Computer Vision:
Corner detection

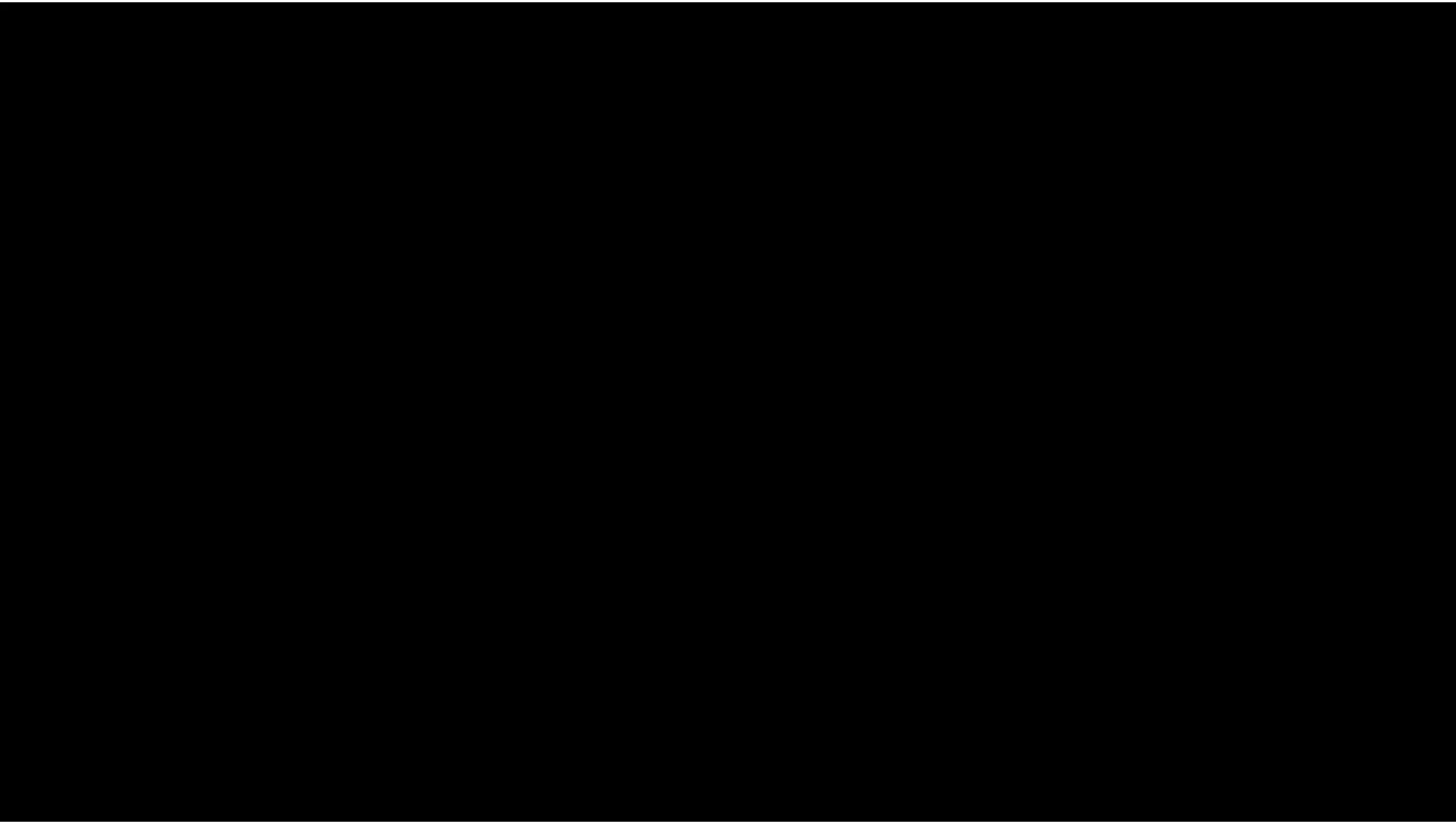
Siheng Chen 陈思衡

Corner detection

Why detect corners

Harris corner detector

Why detect corners



https://www.youtube.com/watch?v=GdPeydPbM0g&list=PLDFDB5B8C80DB3AD6&index=2

Why detect corners

https://www.youtube.com/watch?v=GdPeydPbM0g&list=PLDFDB5B8C80DB3AD6&index=2

Why detect corners

Given two images: how would you align them?

