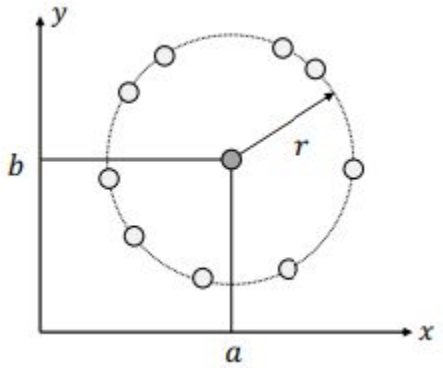
Computer Vision

Lec 7: Hough Transform (contd.), Corner Detection

Dr. Pratik Mazumder

Circle Detection



parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

What is the dimension of the parameter space?

Let's assume radius known

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

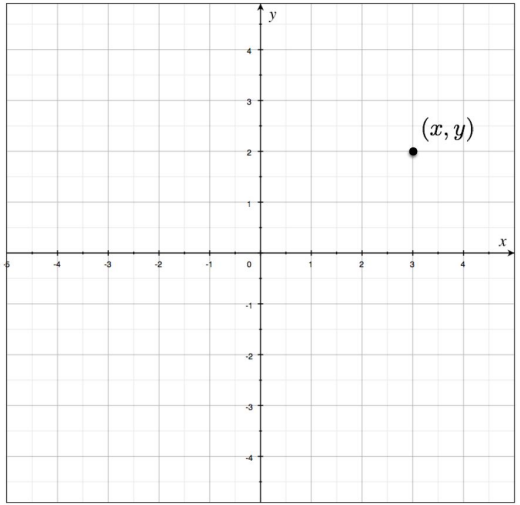
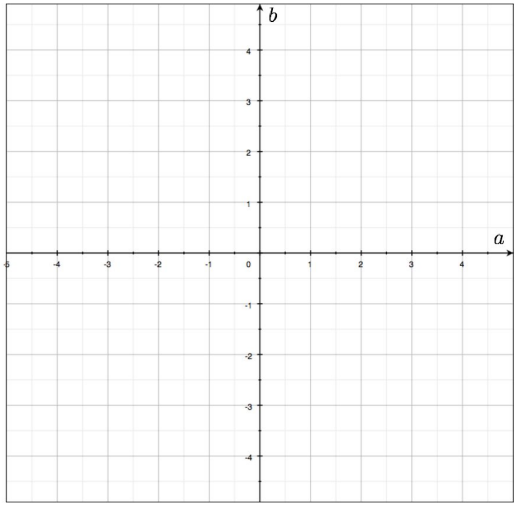


Image space

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



Parameter space

What does a point in image space correspond to in parameter space?

Let’s assume radius known

parameters

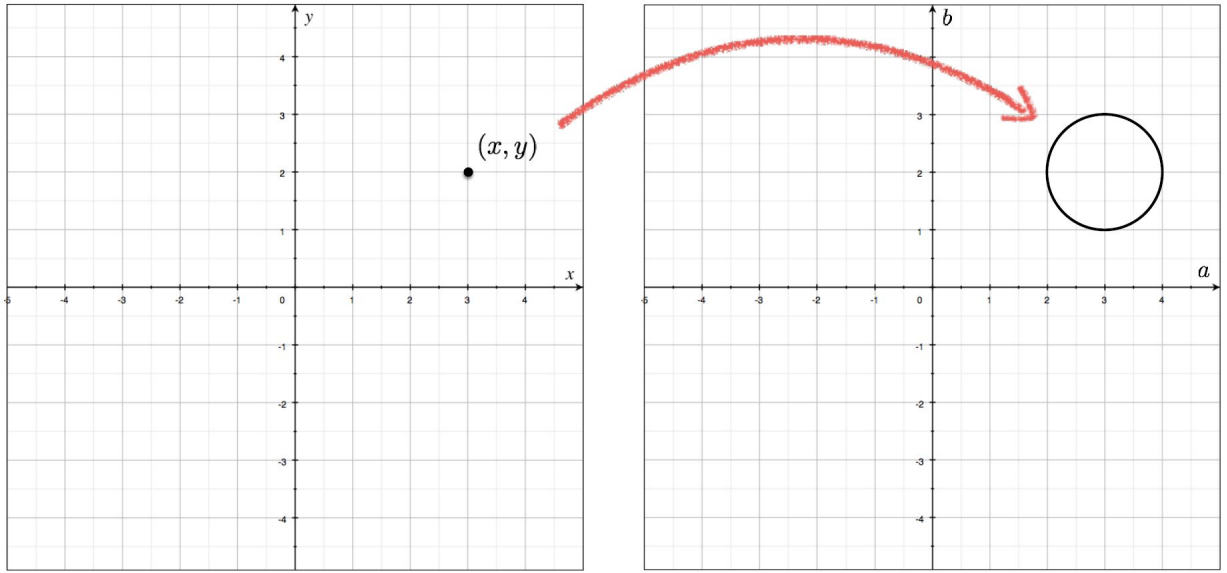
$$(x-a)^2+(y-b)^2=r^2$$

variables

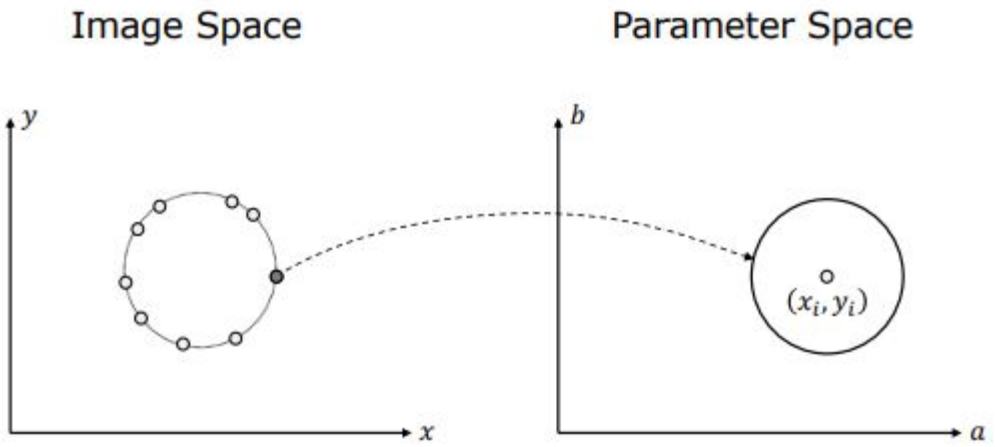
parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables



Let’s assume radius known

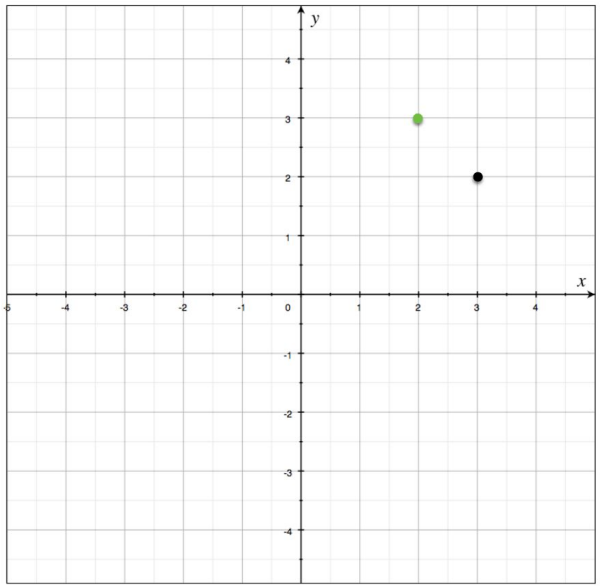


Let’s assume radius known

parameters

$$(x-a)^2+(y-b)^2=r^2$$

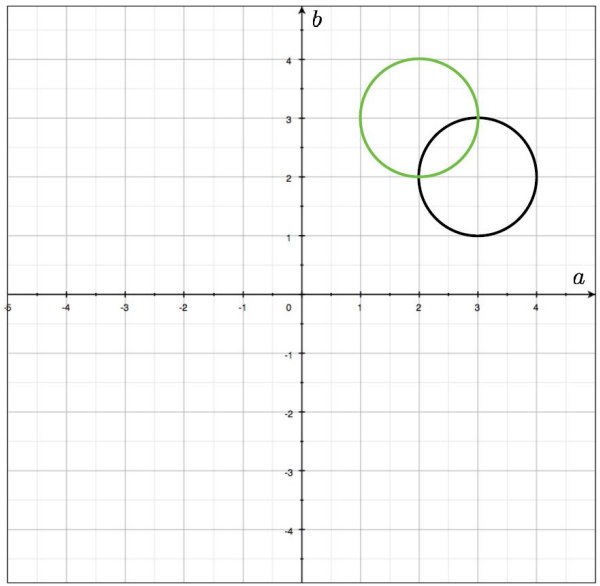
variables



parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

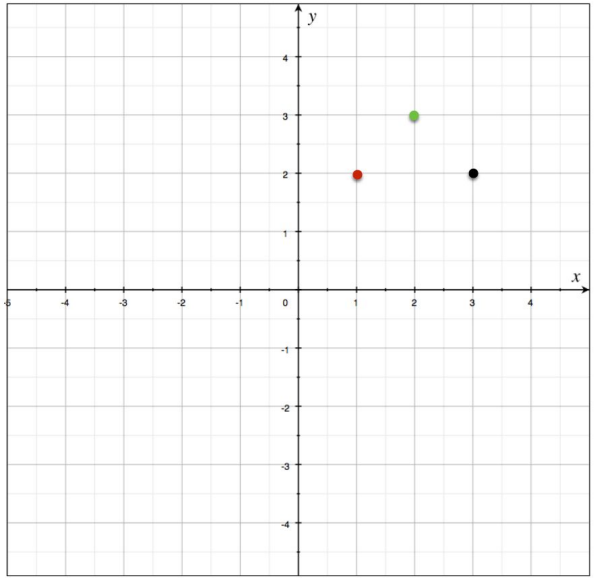


Let’s assume radius known

parameters

$$(x-a)^2+(y-b)^2=r^2$$

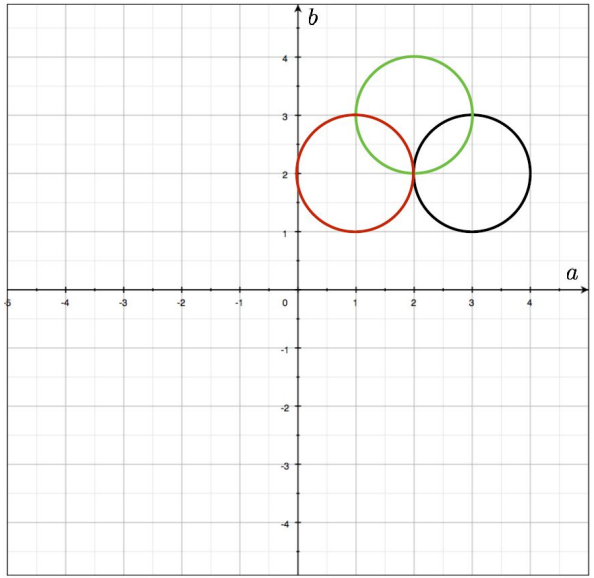
variables



parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

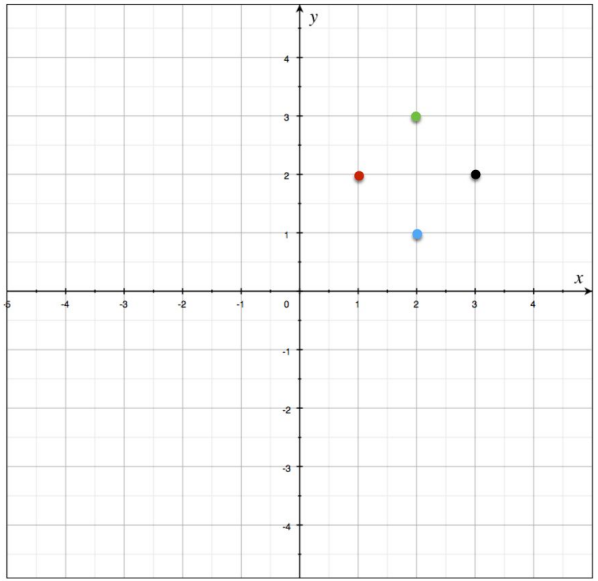


Let’s assume radius known

parameters

$$(x-a)^2+(y-b)^2=r^2$$

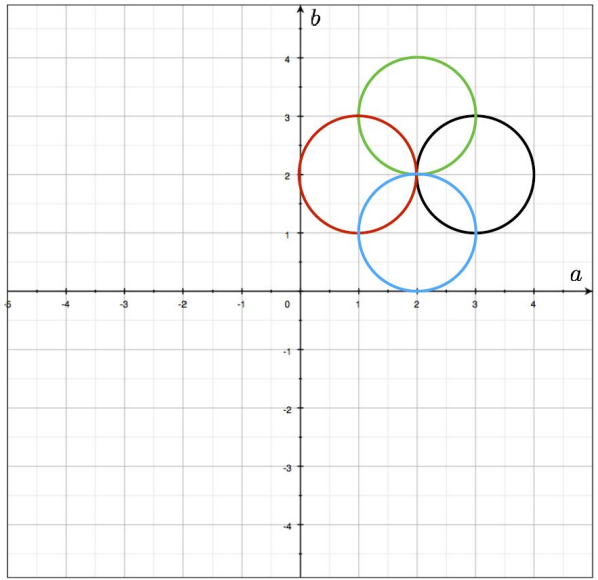
variables



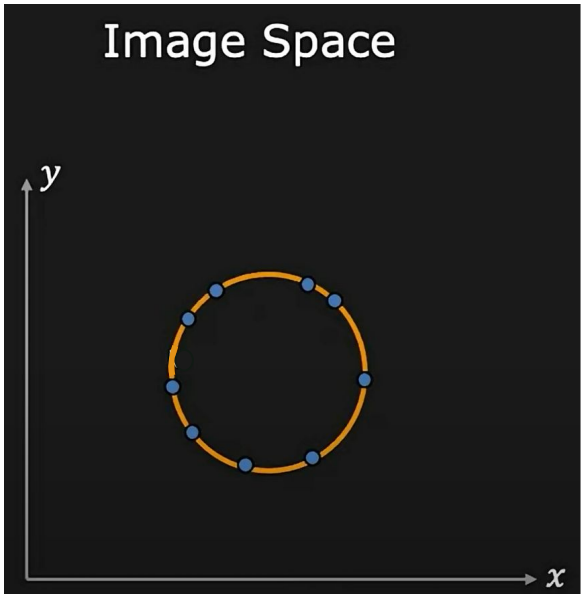
parameters

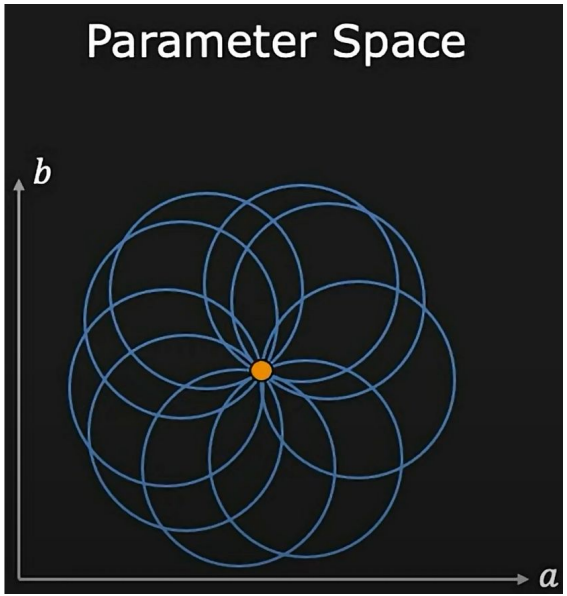
$$(x-a)^2+(y-b)^2=r^2$$

variables



Let’s assume radius known





What if radius is unknown?

