Lecture 10: Scales and Descriptors

Administrative

HW2 out, due 1 week from tomorrow:
Wednesday 2/19/2019, 11:59pm

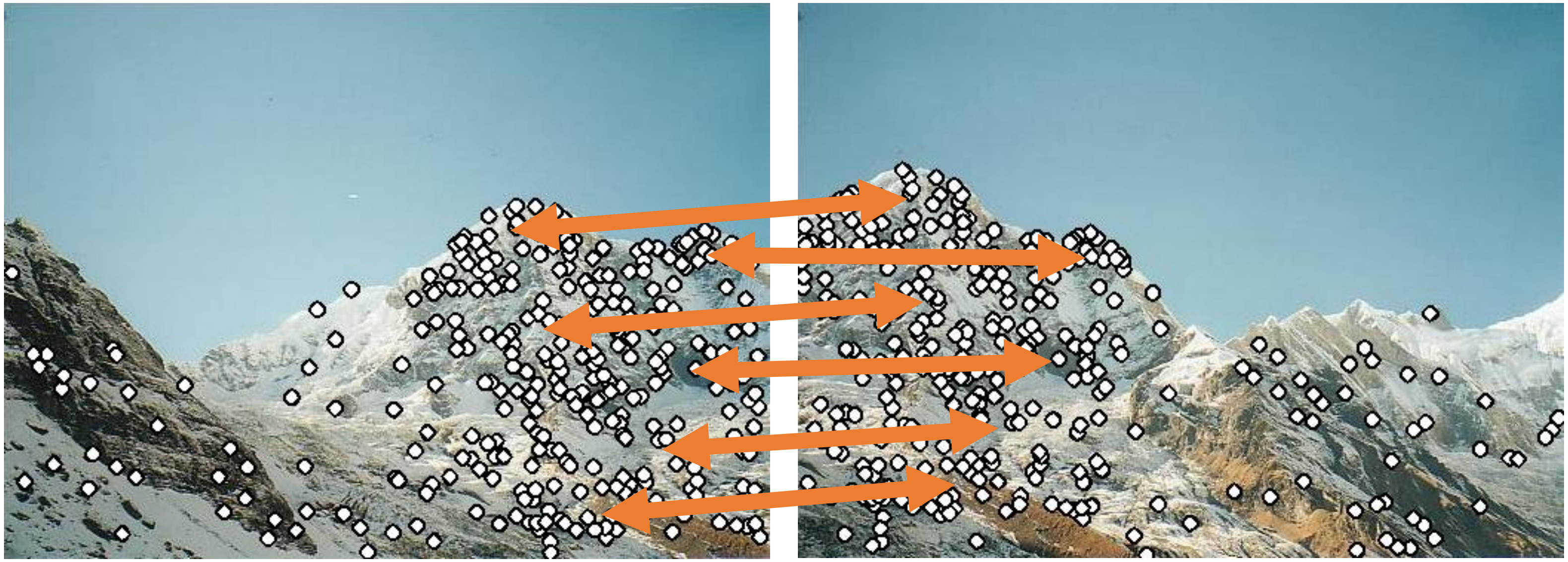
HW3 out tomorrow, due 2 weeks from Friday:
Friday 2/27, 11:59pm

Last Time: Motivation

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

Last Time: Finding + Matching

Finding and Matching



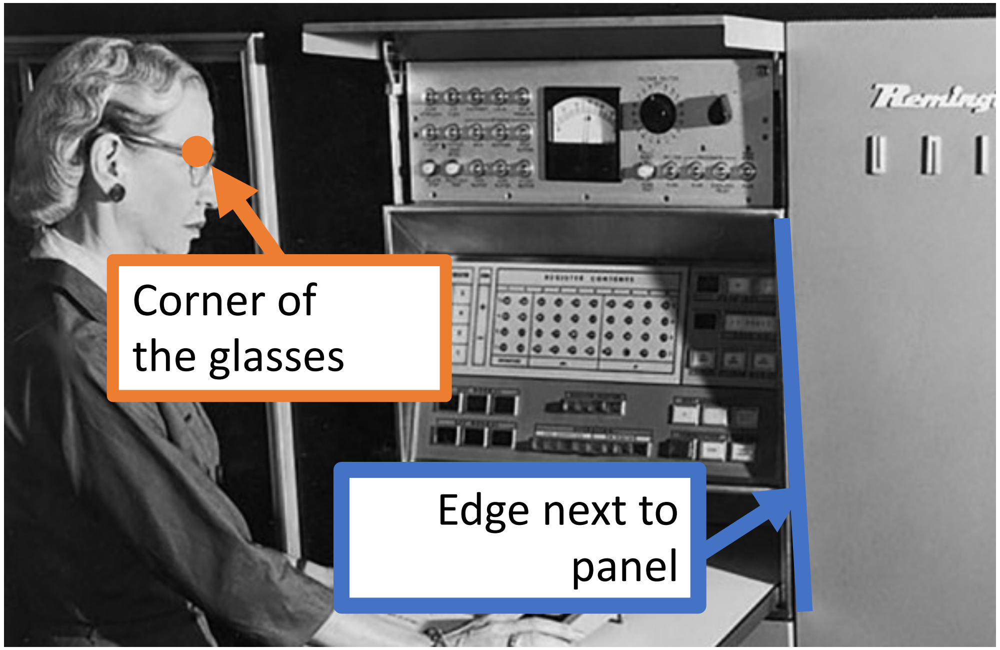
1: find corners+features

2: match based on local image data

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

Last Time: Edges + Corners

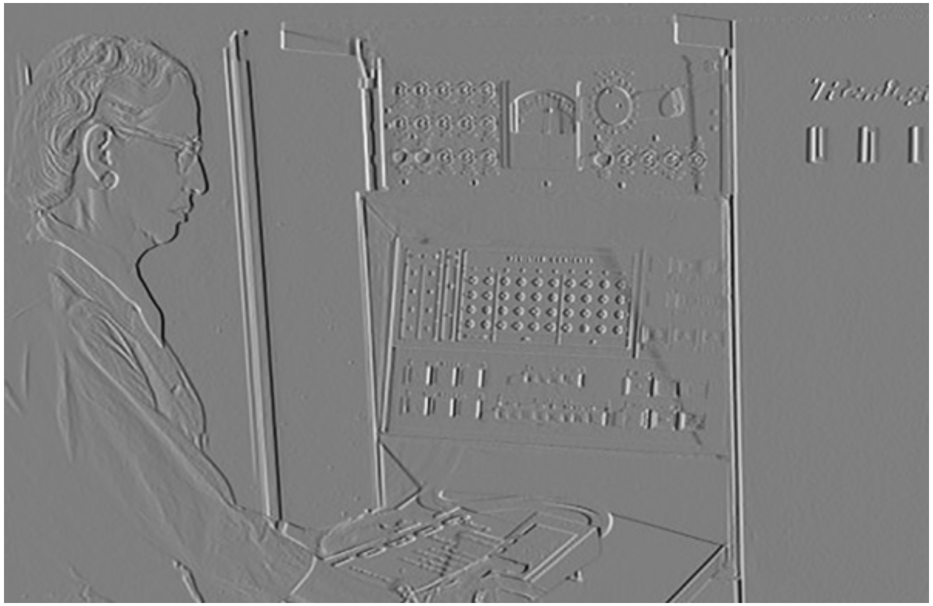
Part 1: Finding Edges
Part 2: Finding Corners



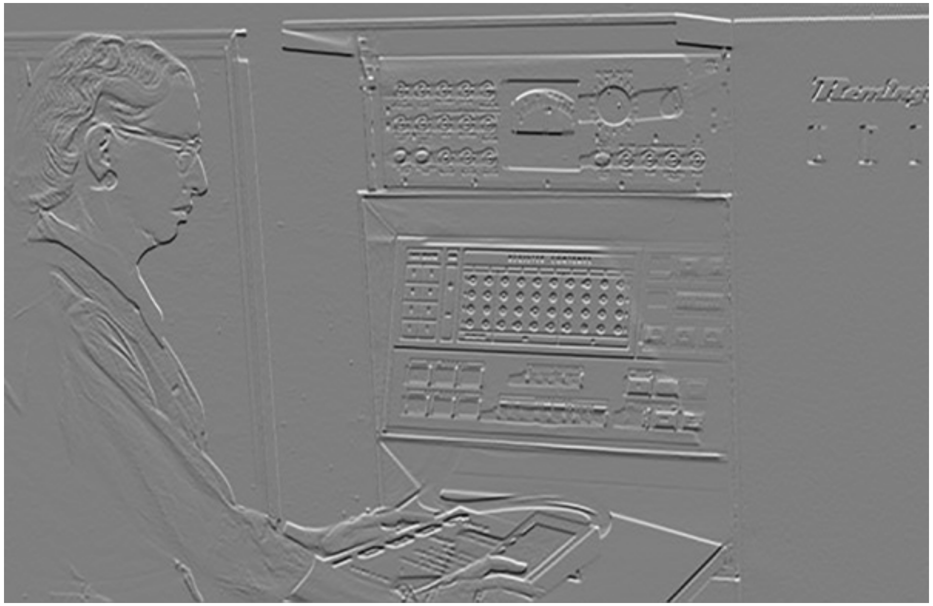
Last Time: Edges via Image Gradients

Compute derivatives Ix and Iy with filters

Ix

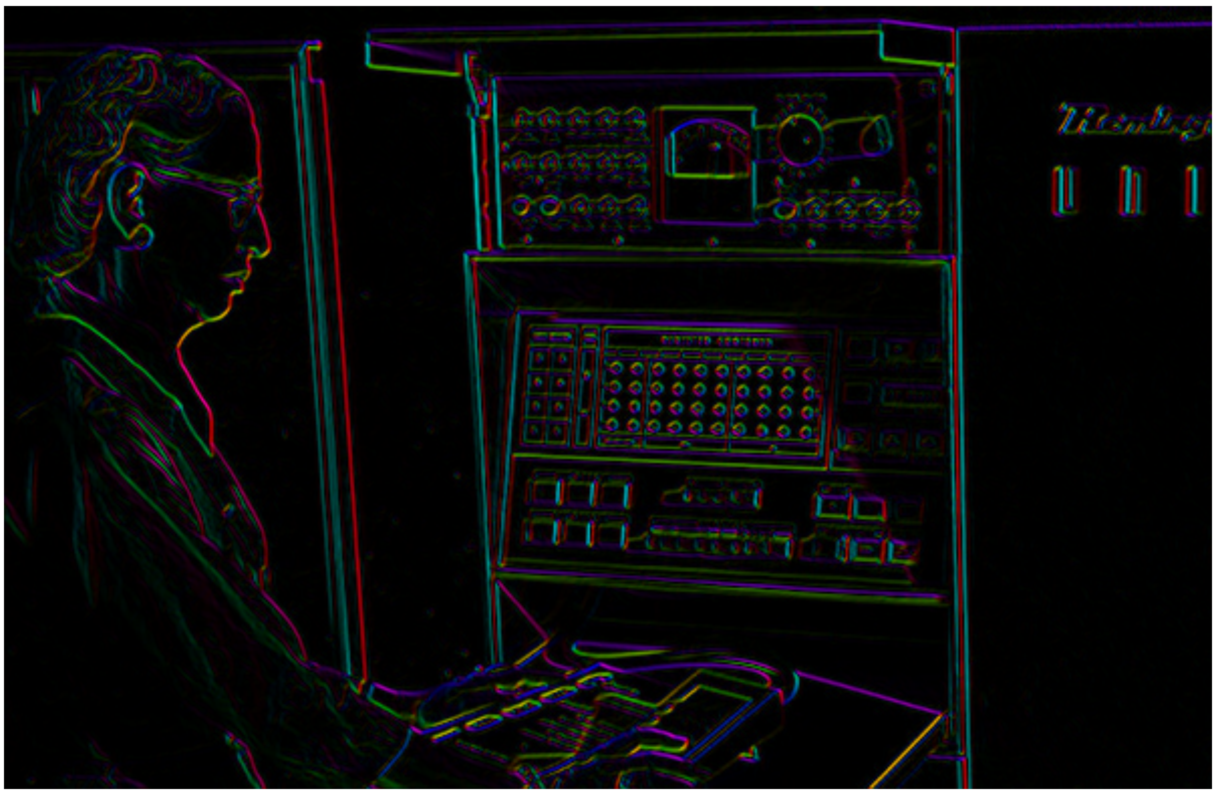


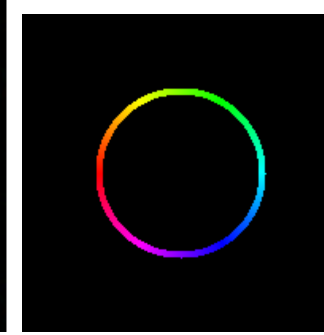
Iy



Last Time: Edges via Image Gradients

Gradient Direction: atan2(Ix, Iy)
Gradient Magnitude: sqrt($Ix^2 + Iy^2$)





I'm making the lightness equal to gradient magnitude

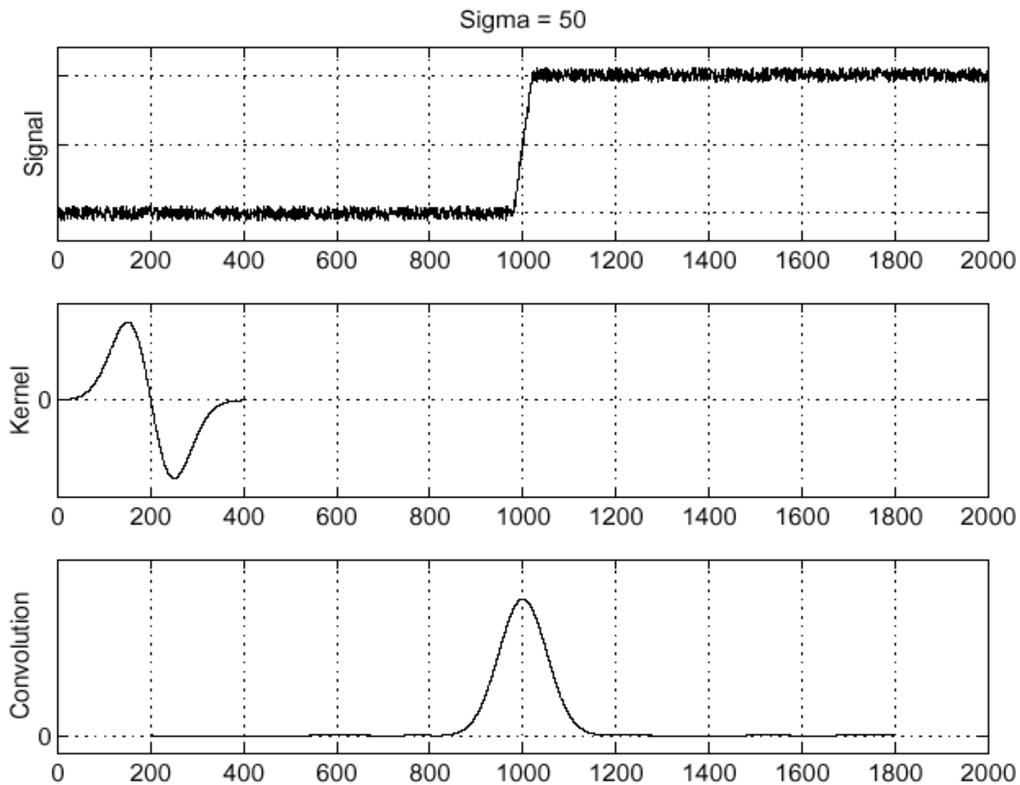
Last Time: Edges via Image Gradients

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

f

$\frac{d}{dx} g$

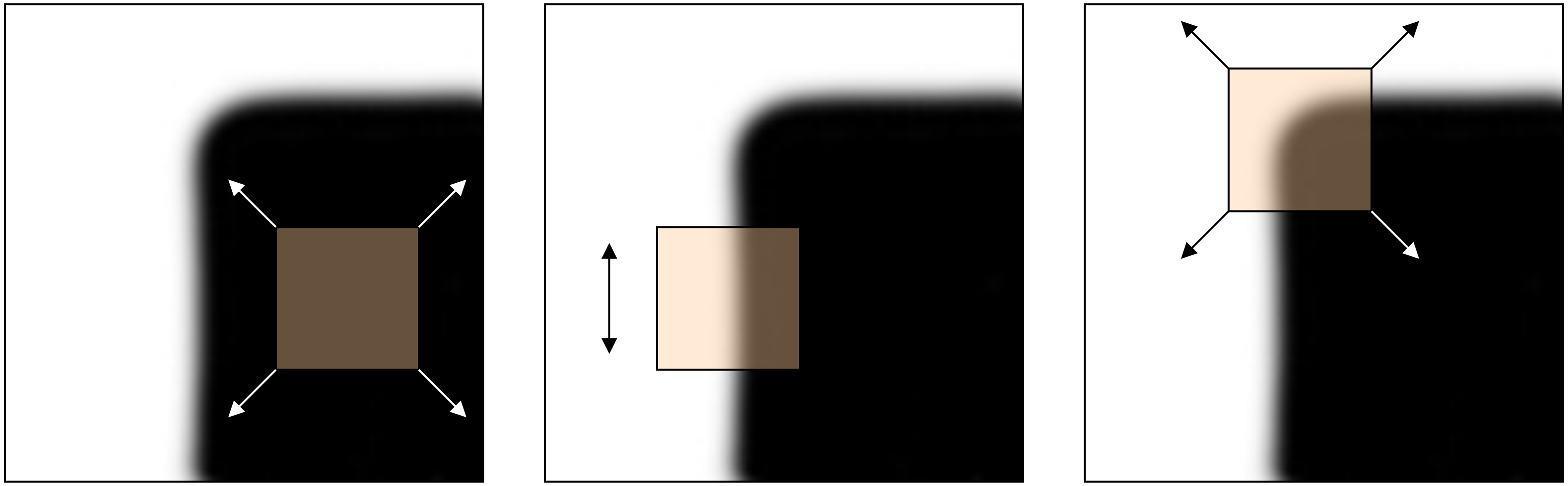
$f * \frac{d}{dx} g$



Slide Credit: S. Seitz

Last Time: Corners

Can localize the location, 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

Diagram credit: S. Lazebnik

Last Time: Detecting Corners

By doing a taylor expansion of the image, the second moment matrix tells us how quickly the image changes and in which directions.

