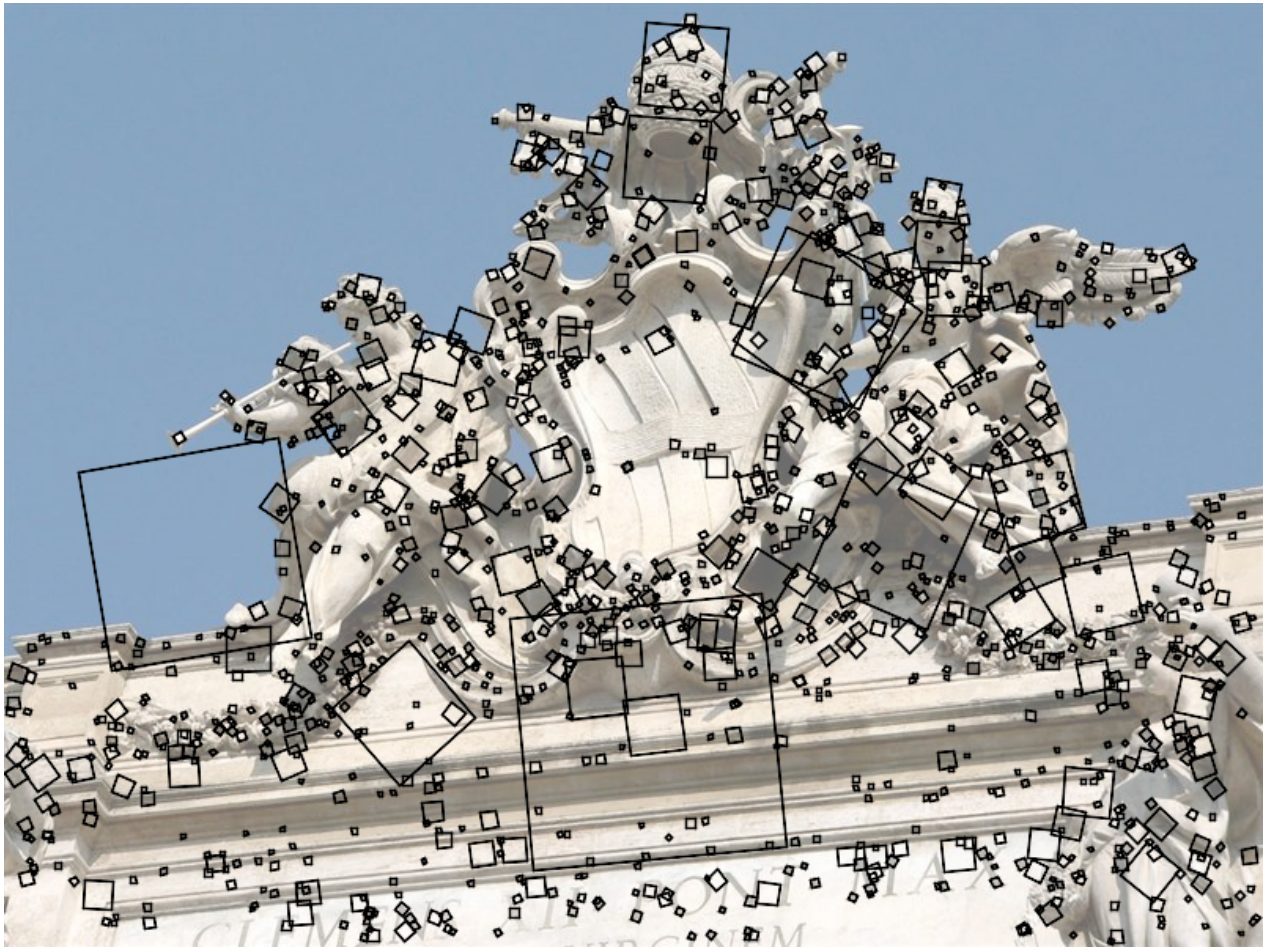
Correspondence



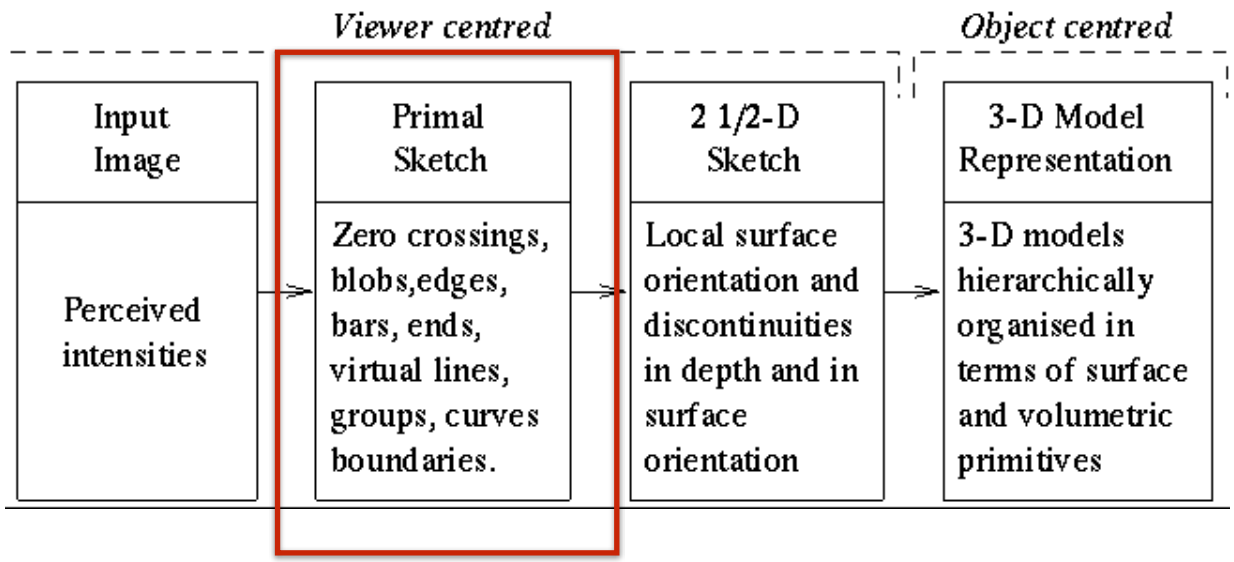
Reminder to post lecture slides

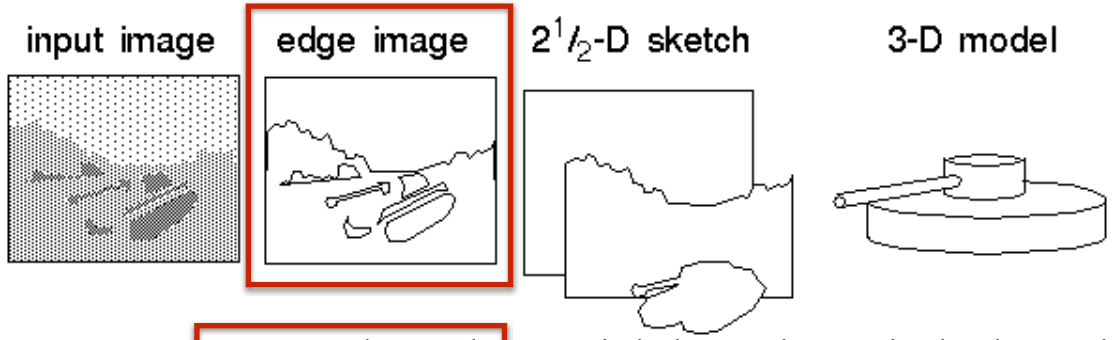
This session will be recorded for educational use by other students in this course

Logistics

- HW1 has been released
- This class can help with question 2.1 Feature Detection and Matching and 2.2.3 RANSAC
- Potentially helpful resource:
  - Homography: https://towardsdatascience.com/estimating-a-homography-matrix-522c70ec4b2c

