Lecture 9:
Edge + Corner Detection

Administrative

- HW1 due yesterday!

- HW2 out yesterday, due Wednesday 2/19 11:59pm

Motivating Problem

Are these pictures of the same object?
If so, how are they related?

Applications to Have in Mind

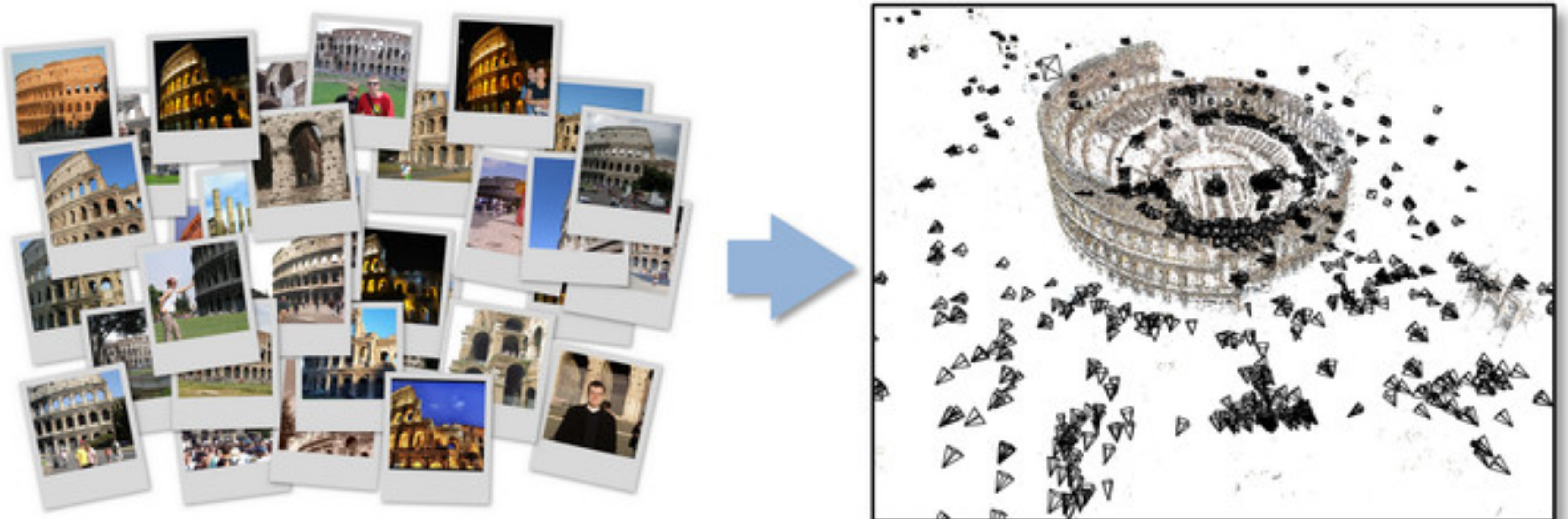
Object Recognition by matching against templates

Labeled Images

Image to Recognize

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?

One (Hopefully Familiar) Solution

```
for y in range(-ySearch,ySearch+1):
    for x in range(-xSearch,xSearch+1):
        #Touches all HxW pixels!
        check_alignment_with_images()
```

A 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

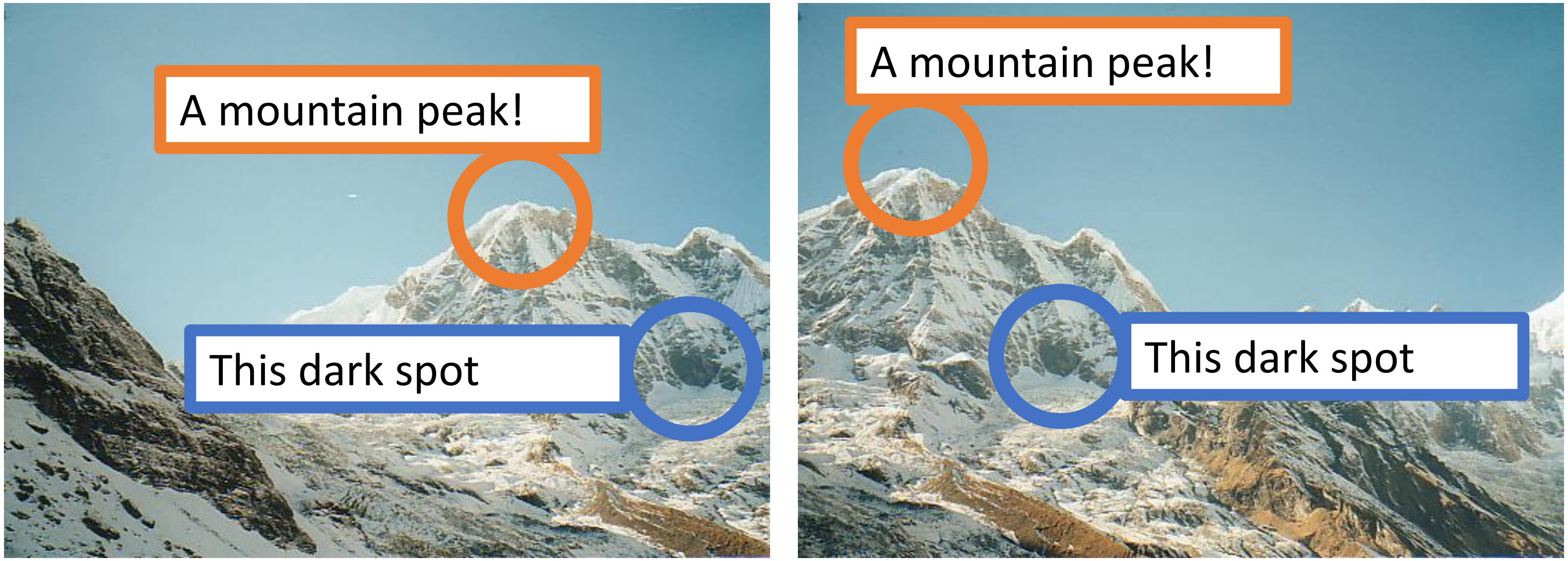
```
for y in yRange:
    for x in xRange:
        for z in zRange:
            for xRot in xRotVals:
                for yRot in yRotVals:
                    for zRot in zRotVals:
                        #touches all HxW pixels!
                        check_alignment_with_images()
```

This code should make you really unhappy

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

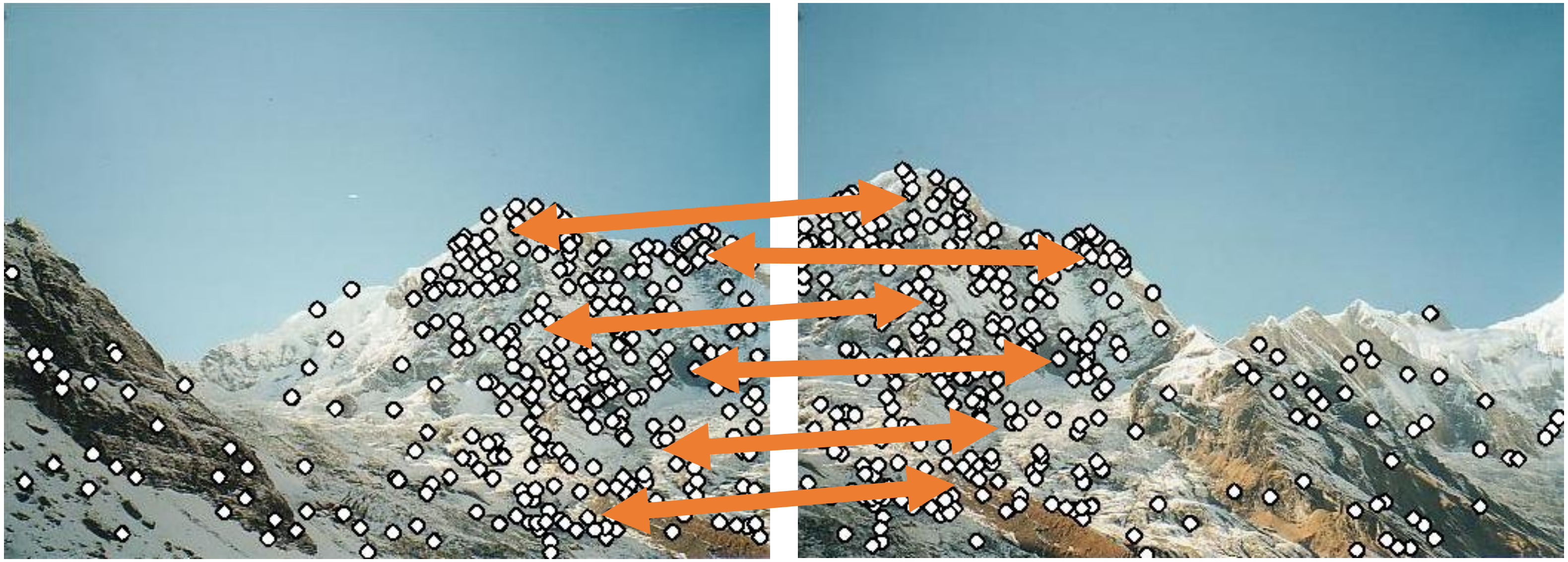
An Alternative Approach

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



An Alternative Approach

Finding and Matching



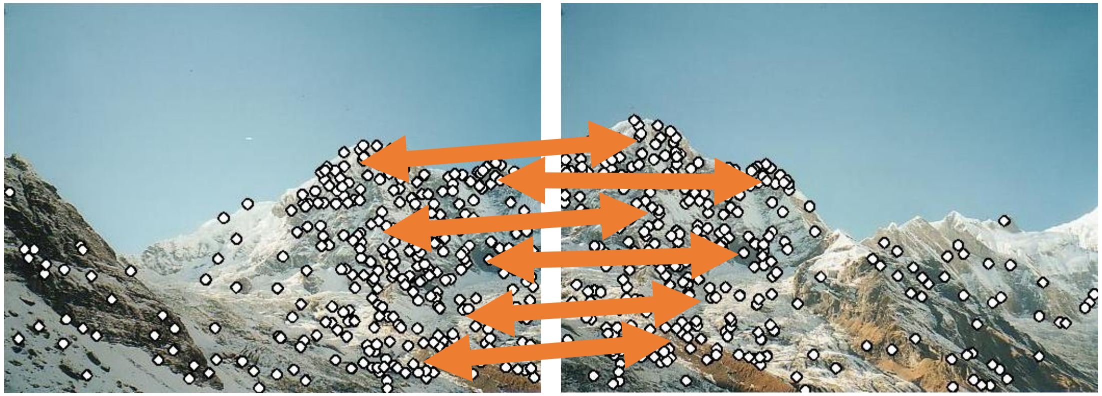
1: find corners+features

2: match based on local image data

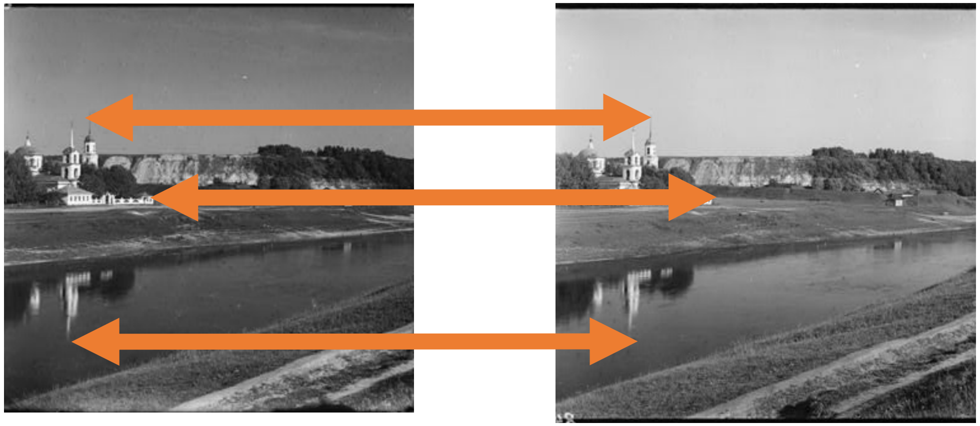
Slide Credit: S. Lazebnik, original figure: M. Brown, D. Lowe

An Alternative Approach

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



Consider translation-only case from HW1.



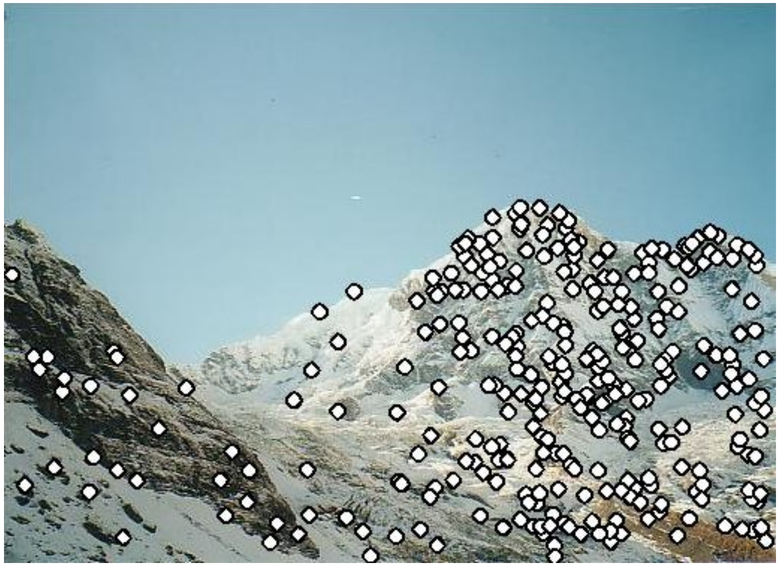
An Alternative Approach

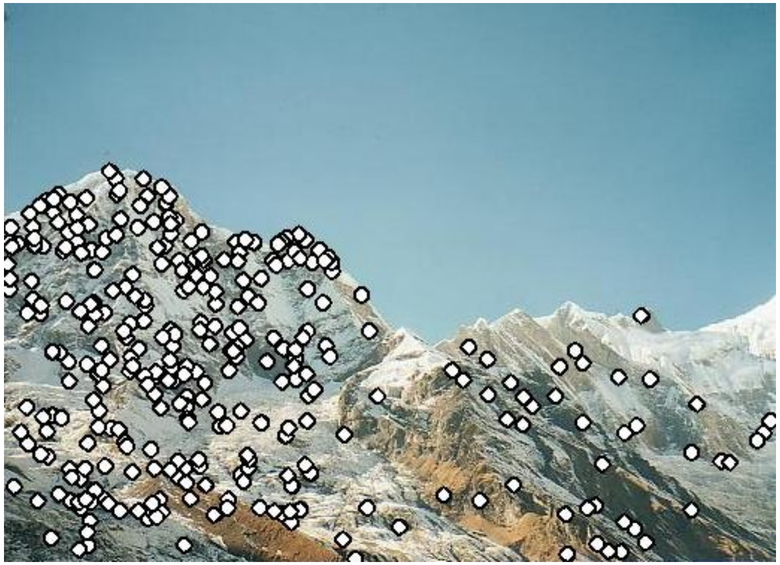
Blend Them Together

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

Photo Credit: M. Brown, D. Lowe

An Alternative Approach



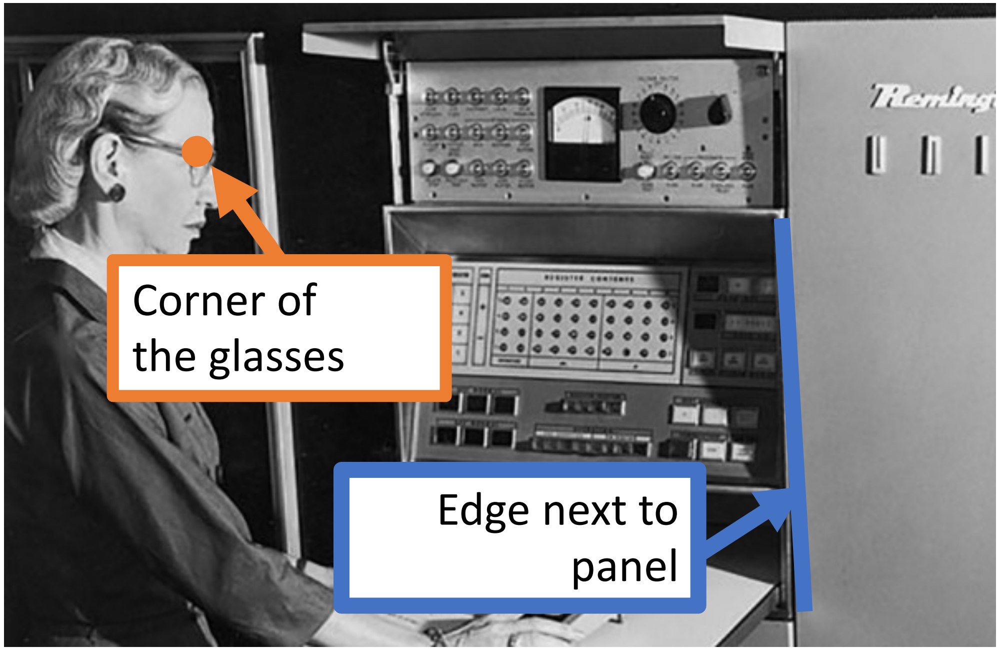


Problem #1 (today): How do we detect points in images?
Problem #2 (next time): How do we describe points in images?

