



COMP3005: Computer Vision

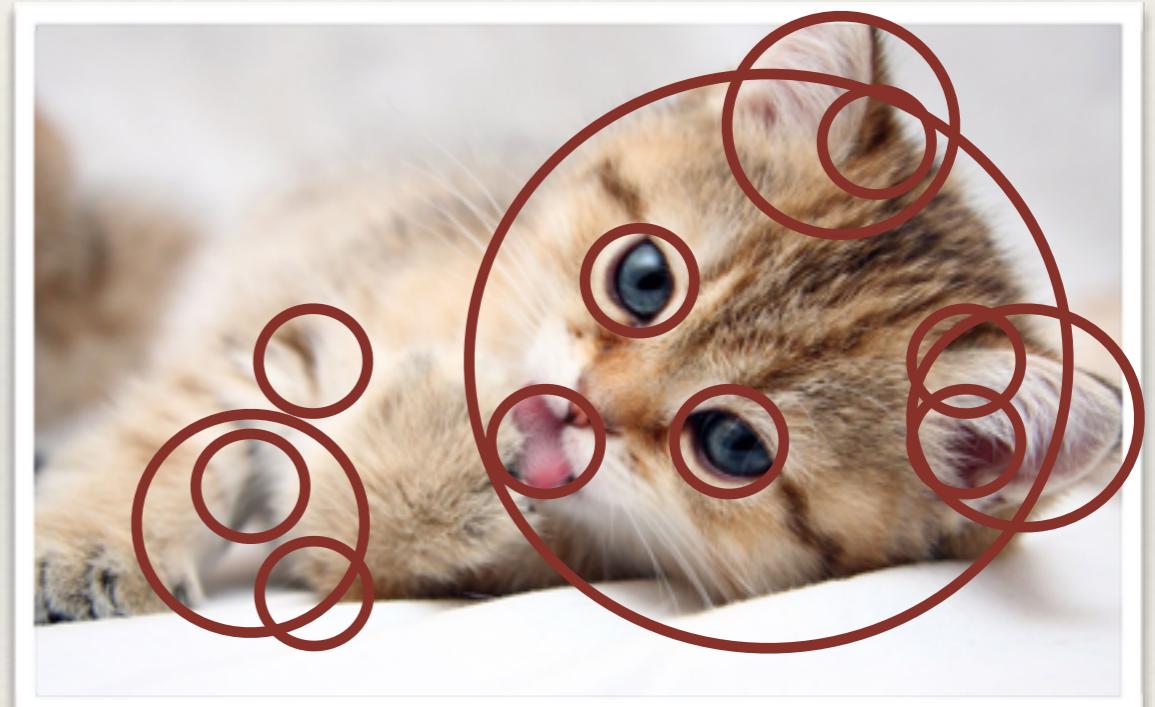
Local interest points

Jonathon Hare
jsh2@ecs.soton.ac.uk

- ❖ Finding stable (repeatable) interest points is a key problem in modern computer vision
 - ❖ Applications in areas such as tracking, matching, image alignment, making robust features for classification and search, robot navigation, 3d reconstruction, ...
 - ❖ We'll look at some of these in more detail in future lectures

What makes a good interest point?

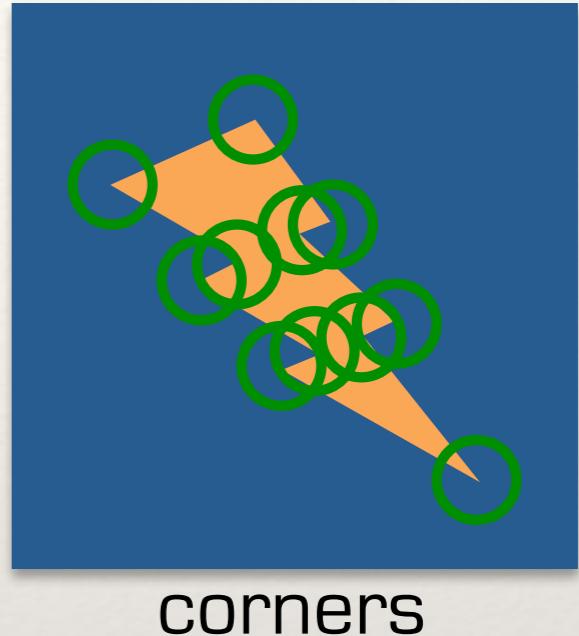
- ❖ Invariance to brightness change (local changes as well as global ones)
- ❖ Sufficient texture variation in the local neighbourhood
 - ❖ But not too much!
- ❖ Invariance to changes between the angle/position of the scene to the camera



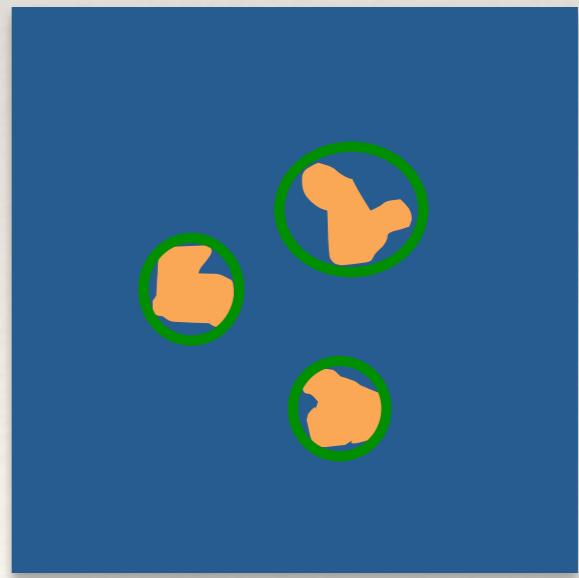
Demo: Stable local interest points

So, how do we find them?

- ❖ Lots of different types of interest point types to choose from.
 - ❖ We'll focus on two specific types and look in detail at common detection algorithms:
 - ❖ Corner detection - *Harris and Stephens*
 - ❖ Blob Detection - *Difference-of-Gaussian Extrema*