David Marr's Taxonomy of Vision

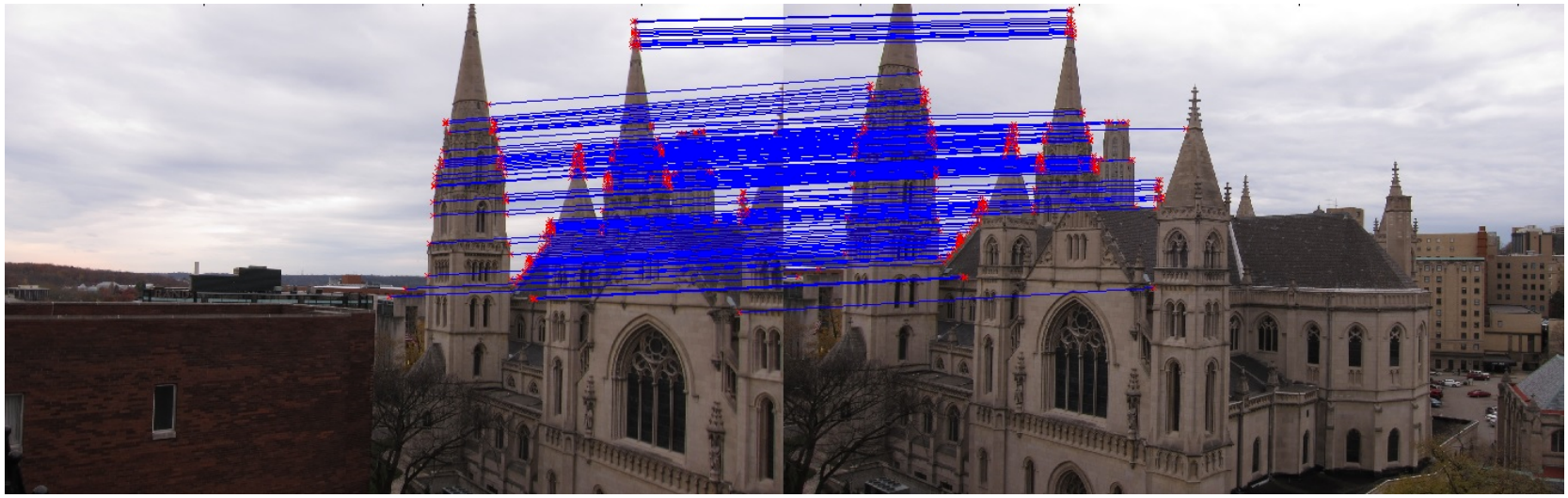


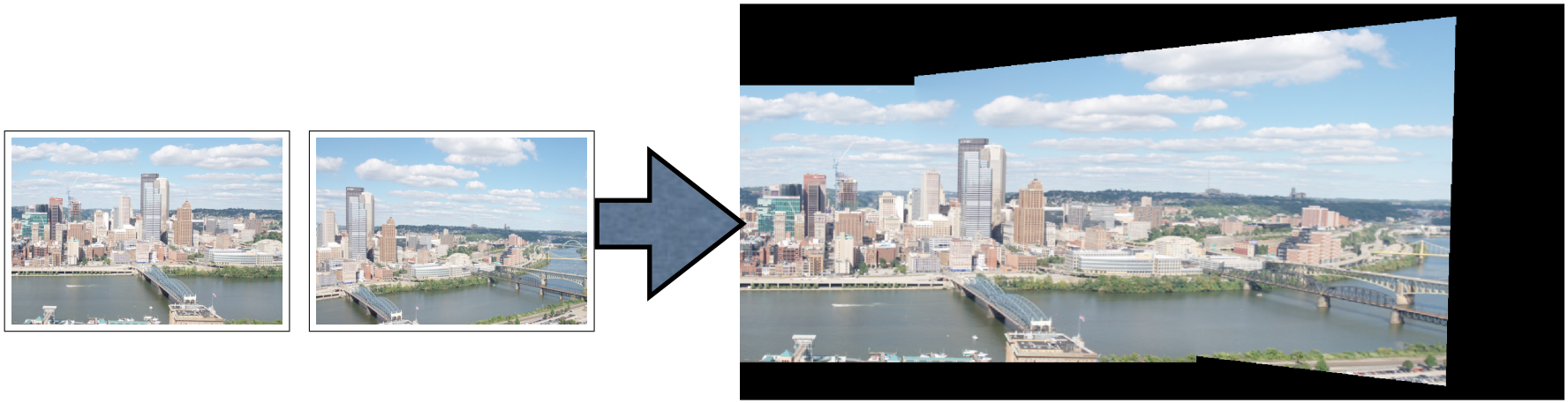


Outline

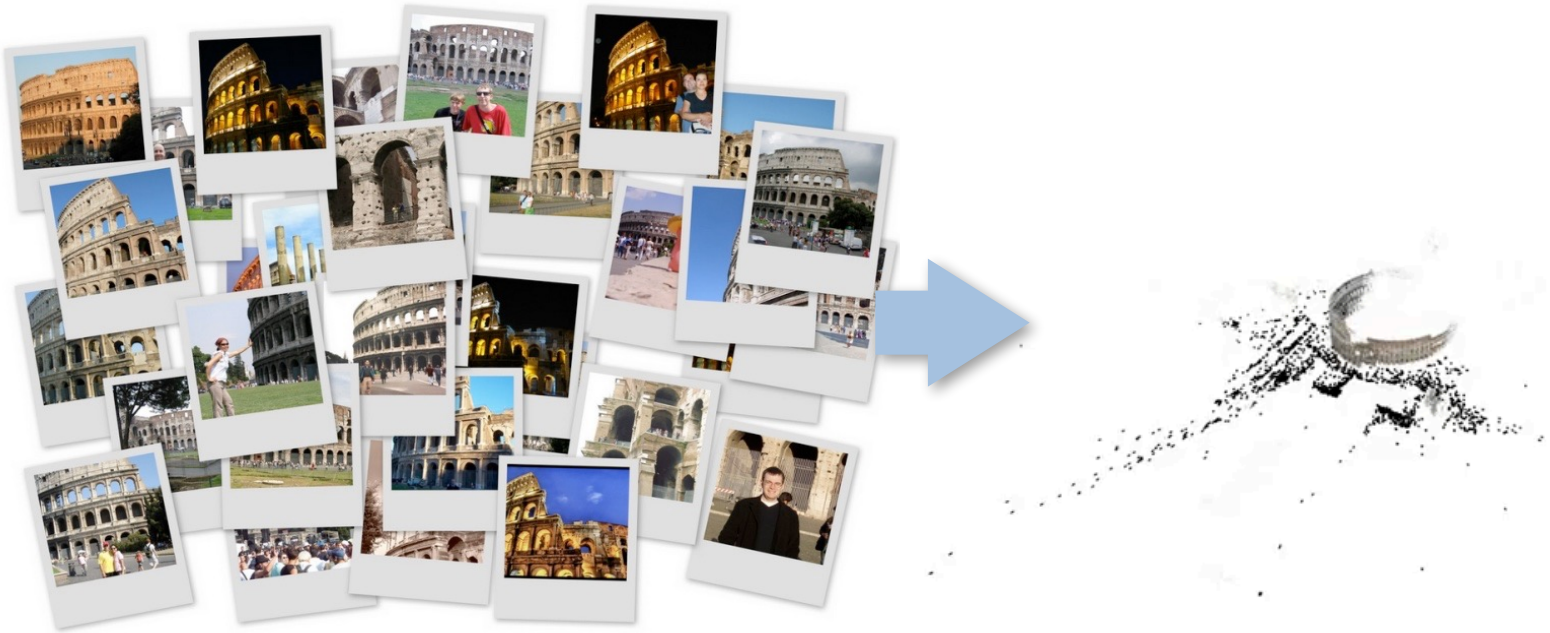
- Interest point detection (Harris Corners)
- Descriptors (SIFT, BRIEF, Filter Banks)
- RANSAC