Our points must be robust to viewpoint and illumination change!

Today

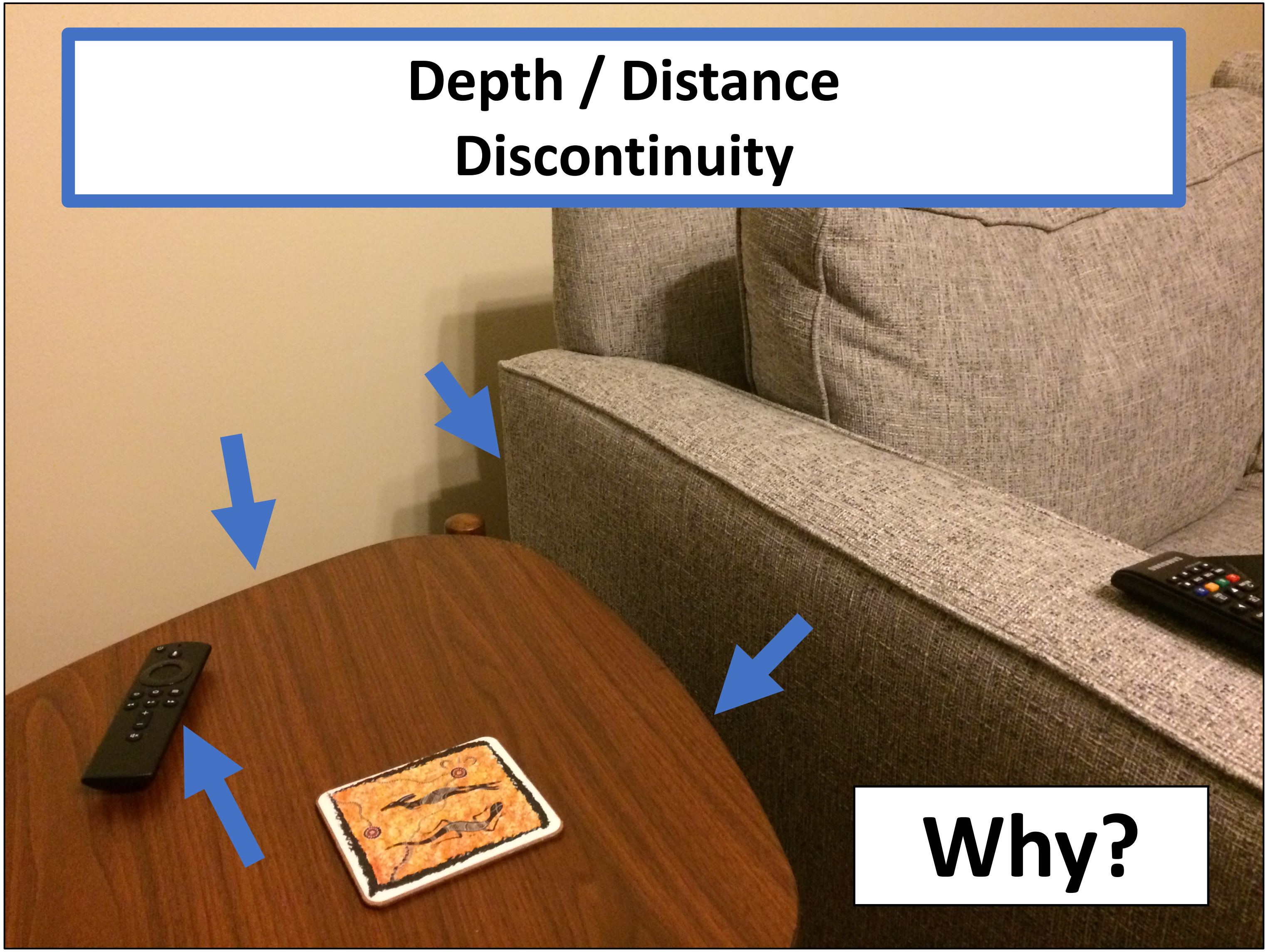
Part 1: Finding Edges
Part 2: Finding Corners



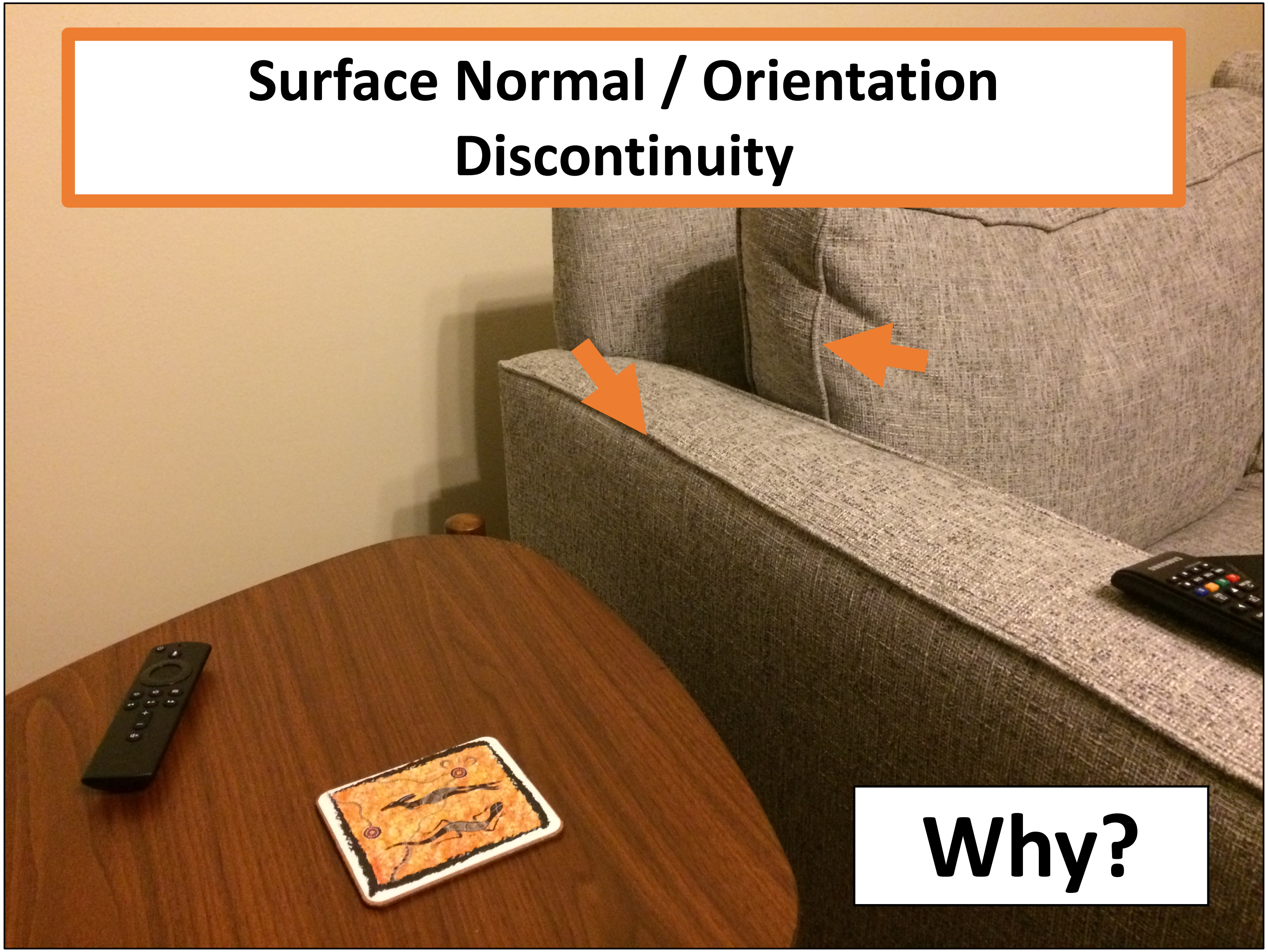
Part I: Edges

Where do Edges Come From?

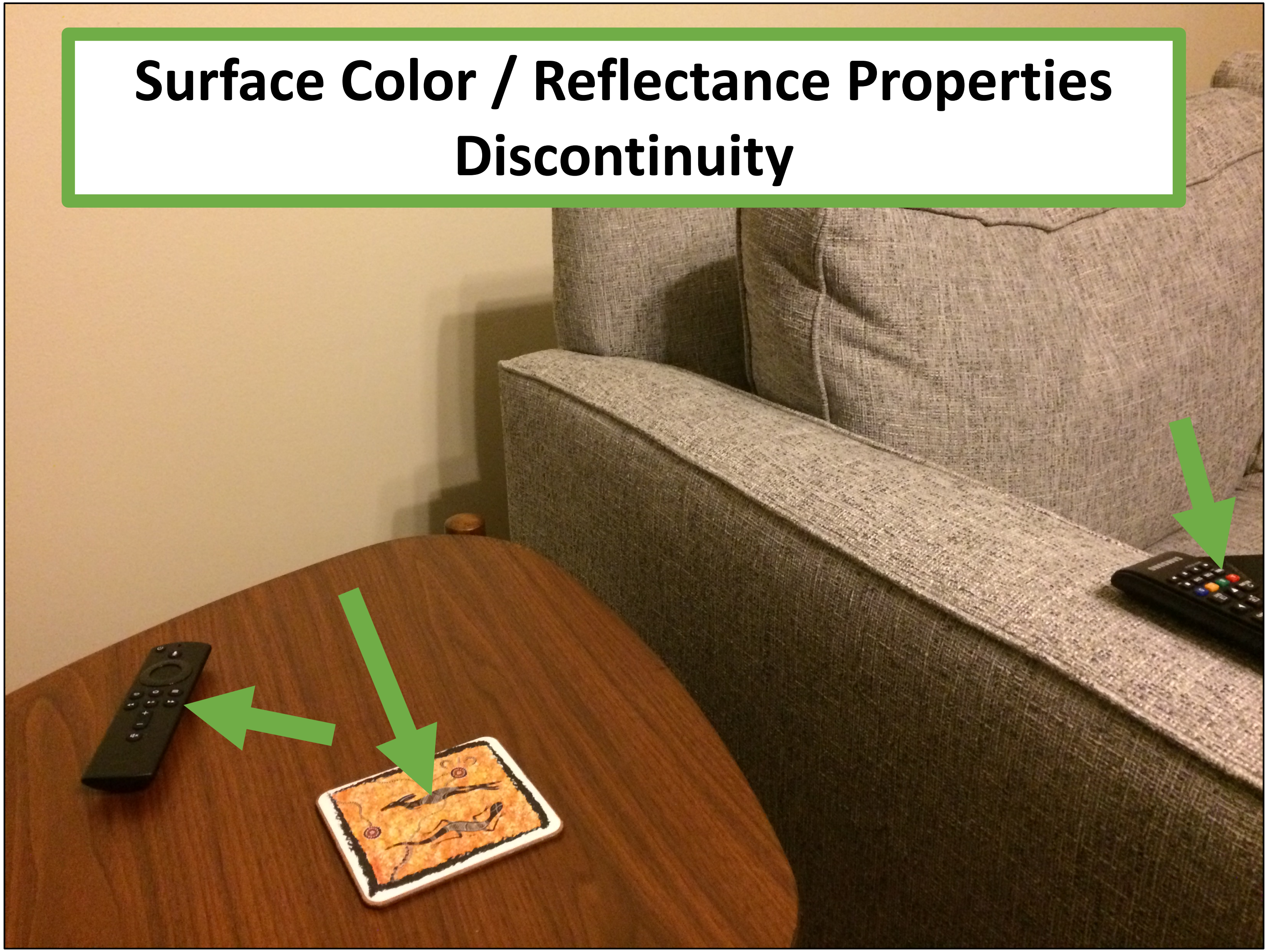
Where do Edges Come From?



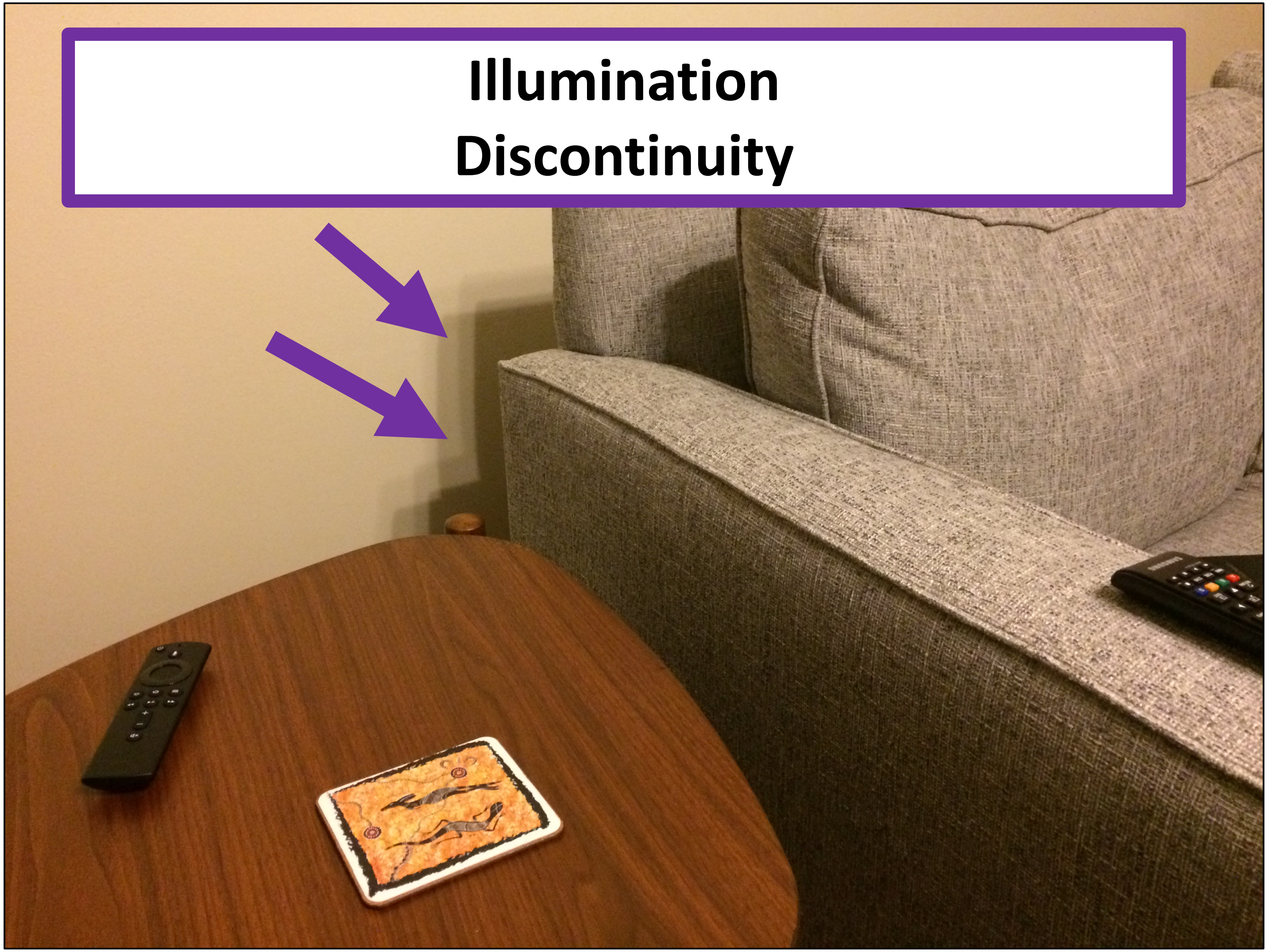
Where do Edges Come From?



Where do Edges Come From?



Where do Edges Come From?