Can compute at each pixel

Directions

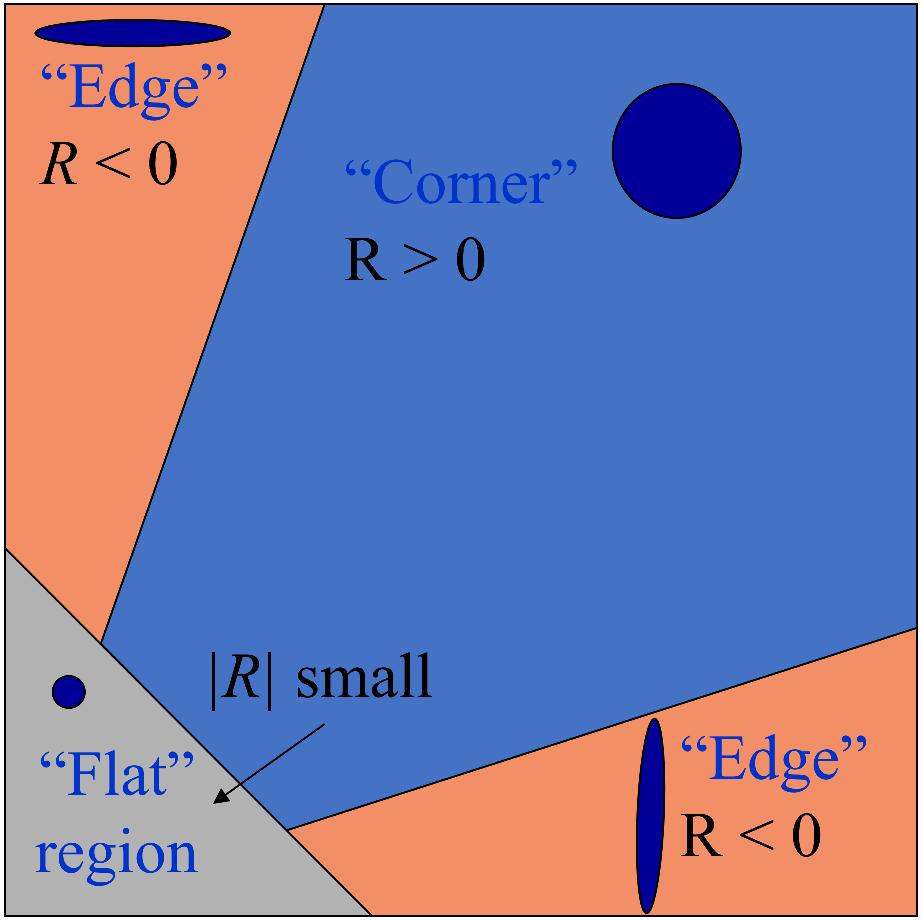
$$\boldsymbol{M} = \overbrace{\begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix}}^{\text{Can compute at each pixel}} = \boldsymbol{R}^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \boldsymbol{R}$$

Amounts

Last Time: Detecting Corners

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

α: constant (0.04 to 0.06)



Slide credit: S. Lazebnik; Note: this refers to visualization 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

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

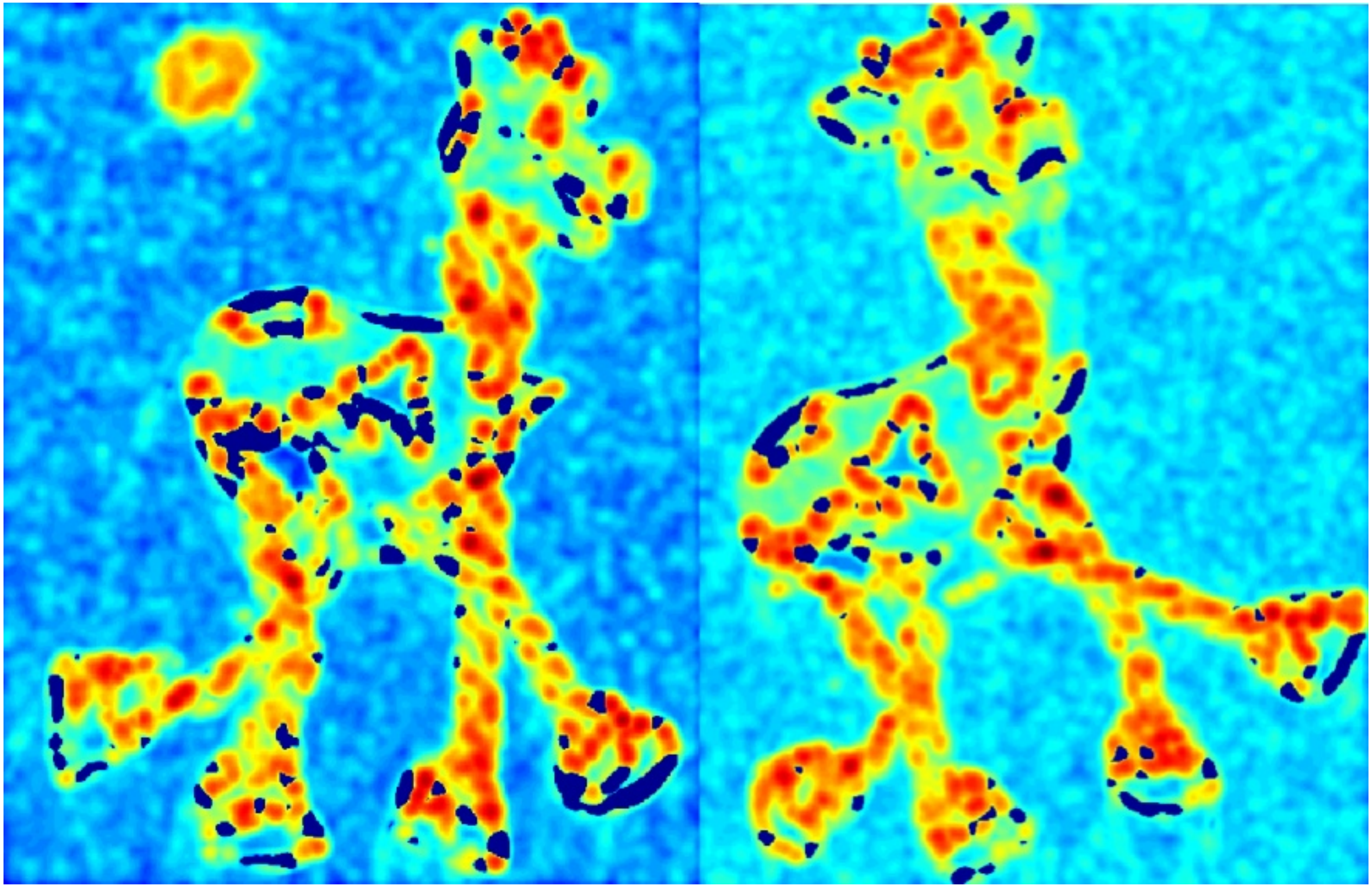
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Slide credit: S. Lazebnik

Computing R

Slide credit: S. Lazebnik

Computing R



Slide credit: S. Lazebnik

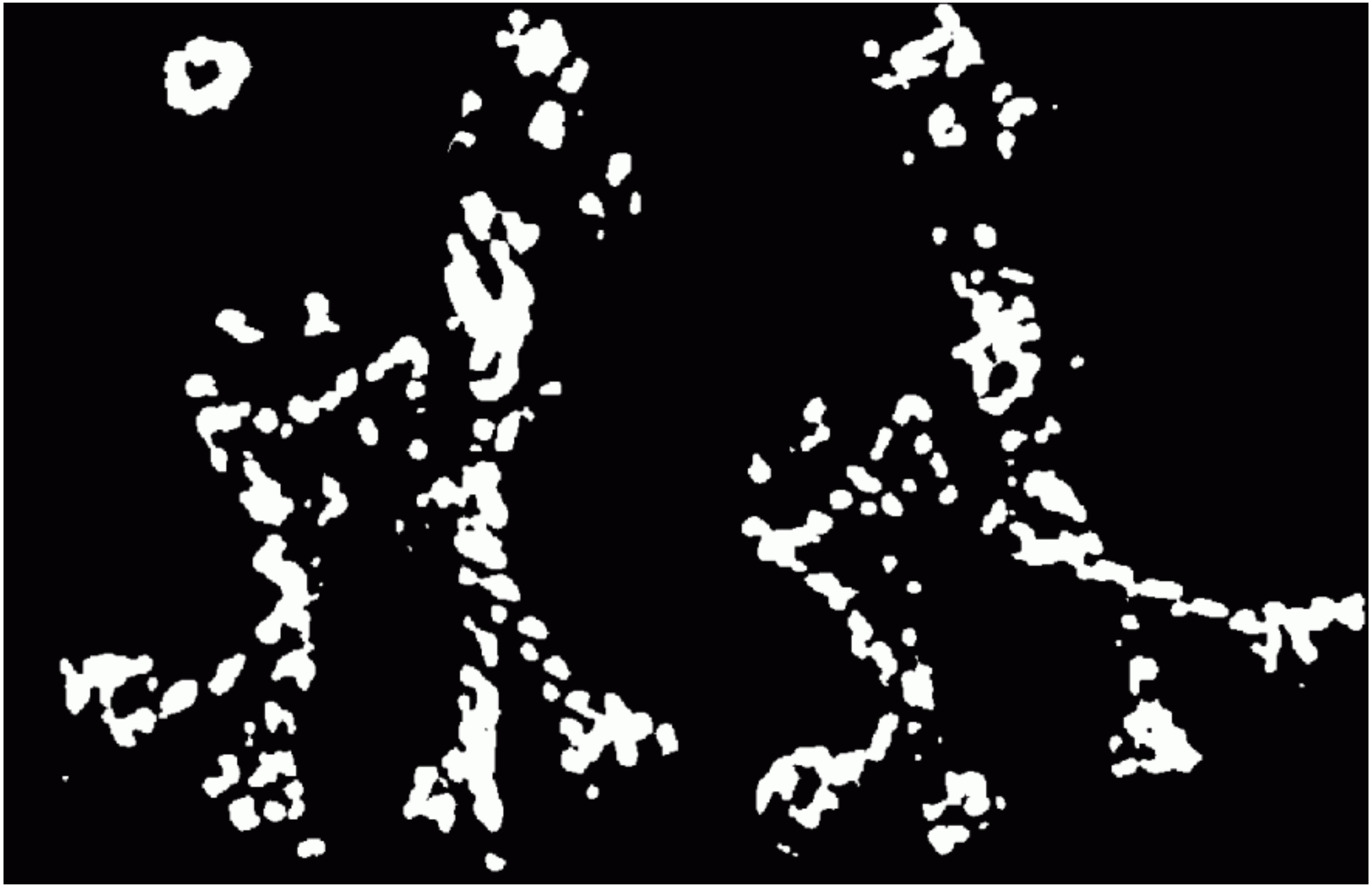
Harris Corner Detector

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

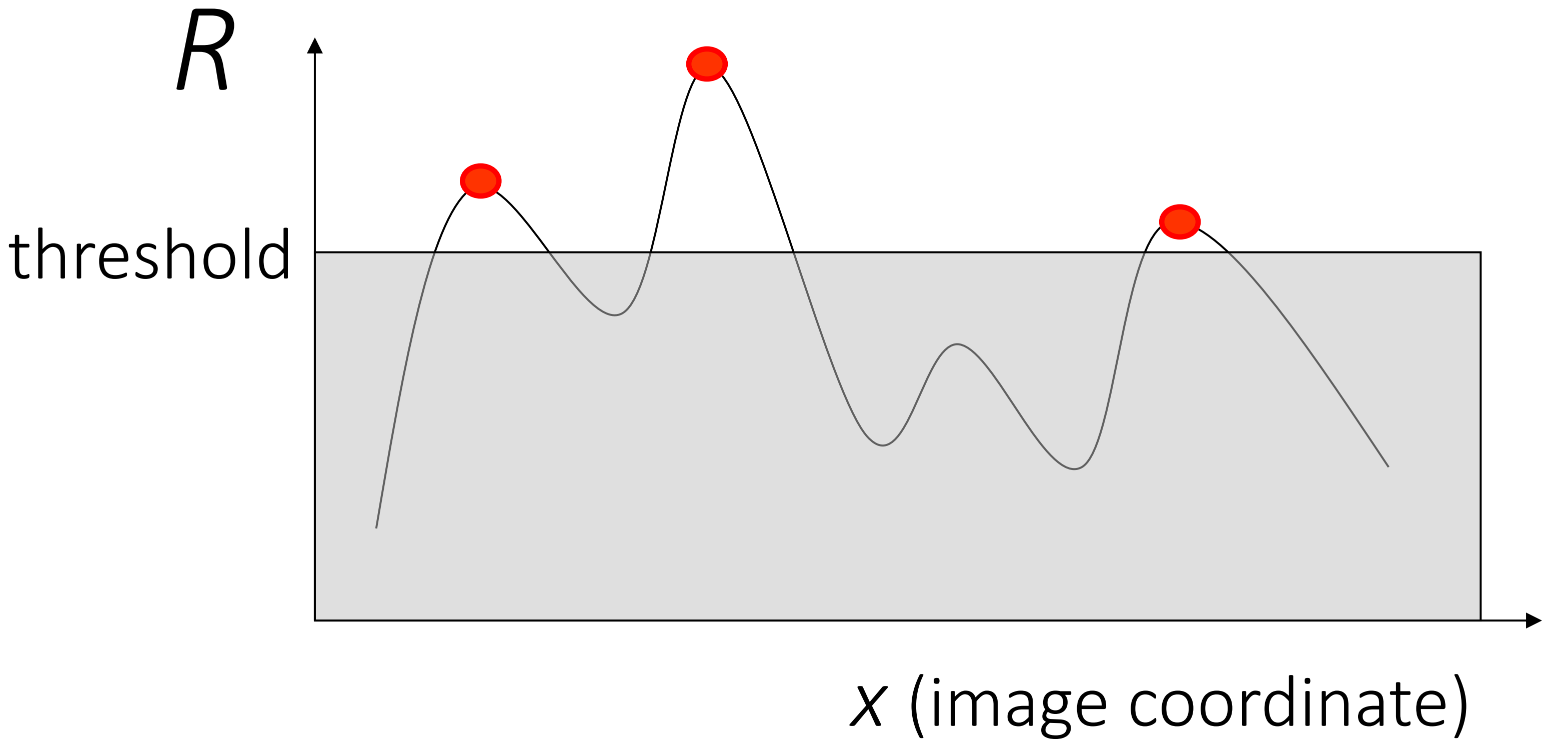
1. Compute partial derivatives Ix, Iy per pixel
2. Compute **M** at each pixel, using Gaussian weighting w
3. Compute response function R
4. Threshold R
5. Take only local maxima (Non-Maxima Suppression, NMS)

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

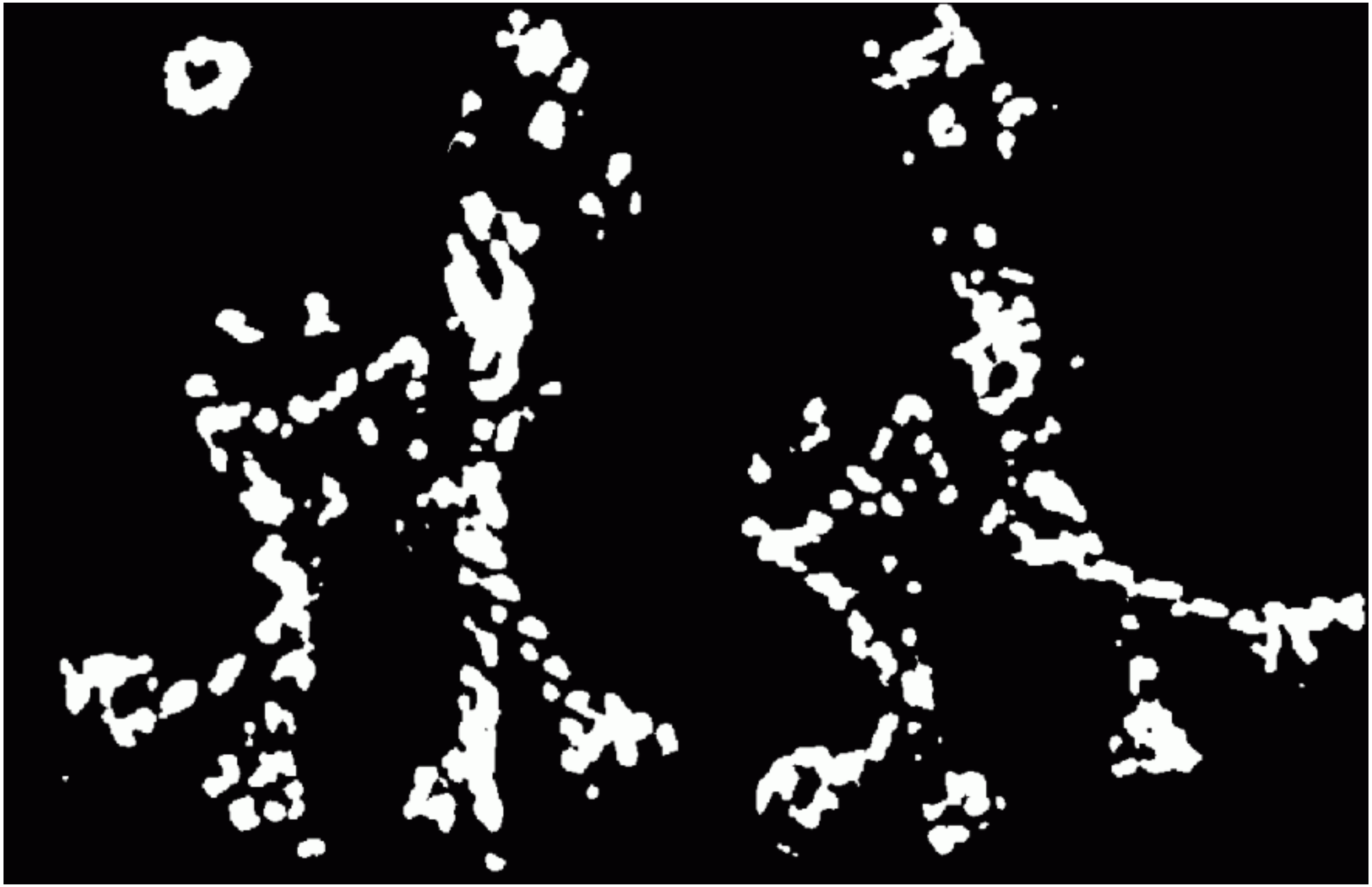
Slide credit: S. Lazebnik

Harris Corner Detector: NMS

Local Maxima are pixels with a higher R value than their neighbors



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: D(T(I)) = D(I)
- **Covariant/equivariant** with some things: image is transformed and corners transform with it: D(T(I)) = T(D(I))