Core visual understanding task: finding correspondences between images



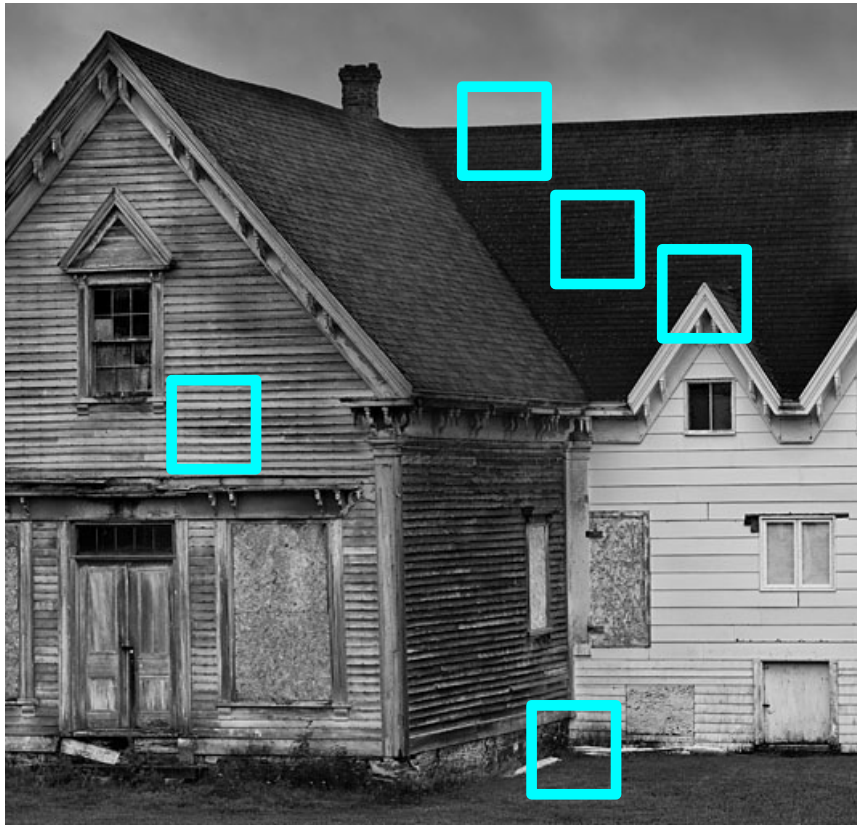


Example: image matching of landmarks



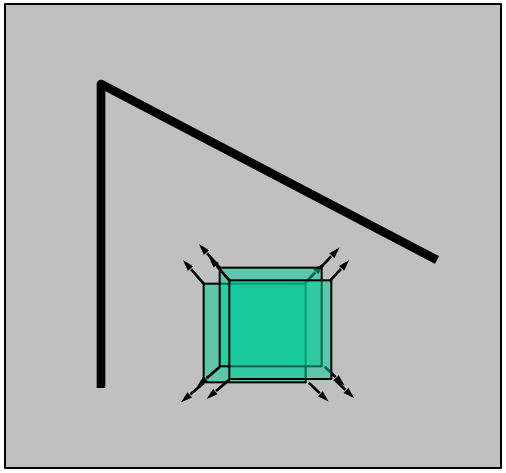
Correspondence + geometry estimation

Which of these patches are easier to match (and why)?

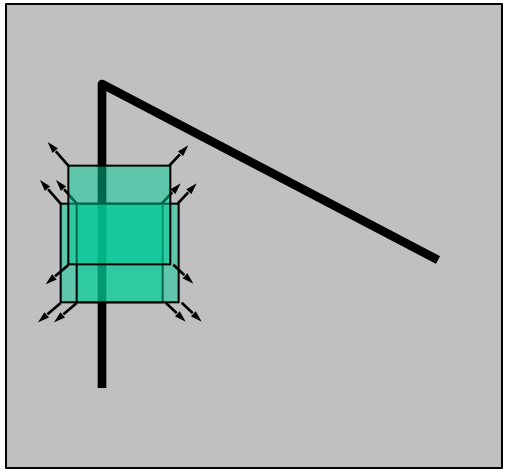


How can we mathematically operationalize this?

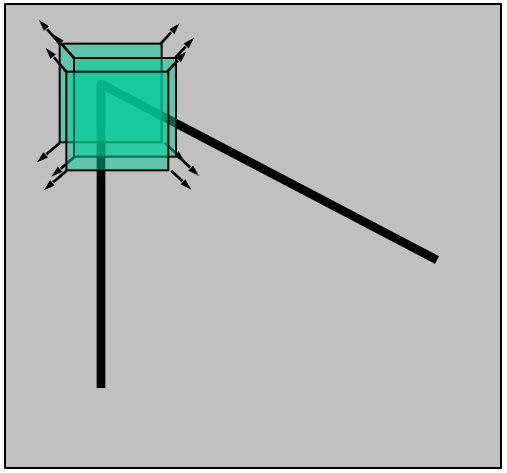
Easy to Match = Corner



"flat" region:
no change in any direction



"edge":
no change along the edge direction

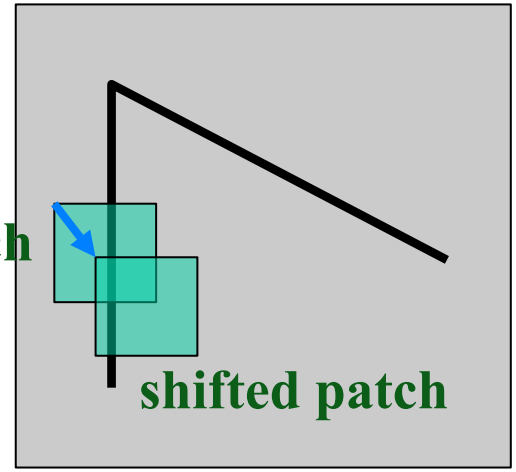


"corner":
significant change in all directions

Definition: a patch is "easy-to-match" if small shifts always produce a large difference from the original patch (for example, a large SSD error)

Mathematical Formulation

Definition: a patch is "easy-to-match" if small shifts always produce a large difference from the original patch (for example, a large SSD error)



why is this a min and not a max?

$$\text{cornerness}(x_0, y_0) = \min_{u,v} E_{x_0,y_0}(u, v)$$

$u^2 + v^2 = 1$

shift parameters

$$E_{x_0,y_0}(u, v) = \sum_{(x,y) \in W(x_0,y_0)} [I(x + u, y + v) - I(x, y)]^2$$

pixels in a window neighborhood of (x_0, y_0)

pixel value in shifted patch

pixel value in original patch

Wait - this isn't quite right. What value of (u, v) will minimize this?

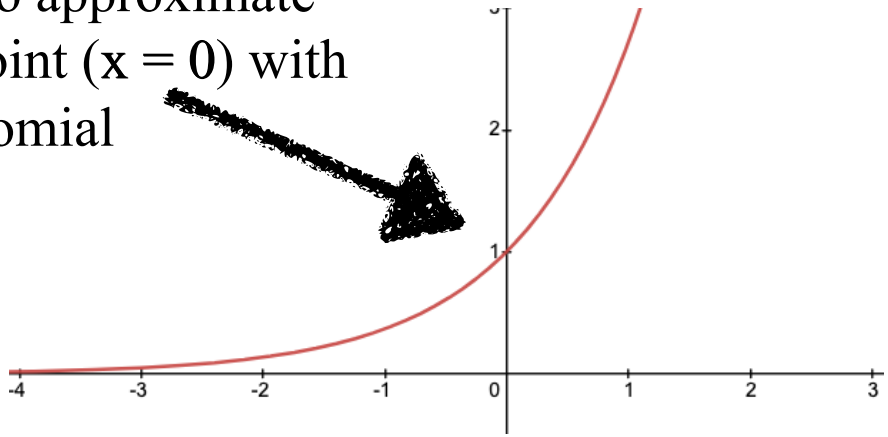
Great! But how do we solve this?

Background: Taylor Series

- How do we find a polynomial approximation to any function $f(x)$?
- $f(x)$ is not a polynomial!
- We need to pick a point a that we want to approximate around
- We can find a polynomial function that is close to $f(x)$ in the region near a

$$f(x) = e^x$$

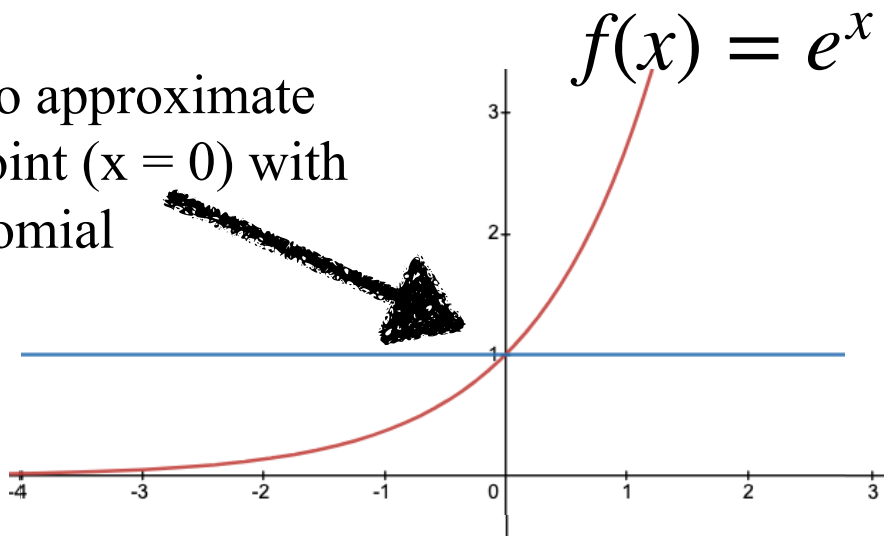
Example: Want to approximate $f(x)$ around this point (x = 0) with a polynomial