These aren't off by a small 2D translation but instead by a 3D rotation + translation of the camera

Why detect corners

Given two images: how would you align them?

Blend Them Together

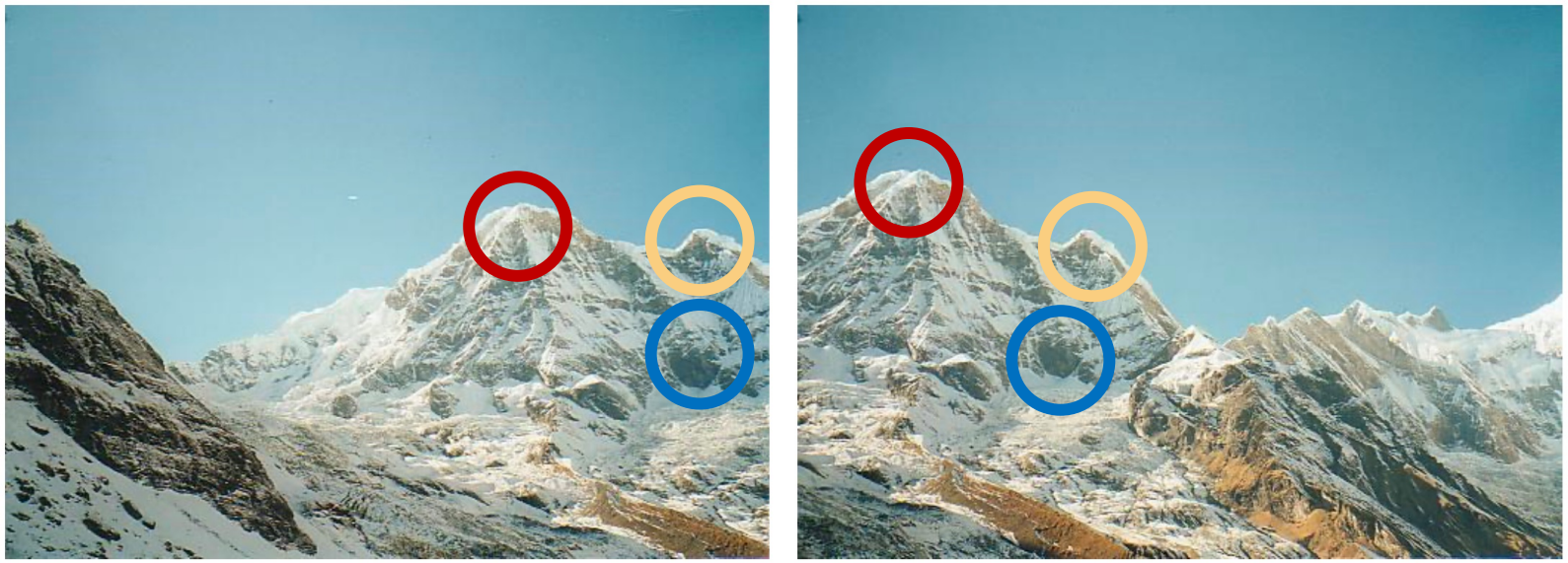
Why detect corners

```
for y in yRange:
    for x in xRange:
        for z in zRange:
            for xRot in xRotVals:
                for yRot in yRotVals:
                    for zRot in zRotVals:
                        #touches all HxW pixels!
                        check_alignment_with_images()
```

Too expensive!