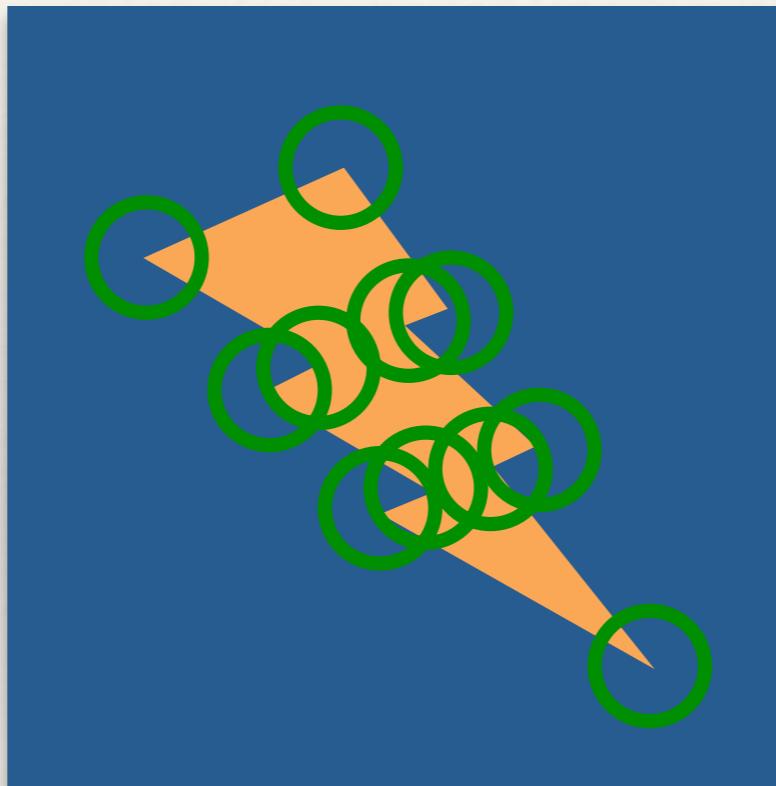


corners



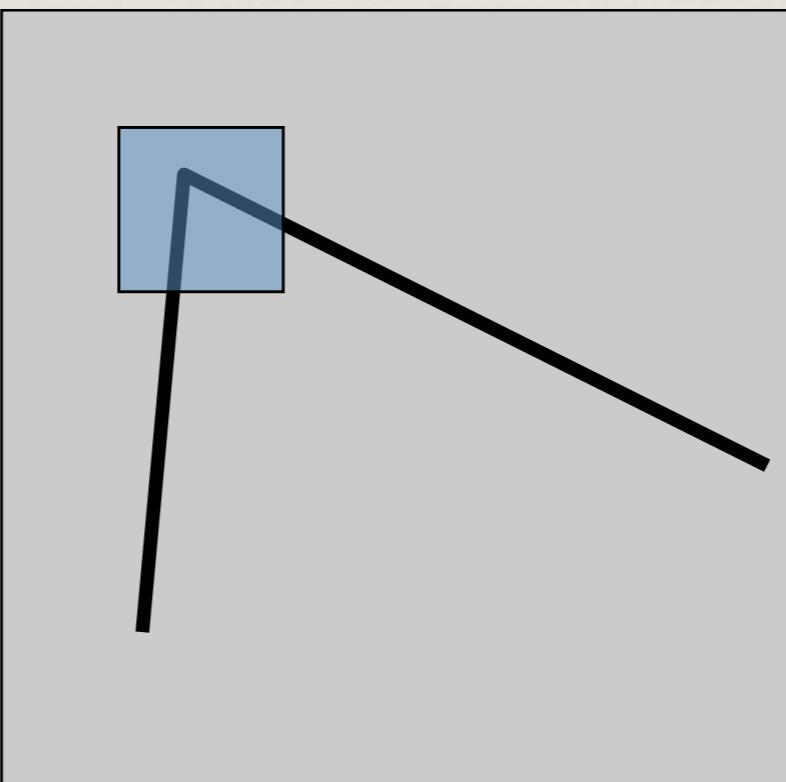
blobs

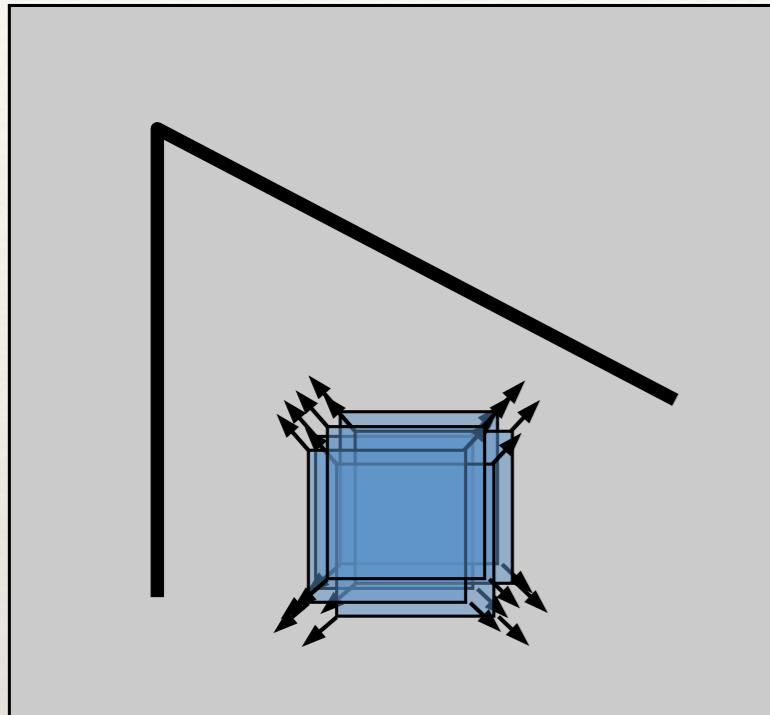
The Harris and Stephens corner detector



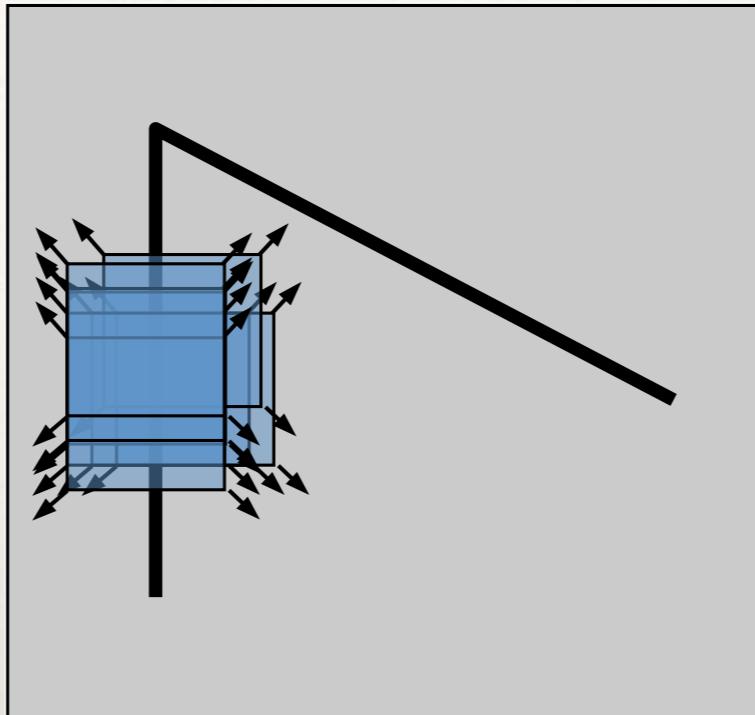
Basic Idea

- ❖ Search for corners by looking through a small window.
- ❖ Shifting that window by a small amount in *any direction* should give a *large change* in intensity

