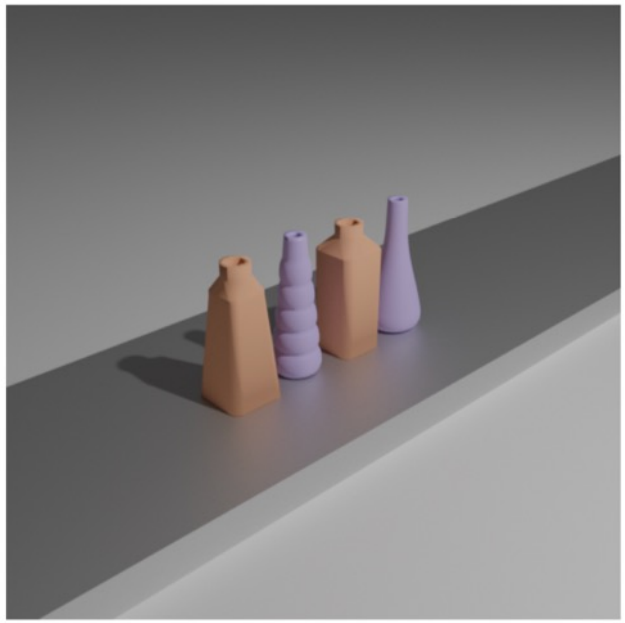
Features and Invariance

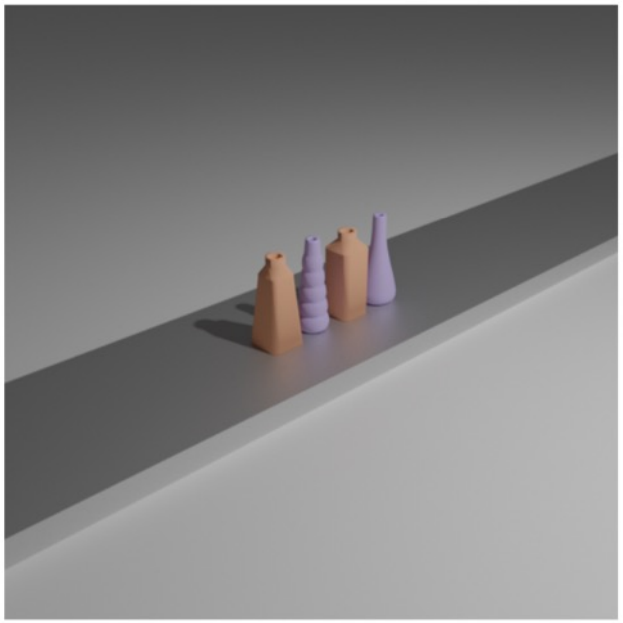
CS 482, Prof. Stein

Lecture 06

Motivation: Scale remains a challenge

What if we want to find objects common to two images:





What if we take the derivative?

What if we use curvature?

Neither of these properties are *scale invariant.*

Handling *scale* is a challenging problem in image-space.

Harris features still depends on scale!

Reading & Slide Credits

Readings:

- Szeliski: 4.1

Slide Credits (from which many of these slides are either directly taken or adapted)

- CMU Computer Vision Course (Yannis Gkioulekas)
- Cornell Tech Computer Vision Course (Noah Snavely)
- UNC CV Course (Svetlana Lazebnik)

Invariance and Equivariance

What do we mean when we say we want features that are *robust to change*?

Invariance: a feature is *invariant* to a change if the image is changed and the locations of the feature (corner) do not change.

Equivariance: a feature is *equivariant* to a change if it appears in corresponding locations before and after an image transformation is applied.

What do we mean when we say we want features that are *robust to change*?

Notice how many of the features appear in similar places on the toy between the two images.

What do we mean when we say we want features that are *robust to change*?

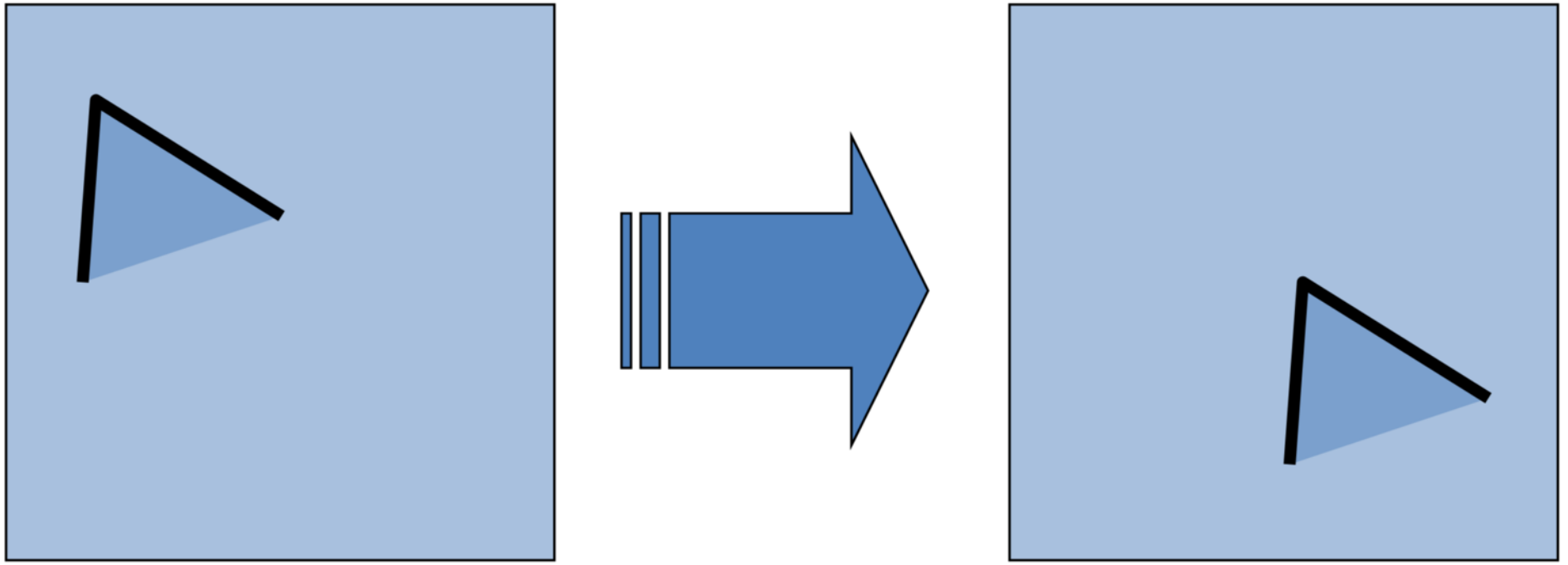
Invariance: a feature is *invariant* to a change if the image is changed and the locations of the feature (corner) do not change.

Equivariance: a feature is *equivariant* to a change if it appears in corresponding locations before and after an image transformation is applied.

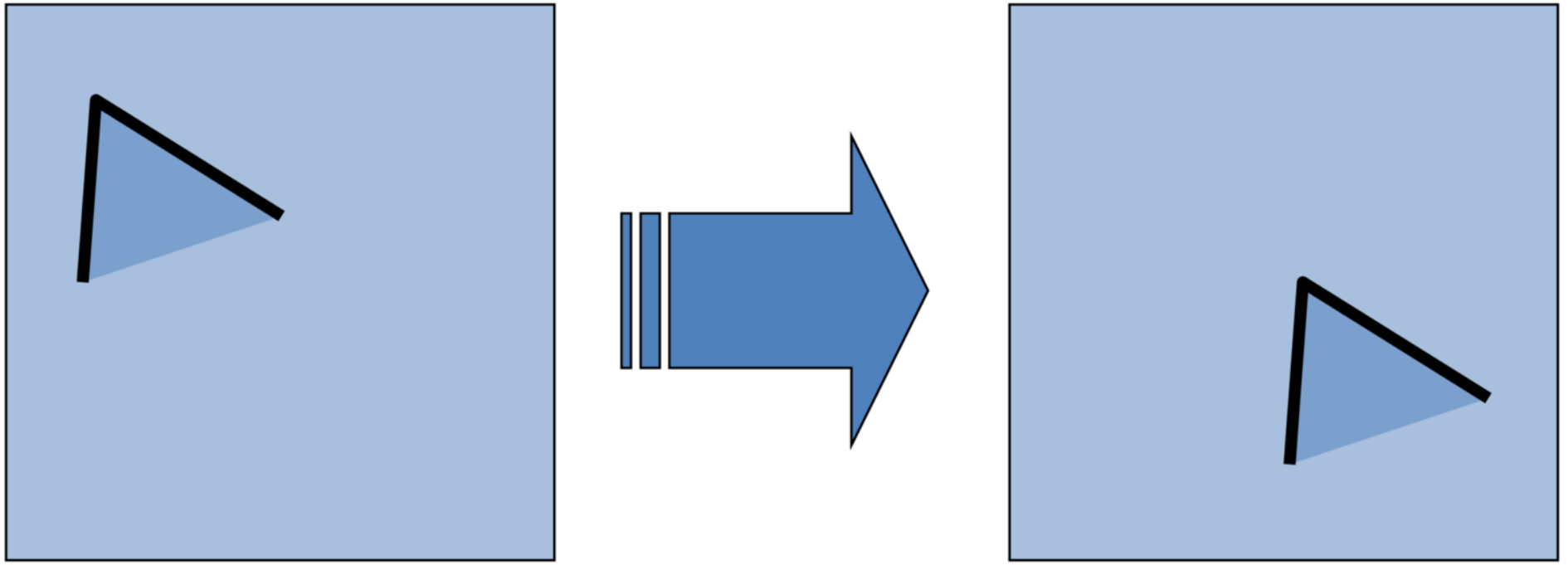
Confusingly:

- Sometimes “invariant” means either “invariant” or “equivariant”
- Sometimes “invariant” is called “covariant”

Harris Corners under image translation: Invariant or Equivariant?

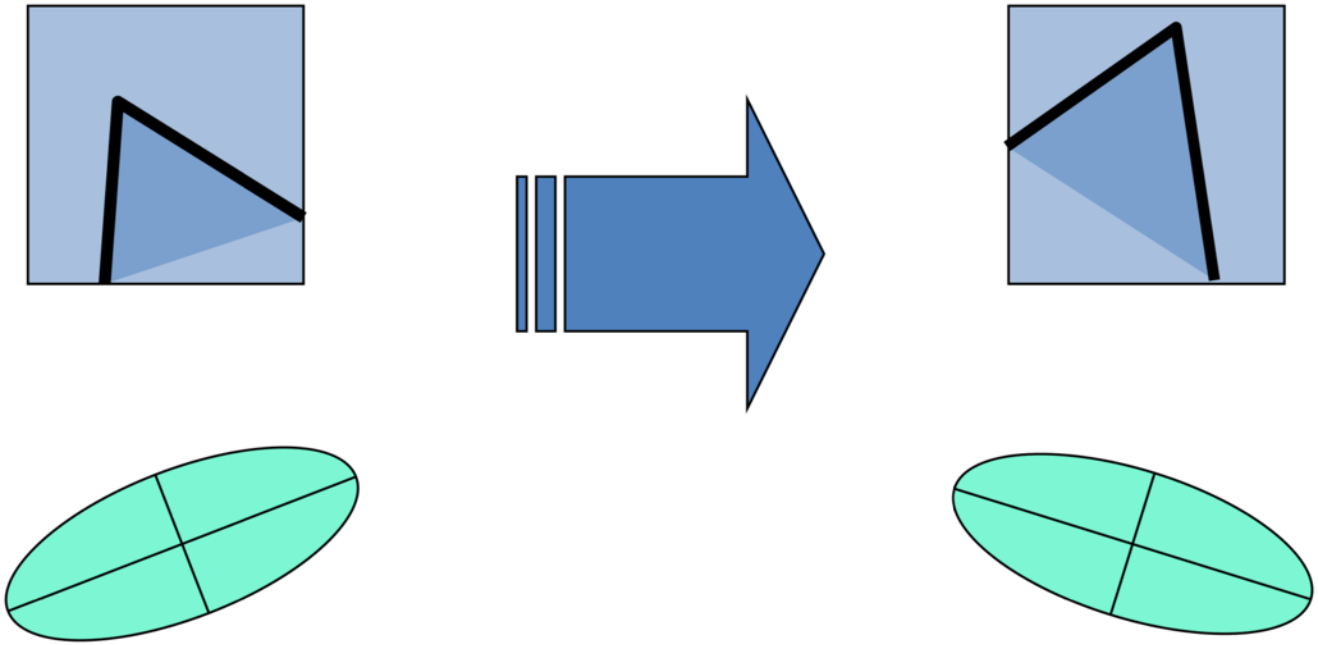


Harris Corners under image translation: Invariant or Equivariant?

