Detectors and Descriptors

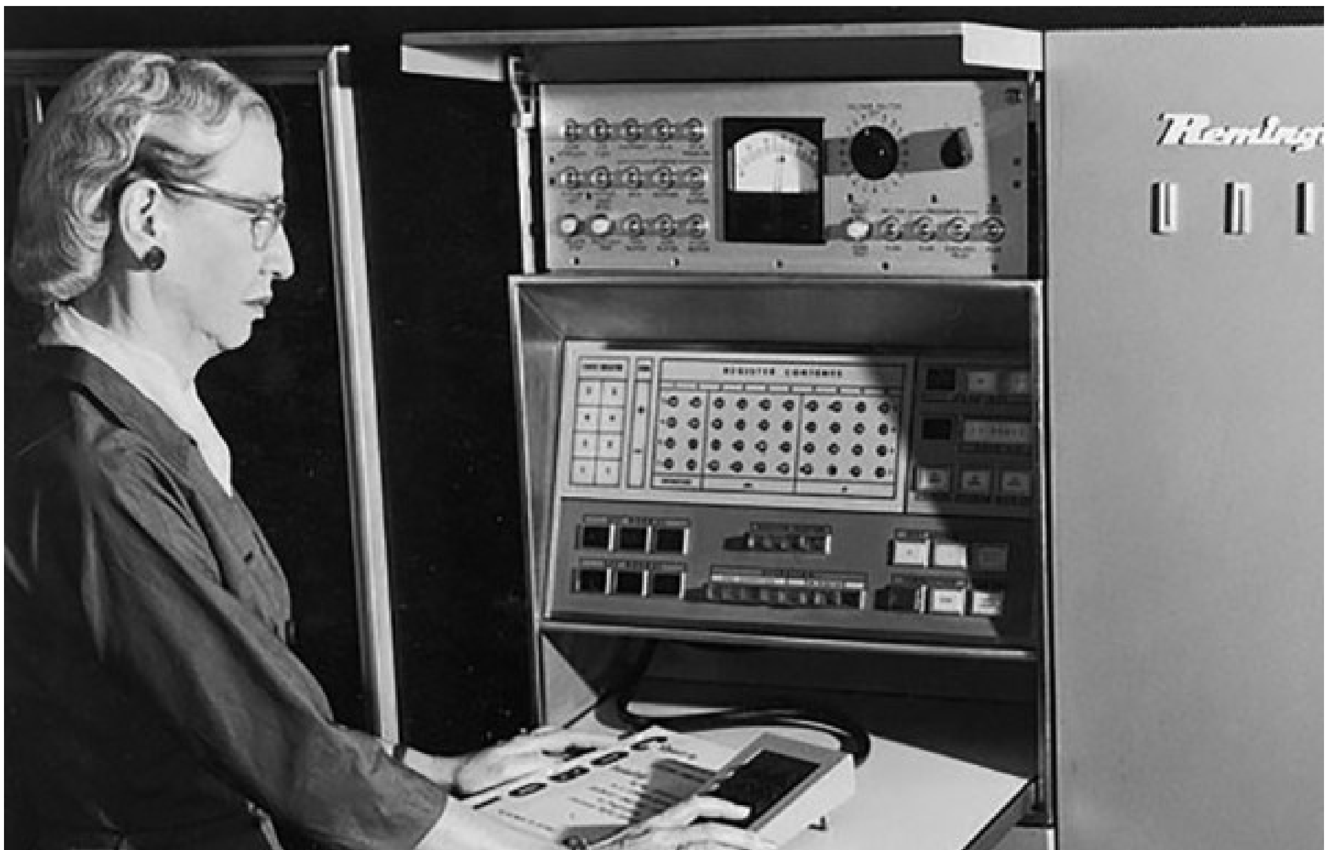
EECS 442 – David Fouhey
Winter 2023, University of Michigan

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

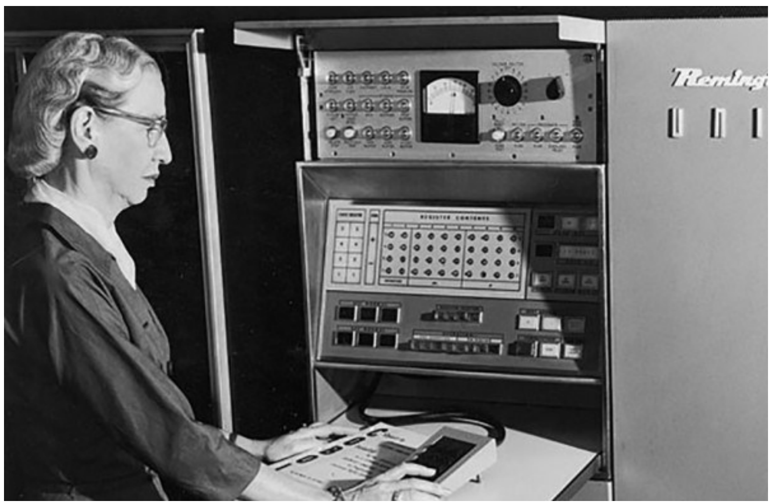
Goal

How big is this image as a vector?

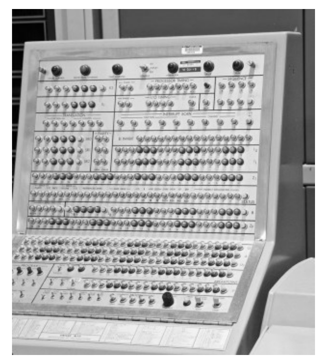
389x600 = 233,400 dimensions **(big)**



Applications To Have In Mind



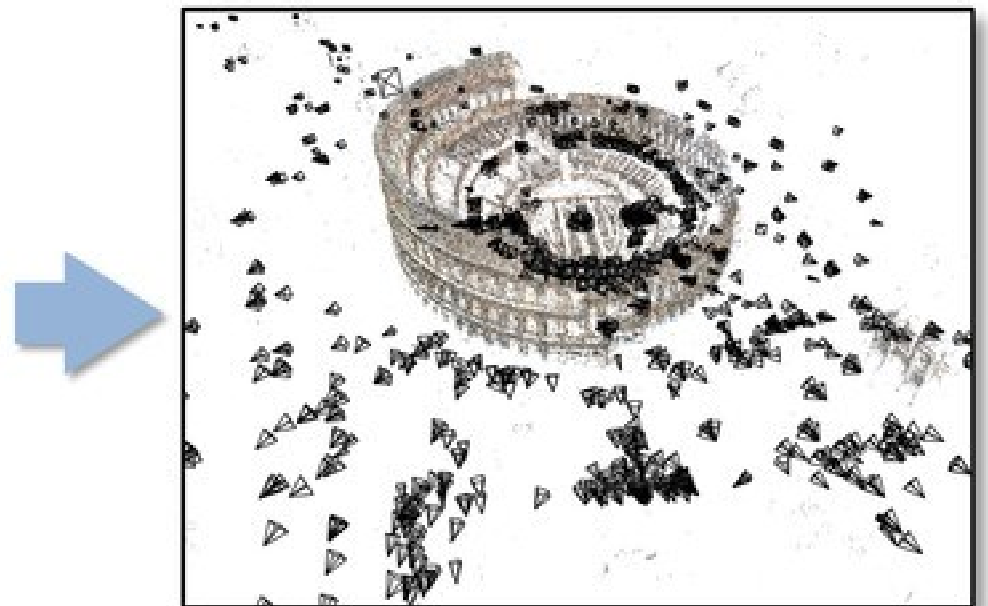
Part of the same photo?



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?

One Solution

```
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

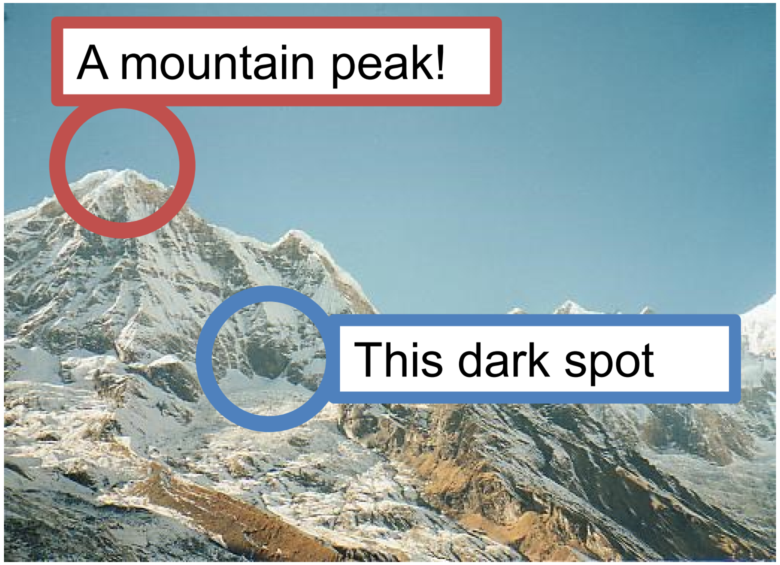
```
for y in yRange:
    for x in xRange:
        for z in zRange:
            for xRot in xRotVals:
                for yRot in yRotVals:
                    for zRot in zRotVals:
                        #touches all HxW pixels!
                        check_alignment_with_images()
```

This code should make you really <u>unhappy</u>

Note: this actually isn't even the full number of parameters; it's actually 8 for loops.

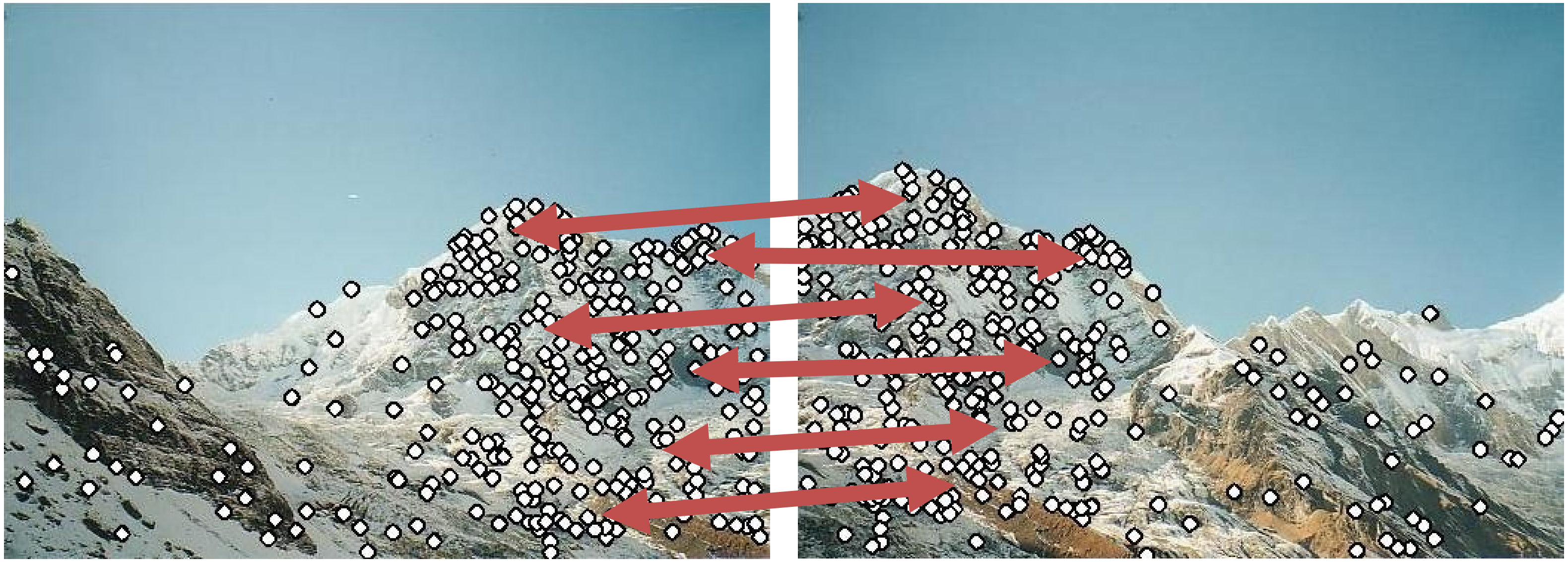
An Alternate Approach

Given these images: how would **you** align them?



An Alternate Approach

Finding and Matching



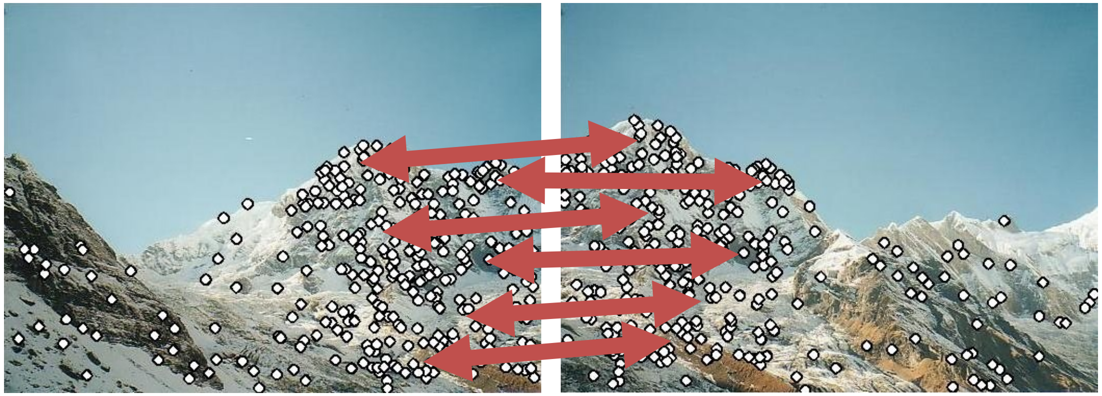
1: find corners+features

2: match based on local image data

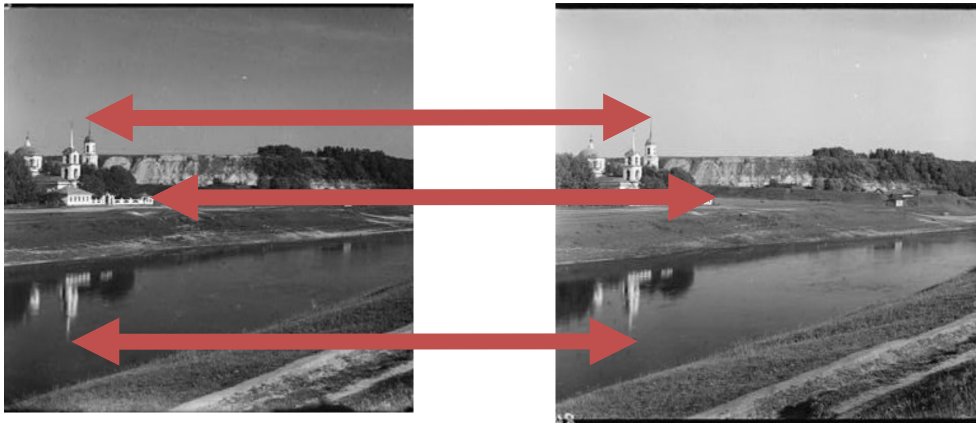
Slide Credit: S. Lazebnik, original figure: M. Brown, D. Lowe

What Now?

Given pairs **p1**,**p2** of correspondence, **how do I align?**

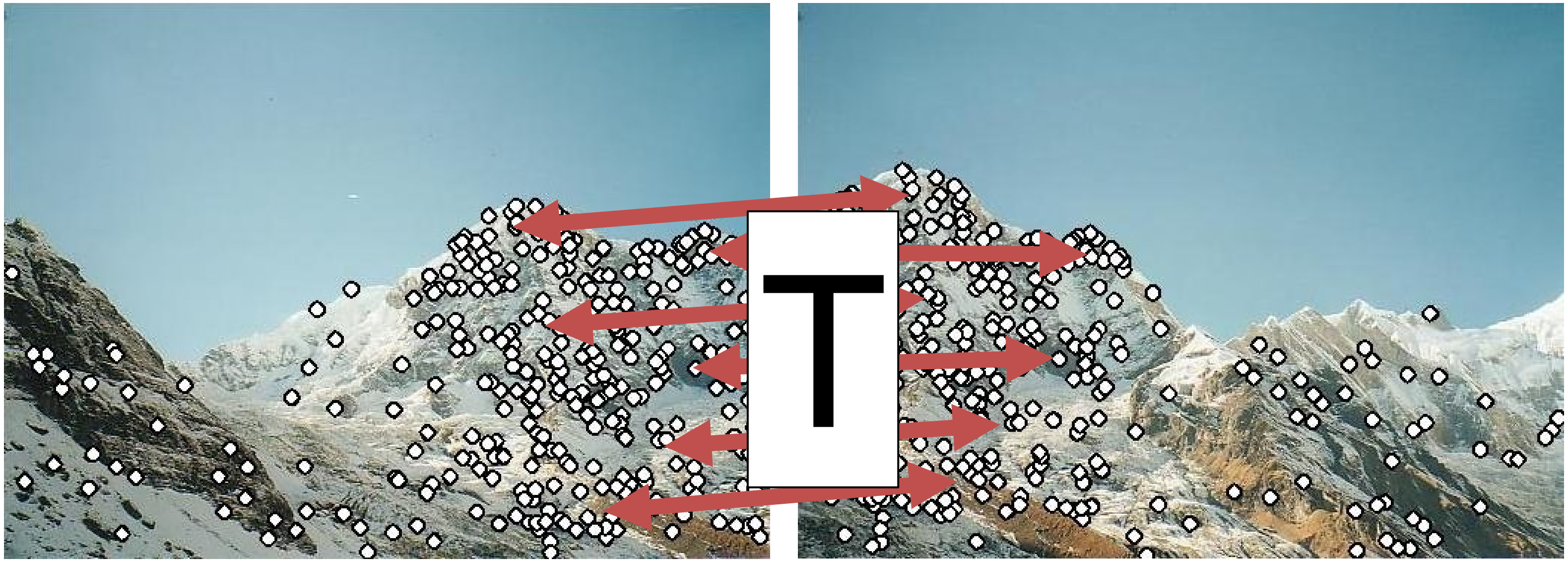


Consider translation-only case from HW1.



An Alternate Approach

Solving for a Transformation



3: Solve for transformation T (e.g. such that **p1 ≡ 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

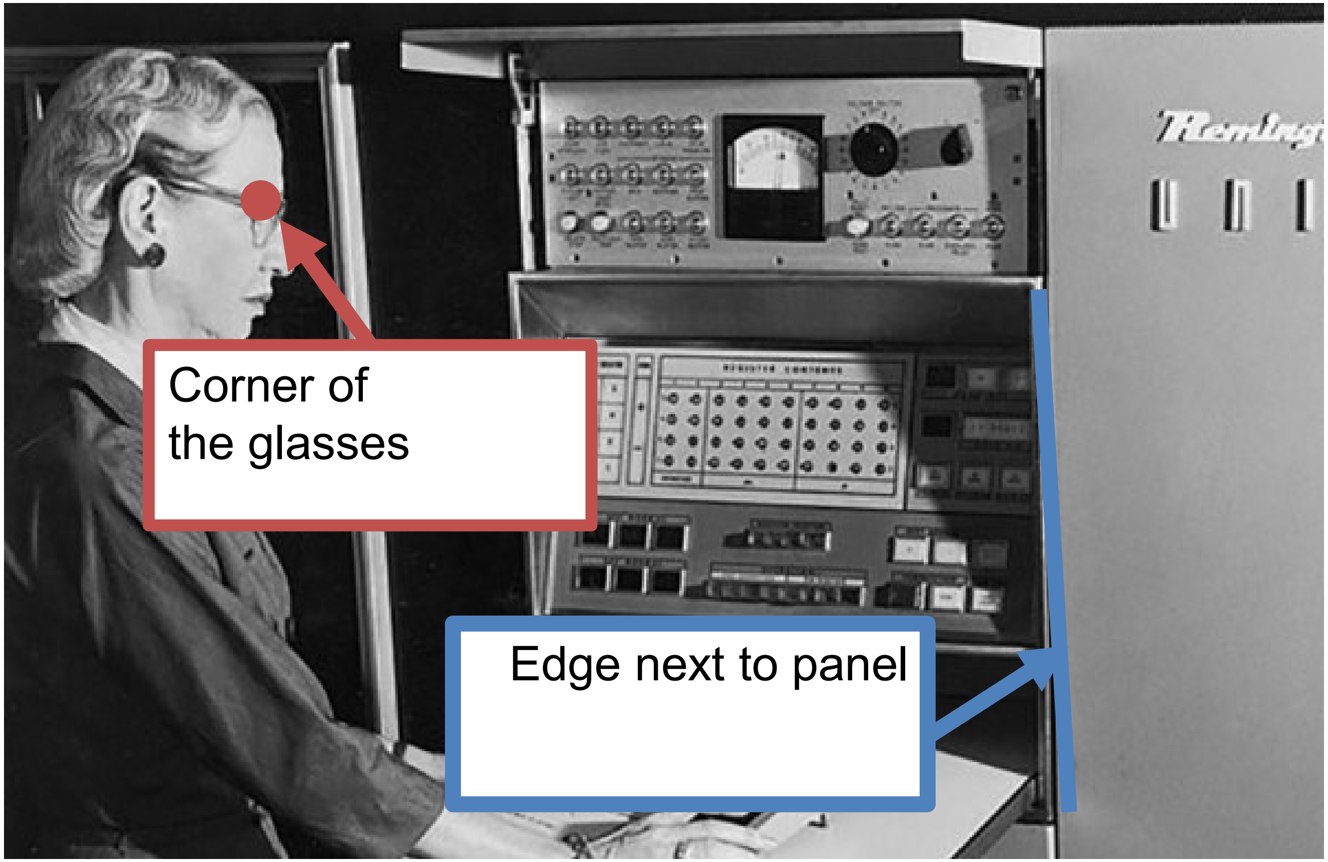
Blend Them Together

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

Photo Credit: M. Brown, D. Lowe

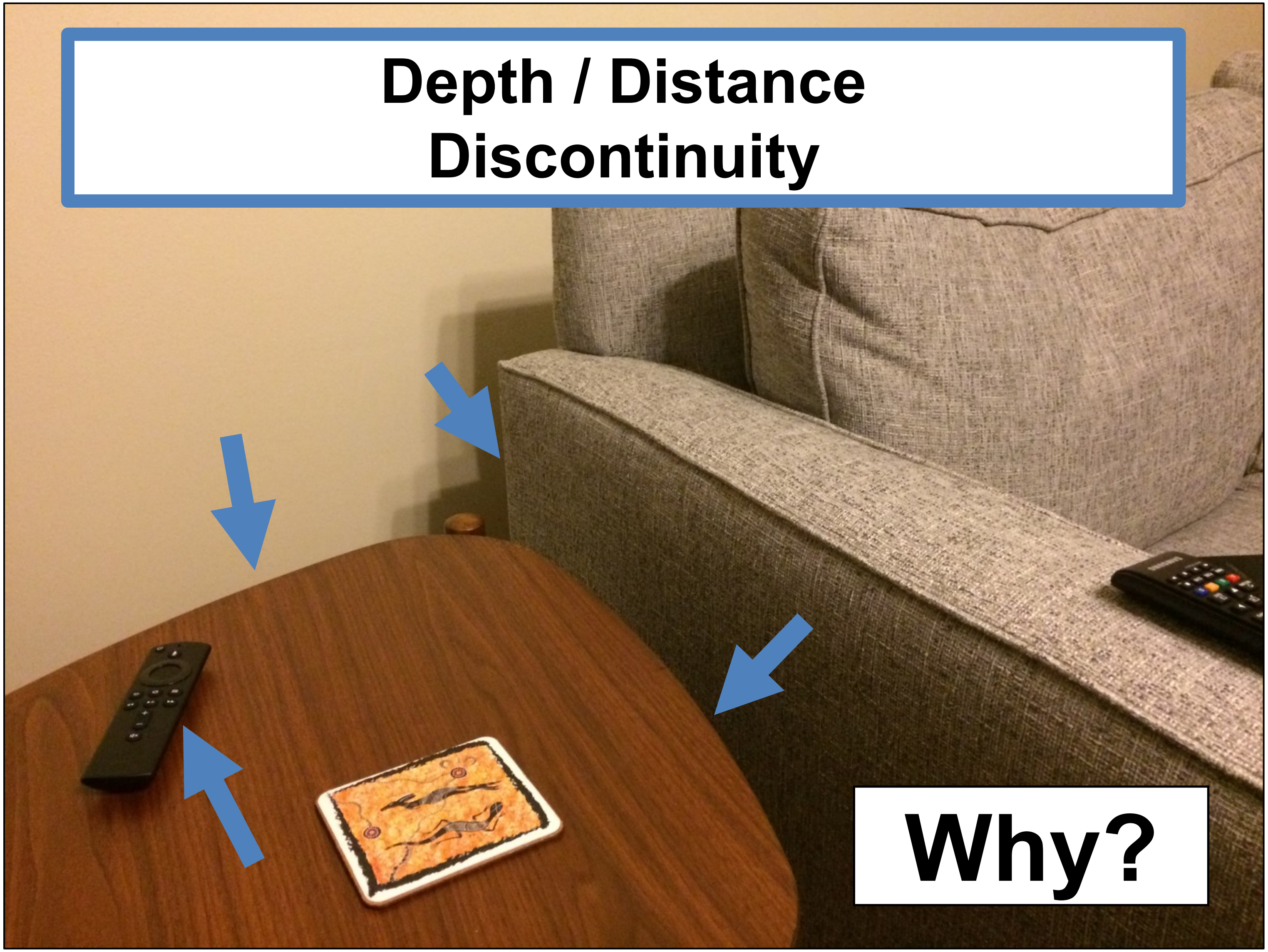
Today

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

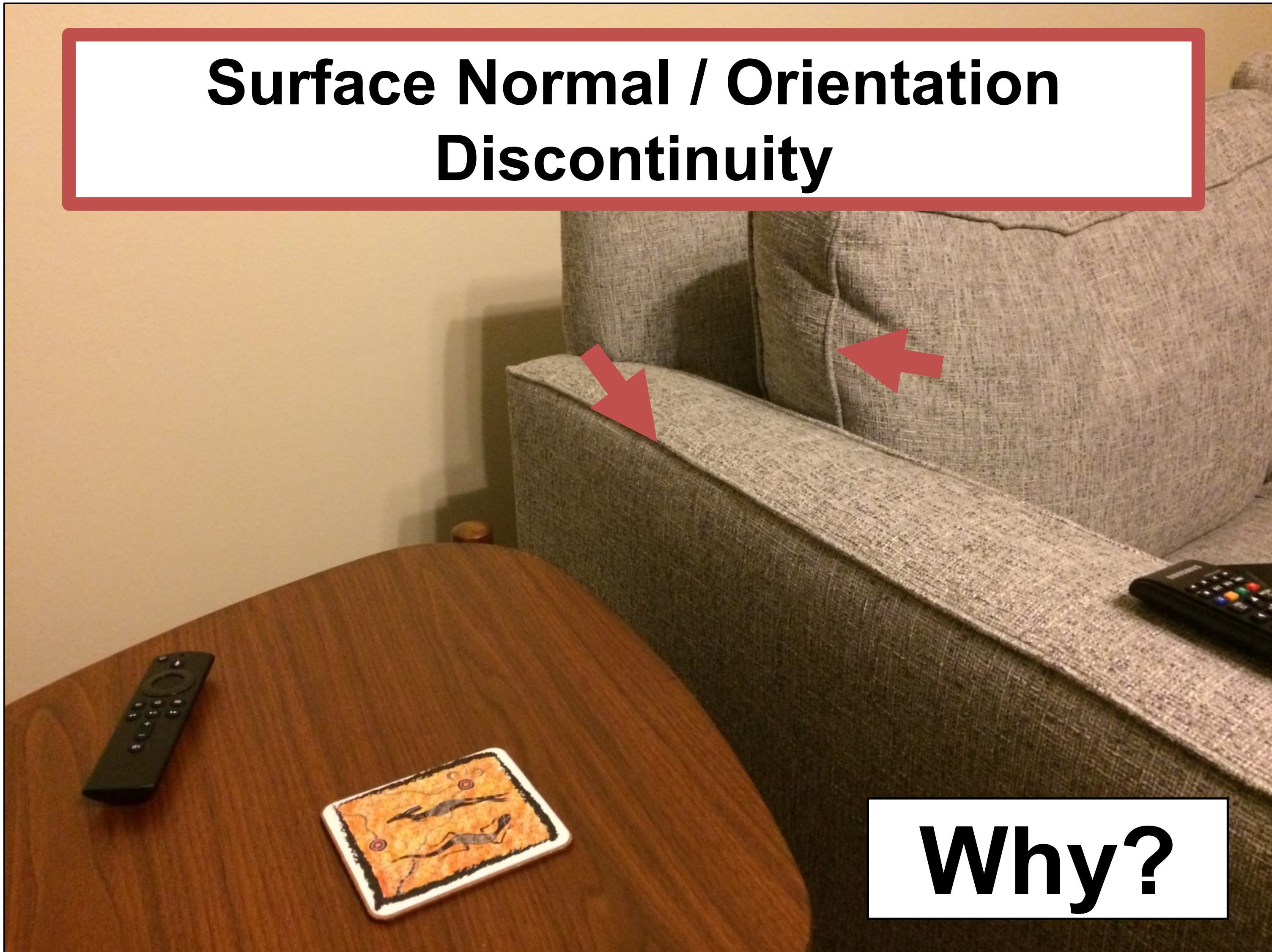


Where do Edges Come From?