Last Time: Image Gradient

Compute derivatives Ix and Iy with filters

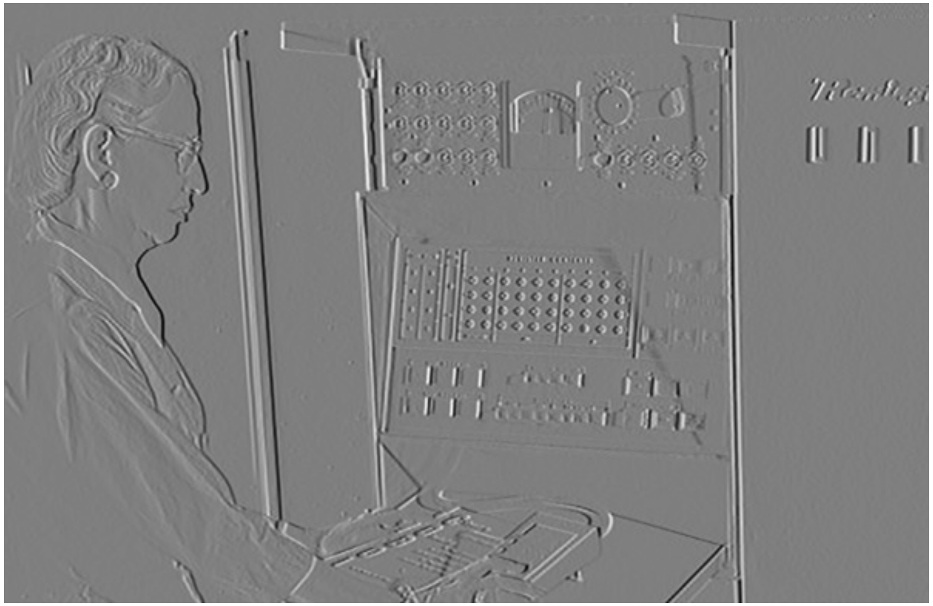
Ix

Iy

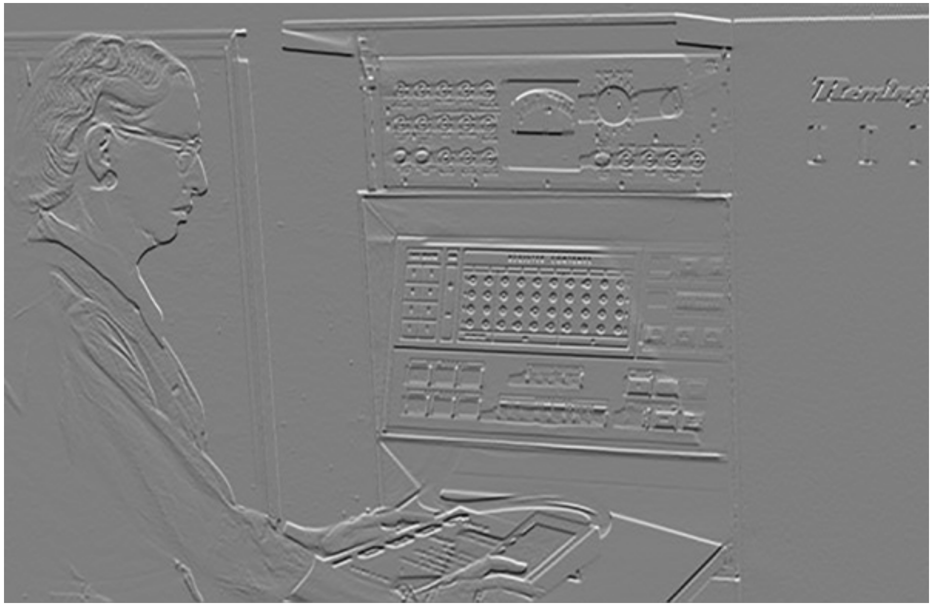
Last Time: Image Gradient

Compute derivatives Ix and Iy with filters

Ix



Iy



Last Time: Gradient Magnitude

Gradient Magnitude $(Ix^2 + Iy^2)^{1/2}$
Gives rate of change at each pixel

Last Time: Gradient Magnitude

Gradient Magnitude $(Ix^2 + Iy^2)^{1/2}$

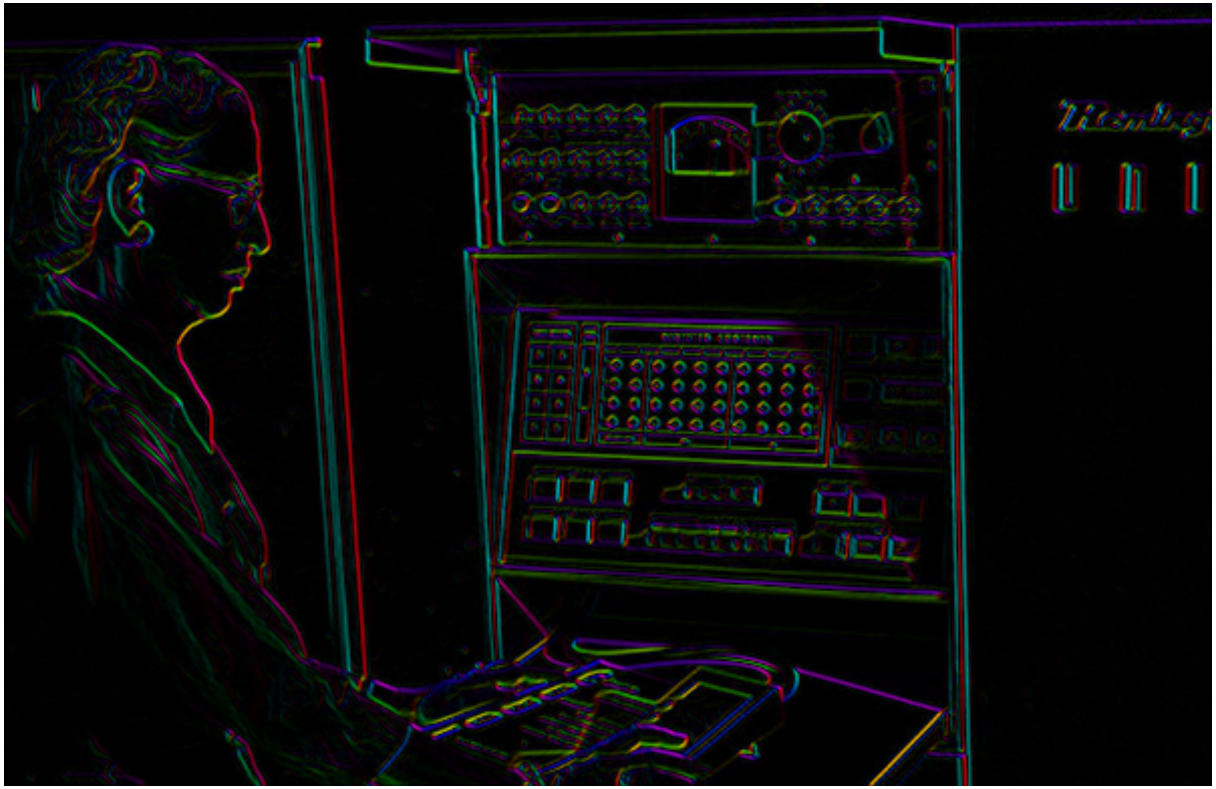
Gives rate of change at each pixel

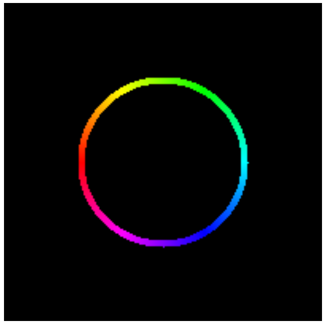
Last Time: Gradient Direction

Gradient Direction atan2(Ix, Iy)
Gives direction of change at each pixel

Last Time: Gradient Direction

Gradient Direction atan2(Ix, Iy)
Gives direction of change at each pixel

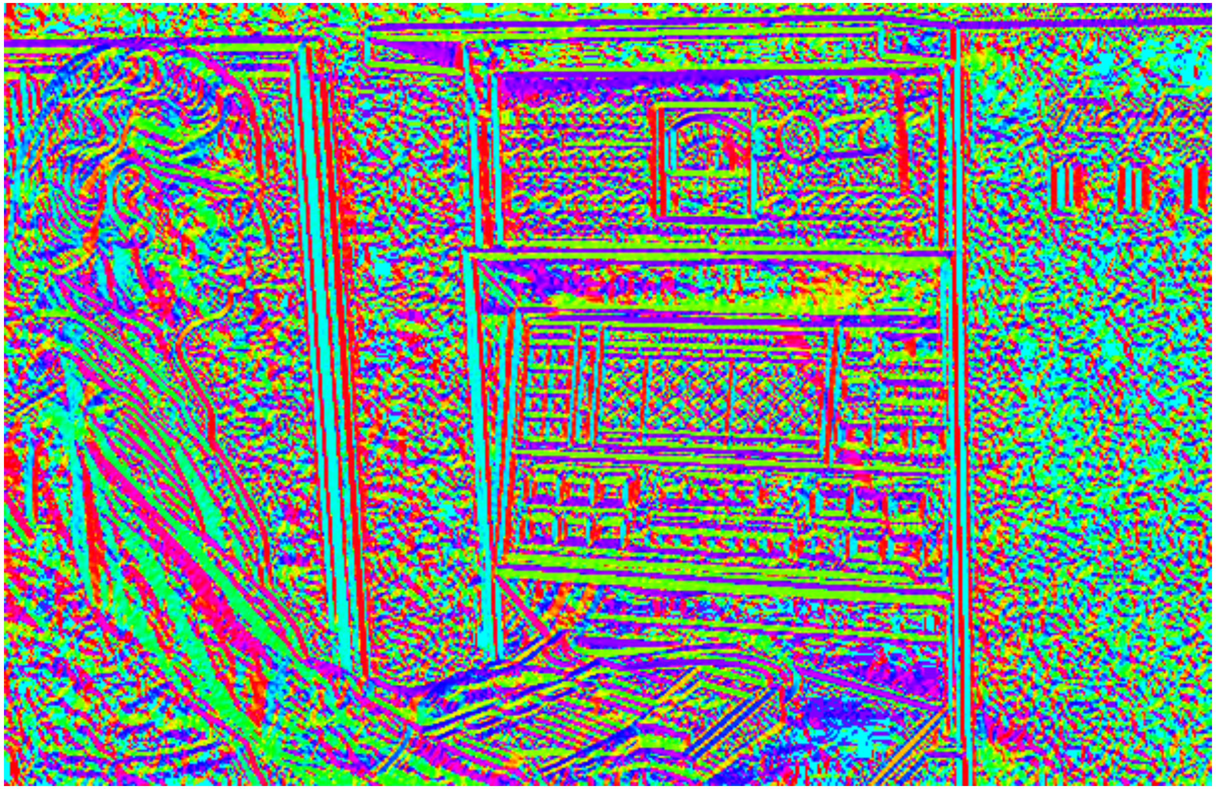


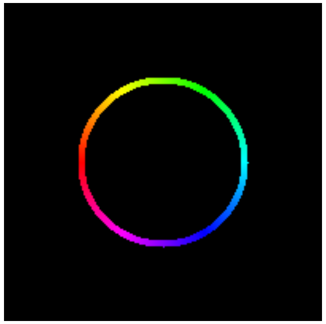


I'm making the lightness equal to gradient magnitude

Last Time: Gradient Direction

Gradient Direction atan2(Ix, Iy)
Gives direction of change at each pixel



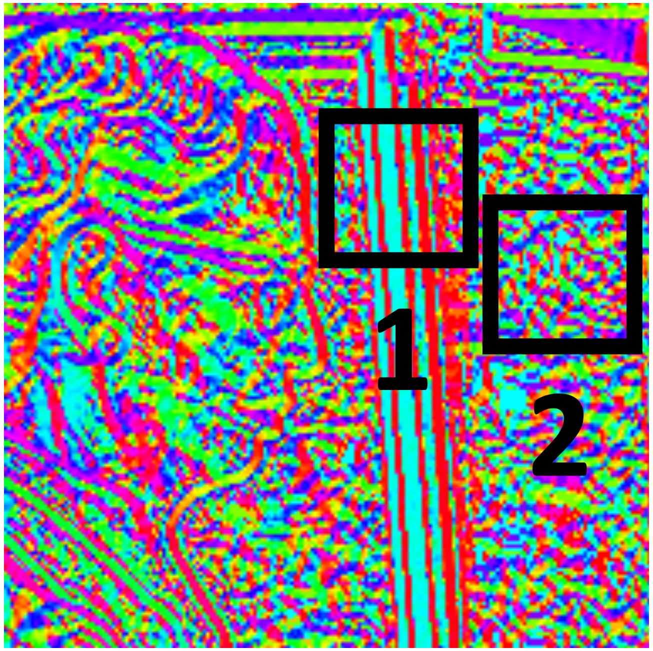


Showing the gradient direction at every pixel

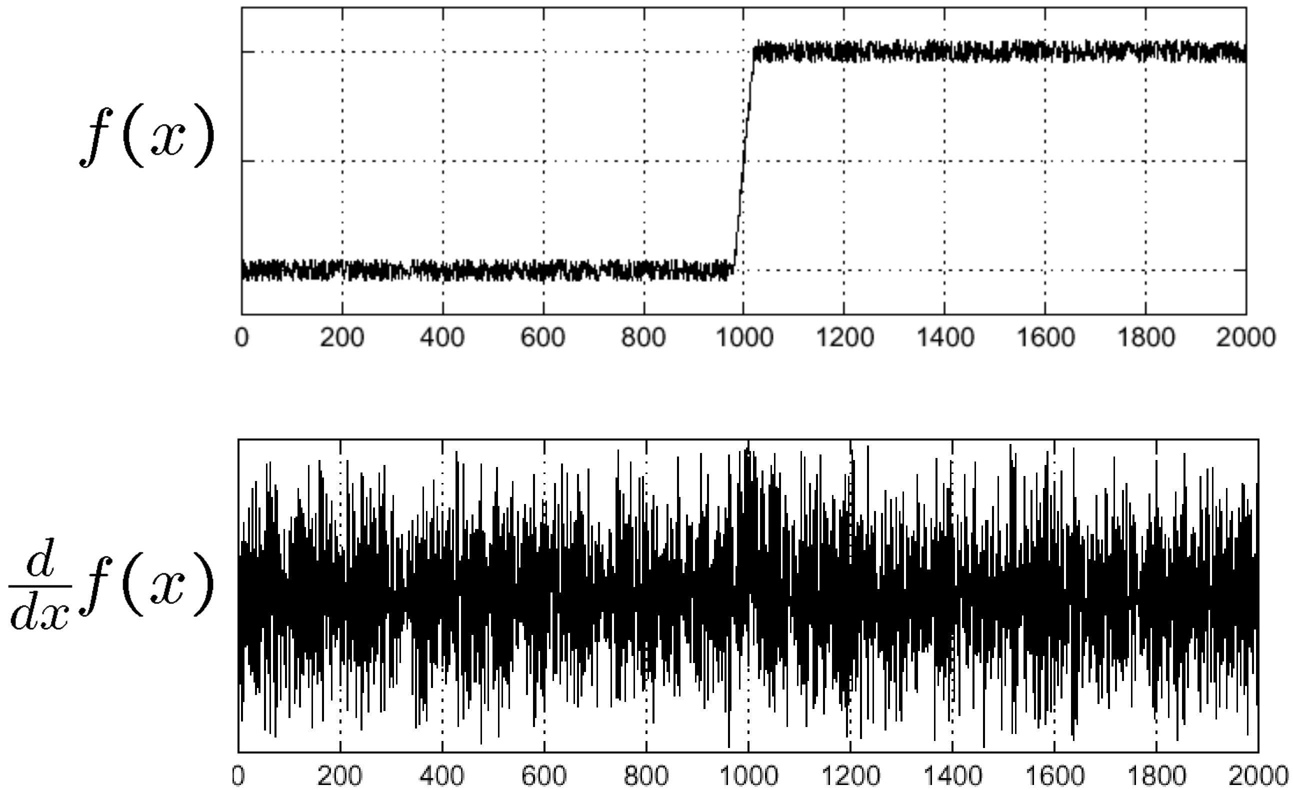
Gradient Direction

atan2(Iy,Ix): orientation

Why is there structure at 1 and not at 2?