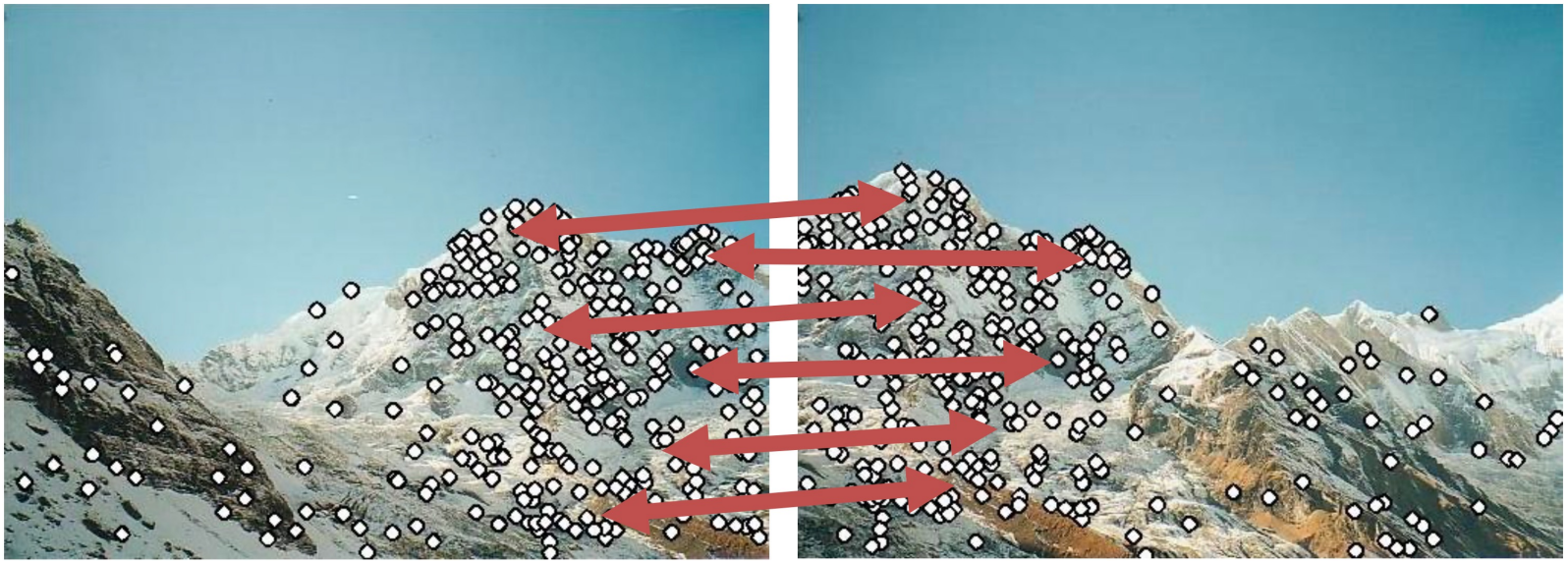
Why detect corners

Given two images: how would you align them?



Why detect corners

Given two images: how would you align them?