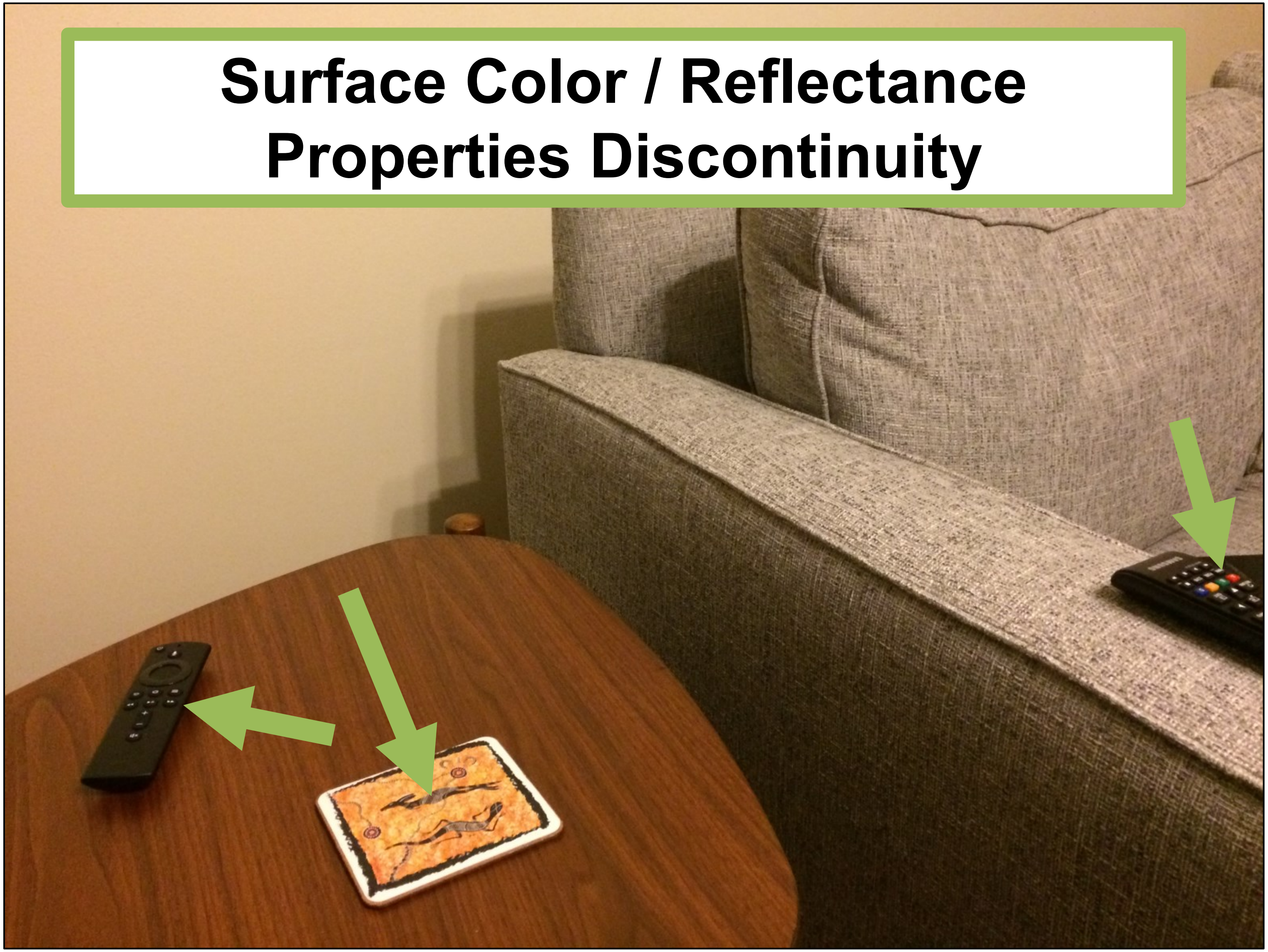
Where do Edges Come From?



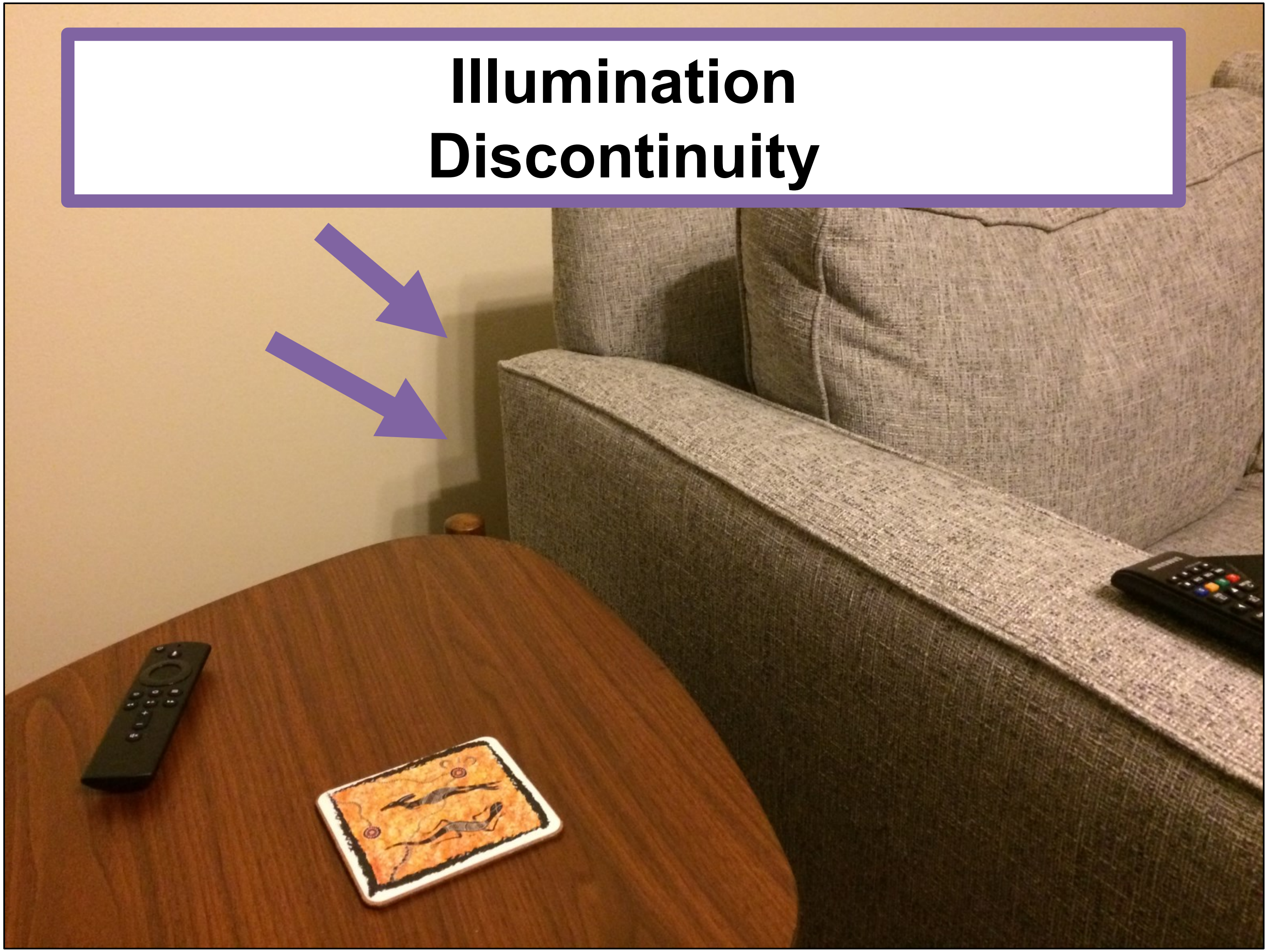
Where do Edges Come From?



Where do Edges Come From?



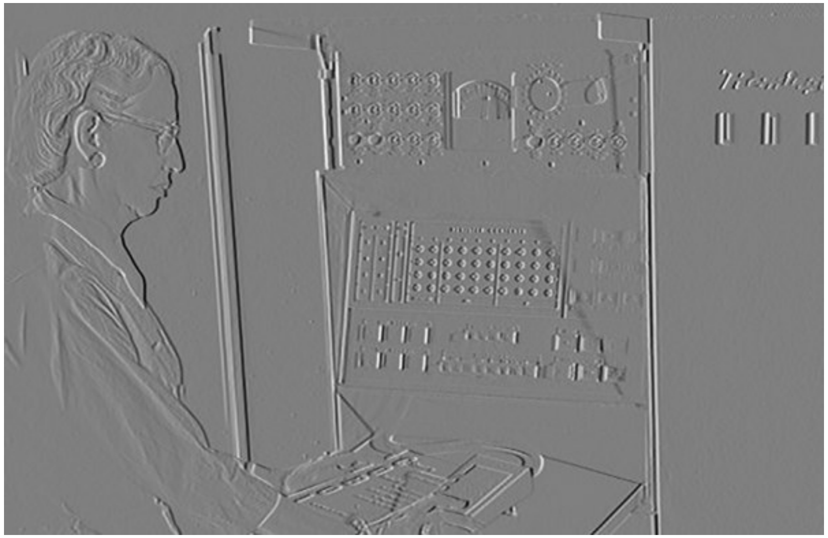
Where do Edges Come From?



Last Time

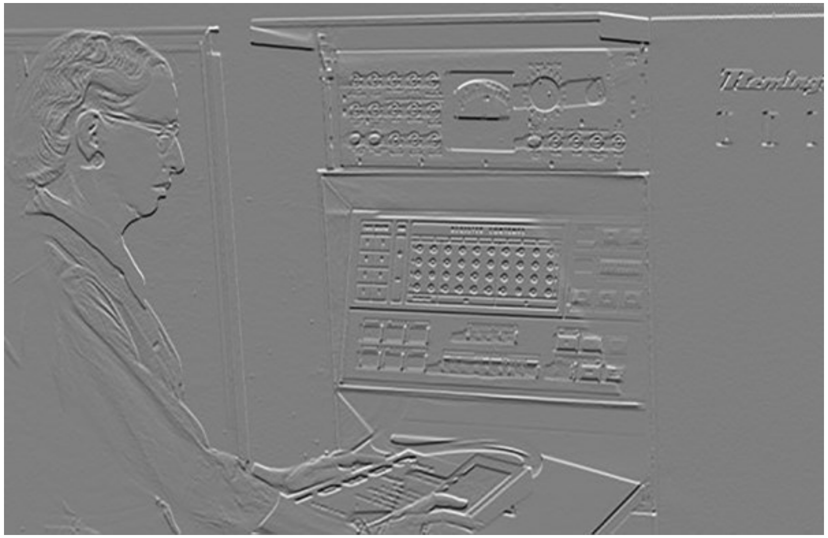
-1	0	1

Ix



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

Iy



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.

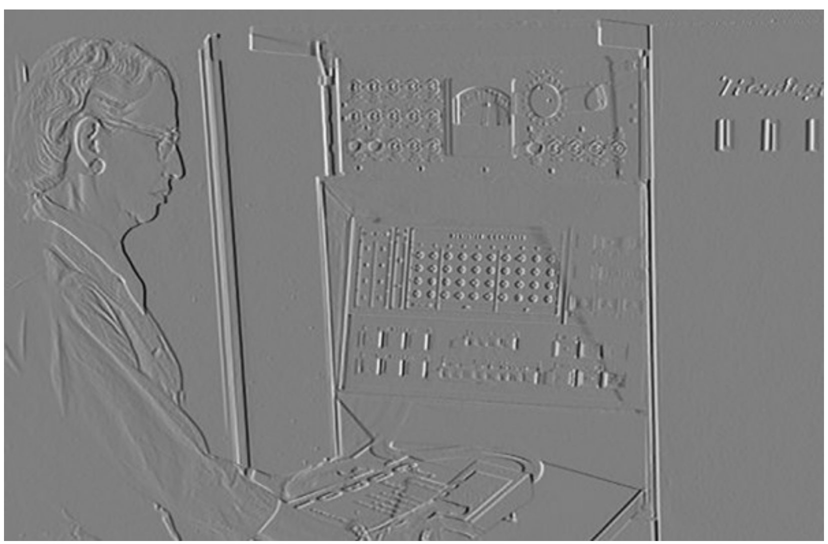
What Should I Know?

- Gradients are simply partial derivatives per-dimension: if $\boldsymbol{x}$ in $f(\boldsymbol{x})$ 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 $f(\boldsymbol{x}) \rightarrow \nabla_{\mathrm{f}}(\mathrm{a}) = \mathbf{0}$
- Reverse is not true, especially in high-dimensional spaces

Last Time

-1	0	1

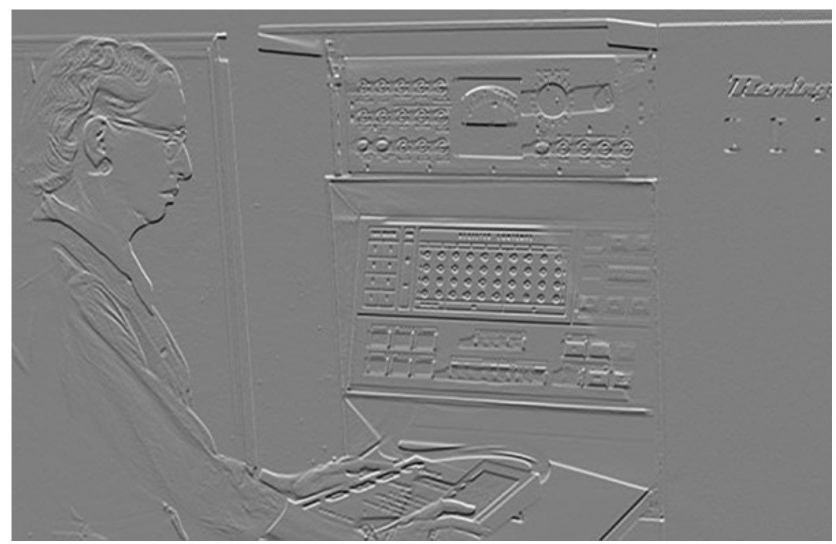
Ix



-1	0	1

T

Iy



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:

-1	1

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

Another one:

-1	0	1

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

Other Differentiation Operations

	Horizontal	Vertical
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Why might people use these compared to [-1,0,1]?

Images as Functions or Points

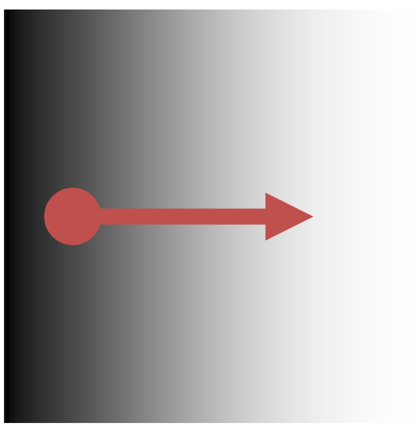
Key idea: can treat image as a point in $R^{(H \times W)}$ or as a function of x,y.

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

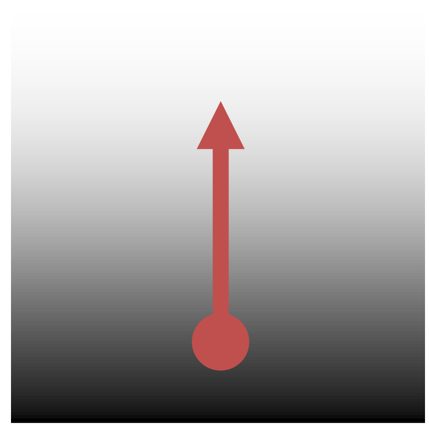
How much the intensity of the image changes as you go horizontally at (x,y)
(Often called Ix)

Image Gradient Direction

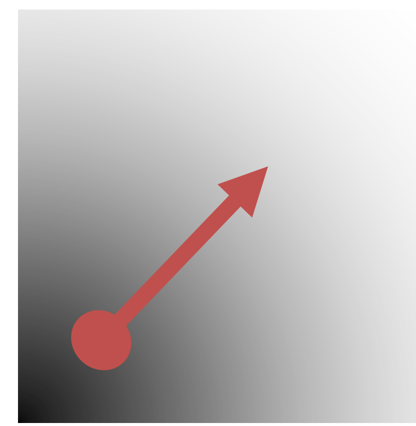
Some gradients



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



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



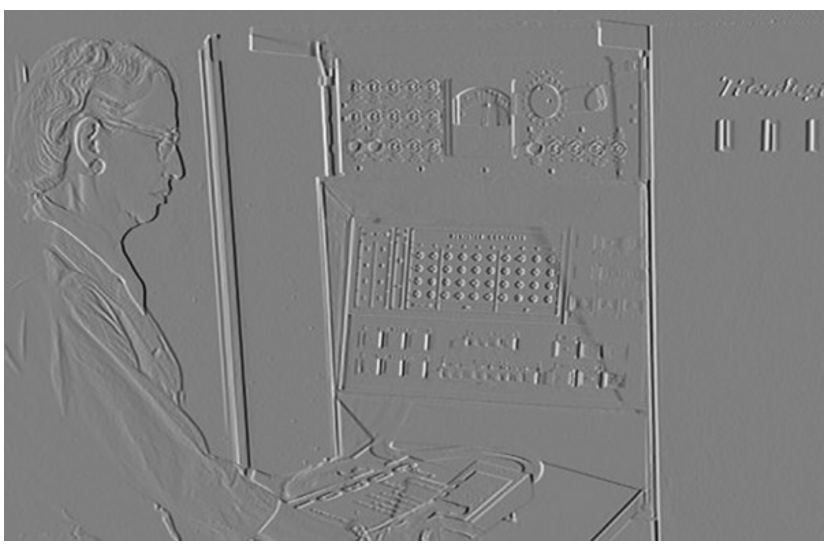
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Figure Credit: S. Seitz

Image Gradient

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

I_x



I_y

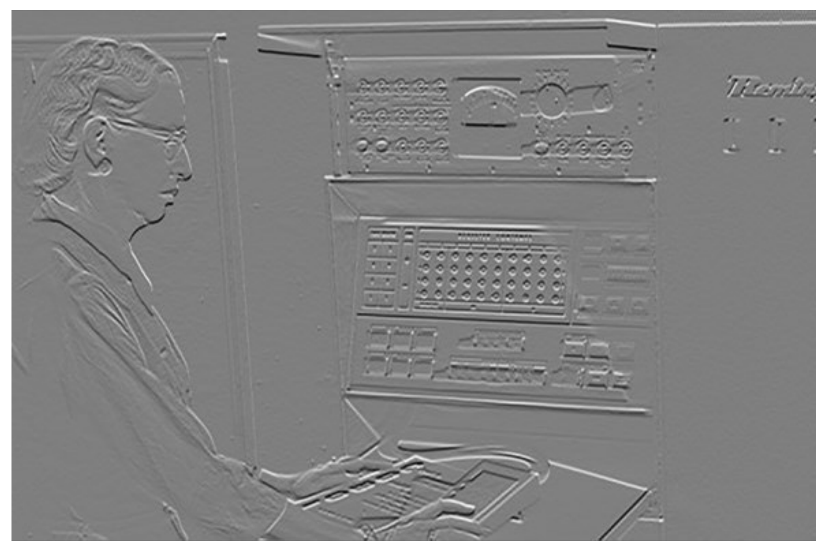
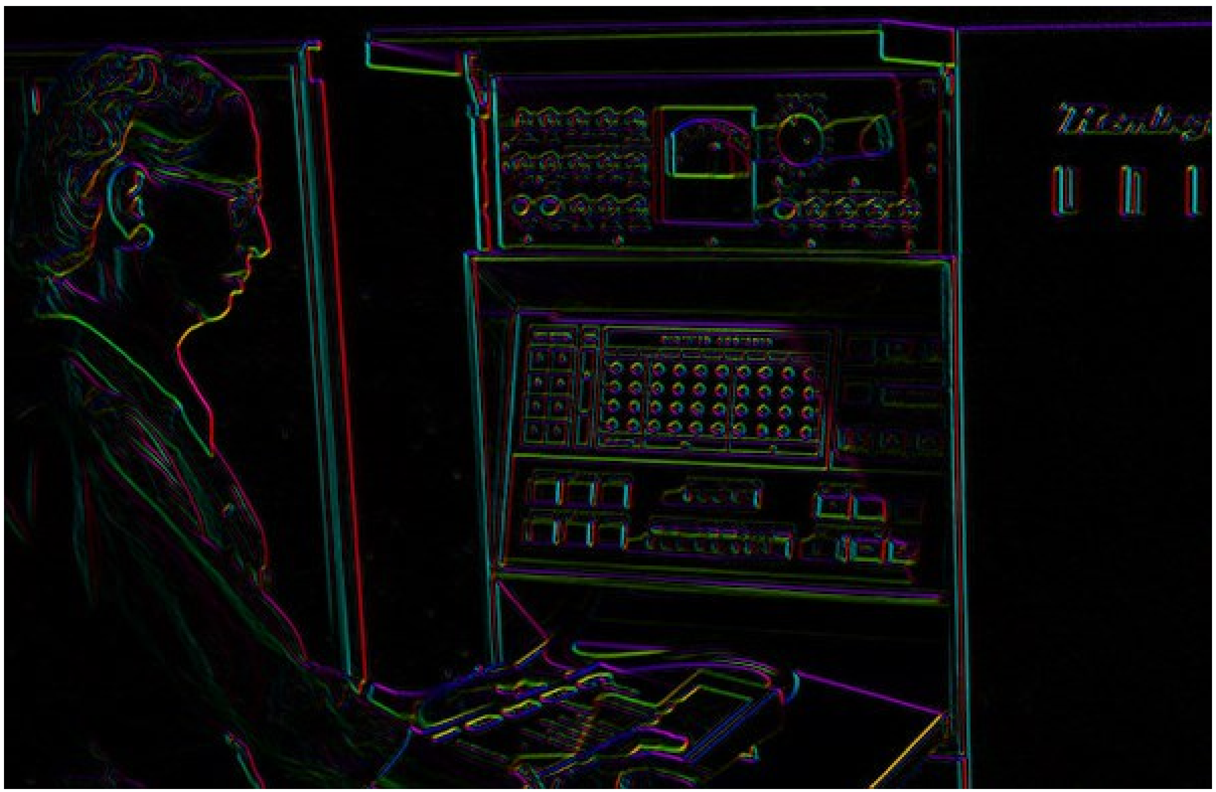


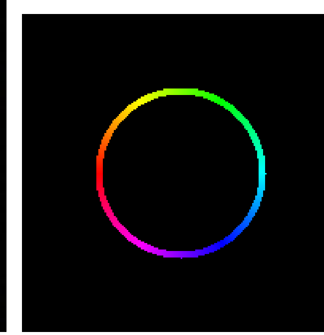
Image Gradient

$(Ix^2 + Iy^2)^{1/2}$: magnitude

Image Gradient

atan2(Iy,Ix): orientation

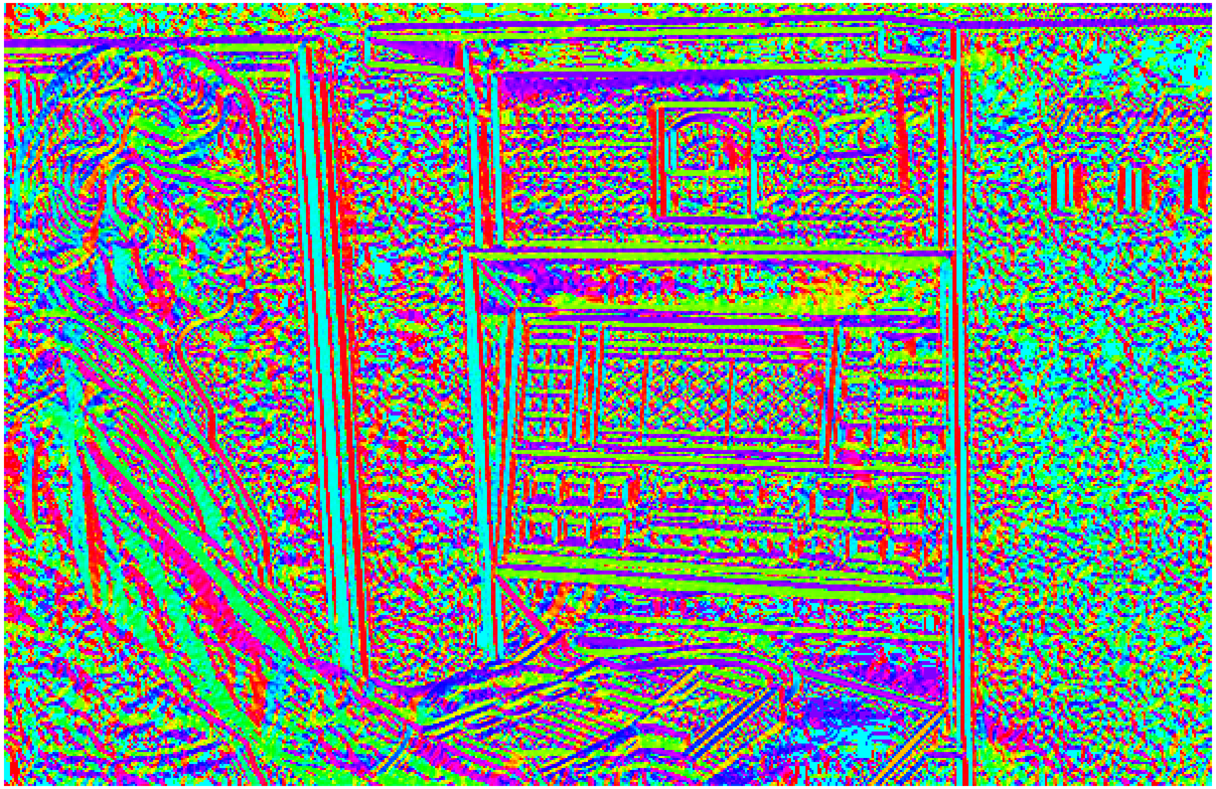


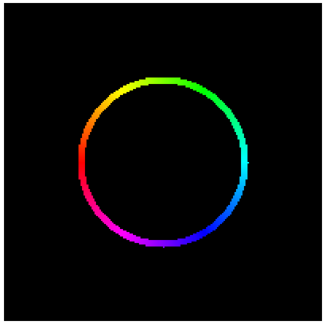


I'm making the lightness equal to gradient magnitude

Image Gradient

atan2(Iy,Ix): orientation



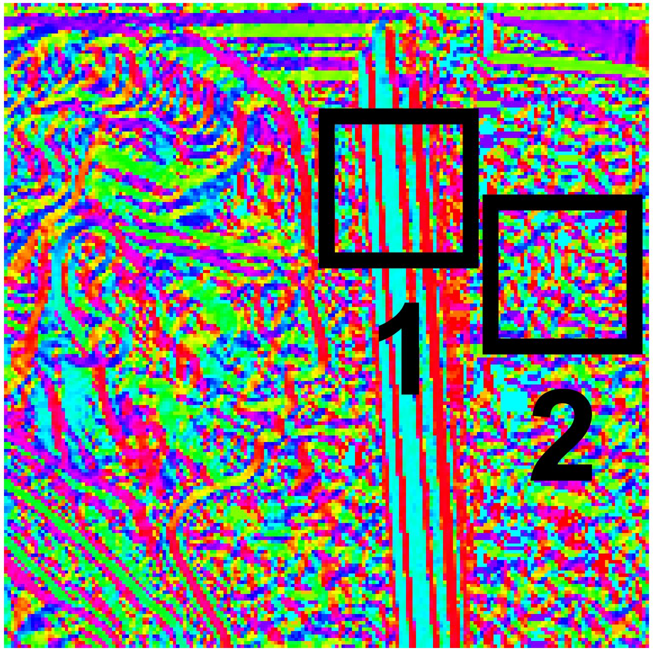


Now I'm showing *all* the gradients

Image Gradient

atan2(Iy,Ix): orientation

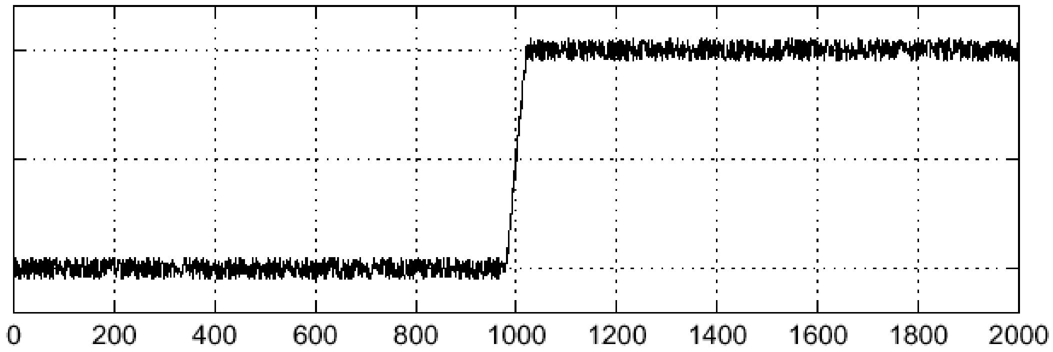
Why is there structure at 1 and not at 2?