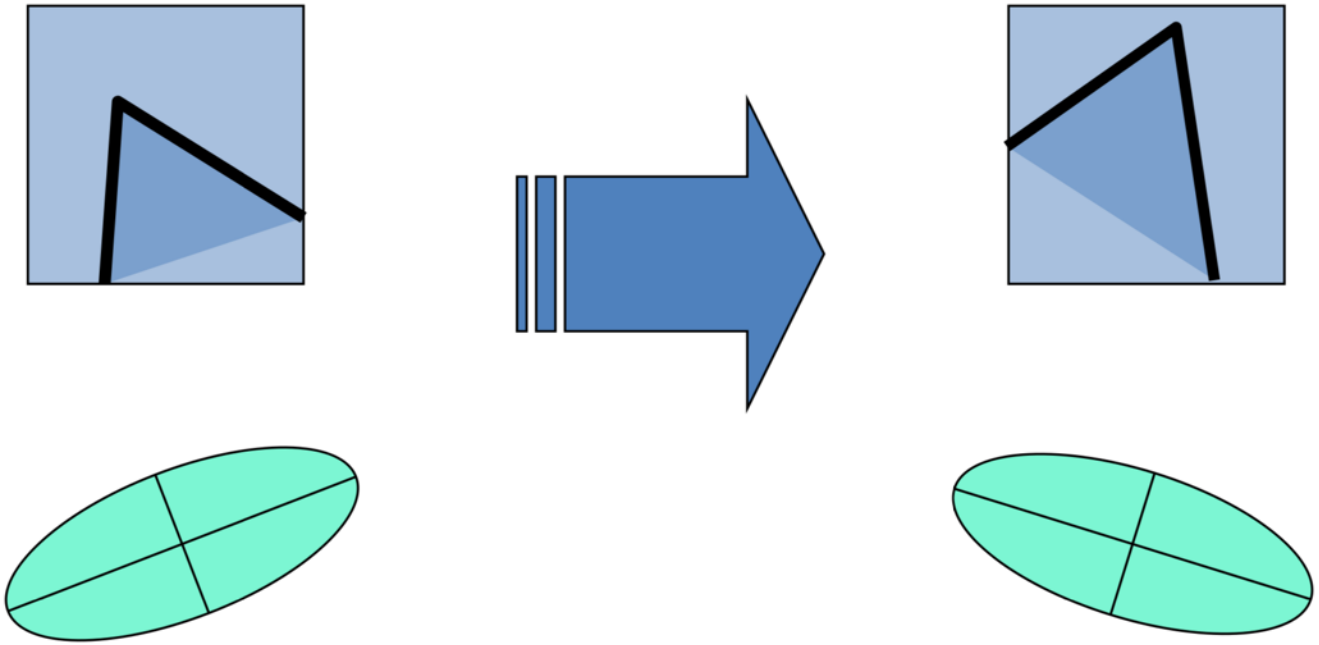


Harris Corners/Features are *equivariant* to image translation: they move with the image after transformation.

Harris Corners under image rotation: Invariant or Equivariant?

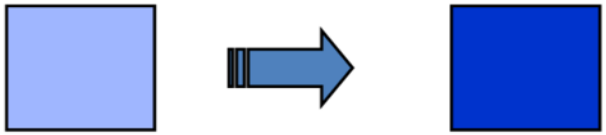


Harris Corners under image rotation: Invariant or Equivariant?



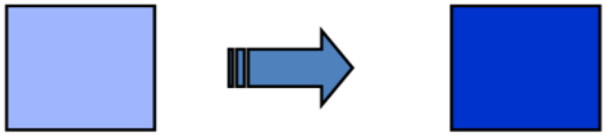
Harris Corners/Features are *equivariant* to image rotation: the eigenvectors change, but the eigenvalues themselves remain unchanged (discretization will change this *slightly*).

Harris Corners under linear intensity change: Invariant or Equivariant?

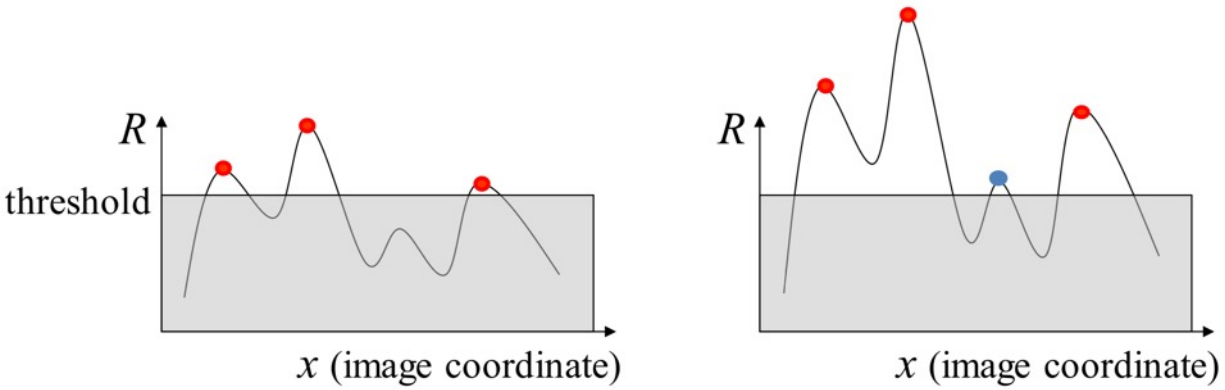


$$I \rightarrow a\,I + b$$

Harris Corners under linear intensity change: Invariant or Equivariant?



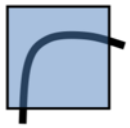
$$I \rightarrow a\,I + b$$



Harris Corners/Features are partially (not completely) invariant under linear/affine intensity changes.

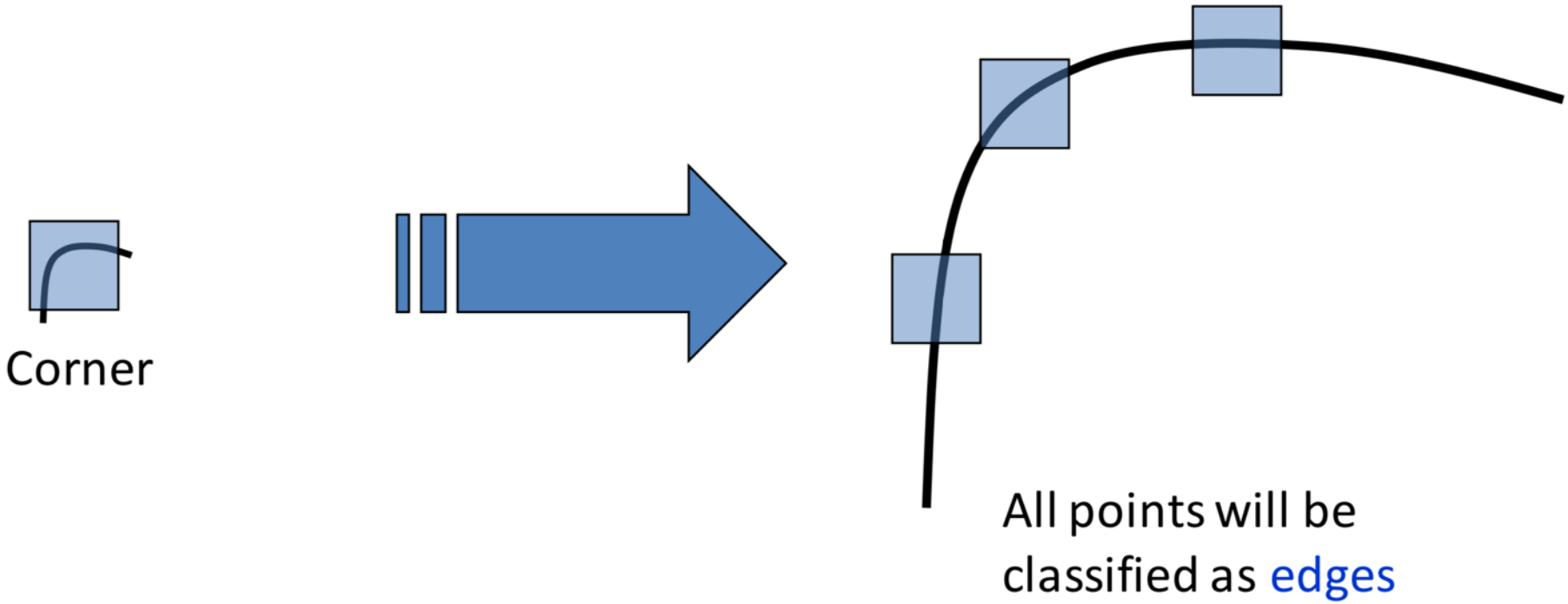
What about scale?

What about scale?



Corner

What about scale?

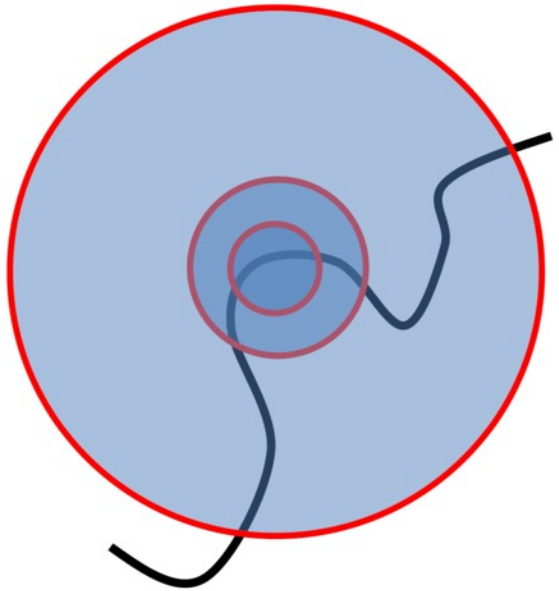


Harris Corners/Features are neither invariant nor equivariant under scale.

Multi-scale Feature Detection

Let's say we're looking to detect corners in an image

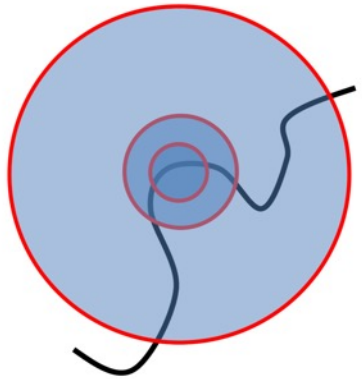
What if we tried detecting features at multiple scales simultaneously:



Which of these scales is "best"

Let's say we're looking to detect corners in an image

Key idea for multi-scale detection: find a scale for which the *f* (our measure of *cornerness*) is maximized.