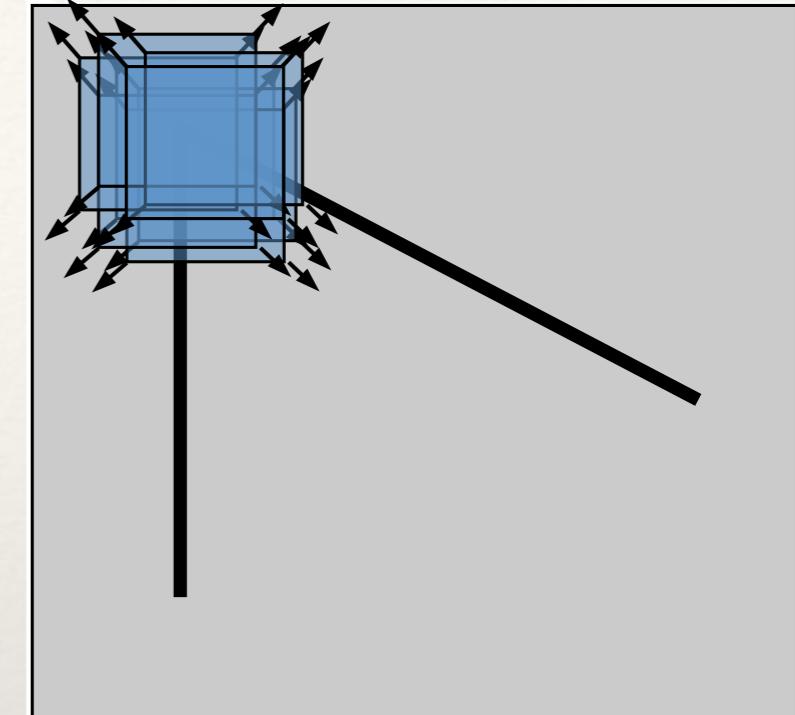




“flat” region: no
change in all
directions



“edge”:
no change along
the edge direction



“corner”:
significant change
in all directions

Harris & Stephens: Mathematics

Weighted average change in intensity between a window and a shifted version [by $(\Delta x, \Delta y)$] of that window:

$$E(x, y) = \sum_w f(x_i, y_i)[I(x_i, y_i) - I(x_i + \Delta x, y_i + \Delta y)]^2$$

weighting function intensity in window intensity in shifted window

The diagram shows two side-by-side plots. The left plot, labeled 'flat', features a horizontal red rectangle representing a uniform weighting function over a rectangular window defined by dashed lines. The right plot, labeled 'Gaussian', features a red bell-shaped curve representing a Gaussian weighting function, also centered over a rectangular window defined by dashed lines.

Harris & Stephens: Mathematics

- ❖ The Taylor expansion allows us to approximate the shifted intensity.
- ❖ Taking the first order terms we get this:

$$I(x_i + \Delta x, y_i + \Delta y) \approx I(x_i, y_i) + [I_x(x_i, y_i) \quad I_y(x_i, y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$


(partial derivatives of the image)

Harris & Stephens: Mathematics

- ❖ Substituting and simplifying gives:

$$\begin{aligned} E(x, y) &= \sum_w [I(x_i, y_i) - I(x_i + \Delta x, y_i + \Delta y)]^2 \\ &= \sum_w \left(I(x_i, y_i) - I(x_i, y_i) - [I_x(x_i, y_i) \quad I_y(x_i, y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 \\ &= \sum_w \left(-[I_x(x_i, y_i) \quad I_y(x_i, y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 \\ &= \sum_w \left([I_x(x_i, y_i) \quad I_y(x_i, y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 \\ &= [\Delta x \quad \Delta y] \begin{bmatrix} \sum_w (I_x(x_i, y_i))^2 & \sum_w I_x(x_i, y_i) I_y(x_i, y_i) \\ \sum_w I_x(x_i, y_i) I_y(x_i, y_i) & \sum_w (I_y(x_i, y_i))^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\ &= [\Delta x \quad \Delta y] \mathbf{M} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \end{aligned}$$

Structure Tensor

The **square symmetric** matrix \mathbf{M} is called the *Structure Tensor* or the *Second Moment matrix*

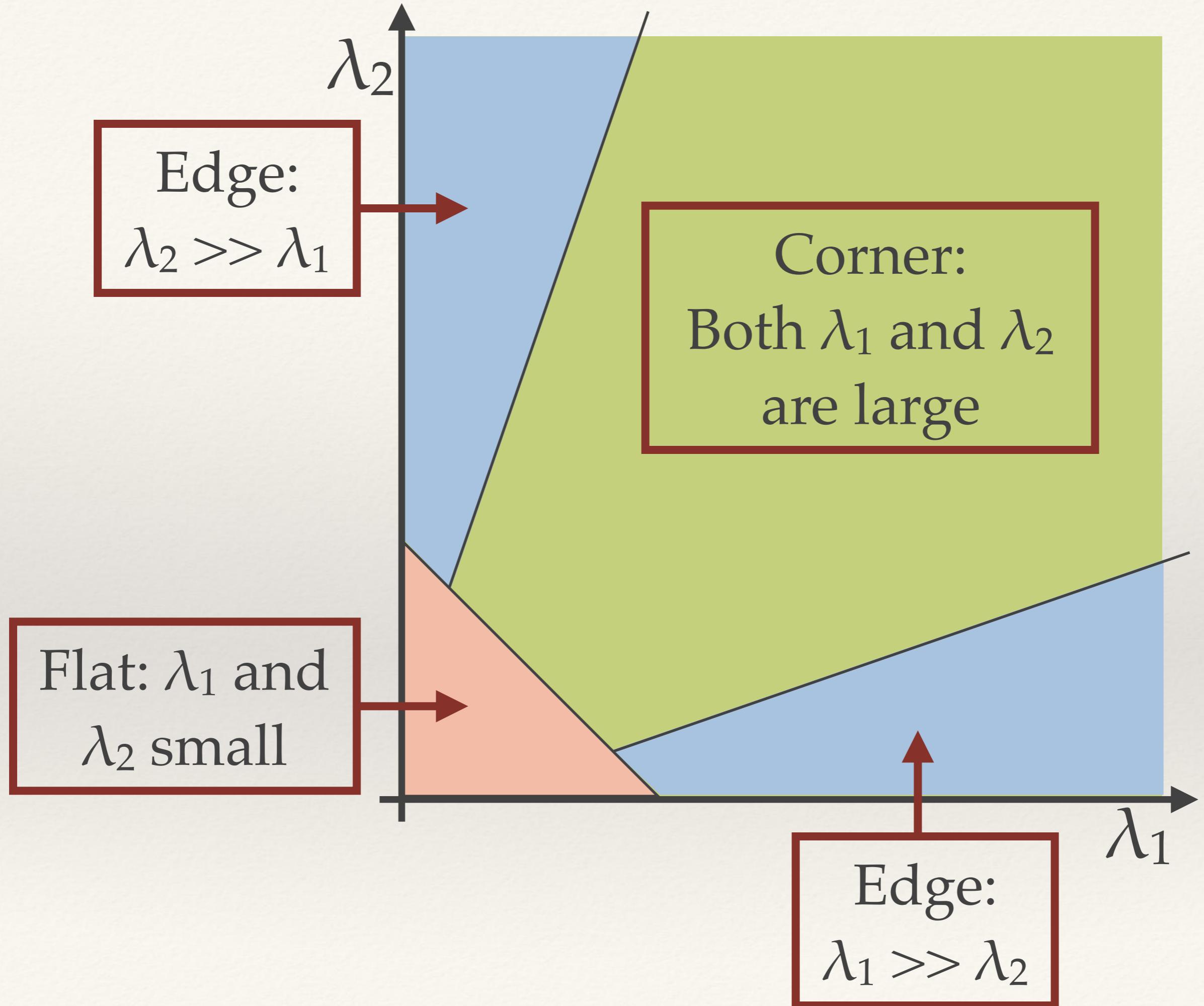
$$\mathbf{M} = \begin{bmatrix} \sum_w (I_x(x_i, y_i))^2 & \sum_w I_x(x_i, y_i) I_y(x_i, y_i) \\ \sum_w I_x(x_i, y_i) I_y(x_i, y_i) & \sum_w (I_y(x_i, y_i))^2 \end{bmatrix}$$