Gradients of Noisy Images



Slide Credit: S. Seitz

Gradients of Noisy Images

Conv. image + per-pixel noise with

-1	0	1

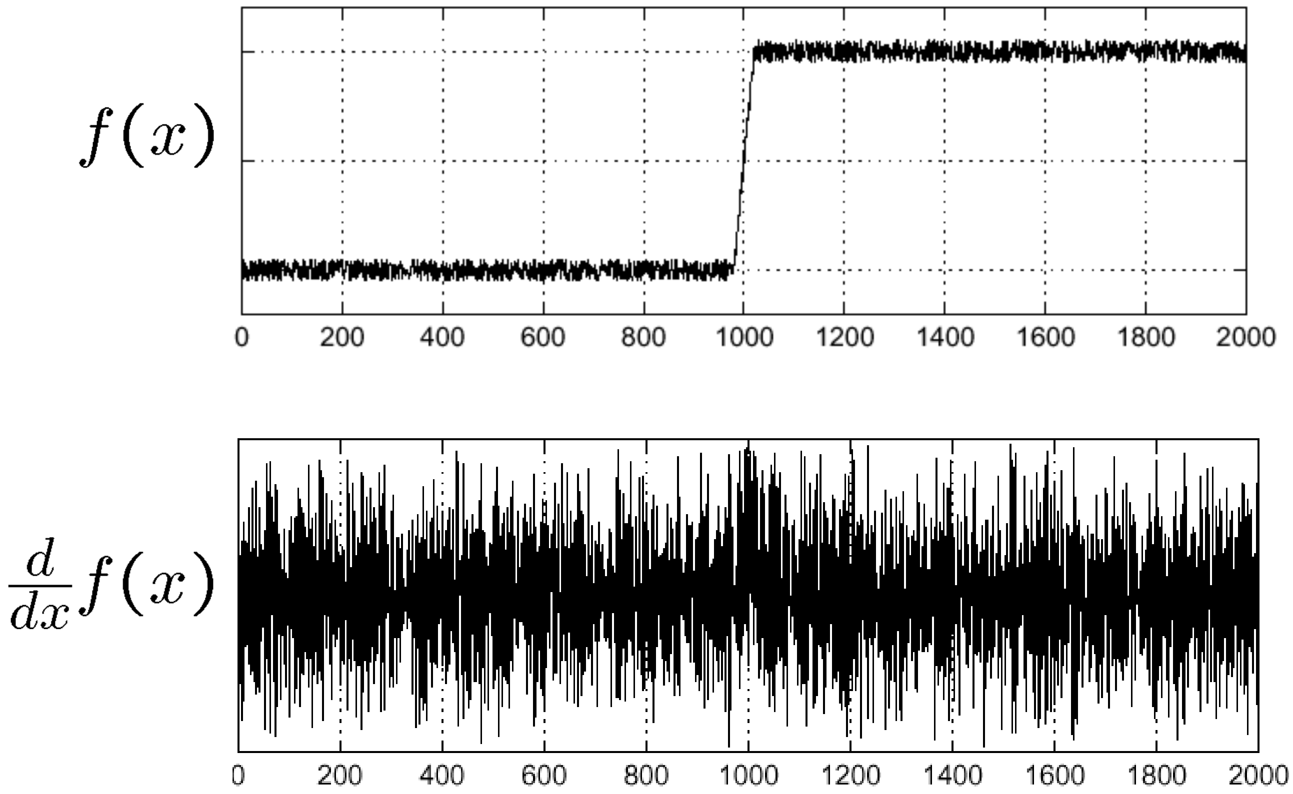
$I_{i,j} =$ True image $\qquad \epsilon_{i,j} \sim N(0, \sigma^2)$

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

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

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

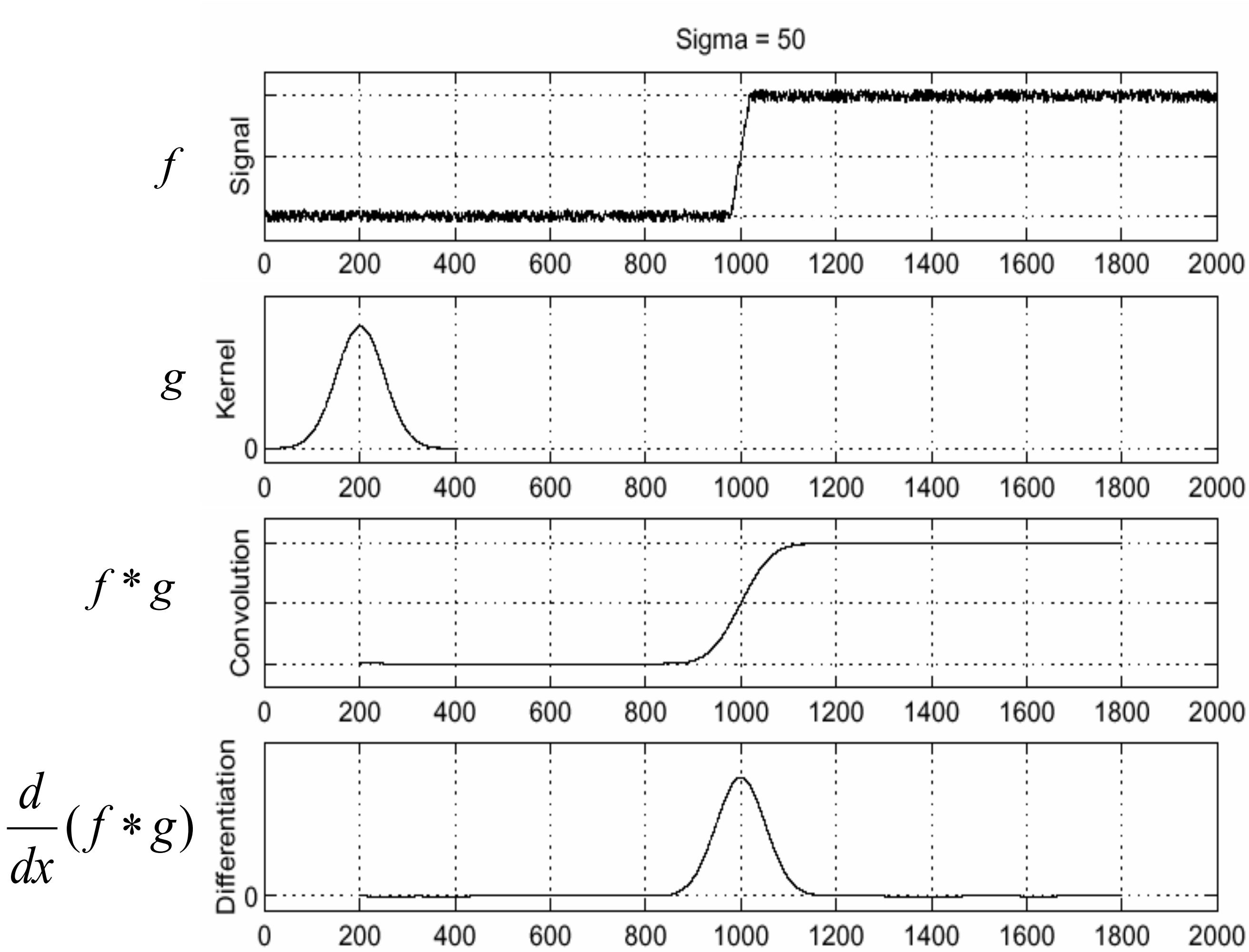
Gradients of Noisy Images



How can we use the last class to fix this?

Slide Credit: S. Seitz

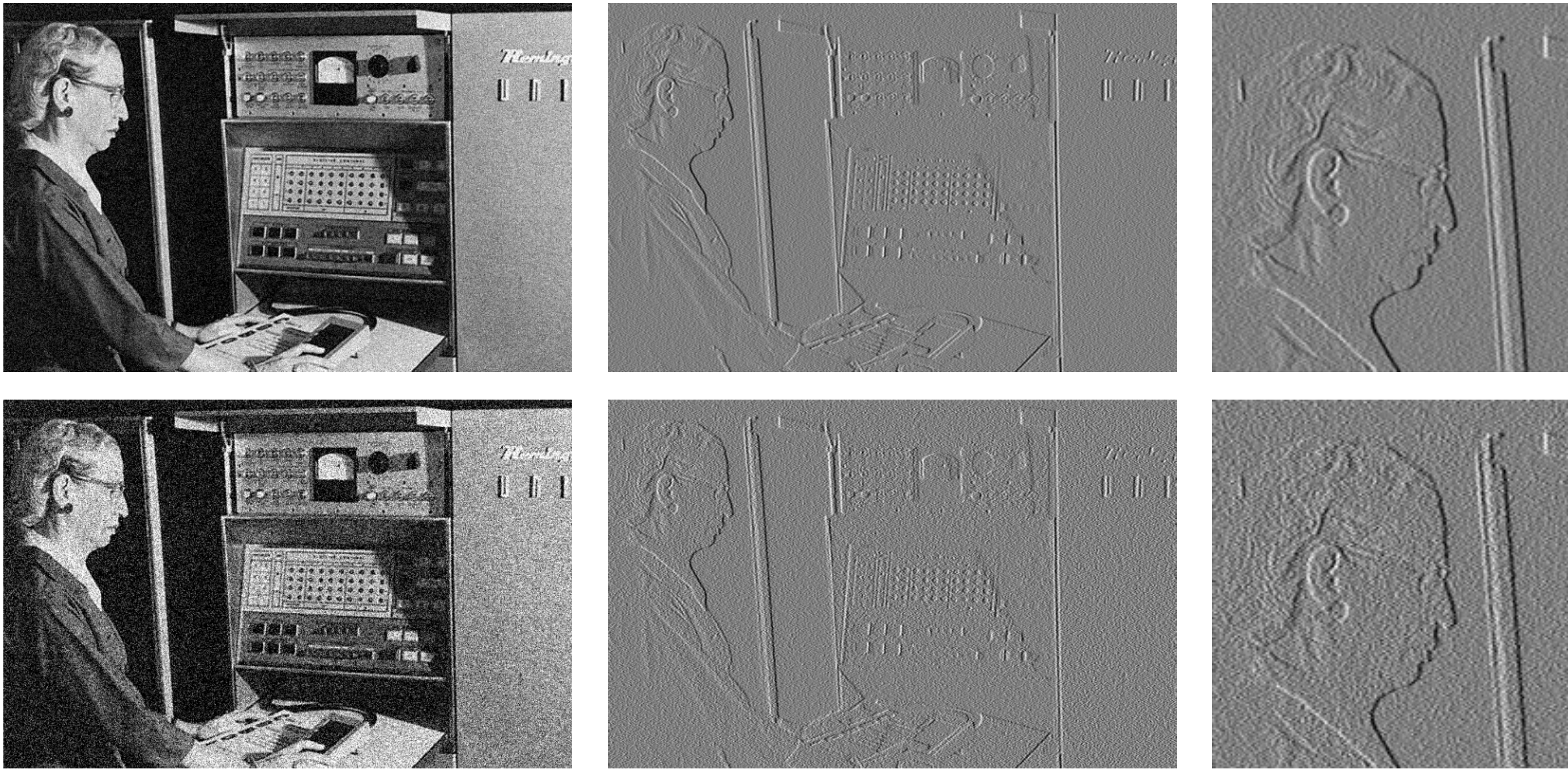
Handling Noise



Slide Credit: S. Seitz

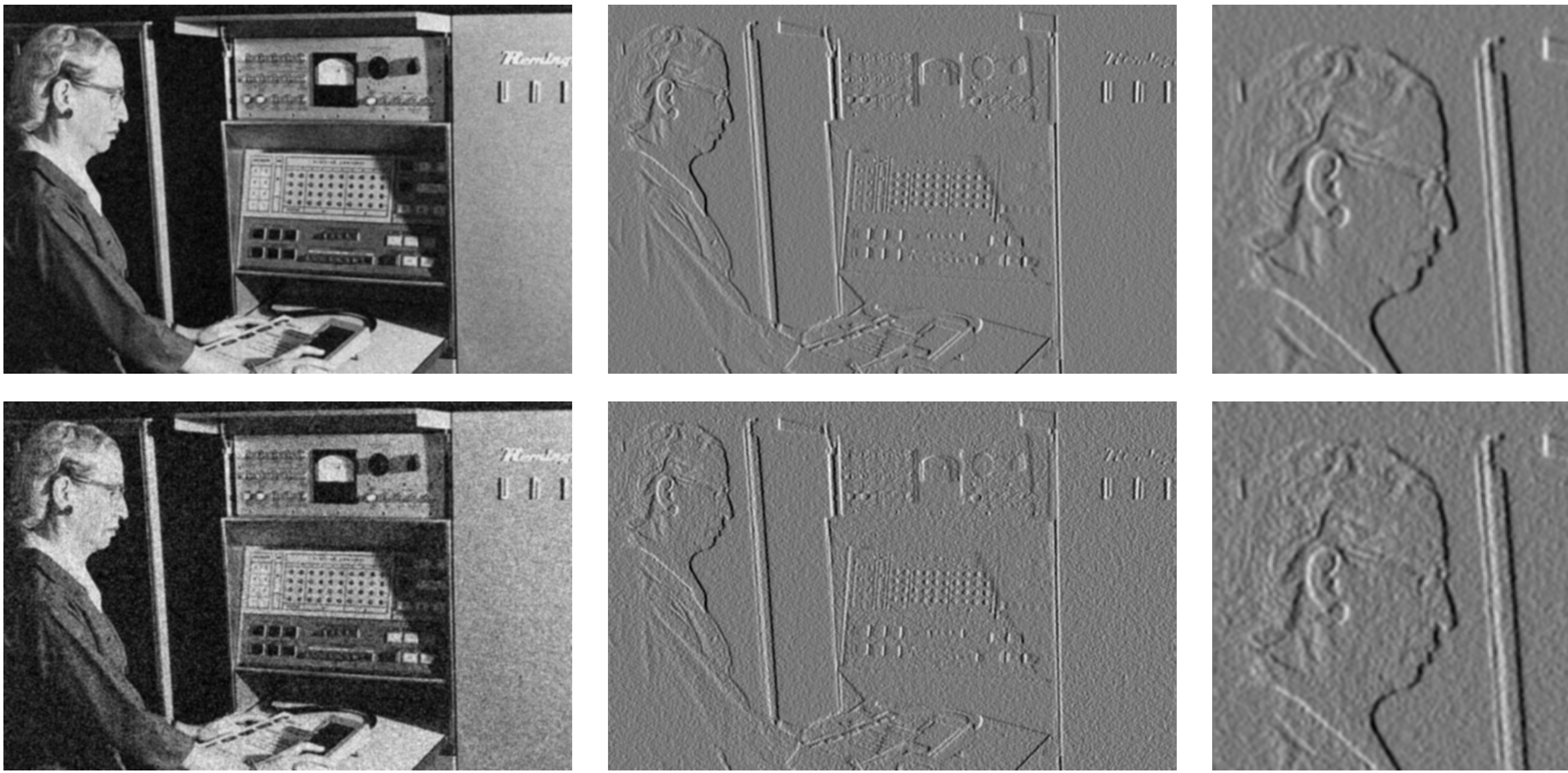
Noise in 2D

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



Noise + Smoothing

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



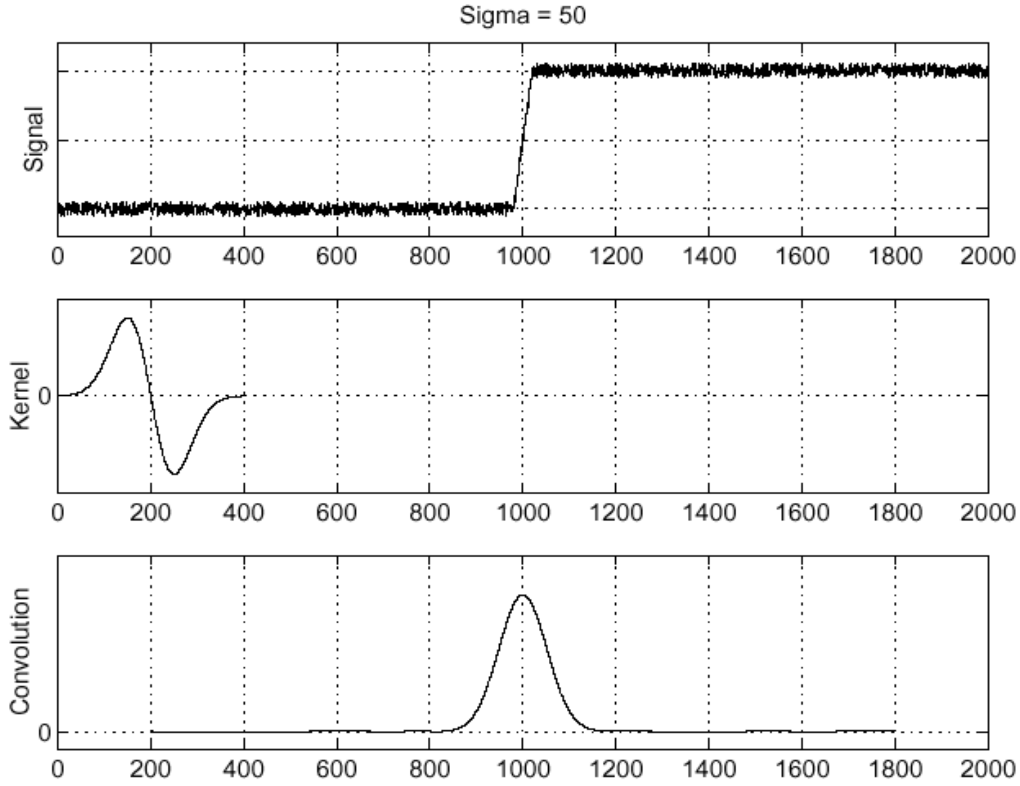
Smooth + Derivative in One Pass (1D)