Background: Taylor Series

- How do we find a polynomial approximation to any function $f(x)$?
- The simplest approximation is a constant: $f(x) \approx e^0 = 1$

Example: Want to approximate $f(x)$ around this point (x = 0) with a polynomial



0th order approximation

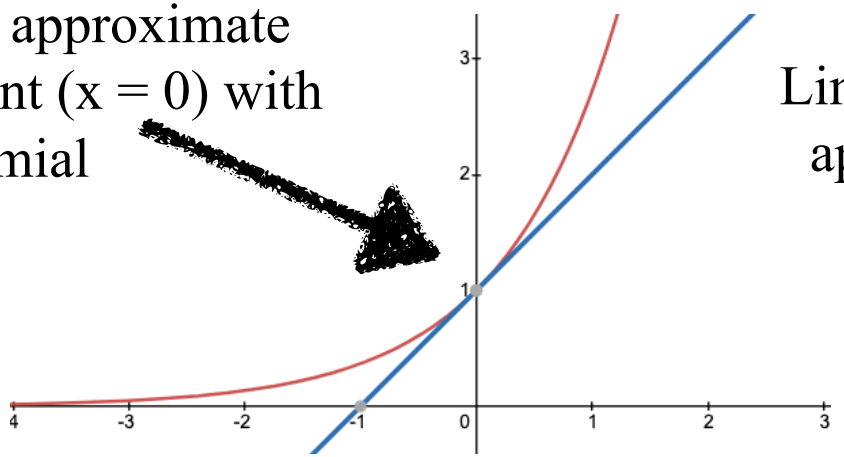
Correct at x = 0 but not very good anywhere else!

Background: Taylor Series

- Example: suppose that we have some function $f(x) = e^x$ and we want to approximate around $x = 0$
- Next best: linear approximation
  - Linear approximation: $f(x) \approx f(a) + f'(x)(x - a)$ (a=0)

$$f(x) \approx 1 + 1 \cdot (x - 0) = 1 + x$$

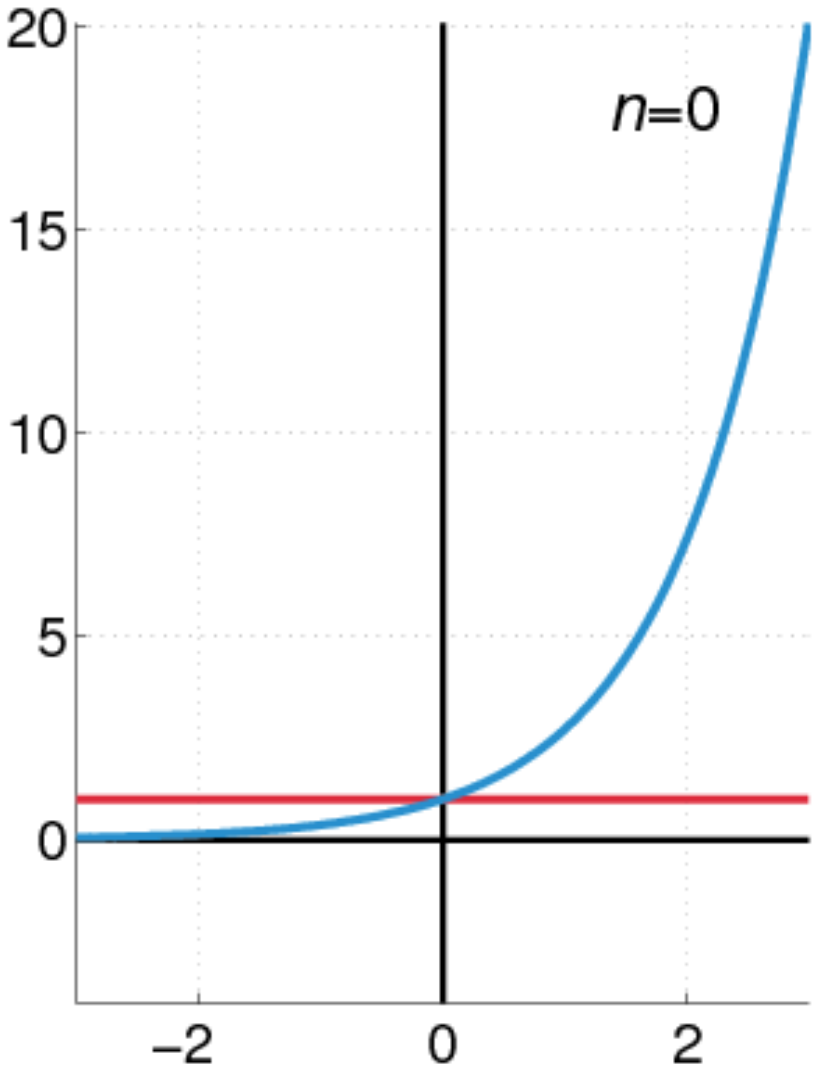
Example: Want to approximate $f(x)$ around this point (x = 0) with a polynomial



Linear / 1st order approximation

A little better - correct in the region around x = 0

Higher order approximations of $f(x) = e^x$ at $x = 0$



$f(x) = e^x$

red = nth order polynomial approximations

- Linear approximation: $f(x) \approx f(a) + f'(x)(x - a)$

In our case, the "function" is an image I

We are taking an approximation around the point (x,y)

We are shifting by (u, v) pixels

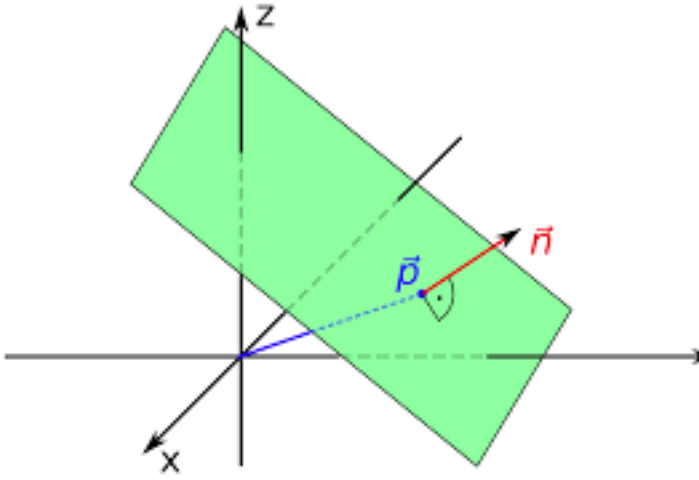
$$I(x + u, y + v) \approx ?$$

Multivariate 1st order approximation (plane):

$$I(x + u, y + v) \approx \mathbf{I} + \mathbf{I}_x u + \mathbf{I}_y v$$

where

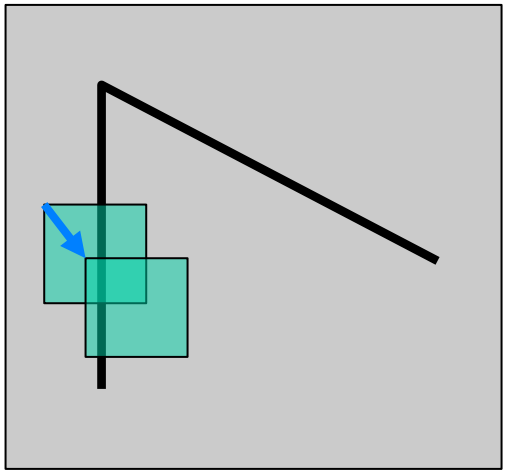
$$\mathbf{I}_x = \frac{\partial I(x, y)}{\partial x}$$



Feature detection: The Math

Consider shifting an image patch by (u,v)

- How do the pixels in the patch change?
- Compare each pixel before and after by summing up the squared differences
- This defines an "error" of E(u,v):



$$\text{Corner}(x,y) = \min_{u^2+v^2=1} E_{x_0,y_0}(u,v)$$

shifted patch original patch

$$E_{x_0,y_0}(u,v) = \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2$$

1st order approx

$$\approx \sum_{(x,y)\in W} [\mathbf{I} + \mathbf{I}_x u + \mathbf{I}_y v - \mathbf{I}]^2$$

I terms cancel

$$= \sum_{(x,y)\in W} [\mathbf{I}_x^2 u^2 + \mathbf{I}_y^2 v^2 + 2\mathbf{I}_x\mathbf{I}_y uv]$$

expand the square

$$= \begin{bmatrix} u & v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix}, \quad A = \sum_{(x,y)\in W} \begin{bmatrix} \mathbf{I}_x^2 & \mathbf{I}_x\mathbf{I}_y \\ \mathbf{I}_y\mathbf{I}_x & \mathbf{I}_y^2 \end{bmatrix}$$

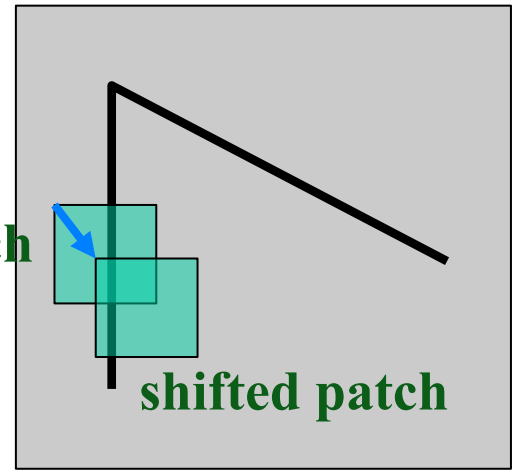
Image derivatives!