1: find corners at each image

2: find the correspondence

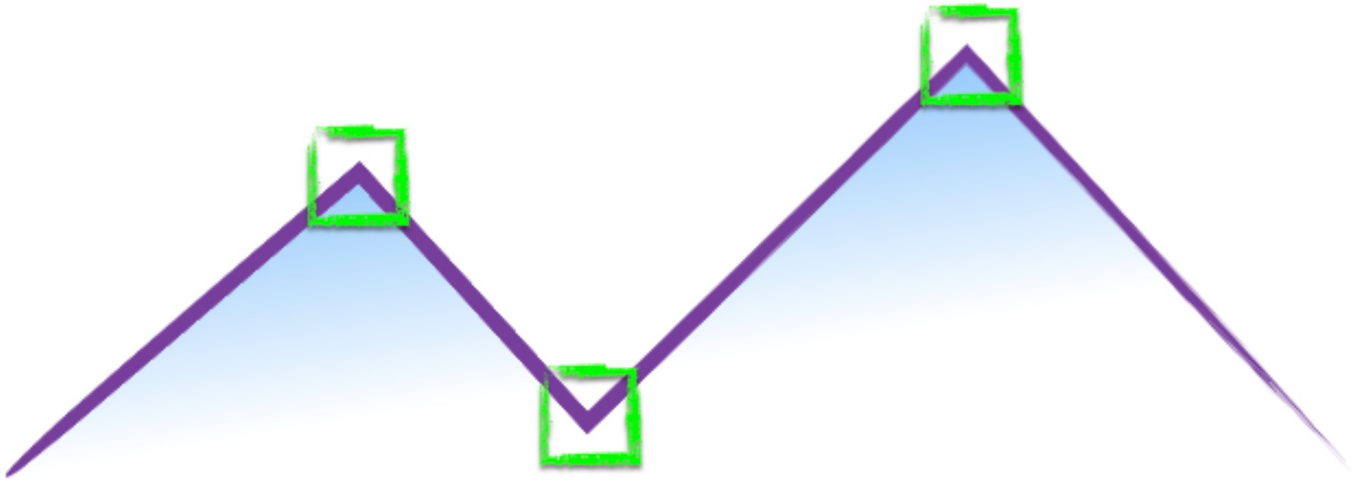
Corner detection

Why detect corners

Harris corner detector

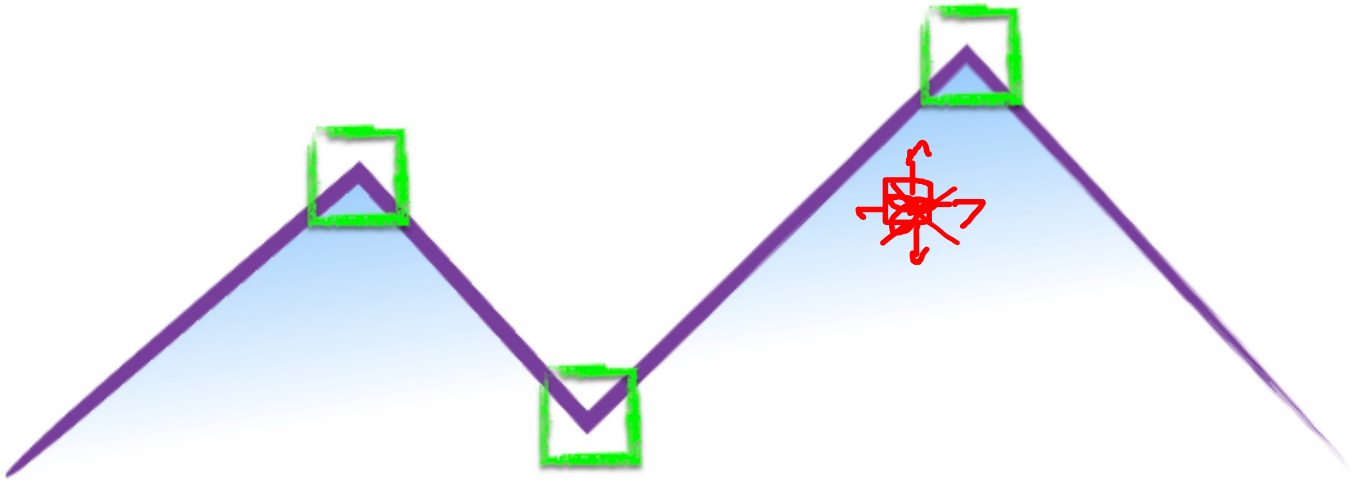
Harris corner detector

How do you find a corner?



Harris corner detector

How do you find a corner?



Easily recognized by looking through a small window.