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

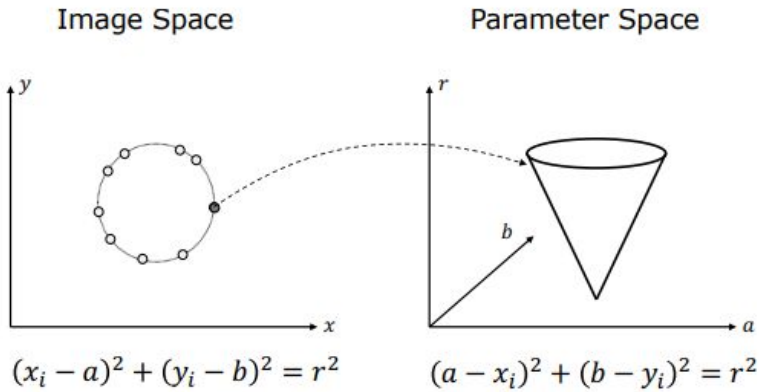
parameters

$$(x-a)^2+(y-b)^2=r^2$$

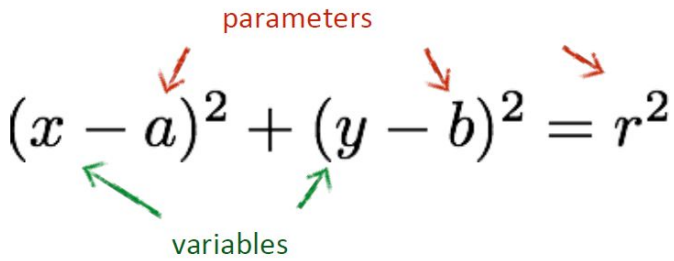
variables

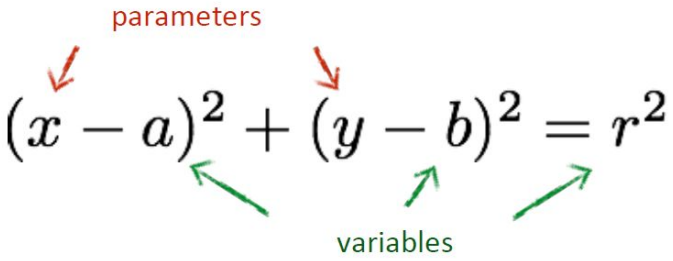
If radius is not known: 3D Hough Space!

Use Accumulator array $A(a,b,r)$



What if radius is unknown?





If radius is not known: 3D Hough Space!

Use Accumulator array $A(a,b,r)$

The work that needs to be done in parameter space increases exponentially with the number of unknown parameters.

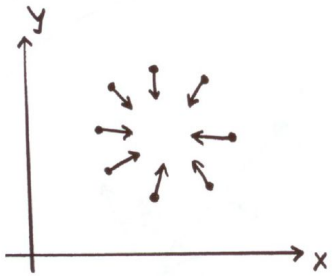
In short, as the parametric shape we are looking for increases in complexity, the Hough transform becomes less and less practical.

Using Gradient Information

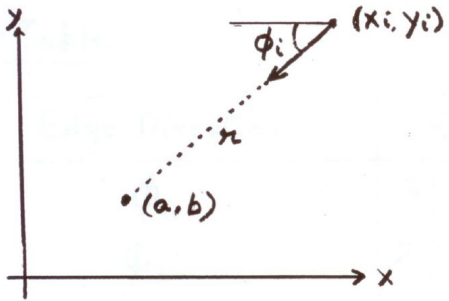
Gradient information can save lot of computation:

Edge Location (x_i, y_i)

Edge Direction ϕ_i



Assume radius is known:



$$a = x - r\cos\phi$$

$$b = y - r\sin\phi$$

Need to increment only one point in accumulator!