Where I is an image, T is a transformation, and D is our detector

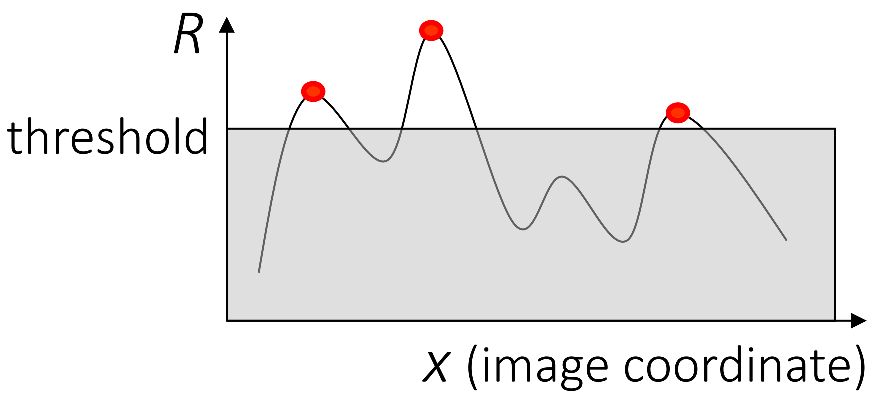
Slide credit: S. Lazebnik

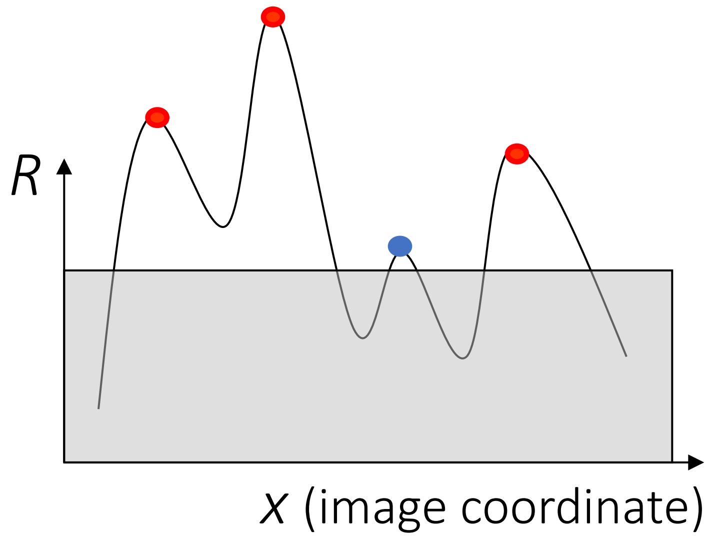
Affine Intensity Change

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

M only depends on derivatives, so b is irrelevant

But a scales derivatives and there's a threshold

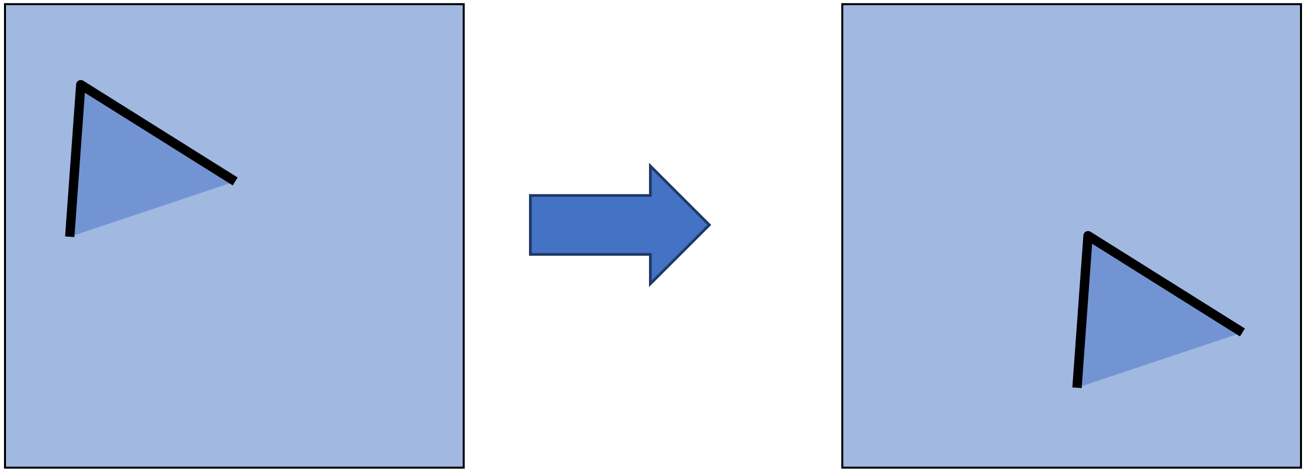




Partially invariant to affine intensity changes

Slide credit: S. Lazebnik

Image Translation

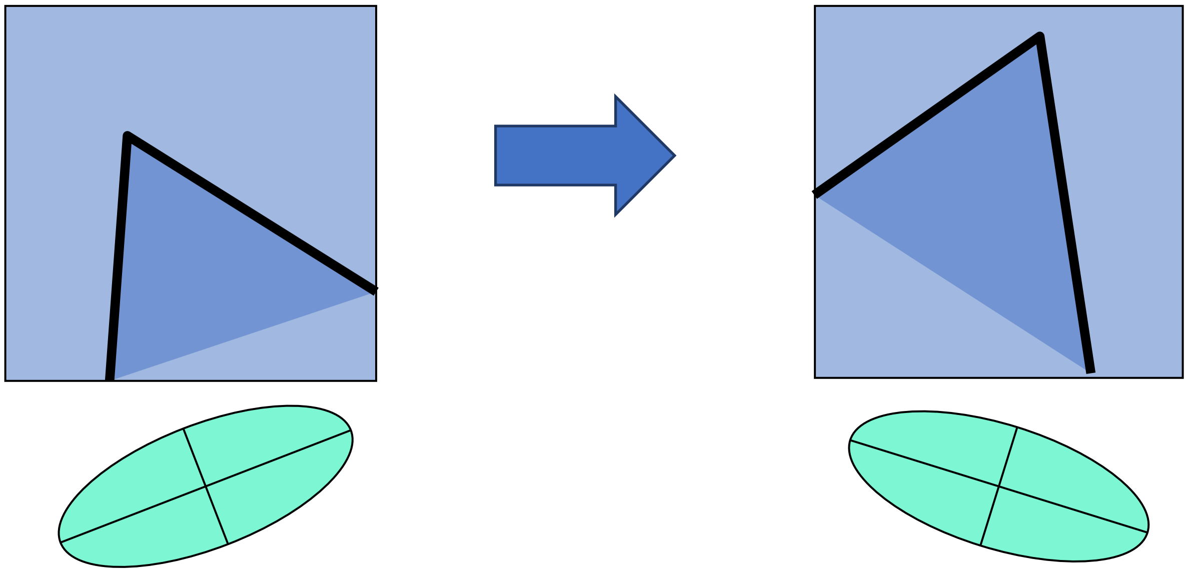


All done with convolution. Convolution is translation invariant.

Equivariant with translation

Slide credit: S. Lazebnik

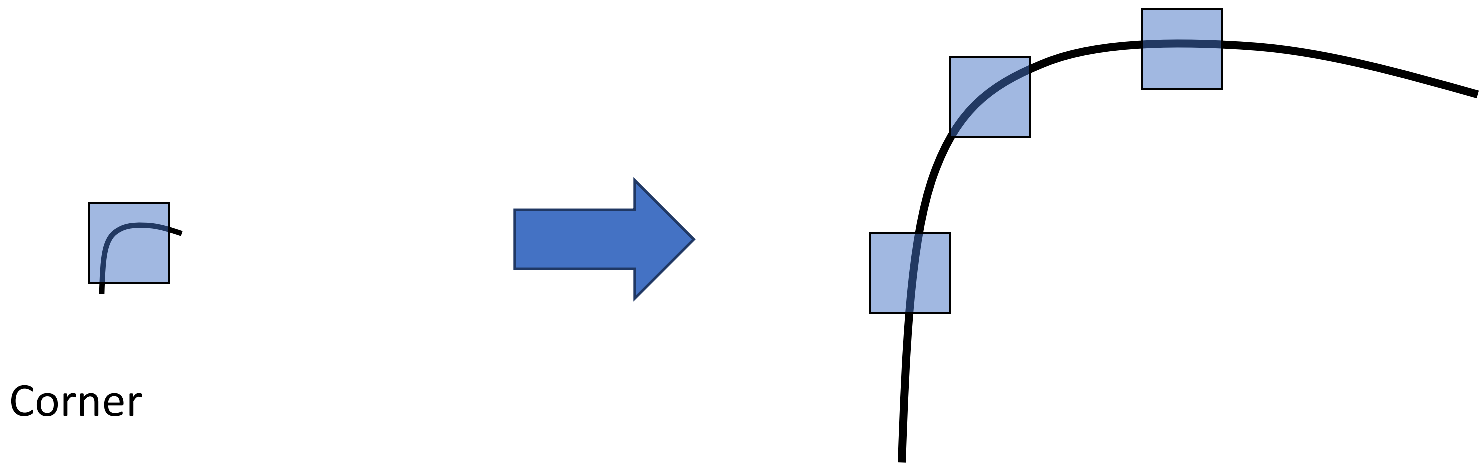
Image Rotation



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

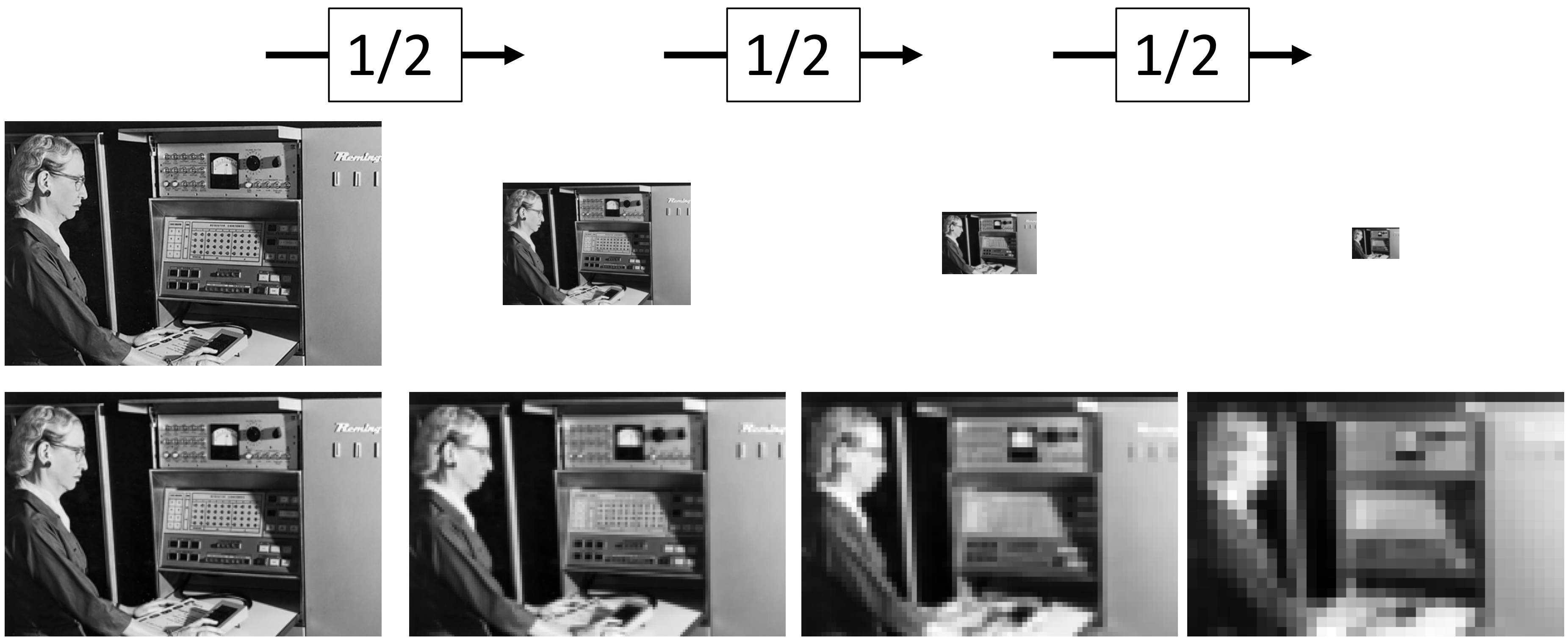
Today

- Fixing scaling by making detectors in both location **and scale**

- Enabling matching between features by **describing regions**

Key Idea: Scale

Left to right: each image is half-sized
Upsampled with big pixels below

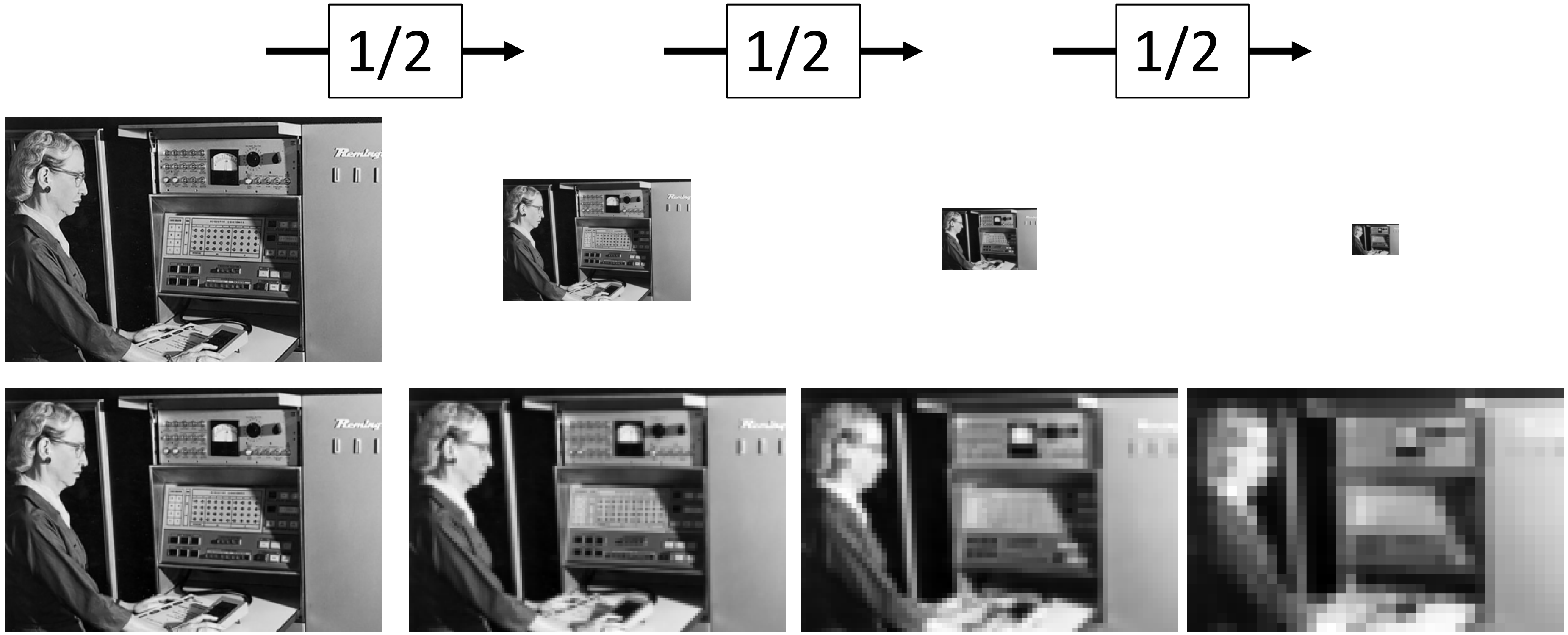


Note: I'm also slightly blurring to prevent aliasing (https://en.wikipedia.org/wiki/Aliasing)

Key Idea: Scale

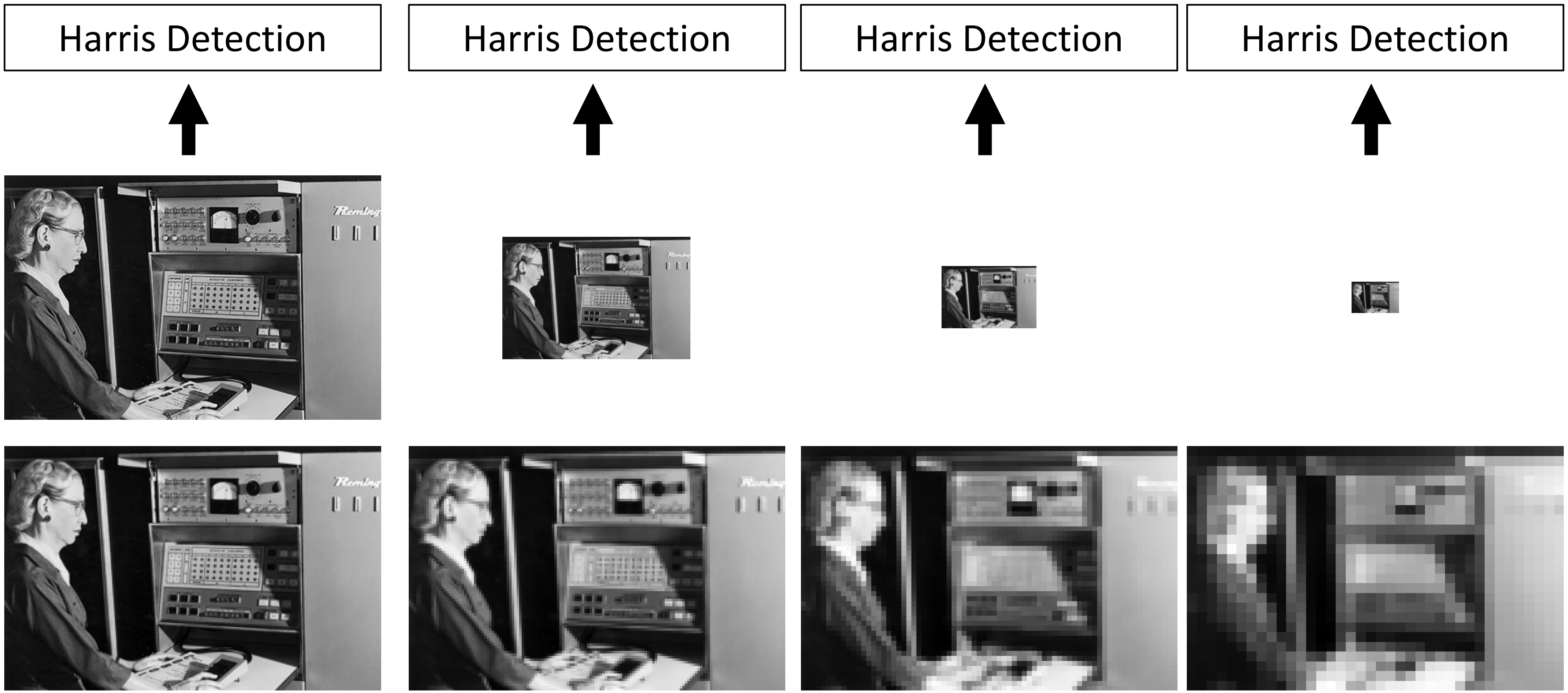
Left to right: each image is half-sized

If I apply a KxK filter, how much of the original image does it see in each image?