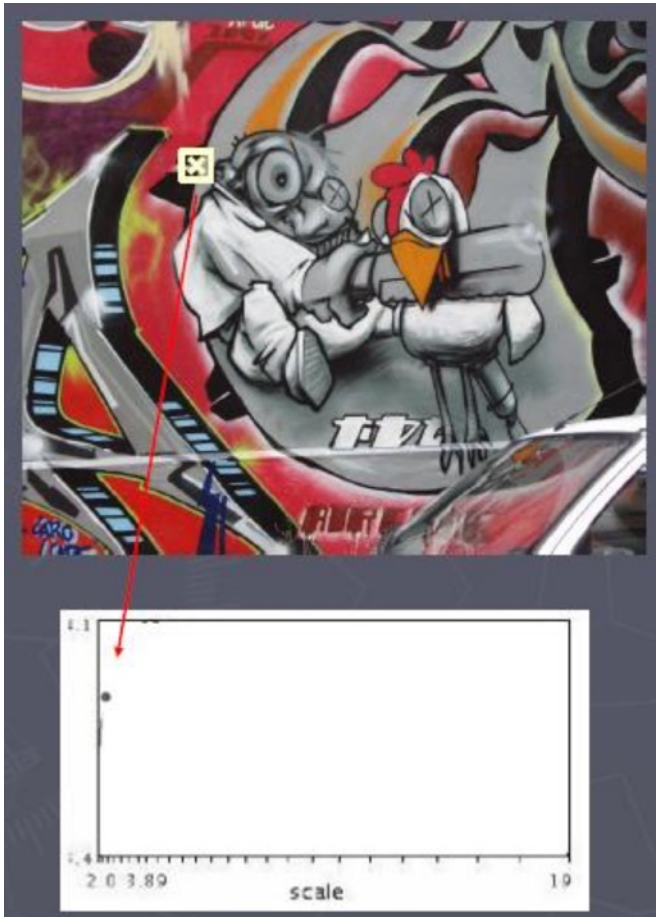


We can change:

- The size of the feature kernel
- The size of the image

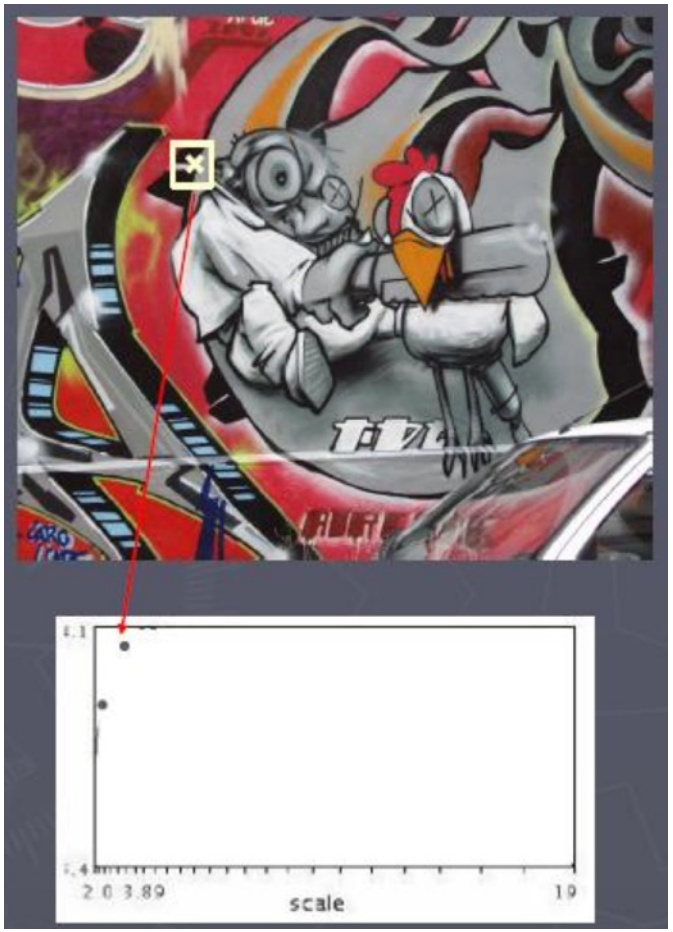
Usually, which we choose is a matter of implementation.

Example: Automatic Scale Selection



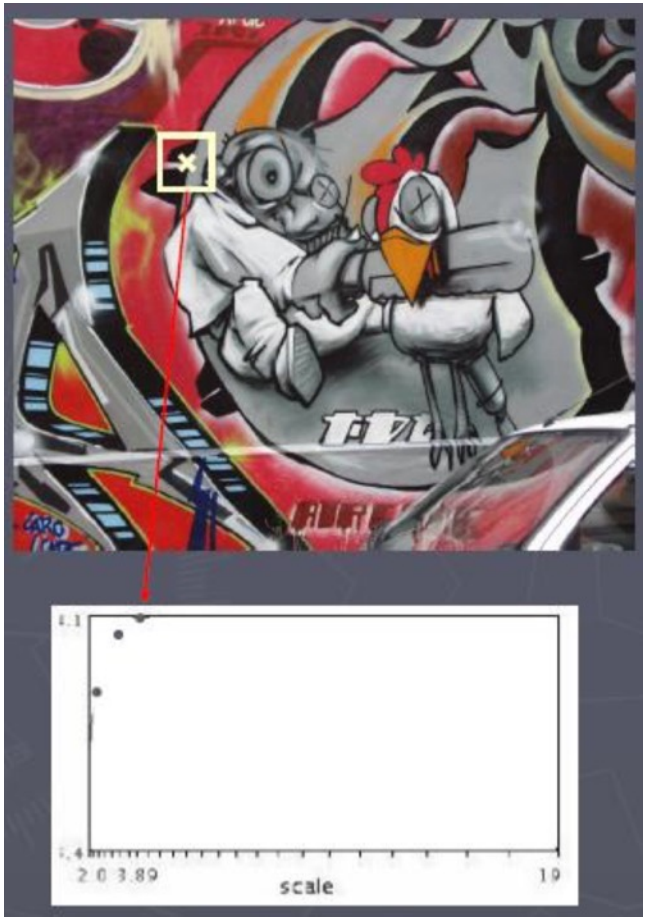
If we change the size of the kernel, its response will change.

Example: Automatic Scale Selection



If we change the size of the kernel, its response will change.

Example: Automatic Scale Selection



If we change the size of the kernel, its response will change.

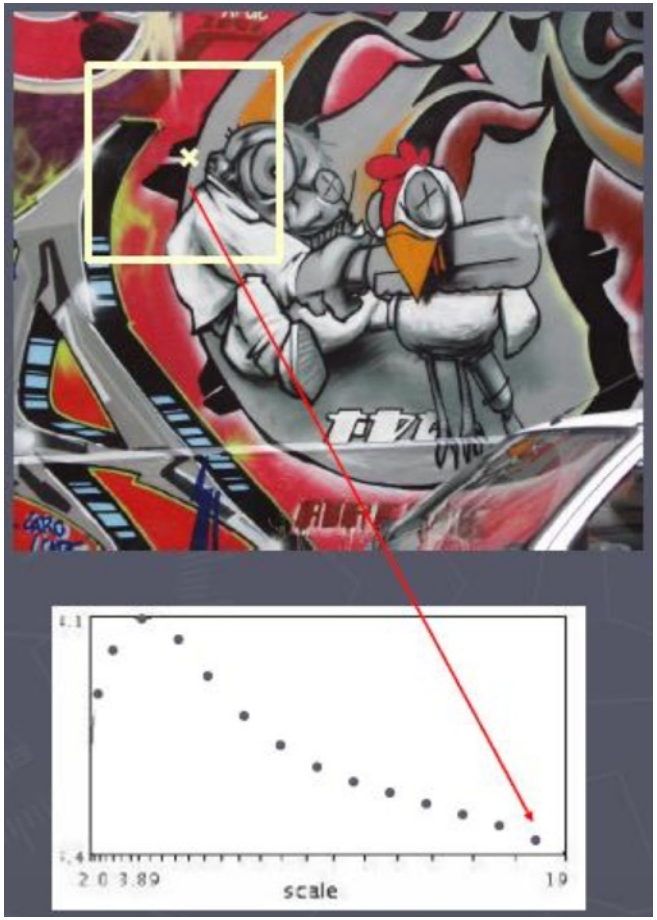
Example: Automatic Scale Selection

If we change the size of the kernel, its response will change.

Example: Automatic Scale Selection

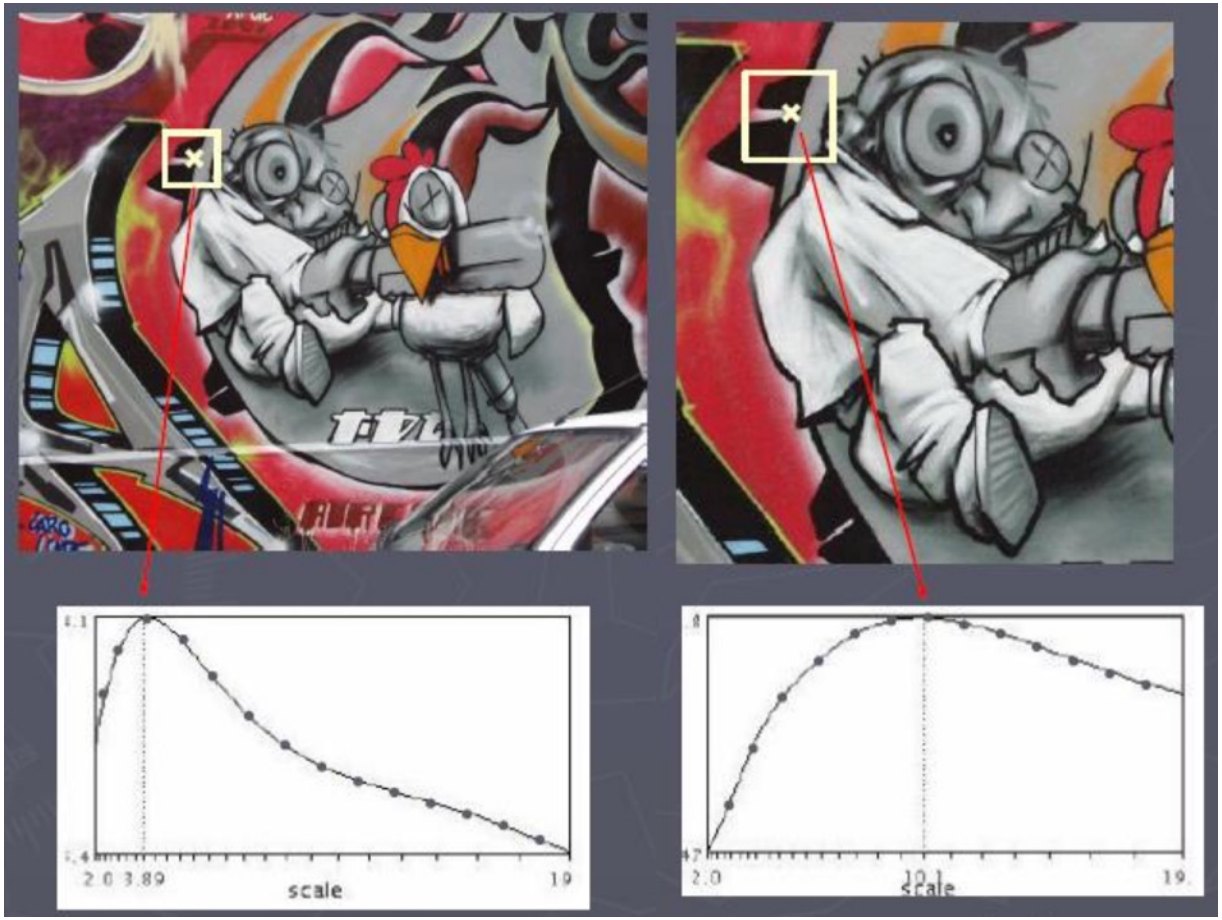
If we change the size of the kernel, its response will change.

Example: Automatic Scale Selection



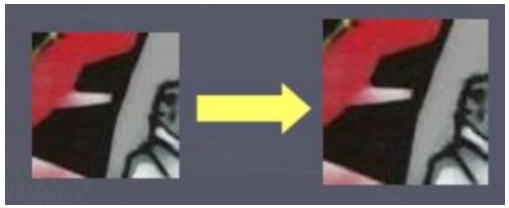
If we change the size of the kernel, its response will change.

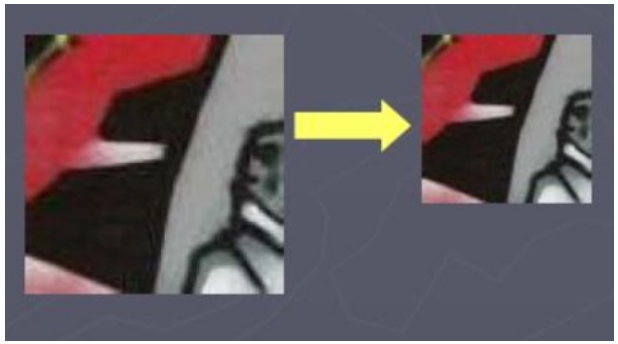
Example: Automatic Scale Selection



Matching features in images of different scales should have corresponding automatically-detected scales, which we can use to resize the features themselves.