Using Gradient Information

parameters

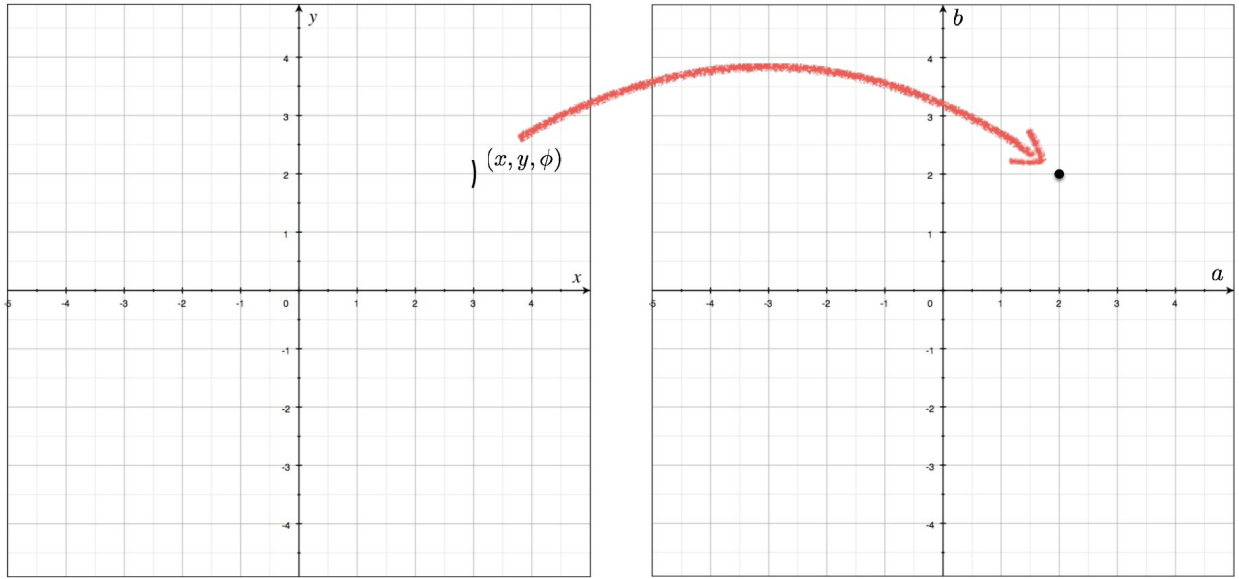
$$(x-a)^2+(y-b)^2=r^2$$

variables

parameters

$$(x-a)^2+(y-b)^2=r^2$$

variables

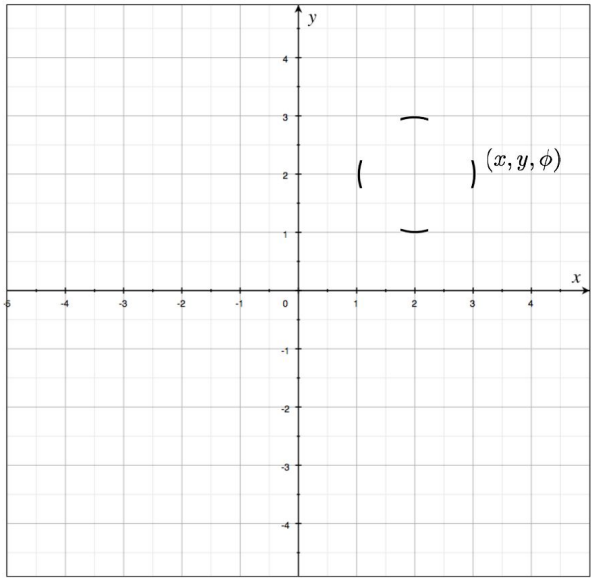


Using Gradient Information

parameters

$$(x-a)^2+(y-b)^2=r^2$$

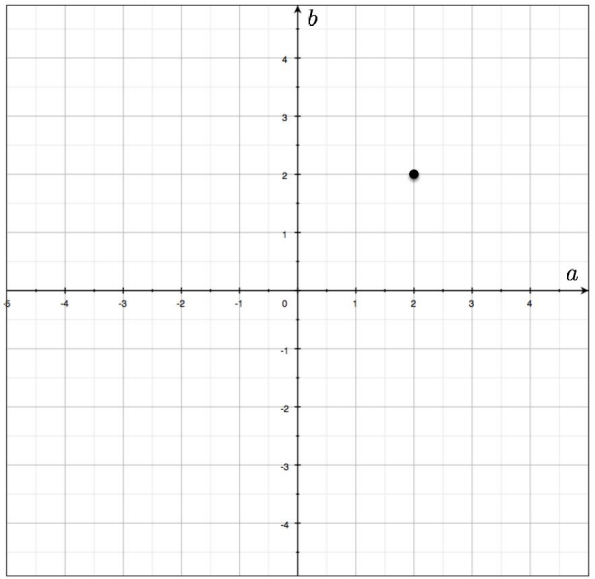
variables



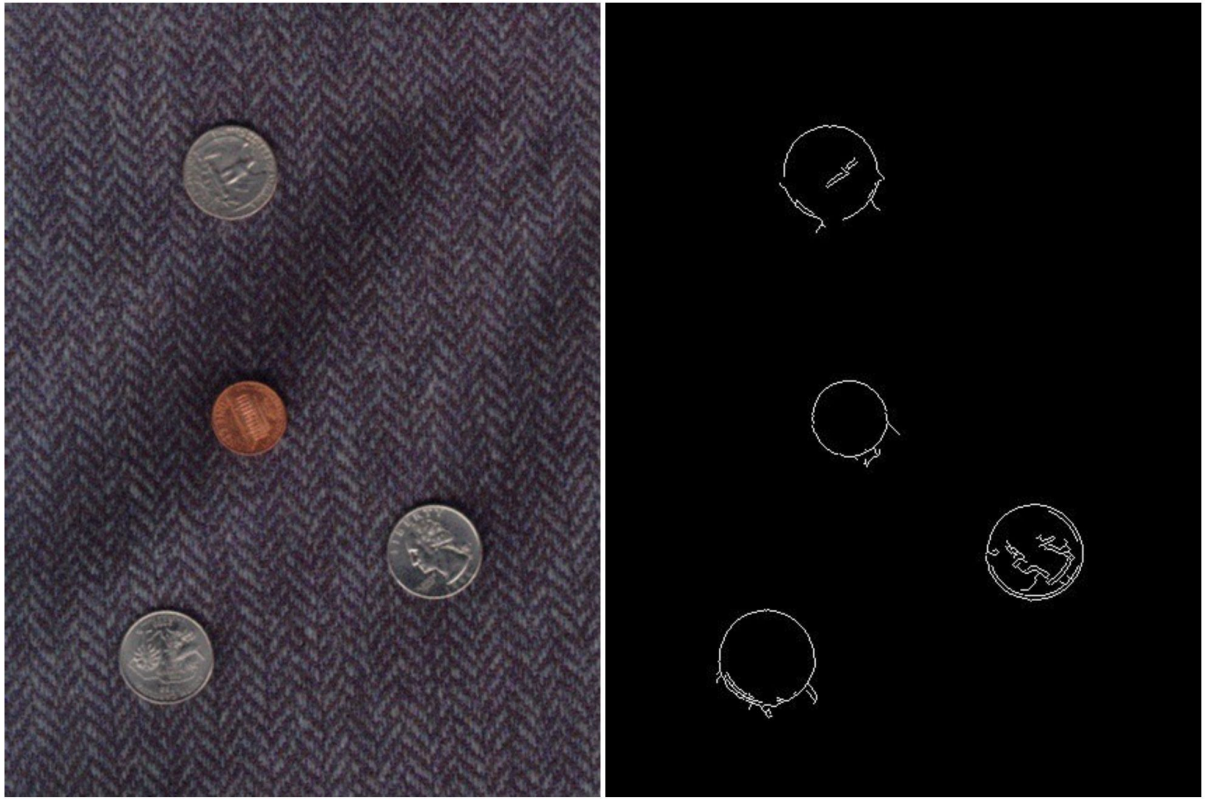
parameters

$$(x-a)^2+(y-b)^2=r^2$$

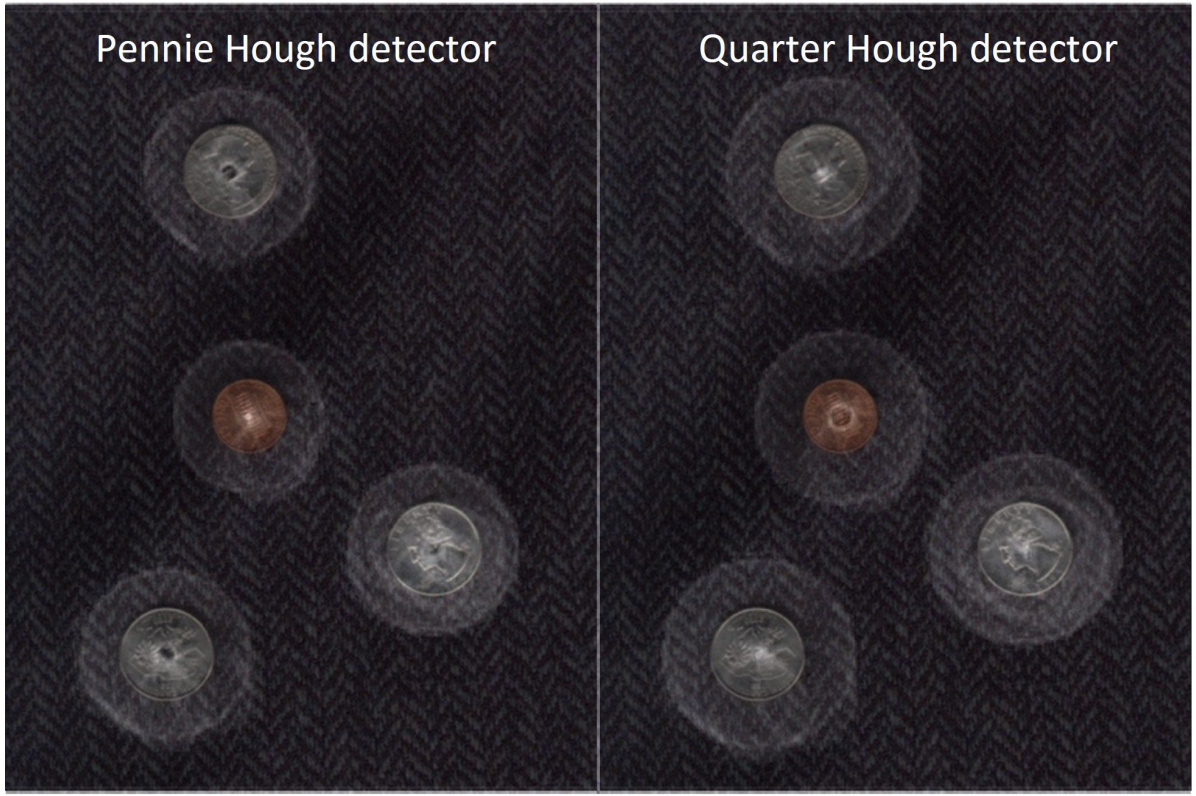
variables



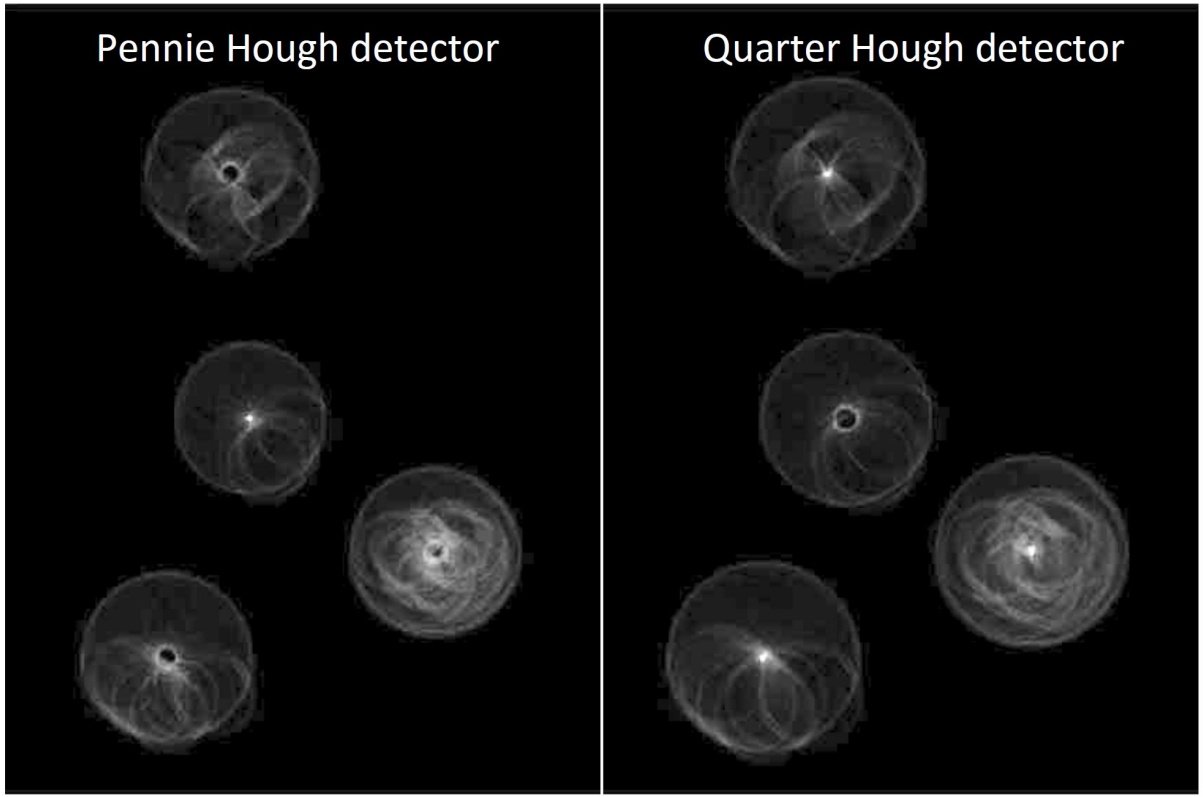
Hough Circle Detection Example



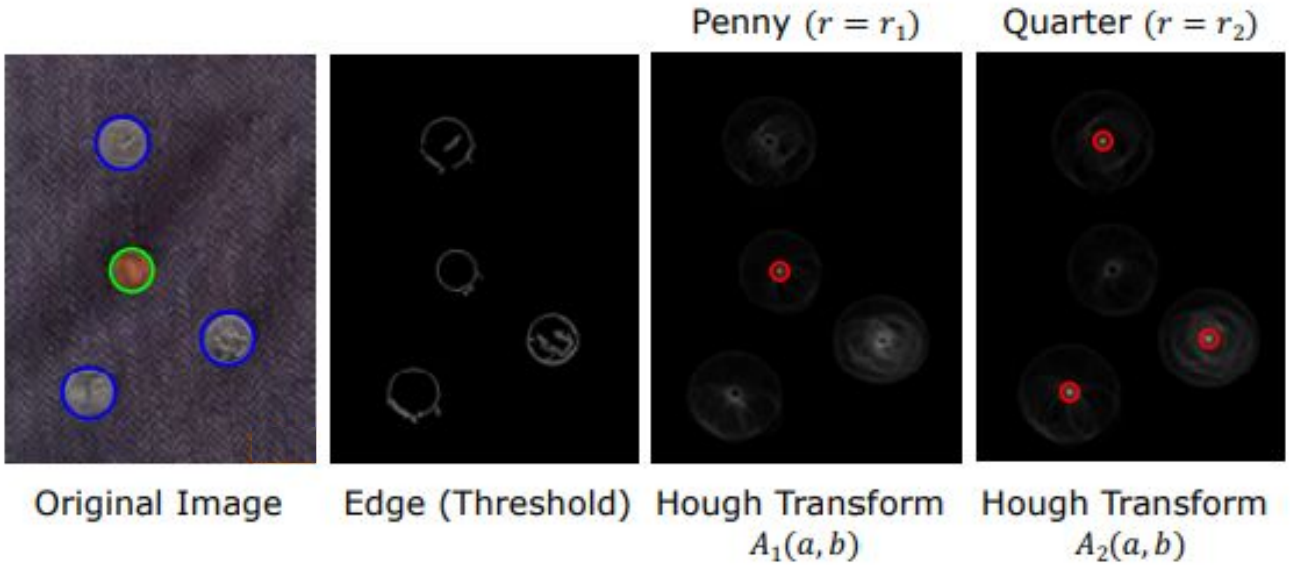
Hough Circle Detection Example



Hough Circle Detection Example

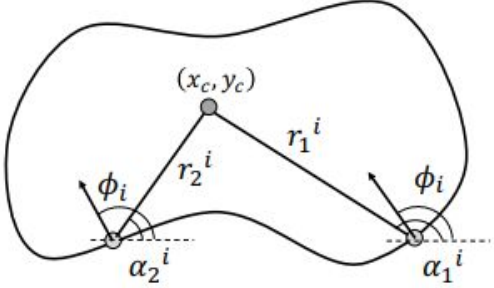


Hough Circle Detection Example



Generalized Hough Transform

GHT helps to find complicated shapes.



Reference point: (x_c, y_c)

Edge direction: ϕ_i $\quad 0 \le \phi_i < 2\pi$

Edge location: $\vec{r}_k{}^i = (r_k{}^i, \alpha_k{}^i)$

1. The first step is to create a model of the object that can be used by the Hough transform. This step is done off-line

2. Define a reference point (x_c, y_c) for the shape.
3. For each point on the object's boundary in the template, we are going to assume that we have both the edge location and the edge direction.

4. Then, for each point boundary point, we represent it using the vector ($r_k{}^i$, $\alpha_k{}^i$), which includes the distance $r_k{}^i$ of the point from the reference point and the angle $\alpha_k{}^i$ the edge makes with respect to the horizontal axis.

5. We use the above approach to create a model of the object in the form of a table, called the ϕ-table.

6. The index to the table is the edge direction ϕ_i and the entry is a list of the vectors ($r_k{}^i$,$\alpha_k{}^i$) corresponding to all the points on the object's boundary that have that edge direction ϕ_i.

ϕ-Table

Edge Direction	$\vec{r} = (r, \alpha)$
ϕ_1	$\vec{r}_1{}^1, \vec{r}_2{}^1, \vec{r}_3{}^1$
ϕ_2	$\vec{r}_1{}^2, \vec{r}_2{}^2$
⋮	⋮
ϕ_n	$\vec{r}_1{}^n, \vec{r}_2{}^n, \vec{r}_3{}^n, \vec{r}_4{}^n$

Generalized Hough Transform

7. We will create an accumulator array with parameters x_c and y_c, initialize it to zero, and we are going to be voting for the location of the reference poin of the object.
8. In our input edge pixel image, we have both the location and the direction of the edge at each edge pixel.
9. So, we use the edge direction ϕ_i of the point as an index into our ϕ-table to find all the vectors (r_k^i, α_k^i) associated with it.
10. We use the vectors to vote for the reference point in the accumulator array.
11. If we get a strong peak in the array, then we have found the object, and its location in the image is determined by the location of its reference point.

ϕ-Table

Edge Direction	$\vec{r} = (r, \alpha)$
ϕ_1	$\vec{r}_1^{\,1}, \vec{r}_2^{\,1}, \vec{r}_3^{\,1}$
ϕ_2	$\vec{r}_1^{\,2}, \vec{r}_2^{\,2}$
⋮	⋮
ϕ_n	$\vec{r}_1^{\,n}, \vec{r}_2^{\,n}, \vec{r}_3^{\,n}, \vec{r}_4^{\,n}$

- Create accumulator array $A(x_c, y_c)$
- Set $A(x_c, y_c) = 0$ for all (x_c, y_c)
- For each edge point (x_i, y_i, ϕ_i),

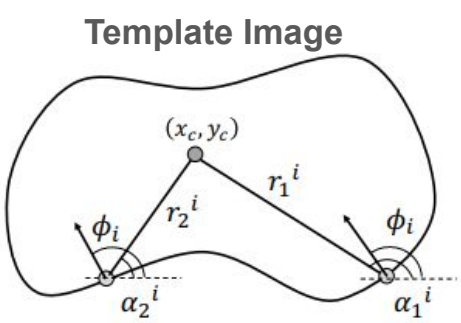
  For each entry $\phi_i \rightarrow \vec{r}_k^{\,i}$ in ϕ – table

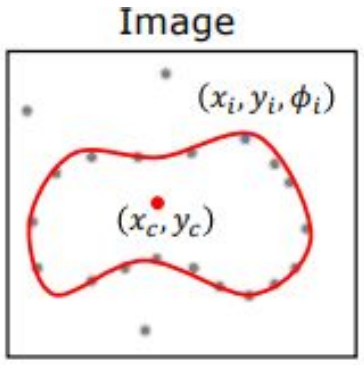
$$x_c = x_i \pm r_k^{\,i} \cos(\alpha_k^{\,i})$$

$$y_c = y_i \pm r_k^{\,i} \sin(\alpha_k^{\,i})$$

$$A(x_c, y_c) = A(x_c, y_c) + 1$$

- Find local maxima in $A(x_c, y_c)$



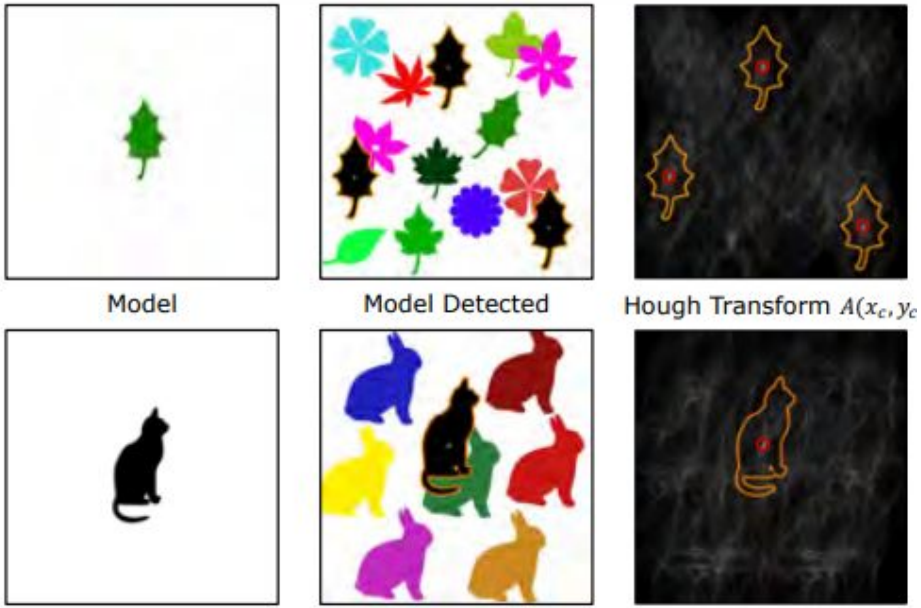


$A(x_c, y_c)$

x_c					
	0	0	0	0	0
	0	2	0	1	0
	0	0	4	1	0
	0	2	0	0	0
	0	0	0	1	0

y_c

Generalized Hough Transform



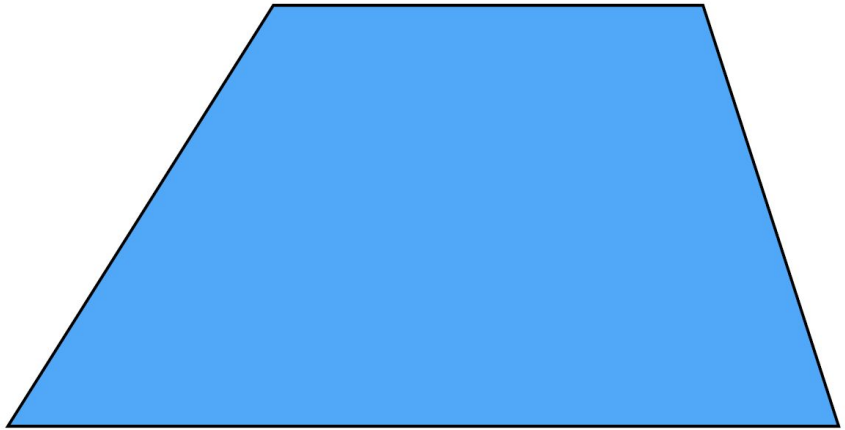
Assumption: the object should appear in the image with the same orientation and scale.

Hough Transform: Comments

- Works on disconnected edges.
- Relatively insensitive to occlusion and noise.
- Effective for simple shapes (lines, circles, etc.).
- Complex Shapes: Generalized Hough Transform.
- Trade-off between work in image space and parameter space.

Corner Detection

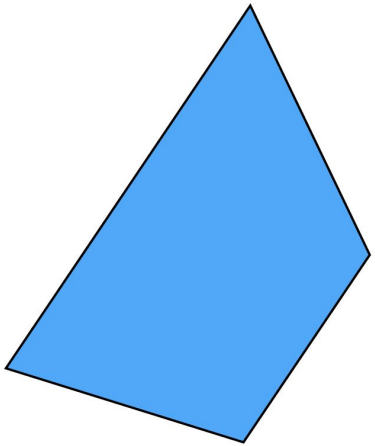
Image matching



Pick a point in the image.
Find it again in the next image.

What type of feature would you select?

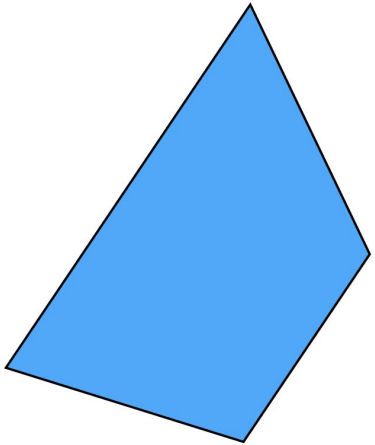
Image matching



Pick a point in the image.
Find it again in the next image.

What type of feature would you select?

Image matching

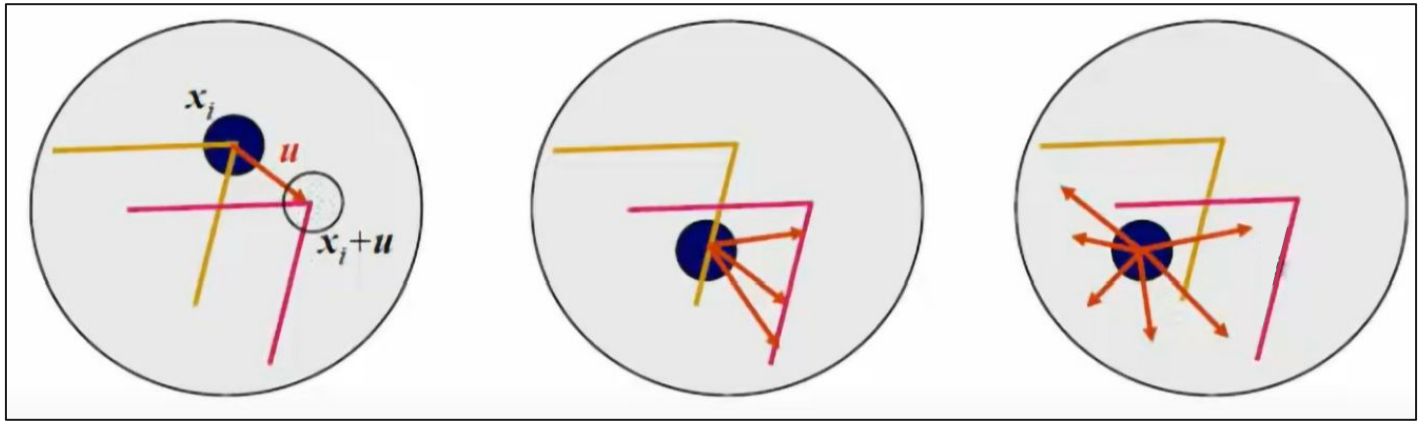


Pick a point in the image.
Find it again in the next image.

What type of feature would you select?

a corner

Image matching - Aperture Problem



Corner points are easier to match uniquely

Corner Detection

- Sudden change in intensity in more than one direction
- A corner can be defined as the intersection of two edges.
- A corner can also be defined as a point for which there are two dominant and different edge directions in a local neighborhood of the point.
- In practice, many corner detection methods detect interest points in general, and in fact, the terms "corner" and "interest point" are often used interchangeably.
- These interest points, often located at corners or junctions of edges, serve as key features for various computer vision tasks.

Why detect corners?

- Image alignment (homography, fundamental matrix)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation

Background: Matrix, Eigen Values, Eigen Vectors, SVD

Matrix Vector Operation

$$\underset{n}{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ . & . & . & . \\ . & . & . & . \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}} \quad \underset{m}{\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_m \end{bmatrix}}$$

A x

Ax = ?

A can be considered as a set of row vectors.

$$A = \begin{bmatrix} row_1 \\ row_2 \\ . \\ . \\ row_n \end{bmatrix}$$

Then Ax is a column vector where each element i is the dot product of the i^{th} row and x vector.

$$Ax = \begin{bmatrix} row_1 . x \\ row_2 . x \\ . \\ . \\ row_n . x \end{bmatrix}$$

Matrix Vector Operation

$$\underset{n}{} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ . & . & . & . \\ . & . & . & . \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \underset{m}{} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_m \end{bmatrix}$$

A

x

m

Ax = ?

$$A = [col_1\ ,\ col_2,\ \dots,\ col_m]$$

A can be considered as a set of column vectors.

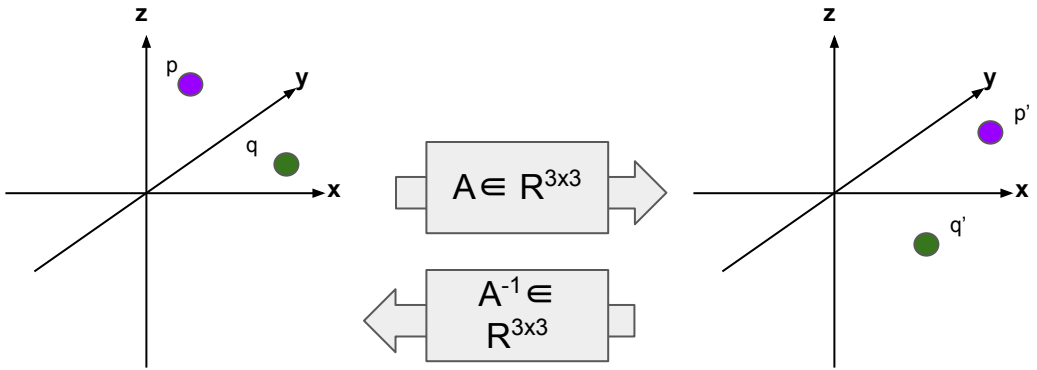
Then Ax is a column vector obtained by the scalar multiplication of i^{th} column vector with the i^{th} element of x and adding the resulting scaled vectors, i.e., scale each column vector with the corresponding element and add the scaled column vectors.

$$x = col_1 \times x_1 \ + \ col_2 \times x_2 \ + \ \dots + col_m \times x_m$$

Geometric Interpretation

$A \in R^{nxm}$, $x \in R^{m}$, $Ax \in R^{m}$

Matrix $A \in R^{3x3}$, can be considered as a function that takes a 3 dimensional vector as input and outputs another 3 dimensional vector (if A is full rank).