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

f

$\frac{d}{dx} g$

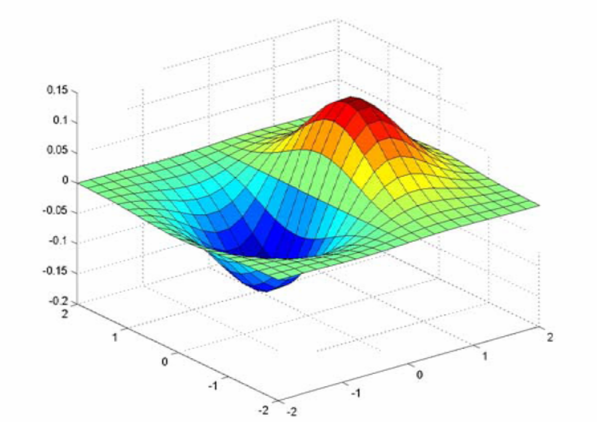
$f * \frac{d}{dx} g$

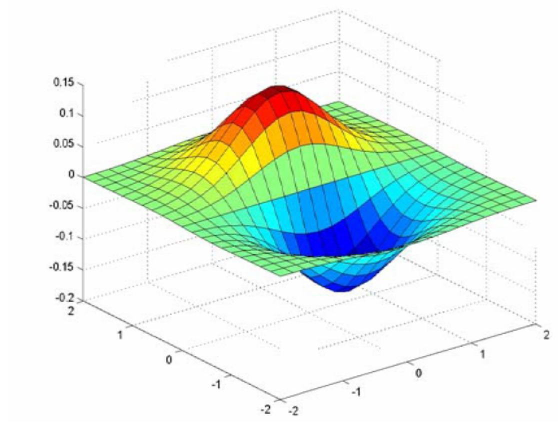


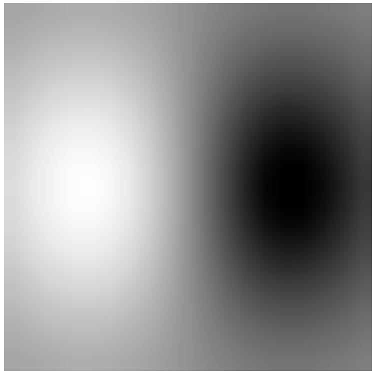
Slide Credit: S. Seitz

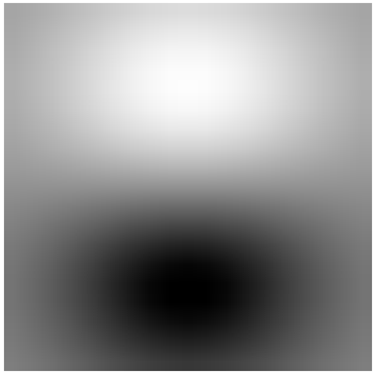
Smooth + Derivative in One Pass (2D)

Gaussian Derivative Filter





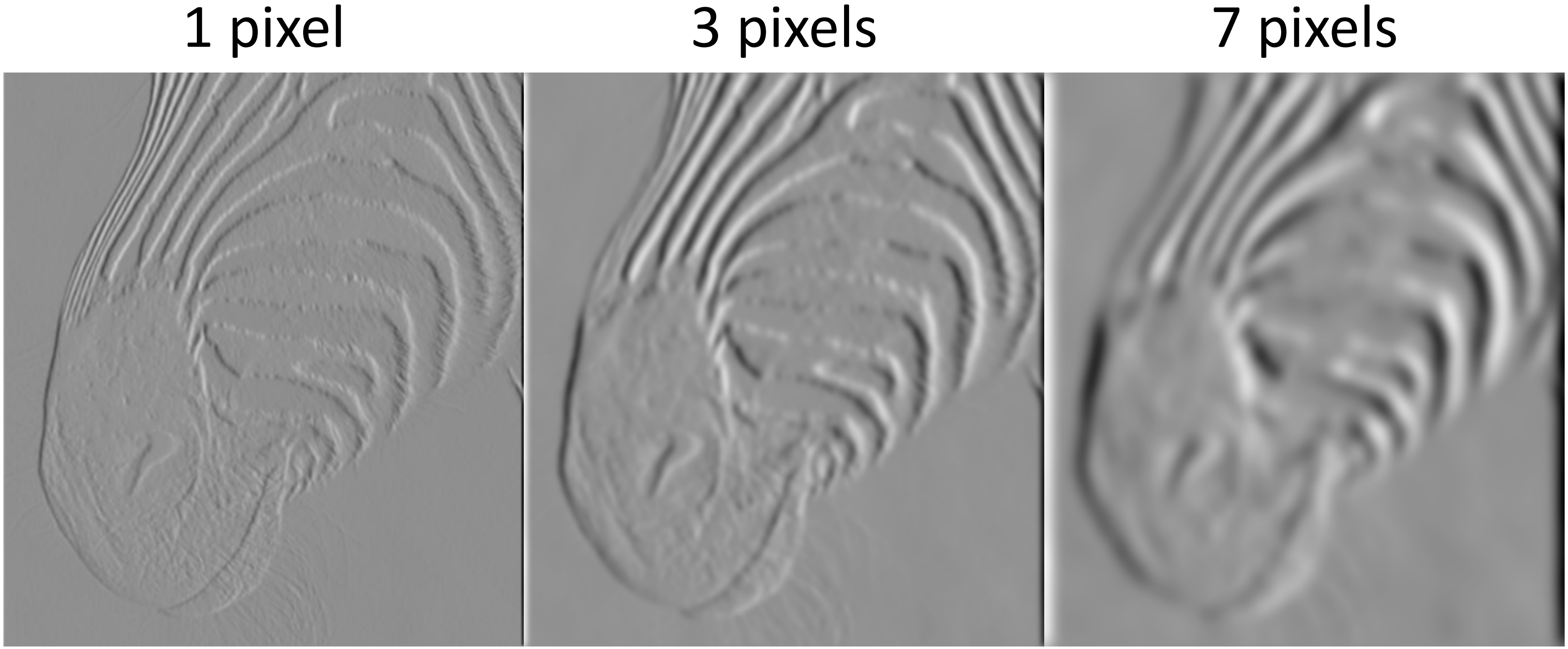




Which one finds the X direction?

Slide Credit: L. Lazebnik

Gaussian Derivative Filter

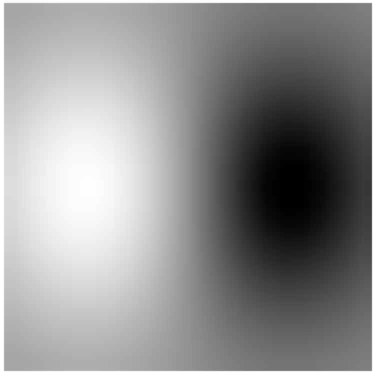


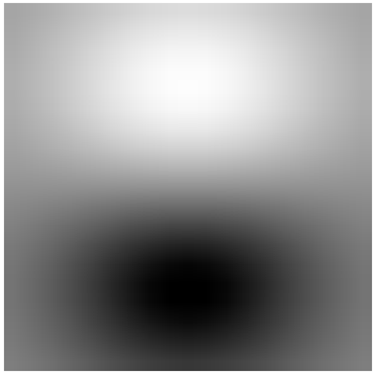
Removes noise, but blurs edge

Slide Credit: D. Forsyth

Filters We've Seen

Gaussian Derivative



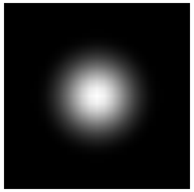
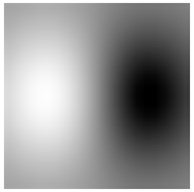
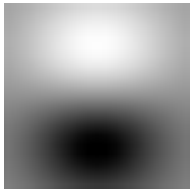


Sobel Filter

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

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 active area of research

Part II: Corners

Corners

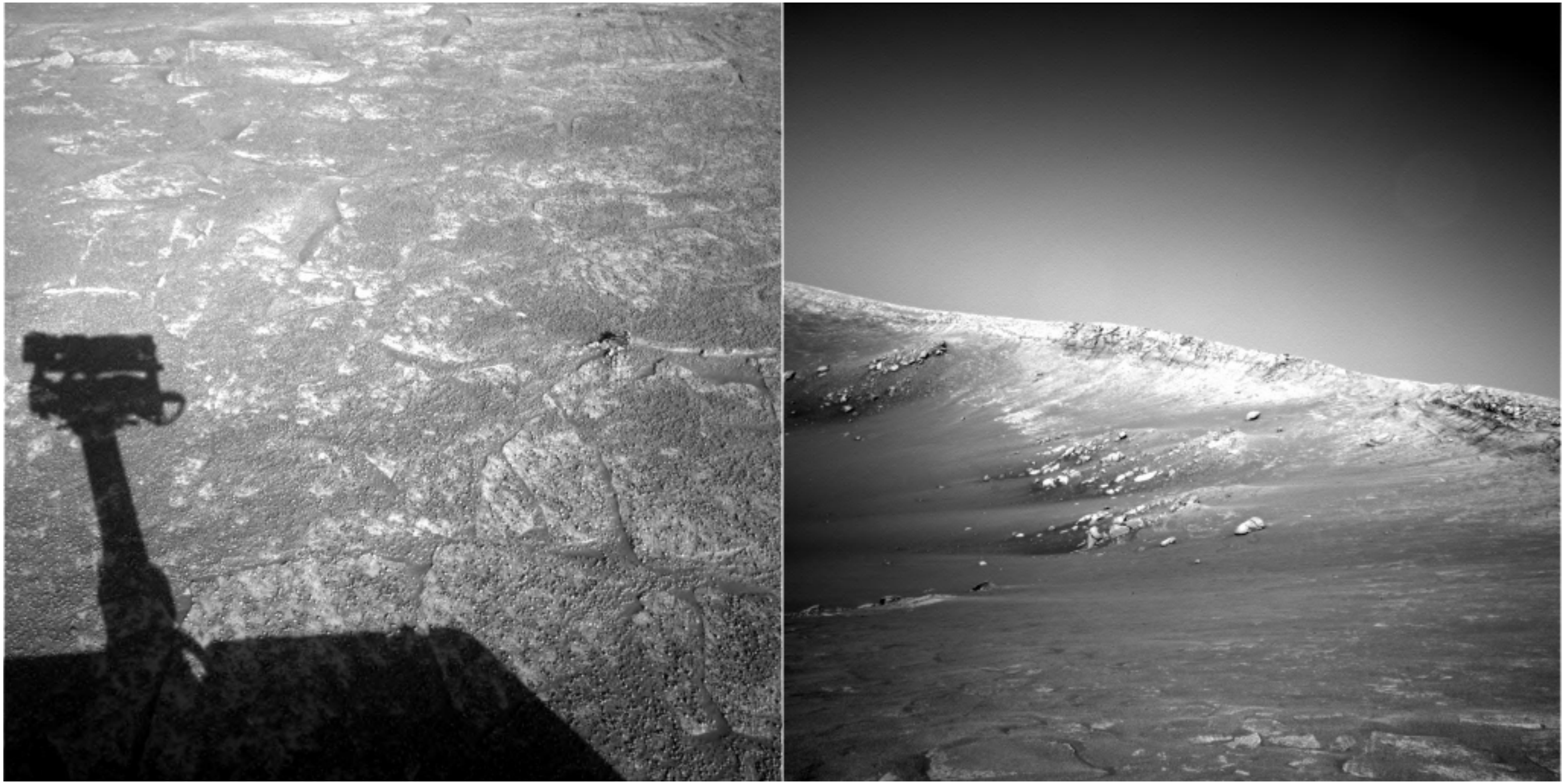
Slide Credit: S. Lazebnik

Corners: Desired Properties

- **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

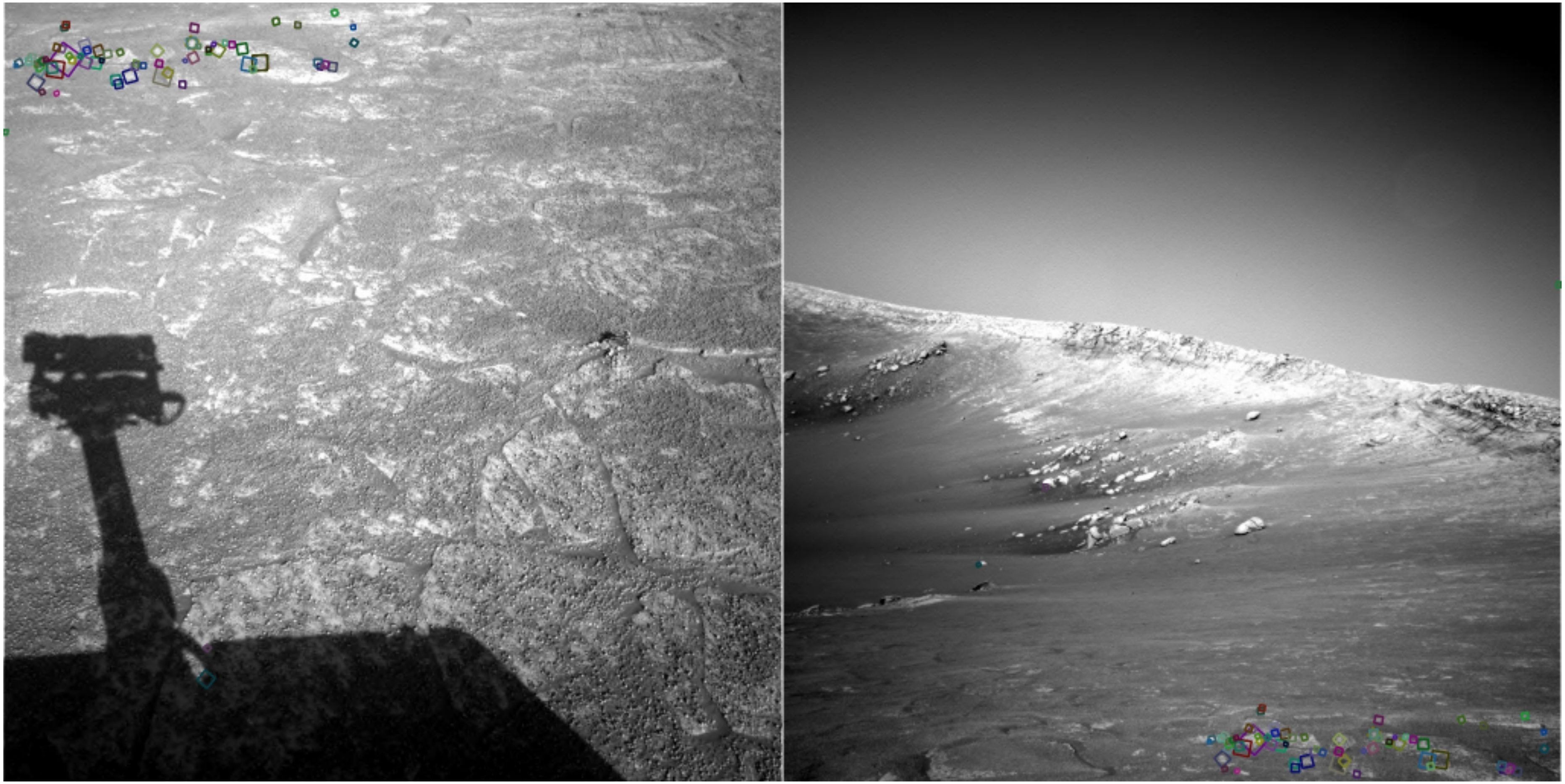
Corners: Hard Example



Can you find the correspondences?

Slide credit: N. Snavely

Corners: Hard Example

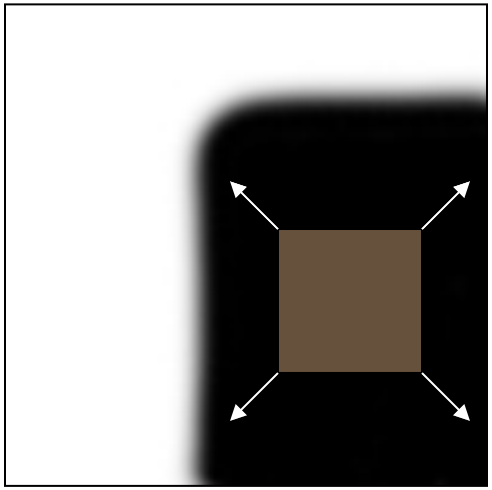


Look for the colored squares

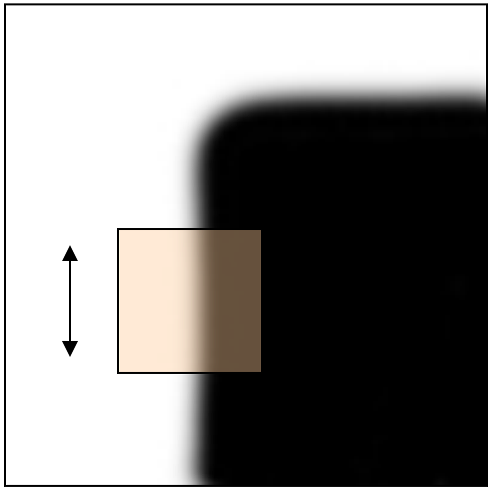
Slide credit: N. Snavely

Corners: Intuition

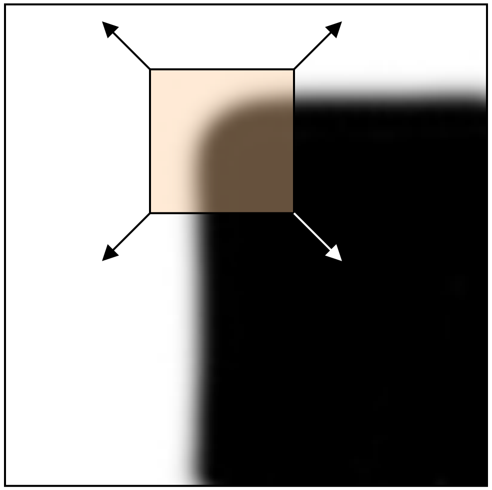
Should see where we are based on small window, or any shift → big intensity change.