It concisely encodes the how the local shape intensity function of the window changes with small shifts

Eigenvalues of the Structure Tensor

- ❖ Think back to Lecture 3 where we looked at covariance matrices...
 - ❖ As with the 2d covariance matrix, the structure tensor describes an ellipse: $x^T M x = c$ (this is a *quadratic form*)
 - ❖ The eigenvalues and vectors tell us the rates of change and their respective directions

Demo: Structure Tensor eigenvalues



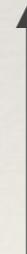
Harris & Stephens Response Function

- ❖ Rather than compute the eigenvalues directly, Harris and Stephens defined a corner response function in terms of the determinant and trace of \mathbf{M} :

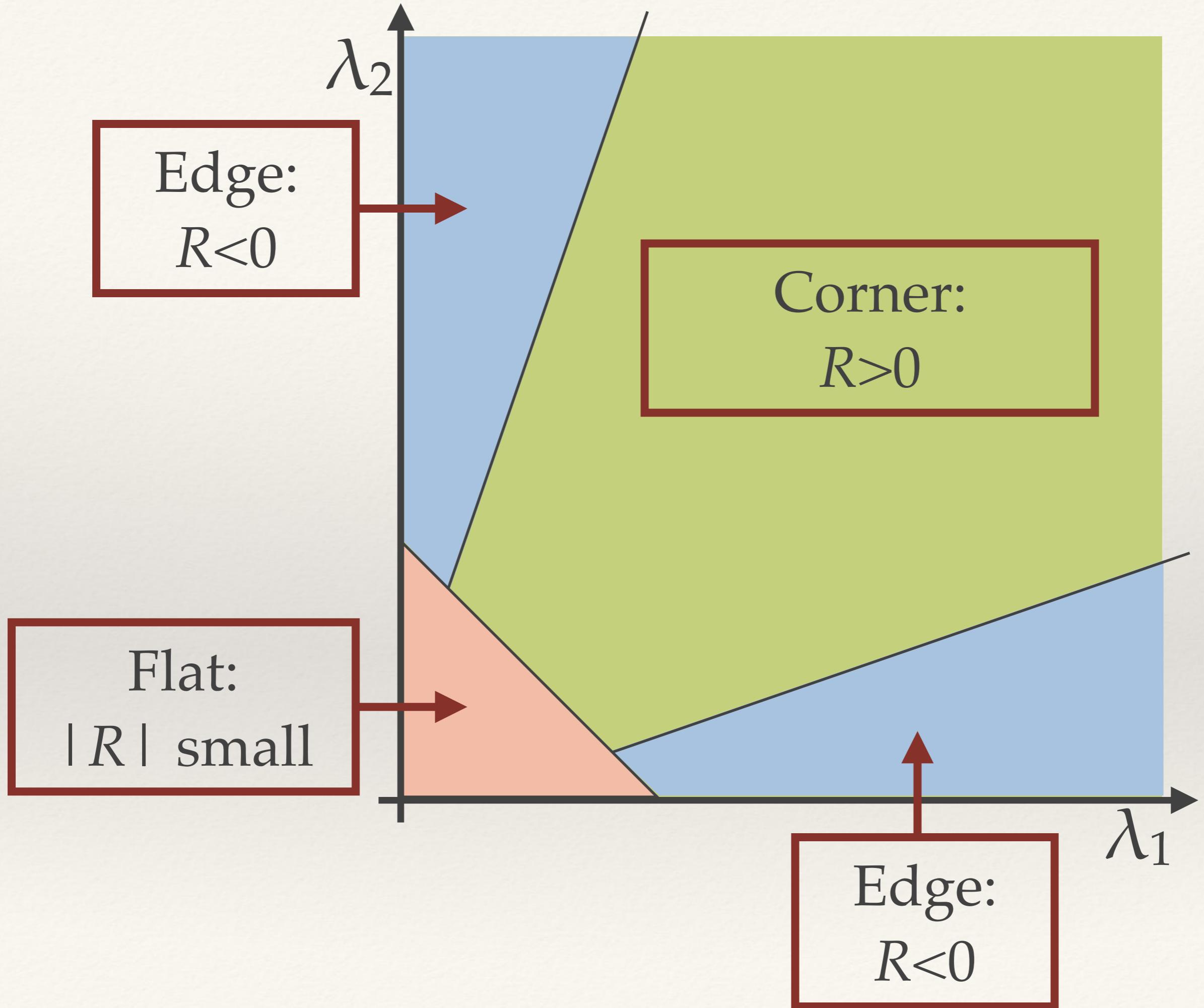
$$\det(\mathbf{M}) = M_{00}M_{11} - M_{01}M_{10} = M_{00}M_{11} - M_{10}^2 = \lambda_1\lambda_2$$

$$\text{trace}(\mathbf{M}) = M_{00} + M_{11} = \lambda_1 + \lambda_2$$

$$R = \det(\mathbf{M}) - k \text{ trace}(\mathbf{M})^2$$



k is a small empirically set
constant (usually 0.04 - 0.06)



Demo: Harris & Stephens Response

Harris & Stephens Detector

- ❖ Simple algorithm:
 - ❖ Take all points with the response value above a threshold
 - ❖ Keep only the points that are local maxima (i.e. where the current response is bigger than the 8 neighbouring pixels)

*Demo: Thresholded Harris &
Stephens Response*

*Demo: Thresholded Harris &
Stephens Points*

Scale in Computer Vision

The problem of scale

- ❖ As an object moves closer to the camera it get larger with more detail... as it moves further away it gets smaller and loses detail...
- ❖ If you're using a technique that uses a fixed size processing window (e.g. Harris corners, or indeed anything that involves a fixed kernel) then this is a bit of a problem!