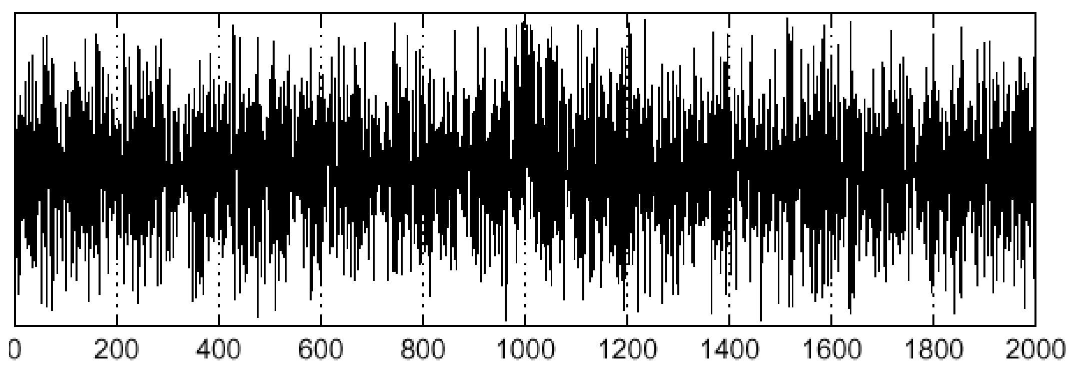
Noise

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

$f(x)$



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



Slide Credit: S. Seitz

Noise

Conv. image + per-pixel noise with

-1	0	1

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

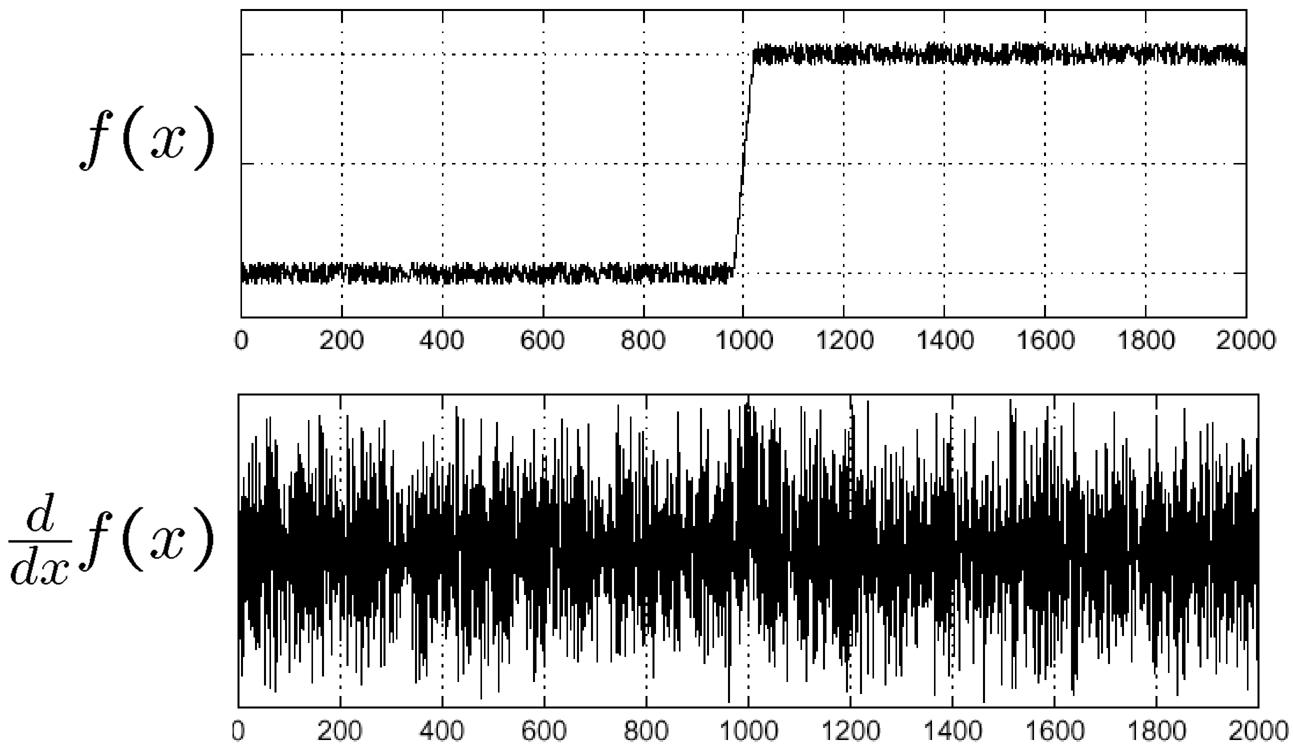
$$D_{i,j} = (I_{i,j+1} + \epsilon_{i,j+1}) - (I_{i,j-1} + \epsilon_{i,j-1})$$

$$D_{i,j} = \underbrace{(I_{i,j+1} - I_{i,j-1})}_{\text{True difference}} + \underbrace{\epsilon_{i,j+1} - \epsilon_{i,j-1}}_{\text{Sum of 2 Gaussians}}$$

$\epsilon_{i,j} - \epsilon_{k,l} \sim N\left(0, 2\sigma^2\right) \rightarrow$ Variance doubles!

Noise

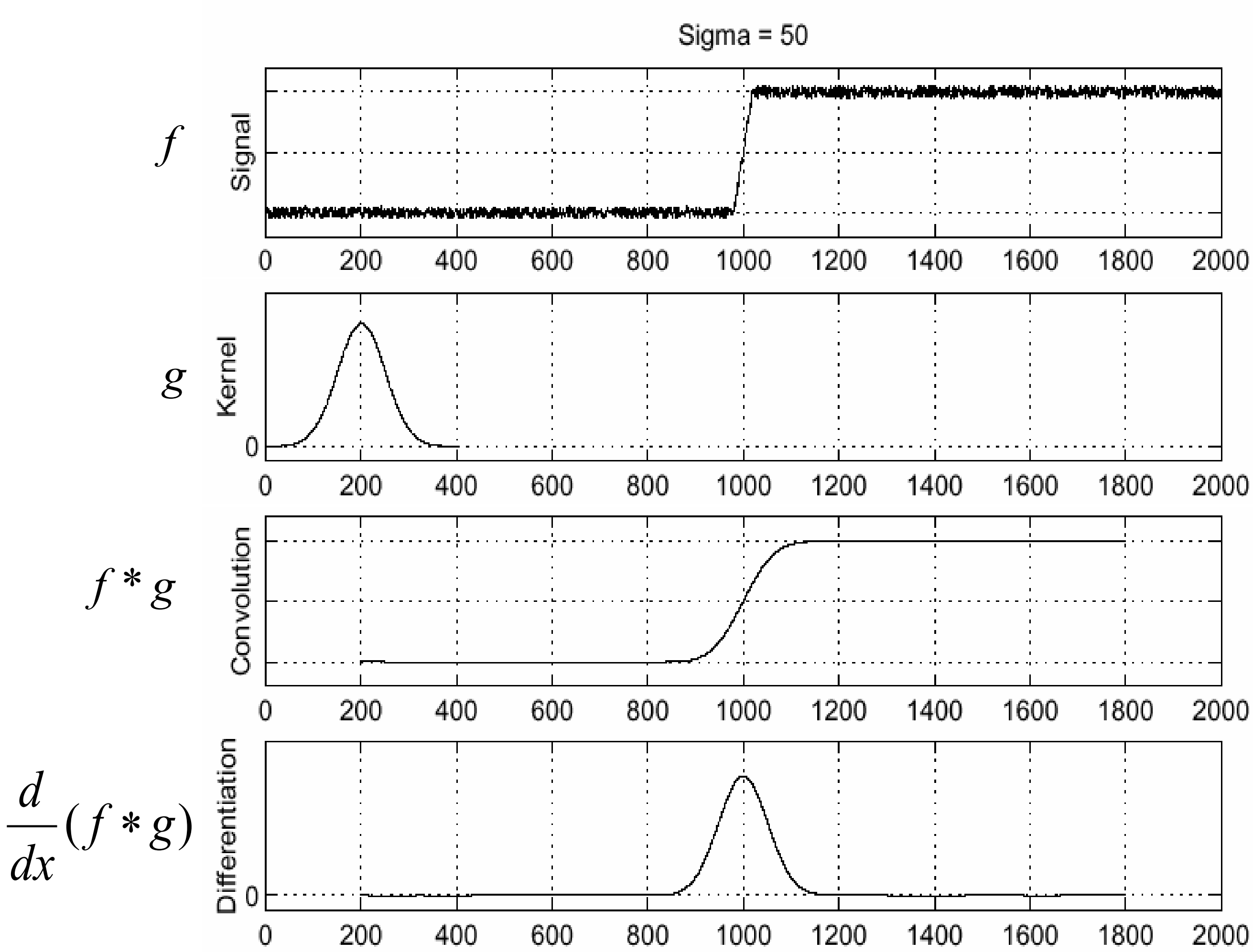
Consider a row of f(x,y) (i.e., make y constant)



How can we use the last class to fix this?

Slide Credit: S. Seitz

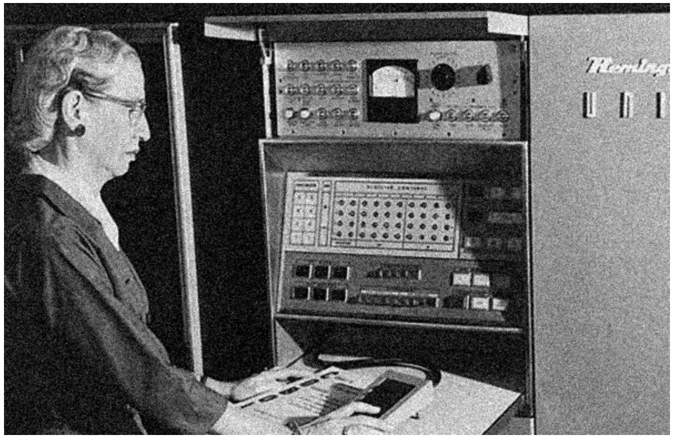
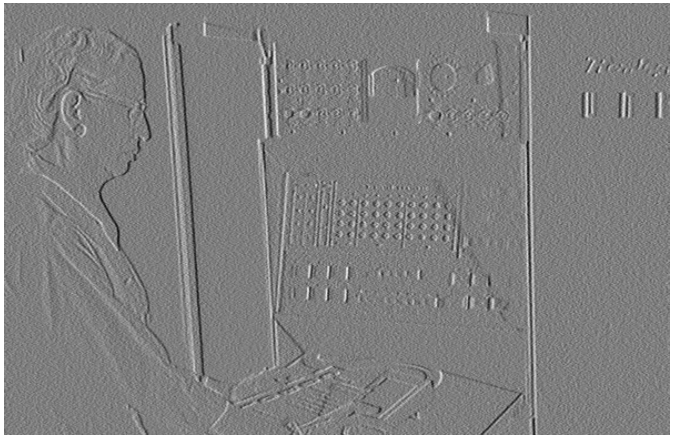
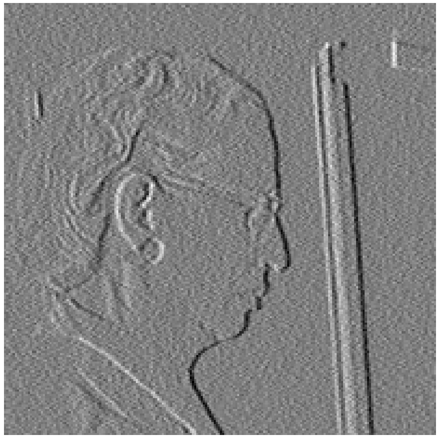
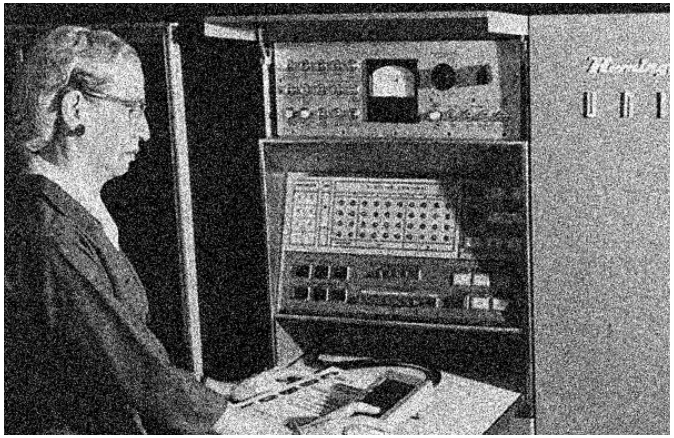
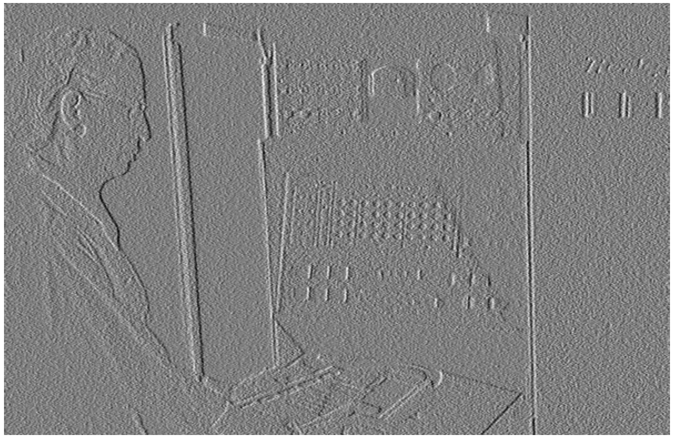
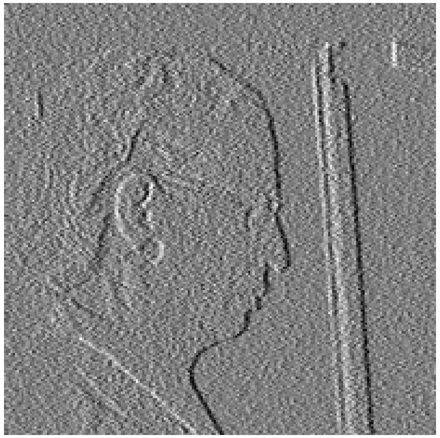
Handling Noise



Slide Credit: S. Seitz

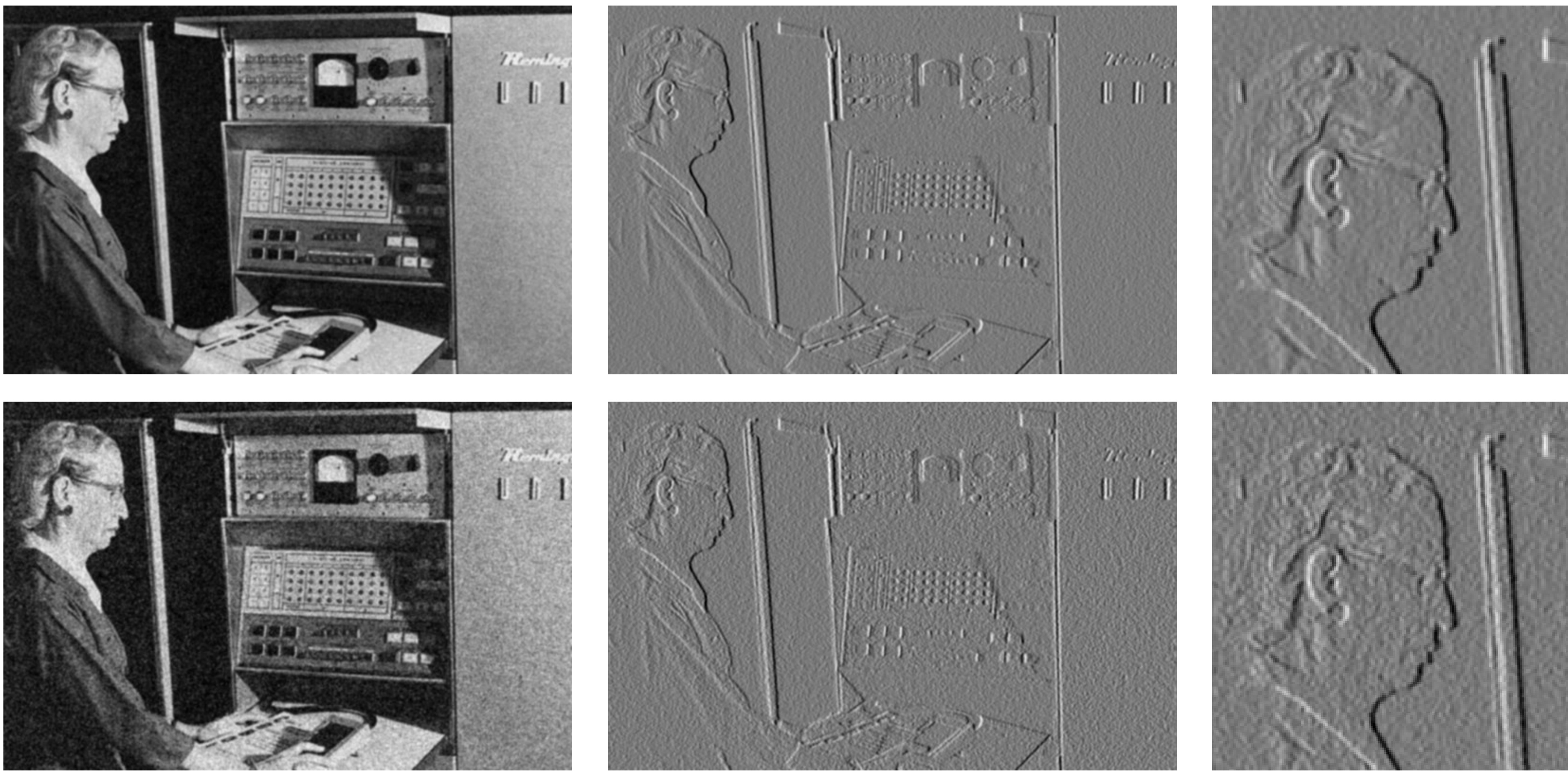
Noise in 2D

Noisy Input | Ix via [-1,01] | Zoom

Noise + Smoothing

Smoothed Input | Ix via [-1,01] | Zoom



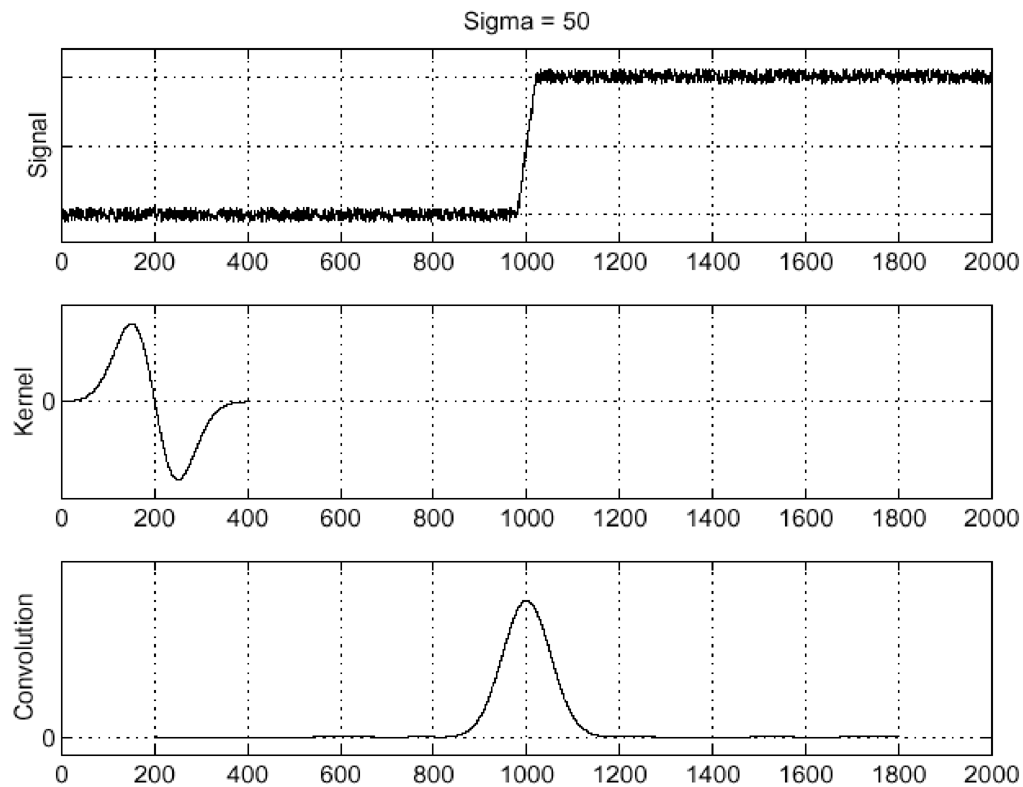
Let's Make It One Pass (1D)

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

f

$\frac{d}{dx} g$

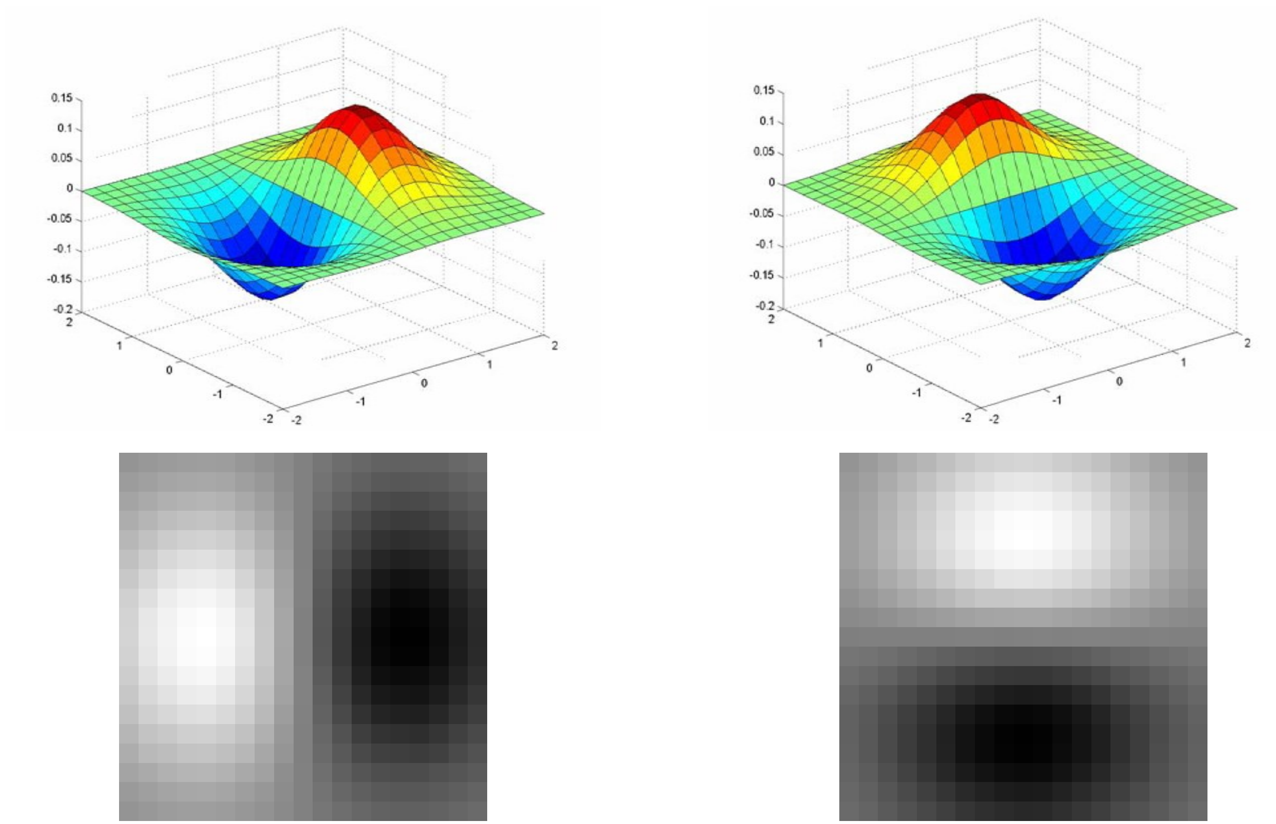
$f * \frac{d}{dx} g$



Slide Credit: S. Seitz

Let’s Make It One Pass (2D)

Gaussian Derivative Filter

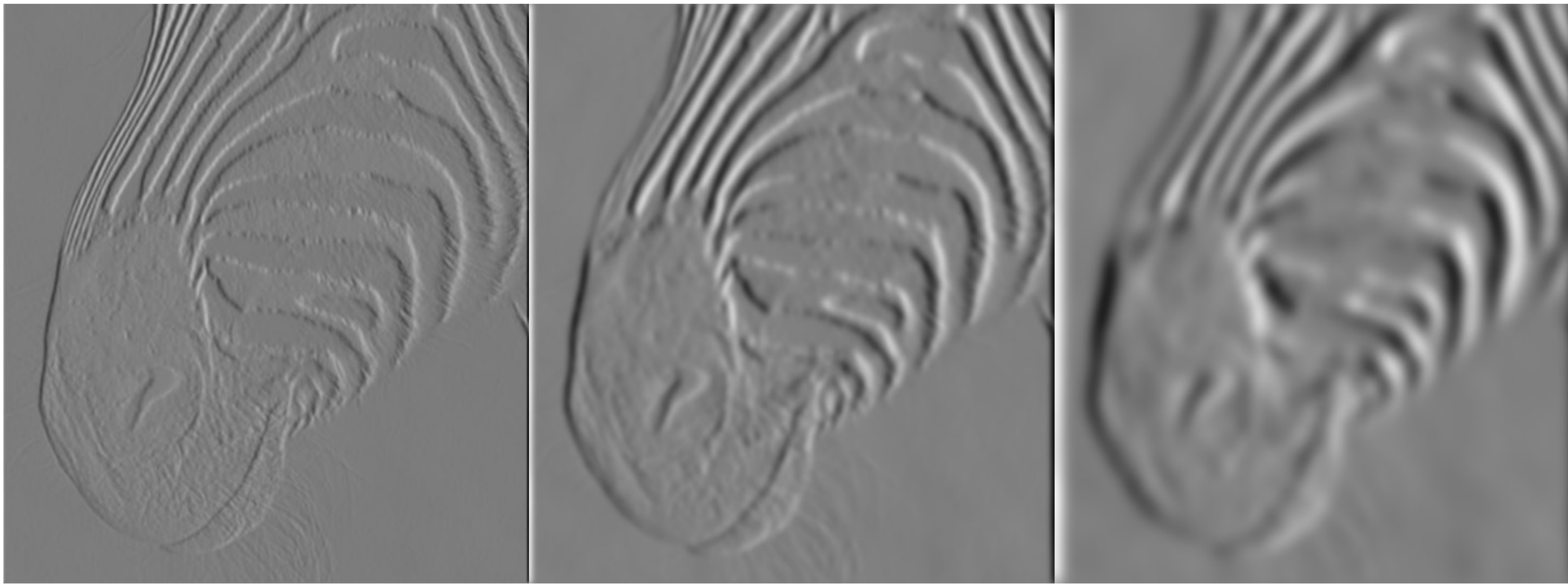


Which one finds the X direction?