Note: I'm also slightly blurring to prevent aliasing (https://en.wikipedia.org/wiki/Aliasing)

Solution to Scales: Try them all!



See: Multi-Image Matching using Multi-Scale Oriented Patches, Brown et al. CVPR 2005

Aside: This is a common trick!

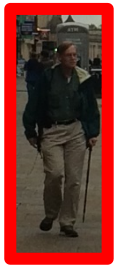
Given a 50x16 person detector, how do I detect:
(a) 250x80 (b) 150x48 (c) 100x32 (d) 25x8 people?

Sample people from image

Aside: This is a common trick!

Detecting all the people
The red box is a fixed size

Sample people from image



Aside: This is a common trick!

Detecting all the people
The red box is a fixed size

Sample people from image

Aside: This is a common trick!

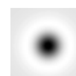
Detecting all the people
The red box is a fixed size

Sample people from image

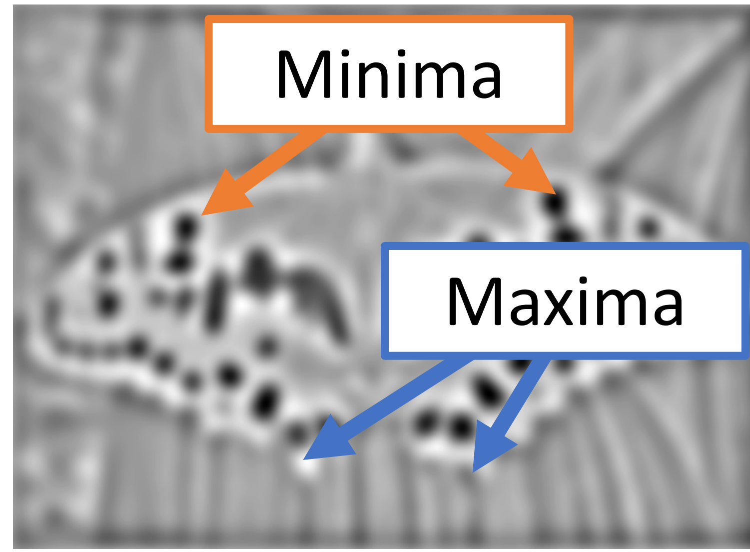
Blob Detection

Another detector (has some nice properties)

*



=

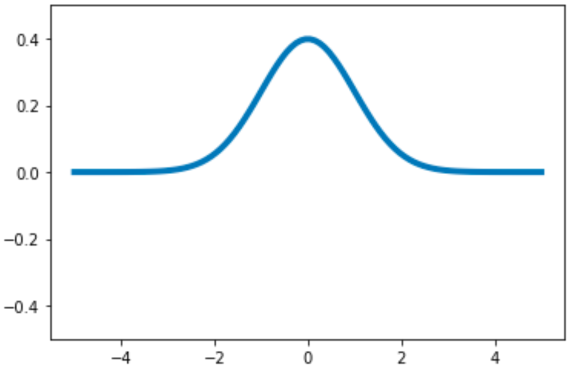


Find maxima ***and minima*** of blob filter response in scale ***and space***

Slide credit: N. Snavely

Gaussian Derivatives (1D)

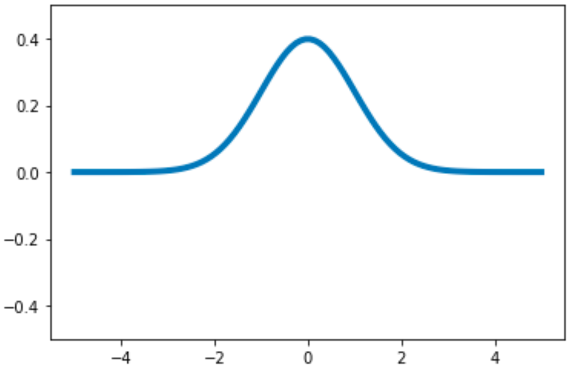
$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$



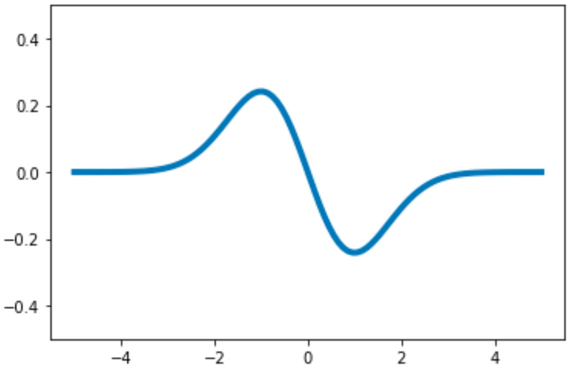
$G_\sigma(x)$

Gaussian Derivatives (1D)

$$\frac{\partial}{\partial x} G_\sigma(x) = \frac{x}{\sigma^3 \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) = -\frac{x}{\sigma^2} G_\sigma(x)$$



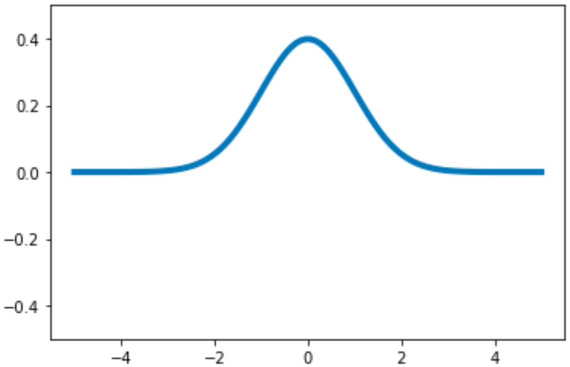
$G_\sigma(x)$



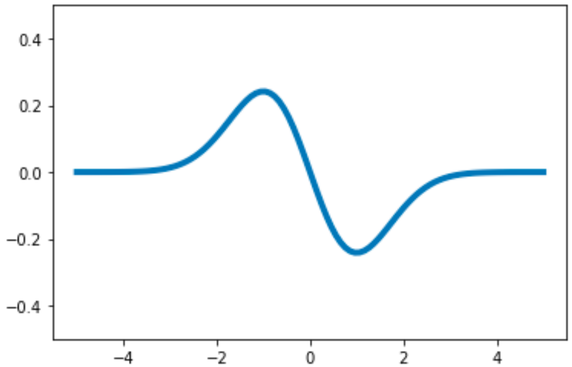
$\frac{\partial}{\partial x} G_\sigma(x)$

Gaussian Derivatives (1D)

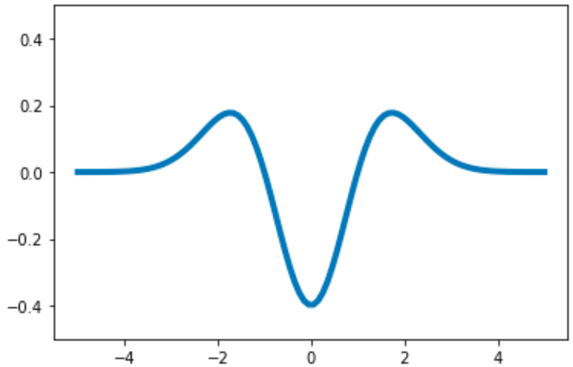
$$\frac{\partial^2}{\partial x^2} G_\sigma(x) = \frac{x^2 - \sigma^2}{\sigma^5 \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) G_\sigma(x)$$



$G_\sigma(x)$



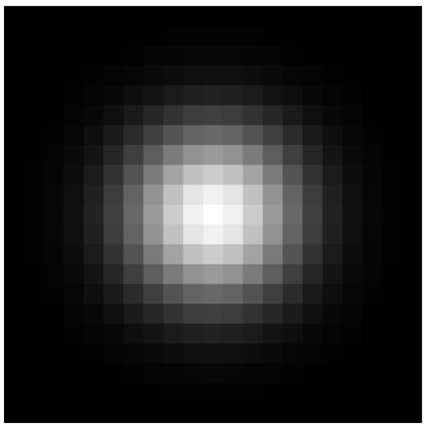
$\frac{\partial}{\partial x} G_\sigma(x)$



$\frac{\partial^2}{\partial x^2} G_\sigma(x)$

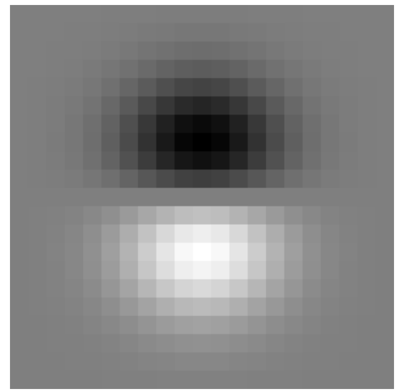
Gaussian Derivatives (2D)

Gaussian

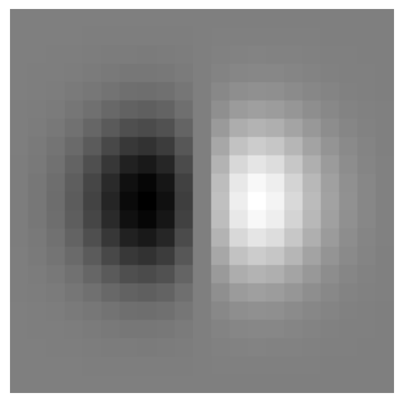


1st Deriv

$$\frac{\partial}{\partial y} g$$

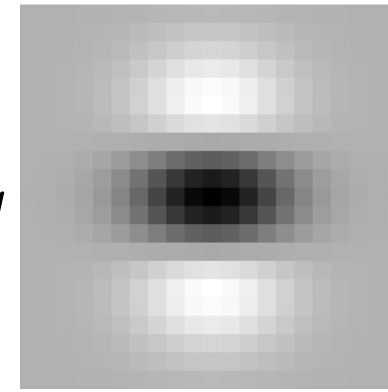


$$\frac{\partial}{\partial x} g$$

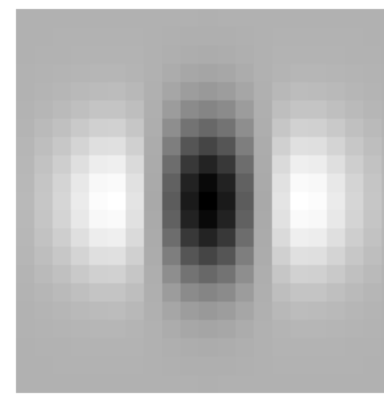


2nd Deriv

$$\frac{\partial^2}{\partial^2 y} g$$



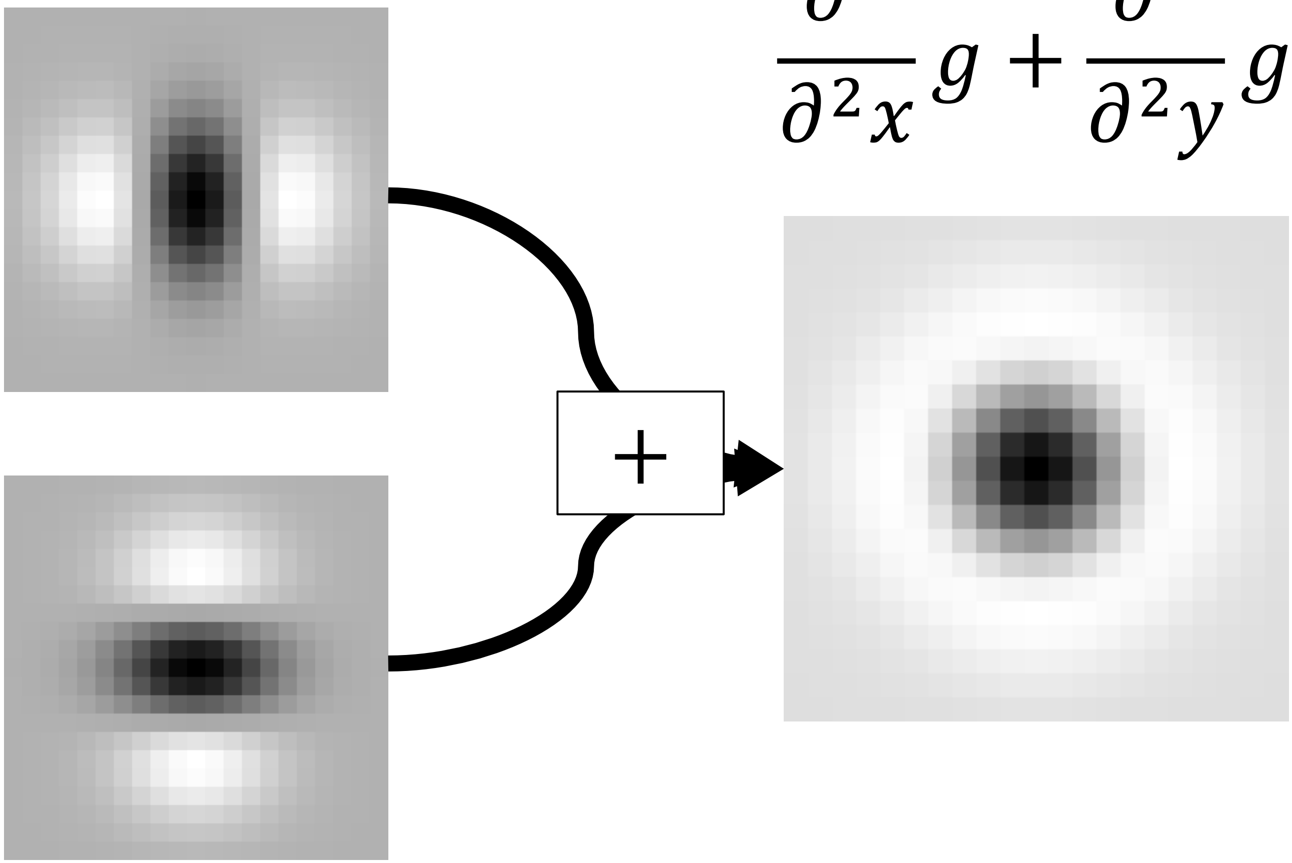
$$\frac{\partial^2}{\partial^2 x} g$$



Laplace of Gaussian (2D)

$$\frac{\partial^2}{\partial^2 y} g$$

$$\frac{\partial^2}{\partial^2 x} g$$



$$\frac{\partial^2}{\partial^2 x} g + \frac{\partial^2}{\partial^2 y} g$$

Slight detail: for technical reasons, you need to scale the Laplacian.