Scale space theory

- ❖ Scale space theory is a formal framework for handling the scale problem.
 - ❖ Represents the image by a series of increasingly smoothed/blurred images parameterised by a scale parameter t .
 - ❖ t represents the amount of smoothing.
 - ❖ **Key notion:** Image structures smaller than $\text{sqrt}(t)$ have been smoothed away at scale t .

The Gaussian Scale Space

- ❖ Many possible types of scale space are possible (depending on the smoothing function), but only the Gaussian function has the desired properties for image representation.
- ❖ These provable properties are called the “*scale space axioms*”.

Gaussian Scale Space

Formally, Gaussian scale space defined as:

$$L(\cdot, \cdot; t) = g(\cdot, \cdot; t) * f(\cdot, \cdot)$$

Note: convolution is over x, y for fixed t

where $t \geq 0$ and,

$$g(x, y; t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/2t}$$

Note: $t = \sigma^2 = \text{variance of the Gaussian}$

Normally, only a fixed set of values of t are used - it's common to use integer powers of 2 or $\sqrt{2}$



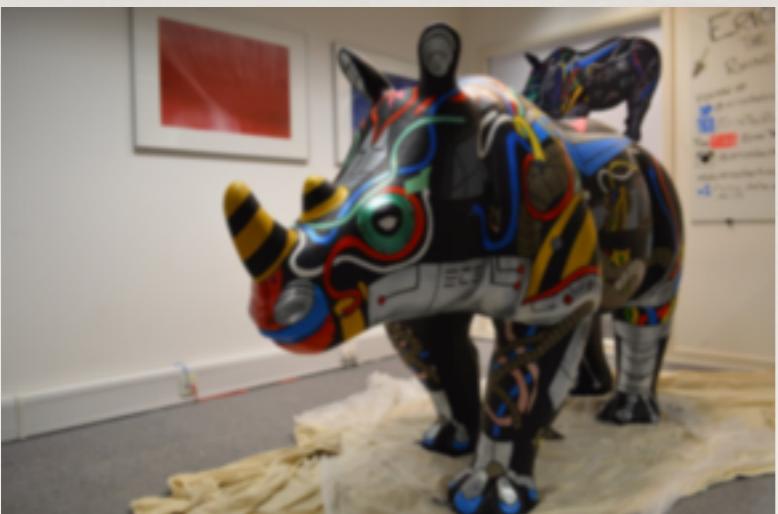
$t=0$



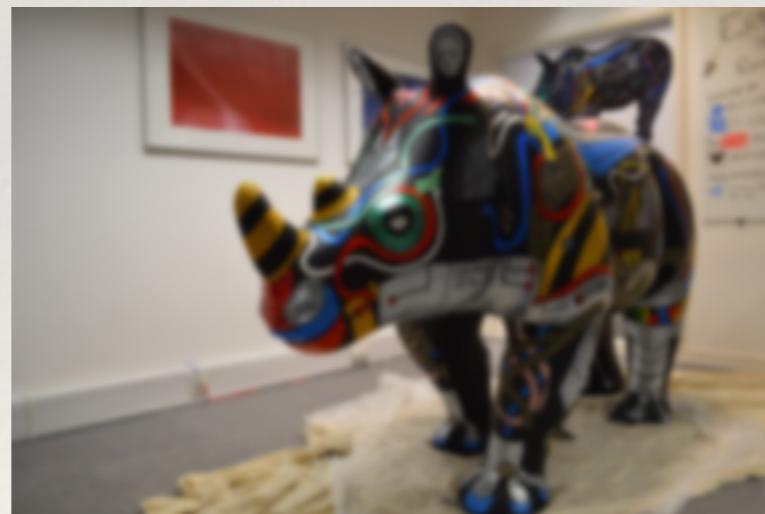
$t=1$



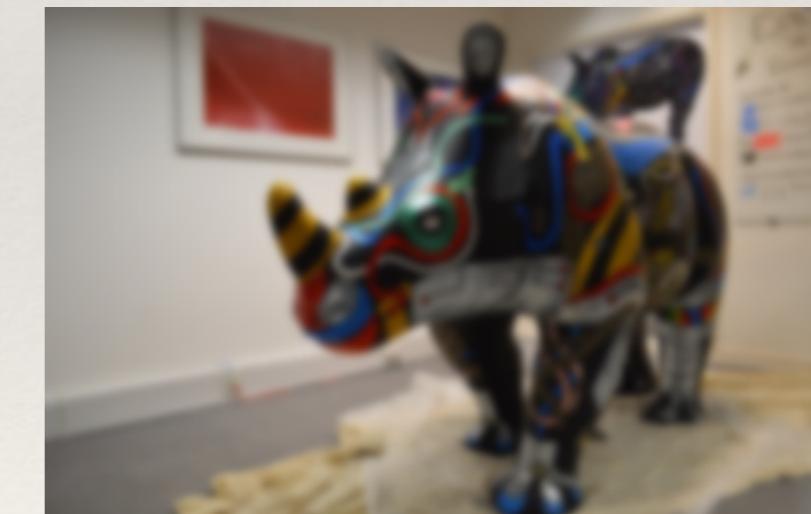
$t=2$



$t=4$



$t=16$



$t=32$

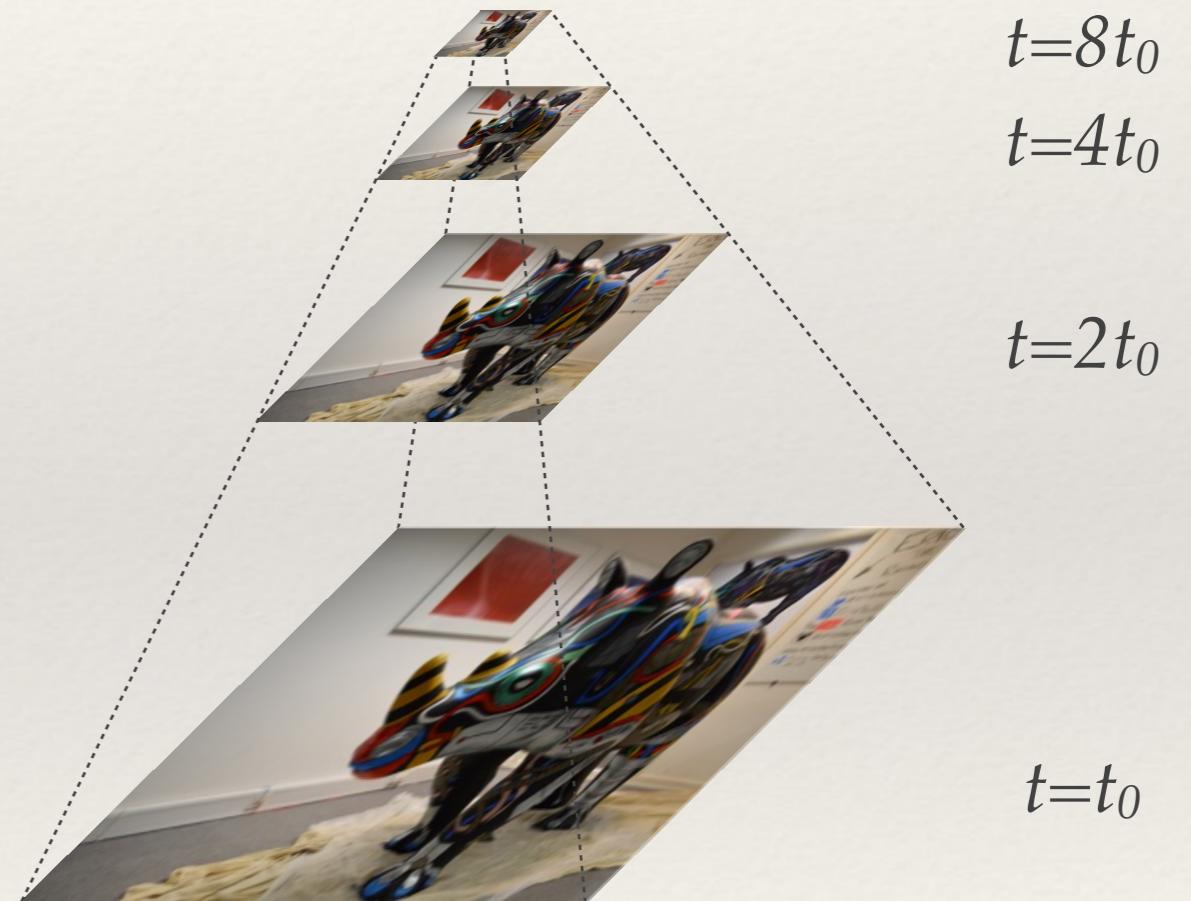
Nyquist-Shannon Sampling theorem

If a function $x(t)$ contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced $1/(2B)$ seconds apart.

...so, if you half the filter the signal with a low-pass filter that halves the frequency content, you can also half the sampling rate without loss of information...

Gaussian Pyramid

- ❖ Every time you double t in scale space, you can half the image size without loss of information!
- ❖ Leads to a much more efficient representation
 - ❖ faster processing
 - ❖ less memory



Multi-scale Harris & Stephens

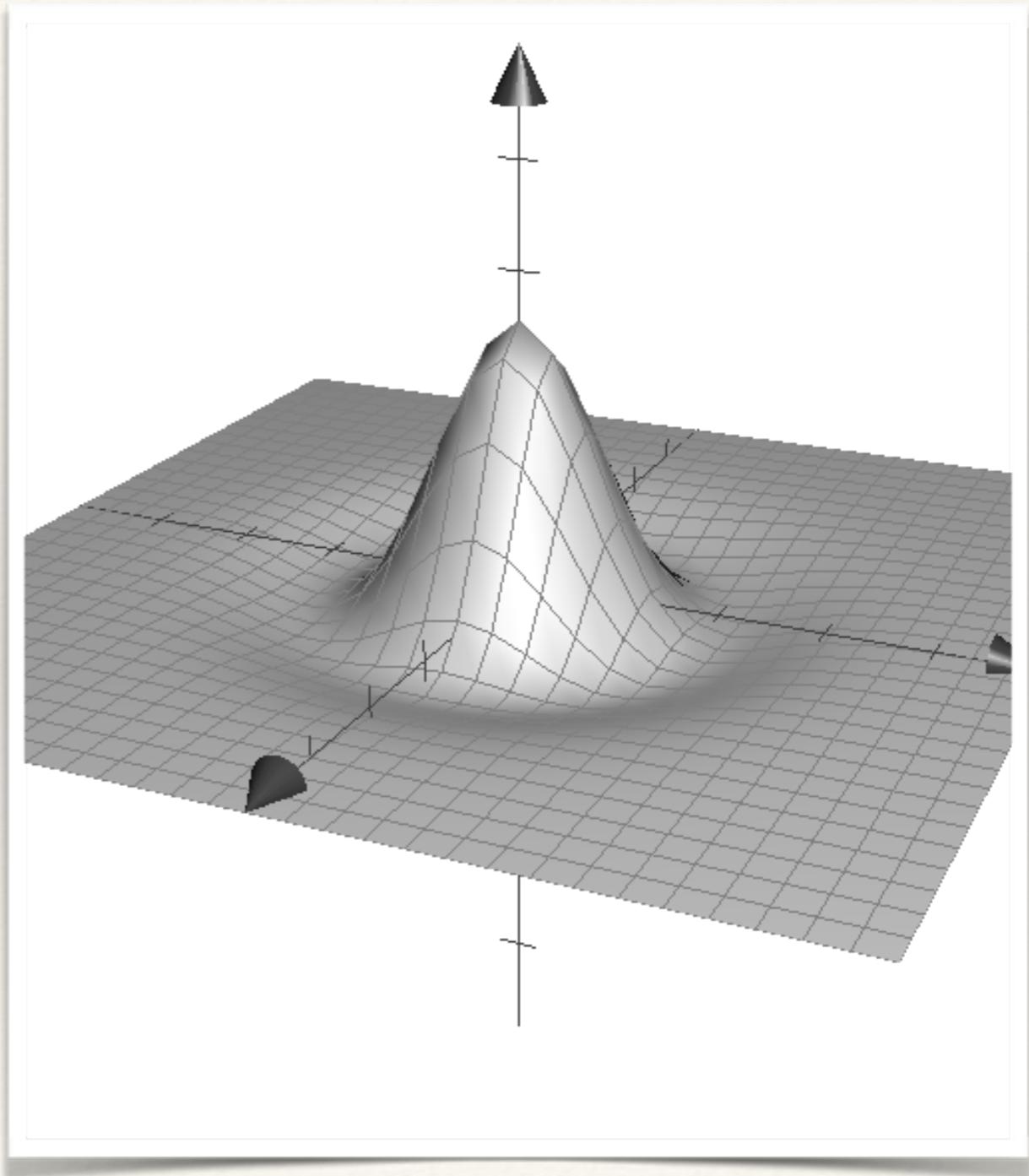
- ❖ Extending the Harris and Stephens detector to work across scales is easy...
- ❖ We define a Gaussian scale space with a fixed set of scales and compute the corner response function at every pixel of each scale and keep only those with a response above a certain threshold.