The transformation is a one-to-one function only if A is full rank, i.e. rank of $A \in R^{3x3} = 3$

If A^{-1} exists, then it can reverse the transformation induced by A

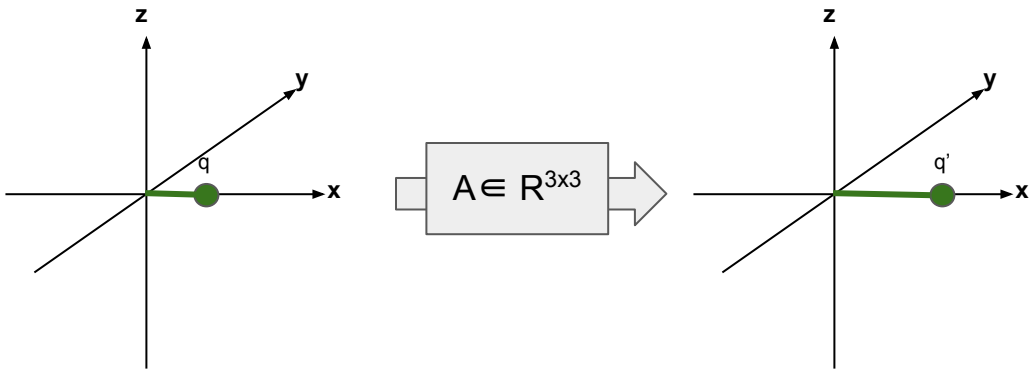
Eigenvalues and Eigenvectors

Suppose, $A \in R^{nxm}$, $q \in R^{m}$, and $Aq = \lambda q$

i.e., Applying transformation A to vector q, produces a scaled version of the vector q

Such vectors are known as the eigen vectors of A, and the corresponding scaling factors are the eigen values.

If $A \in R^{3x3}$, at most 3 linearly independent eigen vectors are possible



Input Space

Output Space

Eigenvalues and Eigenvectors

Suppose, $A \in R^{nxm}$, $q \in R^{m}$, and $Aq = \lambda q$

The eigen values are the roots of the following characteristics equation

$p(\lambda) = \det(A - \lambda I) = 0$

Solve for the Eigenvectors: For each eigenvalue λ, solve the equation: $(A - \lambda I)v=0$

Eigen values: $\lambda_1,\lambda_2,...,\lambda_N$ Eigen vectors $S= [v_1,v_2,...,v_N]$

$$S^{-1}AS = \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & . & \\ & & & \lambda_N \end{bmatrix}$$

Singular Value Decomposition

- **Singular values**: Non negative square roots of the eigenvalues of $\mathbf{A}^t\mathbf{A}$. Denoted σ_i, $i=1,\ldots,n$

- SVD: If $\mathbf{A}$ is a real m by n matrix then there exist orthogonal matrices $\mathbf{U}$ ($\in \mathbb{R}^{m\times m}$) and $\mathbf{V}$ ($\in \mathbb{R}^{n\times n}$) such that

$$A = U \Sigma V^{-1} \qquad \mathrm{U}^{-1}\mathrm{AV} = \Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & . & \\ & & & \sigma_N \end{bmatrix}$$

Singular Value Decomposition

Every matrix $A \in R^{mxn}$, factorizes into $U\Sigma V^T = A$

Where $U \in R^{mxm}$ and $V \in R^{nxn}$ are orthogonal matrices and $\Sigma \in R^{mxn}$ is a diagonal matrix.

$$\begin{bmatrix} u1 & u2 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix}$$

For example:

$$\underset{U}{\begin{bmatrix} -.40 & .916 \\ .916 & .40 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 5.39 & 0 \\ 0 & 3.154 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.05 & .999 \\ .999 & .05 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 3 & -2 \\ 1 & 5 \end{bmatrix}}$$

Singular Value Decomposition

- Compute the transpose of the matrix, A^T and then compute A^TA.
- Find the eigenvalues of A^TA and sort them in descending order.
- Construct a diagonal matrix Σ by placing the singular values in descending order along the diagonal.
- Use the ordered eigenvalues from step 2 and compute the eigenvectors of A^TA.
- The eigenvectors of A^TA make up the columns of V, and the eigenvectors of AA^T make up the columns of U.
- The singular values in Σ are square roots of eigenvalues from AA^T or A^TA.

$$\begin{bmatrix} u1 & u2 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix}$$

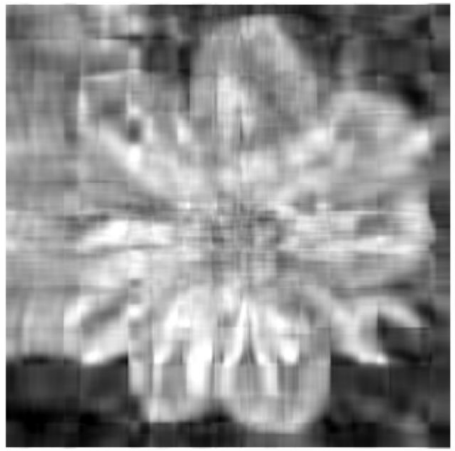
Singular Value Decomposition

$$\underset{U}{\begin{bmatrix} -.39 & -.92 \\ -.92 & .39 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}$$

We can look at **Σ** to see that the first column has a large effect

while the second column has a much smaller effect in this example

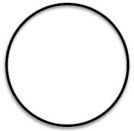
SVD Applications



- For this image, using **only the first 10** of 300 singular values produces a recognizable reconstruction
- So, SVD can be used for image compression

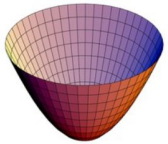
Visualizing quadratics

Visualizing quadratics



Equation of a circle

$$1 = x^2 + y^2$$



Equation of a ‘bowl’ (paraboloid)

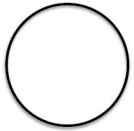
$$f(x, y) = x^2 + y^2$$

If you slice the bowl at

$$f(x, y) = 1$$

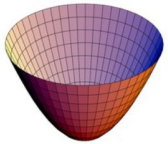
what do you get?

Visualizing quadratics



Equation of a circle

$$1 = x^2 + y^2$$



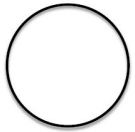
Equation of a ‘bowl’ (paraboloid)

$$f(x, y) = x^2 + y^2$$

If you slice the bowl at

$$f(x, y) = 1$$

what do you get?



Visualizing quadratics

$$f(x, y) = x^2 + y^2$$

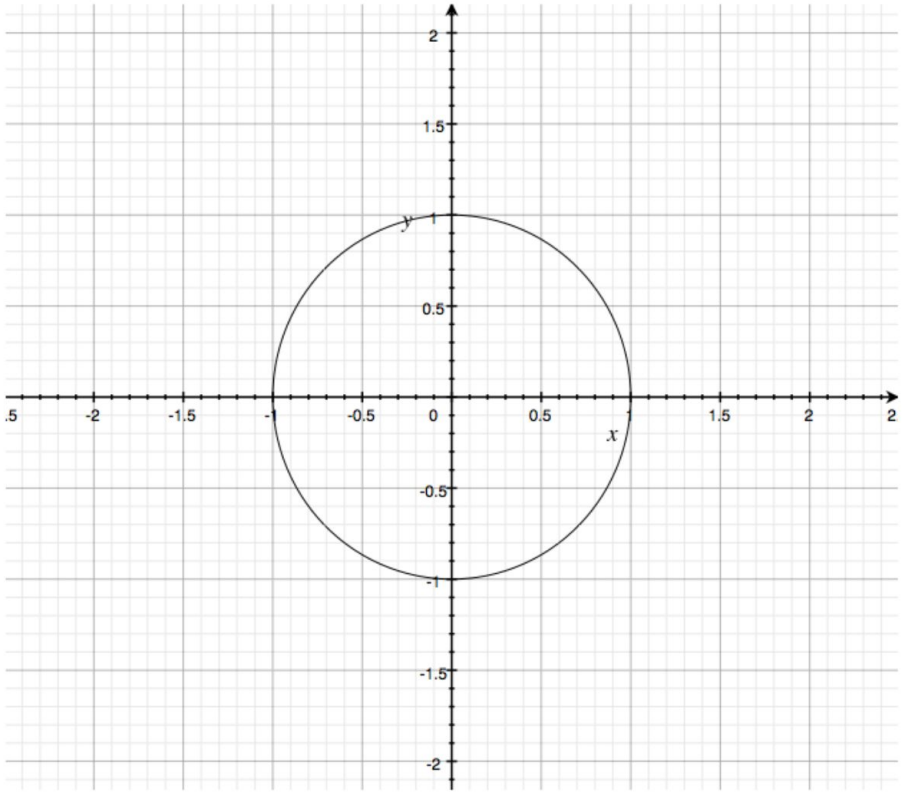
can be written in matrix form like this…

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Visualizing quadratics

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

‘sliced at 1’



Visualizing quadratics

*What happens if you **increase** coefficient on **x**?*

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

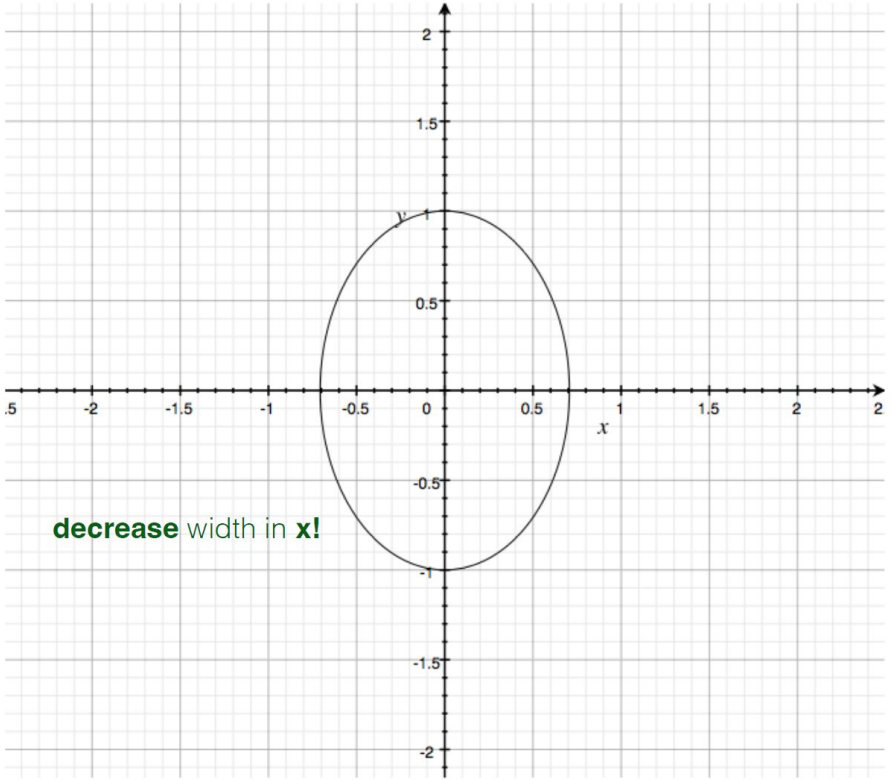
and slice at 1

Visualizing quadratics

*What happens if you **increase** coefficient on **x**?*

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1



decrease width in **x!**

Visualizing quadratics

*What happens if you **increase** coefficient on **y**?*

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

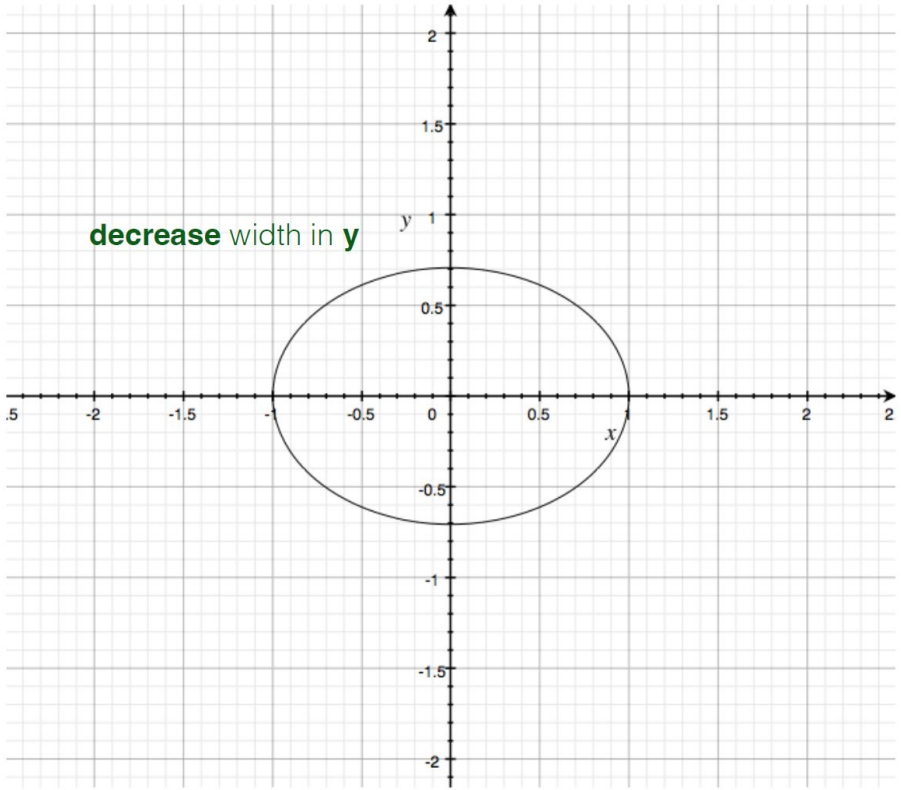
and slice at 1

Visualizing quadratics

*What happens if you **increase** coefficient on **y**?*

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1



Visualizing quadratics

$$f(x, y) = x^2 + y^2$$

can be written in matrix form like this…

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

What's the shape?
What are the eigenvectors?
What are the eigenvalues?

