. Compute $M$ at each pixel, using Gaussian weighting $w$
3. Compute response function $R$

$$R = \det(M) - \alpha \text{trace}(M)^2$$
$$= \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$


Slide credit: S. Lazebnik
Computing R

Slide credit: S. Lazebnik
Computing R

Slide credit: S. Lazebnik
In Practice

1. Compute partial derivatives $I_x$, $I_y$ per pixel
2. Compute $M$ at each pixel, using Gaussian weighting $w$
3. Compute response function $R$
4. Threshold $R$


Slide credit: S. Lazebnik
Thresholded R

Slide credit: S. Lazebnik
In Practice

1. Compute partial derivatives $I_x$, $I_y$ per pixel
2. Compute $M$ at each pixel, using Gaussian weighting $w$
3. Compute response function $R$
4. Threshold $R$
5. Take only local maxima (called non-maxima suppression)


Slide credit: S. Lazebnik
Thresholded, NMS R
Final Results
Desirable Properties

If our detectors are repeatable, they should be:

• **Invariant** to some things: image is transformed and corners remain the same

• **Covariant/equivariant** with some things: image is transformed and corners transform with it.
Recall Motivating Problem

Images may be different in lighting and geometry
Affine Intensity Change

\[ I_{new} = aI_{old} + b \]

M only depends on derivatives, so b is irrelevant

But a scales derivatives and there’s a threshold

Partially invariant to affine intensity changes
All done with convolution. Convolution is translation invariant.

Equivariant with translation
Rotations just cause the corner rotation to change. Eigenvalues remain the same.

Equivariant with rotation
Image Scaling

Corner

One pixel can become many pixels and vice-versa.

Not equivariant with scaling
For the Curious
Review: Quadratic Forms

Suppose have symmetric matrix $\mathbf{M}$, scalar $a$, vector $[u, v]$:

$$E([u, v]) = [u, v]\mathbf{M}[u, v]^T$$

Then the isocontour / slice-through of $F$, i.e.

$$E([u, v]) = a$$

is an ellipse.
Review: Quadratic Forms

We can look at the shape of this ellipse by decomposing $M$ into a rotation $+ scaling$

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

$\lambda_1$ and $\lambda_2$ are eigenvalues

direction of the fastest change

direction of the slowest change
Interpreting The Matrix $M$

The second moment matrix tells us how quickly the image changes and in which directions.

$$M = \begin{bmatrix}
\sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\
\sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2
\end{bmatrix} = R^{-1} \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} R$$

Can compute at each pixel

Directions

Amounts
Visualizing M

Slide credit: S. Lazebnik
Visualizing M

Technical note: M is often best visualized by first taking inverse, so long edge of ellipse goes along edge

Slide credit: S. Lazebnik
Interpreting Eigenvalues of M

- **“Corner”**
  - $\lambda_1$ and $\lambda_2$ are large,
  - $\lambda_1 \sim \lambda_2$;
  - $E$ increases in all directions

- **“Edge”**
  - $\lambda_2 \gg \lambda_1$

- **“Flat”** region
  - $\lambda_1$ and $\lambda_2$ are small;
  - $E$ is almost constant in all directions

Slide credit: S. Lazebnik; Note: this refers to previous ellipses, not original M ellipse. Other slides on the internet may vary.
Putting Together The Eigenvalues

\[ R = \det(M) - \alpha \text{trace}(M)^2 \]
\[ = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2 \]

\( \alpha \): constant (0.04 to 0.06)

Slide credit: S. Lazebnik; Note: this refers to previous ellipses, not original M ellipse. Other slides on the internet may vary.
Corners

9300 Harris Corners Pkwy, Charlotte, NC
Derivatives Review
Given quadratic function $f(x)$

$$f(x) = (x - 2)^2 + 5$$

$f(x)$ is function

$$g(x) = f'(x)$$

aka

$$g(x) = \frac{d}{dx} f(x)$$
Given quadratic function $f(x)$

$$f(x) = (x - 2)^2 + 5$$

What’s special about $x=2$?

$f(x)$ minim. at $2$
$g(x) = 0$ at $2$

$a = \text{minimum of } f \rightarrow$
$g(a) = 0$

Reverse is $\textit{not true}$
Rates of change

\[ f(x) = (x - 2)^2 + 5 \]

Suppose I want to increase \( f(x) \) by changing \( x \):

Blue area: move left
Red area: move right

Derivative tells you direction of ascent and rate
What Calculus Should I Know

• Really need intuition
• Need chain rule
• Rest you should look up / use a computer algebra system / use a cookbook
• Partial derivatives (and that’s it from multivariable calculus)
Partial Derivatives

• Pretend other variables are constant, take a derivative. That’s it.
• Make our function a function of two variables

\[ f(x) = (x - 2)^2 + 5 \]
\[ \frac{\partial}{\partial x} f(x) = 2(x - 2) \cdot 1 = 2(x - 2) \]

\[ f_2(x, y) = (x - 2)^2 + 5 + (y + 1)^2 \]
\[ \frac{\partial}{\partial x} f_2(x) = 2(x - 2) \]

Pretend it’s constant → derivative = 0
Zooming Out

\[ f_2(x, y) = (x - 2)^2 + 5 + (y + 1)^2 \]

Dark = \( f(x, y) \) low
Bright = \( f(x, y) \) high
Taking a slice of 

\[ f_2(x, y) = (x - 2)^2 + 5 + (y + 1)^2 \]

Slice of \( y=0 \) is the function from before:

\[ f(x) = (x - 2)^2 + 5 \]

\[ f'(x) = 2(x - 2) \]
Taking a slice of

\[ f_2(x, y) = (x - 2)^2 + 5 + (y + 1)^2 \]

The derivative \( \frac{\partial}{\partial x} f_2(x, y) \) is the rate of change and direction in the \( x \) dimension.
Zooming Out

\[ f_2(x, y) = (x - 2)^2 + 5 + (y + 1)^2 \]

\[ \frac{\partial}{\partial y} f_2(x, y) \text{ is } 2(y + 1) \]
and is the rate of change & direction in y dimension
Zooming Out

\[ f_2(x, y) = (x - 2)^2 + 5 + (y + 1)^2 \]

**Gradient/Jacobian:**
Making a vector of

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

gives rate and direction of change.

Arrows point OUT of minimum / basin.