Slide Credit: L. Lazebnik

Applying the Gaussian Derivative

1 pixel 3 pixels 7 pixels

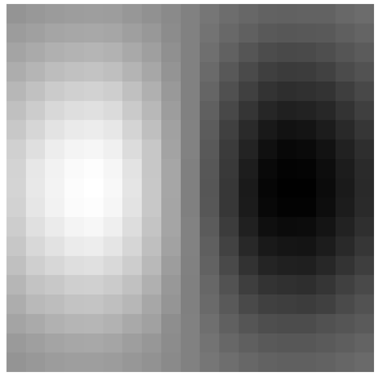


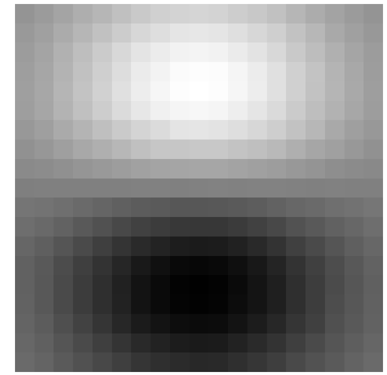
Removes noise, but blurs edge

Slide Credit: D. Forsyth

Compared with the Past

Gaussian Derivative



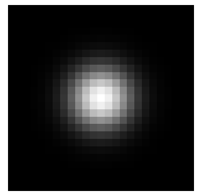
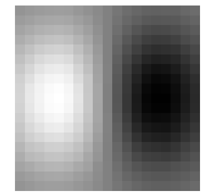
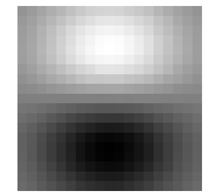


Sobel Filter

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Why would anybody use the bottom filter?

Filters We’ve Seen

	Smoothing	Derivative
		 
Example	Gaussian	Deriv. of gauss
Goal	Remove noise	Find edges
Only +?	**Yes**	**No**
Sums to	1	0

Why sum to 1 or 0, intuitively?

Slide Credit: J. Deng

Problems

Image | human segmentation | gradient magnitude

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

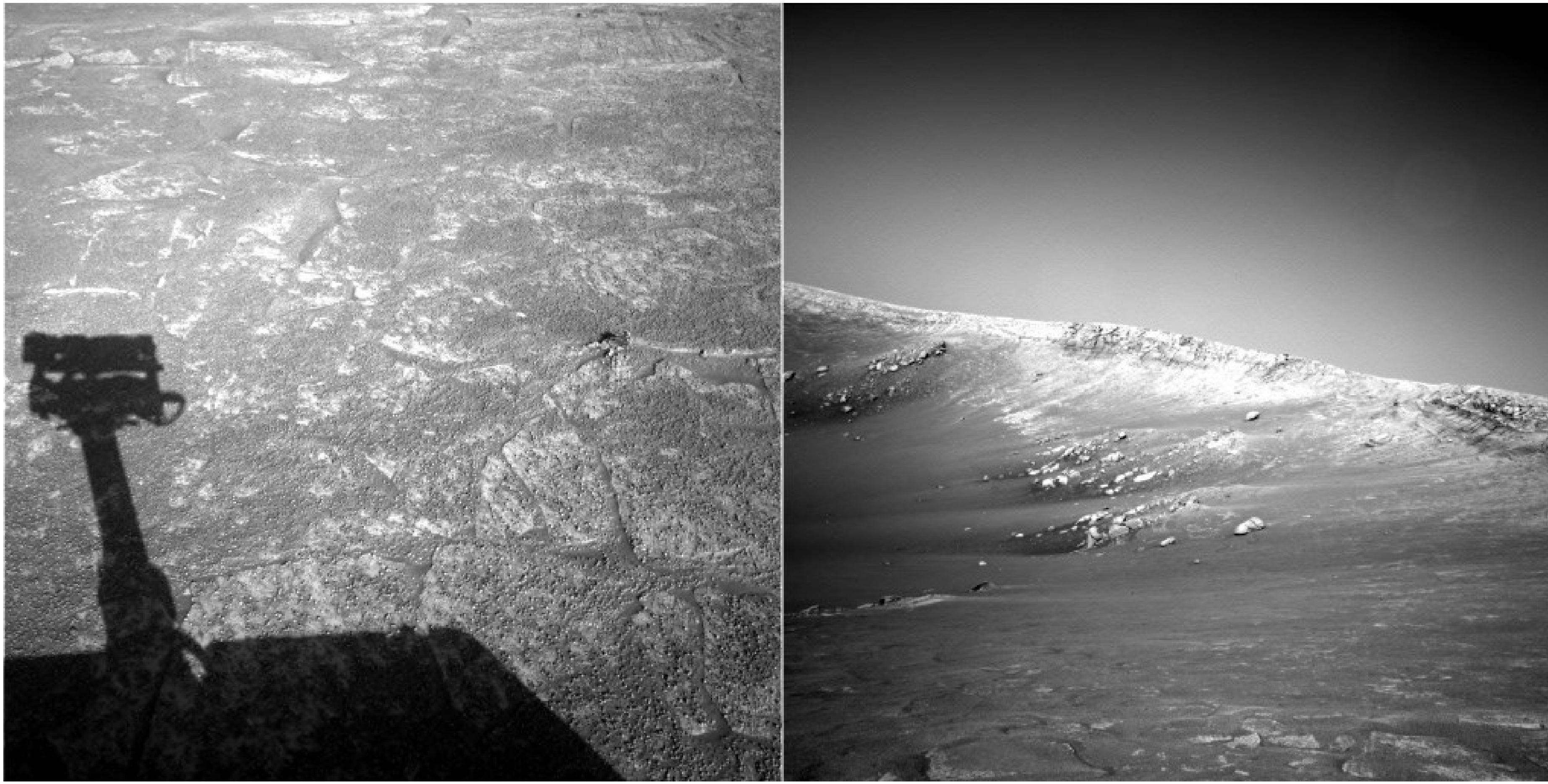
D: On North Campus

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

Property list: S. Lazebnik

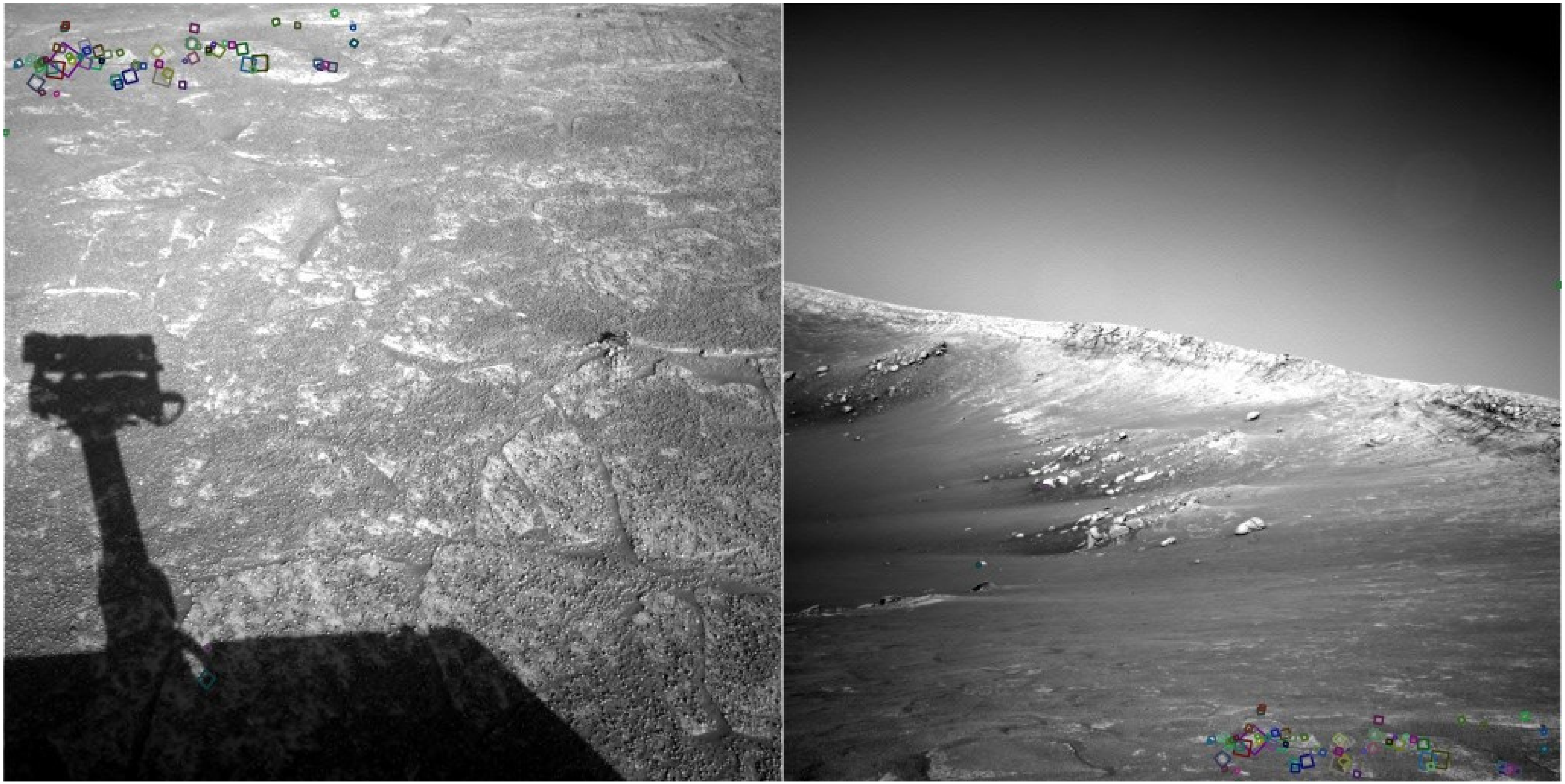
Example



Can you find the correspondences?

Slide credit: N. Snavely

Example Matches

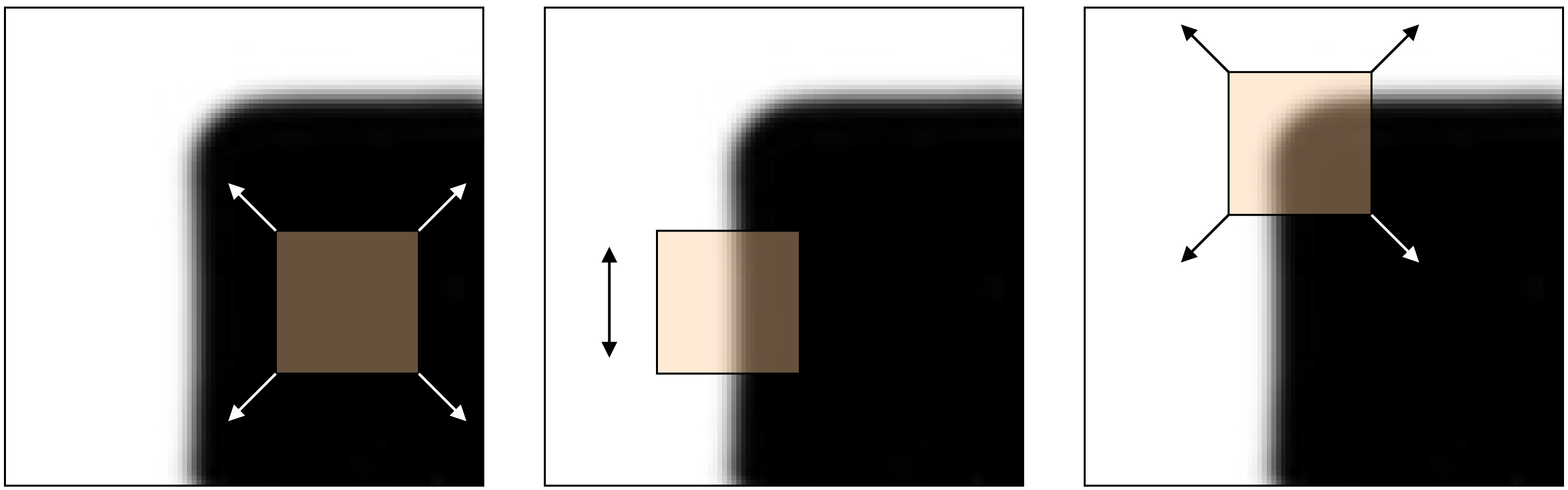


Look for the colored squares

Slide credit: N. Snavely

Basic Idea

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



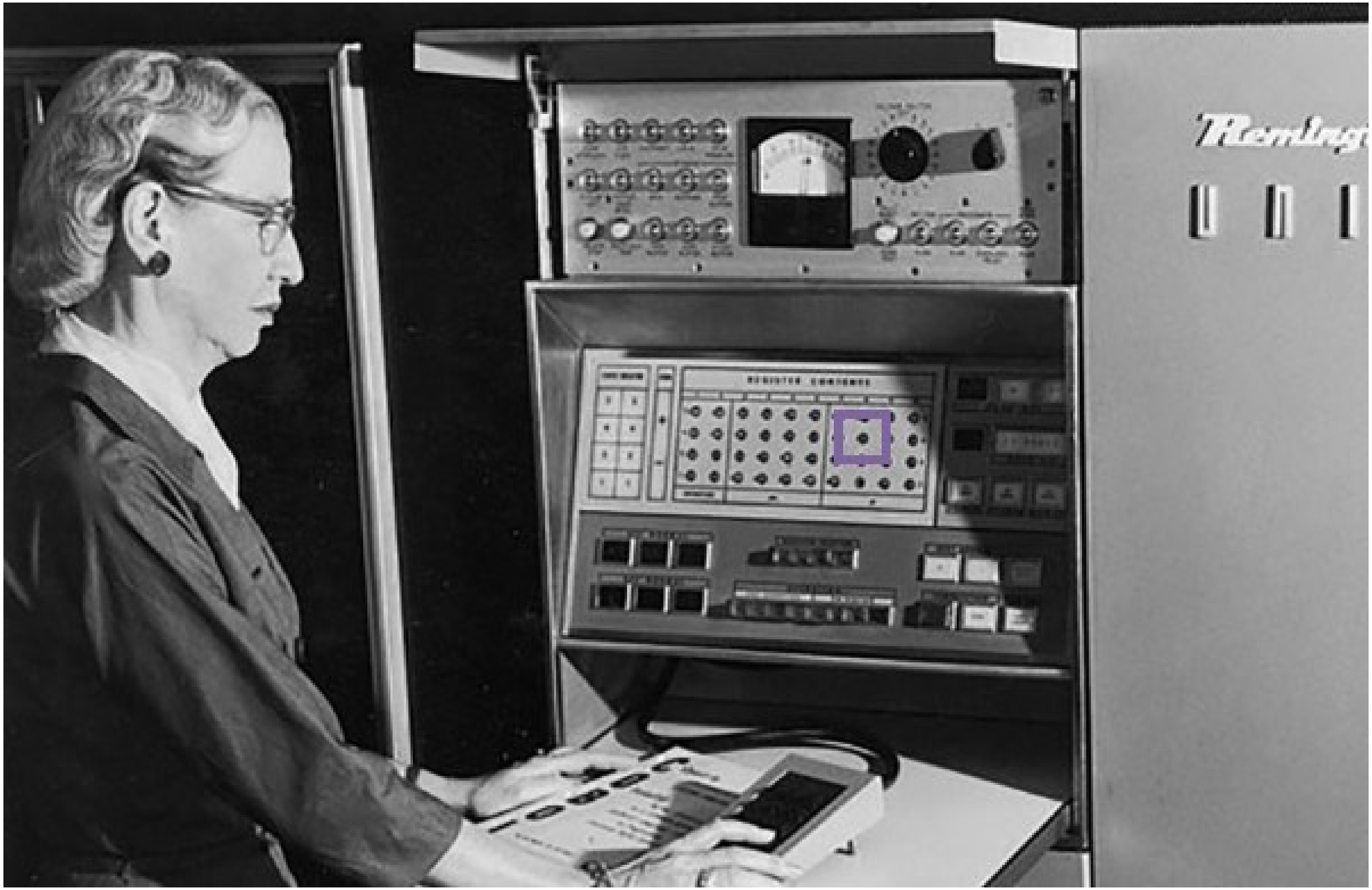
"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

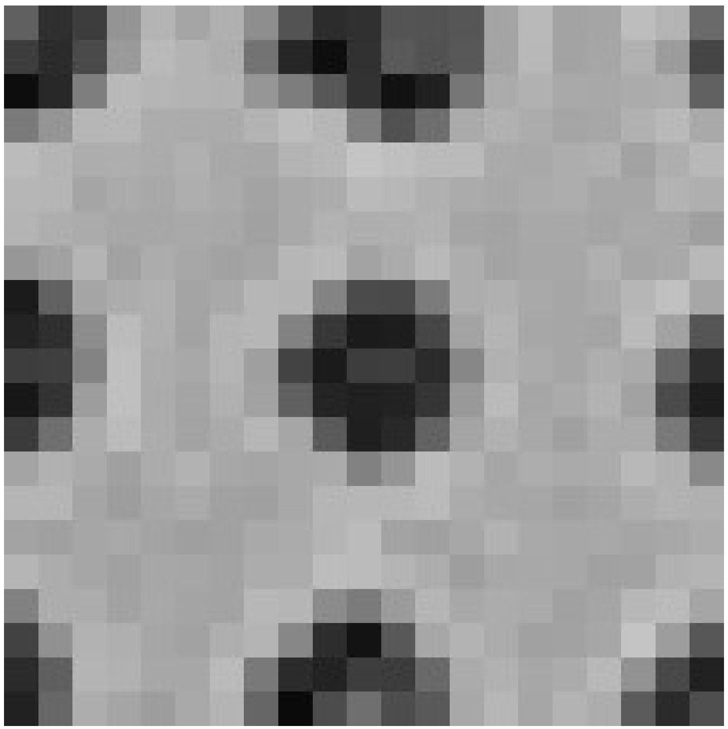
Slide Credit: S. Lazebnik

Formalizing Corner Detection



Formalizing Corner Detection

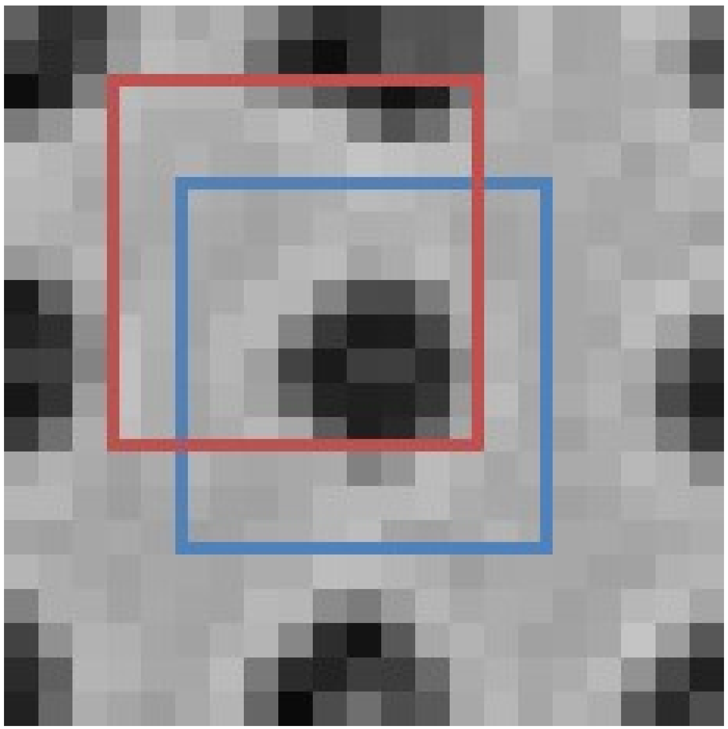
Zoom-In at x,y



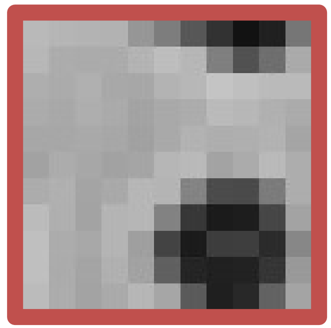
Original Image

Formalizing Corner Detection

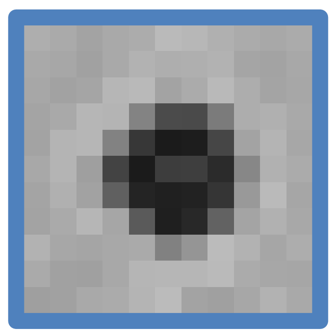
Zoom-In at x,y



Window without and with Offset



"Window"
At x+u, y+v
Here: u=-2,v=-3

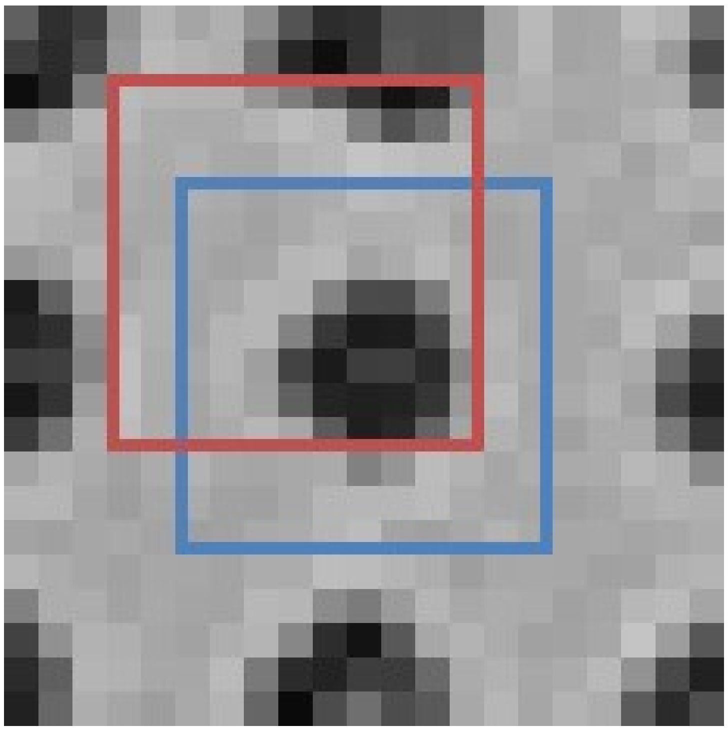


"Window"
At x, y

How might we measure similarity?