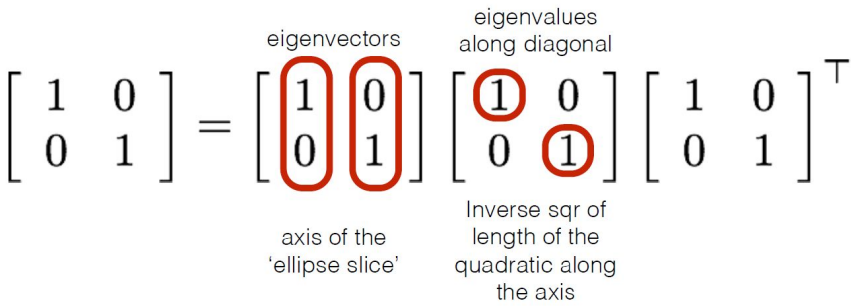
Visualizing quadratics

$$f(x, y) = x^2 + y^2$$

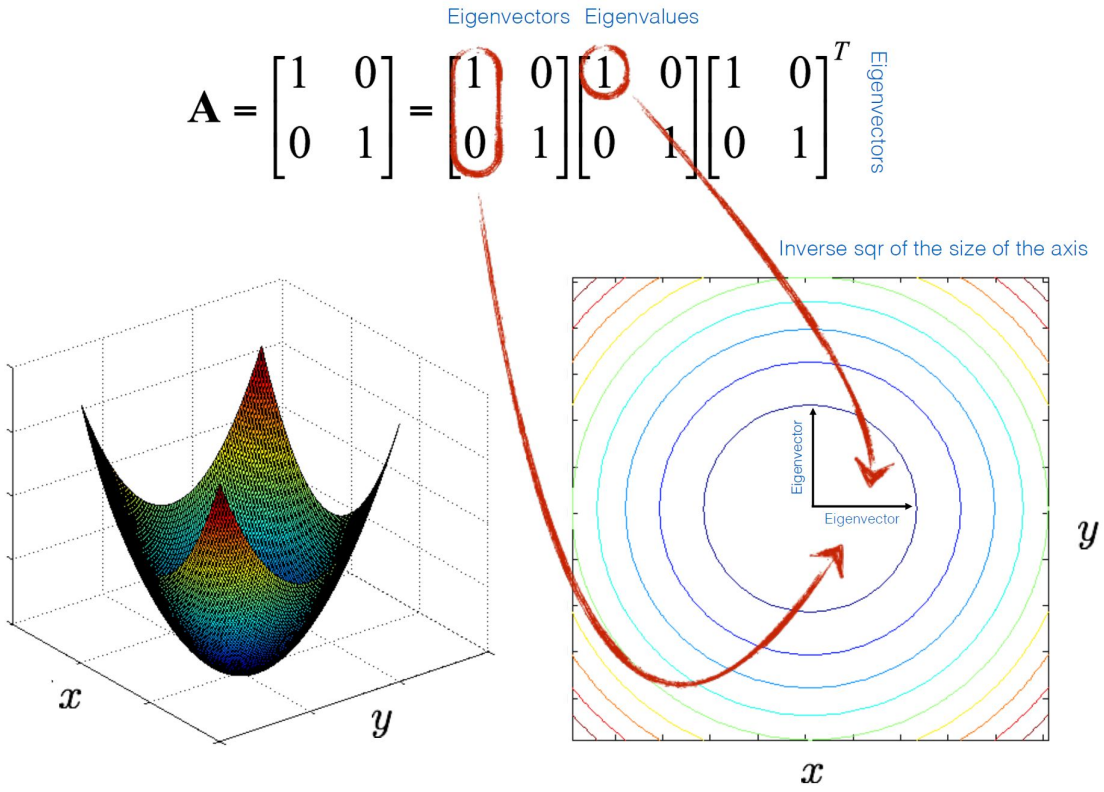
can be written in matrix form like this…

$$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Result of Singular Value Decomposition (SVD)

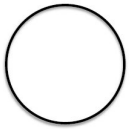


Visualizing quadratics



Visualizing quadratics

Recall:

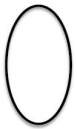


$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

you can smash this bowl in the **y** direction

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

you can smash this bowl in the **x** direction



$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Visualizing quadratics

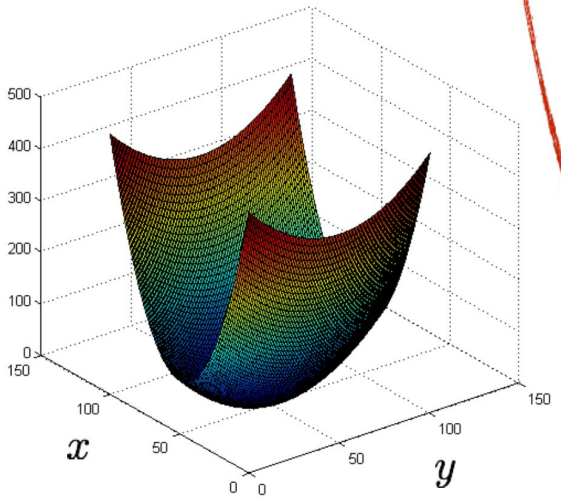
$$\mathbf{A} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T$$

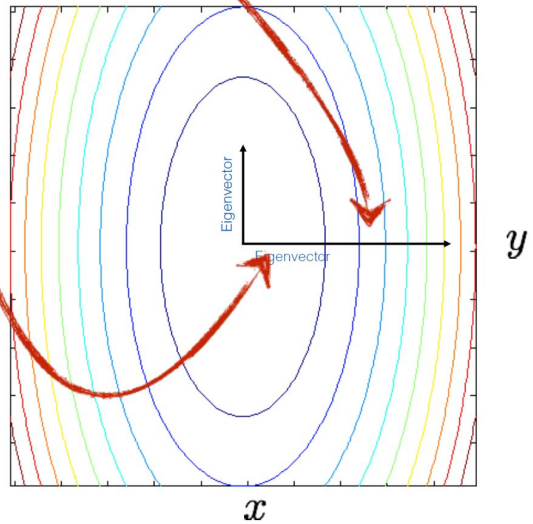
Eigenvalues

Eigenvectors

Eigenvectors

Inverse sqr of length of axis





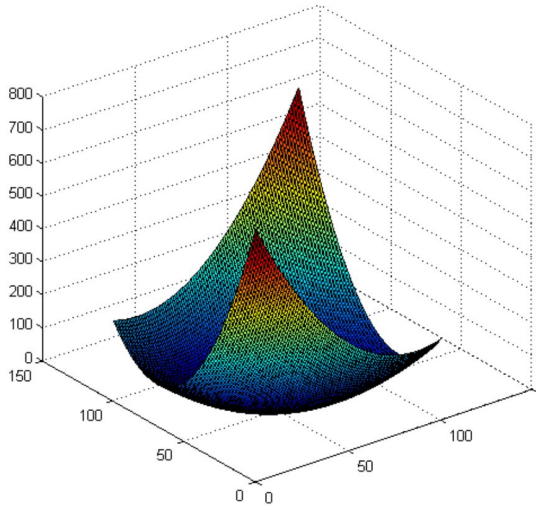
Visualizing quadratics

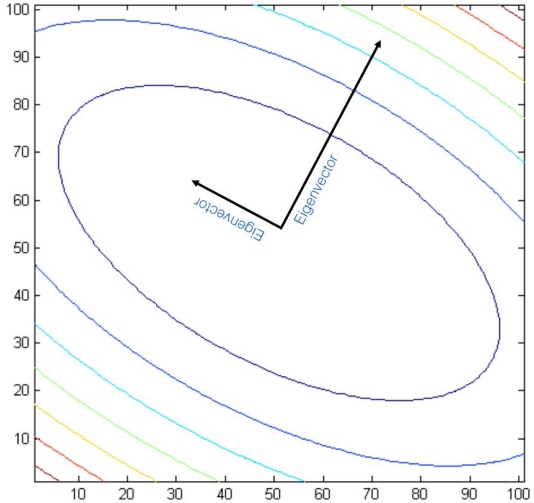
$$\mathbf{A} = \begin{bmatrix} 3.25 & 1.30 \\ 1.30 & 1.75 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T$$

Eigenvalues

Eigenvectors

Eigenvectors





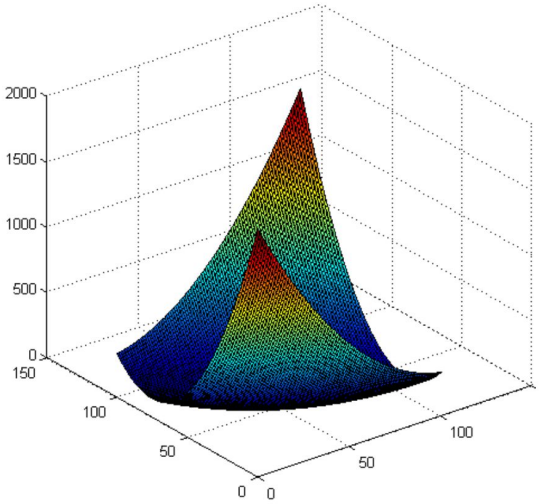
Visualizing quadratics

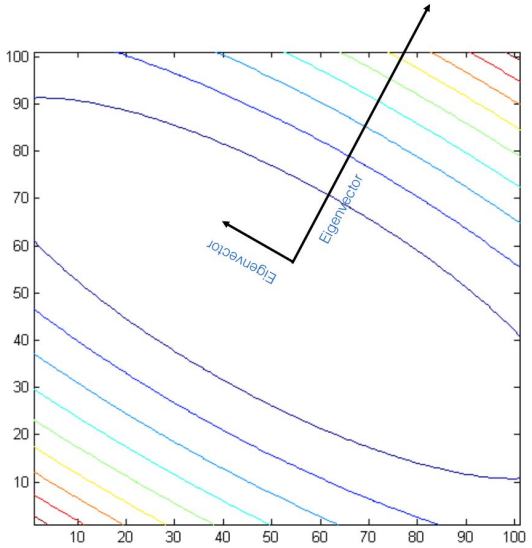
$$\mathbf{A} = \begin{bmatrix} 7.75 & 3.90 \\ 3.90 & 3.25 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T$$

Eigenvalues

Eigenvectors

Eigenvectors





Error function for Harris Corner Detector

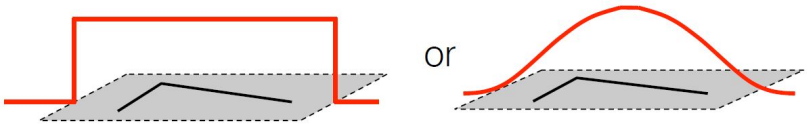
- Let an image be given by *I*.
- Consider taking an image patch over the area (x,y) and shifting it by (u,v).
- The weighted sum of squared differences (SSD) between these two patches, is given by:

Change of intensity for the shift $[u,v]$:

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

Error function | Window function | Shifted intensity | Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

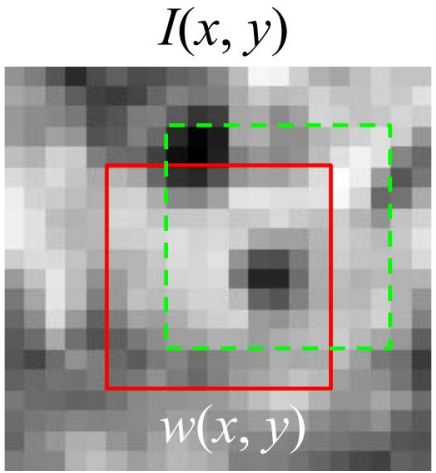
or

Gaussian

Error function for Harris Corner Detector

Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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



Error function approximation for Harris Corner Detector

Taylor Series: $f(x)$ Can be represented at point a in terms of its derivatives

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f^{(3)}(a)}{3!}(x-a)^3+\cdots$$

$$f(x,y)\approx L(x,y)=f(a,b)+f_x(a,b)(x-a)+f_y(a,b)(y-b)+\ldots$$

Error function approximation for Harris Detector

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \; [\underbrace{I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \; [\underbrace{I(x,y) + uI_x + vI_y}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

Taylor Series

$$E(u,v) = \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$E(u,v) = \sum_{x,y} w(x,y)\left[\begin{pmatrix} u & v \end{pmatrix}\begin{pmatrix} I_x \\ I_y \end{pmatrix}\right]^2$$

$$E(u,v) = \sum_{x,y} w(x,y)\begin{pmatrix} u & v \end{pmatrix}\begin{pmatrix} I_x \\ I_y \end{pmatrix}\begin{pmatrix} I_x & I_y \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix}$$

$$E(u,v) = \begin{pmatrix} u & v \end{pmatrix}\left[\sum_{x,y} w(x,y)\begin{pmatrix} I_x \\ I_y \end{pmatrix}\begin{pmatrix} I_x & I_y \end{pmatrix}\right]\begin{pmatrix} u \\ v \end{pmatrix} \qquad E(u,v) = \begin{pmatrix} u & v \end{pmatrix} M \begin{pmatrix} u \\ v \end{pmatrix}$$

Error function approximation for Harris Detector

For small shifts $[u,v]$

Change in appearance for a shift [u,v]

$$E(u,v) \cong \begin{bmatrix} u,v \end{bmatrix} \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

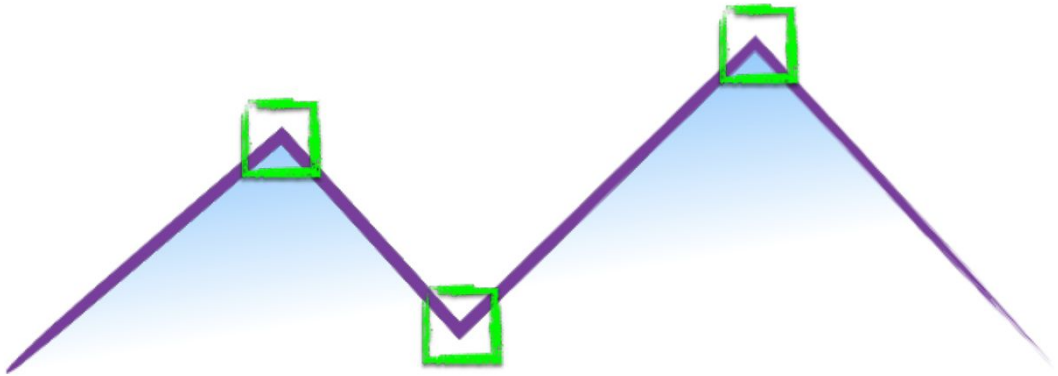
where M is a 2×2 matrix computed from image derivatives:

'second moment' matrix
'structure tensor'

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

How do you find a corner?

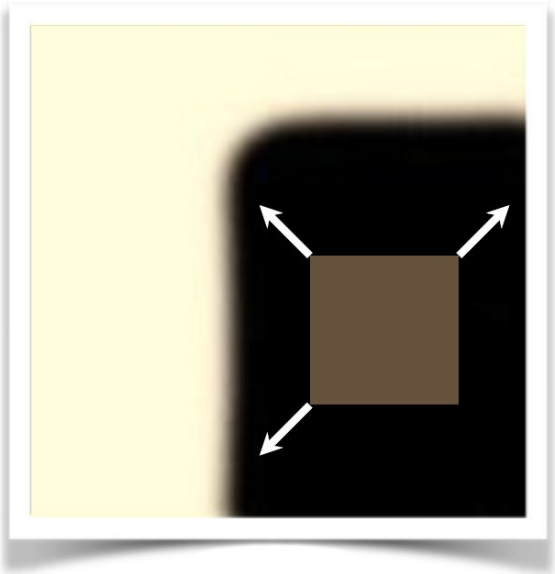
[Moravec 1980]



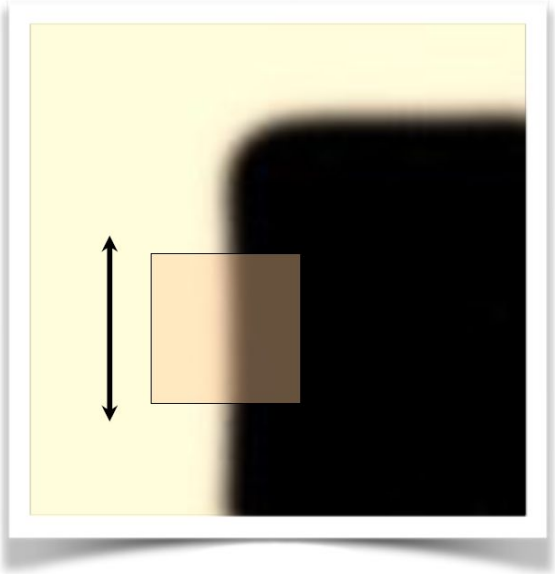
Recognize corners by looking at small window.

Shifting the window in any direction should give a large change in intensity in all directions.

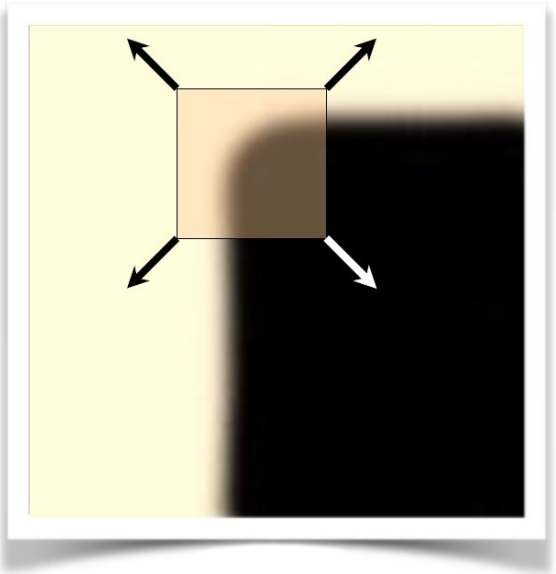
How do you find a corner?