Example: Automatic Scale Selection

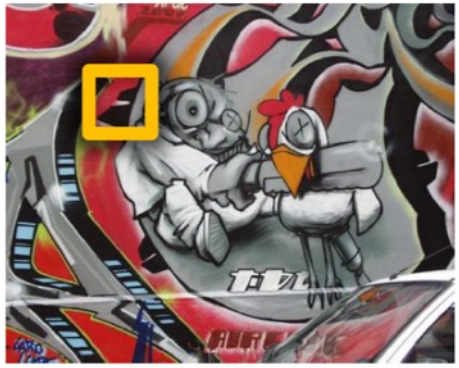


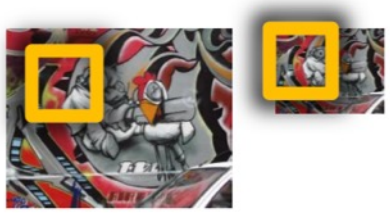


Matching features in images of different scales should have corresponding automatically-detected scales, which we can use to resize the features themselves.

Relatedly: we've already talked about using image resizing to handle scale changes.

Instead of changing the filter size, we might change the size of the image and keep the filter size the same:

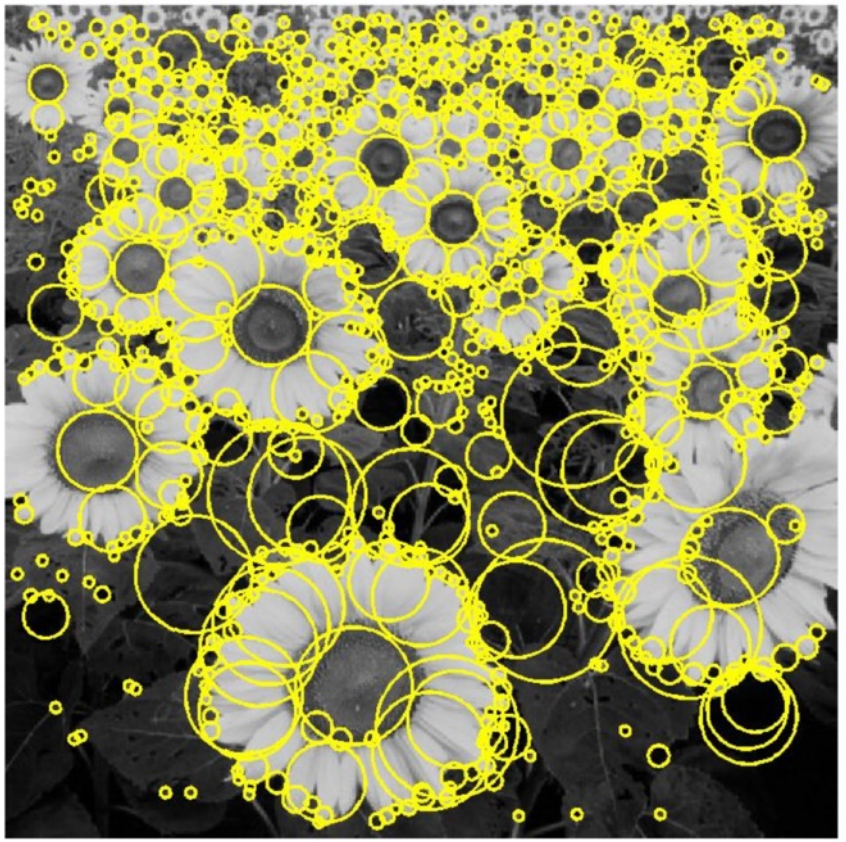




(sometimes need to create in-between levels, e.g. a ¾-size image)

So far, we've mostly talked about corners, but there are different types of features.

Corner Detection

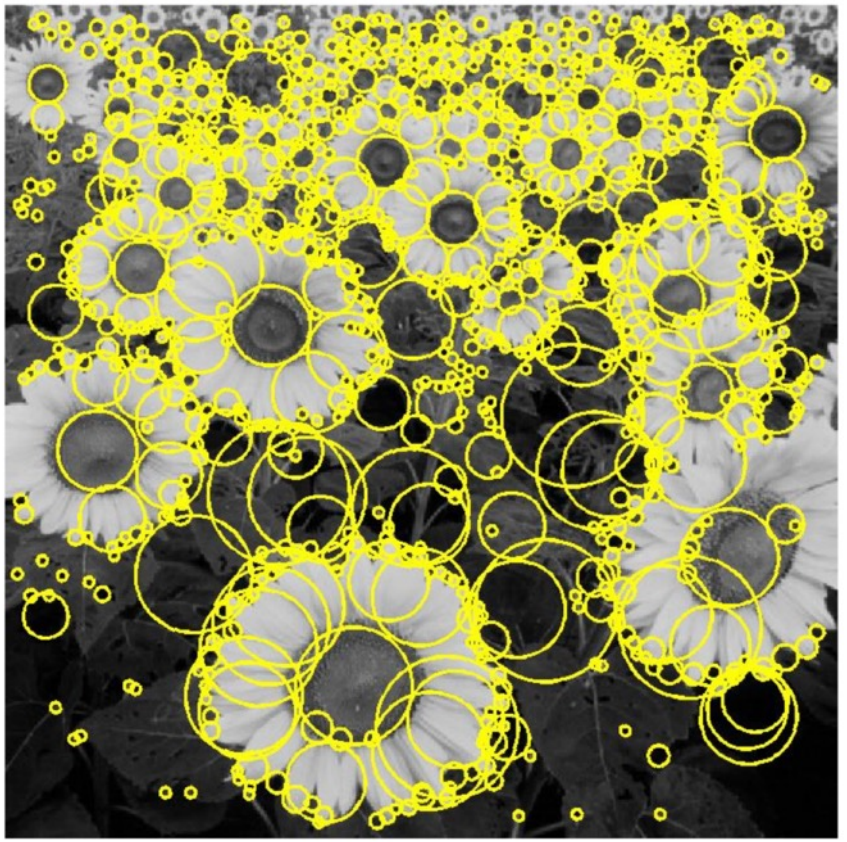


Blob Detection

So far, we've mostly talked about corners, but there are different types of features.

Corner Detection

Harris Operator

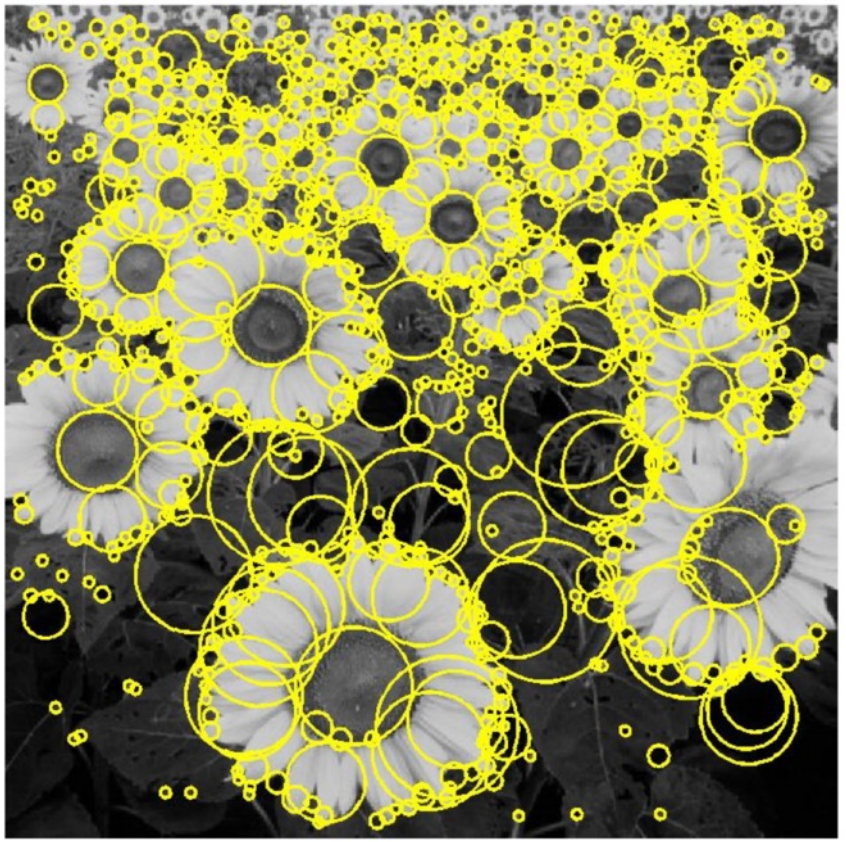


Blob Detection

So far, we've mostly talked about corners, but there are different types of features.

Corner Detection

Harris Operator



Blob Detection

Laplacian of Gaussians

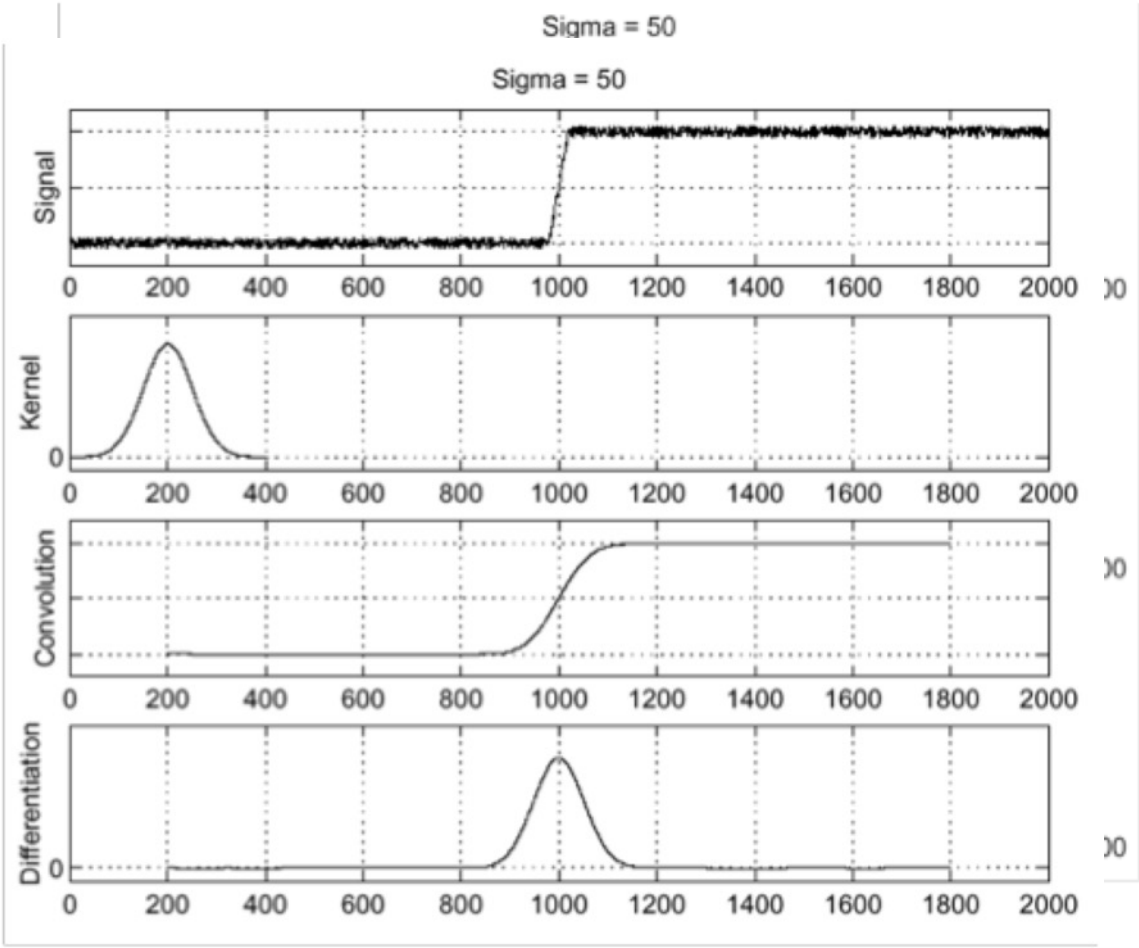
Reminder: Edge detection with a Derivative of Gaussian filter