(How do we compute these?)

factor out [u,v]

Mathematical Formulation

Definition: a patch is "easy-to-match" if small shifts always produce a large difference from the original patch (for example, a large SSD error)



$$\text{cornerness}(x_0, y_0) = \min_{u,v} E_{x_0,y_0}(u, v)$$

$u^2 + v^2 = 1$

shift parameters

$$E_{x_0,y_0}(u, v) = \begin{bmatrix} u & v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix}, \quad A = \sum_{(x,y)\in W} \begin{bmatrix} \mathbf{I}_x^2 & \mathbf{I}_x\mathbf{I}_y \\ \mathbf{I}_y\mathbf{I}_x & \mathbf{I}_y^2 \end{bmatrix}$$

What vector [u,v] minimizes this?

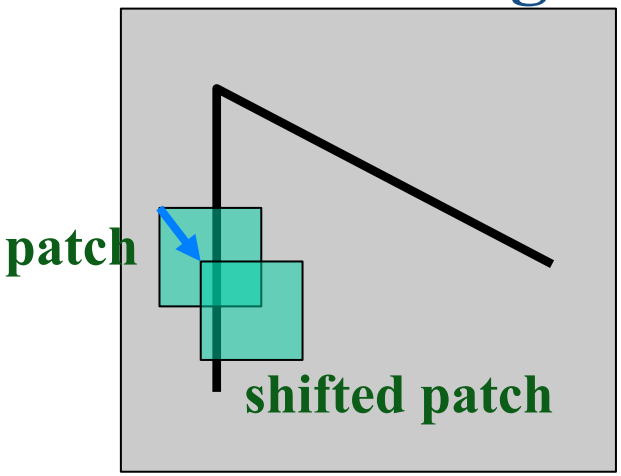
$$E(u, v) = x^T A x$$

Recall eigenvectors: $Ax = \lambda x$ $\quad u^2 + v^2 = 1$

$$E(u, v) = x^T \lambda x = \lambda x^T x = \lambda \|x\| \quad \text{so } \|x\| = 1$$

The minimum is given by the smallest eigenvalue!

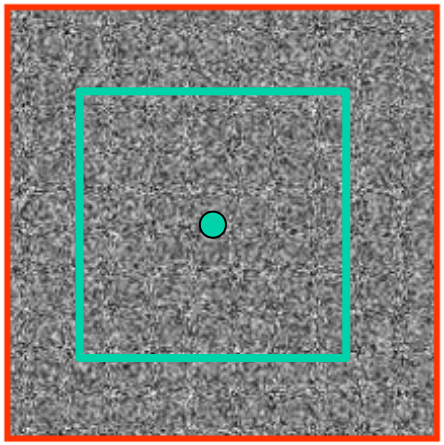
What will eigenvalues (and eigenvectors) look like?



Eigenvector = direction [u,v] of shifting the patch

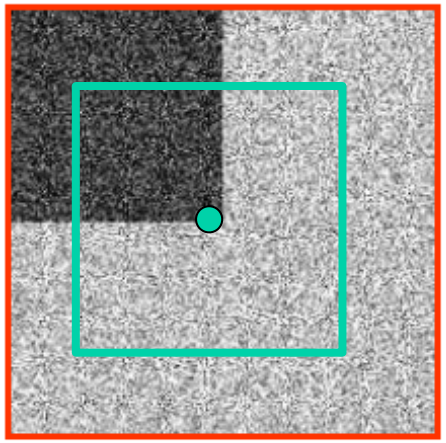
Eigenvalue = SSD error

Flat



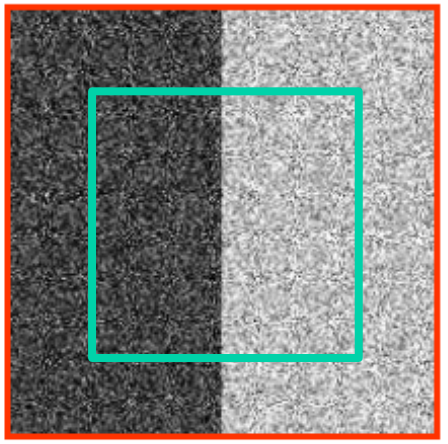
Eigenvalues are small

Corner



Both eigenvalues are large

Linear Edge

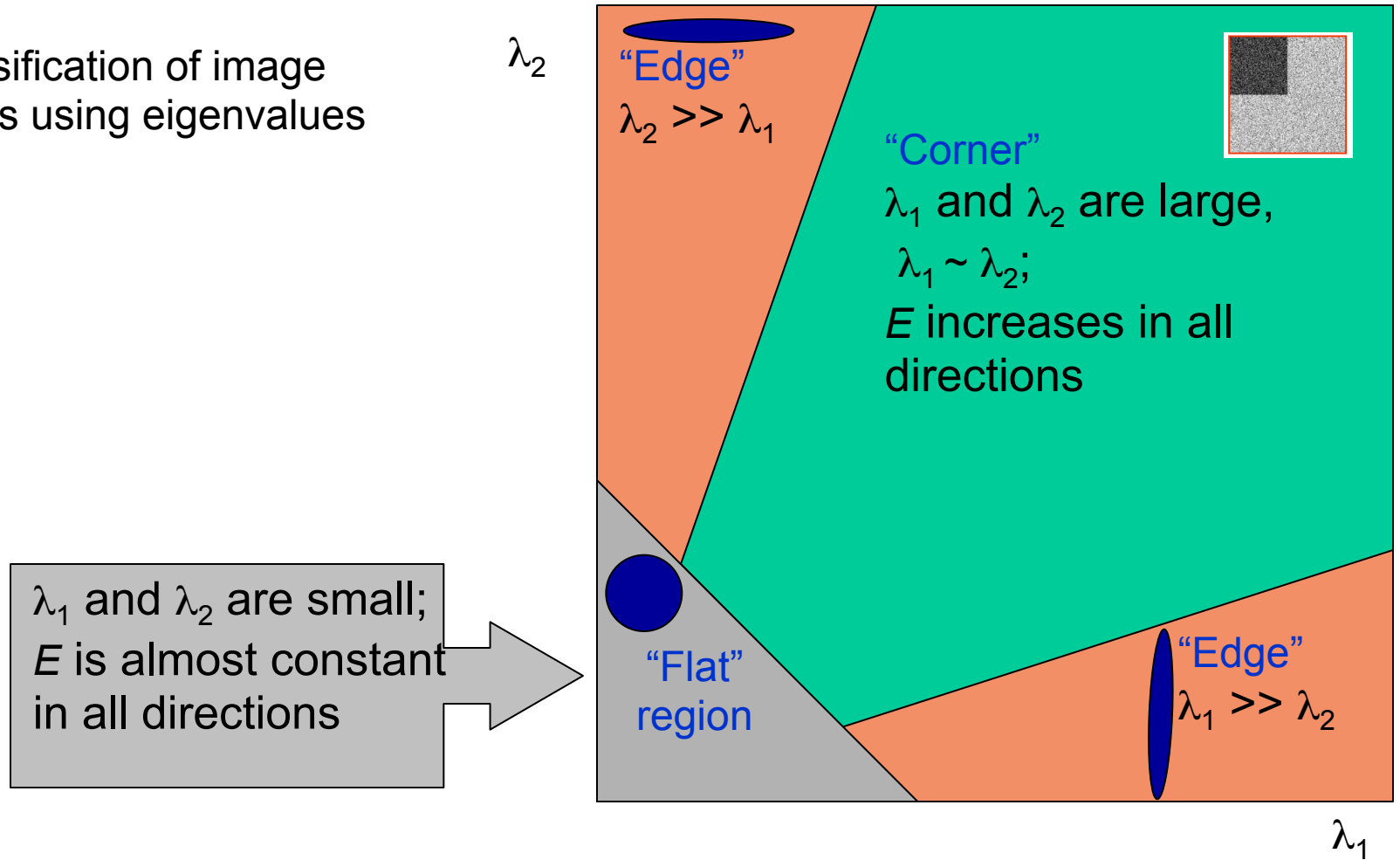


1 eigenvalue is small

1 eigenvalue is large

Intuition behind eigenvalues

Classification of image points using eigenvalues of *A*:



Efficient computation

Computing eigenvalues (and eigenvectors) for every patch is expensive

But - it's easy to compute their sum (trace) and product (determinant)

product of eigenvalues

- $\text{Det}(A) = \lambda_{min}\lambda_{max}$
- $\text{Trace}(A) = \lambda_{min}+\lambda_{max}$ (trace = sum of diagonal entries of a matrix)

sum of eigenvalues

$$R = 4\frac{Det(A)}{Trace(A)^2}$$

What if $\lambda_{max} \gg \lambda_{min}$? Edge - then $R \approx 4\lambda_{min}/\lambda_{max}$

What if $\lambda_{max} \approx \lambda_{min}$? Flat or corner - then $R \approx 1$

$$R = Det(A) - \alpha Trace(A)^2$$

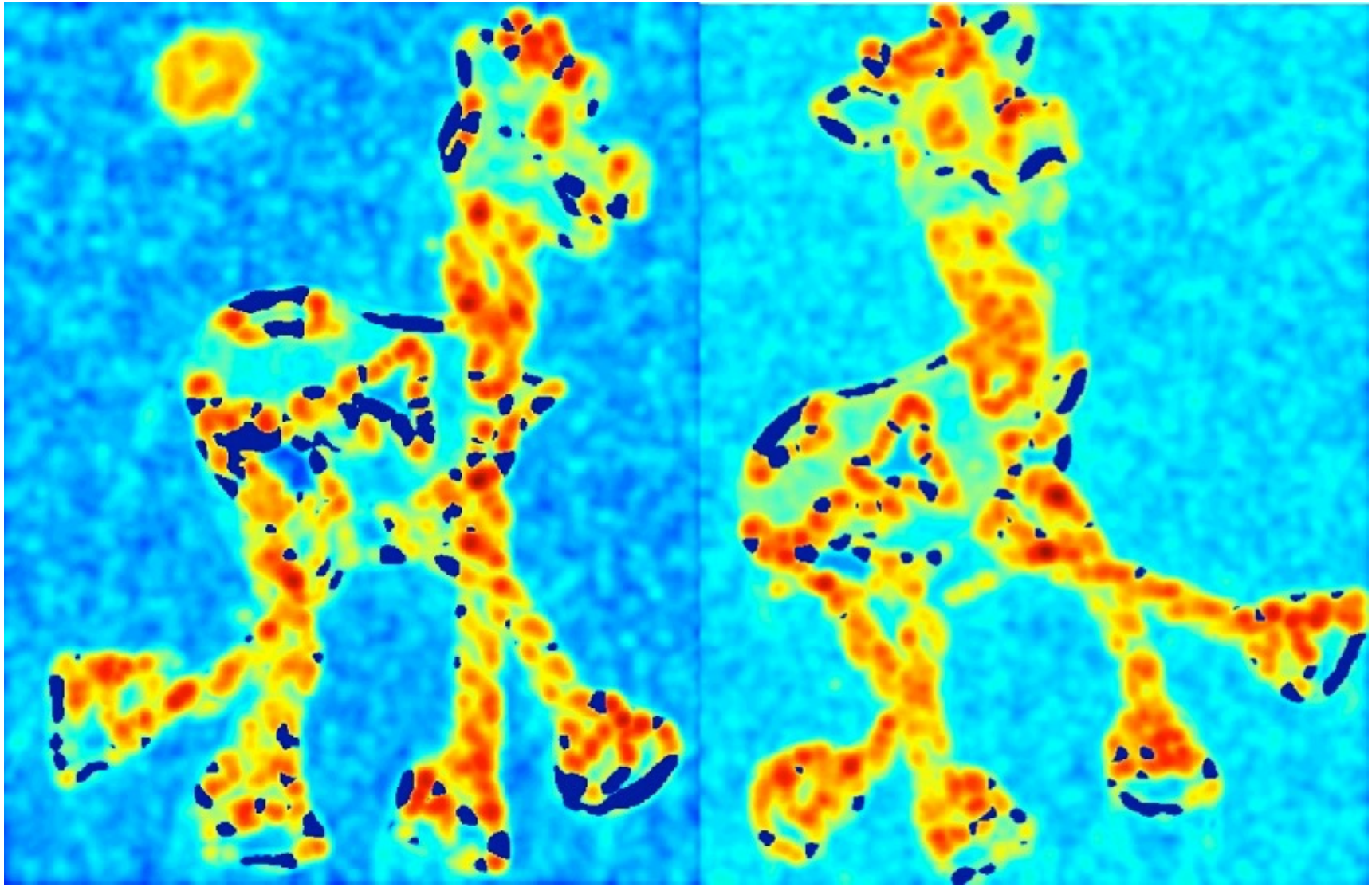
R is large when λ_{min} is large

α is a constant that you can tune

Let's compute $R = Det(A) - \alpha Trace(A)^2$ for every pixel

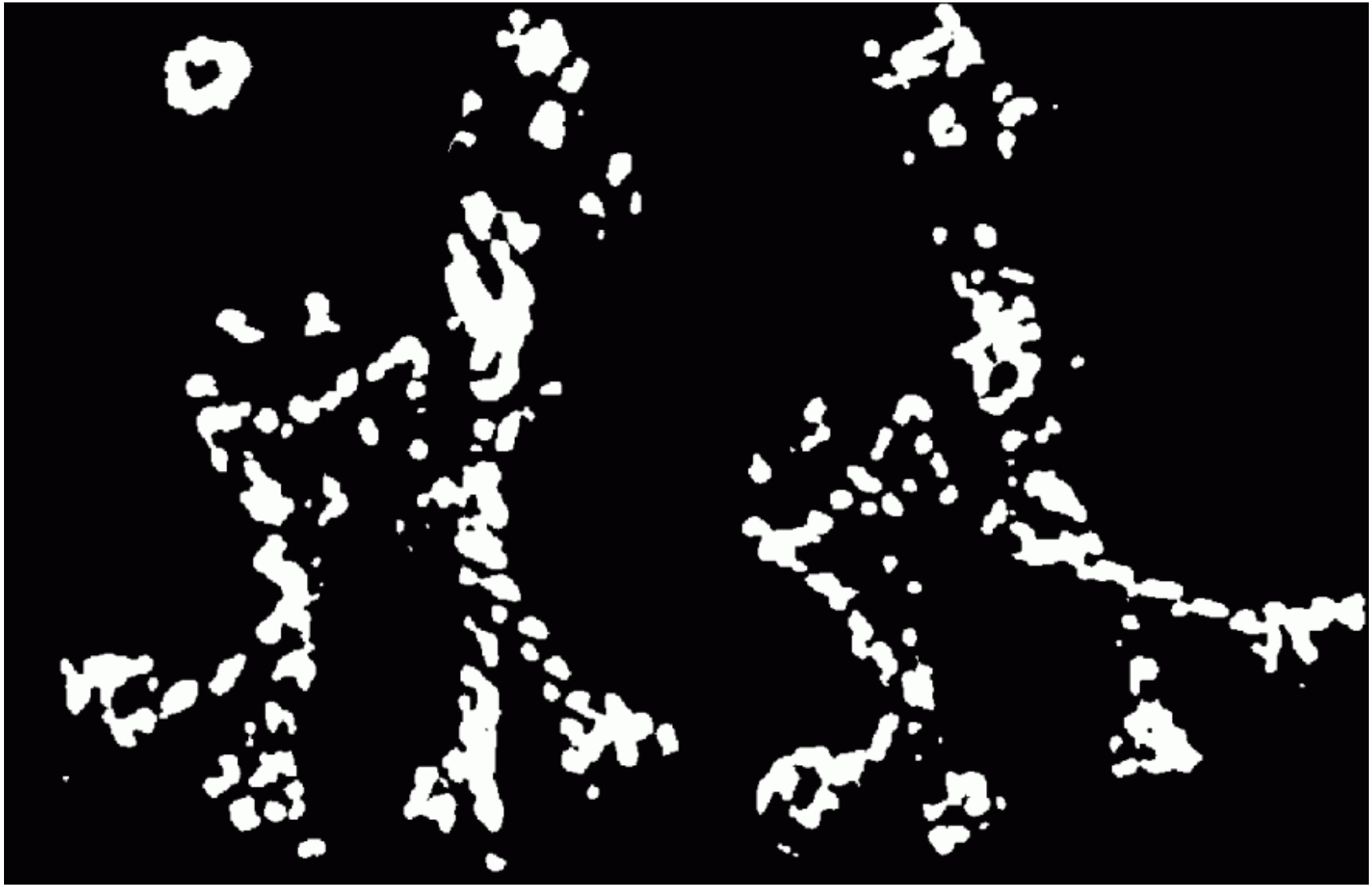
Corner-ness (red high, blue low)