"flat" region:
no change in all directions



"edge":
no change along the edge direction



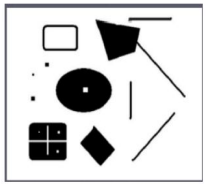
"corner":
significant change in all directions

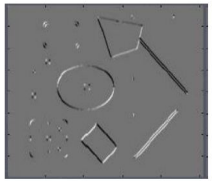
Design a program to detect corners

(hint: use image gradients)

Finding corners using Harris Corner Detector

1. Compute image gradients over a small region/window.
2. Compute the covariance matrix = M (Structure Tensor)
3. Compute eigenvectors and eigenvalues.
4. Use threshold on eigenvalues or on a function of eigenvalues to detect corners





Notation:

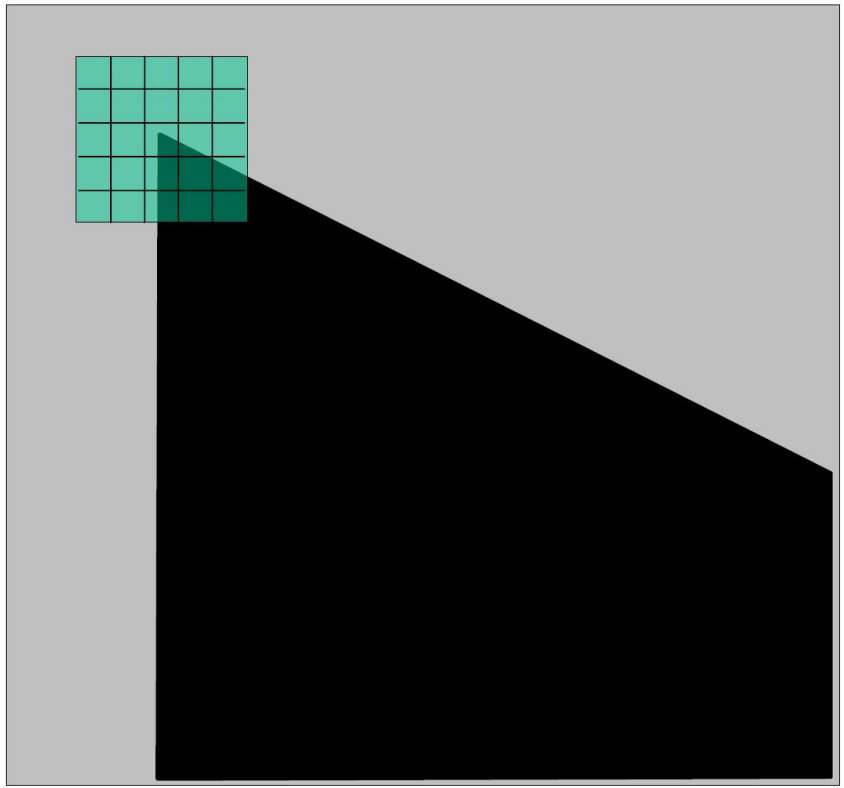
$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

$$\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

1. Compute image gradients over a small region

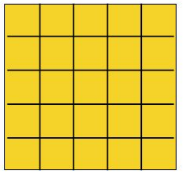
(not just a single pixel)

1. Compute image gradients over a small region



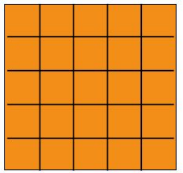
array of x gradients

$$I_x = \frac{\partial I}{\partial x}$$

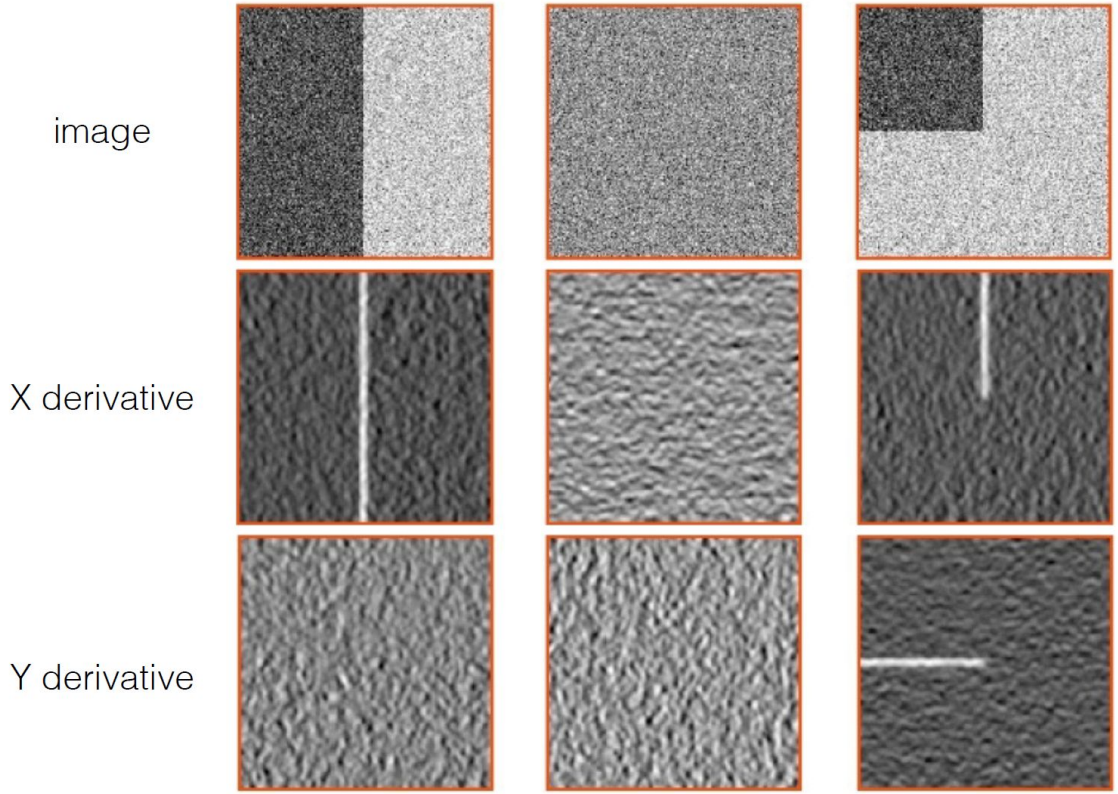


array of y gradients

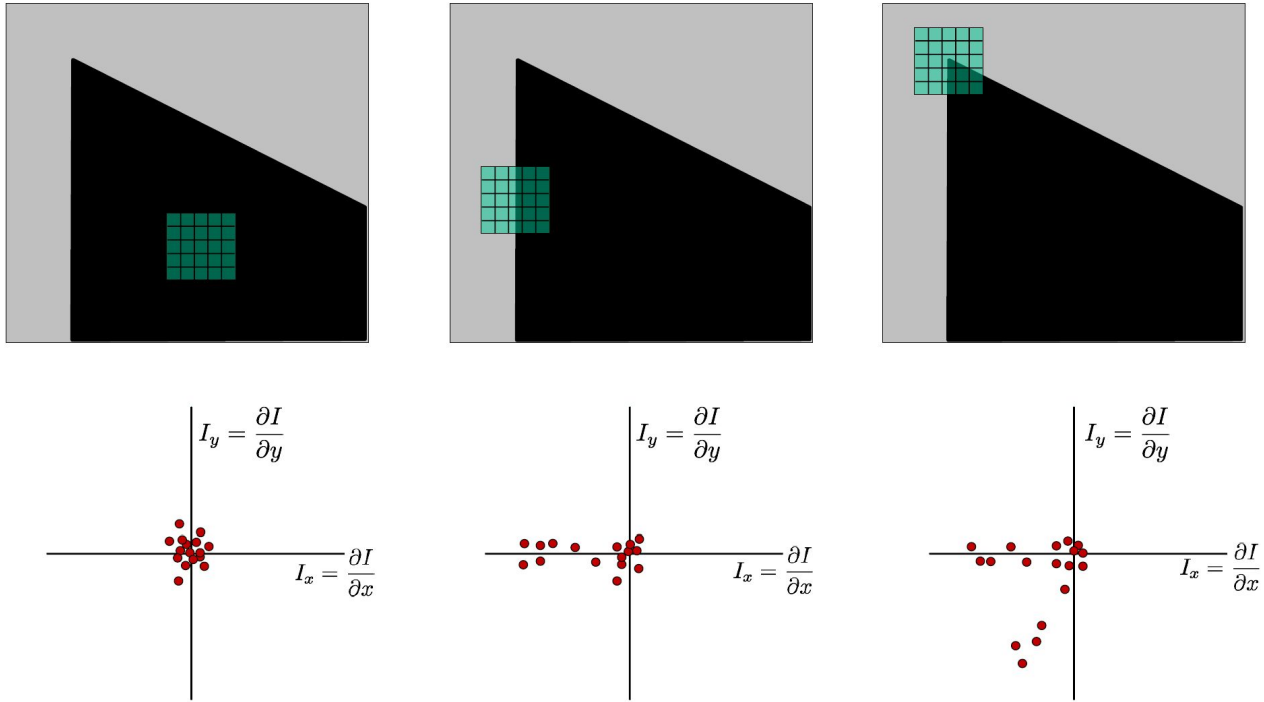
$$I_y = \frac{\partial I}{\partial y}$$



Visualization of gradients

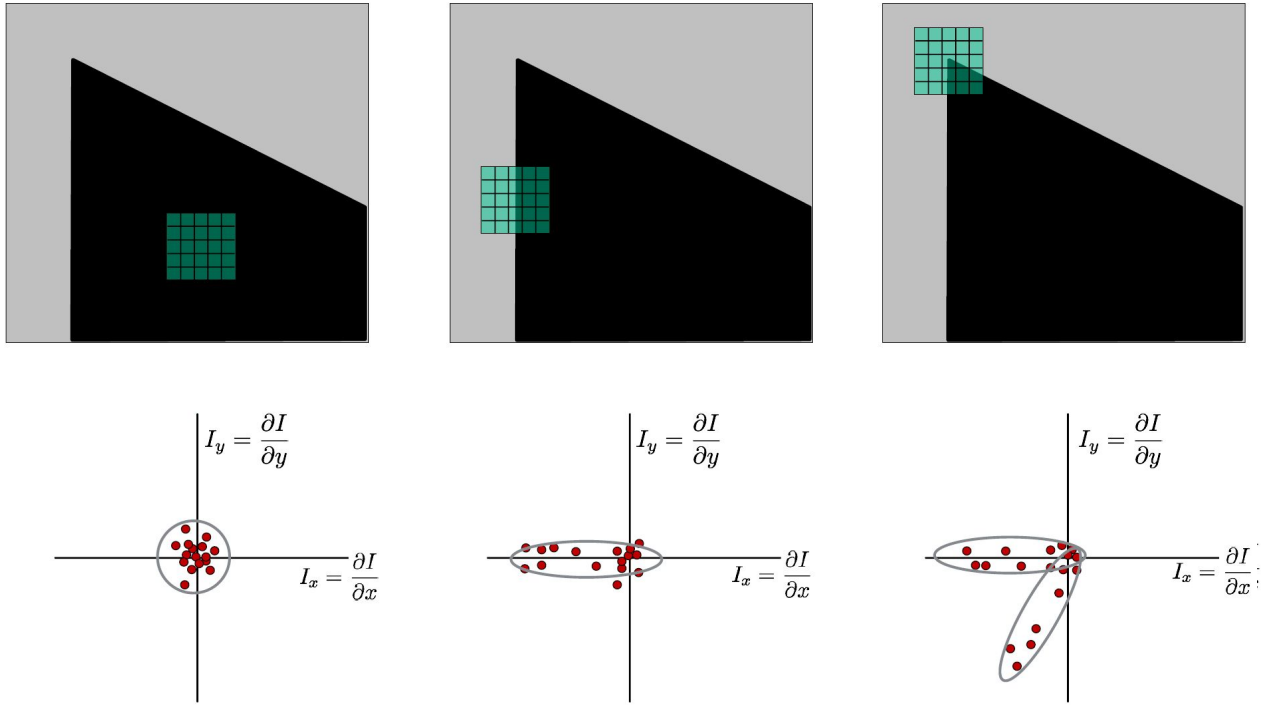


Visualization of gradients



What does the distribution tell you about the region?

Visualization of gradients



distribution reveals edge orientation and magnitude

2. Compute the covariance matrix

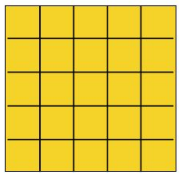
2. Compute the covariance matrix

$$\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

$p \in P$ denotes all pixels in the window

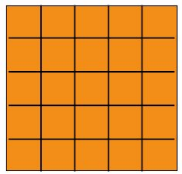
$$\sum_{p \in P} I_x I_y = \text{sum}(\quad * \quad)$$

$I_x = \frac{\partial I}{\partial x}$



array of x gradients

$I_y = \frac{\partial I}{\partial y}$

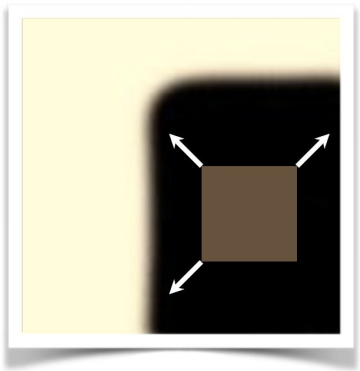


array of y gradients

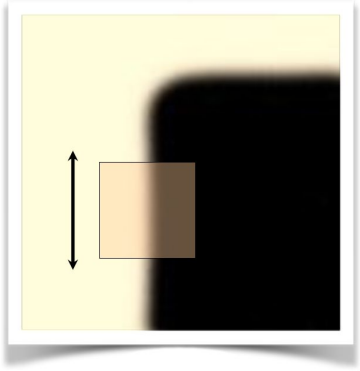
2. Compute the covariance matrix

Easily recognized by looking through a small window

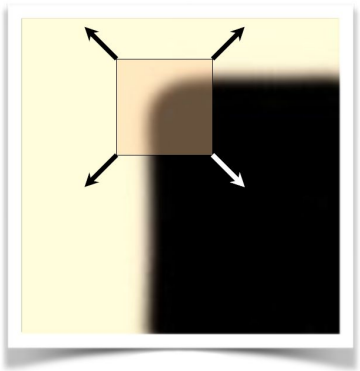
Shifting the window should give large change in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

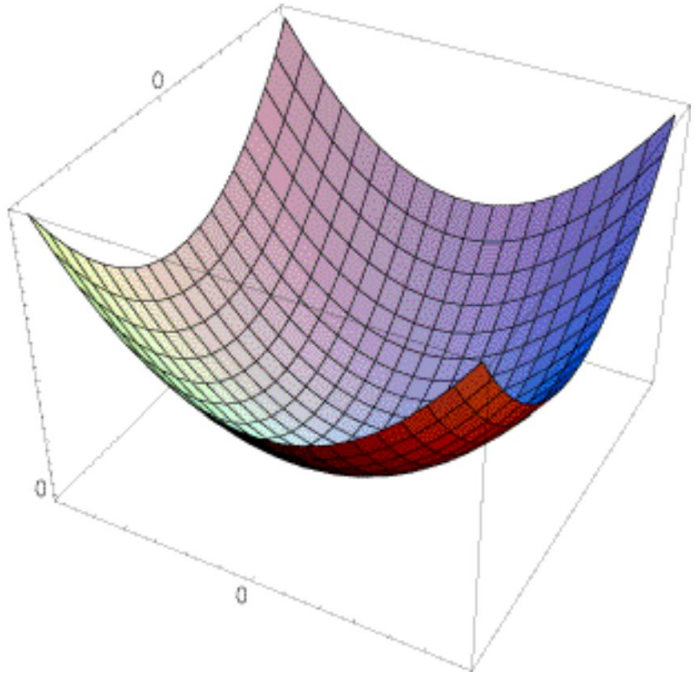
[Moravec 1980]

Visualization of a quadratic

The surface $E(u,v)$ is locally approximated by a quadratic form

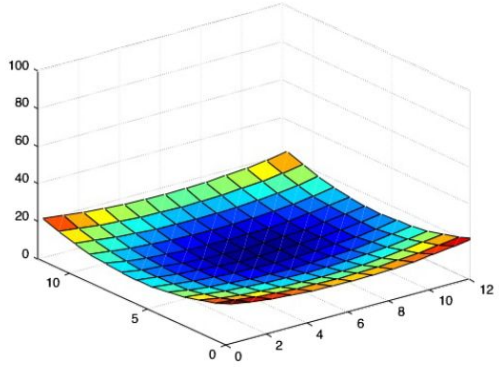
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

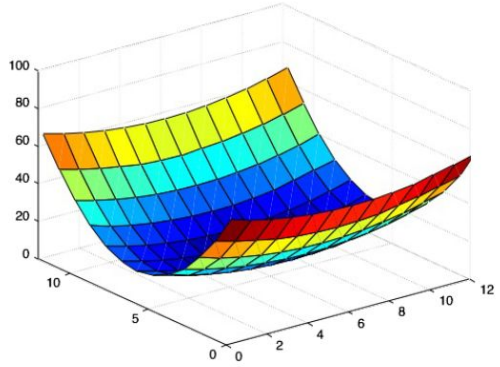
$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

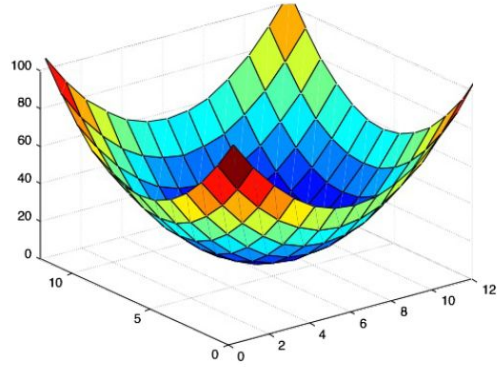


Visualization of a quadratic

Which error surface indicates a good image feature?