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

Edge Detection with Laplacian

f

Edge

$\frac{\partial^2}{\partial^2 x} g$

Laplacian Of Gaussian

$f * \frac{\partial^2}{\partial^2 x} g$

Edge = Zero-crossing

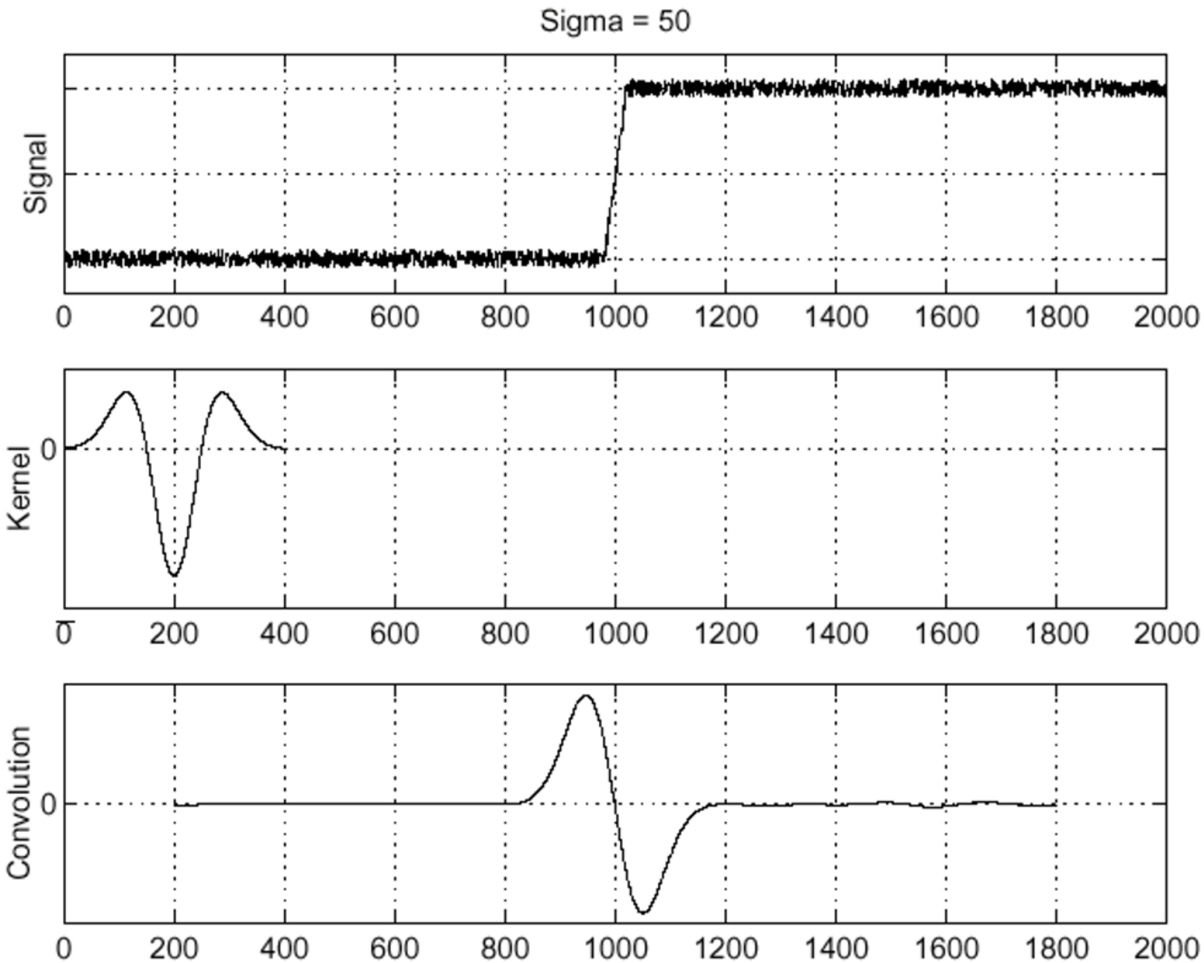


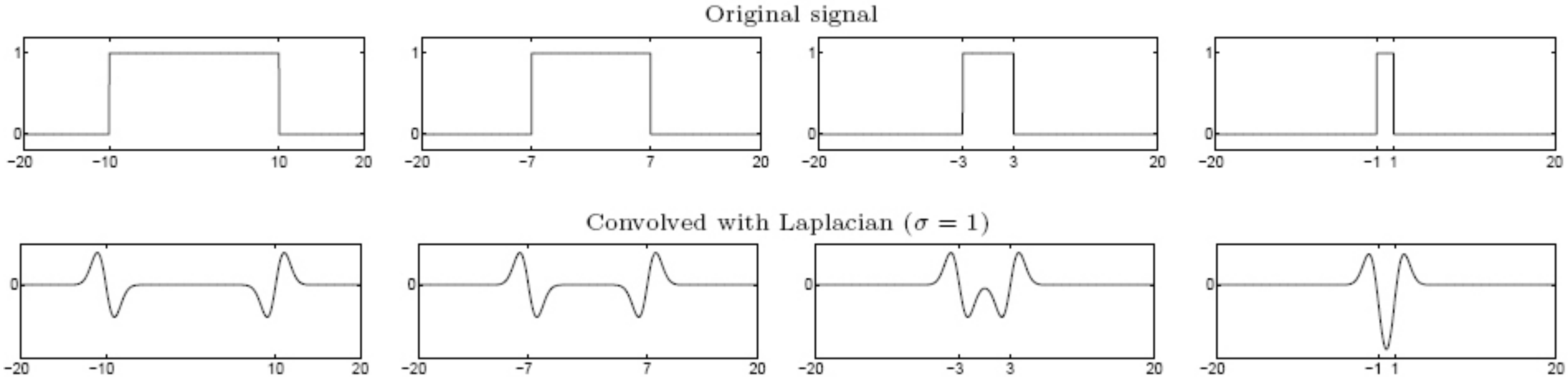
Figure credit: S. Seitz

Blob Detection with Laplacian

Edge: zero-crossing

Blob = Two edges in opposite directions

When blob is just the right size,
Laplacian gives a large absolute value



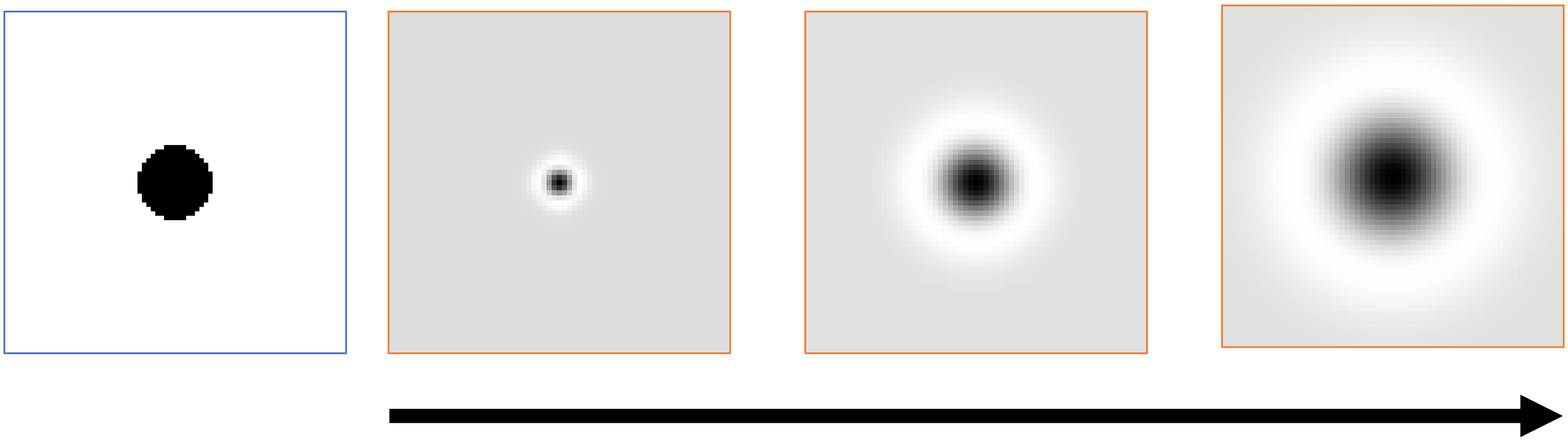
maximum

Figure credit: S. Lazebnik

Scale Selection with Laplacian

Given binary circle and Laplacian filter of scale σ, we can compute the response as a function of the scale.

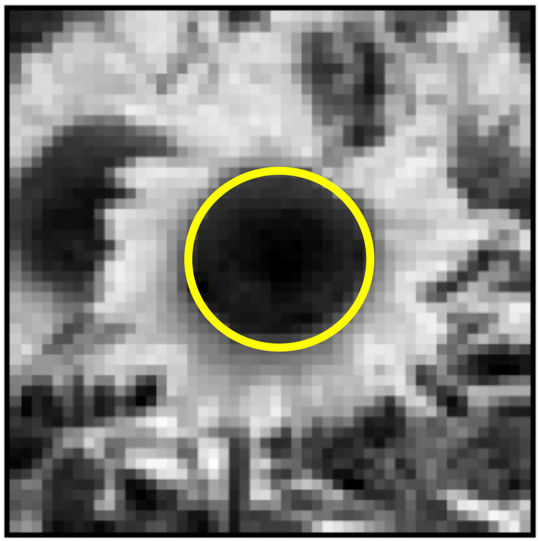
Image	$\sigma = 2$	$\sigma = 6$	$\sigma = 10$
Radius: 8	R: 0.02	R: 2.9	R: 1.8



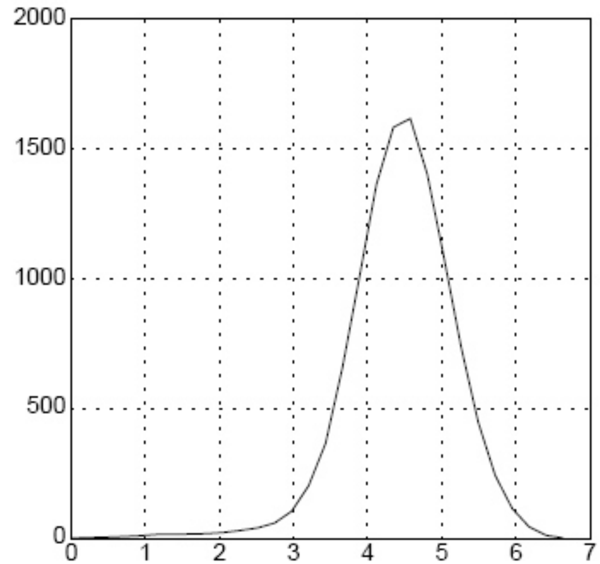
Characteristic Scale

Characteristic scale of a blob is the scale that produces the maximum response

Image



Abs. Response



Slide credit: S. Lazebnik. For more, see: T. Lindeberg (1998). ["Feature detection with automatic scale selection."]() *International Journal of Computer Vision* **30** (2): pp 77--116.

Blob Detection in Scale Space

1. Convolve image with scale-normalized Laplacian at several scales

Slide credit: S. Lazebnik

Blob Detection in Scale Space

Slide credit: S. Lazebnik

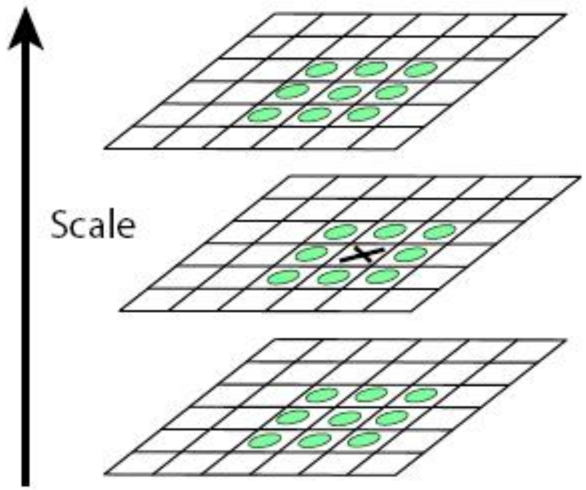
Blob Detection in Scale Space

sigma = 11.9912

Slide credit: S. Lazebnik

Blob Detection in Scale Space

1. Convolve image with scale-normalized Laplacian at several scales
2. Find local maxima and minima of squared Laplacian response in image+scale space



Slide credit: S. Lazebnik

Image and Scale Space

Image
Radius: 8

$\sigma = 2$
R: 0.02

$\sigma = 6$
R: 2.9

$\sigma = 10$
R: 1.8

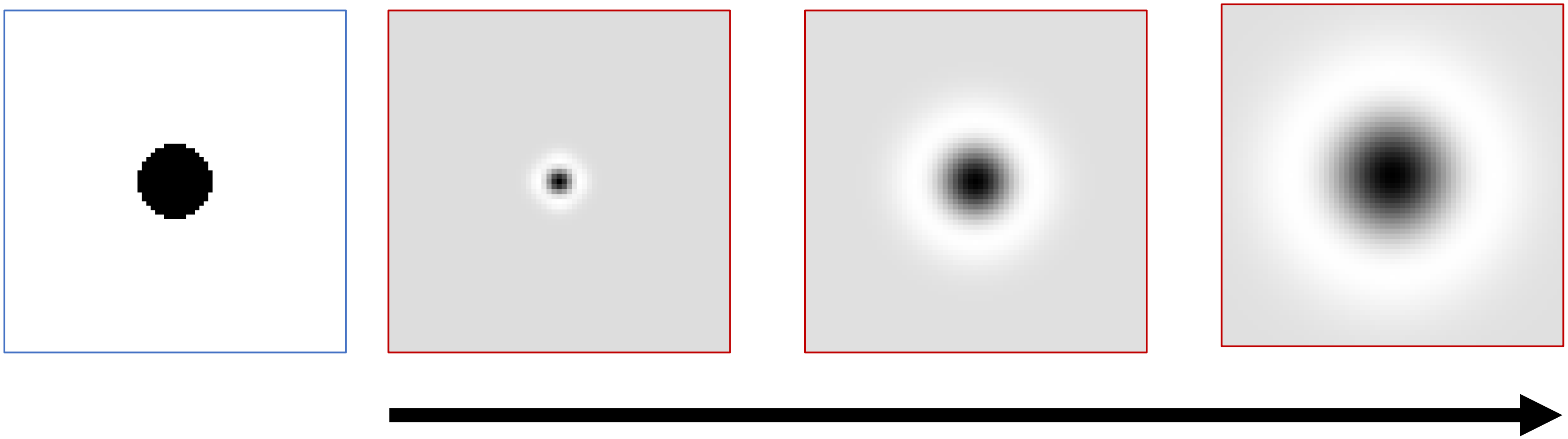
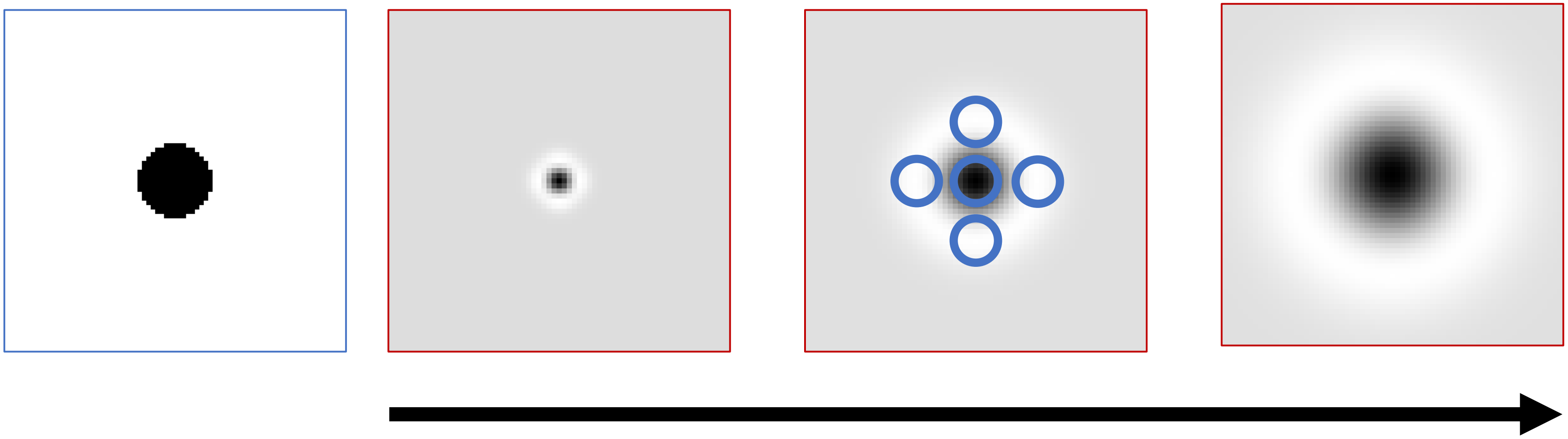


Image-Space Neighbors

Blue points are image-space neighbors

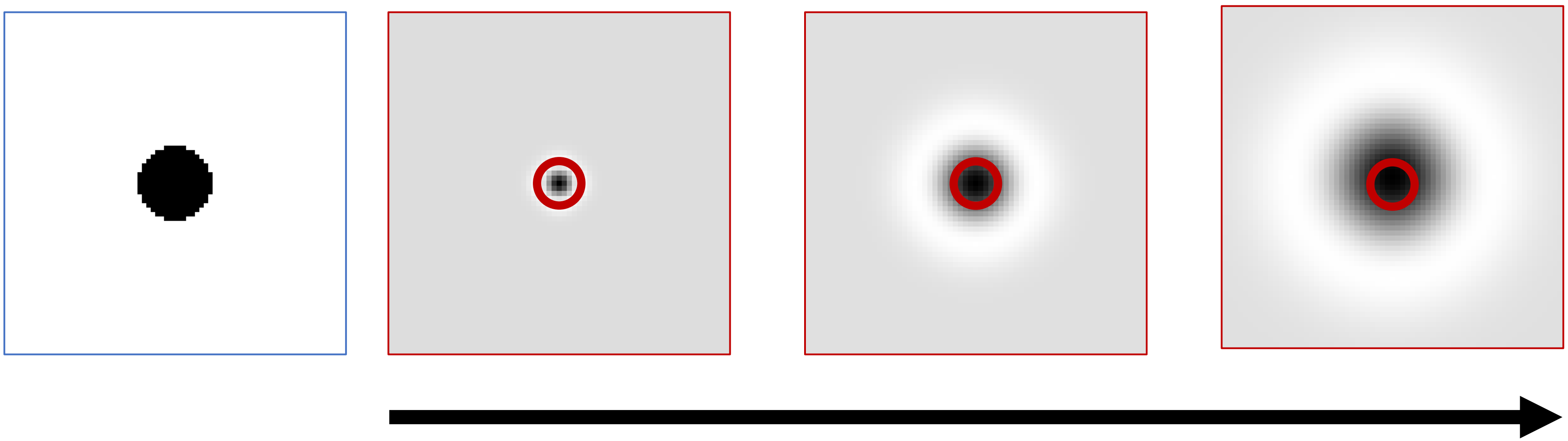
Image	$\sigma = 2$	$\sigma = 6$	$\sigma = 10$
Radius: 8	R: 0.02	R: 2.9	R: 1.8



Scale Space Neighbors

Red points are neighbors in scale space

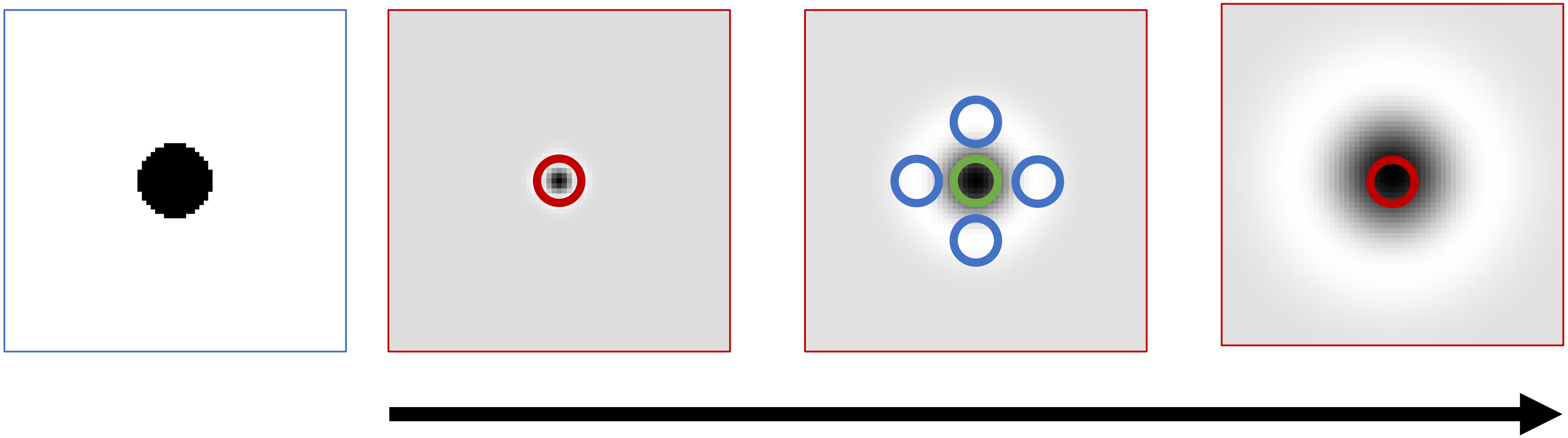
Image	$\sigma = 2$	$\sigma = 6$	$\sigma = 10$
Radius: 8	R: 0.02	R: 2.9	R: 1.8



Local Maxima in Scale and Image

Green point is a local maxima in both image and scale space if:
it is larger than its **image-space neighbors (blue)**
and larger than its **scale-space neighbors (red)**