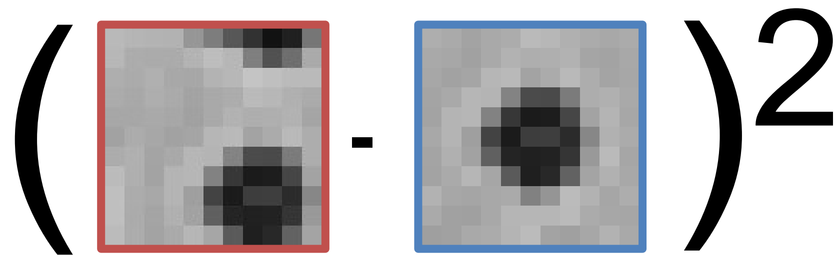
Formalizing Corner Detection

Zoom-In at x,y

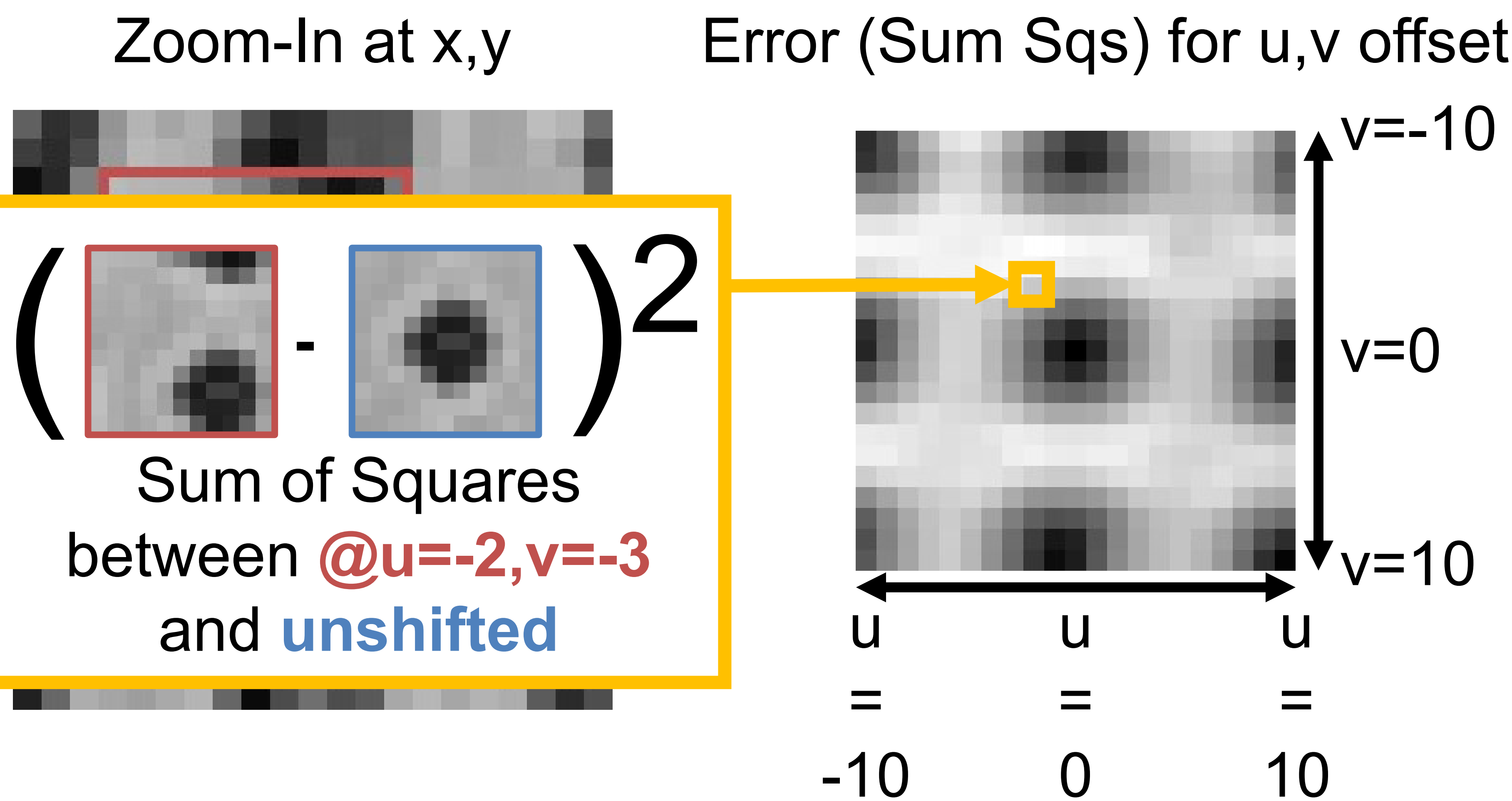


Error (Sum Sqs) for u,v offset

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

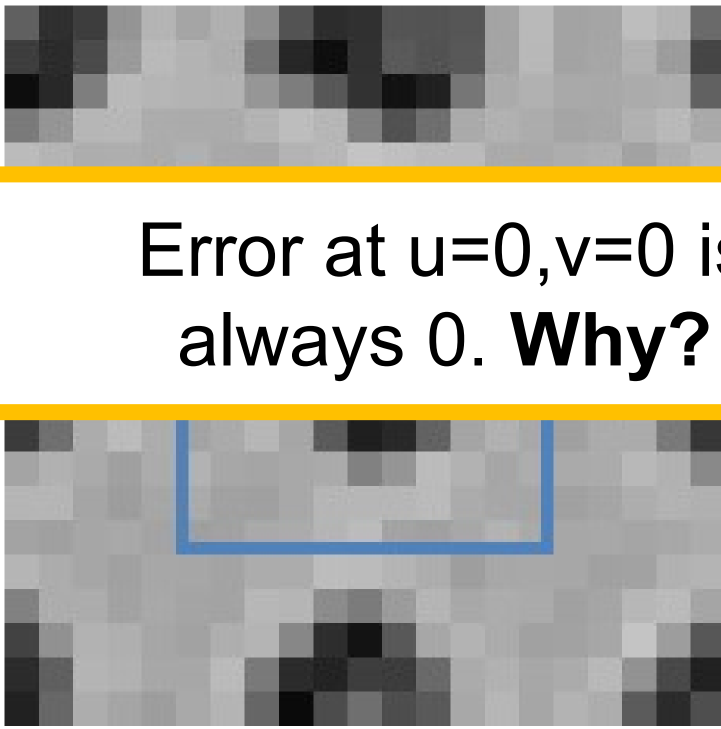


Formalizing Corner Detection

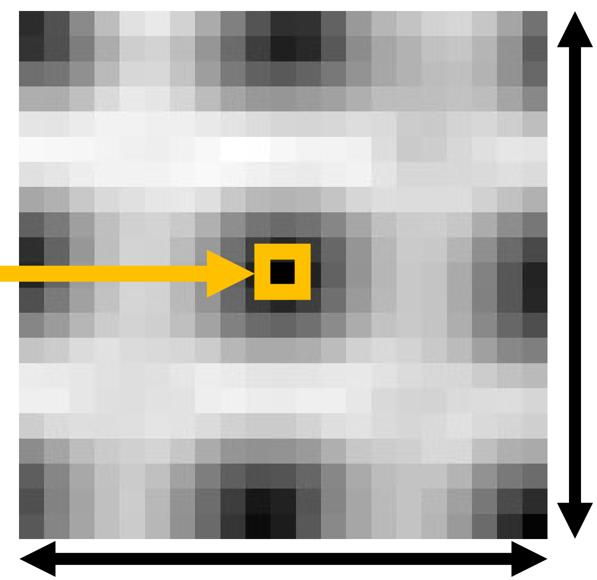


Formalizing Corner Detection

Zoom-In at x,y



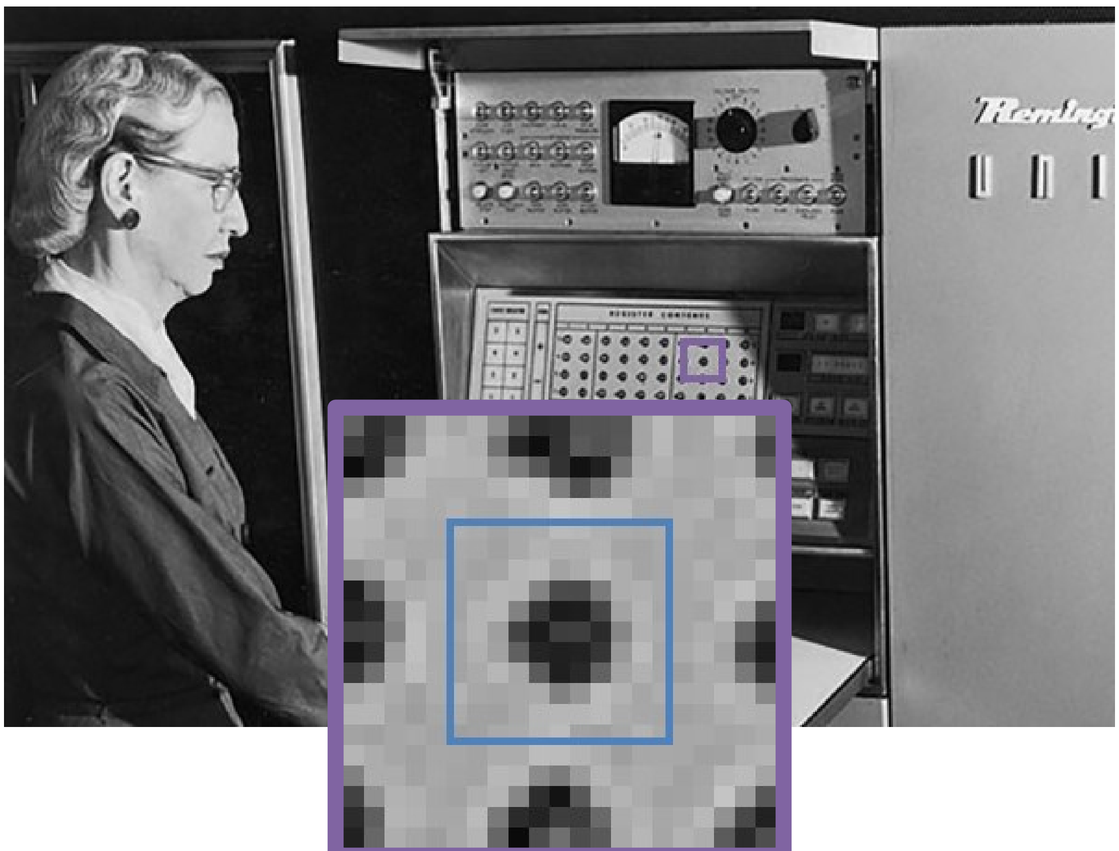
Error (Sum Sqs) for u,v offset



Error at u=0,v=0 is always 0. **Why?**

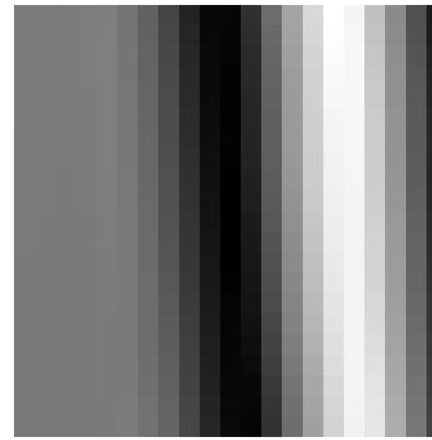
Match The Location and Plot

Original Image and Zoom-In

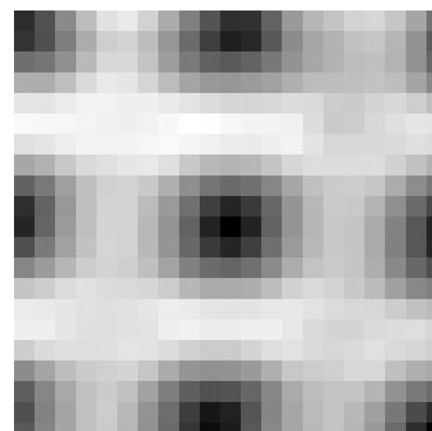


Error Options

A

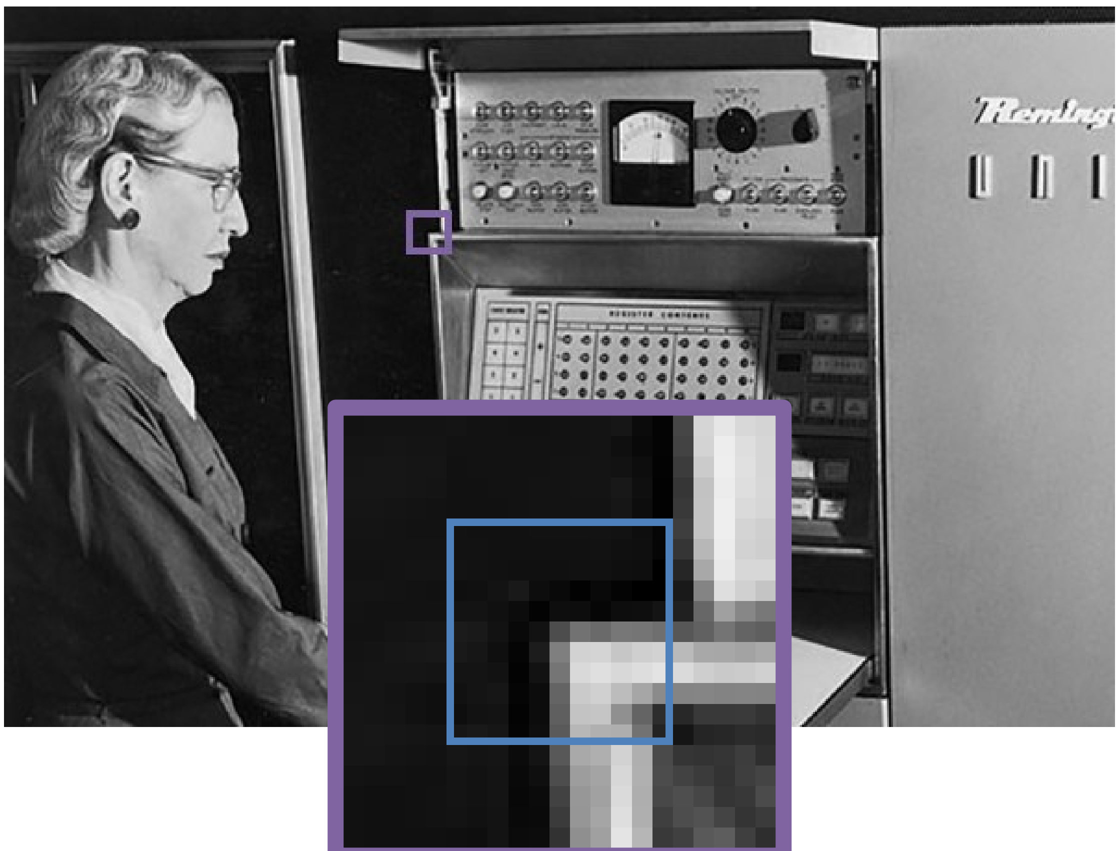


B



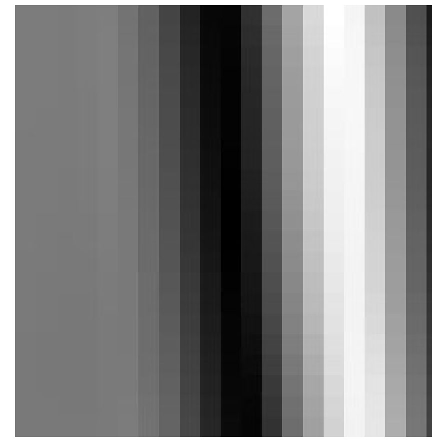
Match The Location and Plot

Original Image and Zoom-In

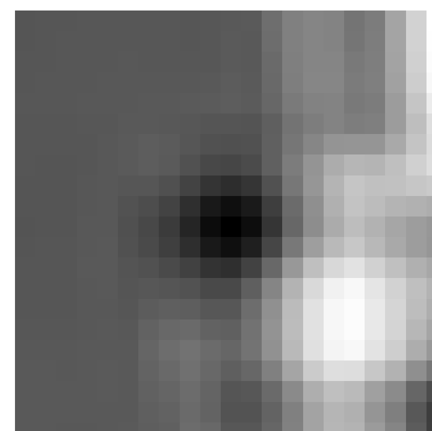


Error Options

A

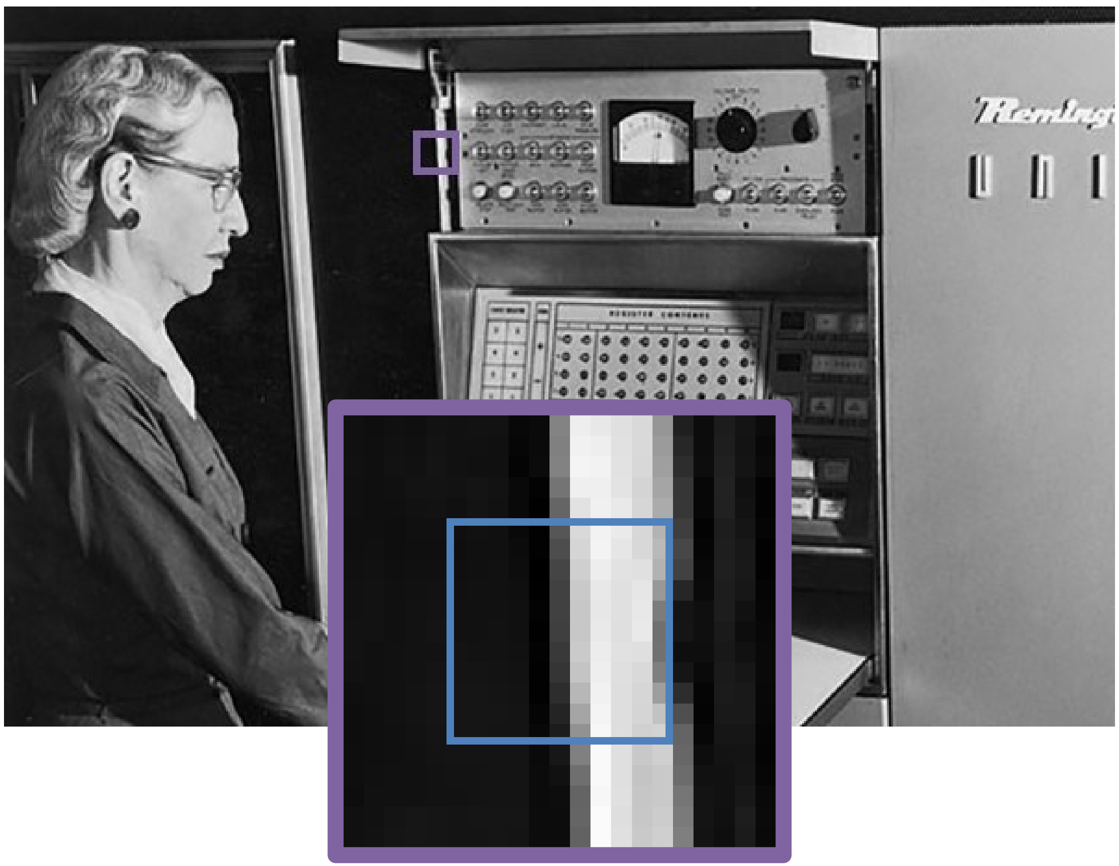


B



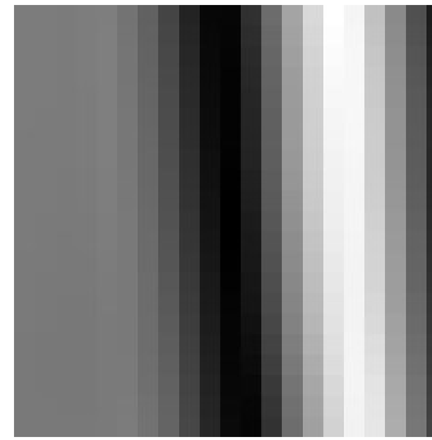
Match The Location and Plot

Original Image and Zoom-In

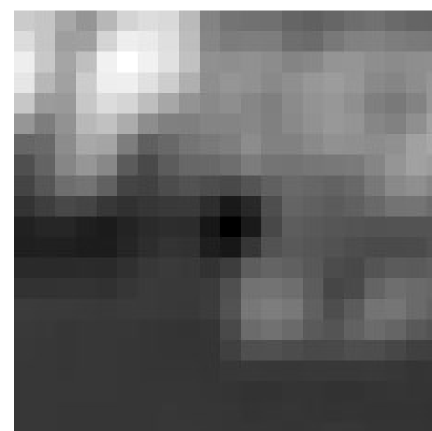


Error Options

A

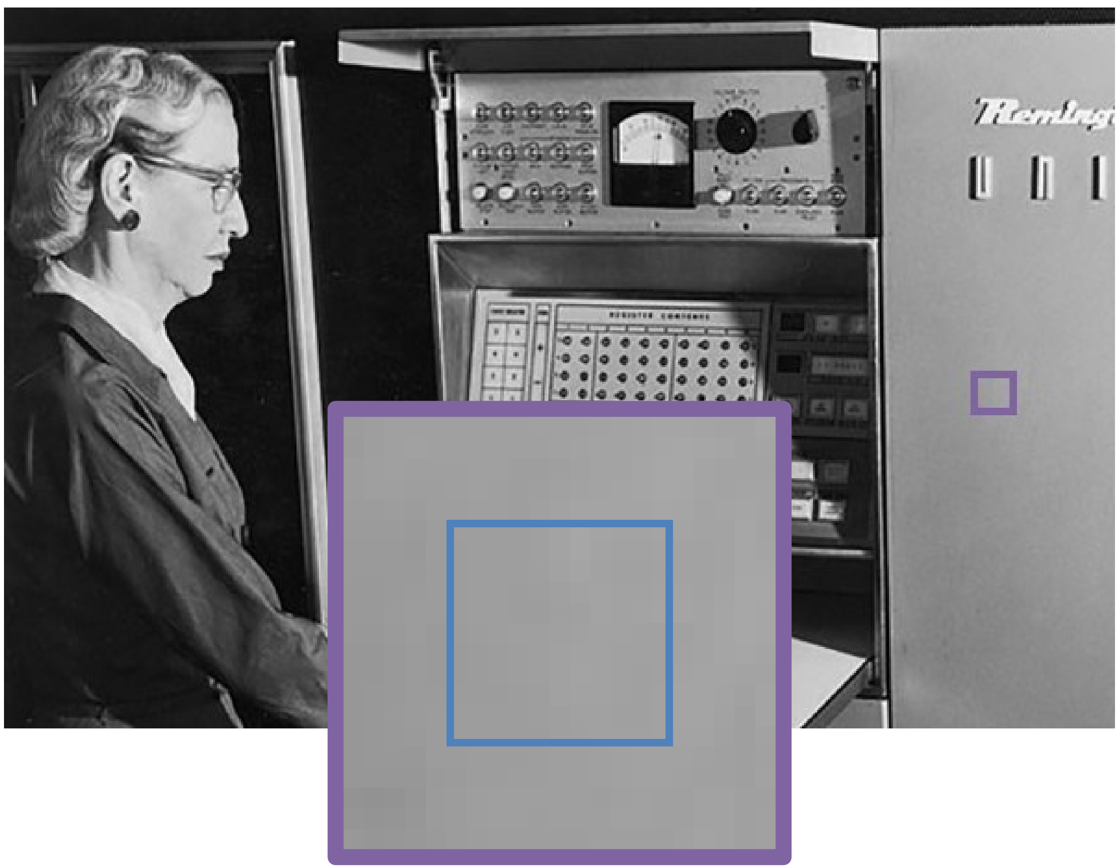


B



Match The Location and Plot

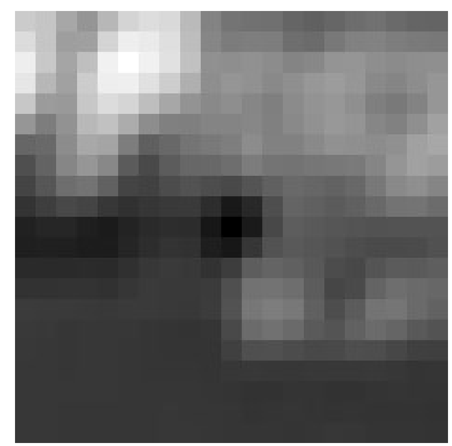
Original Image and Zoom-In



Error Options

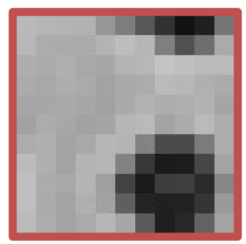
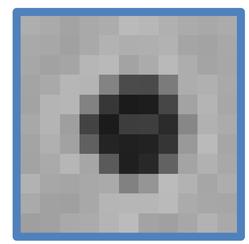
A

B



Ok But Back To Math

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

( - )^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: Ix = Ix at point (x,y), Iy = Iy at point (x,y)

Formalizing Corner Detection

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

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

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

For brevity: Ix = Ix at point (x,y), Iy = Iy at point (x,y)

Formalizing Corner Detection

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

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

$$= [u,v]\boldsymbol{M}[u,v]^T$$

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

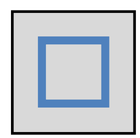
M is called the second moment matrix

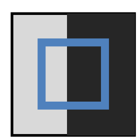
Intuitively what is M?

