Detectors and Descriptors

EECS 442 – Prof. David Fouhey
Winter 2019, University of Michigan

http://web.eecs.umich.edu/~fouhey/teaching/EECS442_W19/
Administrivia

• Today/Thursday: Detecting edges and corners in images and describing them
• Discussion Section: reviewing projection and convolution.

• If you have accommodation needs or need an alternate midterm, please email me this week.
• Please do not schedule stuff for the class that is listed as the midterm going forward.
General Hints for Vision Debugging

• Visualize
• **Visualize**
• **Visualize**
• Break it into bite-sized chunks; verify each
• Test the smallest version possible
• Use a debugger (pdb or use the jupyter cells)
• Try explaining your code verbally
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 Familiar 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 (Hopefully Familiar) 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 \( p_1 = T p_2 \)) 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.
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

\[
\begin{bmatrix}
-1 & 0 & 1
\end{bmatrix}
\]

\(lx\)

\[
\begin{bmatrix}
-1 & 0 & 1
\end{bmatrix}^T
\]

\(ly\)
Last Time

$$(|x|^2 + |y|^2)^{1/2}$$
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><strong>Prewitt</strong></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><strong>Sobel</strong></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 $[-1,0,1]$?
Review: Gradients

Gradient is just the collection of partial derivatives in each dimension/direction. Points in direction that increases the most.

\[
g = \nabla f = \begin{bmatrix}
\frac{\partial f}{\partial x_1} \\
\vdots \\
\frac{\partial f}{\partial x_n}
\end{bmatrix}
\quad g(a) = \begin{bmatrix}
\frac{\partial f}{\partial x_1}(a) \\
\vdots \\
\frac{\partial f}{\partial x_n}(a)
\end{bmatrix}
\]

How much \( f \) changes in the direction of \( x_1 \) at point \( a \).
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 $(x,y)$

(Often called $I_x$)
Image Gradient Direction

Some gradients

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \]

\[ \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right] \]

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

Figure Credit: S. Seitz
Image Gradient

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

$I_x$  

$I_y$
Image Gradient

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

atan2(ly,lx): orientation

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

$\text{atan2}(I_y, I_x)$: 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 \( y \))

\[
f(x)
\]

\[
\frac{d}{dx} f(x)
\]

Slide Credit: S. Seitz
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 \quad \text{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)

How can we use the last class to fix this?
Handling Noise

$\sigma = 50$

$\frac{d}{dx}(f * g)$

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)

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

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

Removes noise, but blurs edge

Slide Credit: D. Forsyth
Compared with the Past

Gaussian Derivative

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

Smoothing

Gaussian

Remove noise

Yes

1

Derivative

Deriv. of gauss

Find edges

No

0

Why sum to 1 or 0, intuitively?

Example

Goal

Only +?

Sums to

Yes

No

Slide Credit: J. Deng
Problems

Image  human segmentation  gradient magnitude

Still an active area of research
Corners

9300 Harris Corners Pkwy, Charlotte, NC

Slide Credit: S. Lazebnik
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 $\rightarrow$ big intensity change.

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions

Slide Credit: S. Lazebnik
Formalizing Corner Detection

Sum of squared differences between image and image shifted $u,v$ pixels over.

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

Sum of squared differences between image and image shifted $u,v$ pixels over.

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

What’s the value of $E(0,0)$?
Formalizing Corner Detection


Slide Credit: S. Lazebnik
Aside: Taylor Series for Images

Recall Taylor Series:

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

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

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

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

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

Cancel

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

Expand

\[ = \sum_{(x,y) \in W} I_xu^2 + 2I_xI_yuv + I_y^2v^2 \]

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

By linearizing image, we can approximate $E(u,v)$ with quadratic function of $u$ and $v$

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


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

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

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

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

If $a,b$ are both small: flat

If one is big, one is small: edge

If $a,b$ both big: corner
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

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

What are $\lambda_1$ and $\lambda_2$?

Slide credit: S. Lazebnik
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
Interpreting 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

Slide credit: S. Lazebnik
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)
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
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

Slide credit: S. Lazebnik
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.

Slide credit: S. Lazebnik
Recall Motivating Problem

Images may be different in lighting and geometry
Affine Intensity Change

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

M only depends on derivatives, so b is irrelevant

But a scales derivatives and there's a threshold

Partially invariant to affine intensity changes
Image Translation

All done with convolution. Convolution is translation invariant.

Equivariant with translation
Image Rotation

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
Next time

- Fixing this scaling issue
- Describing the corners
Desirable Properties

Repeatable: we can find the same place even after photometric metric and geometric distortion.
Desirable Properties

Compactness: we don’t just use all the pixels
Saliency: the place is distinctive
Desirable Properties

Locality: the feature doesn’t depend on the whole image but instead some part