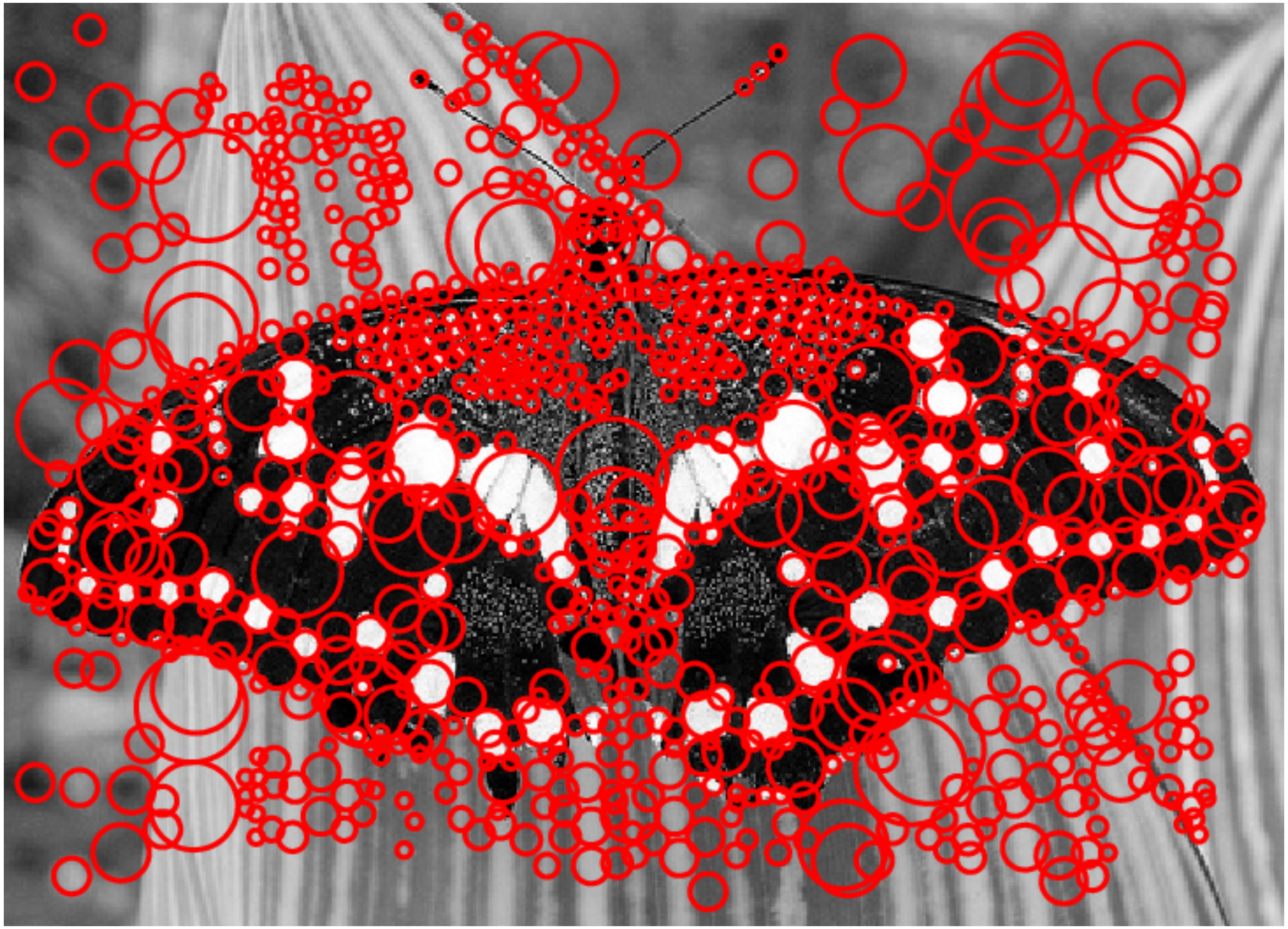
Image	$\sigma = 2$	$\sigma = 6$	$\sigma = 10$
Radius: 8	R: 0.02	R: 2.9	R: 1.8



Blob Detection in Scale Space

Slide credit: S. Lazebnik

Blob Detection in Scale Space



Slide credit: S. Lazebnik

Efficient Implementation

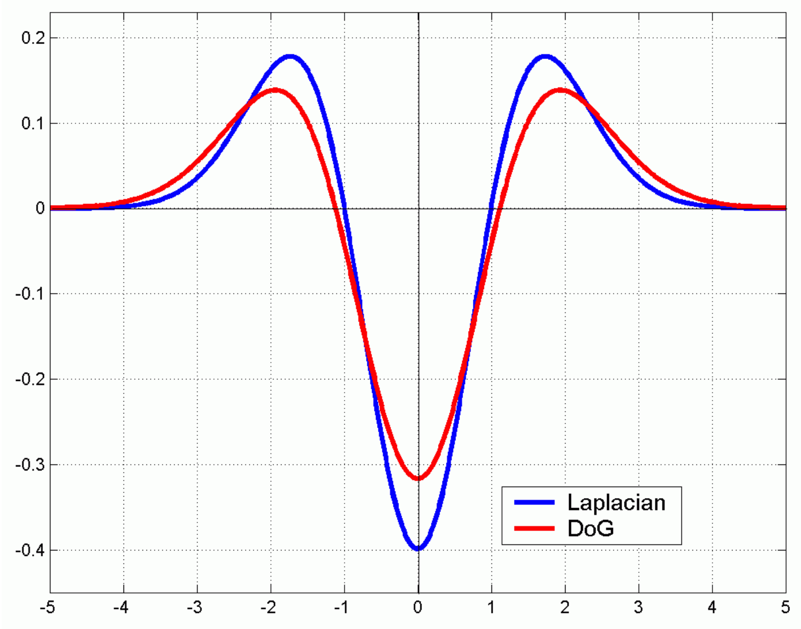
- Approximating the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



Gaussian is separable, so cheaper to compute!

Slide credit: S. Lazebnik

Efficient Implementation

Gaussian, sigma=2

Downsample by 2x

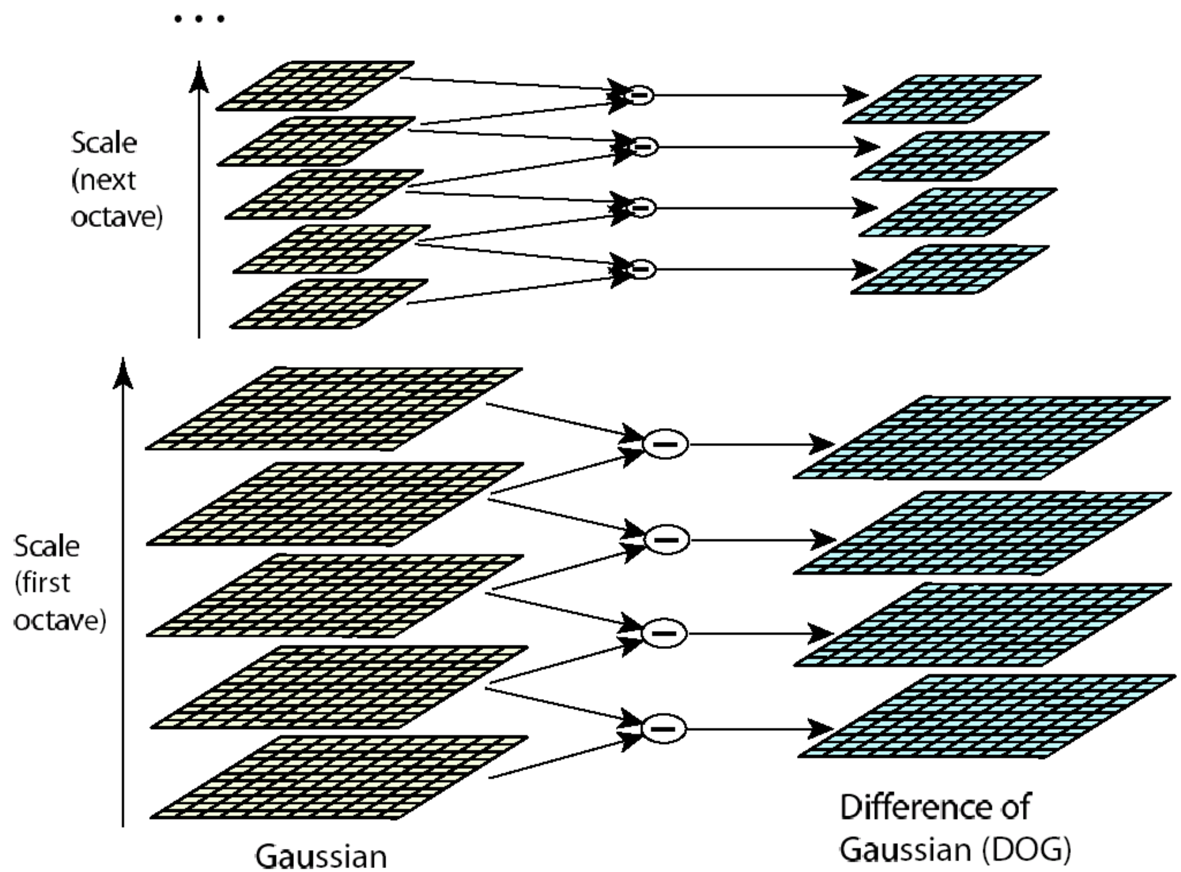
Original Image

Downsample by 2x

Gaussian, sigma=1

Save computation by downsampling before blurring

Efficient Implementation



David G. Lowe. **["Distinctive image features from scale-invariant keypoints."]** *IJCV* 60 (2), pp. 91-110, 2004.

Slide credit: S. Lazebnik

Recall Two Problems for Today:

- Fixing scaling by making detectors in both location **and scale**
- Enabling matching between features by **describing regions**

Problem 1 Solved!

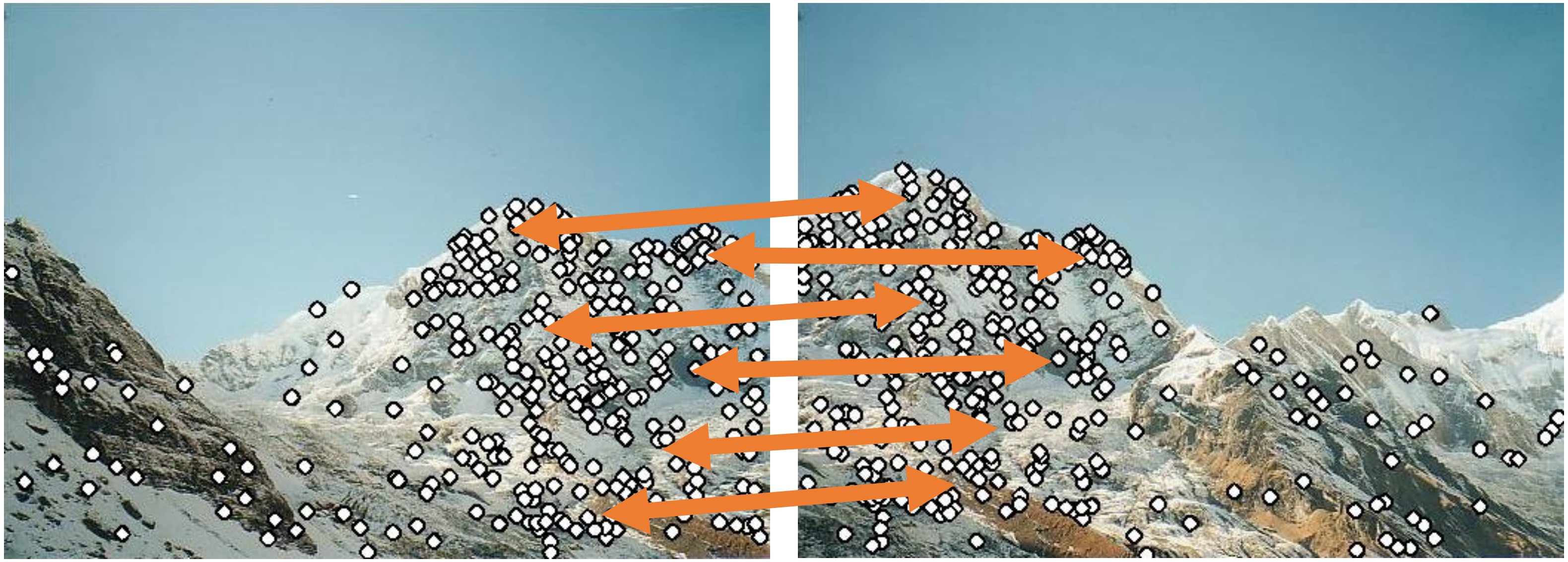
- How do we deal with scales: try them all
- **Why is this efficient?**

Vast majority of effort is in the first and second scales

$$1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{4^i} \dots = \frac{4}{3}$$

Second Problem: Describing Features

Finding and Matching

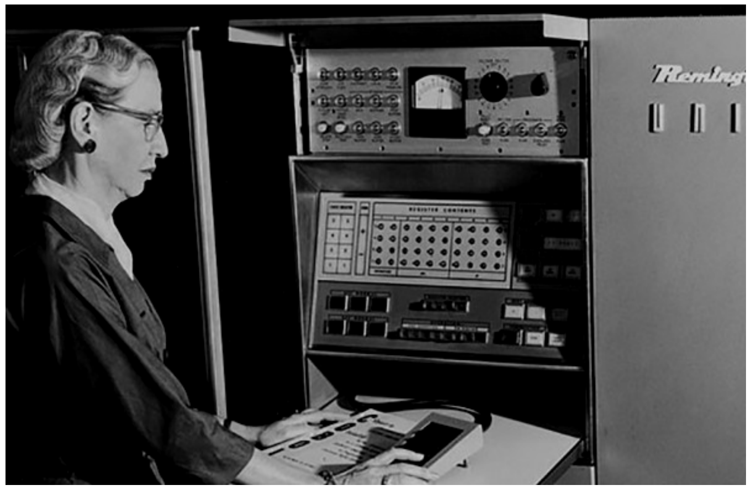


1: find corners+features

2: match based on local image data

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

Second Problem: Describing Features

	Image – 40	1/2 size, rot. 45° Lightened+40
Image		
100x100 crop at Glasses		

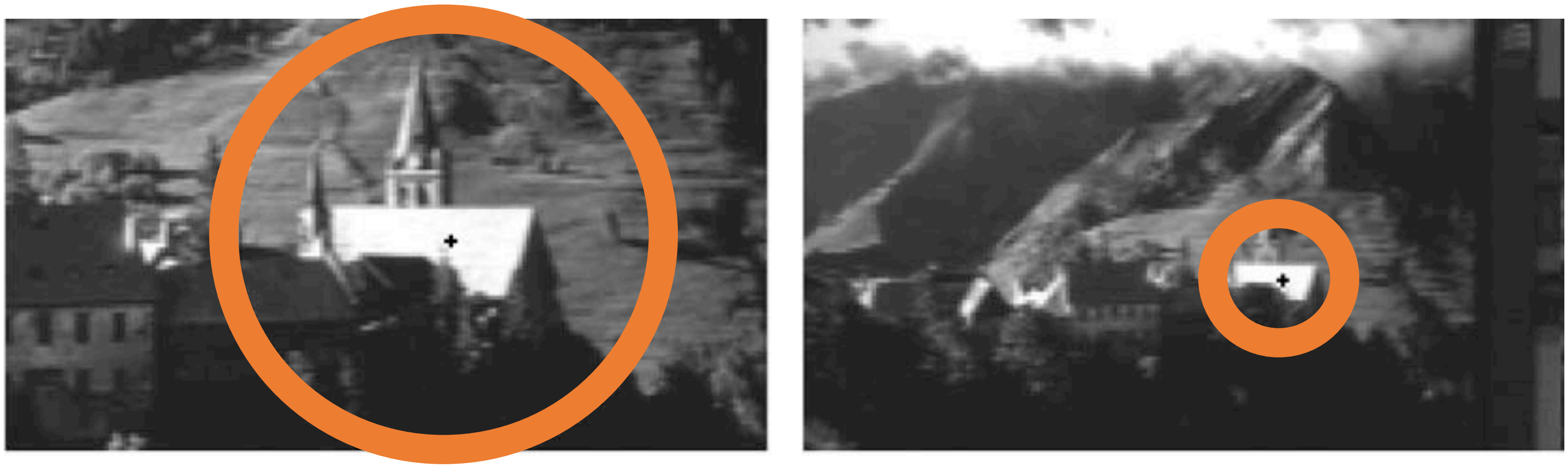
Second Problem: Describing Features

Once we've found a corner/blobs, we can't just use the image nearby. What about:

1. Scale?
2. Rotation?
3. Additive light?

Handling Scale

Given characteristic scale (maximum Laplacian response), we can just rescale image

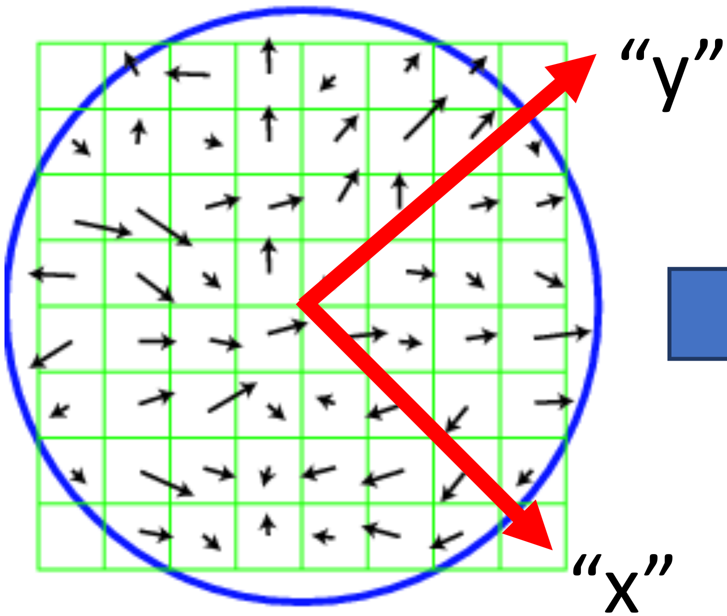


Slide credit: S. Lazebnik

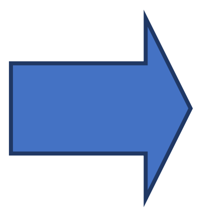
Handling Rotation

Given window, can compute dominant orientation and then rotate image

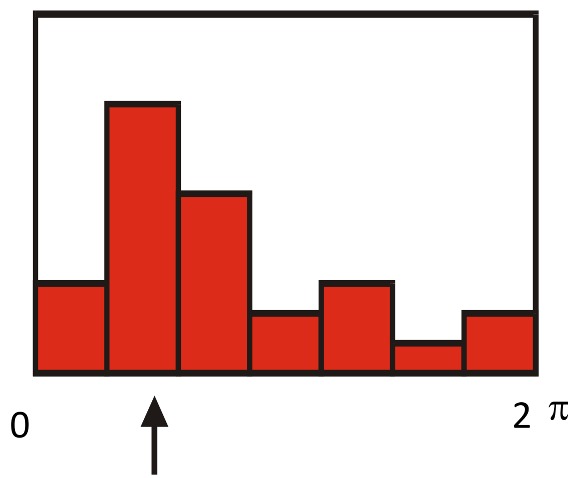
Compute gradient direction at each pixel