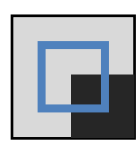
Pretend gradients are *either* vertical or horizontal at a pixel (so Ix Iy = 0)

Obviously Wrong!

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

a,b both small: flat  $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$

One big, other small: edge  $\begin{bmatrix} 50 & 0 \\ 0 & 0.1 \end{bmatrix}$ or $\begin{bmatrix} 0.1 & 0 \\ 0 & 50 \end{bmatrix}$

a,b both big: corner  $\begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}$

Intuitively what is M?

Pretend gradients are *either* vertical or horizontal at a pixel (so Ix Iy = 0)

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

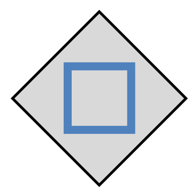
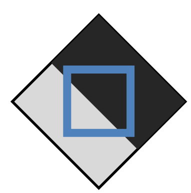
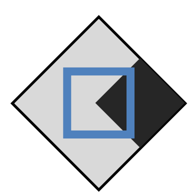
a,b both small: flat 

Image might be rotated by rotation θ!

One big, other small: edge 

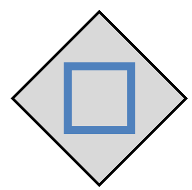
a,b both big: corner 

Intuitively what is M?

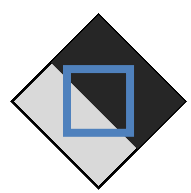
Pretend gradients are *either* vertical or horizontal at a pixel (so Ix Iy = 0)

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

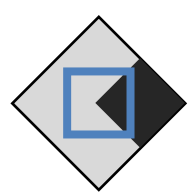
a,b both small: flat



One big, other small: edge



a,b both big: corner



If image rotated by rotation θ / matrix **V**

M will look like

$$\boldsymbol{V}^{-1} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \boldsymbol{V}$$

So What Now?

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

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

Given **M**, can decompose it into eigenvectors **V** and eigenvalues λ_1, λ_2 with $\mathbf{M} = \boldsymbol{V}^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \boldsymbol{V}$.

Really slow. Why?

So What Now?

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

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

Instead: compute quantity R from **M**

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

Easy fast formula for 2x2

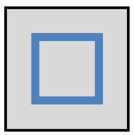
Fast – sum the diagonal

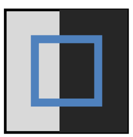
Empirical value, usually 0.04-0.06

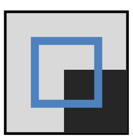
So What Now?

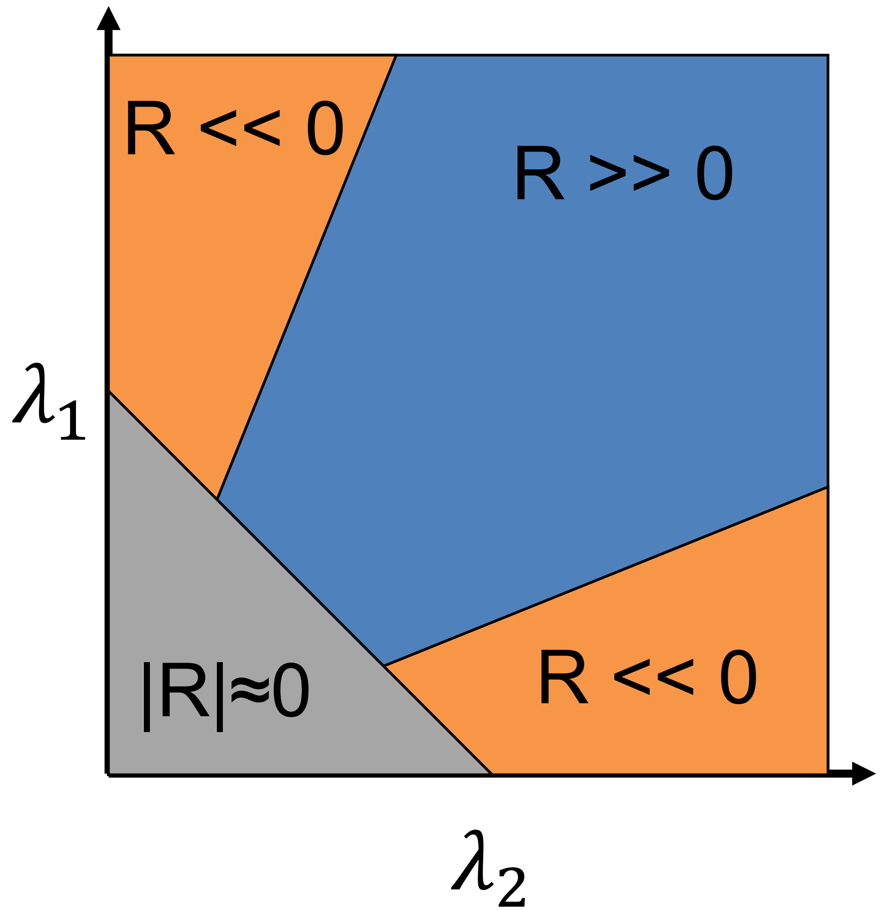
R tells us whether we're at a corner, edge, or flat

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

flat  $\lambda_1, \lambda_2 \approx 0$

edge  $\lambda_1 \gg \lambda_2 \gg 0$
$\lambda_2 \gg \lambda_1 \gg 0$

corner  $\lambda_1 \approx \lambda_2 \gg 0$



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.

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

In Practice

1. Compute partial derivatives Ix, Iy per pixel
2. Compute **M** at each pixel, using Gaussian weighting w

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

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Slide credit: S. Lazebnik

In Practice

1. Compute partial derivatives Ix, Iy per pixel
2. Compute **M** at each pixel, using Gaussian weighting w
3. Compute response function R

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

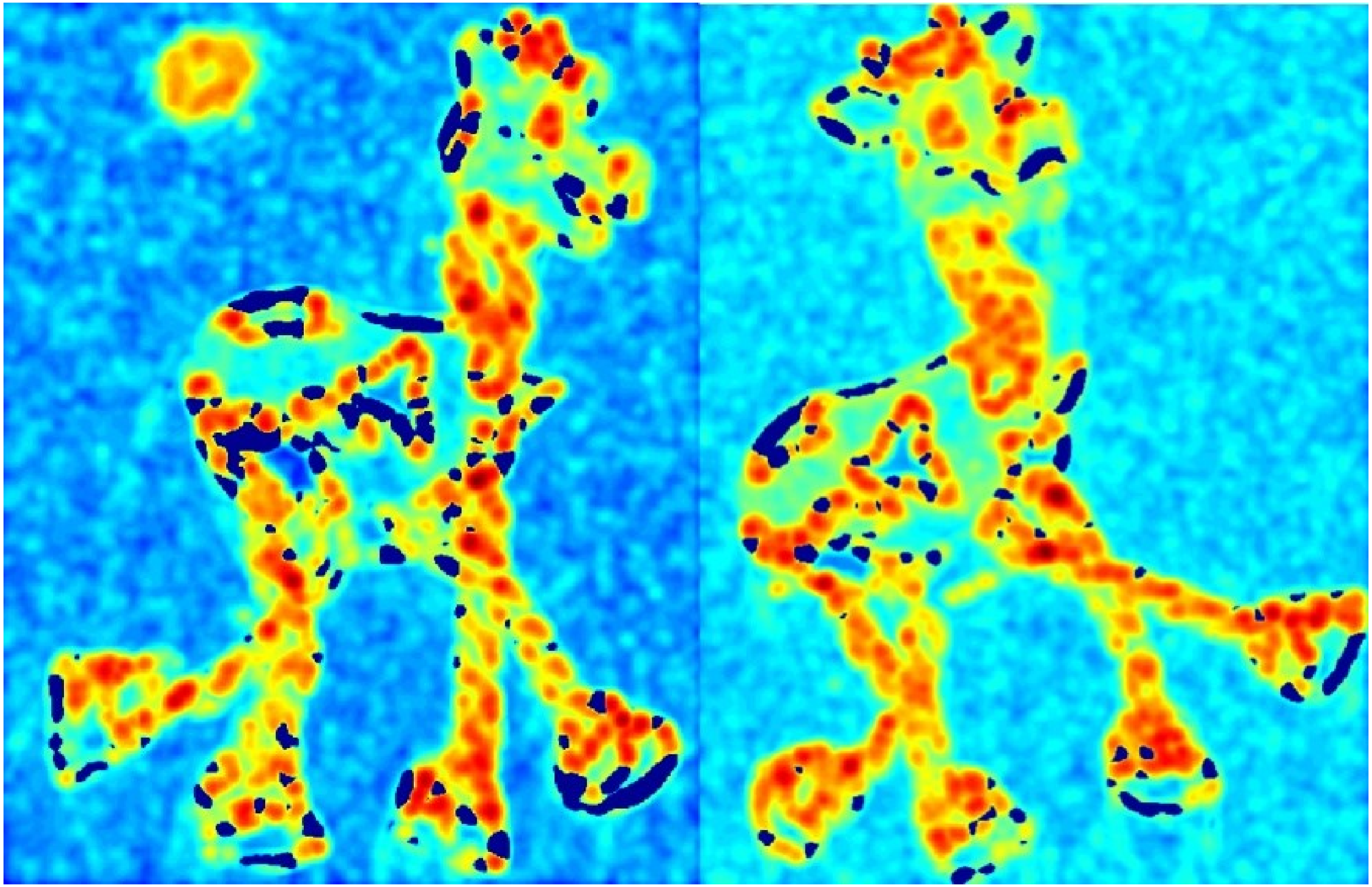
C.Harris and M.Stephens. “A Combined Corner and Edge Detector.” *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Slide credit: S. Lazebnik

Computing R

Slide credit: S. Lazebnik

Computing R



Slide credit: S. Lazebnik

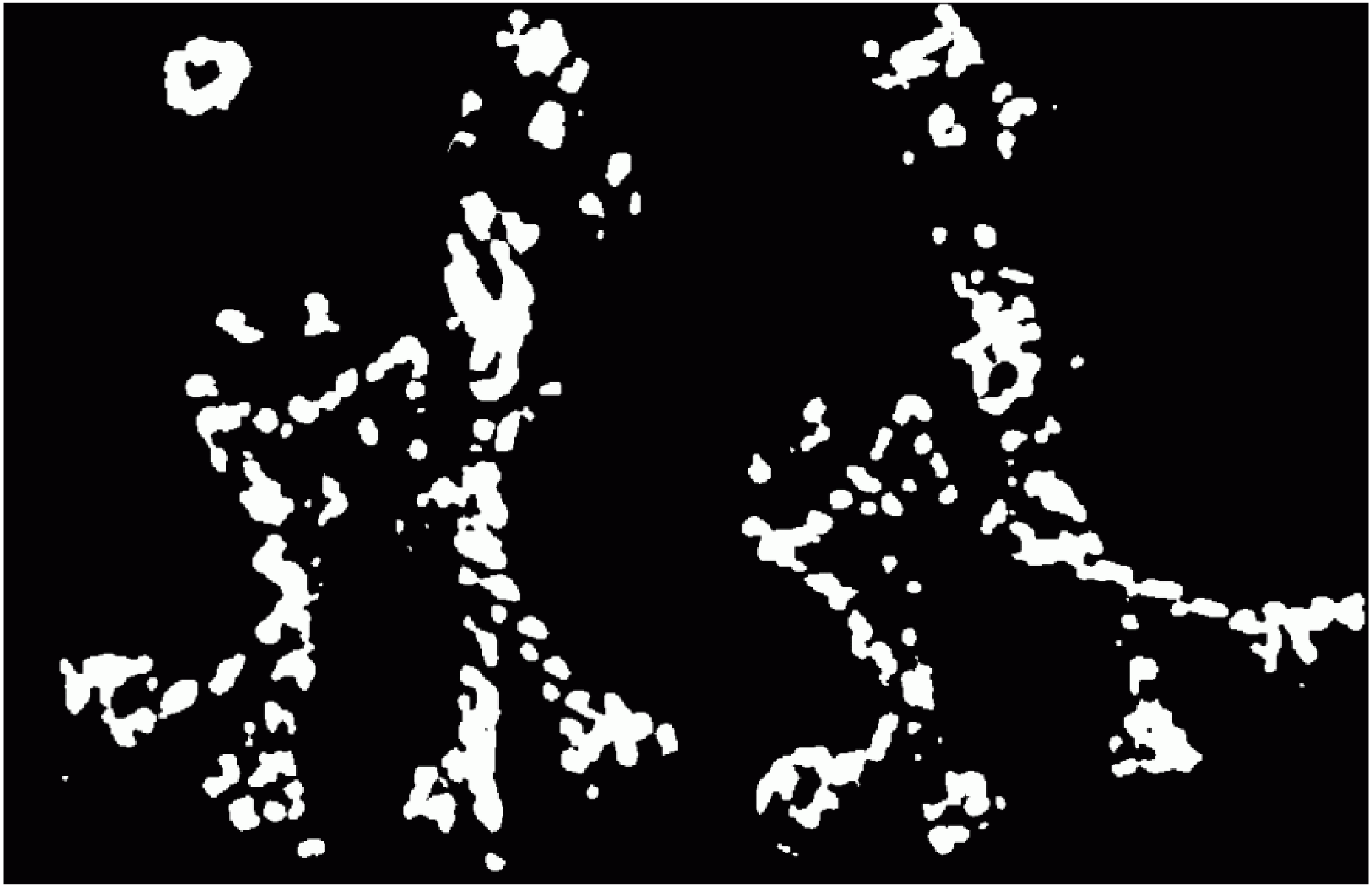
In Practice

1. Compute partial derivatives Ix, Iy per pixel
2. Compute **M** at each pixel, using Gaussian weighting w
3. Compute response function R
4. Threshold R

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Slide credit: S. Lazebnik

Thresholded R



Slide credit: S. Lazebnik

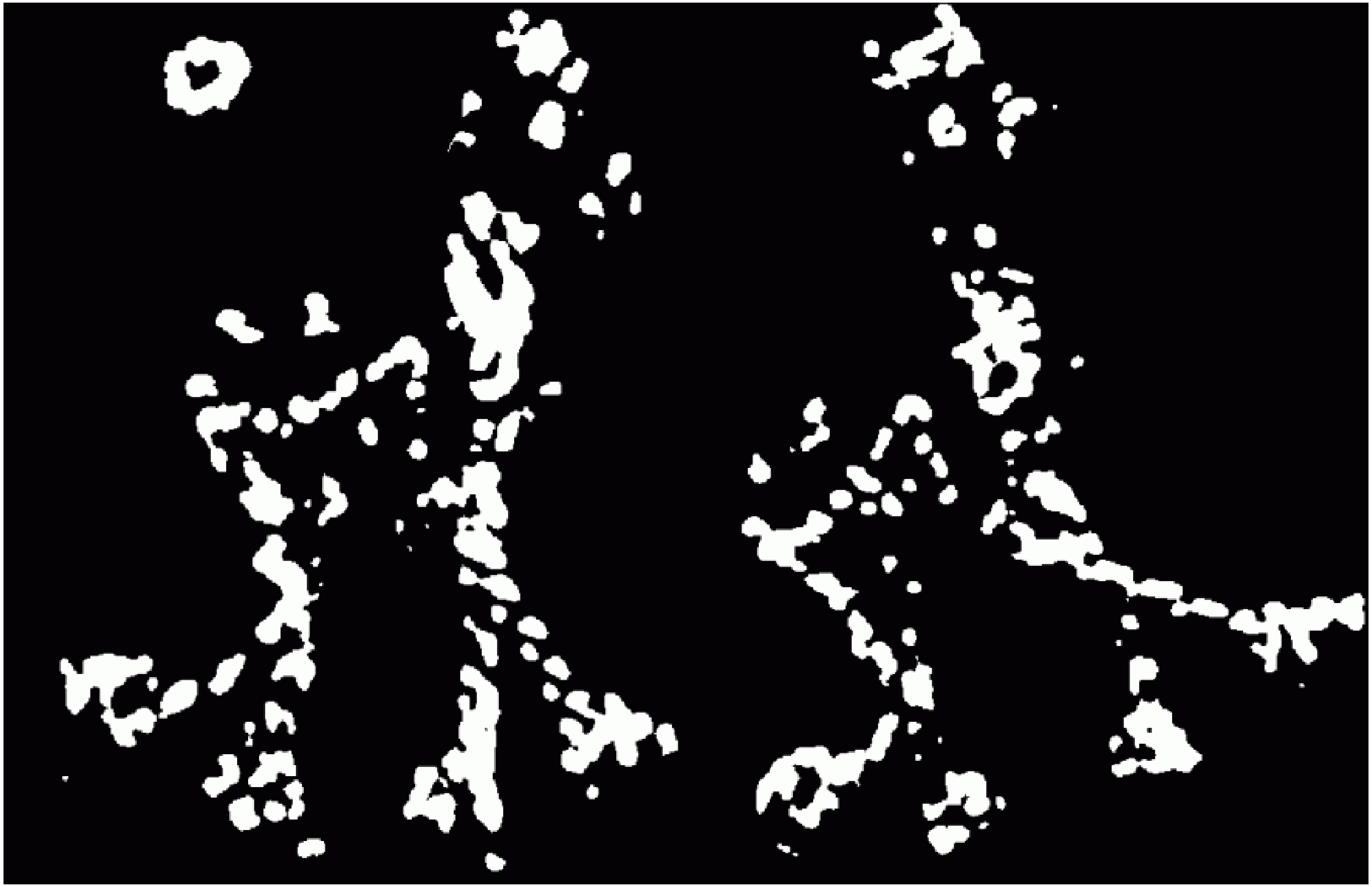
In Practice

1. Compute partial derivatives Ix, Iy 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)

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Slide credit: S. Lazebnik

Thresholded, NMS R



Slide credit: S. Lazebnik

Final Results

Slide credit: S. Lazebnik

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

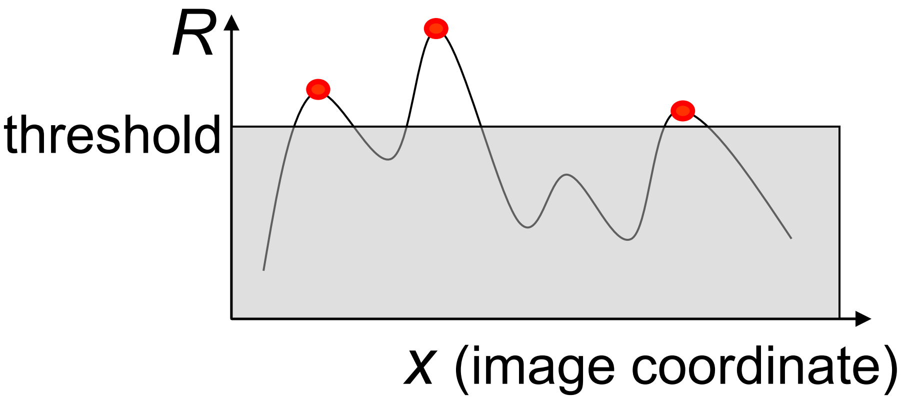
Images may be different in lighting and geometry

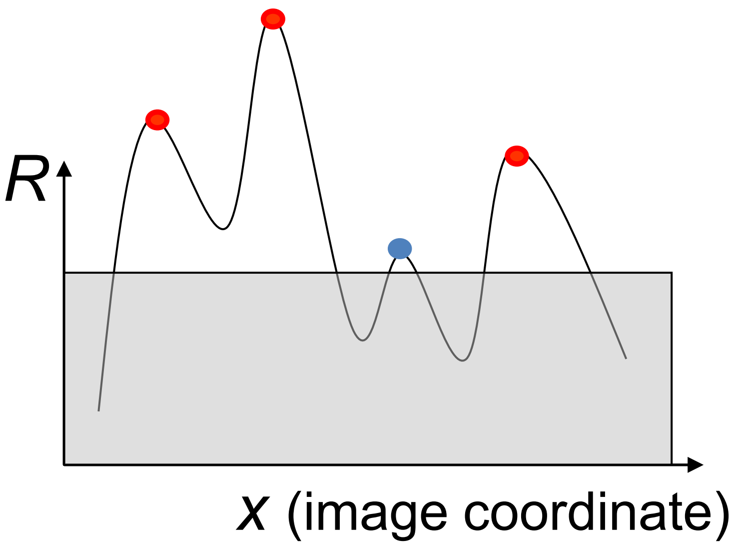
Affine Intensity Change

$$I_{new} = aI_{old} + b$$

M only depends on derivatives, so b is irrelevant

But a scales derivatives and there's a threshold

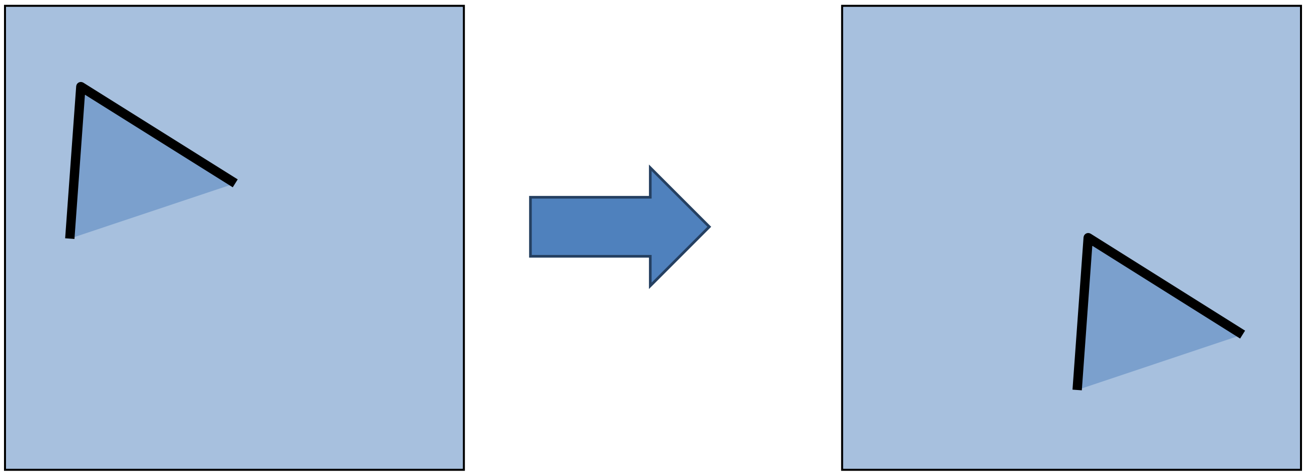




Partially invariant to affine intensity changes

Slide credit: S. Lazebnik

Image Translation

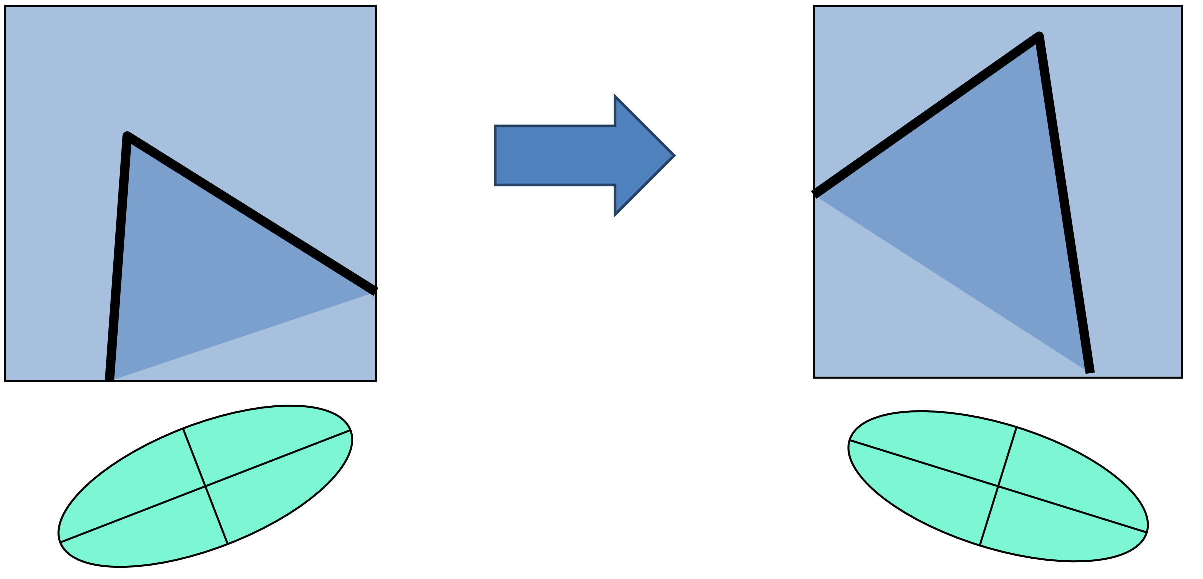


All done with convolution. Convolution is translation invariant.