"flat" region: no change in all directions



"edge": no change along the edge direction



"corner": significant change in all directions

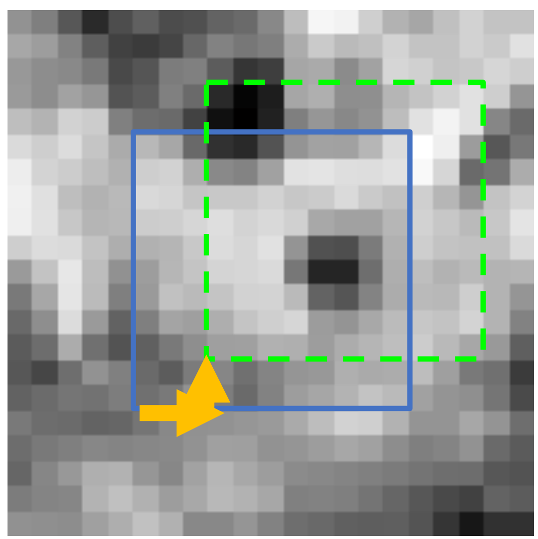
Slide Credit: S. Lazebnik

Formalizing Corner Detection

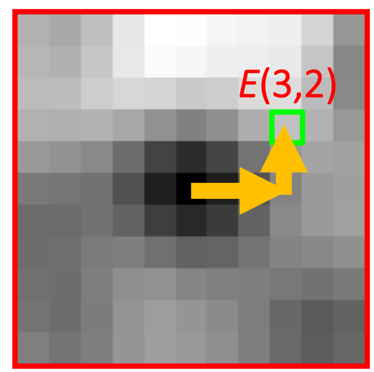
Sum of squared differences between image and image shifted u,v pixels over.

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

Image I(x,y)



Plot of E(u,v)



Slide Credit: S. Lazebnik

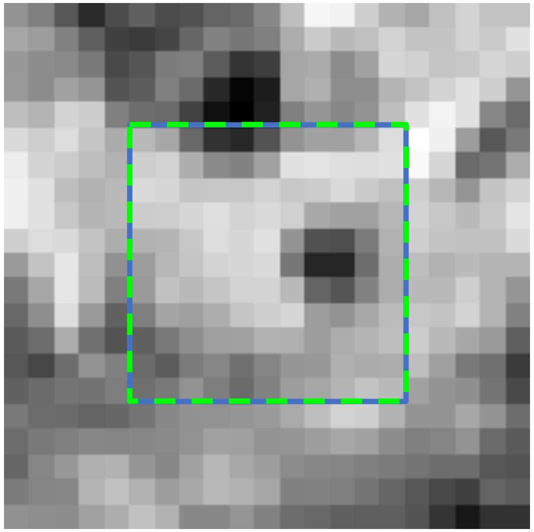
Formalizing Corner Detection

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

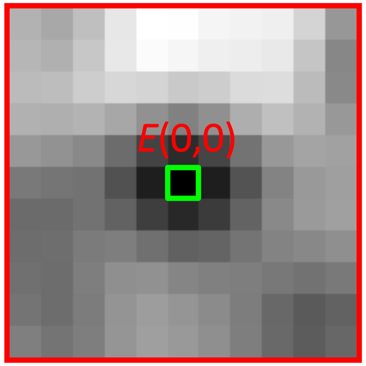
$$E(u,v) = \sum_{(x,y)\in W} \left(I[x+u, y+v] - I[x,y]\right)^2$$

What's the value of E(0,0)?

Image I(x,y)



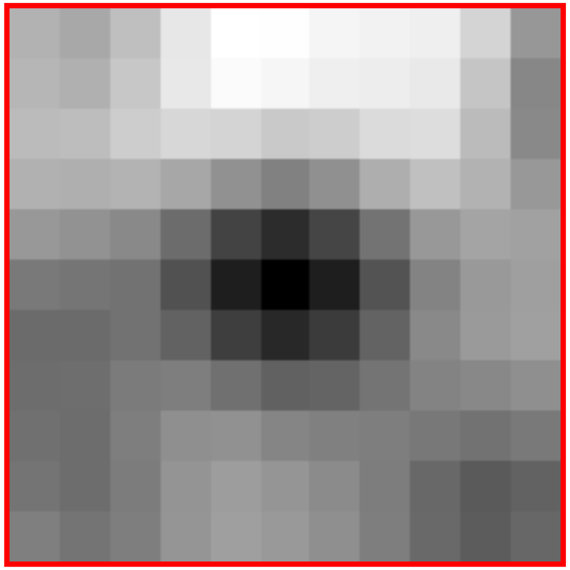
Plot of E(u,v)



Slide Credit: S. Lazebnik

Formalizing Corner Detection

Can compute E[u,v] for any window and u,v.
But we'd like a simpler function of u,v.



Slide Credit: S. Lazebnik

Aside: Taylor Series for Images

Recall Taylor Series:

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

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

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

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

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

Cancel

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

Expand

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

For brevity: $I_x = I_x[x, y]$, $I_y = I_y[x, y]$

Formalizing Corner Detection

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

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

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

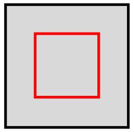
M is called the second moment matrix

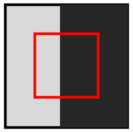
Second Moment Matrix: Intuition

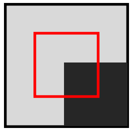
Pretend for now gradients are either vertical or horizontal at a pixel (so Ix Iy = 0)

Obviously Wrong!

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

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 **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.

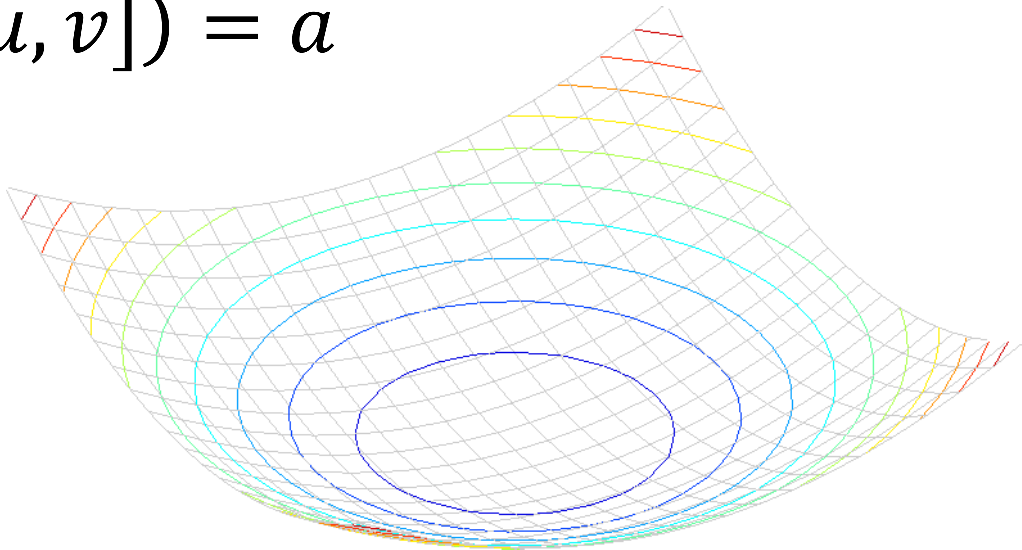


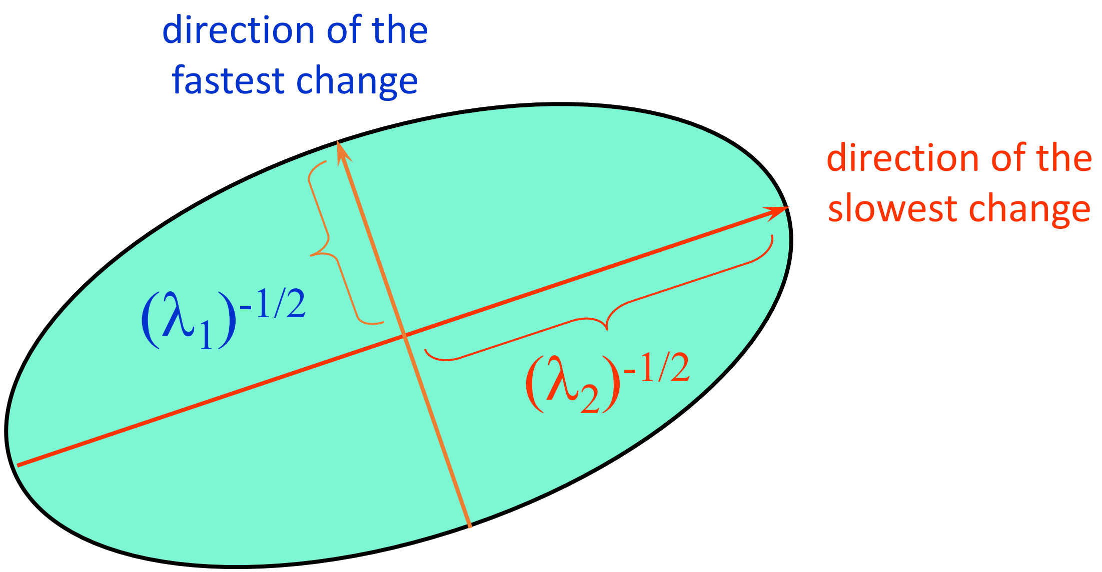
Diagram credit: S. Lazebnik

Review: Quadratic Forms

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

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

What are λ_1 and λ_2?



Slide credit: S. Lazebnik

Second Moment Matrix

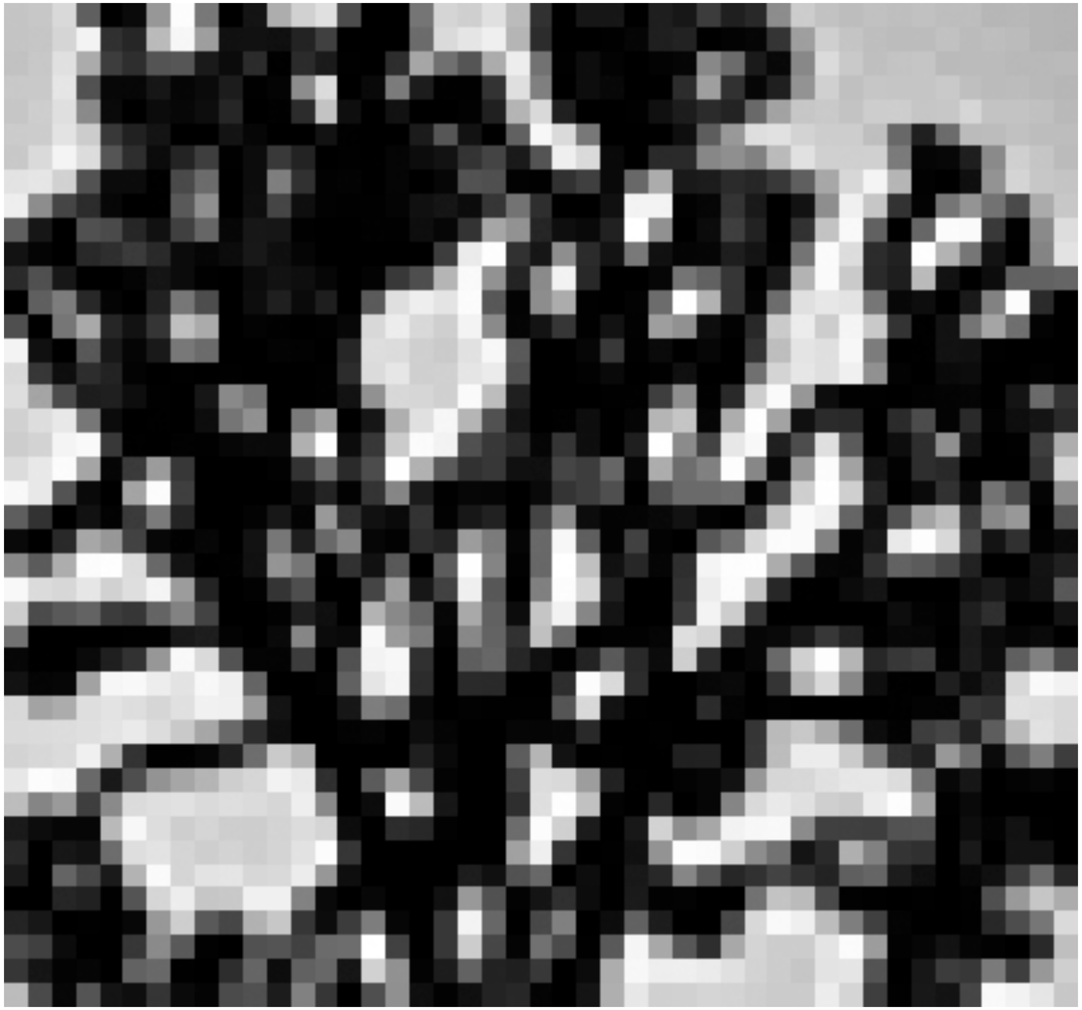
The second moment matrix tells us how quickly the image changes and in which directions.

Directions

$$M = \overbrace{\begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix}}^{\text{Can compute at each pixel}} = \boldsymbol{R}^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \boldsymbol{R}$$