Demo: Multi-scale Harris & Stephens

Blob Detection

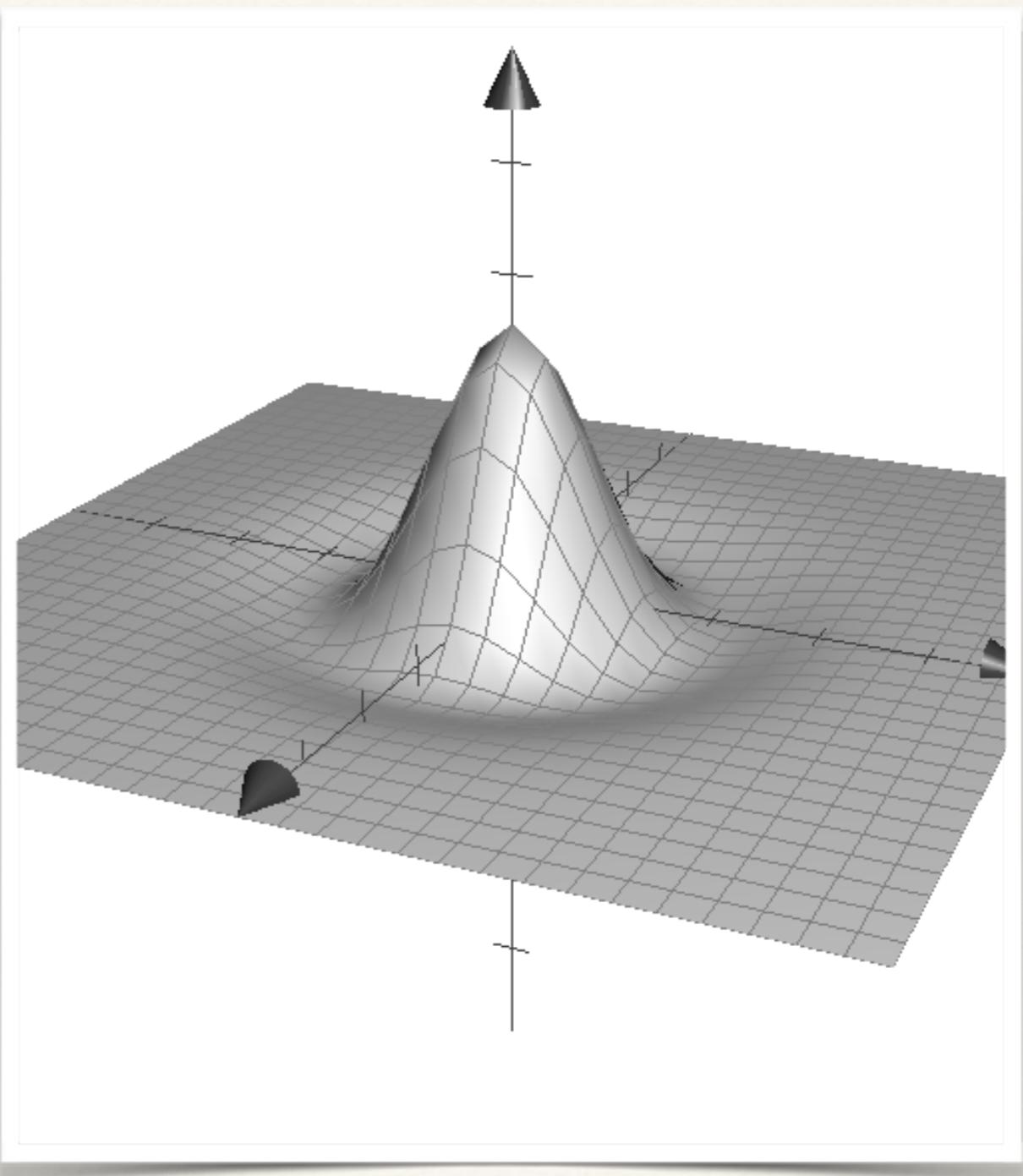
Recap: Laplacian of Gaussian

- ❖ Recall that the LoG is the second derivative of a Gaussian
 - ❖ Used in the Marr-Hildreth edge detector
 - ❖ Zero crossings of LoG convolution