input

Gaussian

blurred

derivative of blurred

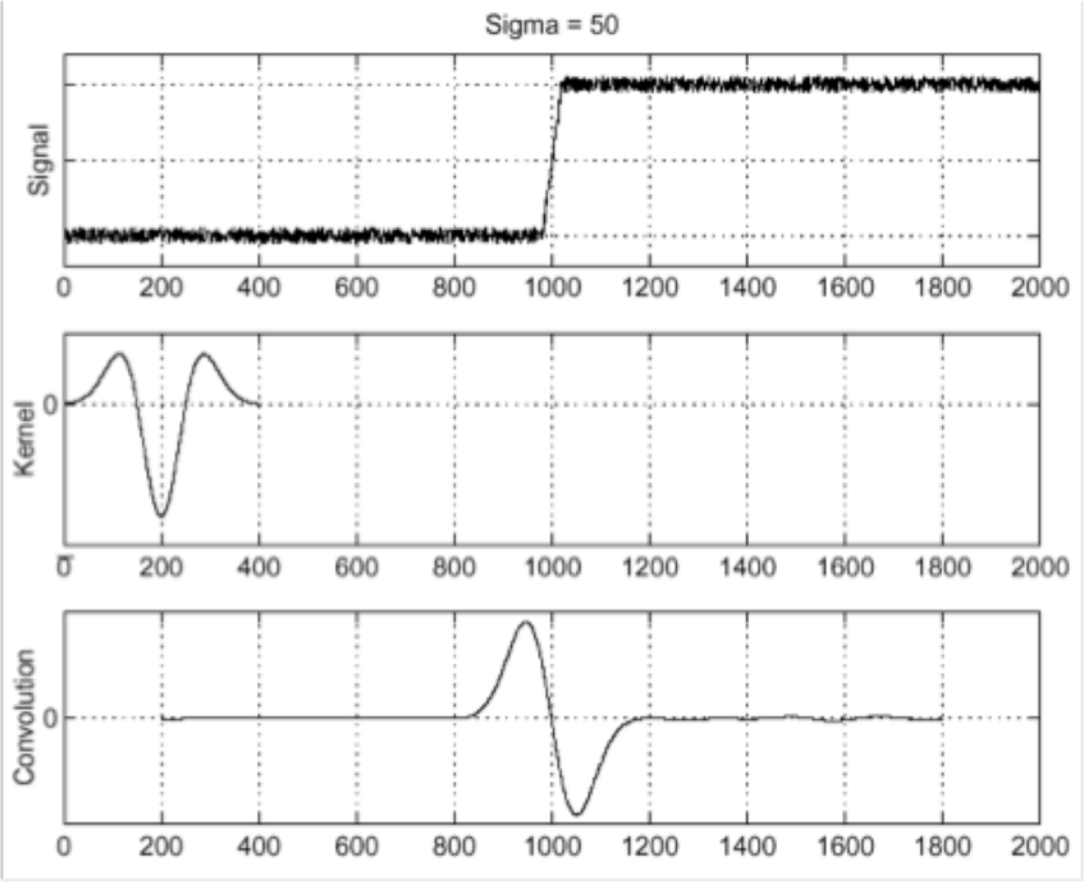


Reminder: Edge detection with a Laplacian of Gaussian

input

Laplacian of Gaussian

output



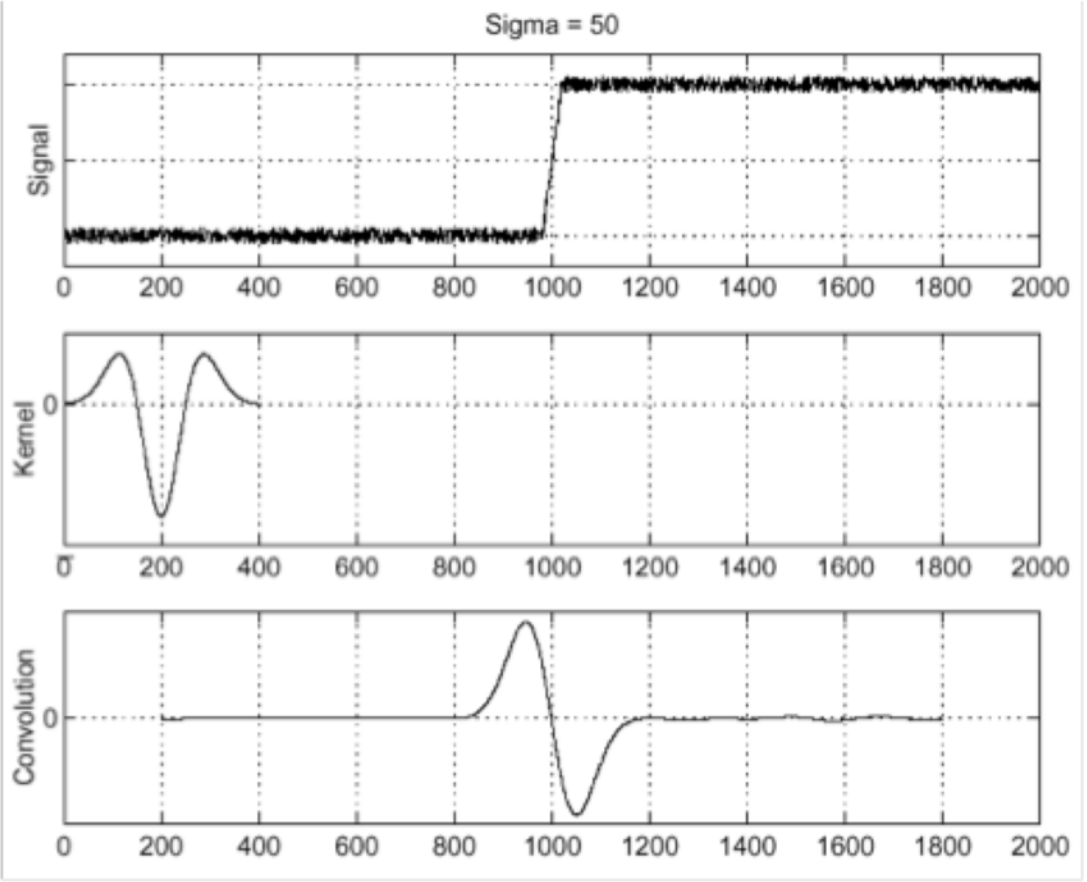
Edges occur at zero-crossings after applying the LoG filter

Reminder: Edge detection with a Laplacian of Gaussian

input

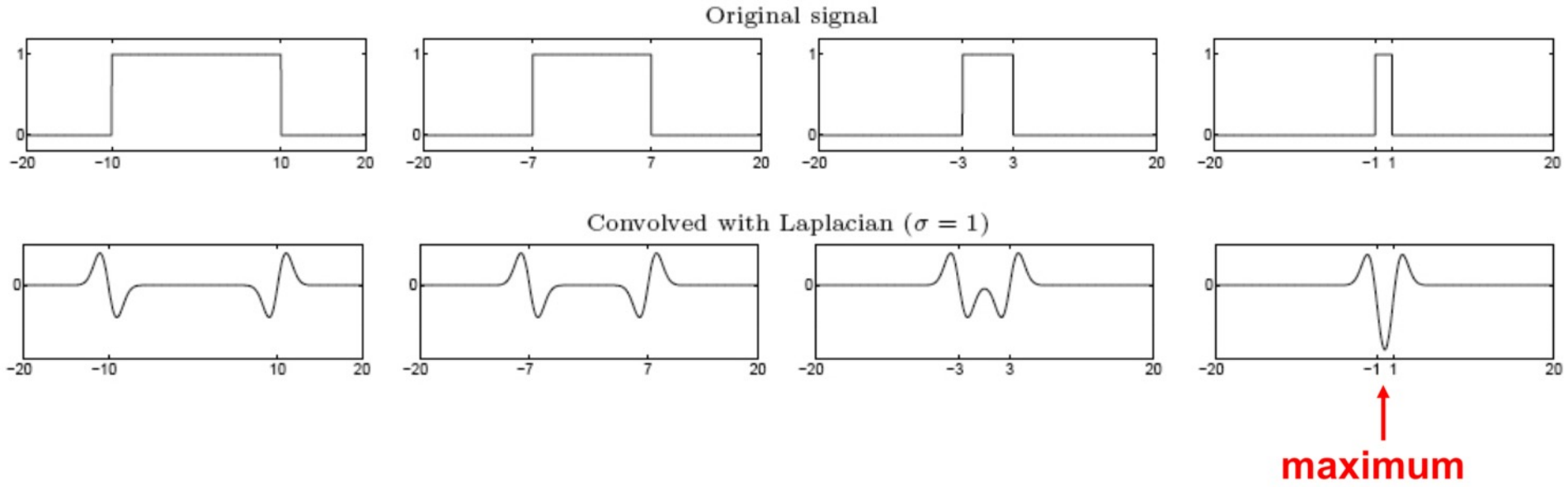
Laplacian of Gaussian

output



Edges occur at zero-crossings after applying the LoG filter

When there are two nearby edges, the responses of the LoG filter will interfere

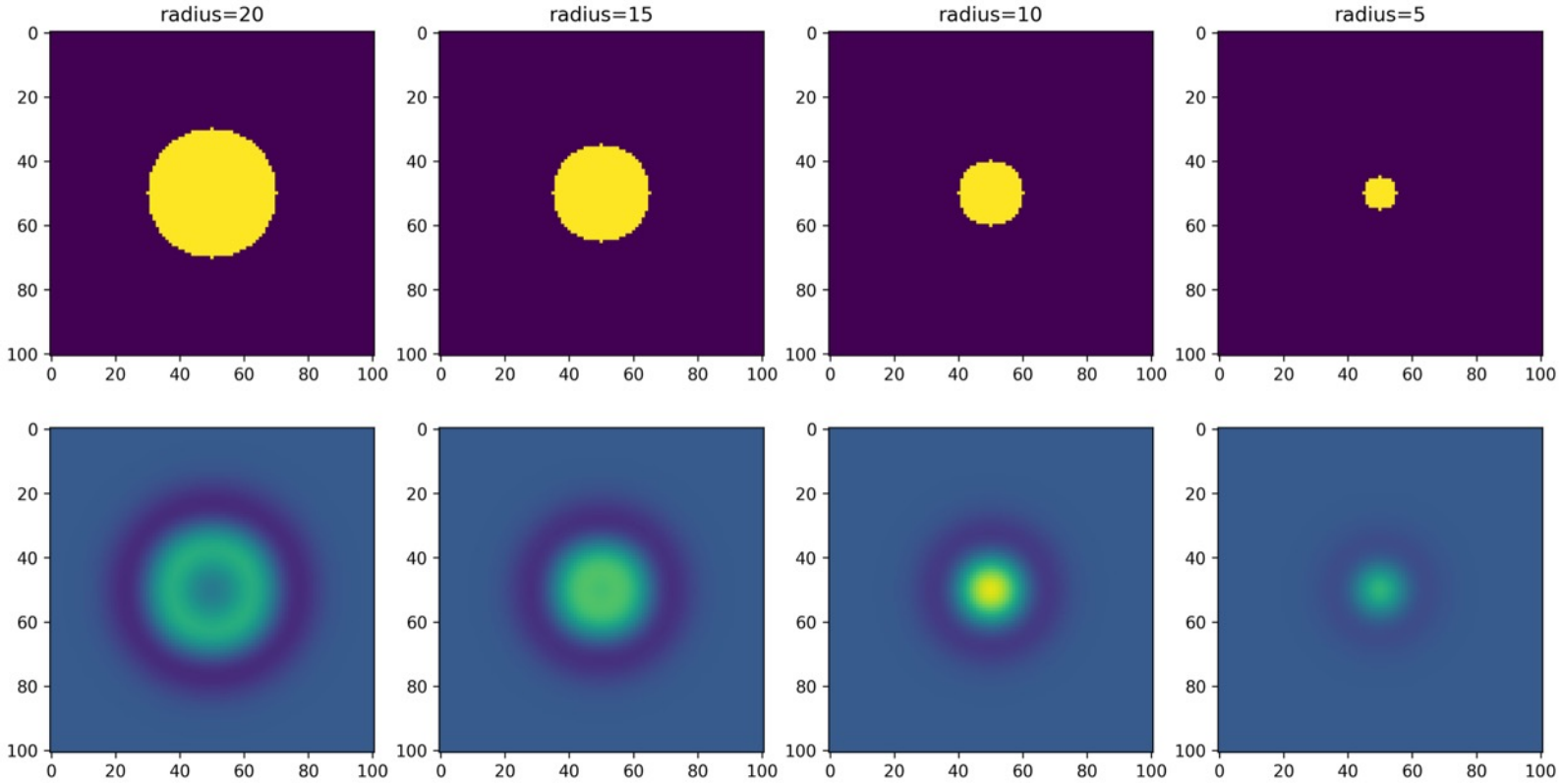


If we pick the "right" size of the LoG filter for the "blob" of interest, we will get a maximum response.