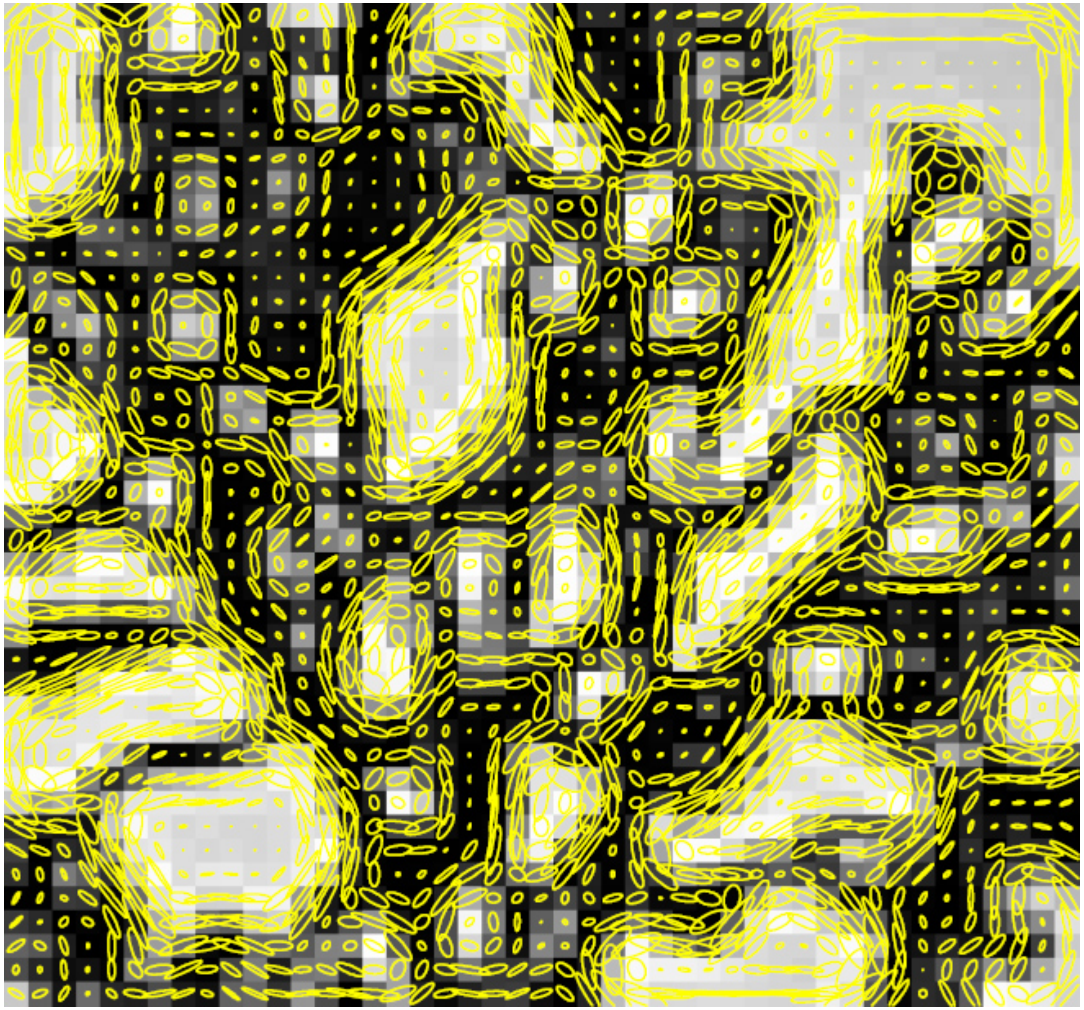
Amounts

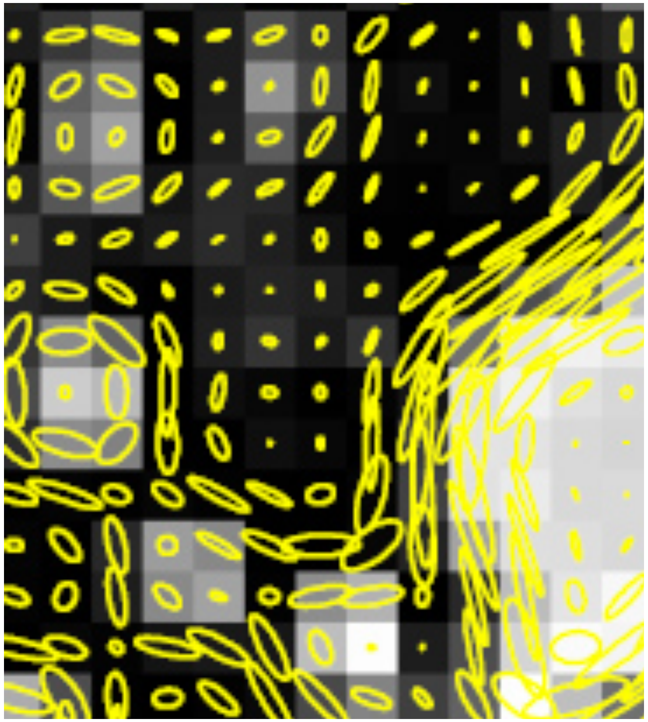
Visualizing Second Moment Matrix



Slide credit: S. Lazebnik

Visualizing Second Moment Matrix

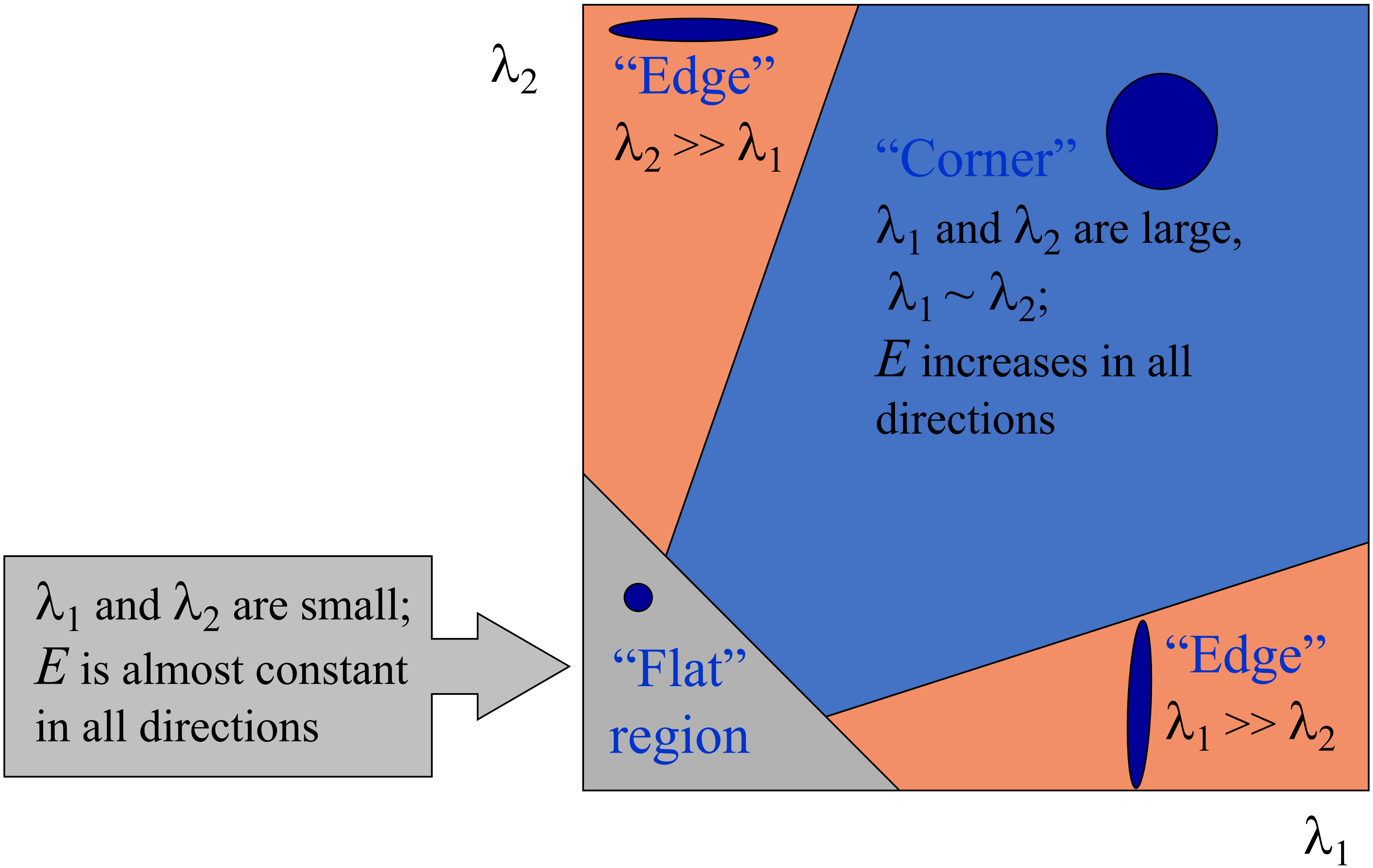




Technical note: M is often best *visualized* by first taking inverse, so long edge of ellipse goes along edge

Slide credit: S. Lazebnik

Eigenvalues of M

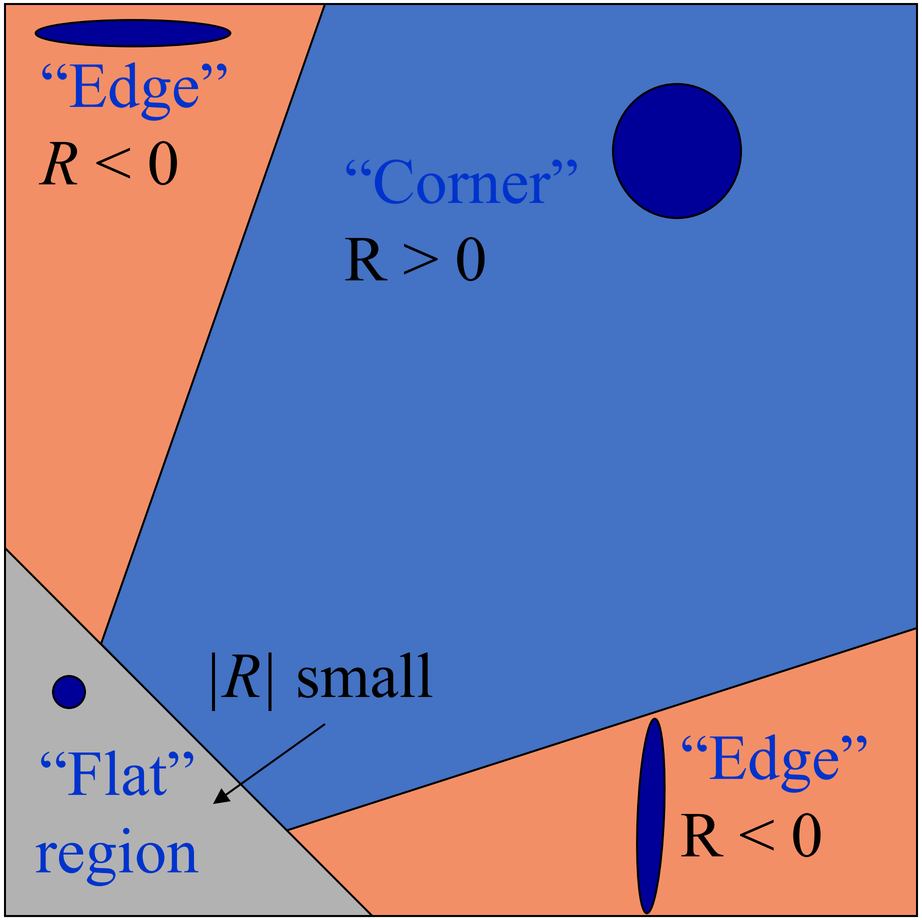


Slide credit: S. Lazebnik; Note: this refers to previous ellipses, not original M ellipse. Other slides on the internet may vary

Eigenvalues of M

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

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

Harris Corner Detector

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

Harris Corner Detector

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

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

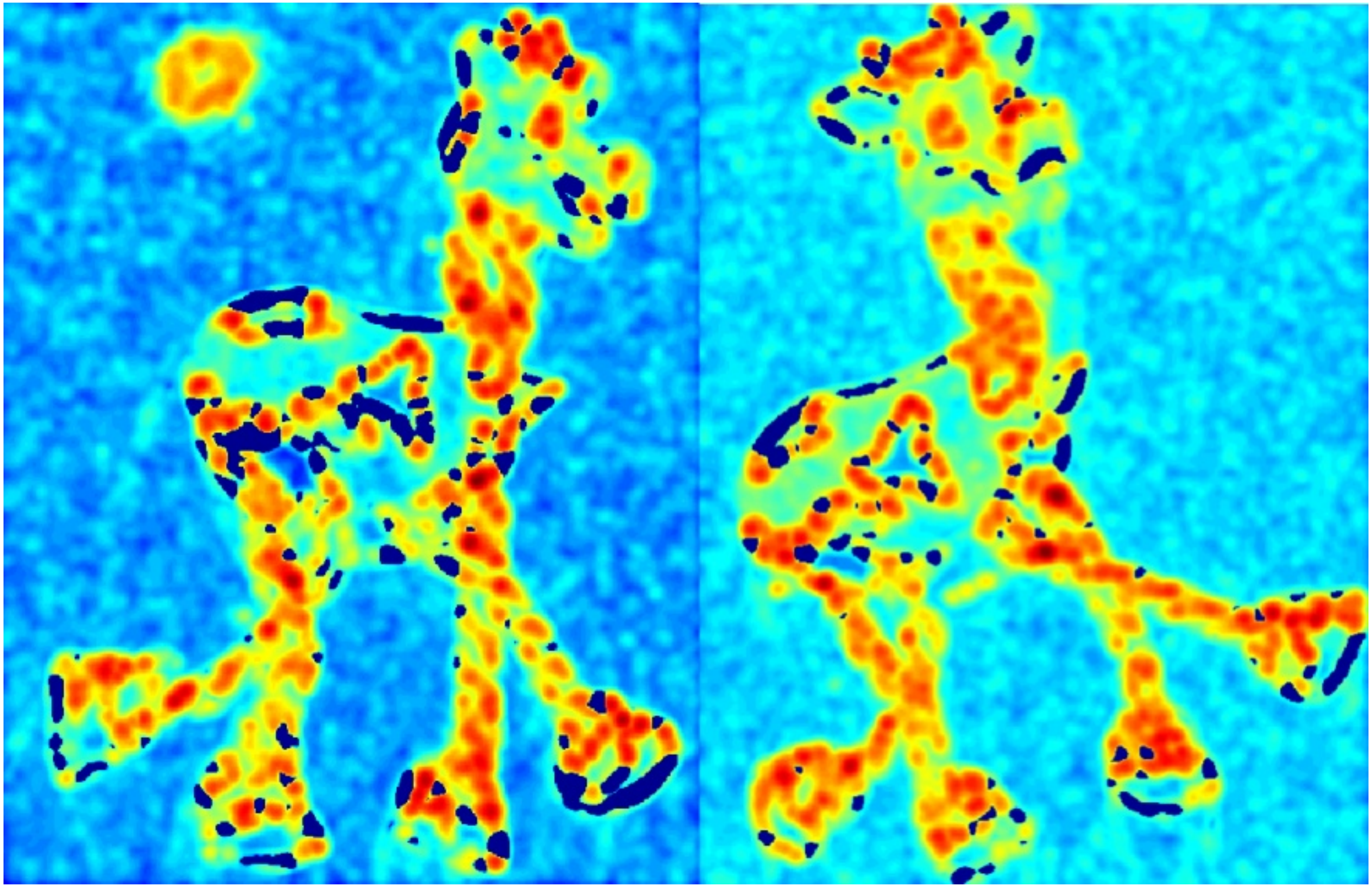
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Slide credit: S. Lazebnik