Let's compute $R = Det(A) - \alpha Trace(A)^2$ for every pixel

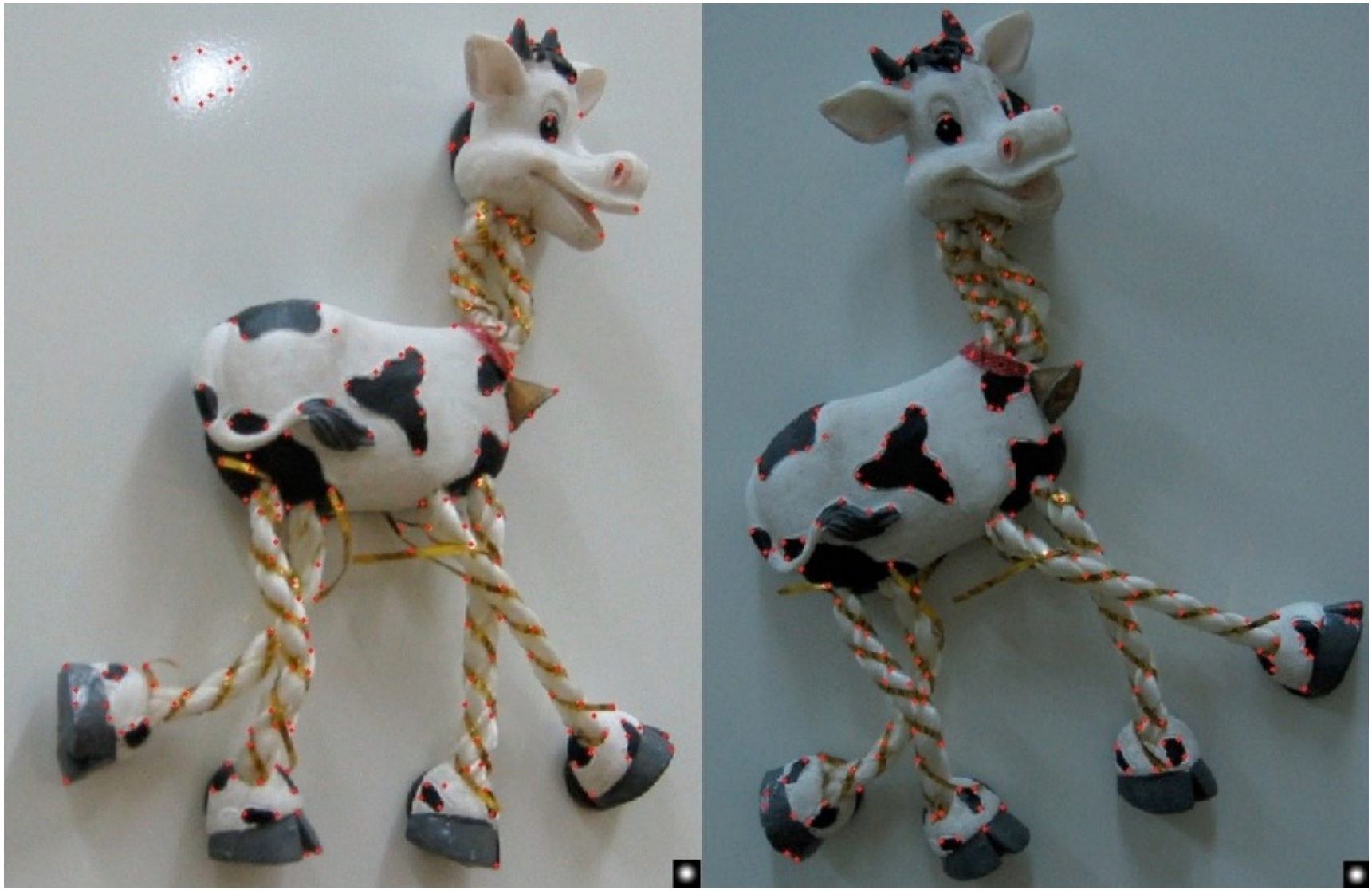


Threshold (f > value)

How do we reduce these to just a single pixel-thickness corner?



NMS -> Harris corners (in red)



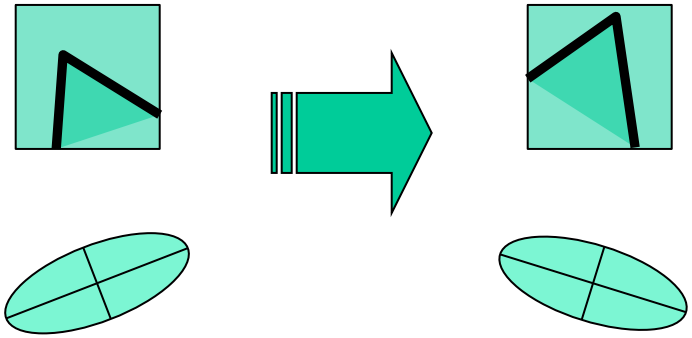
These red dots (corners) are hopefully easy to match across images

Scale and rotation invariance

Will our corner detector return the same exact points on a rotated image?

What about a scaled image?

Rotation invariance (?)



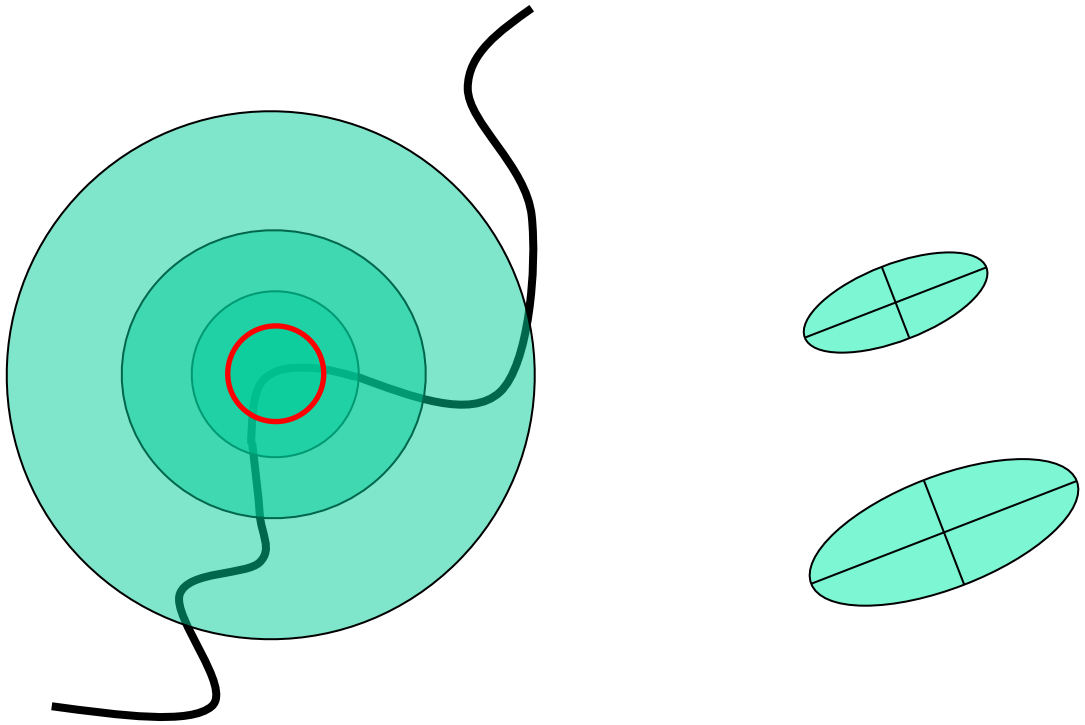
Are eigenvectors stable under rotations? No

(worst-case: flat patch; eigenvalues are equal; eigenvectors can be anything!)