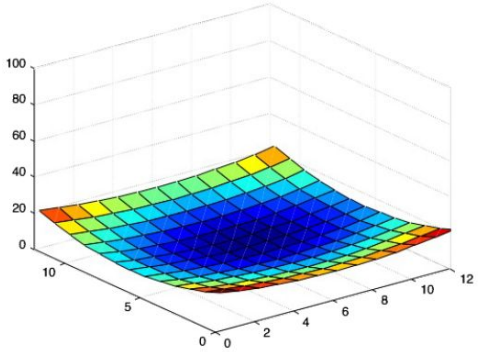




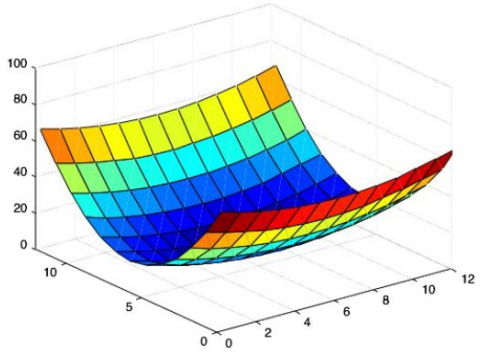
What kind of image patch do these surfaces represent?

Visualization of a quadratic

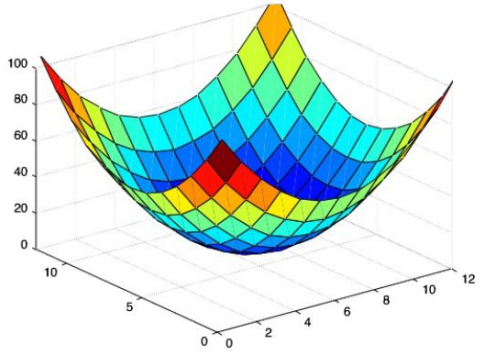
Which error surface indicates a good image feature?



flat



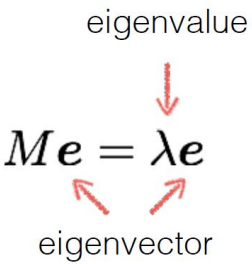
edge
'line'



corner
'dot'

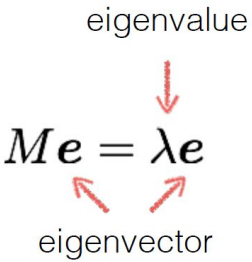
3. Compute eigenvalues and eigenvectors

3. Compute eigenvalues and eigenvectors



$$(M - \lambda I)e = 0$$

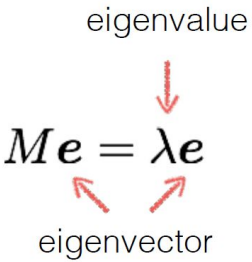
3. Compute eigenvalues and eigenvectors



$$(M - \lambda I)e = 0$$

1. Compute the determinant of $M - \lambda I$
(returns a polynomial)

3. Compute eigenvalues and eigenvectors

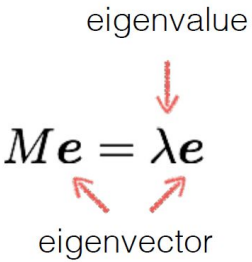


$$(M - \lambda I)e = 0$$

1. Compute the determinant of $M - \lambda I$
   (returns a polynomial)

2. Find the roots of polynomial $\det(M - \lambda I) = 0$
   (returns eigenvalues)

3. Compute eigenvalues and eigenvectors



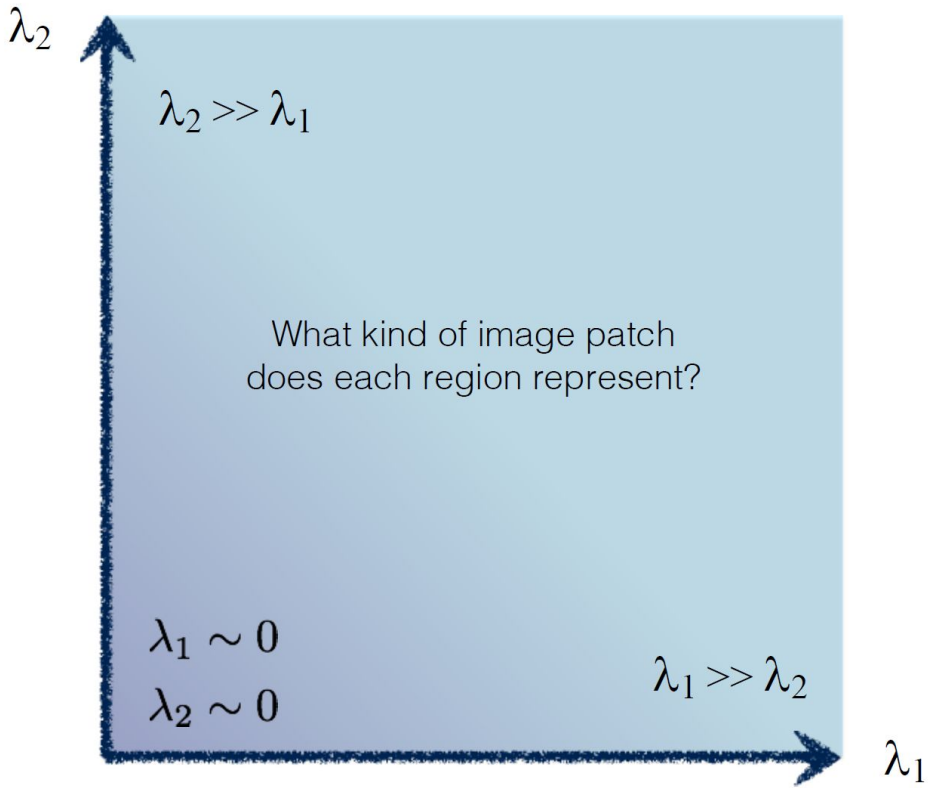
$$(M - \lambda I)e = 0$$

1. Compute the determinant of $M - \lambda I$
   (returns a polynomial)

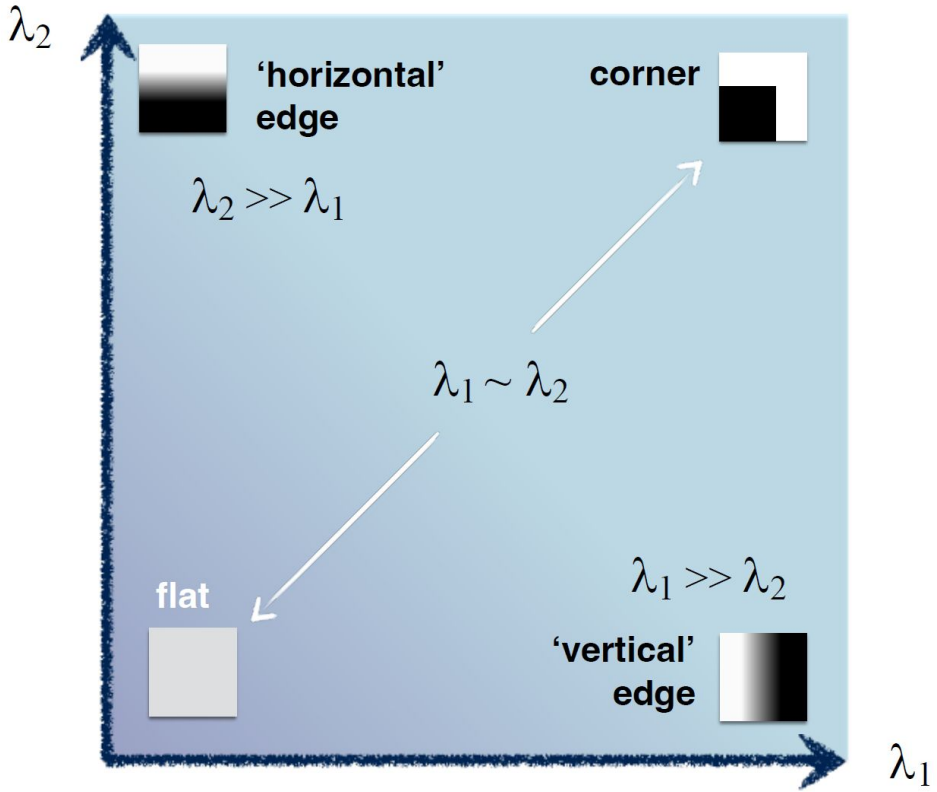
2. Find the roots of polynomial $\det(M - \lambda I) = 0$
   (returns eigenvalues)

3. For each eigenvalue, solve $(M - \lambda I)e = 0$
   (returns eigenvectors)

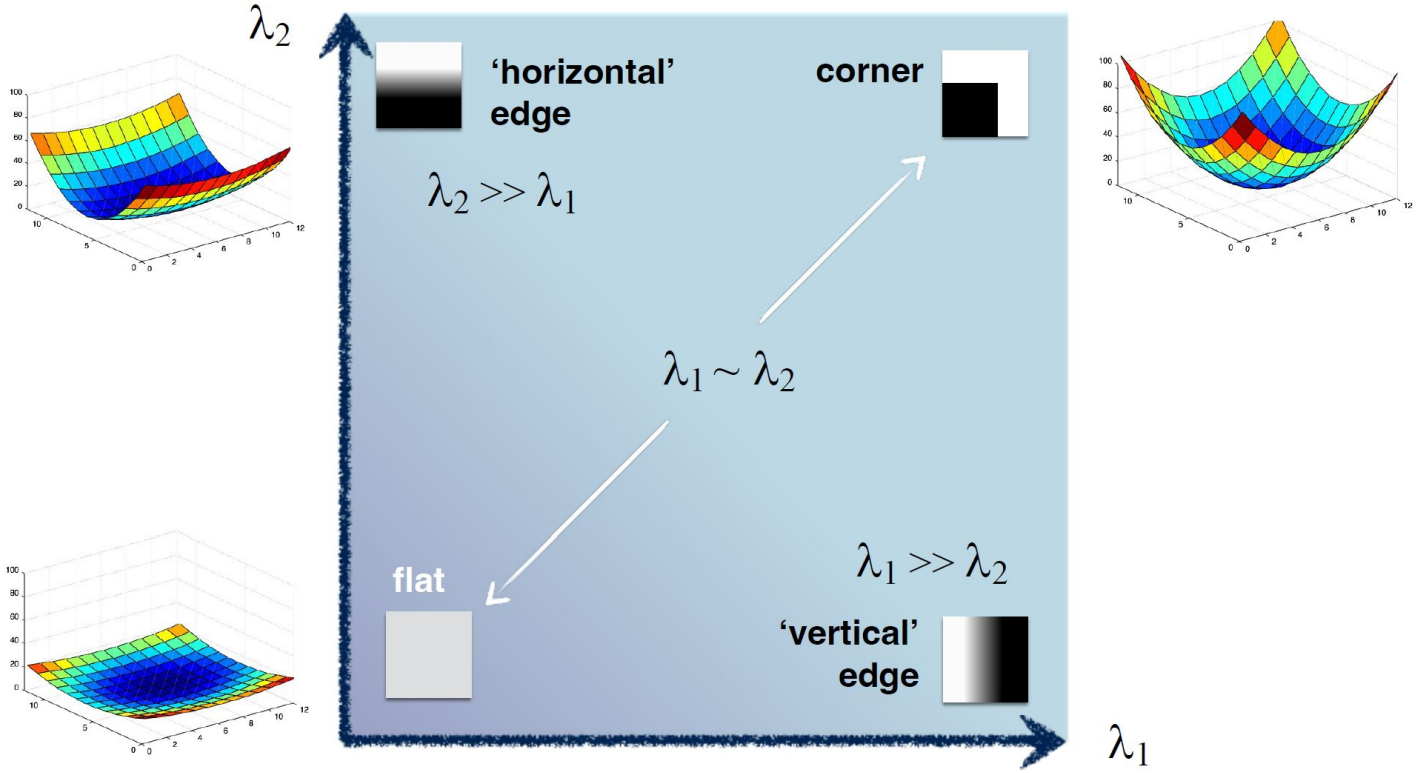
Interpreting eigenvalues



Interpreting eigenvalues

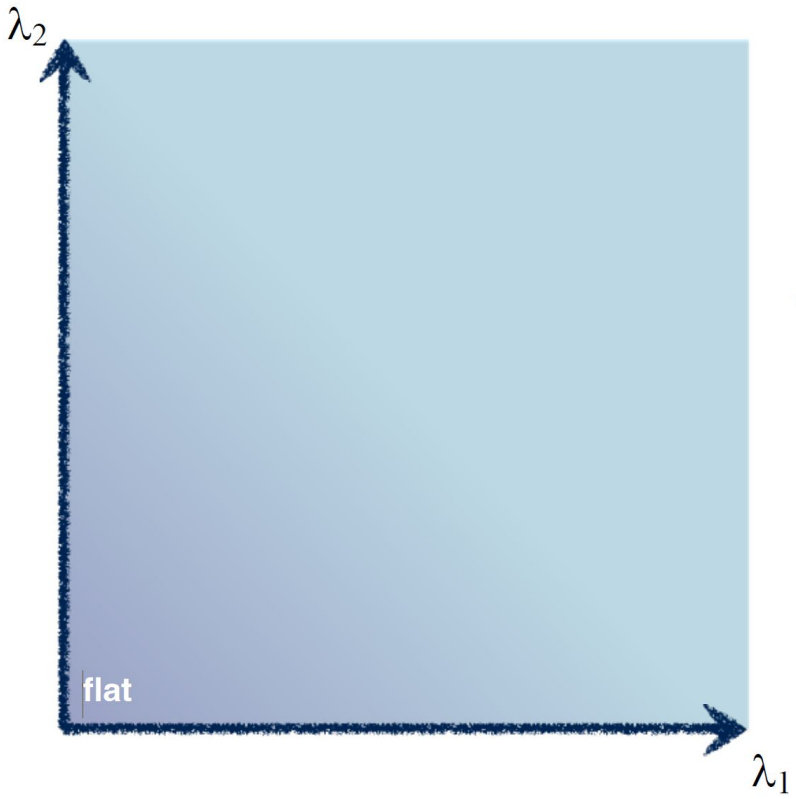


Interpreting eigenvalues



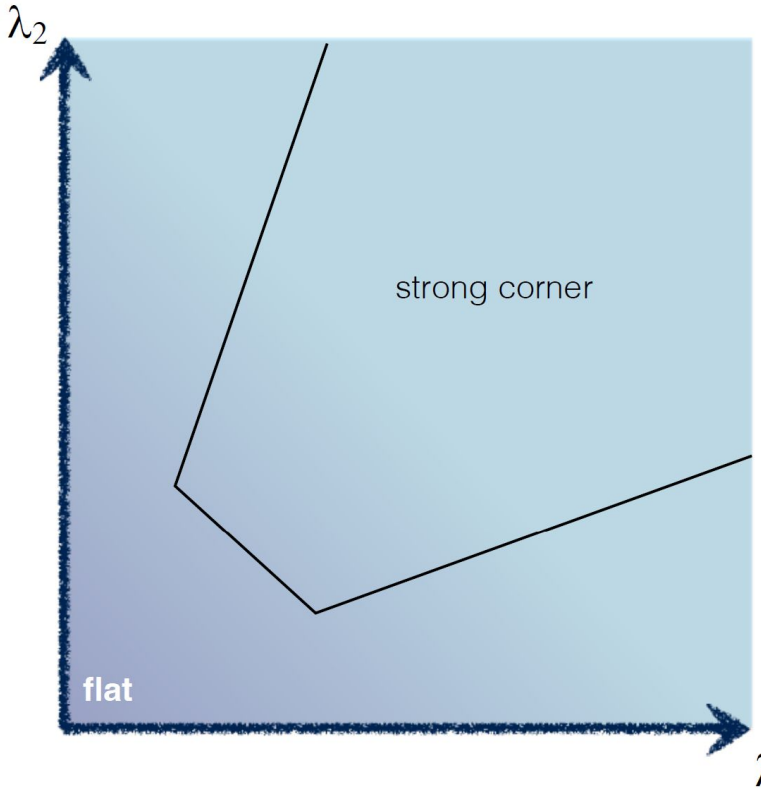
4. Use threshold on eigenvalues or a on function of eigenvalues to detect corners