Rotate so dominant direction is up



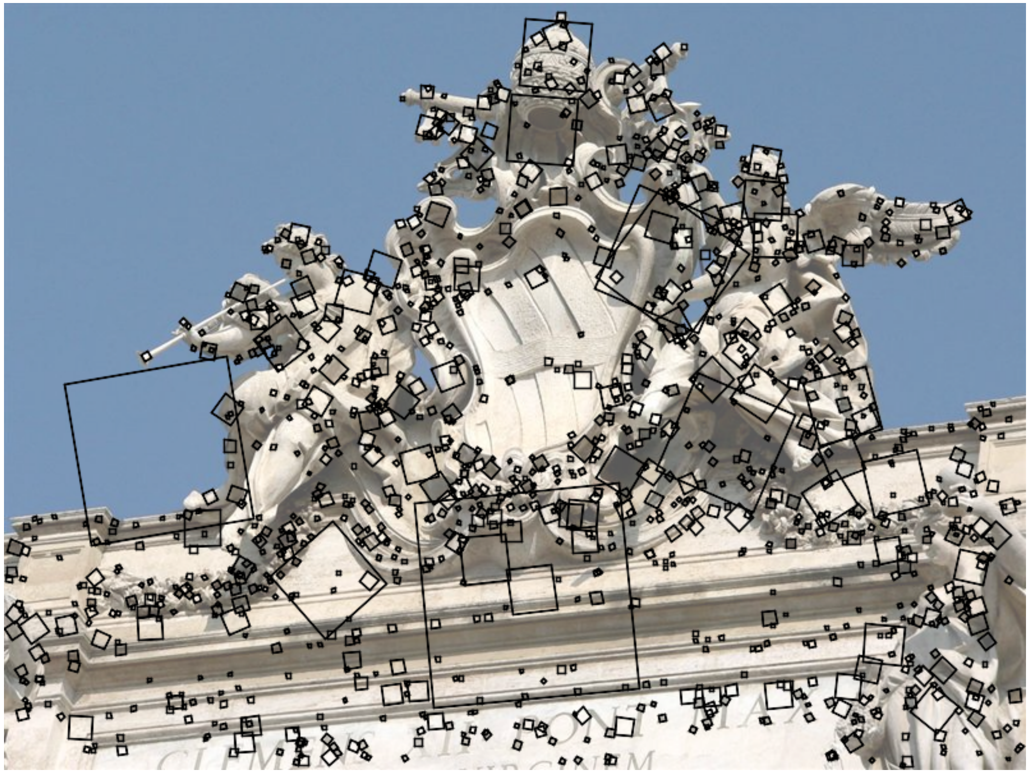
Build a histogram of gradient directions in window



Find dominant direction from histogram

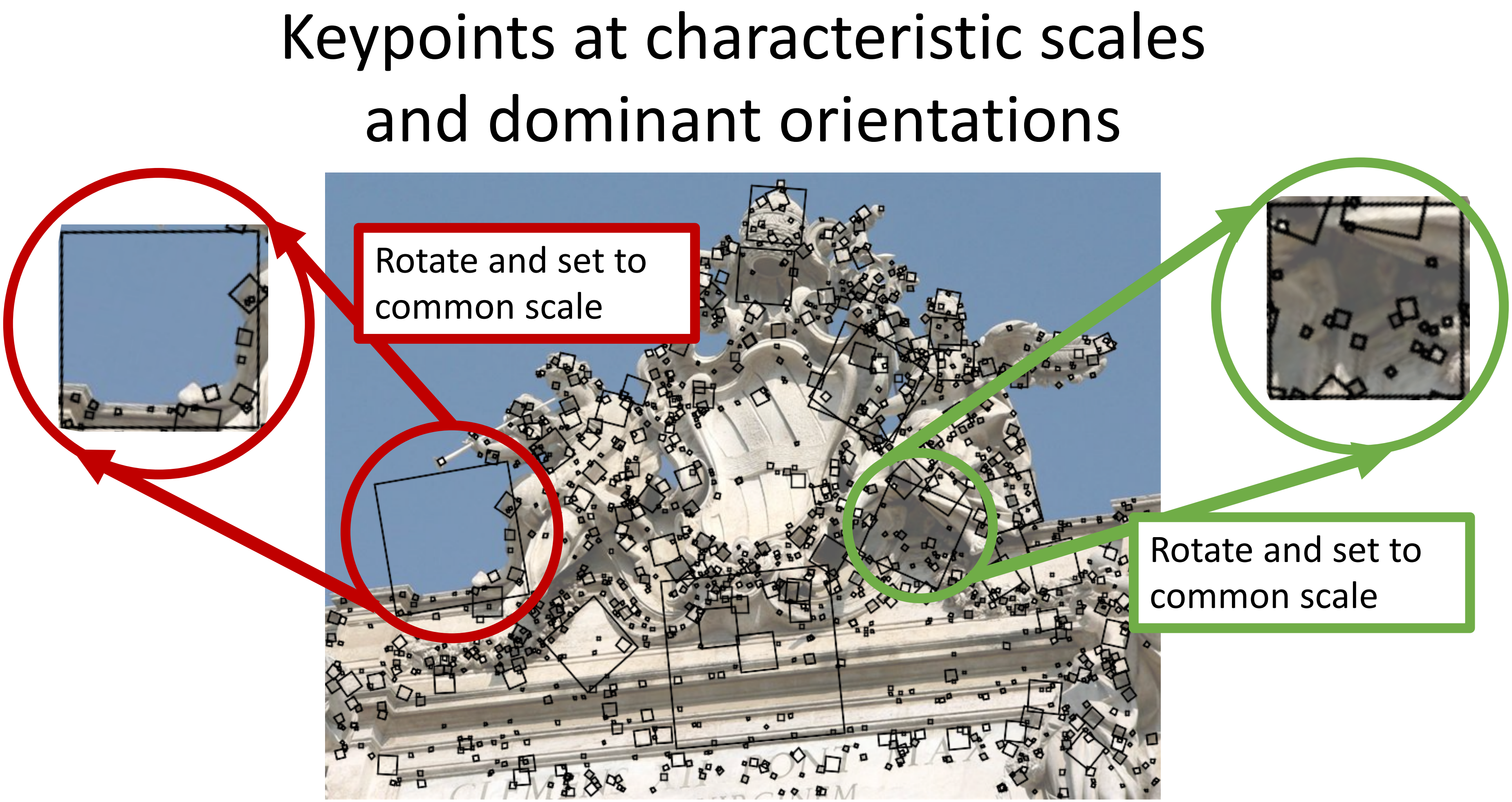
Scale and Rotation

Keypoints at characteristic scales and dominant orientations



Picture credit: S. Lazebnik. Paper: David G. Lowe. ["Distinctive image features from scale-invariant keypoints."](#) *IJCV* 60 (2), pp. 91-110, 2004.

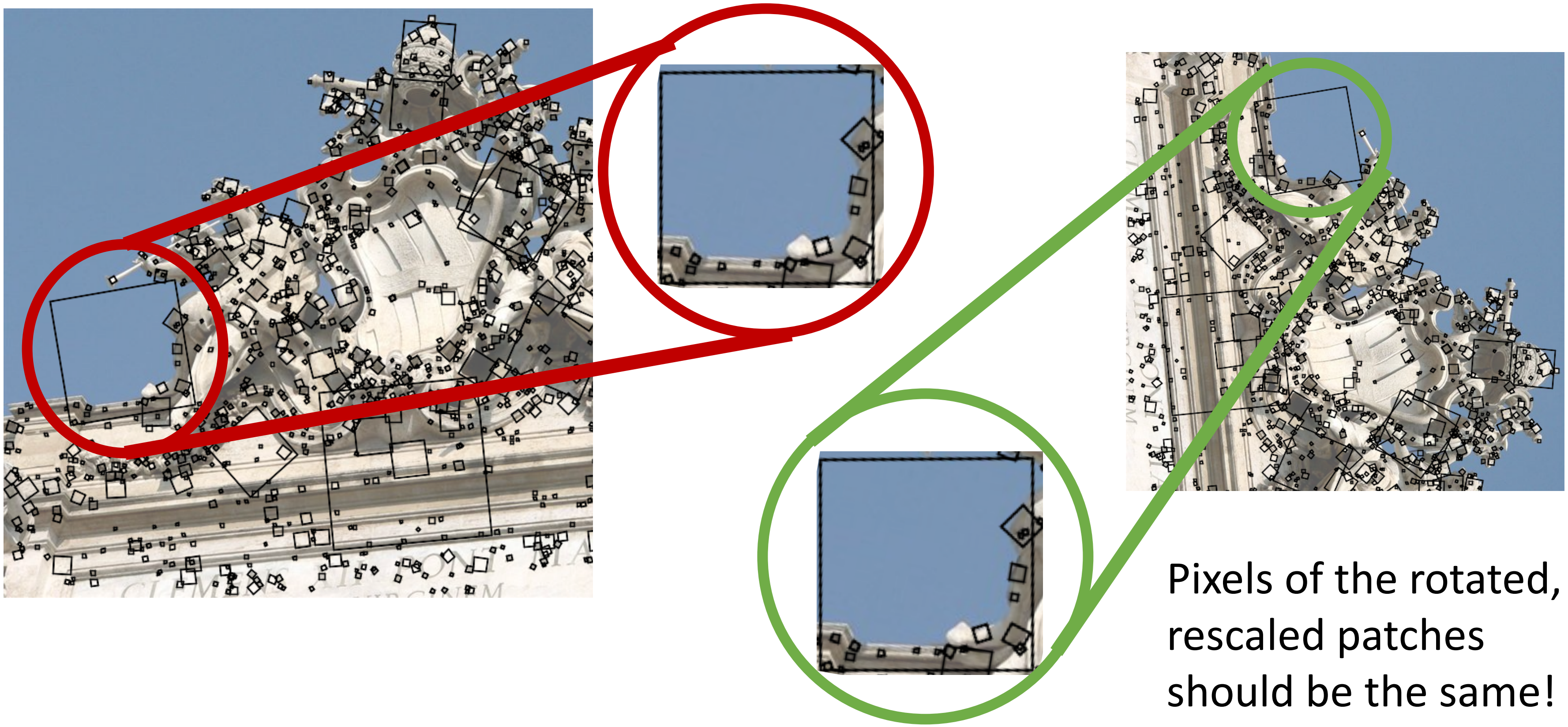
Scale and Rotation



Picture credit: S. Lazebnik. Paper: David G. Lowe. ["Distinctive image features from scale-invariant keypoints."](#)
IJCV 60 (2), pp. 91-110, 2004.

Scale and Rotation

Now if we had two images of the same scene but different scale / rotation, we would find the same keypoints and set them to a common scale / rotation

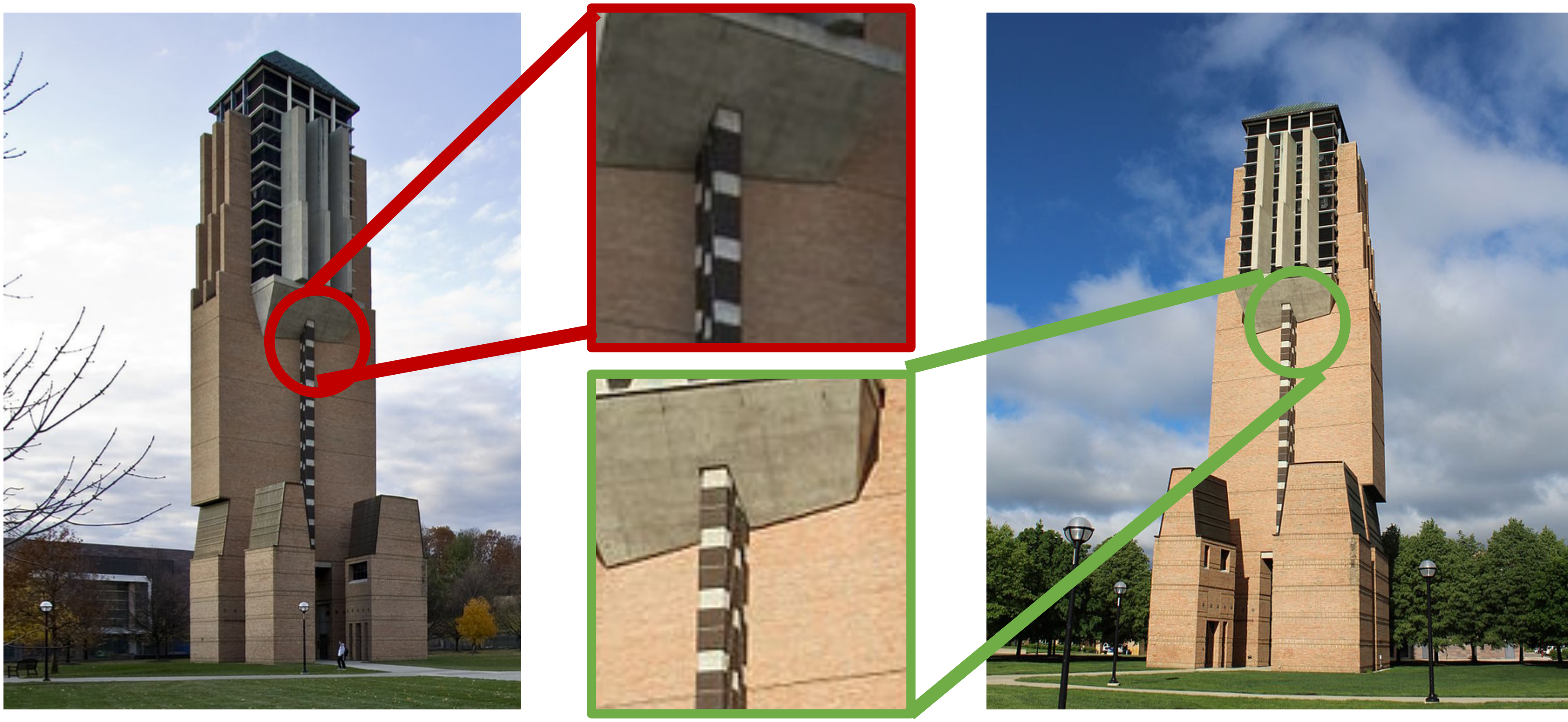


Pixels of the rotated, rescaled patches should be the same!

Illumination, Out-of-plane rotation?

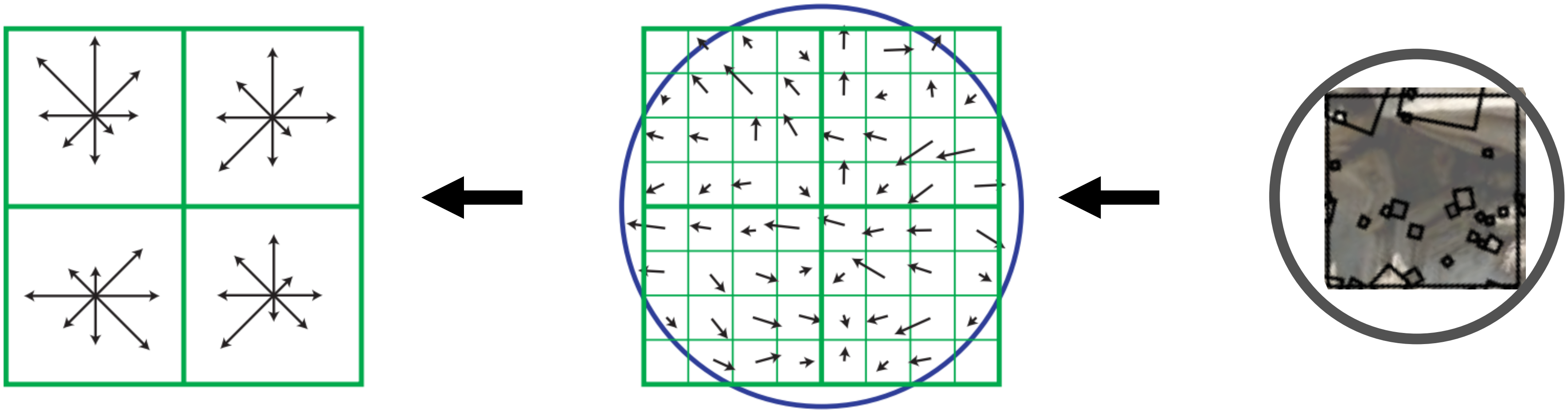
We would like to be able to match these two points

But the local patches look very different!



Idea: Instead of comparing the pixels, instead use a **feature vector** to describe the appearance of each patch

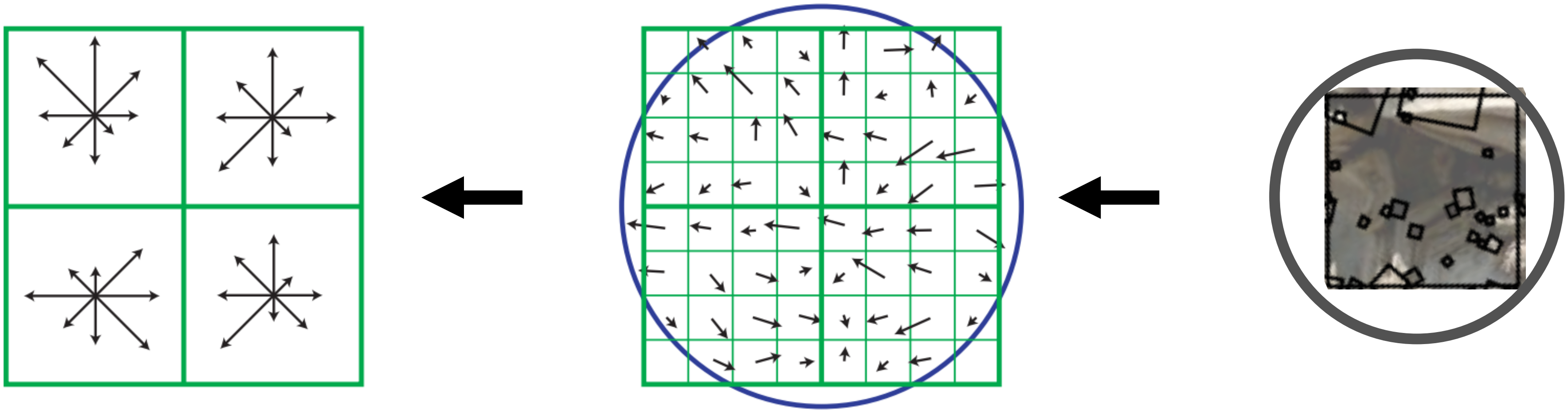
SIFT Descriptors



1. Normalize the rotation / scale of the patch
2. Compute gradient at each pixel
3. Divide into sub-patches (here 2x2, actually 4x4)
4. In each sub-patch, compute histogram of 8 gradient directions
5. Describe the patch with 4*4*8 = 128 numbers

Figure from David G. Lowe. ["Distinctive image features from scale-invariant keypoints."]() *IJCV* 60 (2), pp. 91-110, 2004.