How do we match the two?

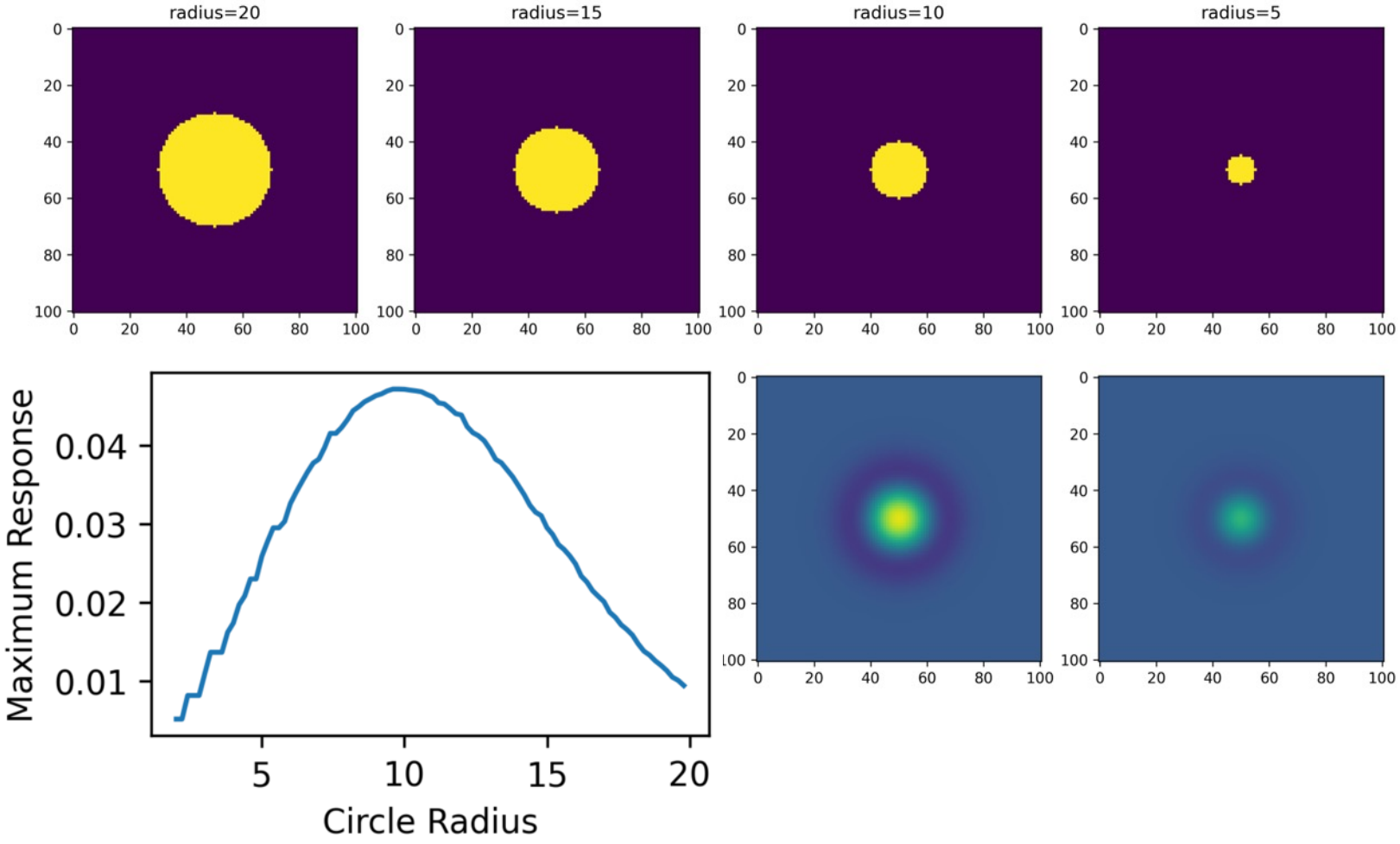
Let's try it! Apply a LoG filter to blobs of different sizes

$$\sigma = 7$$



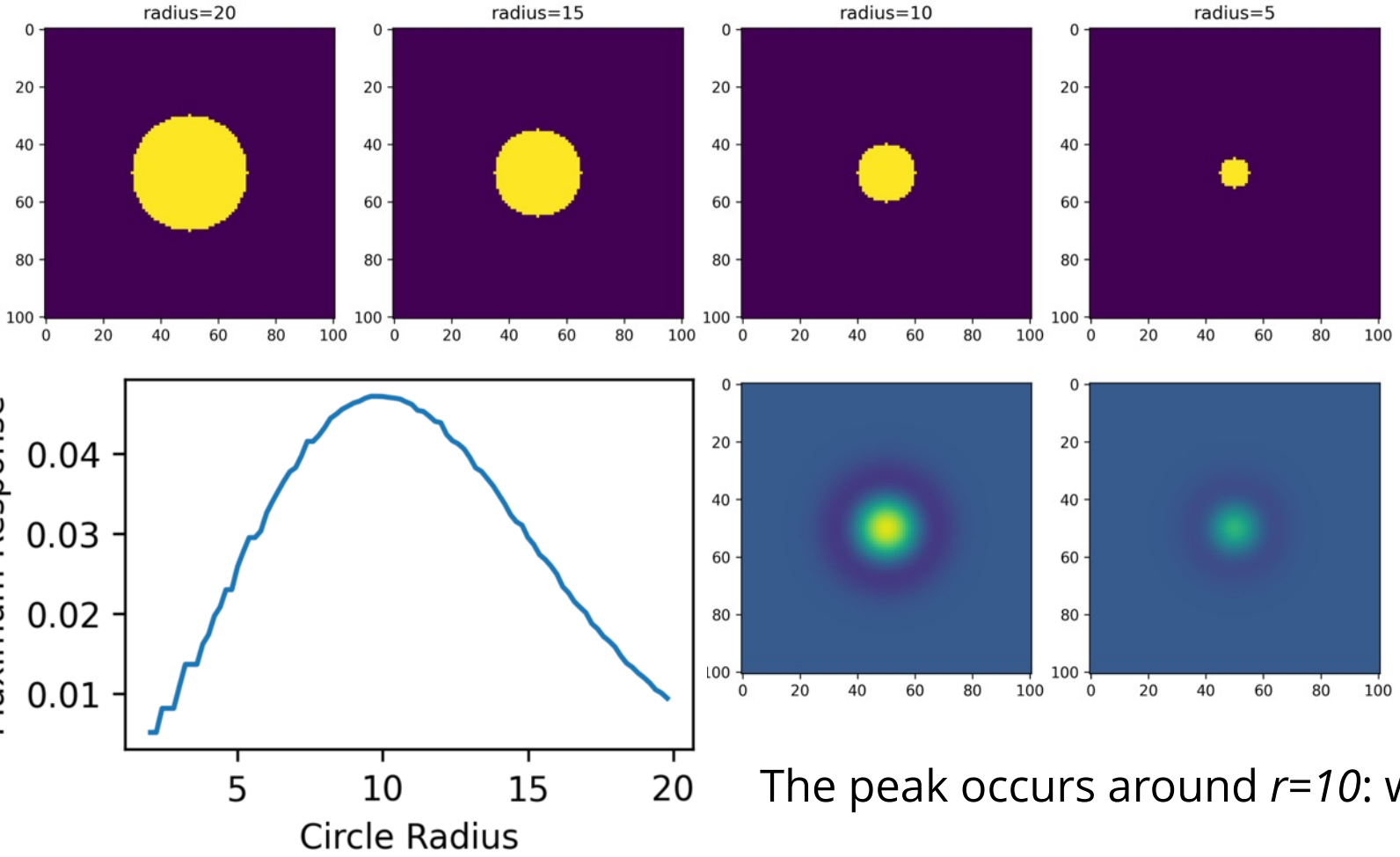
Let's try it! Apply a LoG filter to blobs of different sizes

$$\sigma = 7$$



Let's try it! Apply a LoG filter to blobs of different sizes

$\sigma = 7$



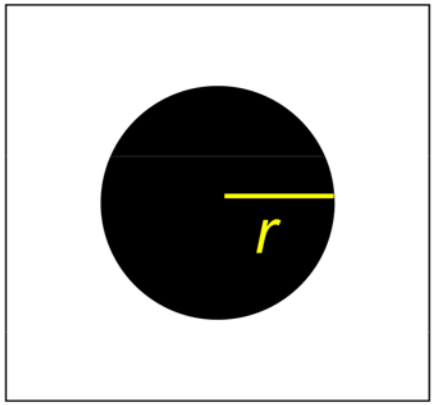
The peak occurs around *r=10*: why?

Matching LoG filters to blobs

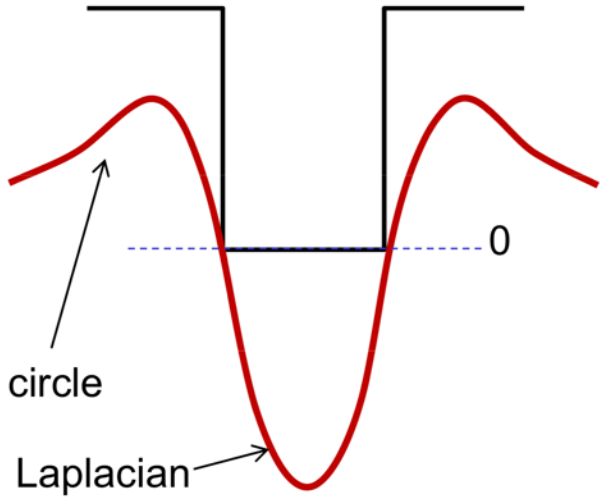
$$\text{LoG} = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right]\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Matching LoG filters to blobs

$$\text{LoG} = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$



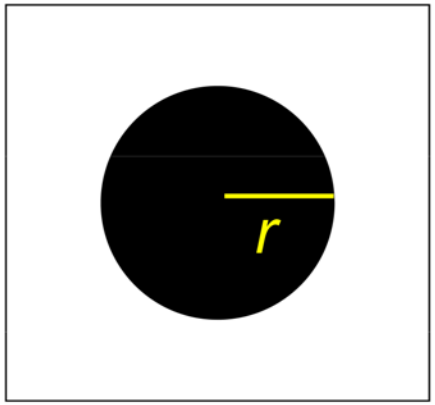
image



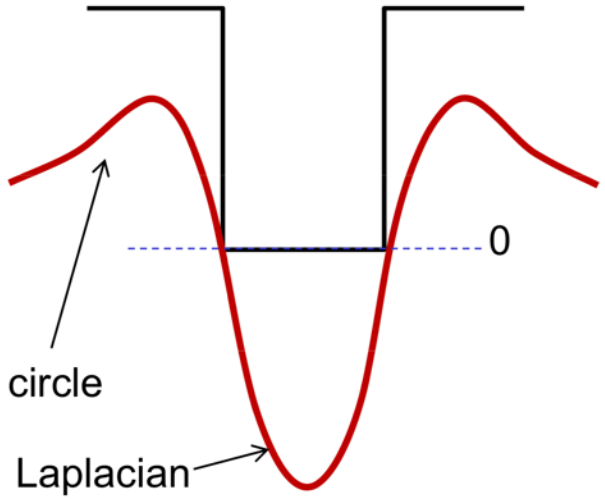
The filter response will be maximized if the zero-crossing of the Laplacian filter centered at the origin occurs at the boundary of the blob.