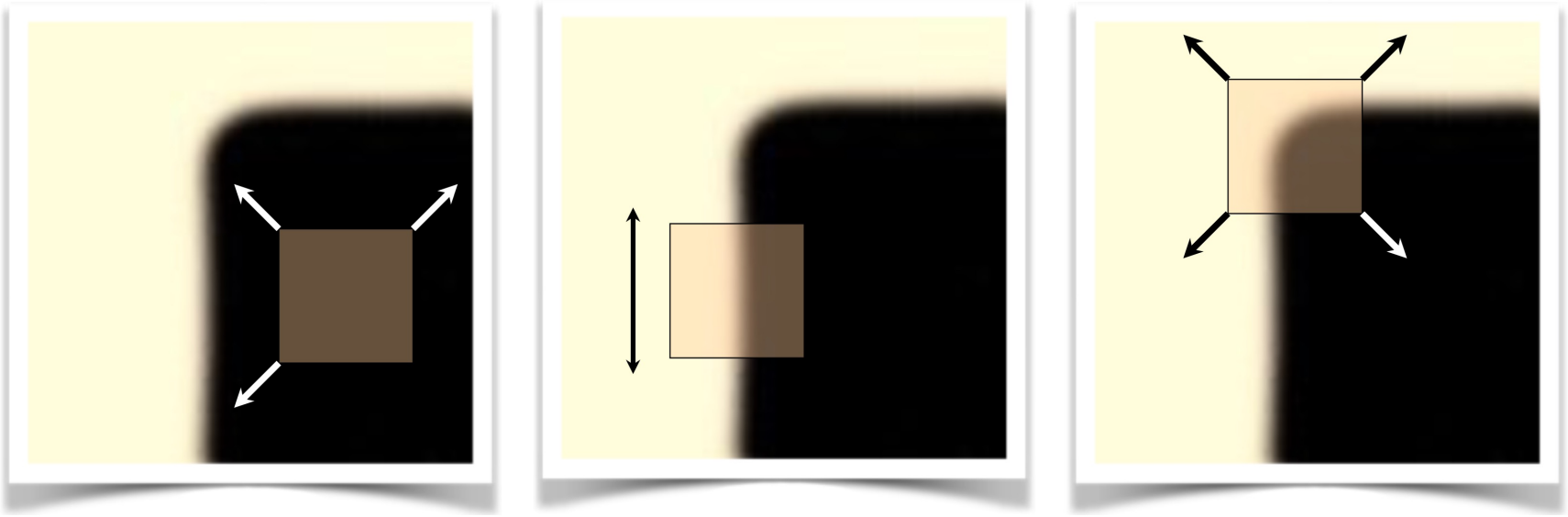
Shifting the window should give large change in intensity

Harris corner detector

"flat" region: no change in all directions

"edge": no change along the edge direction

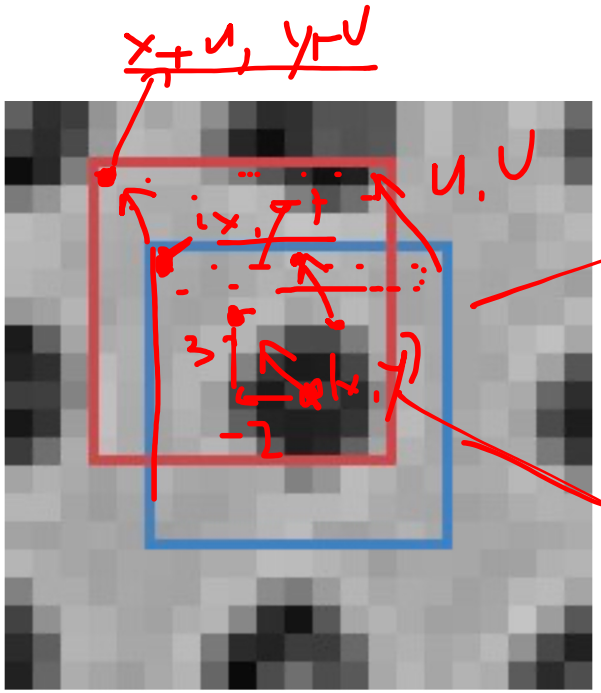
"corner": significant change in all directions