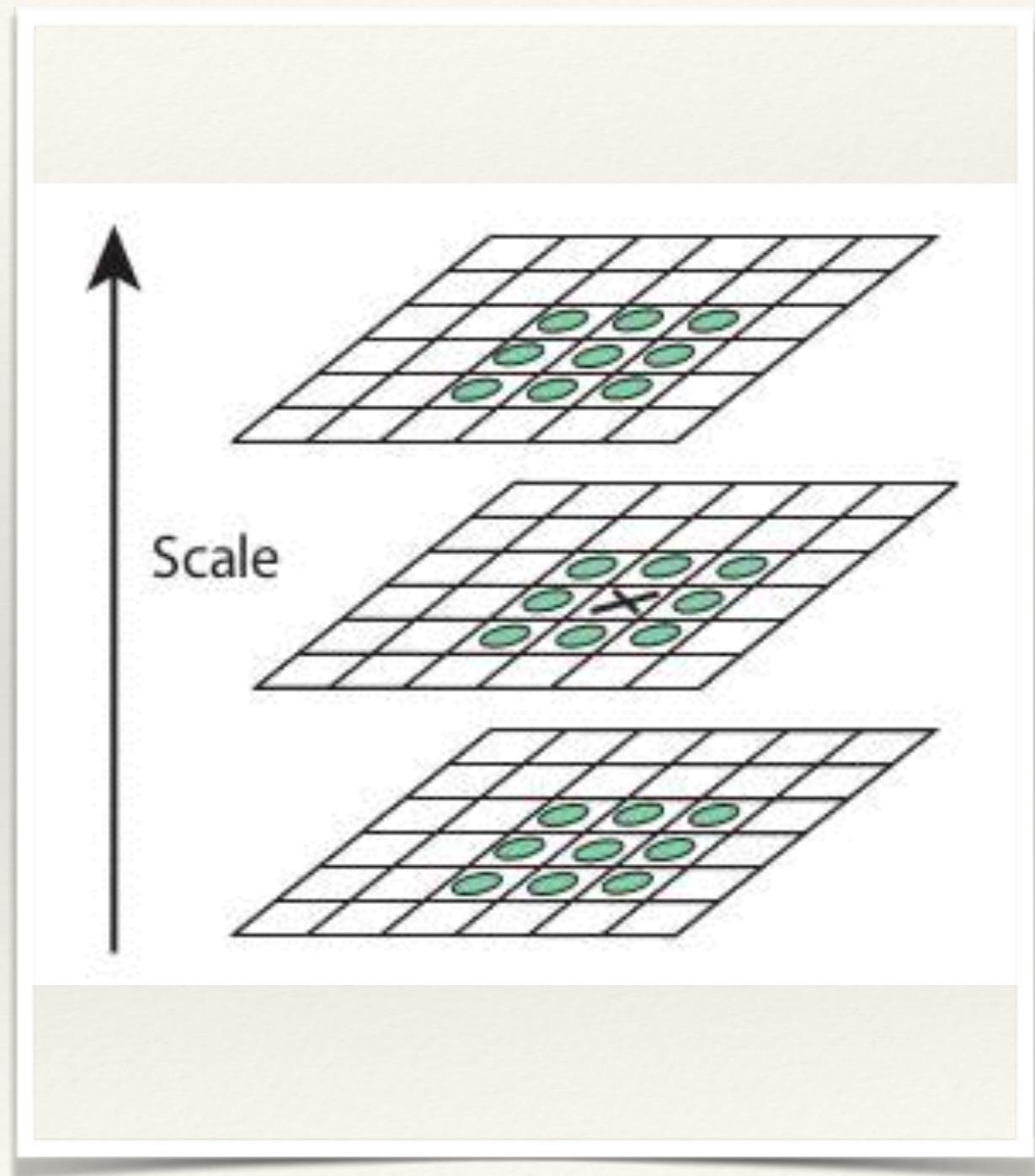
Laplacian of Gaussian

- ❖ By finding local minima or maxima, you get a blob detector!



Scale space LoG

- ❖ Normalised scale space LoG defined as:
$$\nabla_{norm}^2 L(x, y; t) = t(L_{xx} + L_{yy})$$
- ❖ By finding extrema of this function in scale space, you can find *blobs* at their representative scale ($\sim \text{sqrt}(2t)$)
 - ❖ Just need to look at the neighbouring pixels!



Very useful property: if a blob is detected at $(x_0, y_0; t_0)$ in an image, then under a scaling of that image by a factor s , the same blob would be detected at $(sx_0, sy_0; s^2t_0)$ in the scaled image.

Scale space DoG

- ❖ In practice it's computationally expensive to build a LoG scale space.

- ❖ But, the following approximation can be made:

$$\nabla_{norm}^2 L(x, y; t) \approx \frac{t}{\Delta t} (L(x, y; t + \Delta t) - L(x, y; t - \Delta t))$$

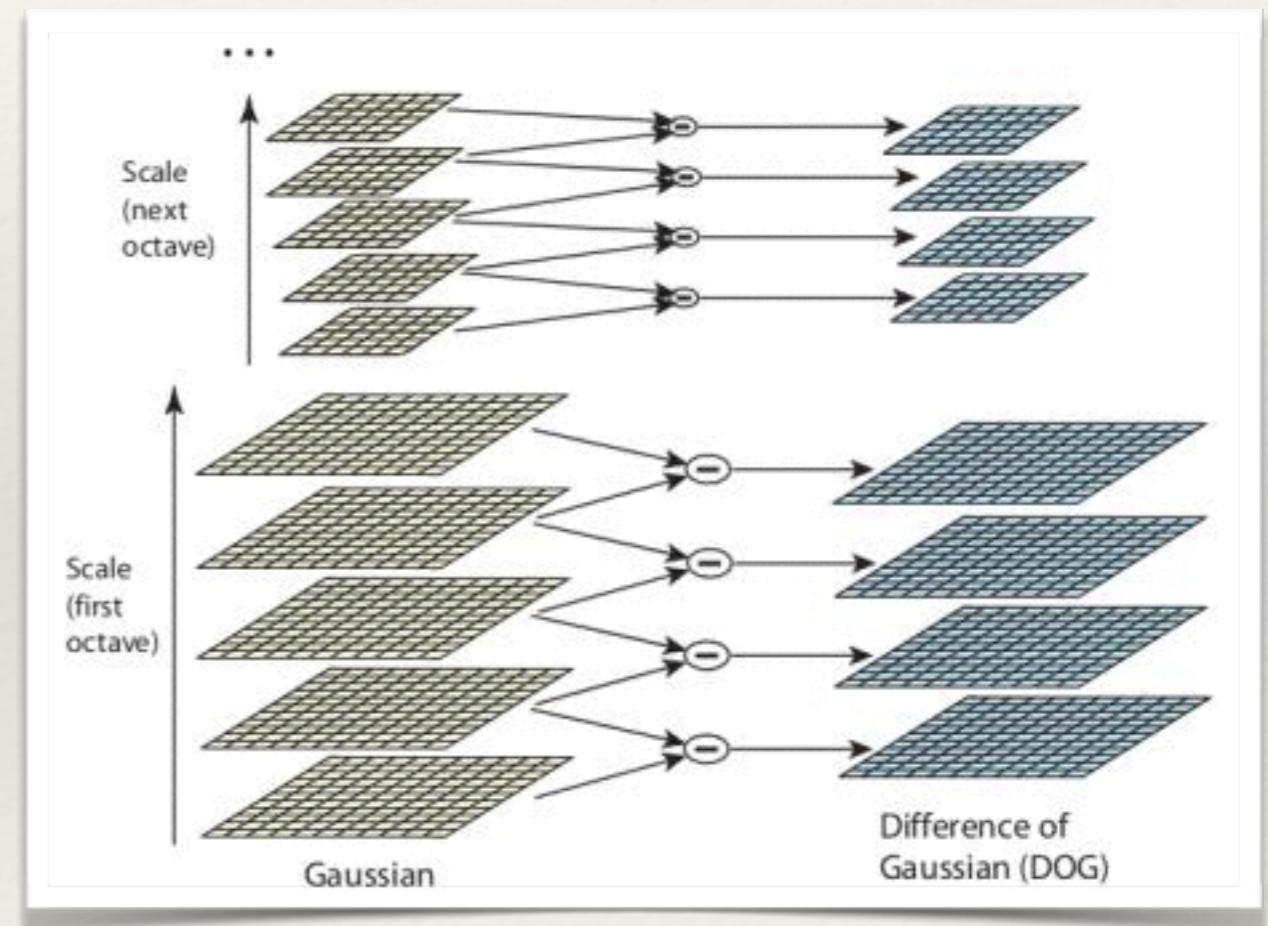
- ❖ This is called a Difference-of-Gaussians (*DoG*)

- ❖ Implies that the LoG scale space can be built from subtracting adjacent scales of a Gaussian scale space

*Demo: Difference of Gaussian
Response*

DoG Pyramid

- ❖ Of course, for efficiency you can also build a DoG pyramid
 - ❖ an *oversampled* pyramid as there are multiple images between a doubling of scale.
 - ❖ Images between a doubling of scale are an *octave*.



*Demo: Multi-scale Difference of
Gaussian Blob detector*

Summary

- ❖ Interest points have loads of applications in computer vision.
 - ❖ They need to be robustly detected, and invariant to rotation, lighting change, etc.
- ❖ We've looked at two types: corners and blobs
 - ❖ Harris & Stephens is a common corner detector
 - ❖ Finding extrema in a multi scale DoG pyramid provides a robust blob detector
- ❖ Scale space theory allows us to find features (corners and blobs) of different sizes