Equivariant with translation

Slide credit: S. Lazebnik

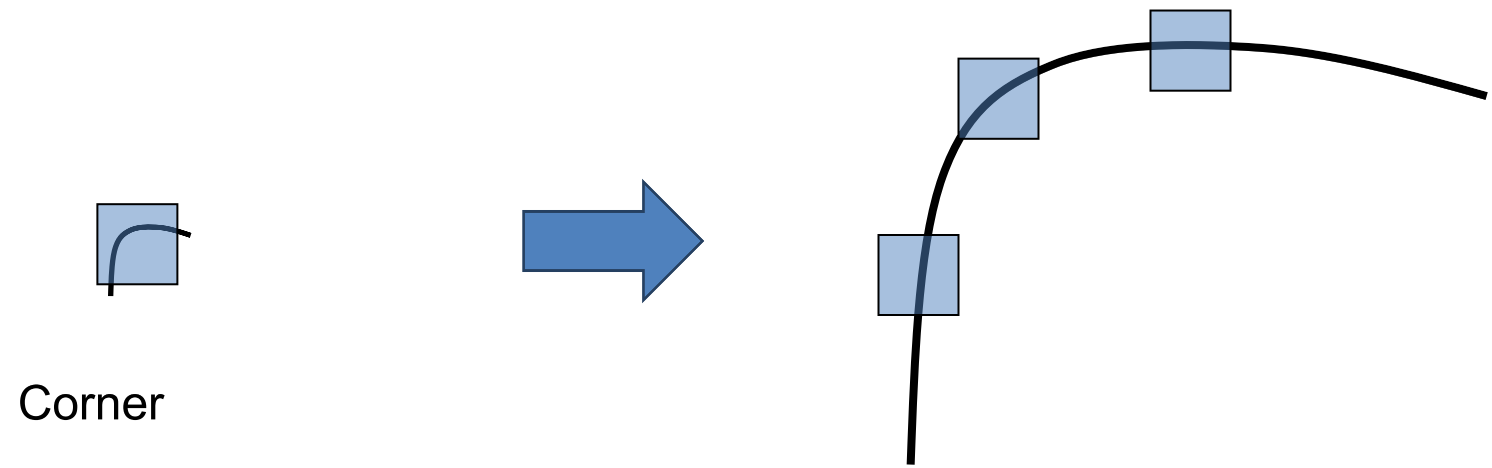
Image Rotation



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

Equivariant with rotation

Image Scaling



One pixel can become many pixels and vice-versa.

Not equivariant with scaling

For the Curious

Review: Quadratic Forms

Suppose have symmetric matrix **M,** scalar a, vector [u,v]:

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

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

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

is an ellipse.

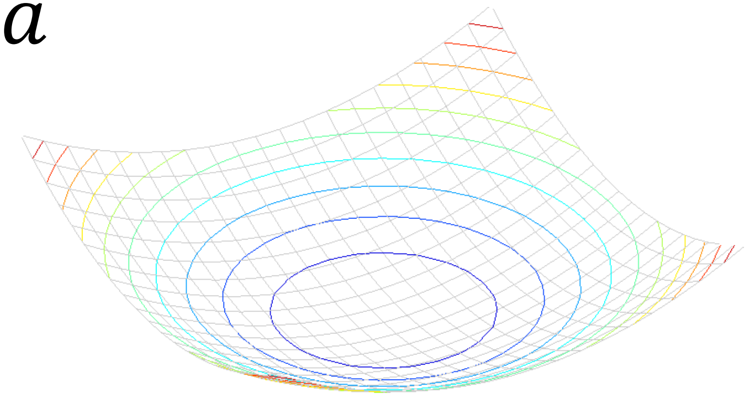


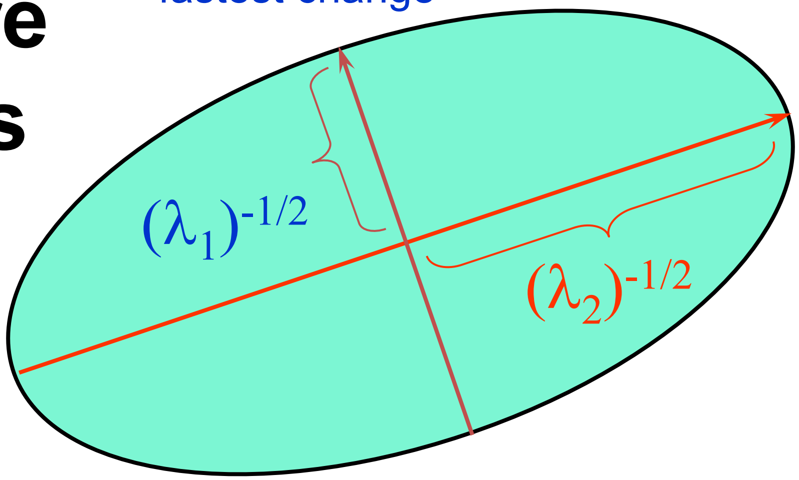
Diagram credit: S. Lazebnik

Review: Quadratic Forms

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

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

λ_1 and λ_2 are eigenvalues



Slide credit: S. Lazebnik

Interpreting The Matrix M

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

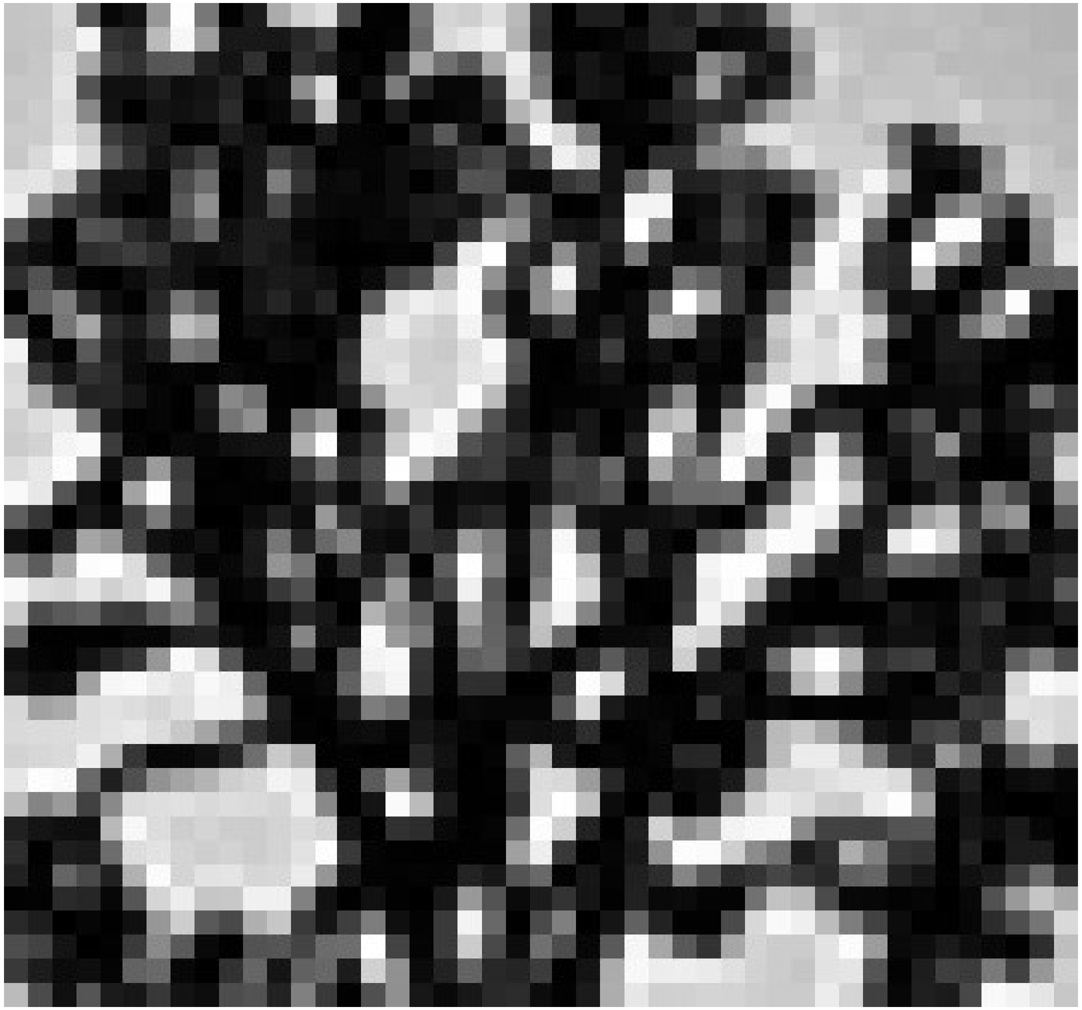
Can compute at each pixel

Directions

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

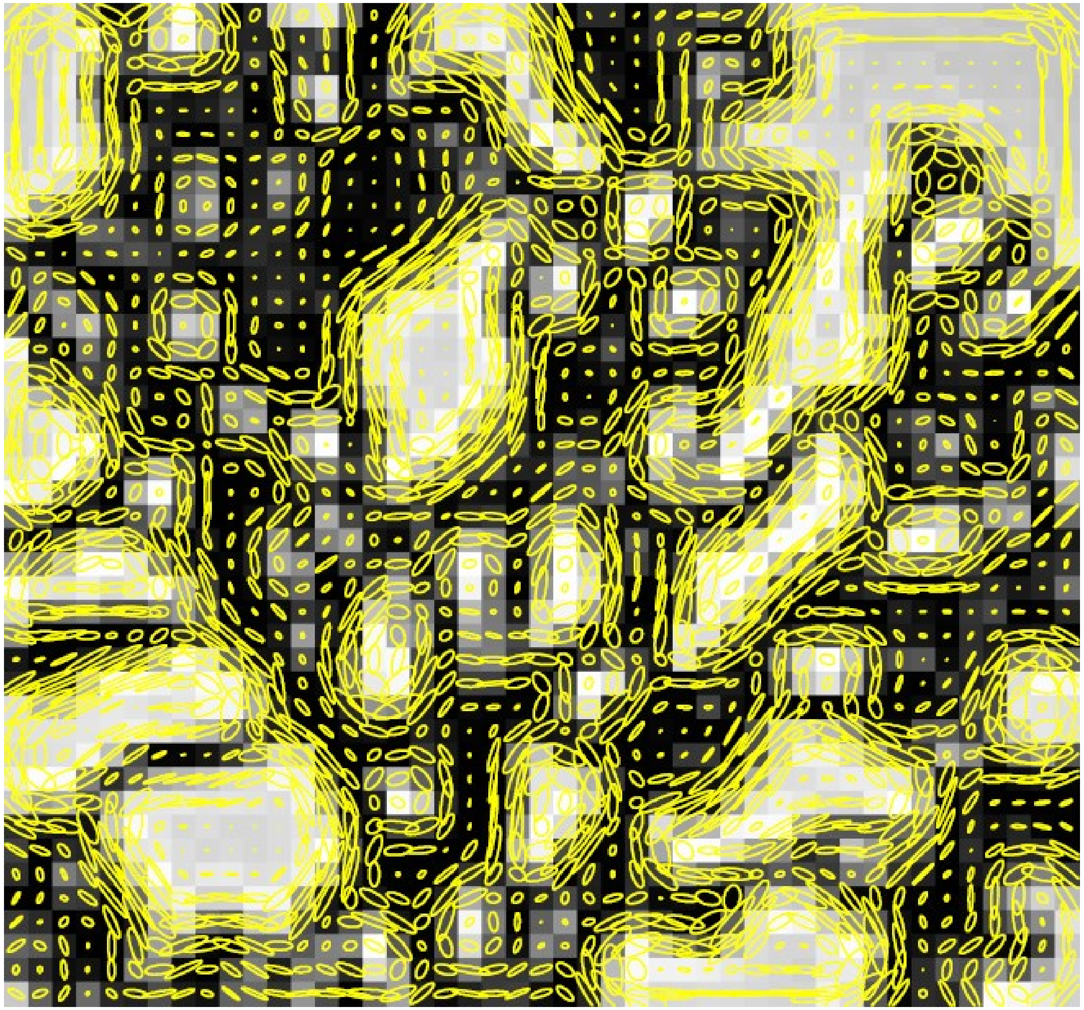
Amounts

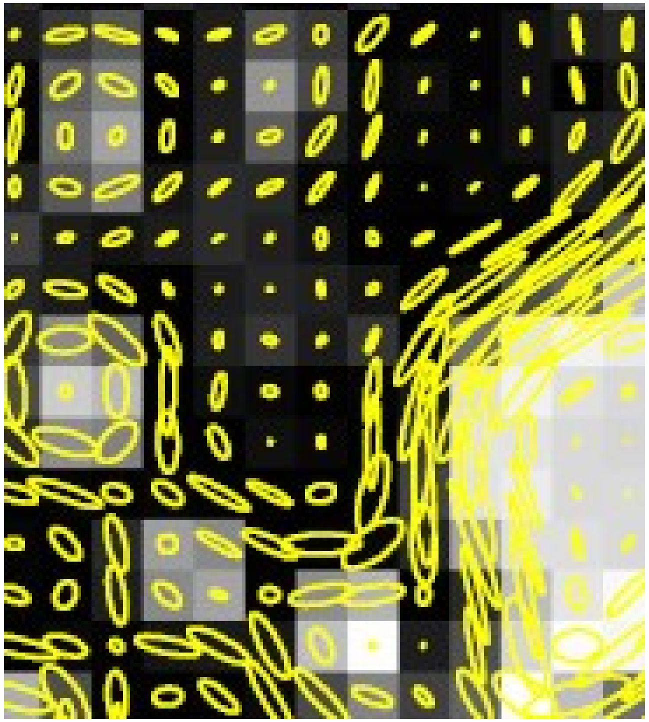
Visualizing M



Slide credit: S. Lazebnik

Visualizing M

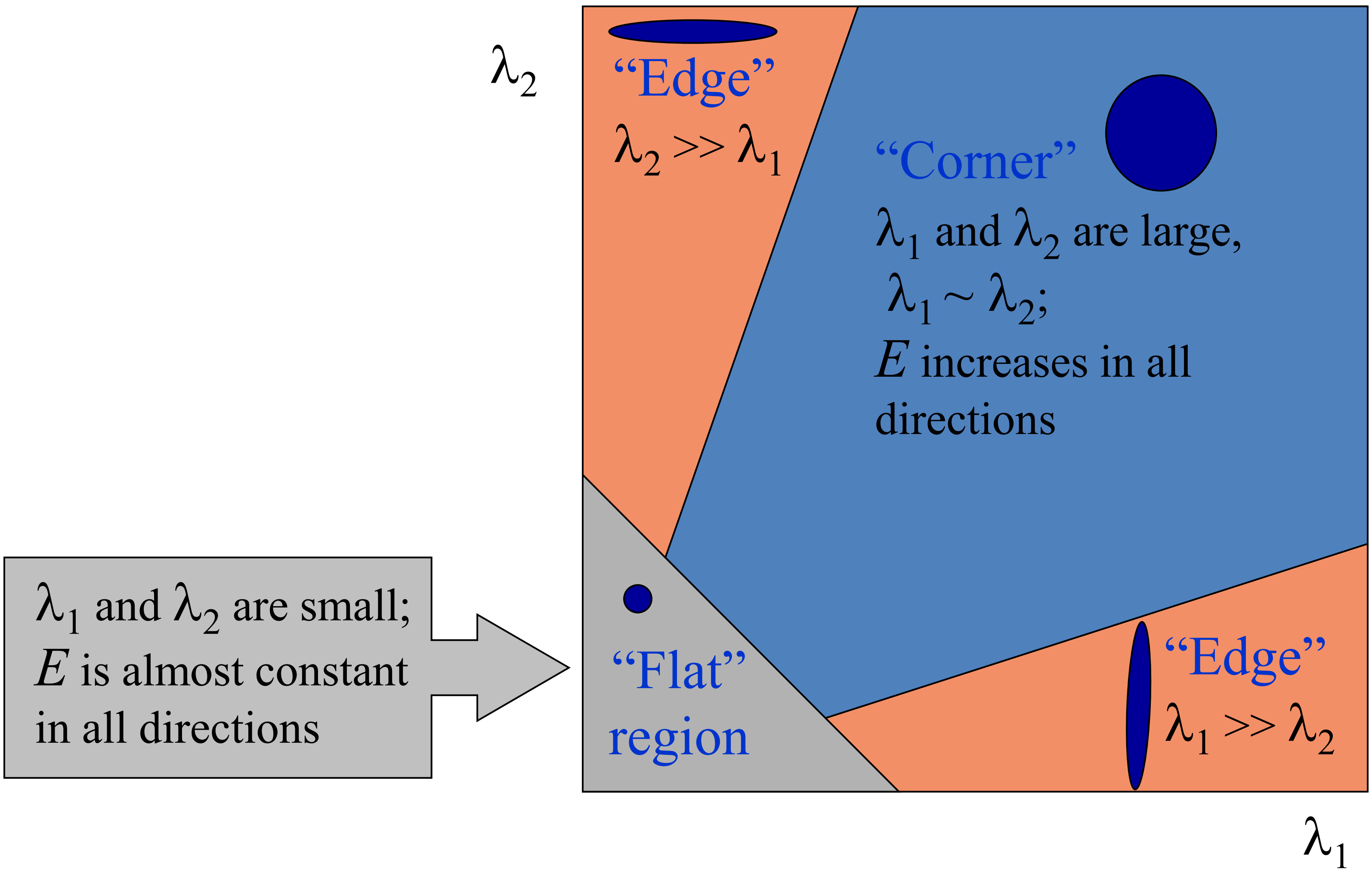




Technical note: M is often best *visualized* by first taking inverse, so long edge of ellipse goes along edge

Slide credit: S. Lazebnik

Interpreting Eigenvalues of M

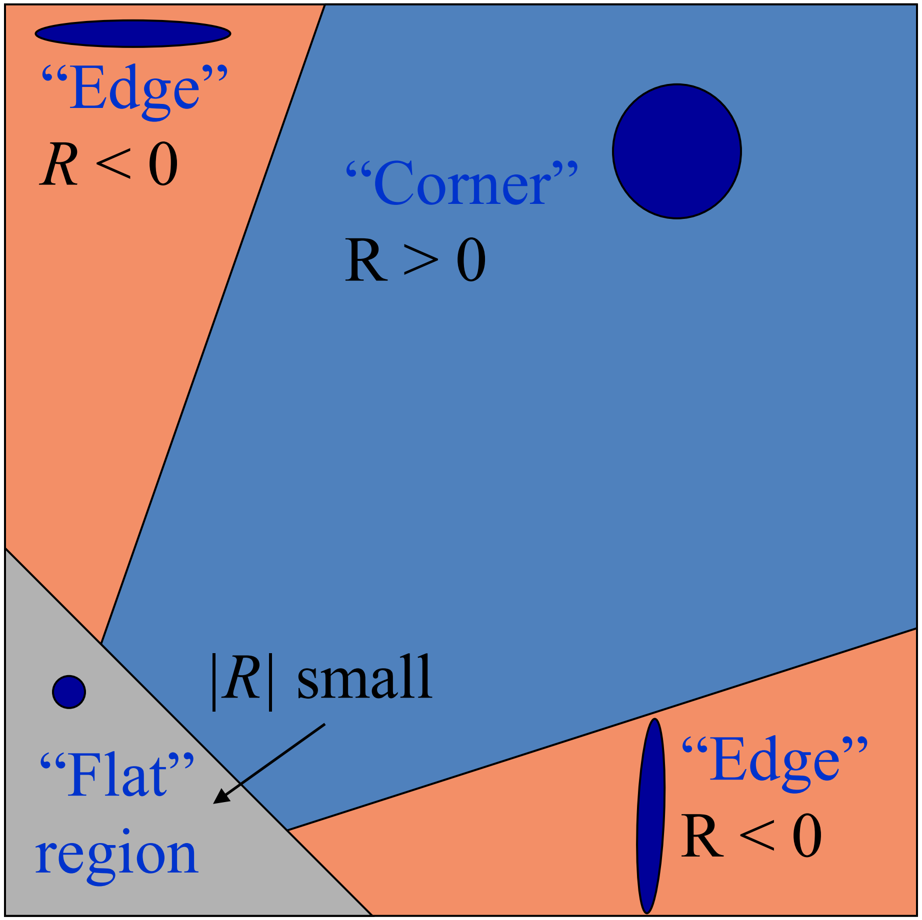


Slide credit: S. Lazebnik; Note: this refers to previous ellipses, not original M ellipse. Other slides on the internet may vary

Putting Together The Eigenvalues

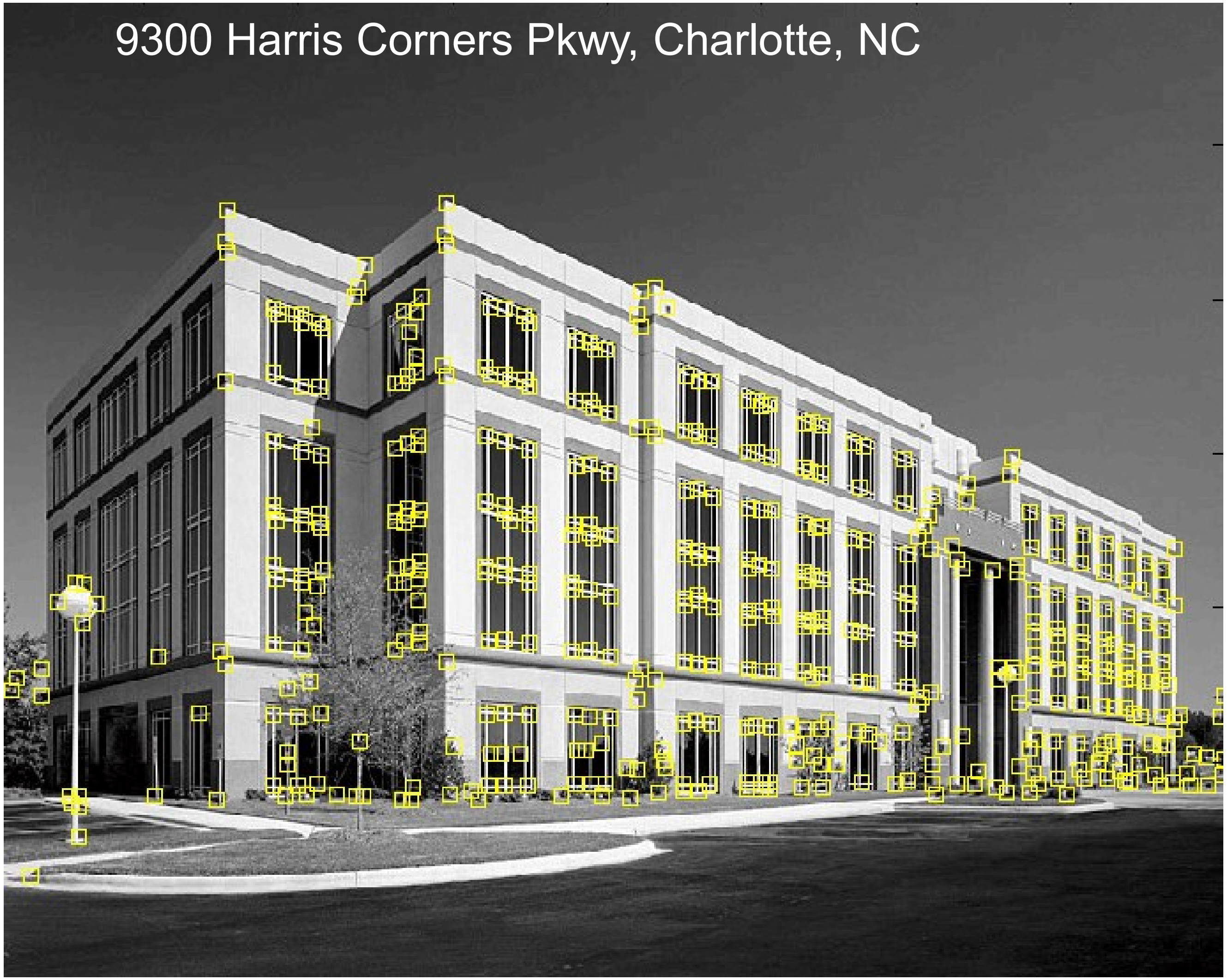
$$R = \det(\boldsymbol{M}) - \alpha\, trace(\boldsymbol{M})^2$$
$$= \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: 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



Slide Credit: S. Lazebnik

Derivatives Review

Given quadratic function f(x)

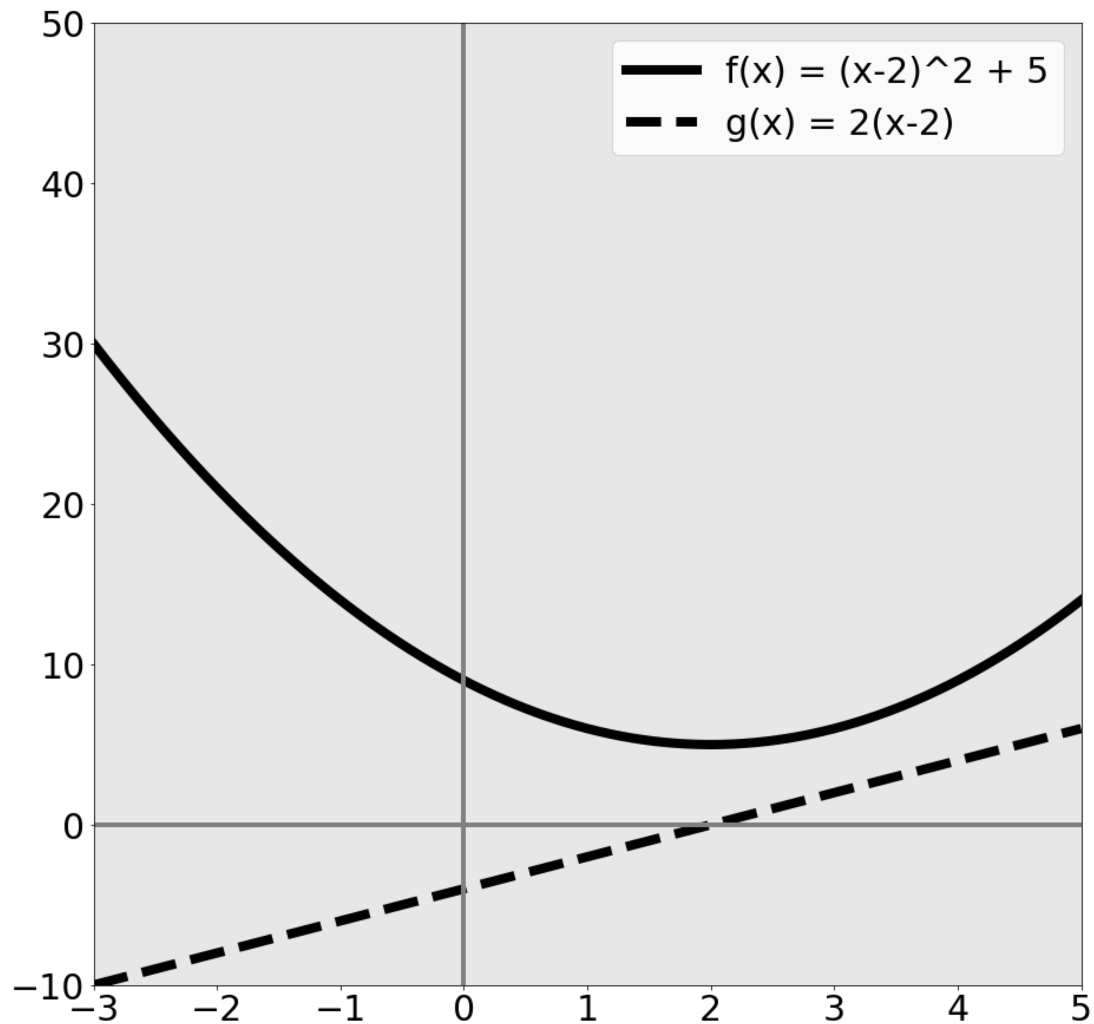
$$f(x) = (x-2)^2 + 5$$

$f(x)$ is function

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

aka

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



Given quadratic function f(x)

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

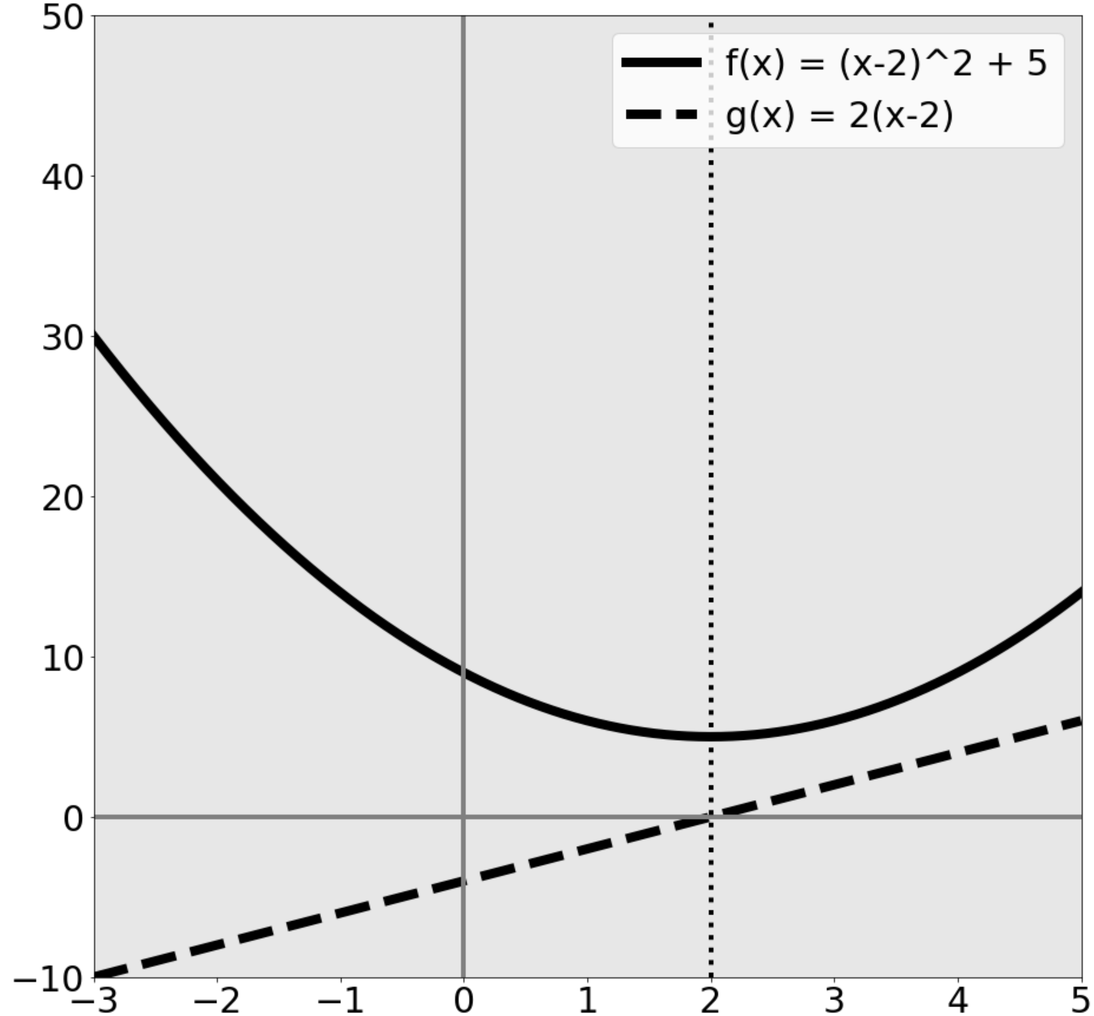
What's special about x=2?