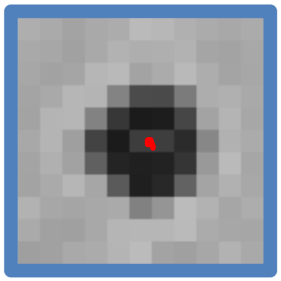


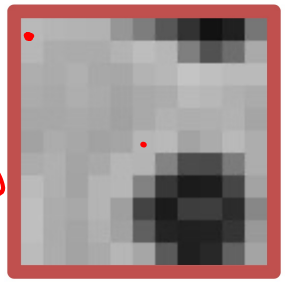
Easily recognized by looking through a small window.

Shifting the window should give large change in intensity

Harris corner detector







"Window"
At x, y

"Window"
At x+u, y+v
Here: u=-2,v=-3

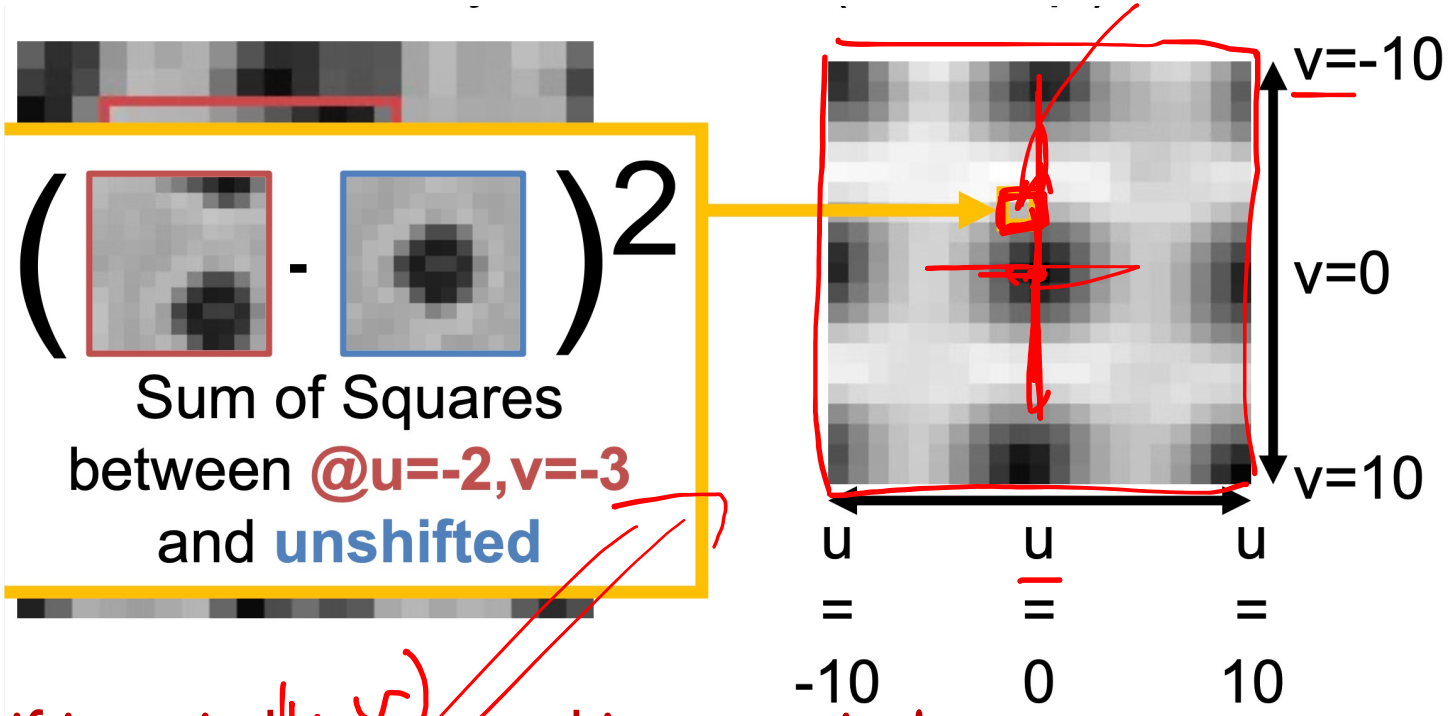
How to measure variation?

$$\left(\text{[image]} - \text{[image]} \right)^2$$

$$\| X_1 - X_2 \|^2$$

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

Harris corner detector



Shifting windows around is expensive!