Computing R

Slide credit: S. Lazebnik

Computing R



Slide credit: S. Lazebnik

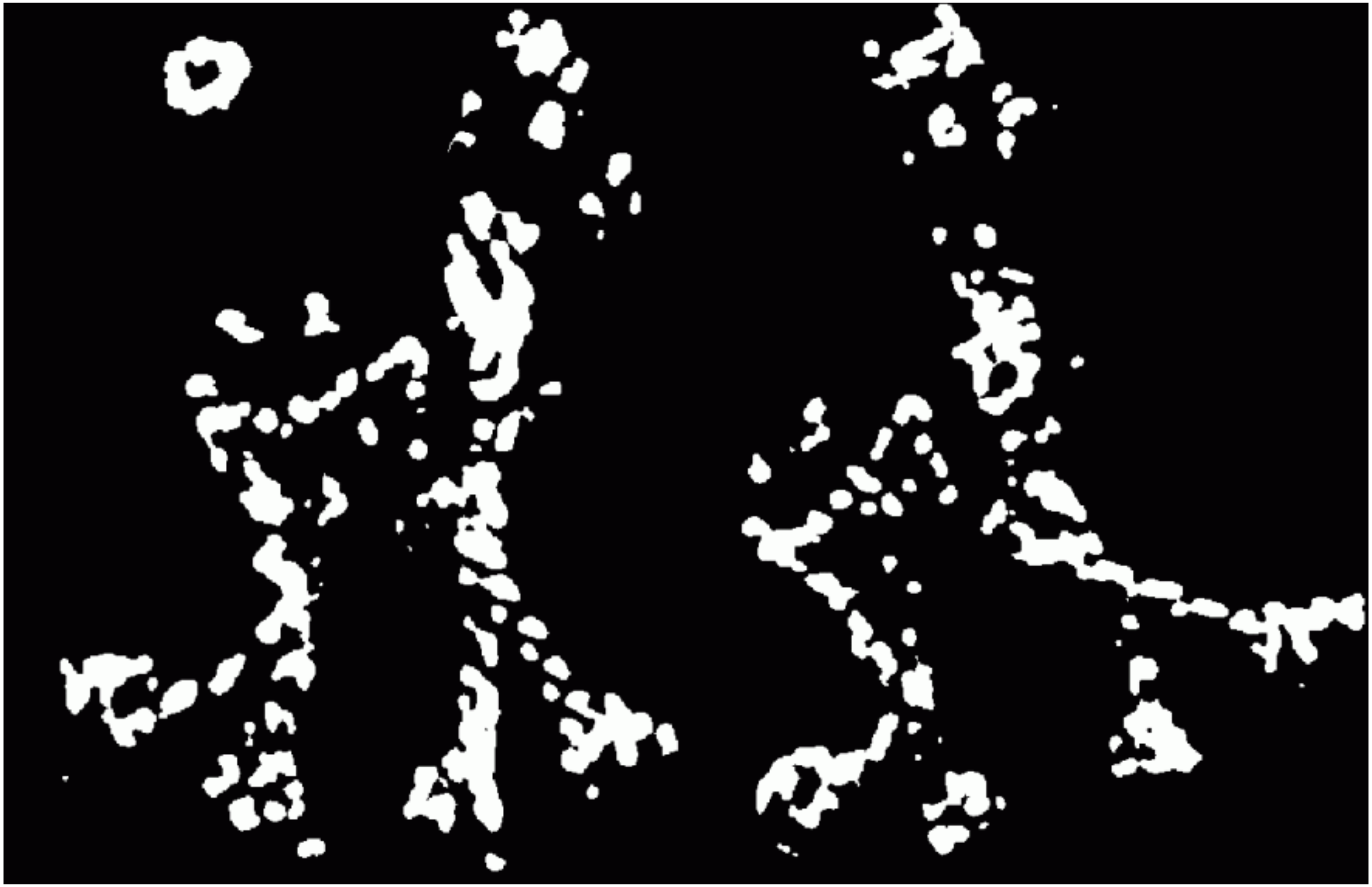
Harris Corner Detector

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

Harris Corner Detector



Slide credit: S. Lazebnik

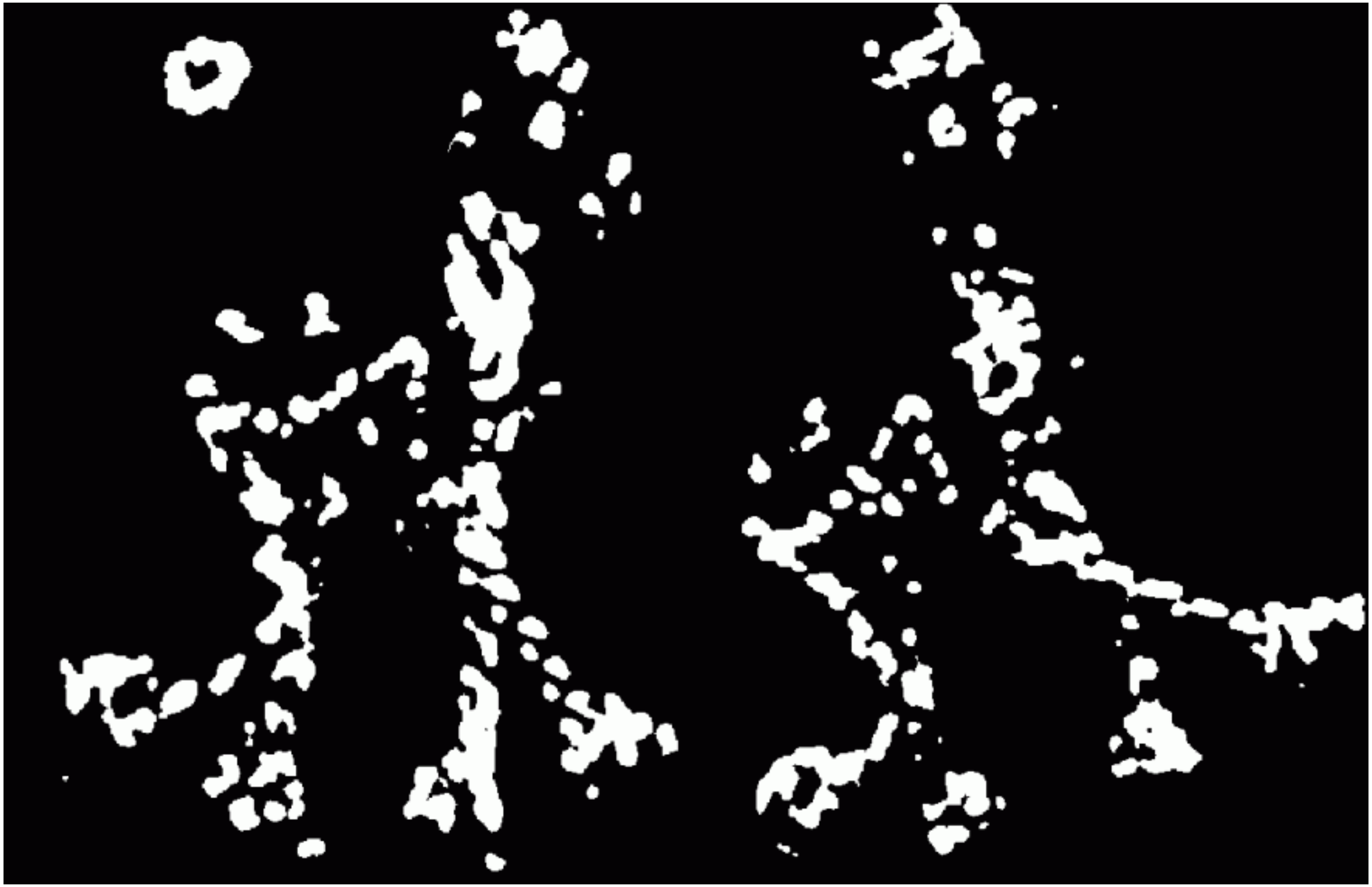
Harris Corner Detector

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 (Non-Maxima Suppression, NMS)

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

Harris Corner Detector



Slide credit: S. Lazebnik

Harris Corner Detector: Result

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

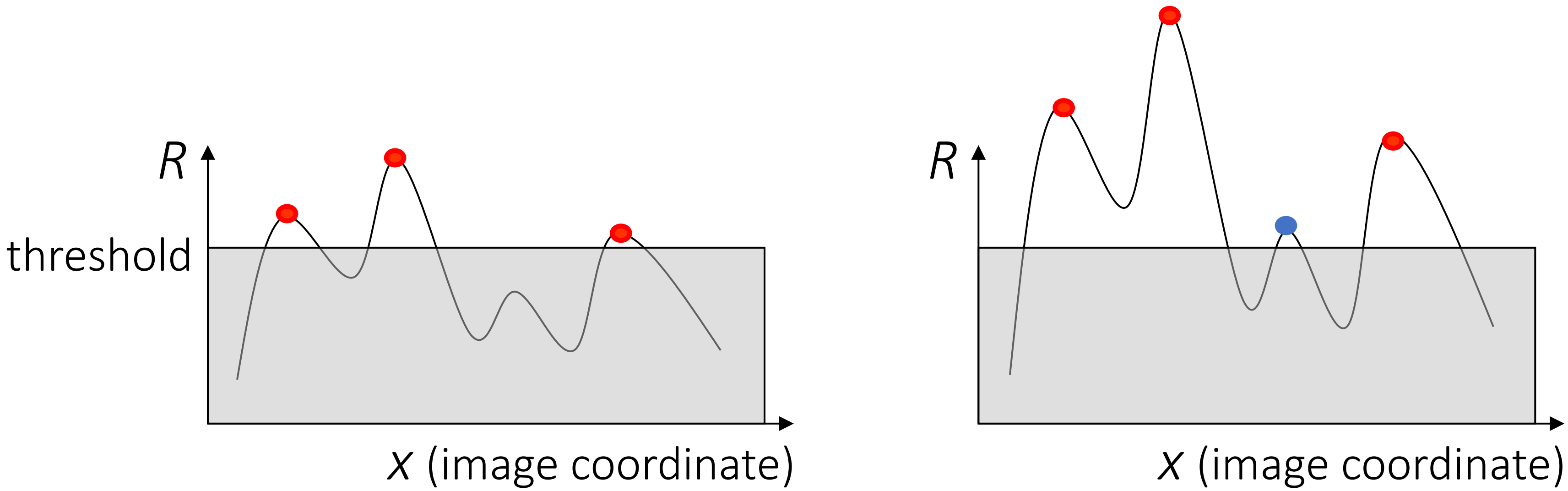
Images may be different in lighting and geometry

Affine Intensity Change

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

M only depends on derivatives, so b is irrelevant

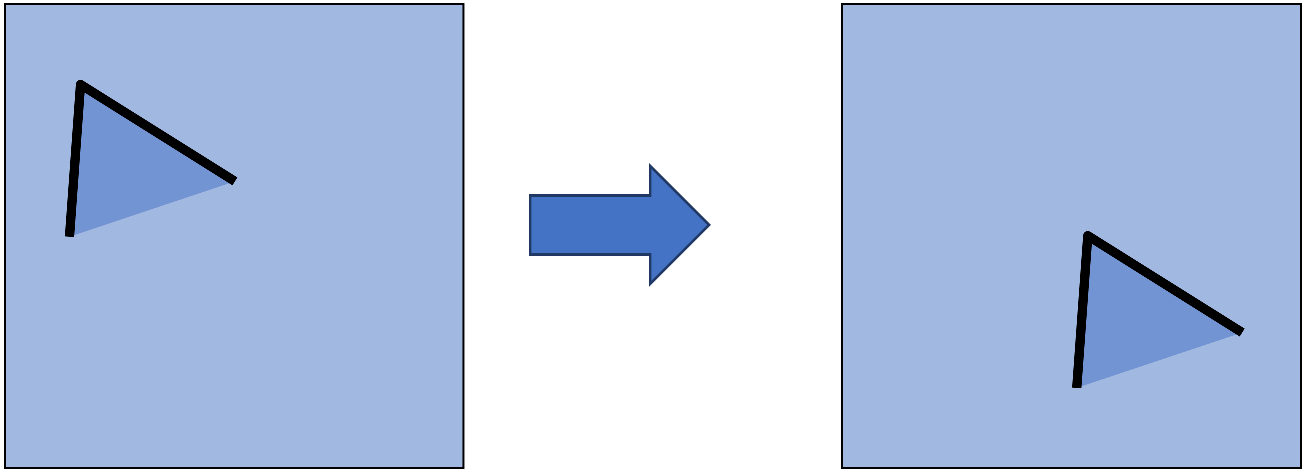
But a scales derivatives and there's a threshold



Partially invariant to affine intensity changes

Slide credit: S. Lazebnik

Image Translation

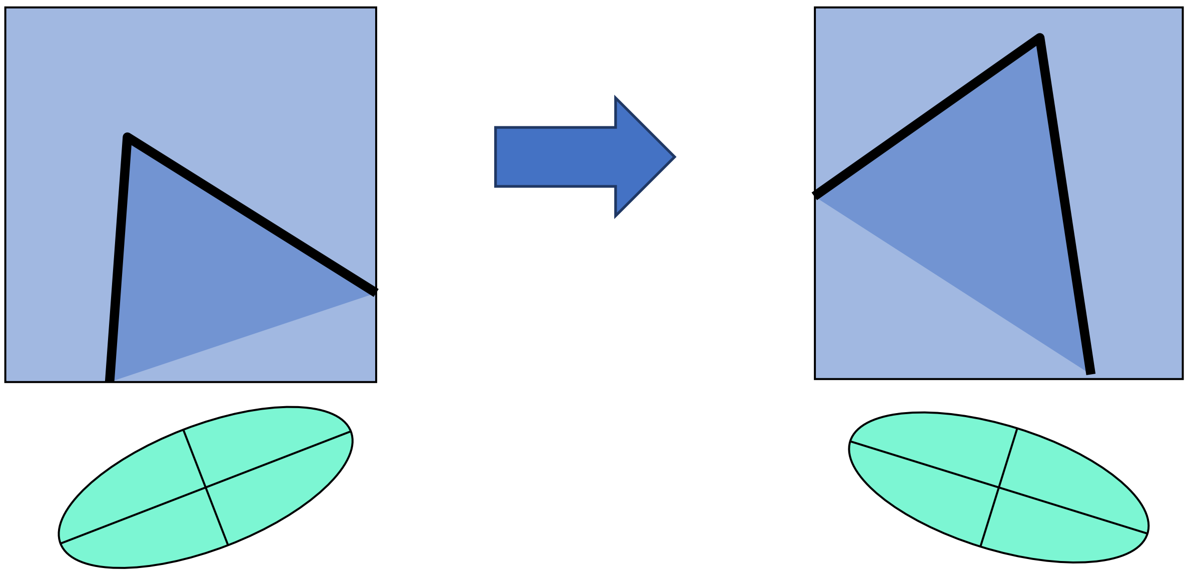


All done with convolution. Convolution is translation invariant.

Equivariant with translation

Slide credit: S. Lazebnik

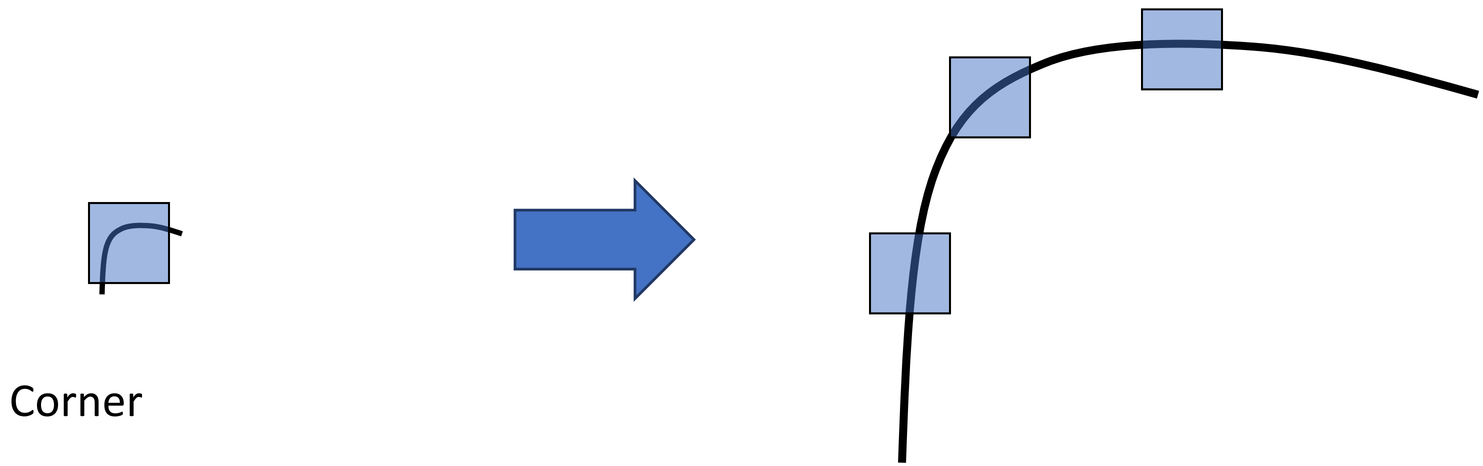
Image Rotation



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

Equivariant with rotation

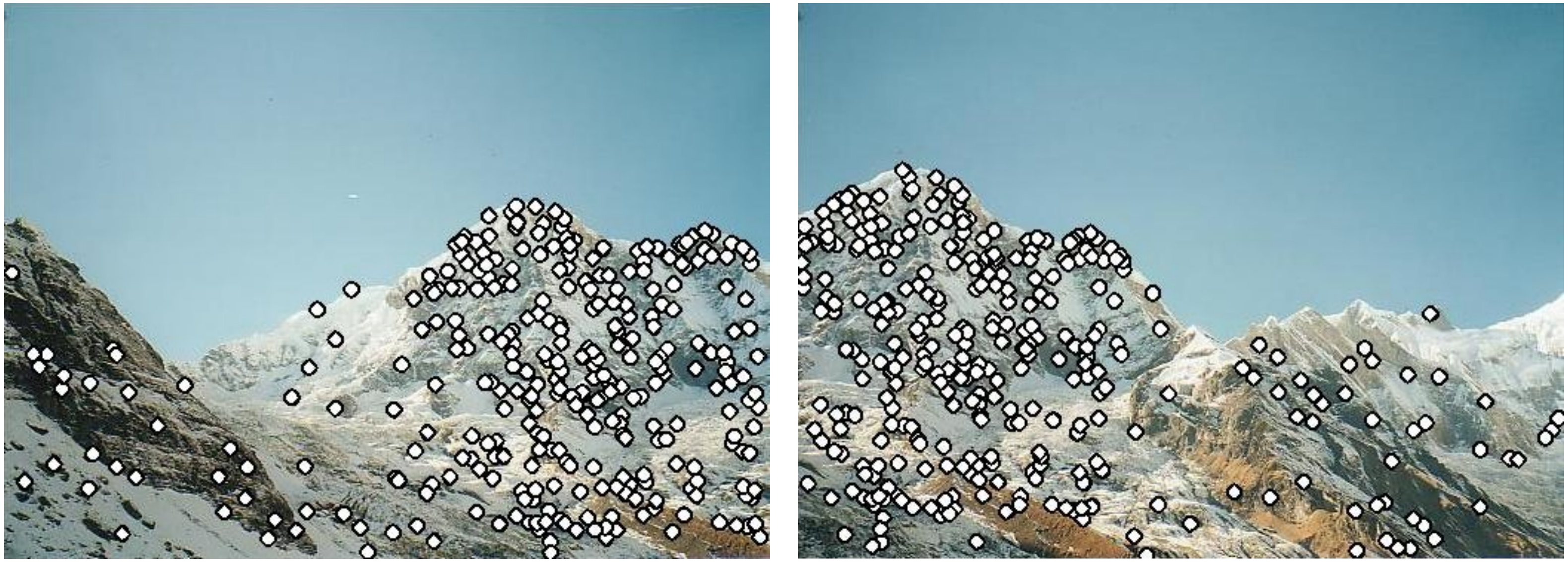
Image Scaling



One pixel can become many pixels and vice-versa.

Not equivariant with scaling

An Alternative Approach



Problem #1 (today): How do we detect points in images?
Problem #2 (next time): How do we describe points in images?

Our points must be robust to viewpoint and illumination change!

Next Time:
Image Descriptors