Matching LoG filters to blobs

$$\text{LoG} = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$



image

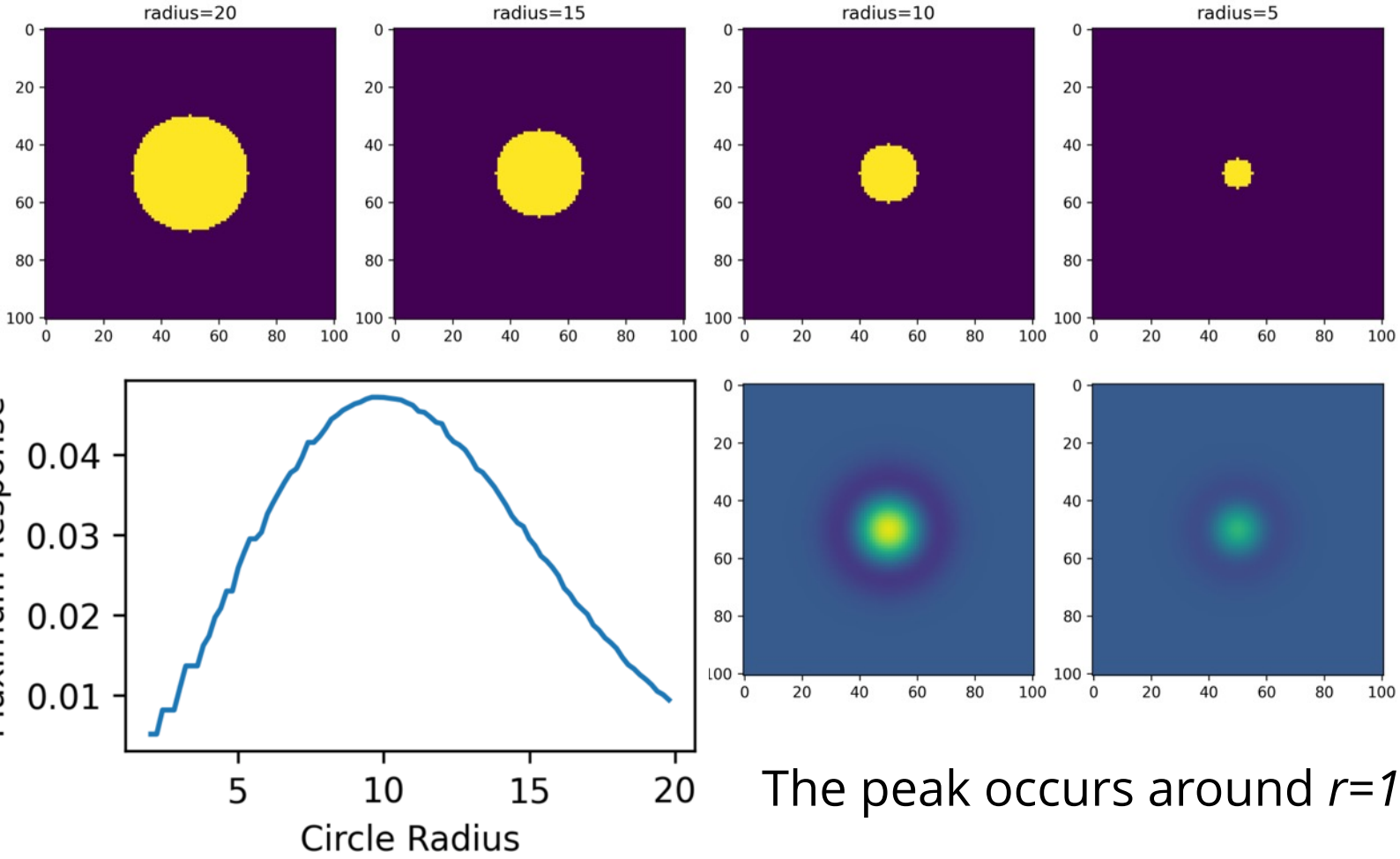


The filter response will be maximized if the zero-crossing of the Laplacian filter centered at the origin occurs at the boundary of the blob.

$$\sigma = r/\sqrt{2}$$

Let's try it! Apply a LoG filter to blobs of different sizes

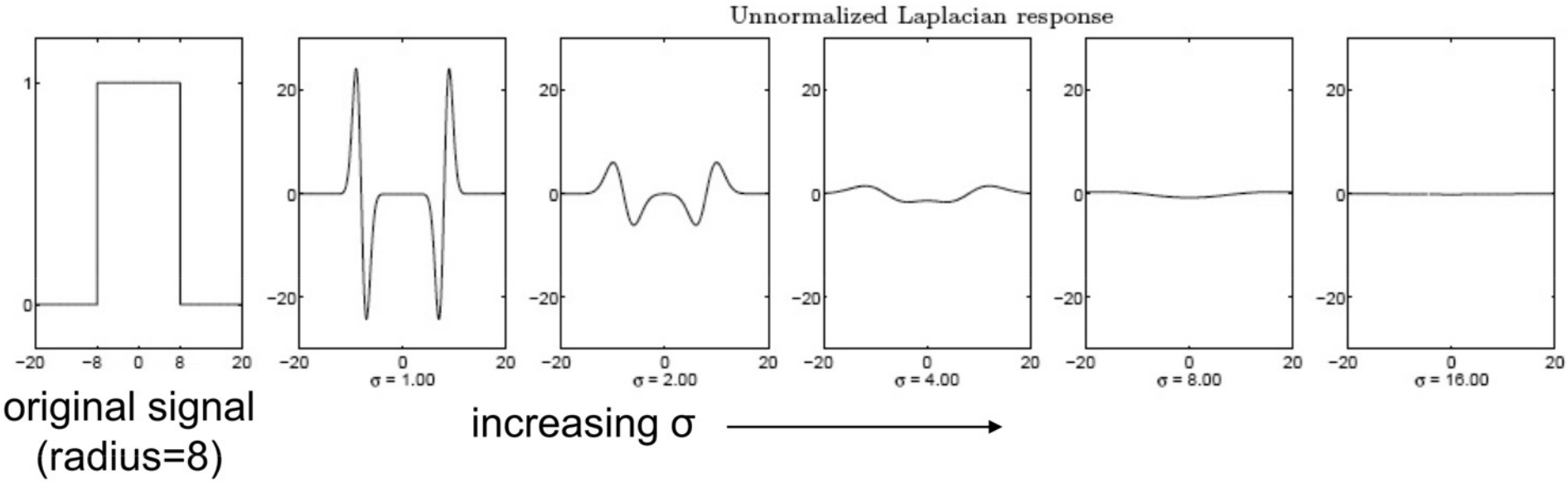
$\sigma = 7$



The peak occurs around *r=10*

Usually, we're given an image and asked to figure out what size the features are

But there's a problem:



As we increase the size of the kernel, the response of the LoG filter decreases.

The response of the LoG filter to an edge decreases as sigma increases

The derivative of the scaled-up feature will shrink because the slope is scaled with the feature of interest. The same is true of our "wider" LoG filter.

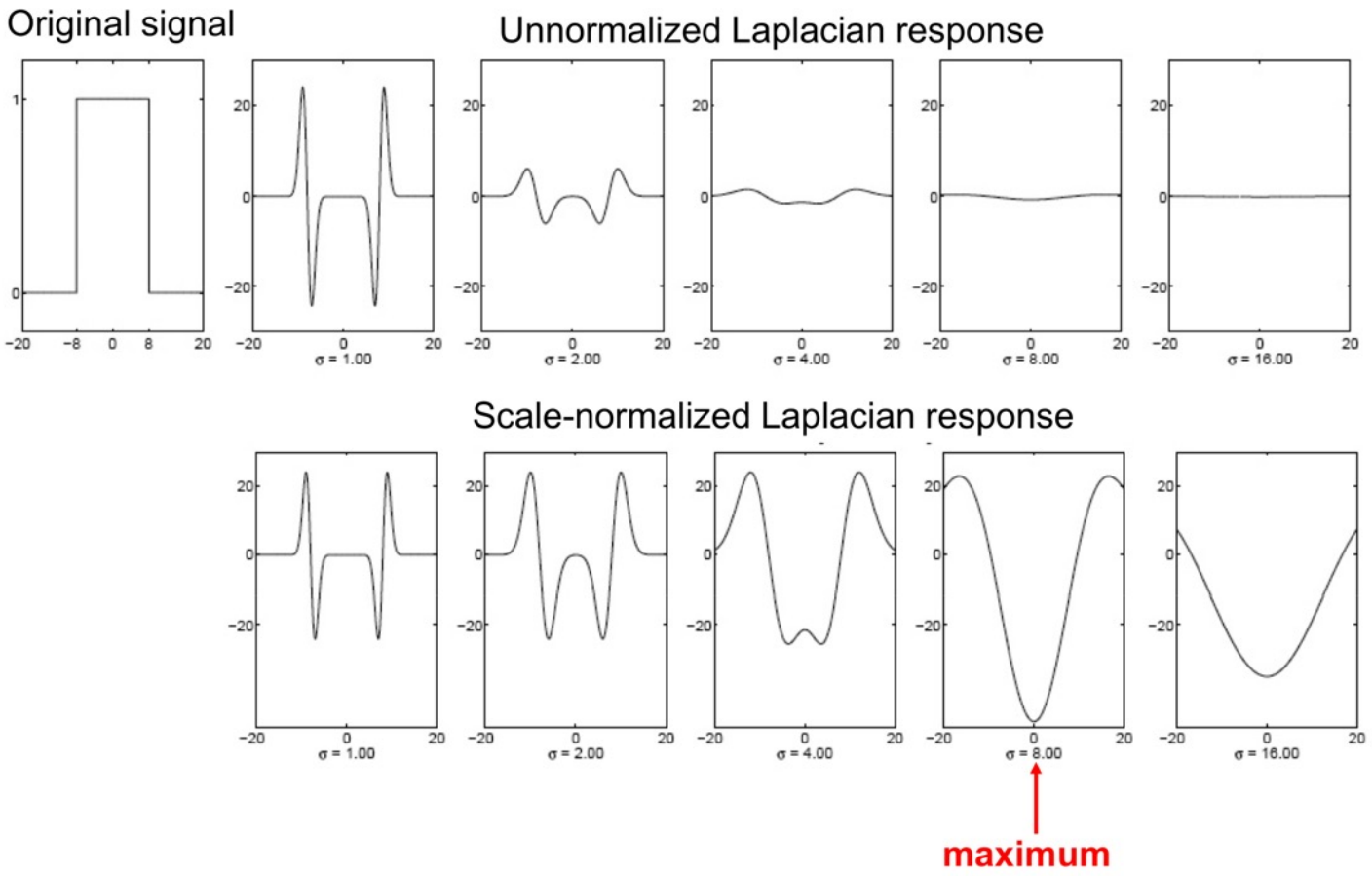
We need to introduce a *scale-normalized* version of the LoG filter:

$$x \rightarrow \frac{x}{\sigma}$$

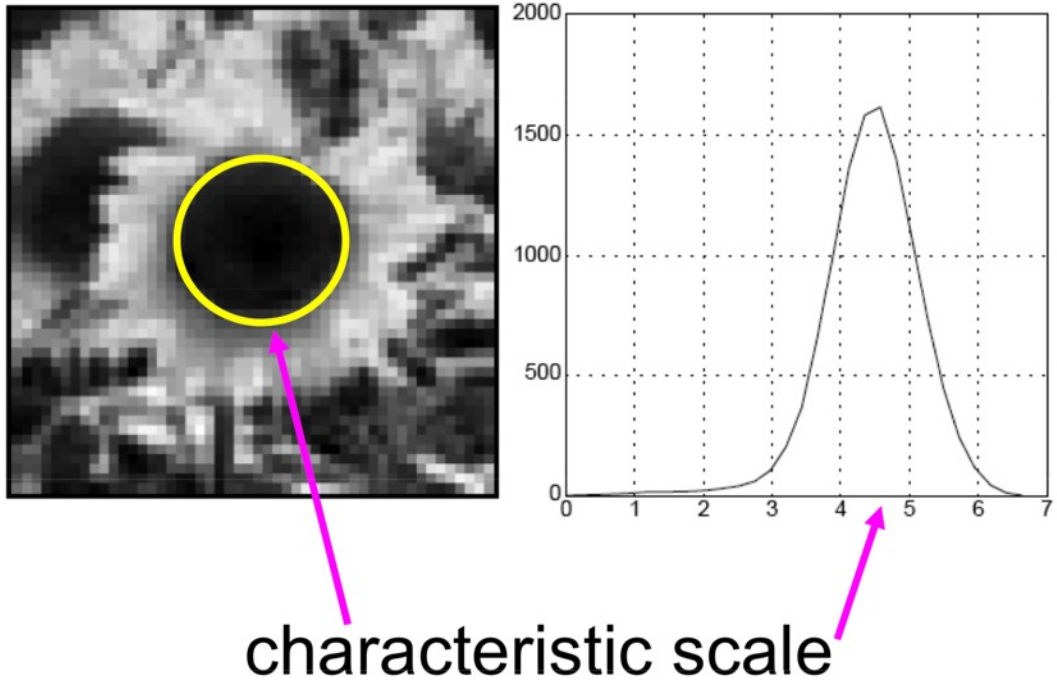
$$\frac{\partial f}{\partial x} \rightarrow \sigma \frac{\partial f}{\partial x}$$

$$\mathrm{LoG}_{\mathrm{norm}} = \sigma^2 \mathrm{LoG} = -\frac{\sigma^2}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Using our scale-normalized LoG filter, the maximum occurs where we expect