We'll find a trick to approximate this.

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

Harris corner detector

Taylor Series: linearizing a function

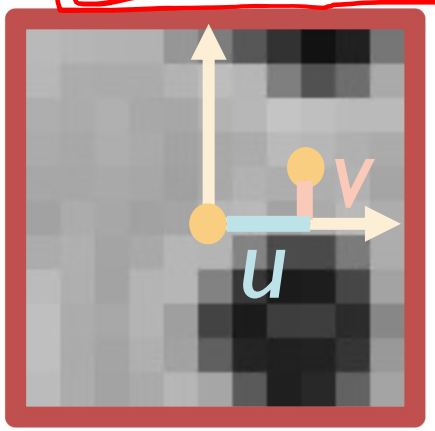
$$f(x+d) \approx f(x) + \frac{\partial f}{\partial x} d$$

Taylor Series for Images

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

gradient along the *y*-axis

gradient along the *x*-axis



Harris corner detector

Taylor Series for Images

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

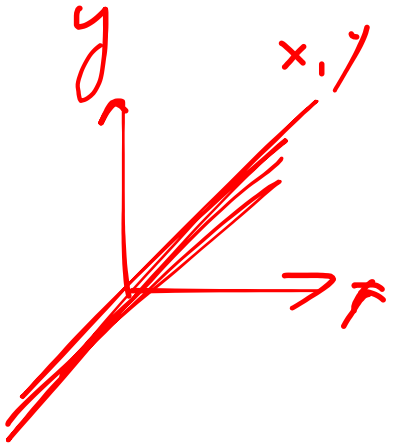
Taylor series

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

algebra

$$= \sum_{(x,y)\in W} (I_x u + I_y v)^2$$

$$= \sum_{(x,y)\in W} I_x^2 u^2 + 2 I_x I_y uv + I_y^2 v^2$$



Harris corner detector

By ***linearizing*** image, we can approximate E(u,v) with quadratic function of u and v

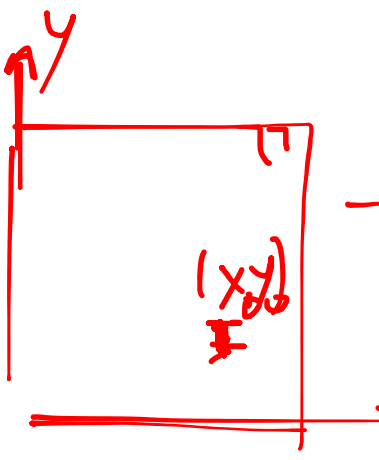
variation

$$E(u,v) \approx \sum_{(x,y)\in W} \left(I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2\right)$$

$$= [u,v]\boldsymbol{M}[u,v]^T$$

second moment matrix

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y\in W} I_x^2 & \sum_{x,y\in W} I_x I_y \\ \sum_{x,y\in W} I_x I_y & \sum_{x,y\in W} I_y^2 \end{bmatrix}$$



$I(x_0, y_0)$ $\left.\frac{\partial I}{\partial y}\right|_{(x_0,y_0)} = I(x_0, y_0+1) - I(x_0, y_0)$