4. Use threshold on eigenvalues to detect corners



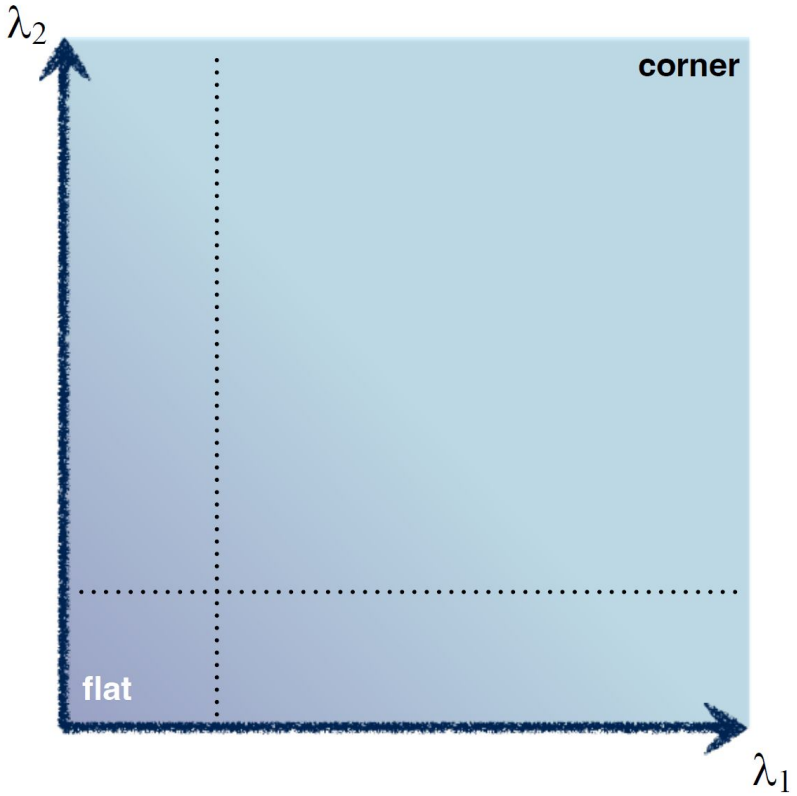
Think of a function to score ‘cornerness’

4. Use threshold on eigenvalues to detect corners



Think of a function to score ‘cornerness’

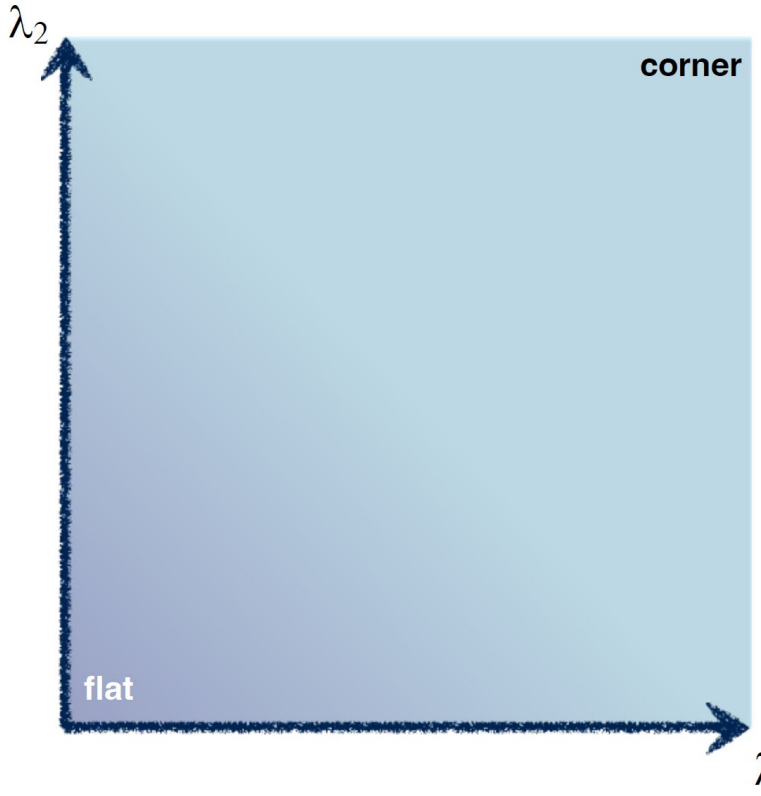
4. Use threshold on “a function of” eigenvalues to detect corners



Use the smallest eigenvalue
as the response function

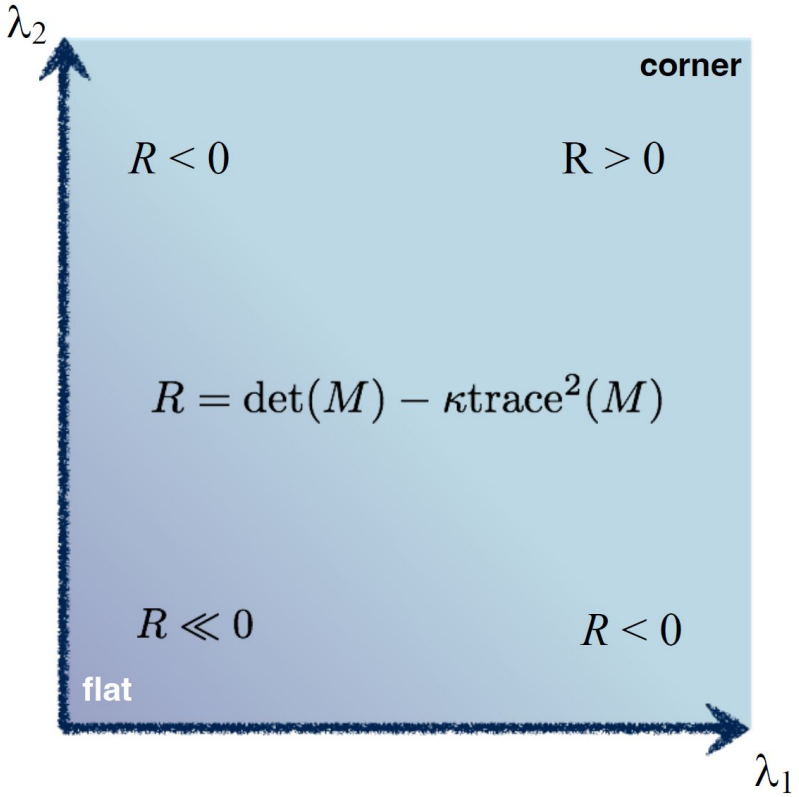
$$R = \min(\lambda_1, \lambda_2)$$

4. Use threshold on “a function of” eigenvalues to detect corners



$$R = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

4. Use threshold on "a function of" eigenvalues to detect corners



$$\det M = \lambda_1 \lambda_2$$

$$\text{trace}\, M = \lambda_1 + \lambda_2$$

$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

Harris & Stephens (1988)

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

κ is usually set between **0.04 and 0.06**.

Kanade & Tomasi (1994)

$$R = \min(\lambda_1, \lambda_2)$$

Nobel (1998)

$$R = \frac{\det(M)}{\text{trace}(M) + \epsilon}$$

Harris Detector

C.Harris and M.Stephens. “A Combined Corner and Edge Detector.”1988.

5. Threshold on R value. Compute non max suppression.

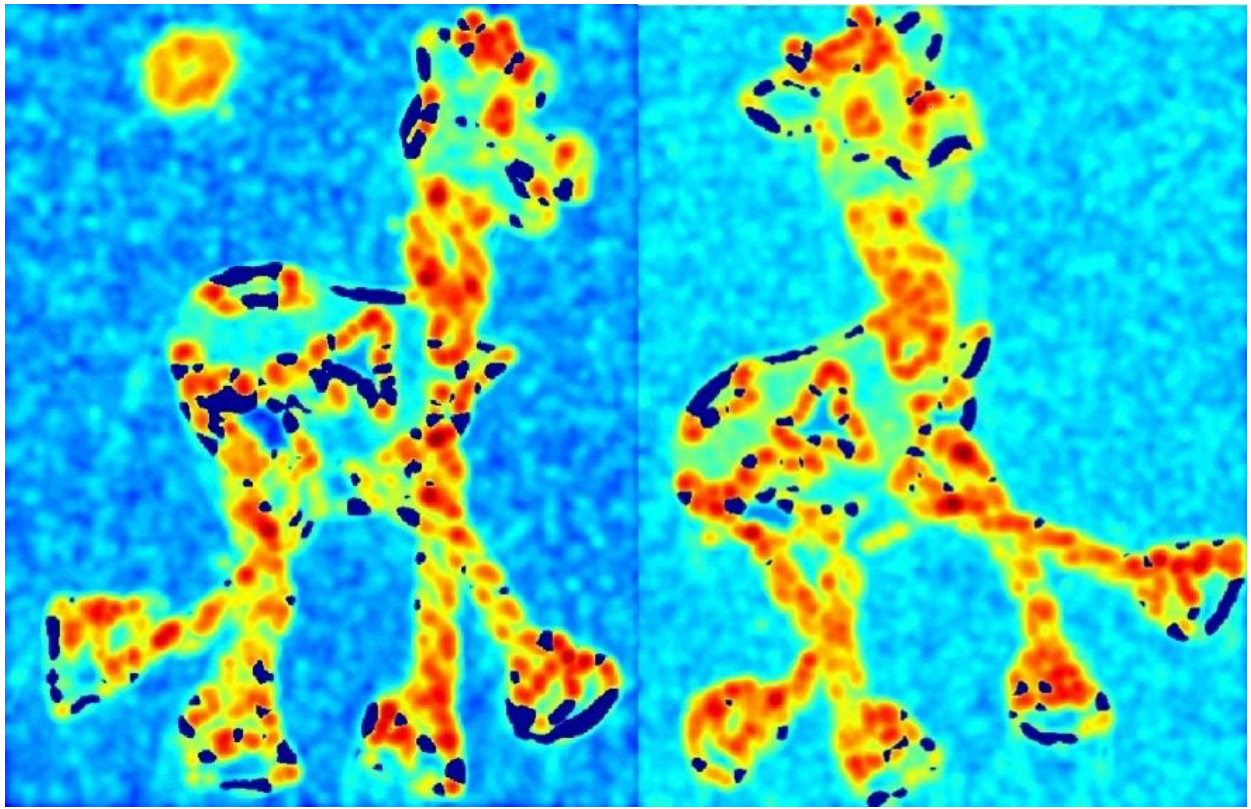
Retain only the pixels that correspond to the local maxima within a NMS window in the corner response map.

The choice of the window size for NMS affects the scale at which non-maximum suppression is applied.

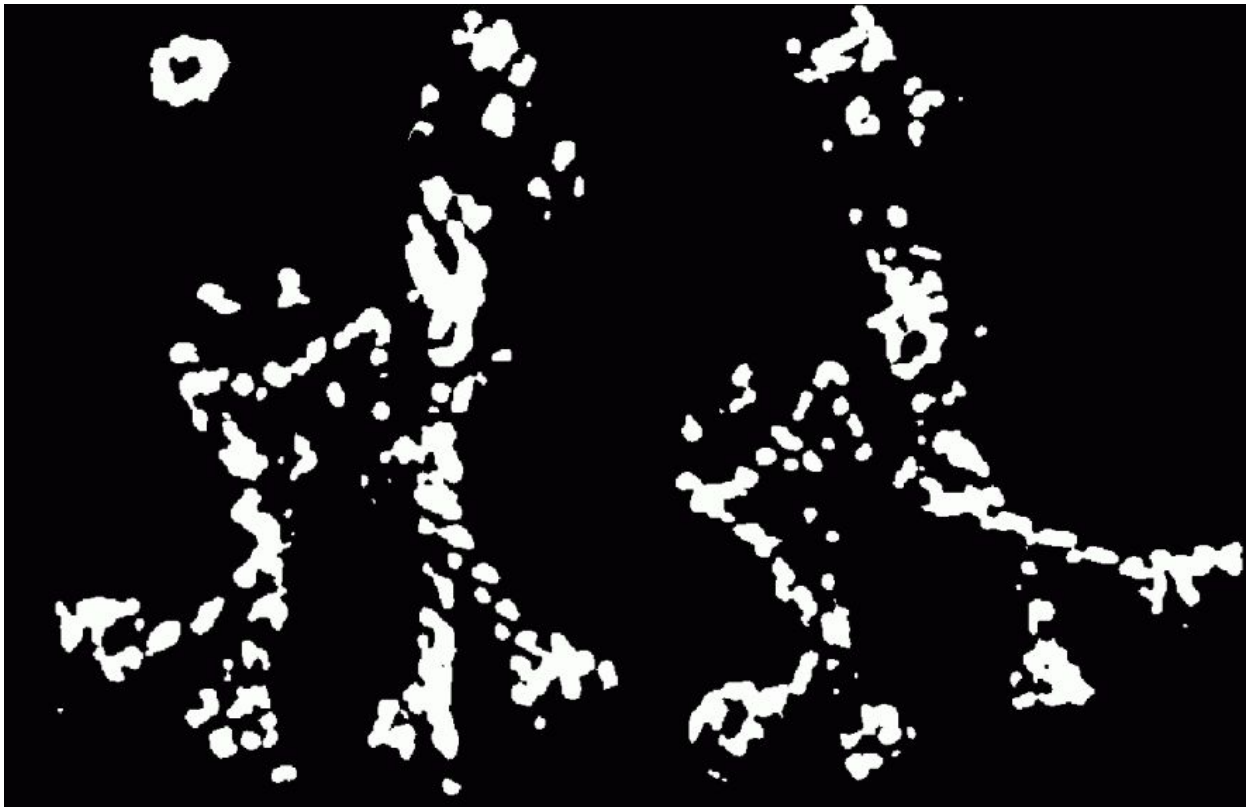
A larger window may lead to the preservation of larger corners, while a smaller window may focus on finer details.

Harris

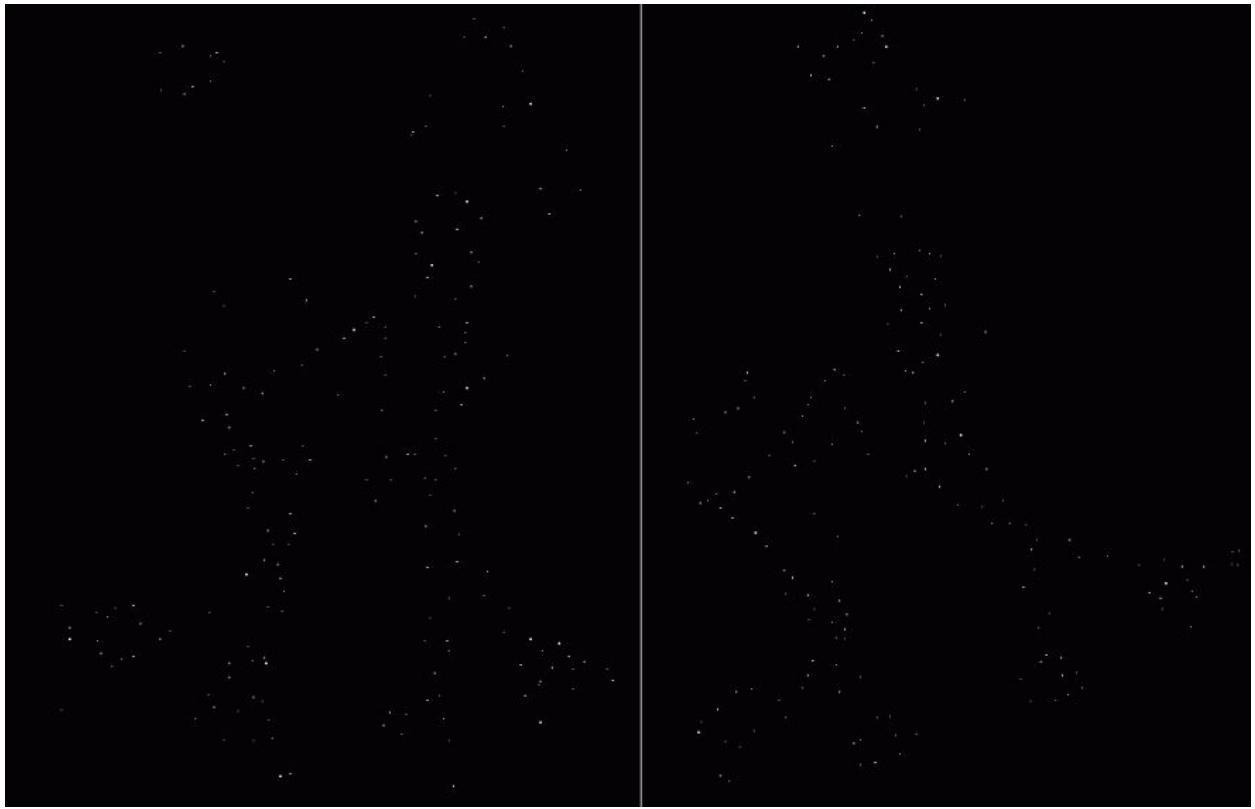
Harris: Corner response



Harris: Thresholded Corner response

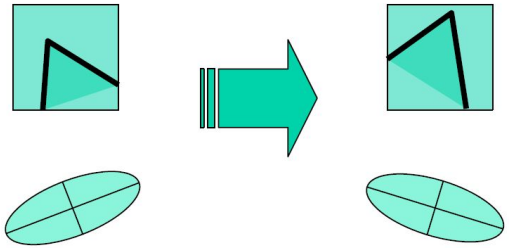


Harris: Non-maximal suppression



Harris corner detection

Harris corner response is
invariant to rotation



Ellipse rotates but its shape
(**eigenvalues**) remains the same

Corner response R is invariant to image rotation

Harris corner detection

The Harris corner detector is not invariant to scale