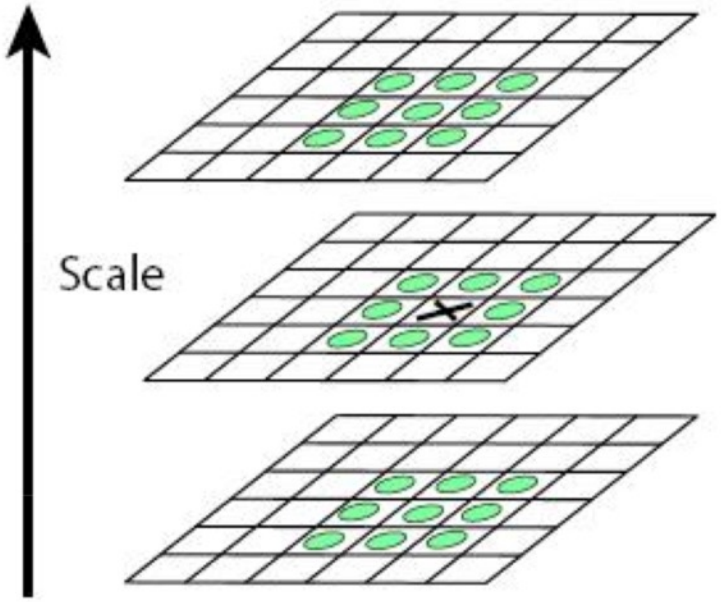
We can visualize these features by plotting both their location and "radius"



Paper: T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

Scale-aware Blob Detection Procedure

1. Apply the scale-normalized Laplacian of Gaussian at several image scales (our “cornerness score”)
2. Find the maxima of the filter in scale-space (needed if the feature appears at multiple scales)



We want to find features that are extrema in image space (as before) *and* in scale-space.

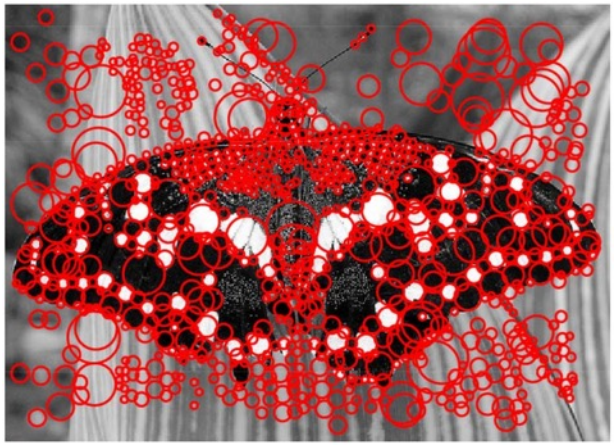
Example of scale-aware blob detection

Example of scale-aware blob detection

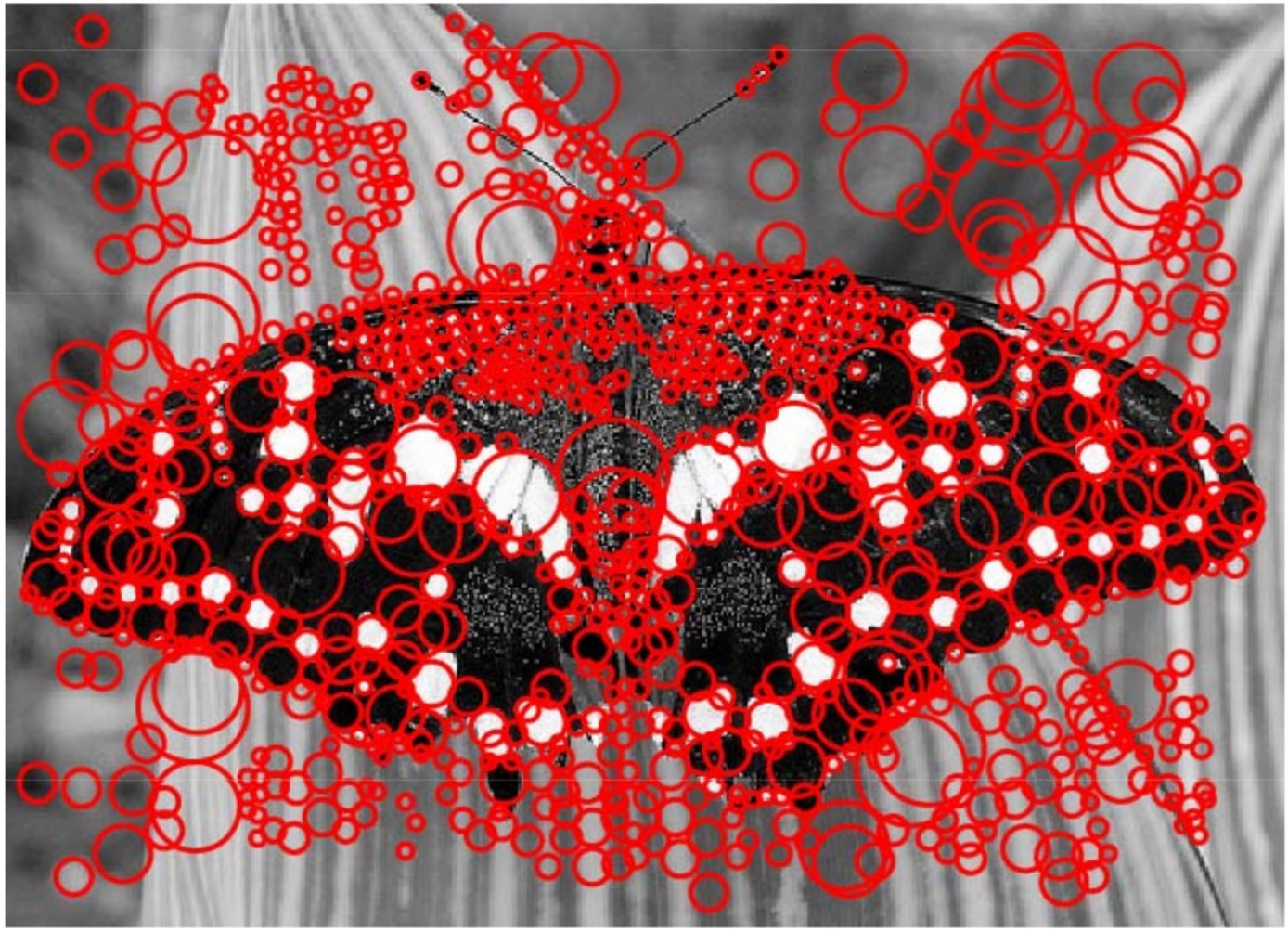
sigma = 11.9912

Example of scale-aware blob detection

sigma = 11.9912



Example of scale-aware blob detection



Taking the LoG many times over can be expensive.

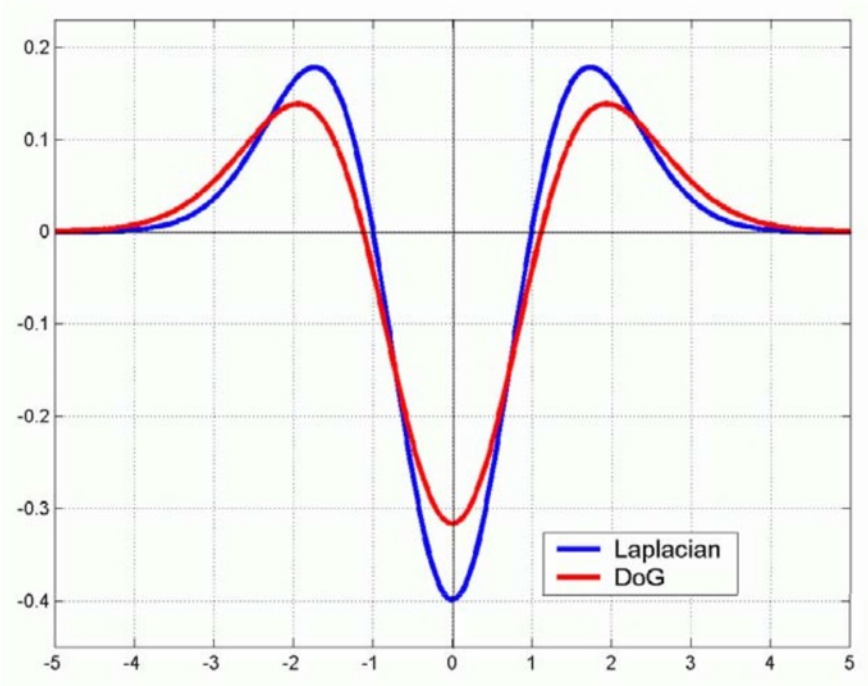
Difference of Gaussians can be used instead of LoG can be more efficient to evaluate

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

(Laplacian)

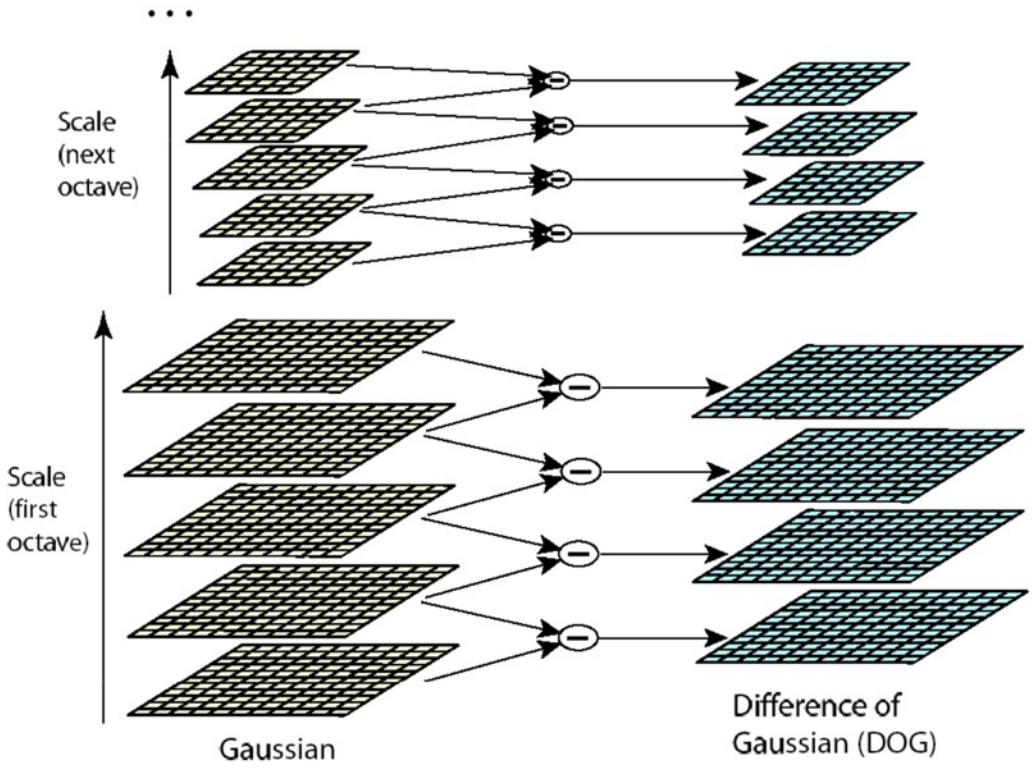
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



The SIFT detector (*very famous and influential*) relies on Difference of Gaussians

Difference of Gaussians can be used instead of LoG can be more efficient to evaluate



Breakout Session: Automatic Scale Selection