$f(x)$ minim. at 2

$g(x) = 0$ at 2

a = minimum of f → $g(a) = 0$

Reverse is ***not true***



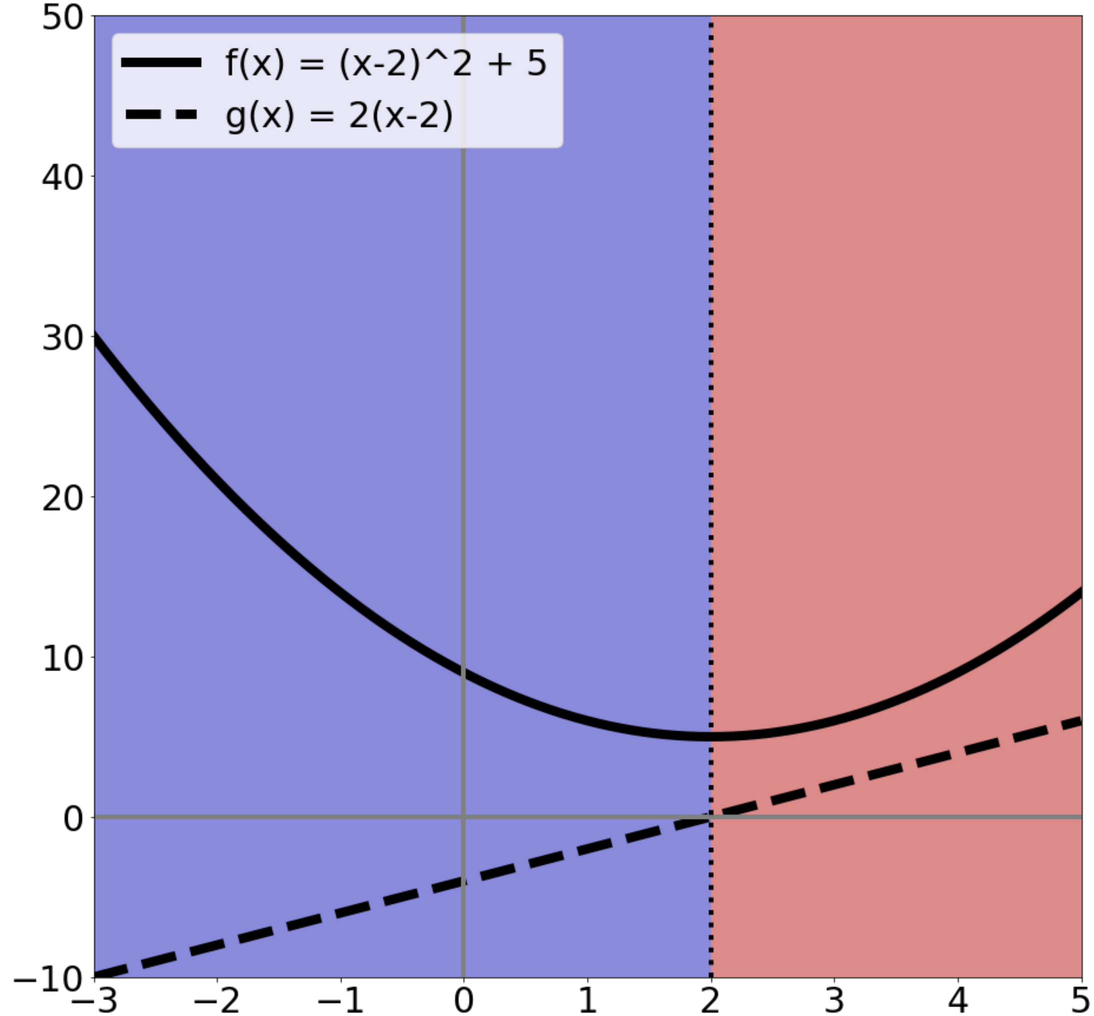
Rates of change

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

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

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

$$\frac{\partial}{\partial x} f(x) = 2(x-2) * 1 = 2(x-2)$$

$$f_2(x,y) = (x-2)^2 + 5 + \boxed{(y+1)^2}$$

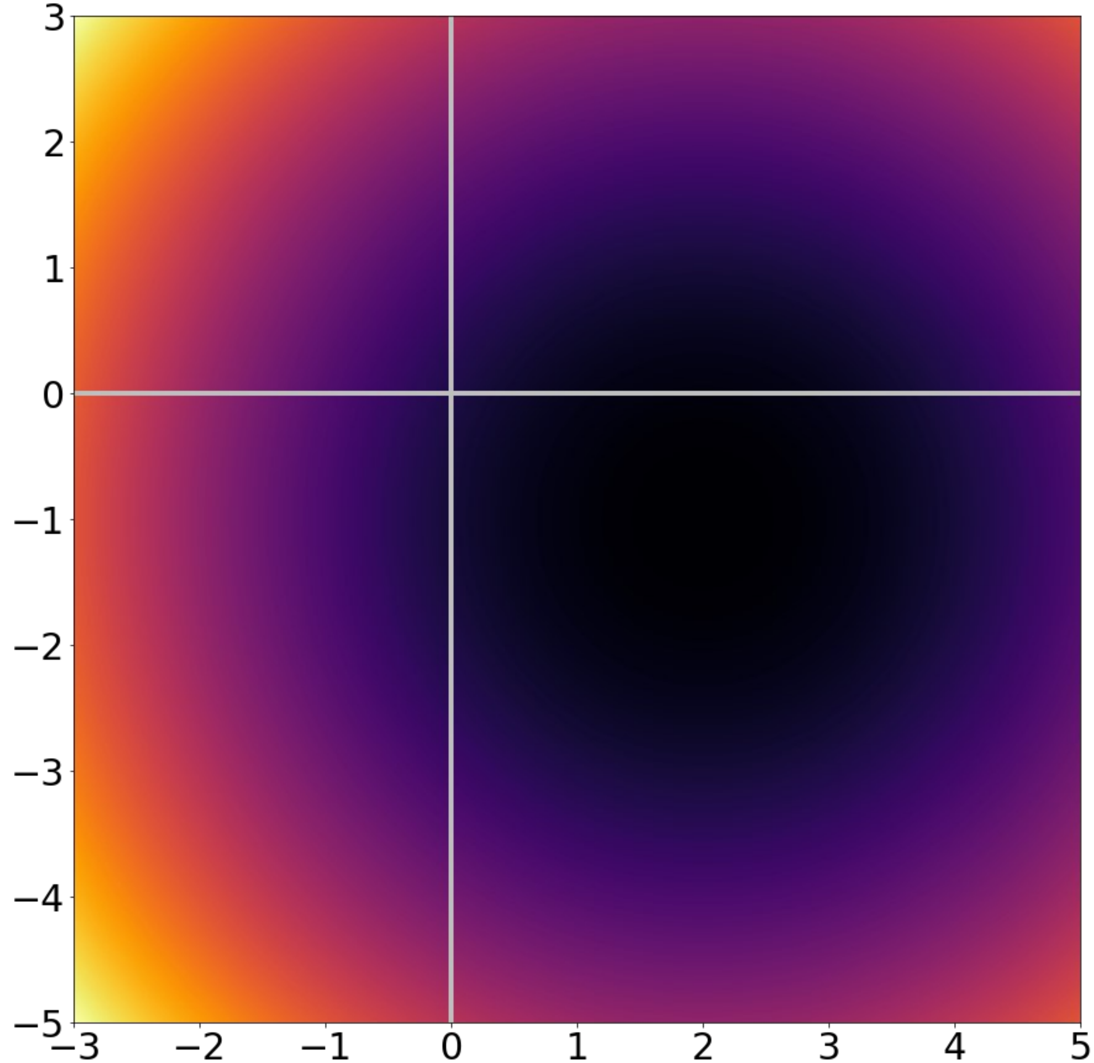
Pretend it's constant → derivative = 0

$$\frac{\partial}{\partial x} f_2(x) = 2(x-2)$$

Zooming Out

$$f_2(x,y) = (x-2)^2 + 5 + (y+1)^2$$

Dark = f(x,y) low
Bright = f(x,y) high



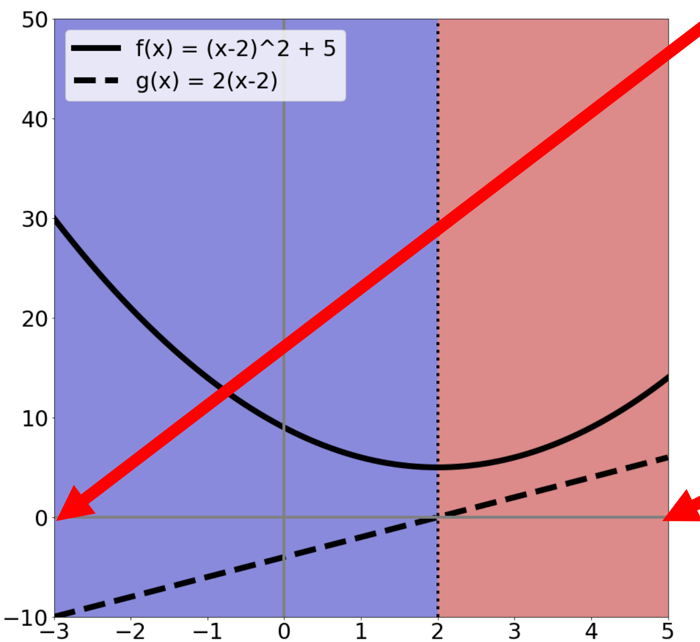
Taking a slice of

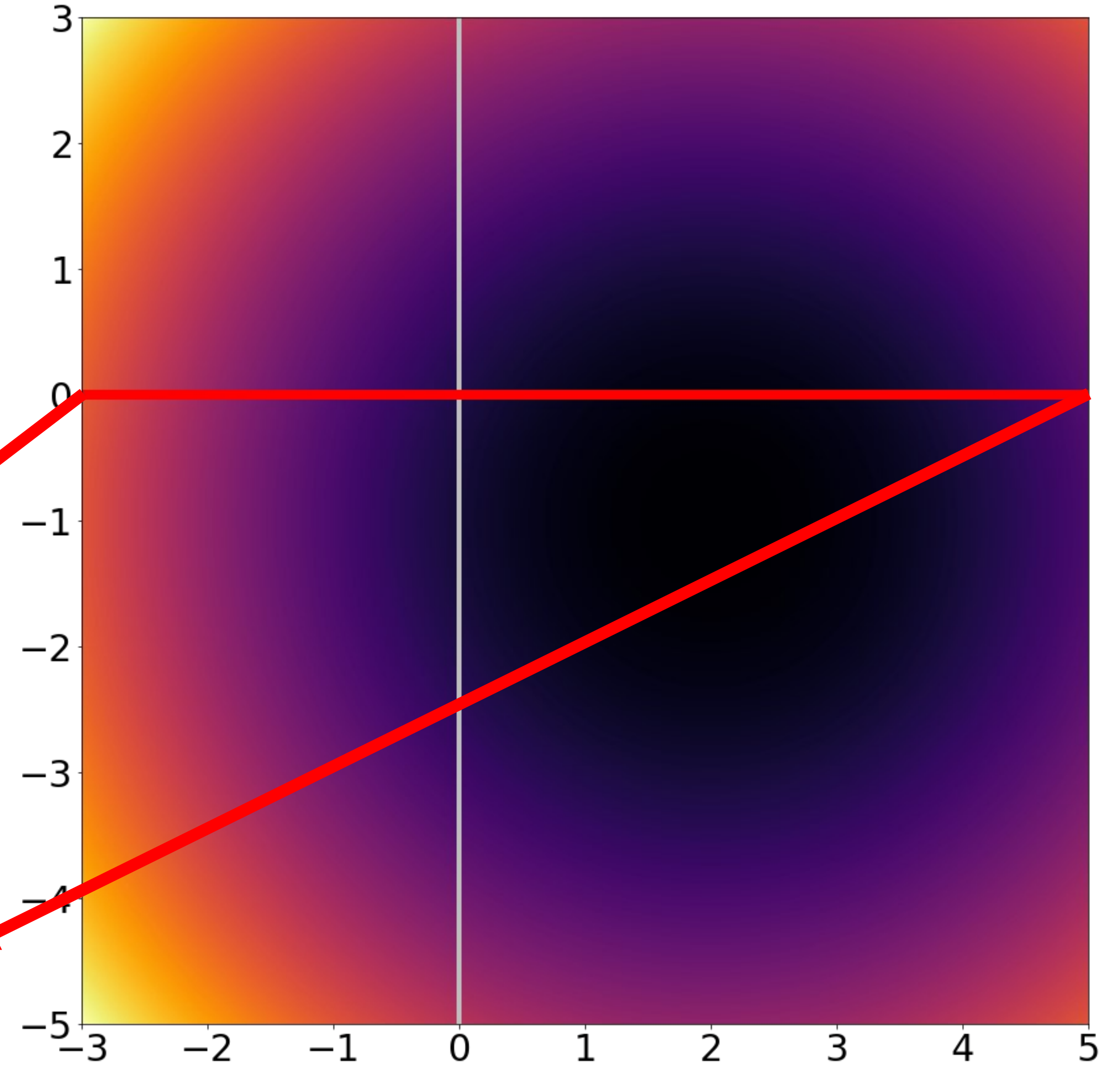
$$f_2(x, y) = (x-2)^2 + 5 + (y+1)^2$$

Slice of y=0 is the function from before:

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

$$f'(x) = 2(x-2)$$

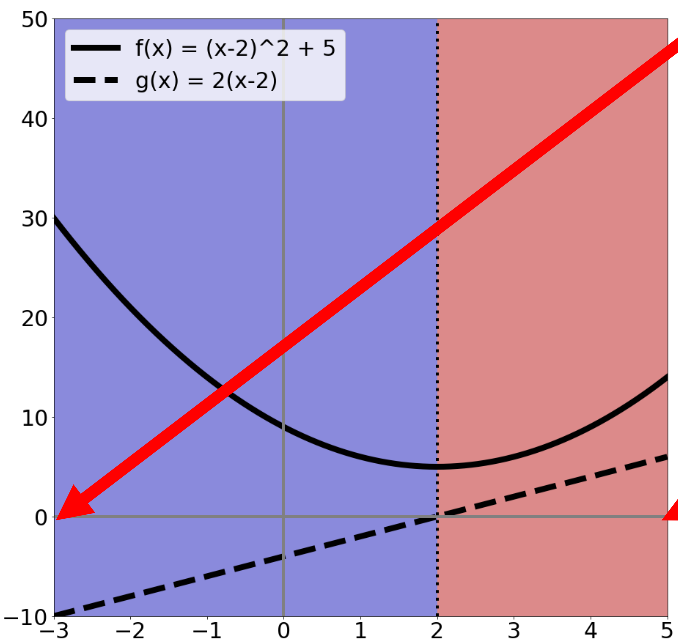


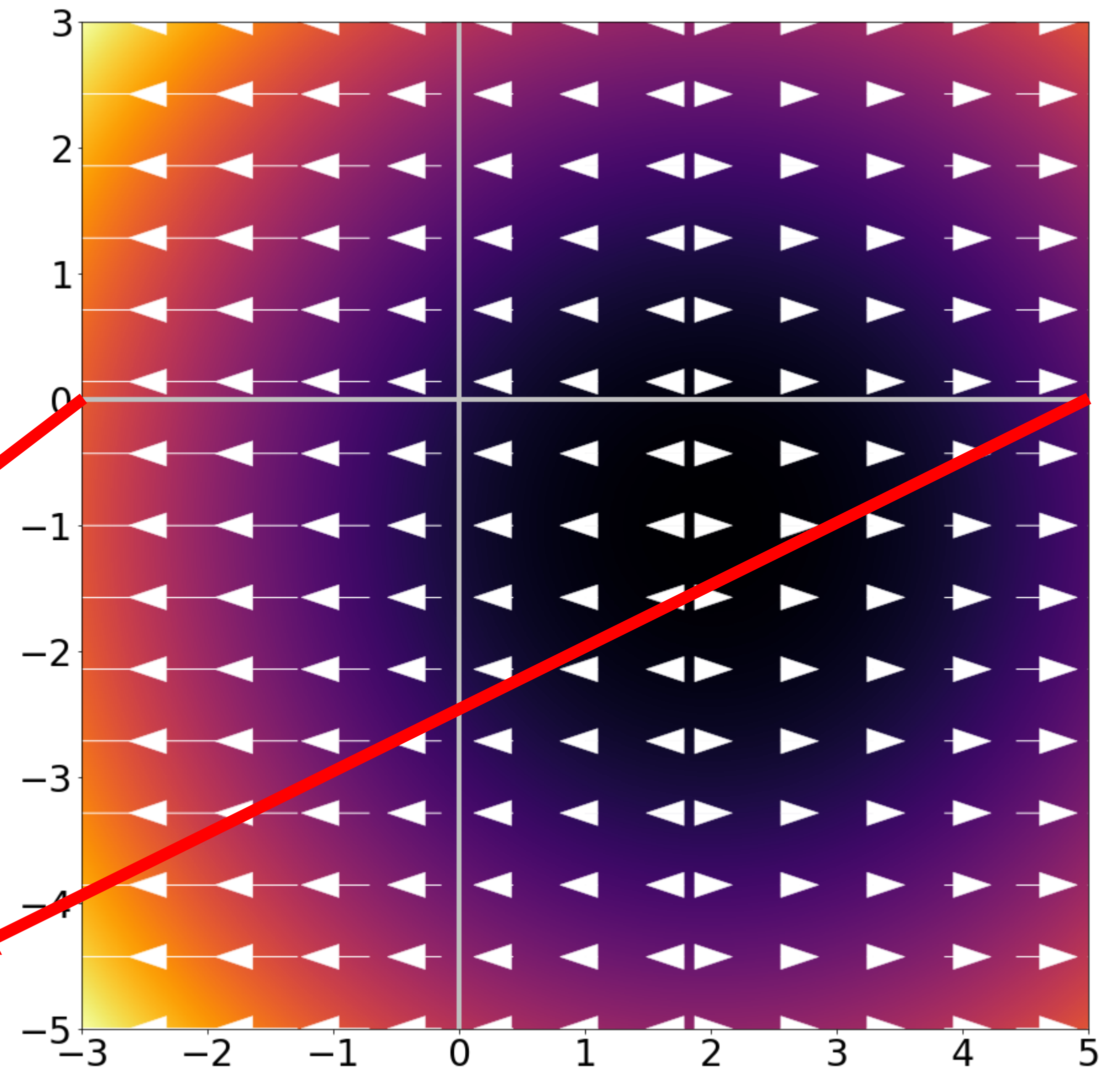


Taking a slice of

$$f_2(x,y) = (x-2)^2 + 5 + (y+1)^2$$

$\frac{\partial}{\partial x} f_2(x,y)$ is rate of change & direction in x dimension





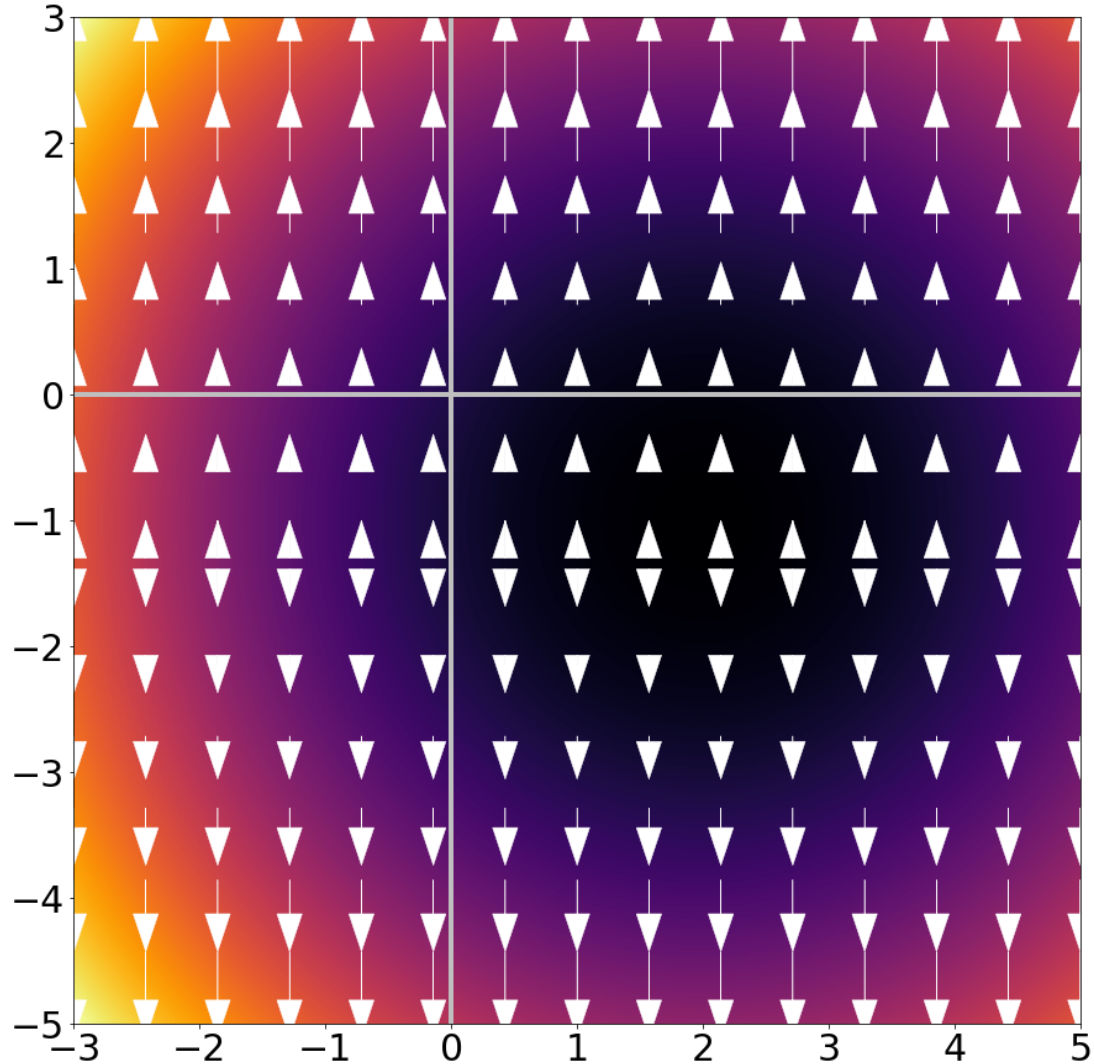
Zooming Out

$$f_2(x, y) = (x-2)^2 + 5 + (y+1)^2$$

$\frac{\partial}{\partial y} f_2(x, y)$ 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.