Are eigenvalues stable under rotations? Yes

So the Harris corner detector is rotation invariant!
(with the caveat of some tricky implementation details)

Scale invariance?



Are eigenvector stable under small changes in scale? Yes

Are eigenvalues stable under small changes in scale? No

A solution to scale

search over image scales

For each pixel, compute $R = Det(A) - \alpha Trace(A)^2$ at every scale

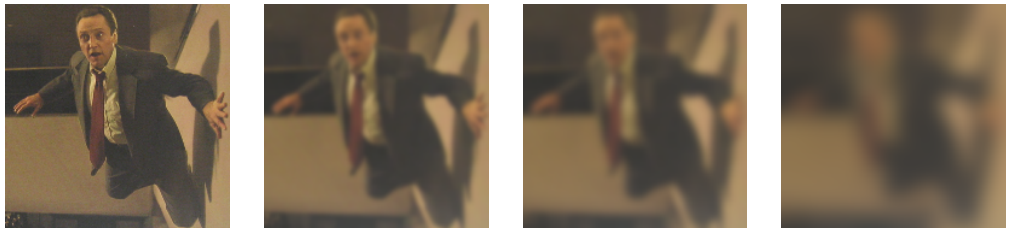
Find the scale that maximizes R

This is the “best” scale for that point!

Annoying “details”:

1. Positions across scales don’t align

Solution: construct blurred versions of an image at the same size (similar to changing “scale”)



Cool outcome: Now we can obtain “fractional scales”!

2. Gradients across scales aren’t comparable
(gradients are smaller on blurred images - less sharp edges!)

Solution: multiply gradients by scale factor

Scale-space theory: A basic tool for analysing structures at different scales

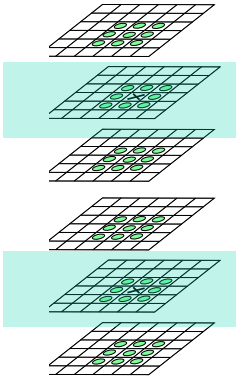
Tony Lindeberg

Computational Vision and Active Perception Laboratory (CVAP)
Department of Numerical Analysis and Computing Science
Royal Institute of Technology, S-100 44 Stockholm, Sweden

Coarse-to-fine search

Challenge: its expensive to compute cornerness(x,y,σ) over all fractional scales (e.g., $\sigma = 1.2$)

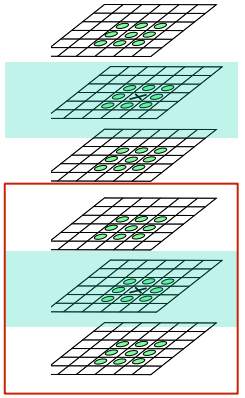
1. Optimize cornerness(x,y,σ) over *coarse* set of locations and scales



2. Fine-tune "sub-pixel" accuracy by iterating the following:

   i. Given (x,y), we can find maximal σ with *finer search*

   ii. Given sigma, find maximal (x,y) of cornerness

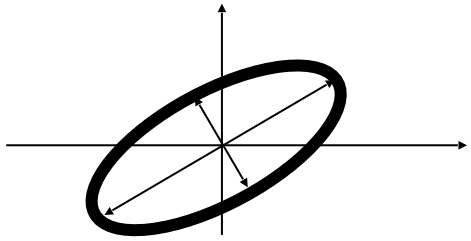


Extension: anisotropic scale

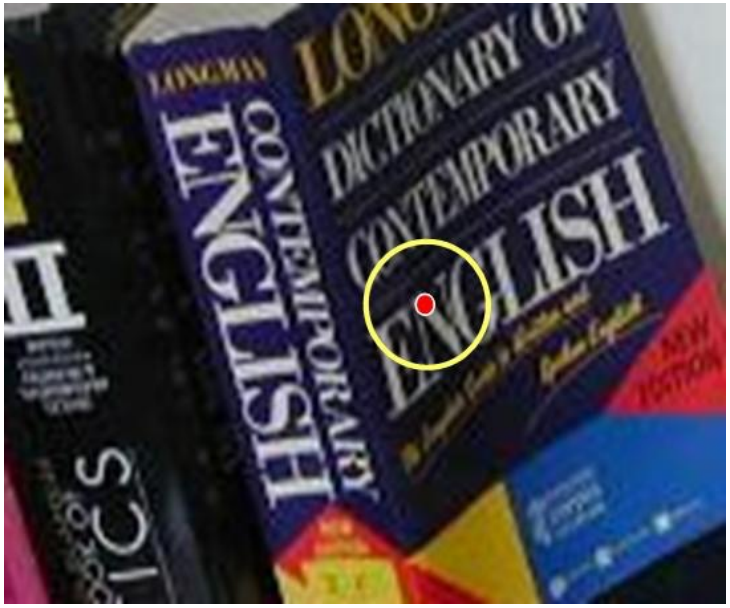
Need richer description of "neighborhood" or scale

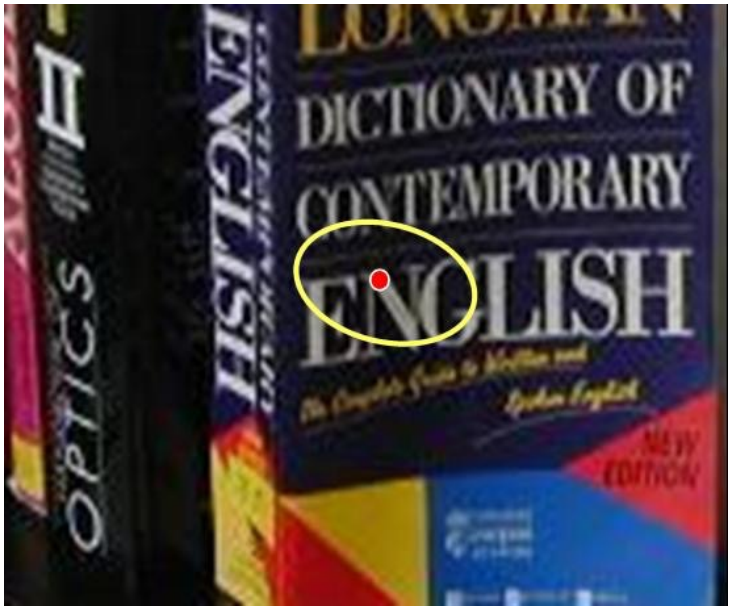
Replace scalar σ with a matrix Σ

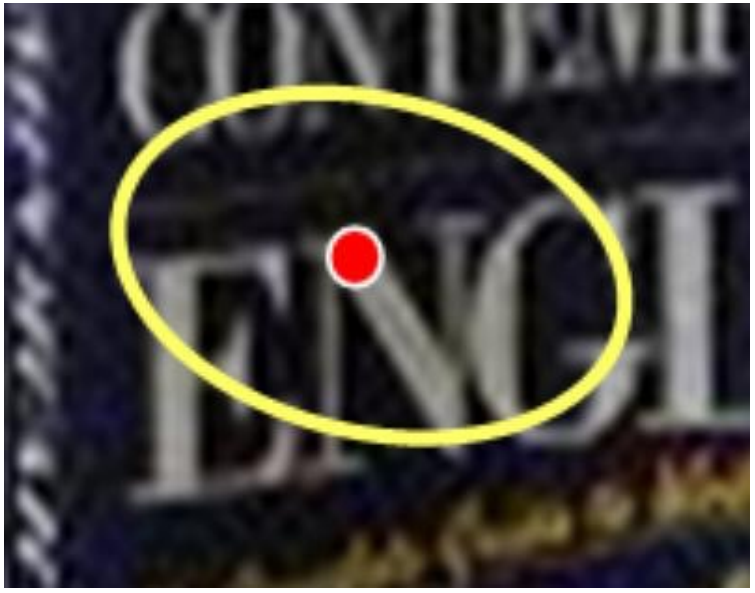
(e.g., scale differently along x and y axes, or even a diagonal axis)



Affine Invariance







Outline

- **Interest point detection (Harris Corners)**
- Descriptors (SIFT, BRIEF, Filter Banks)
- RANSAC