SIFT Descriptors



Nice properties of SIFT:

1. Using gradients gives invariance to illumination
2. Using histograms of patches gives invariance to small shifts / rotations
3. Compactly describe local appearance of patches with 128-dim vector

Figure from David G. Lowe. ["Distinctive image features from scale-invariant keypoints."]() *IJCV* 60 (2), pp. 91-110, 2004.

SIFT Descriptors

- In principle: build a histogram of the gradients
- In reality: quite complicated
  - Gaussian weighting: smooth response
  - Normalization: reduces illumination effects
  - Clamping
  - Affine adaptation

Read the paper for all the gory details…

Figure from David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Properties of SIFT

- Can handle: up to ~60 degree out-of-plane rotation, Changes of illumination
- Fast and efficient and lots of code available

Slide credit: N. Snavely

Feature Descriptors

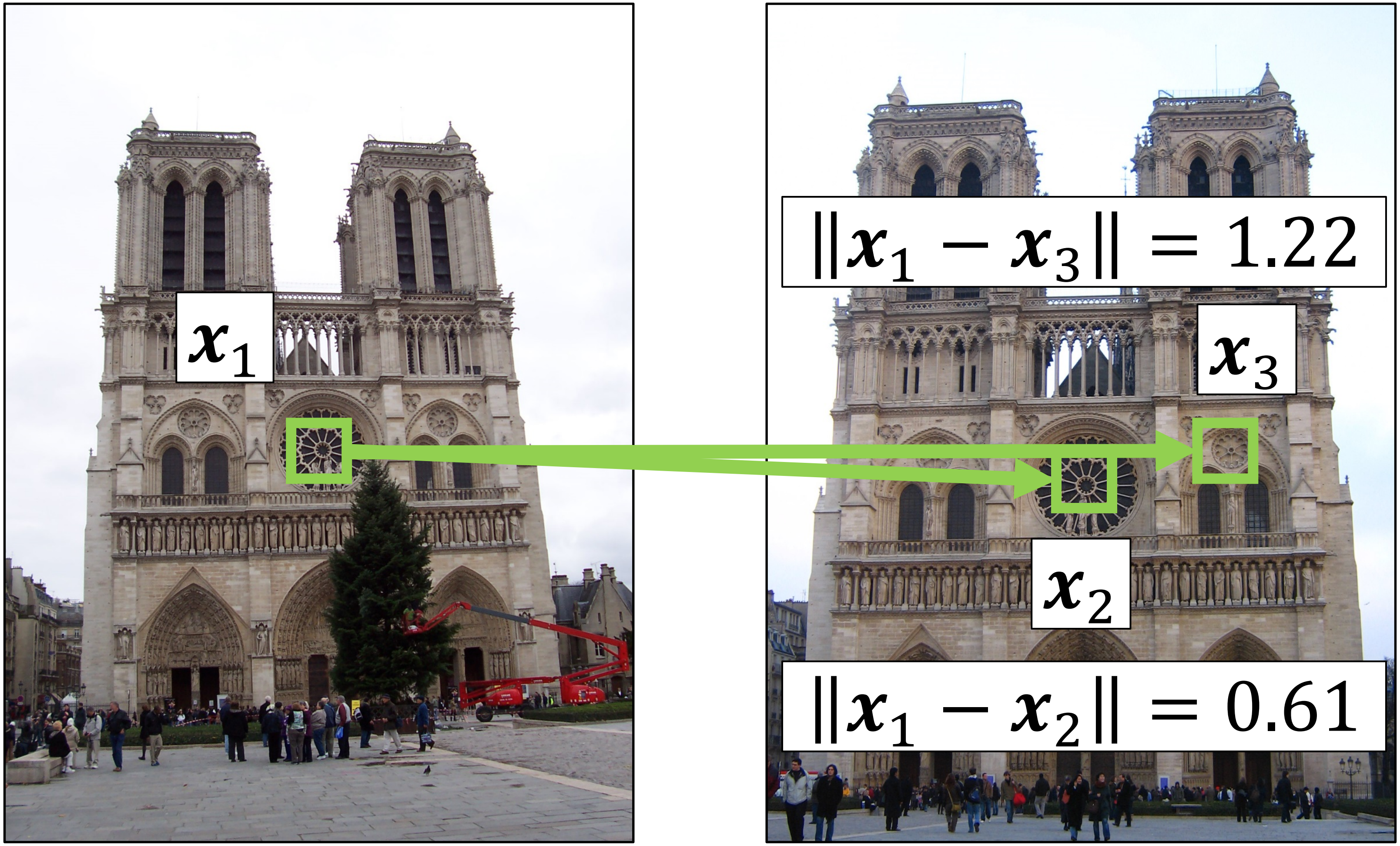
Think of feature as some non-linear filter that maps pixels to 128D feature

Photo credit: N. Snavely

128D vector **x**

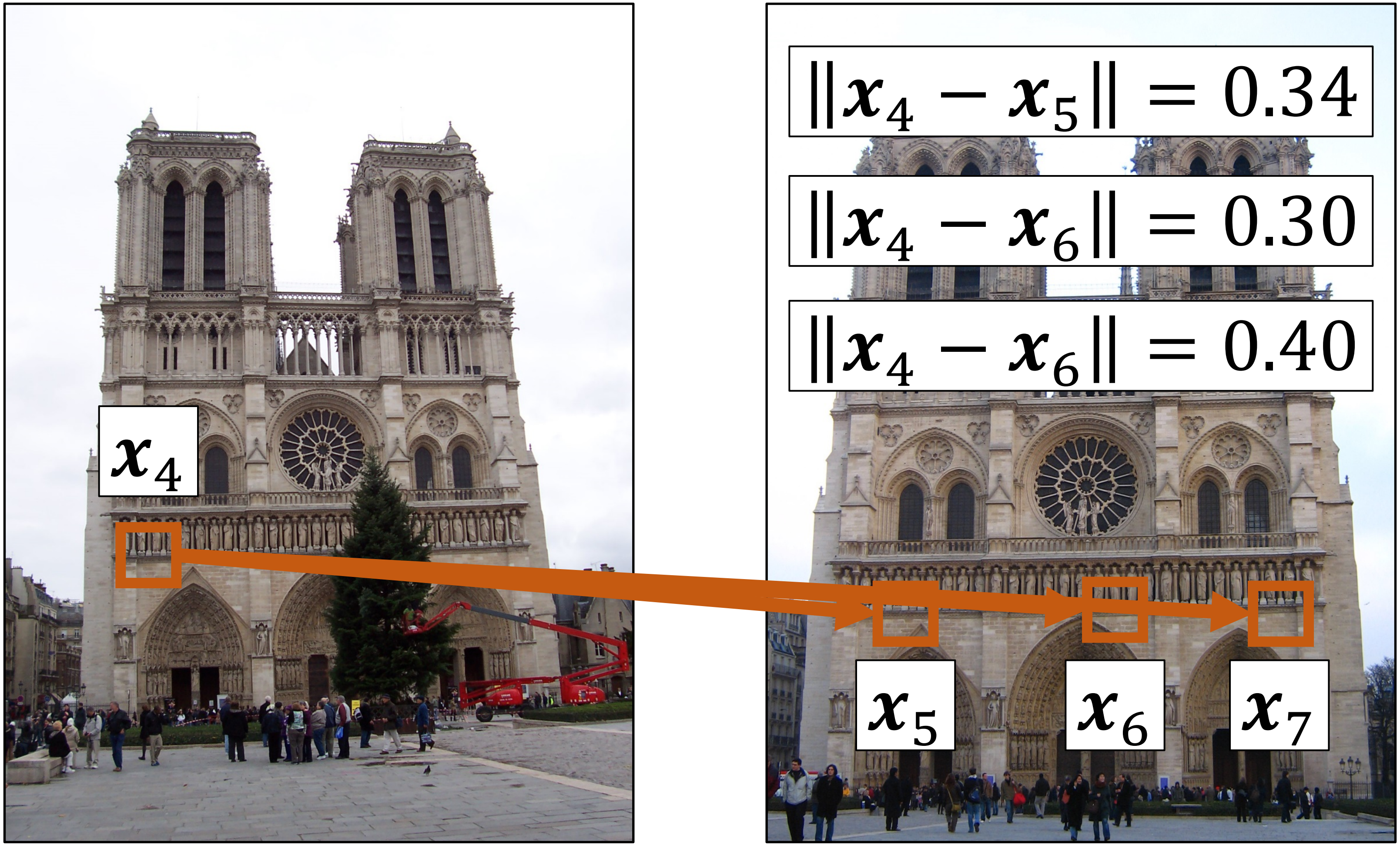
Distance between vectors gives us the visual appearance between patches

SIFT Features: Instance Matching



Example credit: J. Hays

SIFT Features: Instance Matching



Example credit: J. Hays

2^{nd} Nearest Neighbor Trick

- Given a feature x, nearest neighbor to x is a good match, but distances can't be thresholded.
- Instead, find nearest neighbor and second nearest neighbor. This ratio is a good test for matches:

$$r = \frac{\left\| \boldsymbol{x}_q - \boldsymbol{x}_{1NN} \right\|}{\left\| \boldsymbol{x}_q - \boldsymbol{x}_{2NN} \right\|}$$

2^{nd} Nearest Neighbor Trick

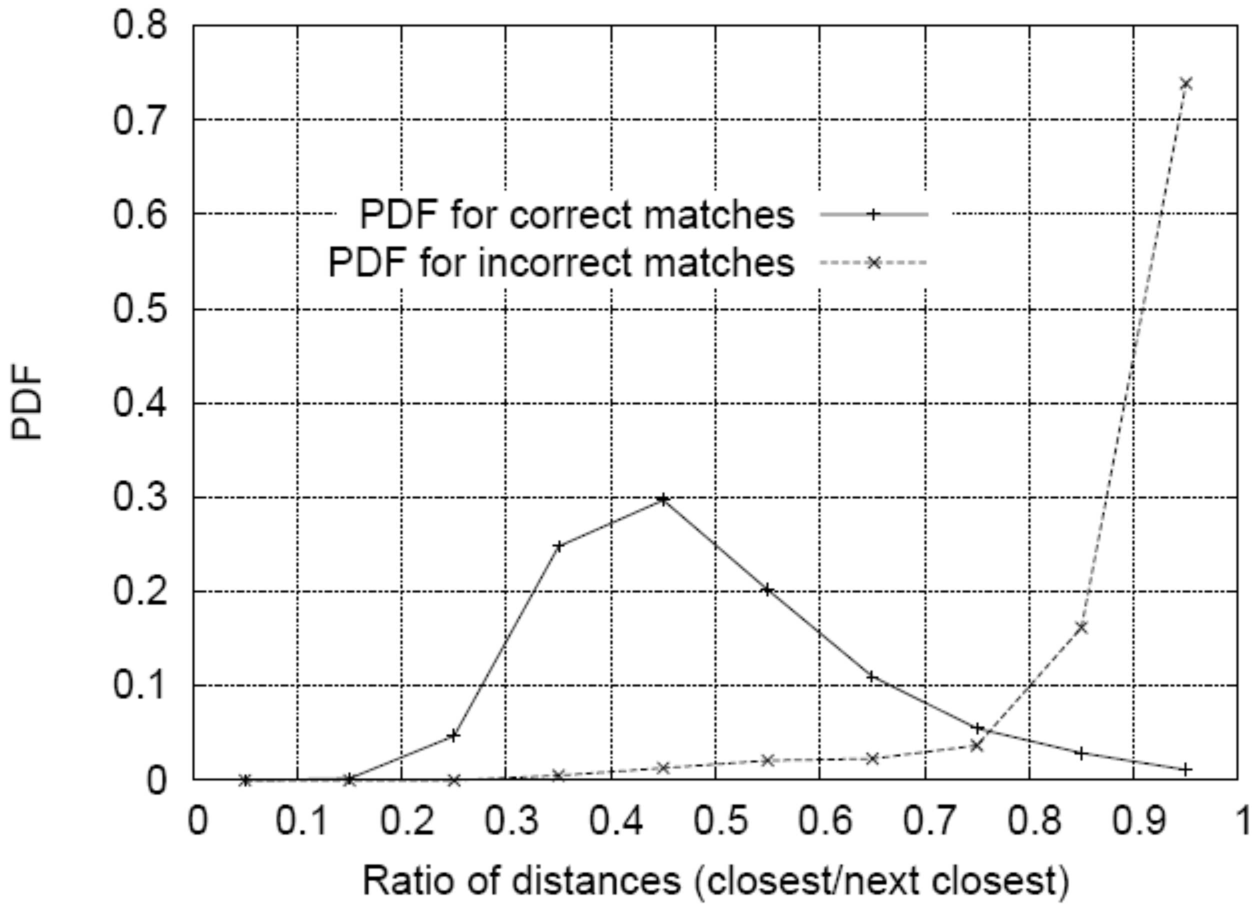
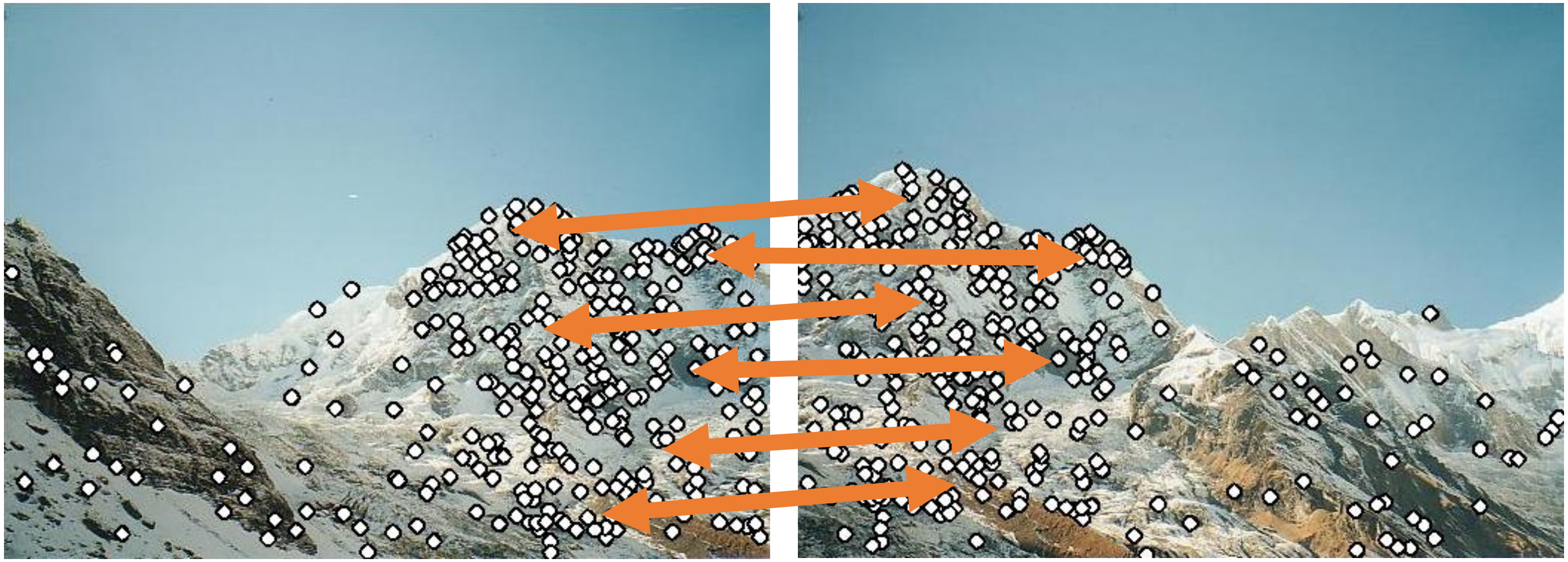


Figure from David G. Lowe. ["Distinctive image features from scale-invariant keypoints."]() *IJCV* 60 (2), pp. 91-110, 2004.

Recap

Finding and Matching

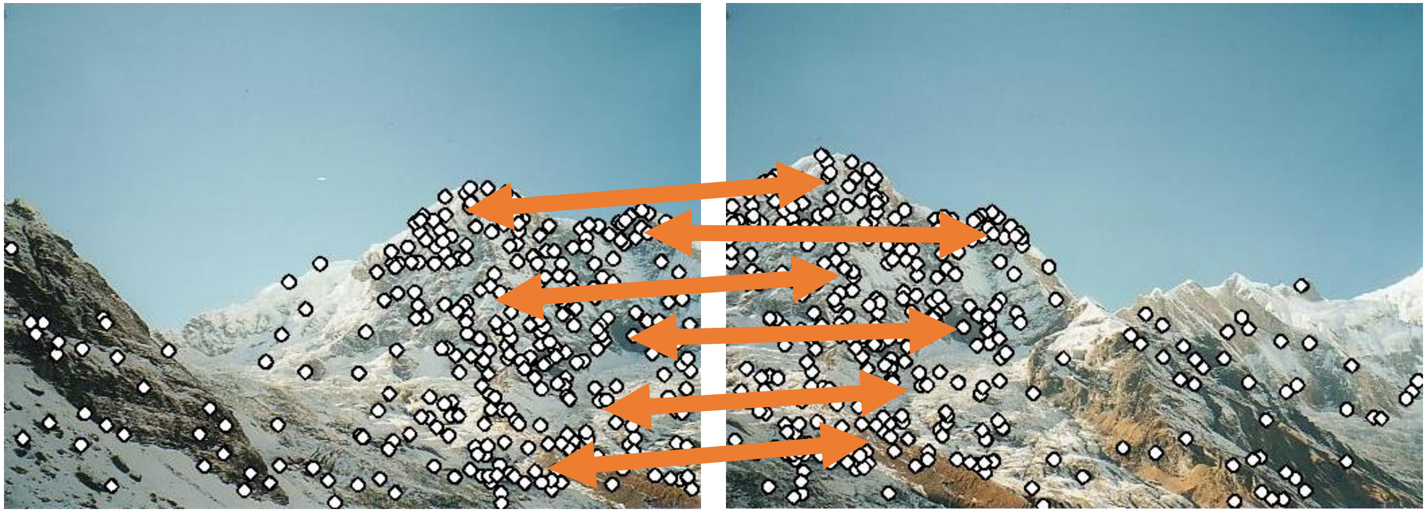


1: find corners+features

2: match based on local image data

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

Next Task: Find a Transform



Find a transform that brings our matched features together!