Harris corner detector

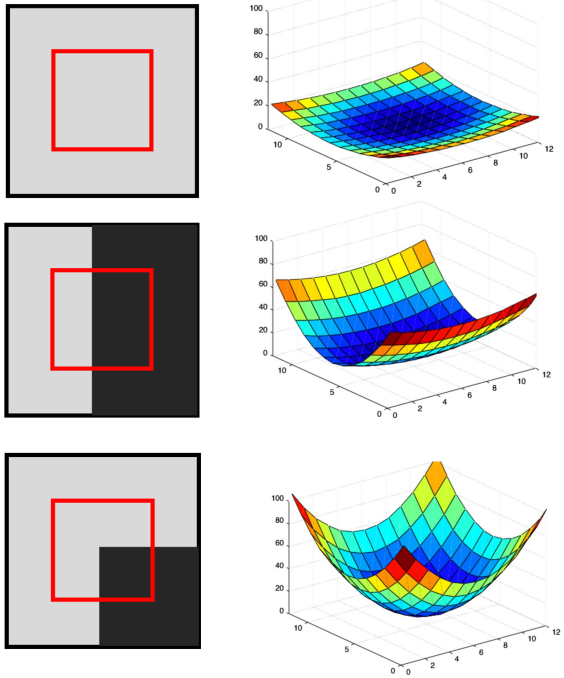
$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

Assume off-diagonal is zero

If a,b are both small: flag

If one is big, one is small: edge

If a,b both big: corner



Harris corner detector

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = \boldsymbol{R}^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \boldsymbol{R}$$

eigen-decomposition

Directions

Magnitude

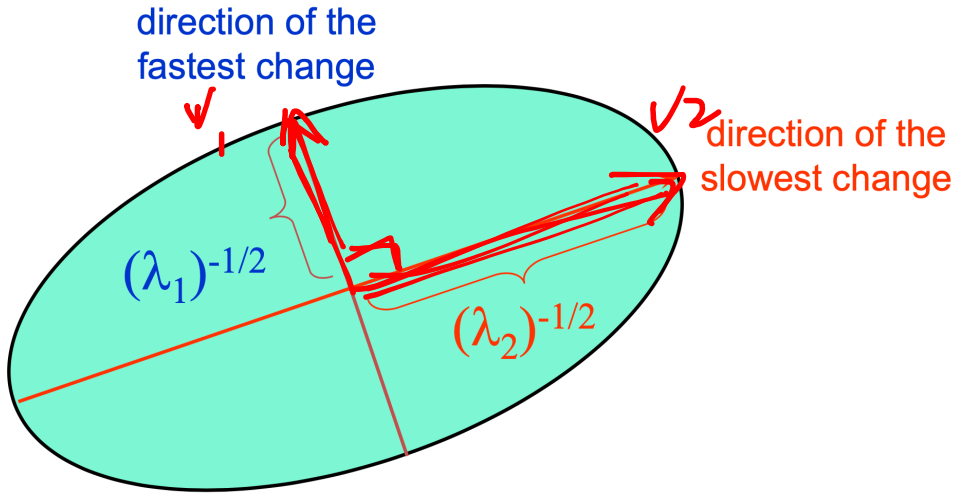
$R \in \mathbb{R}^{2\times 2}$

$M = M^T$

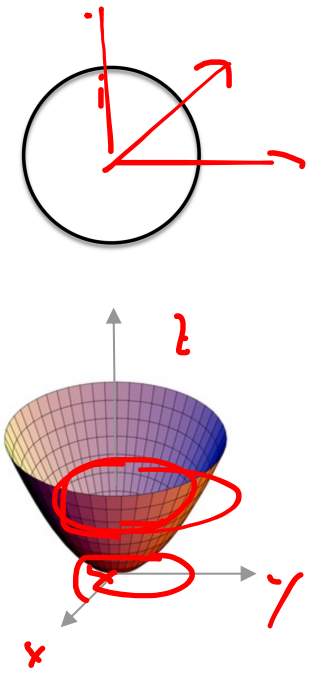
real, sym

$R^{-1} = R^T$

$\therefore \left[\begin{pmatrix} | \\ v_1 \\ | \end{pmatrix} \begin{pmatrix} | \\ v_2 \\ | \end{pmatrix} \right]^T$



Eigen-decomposition



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

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

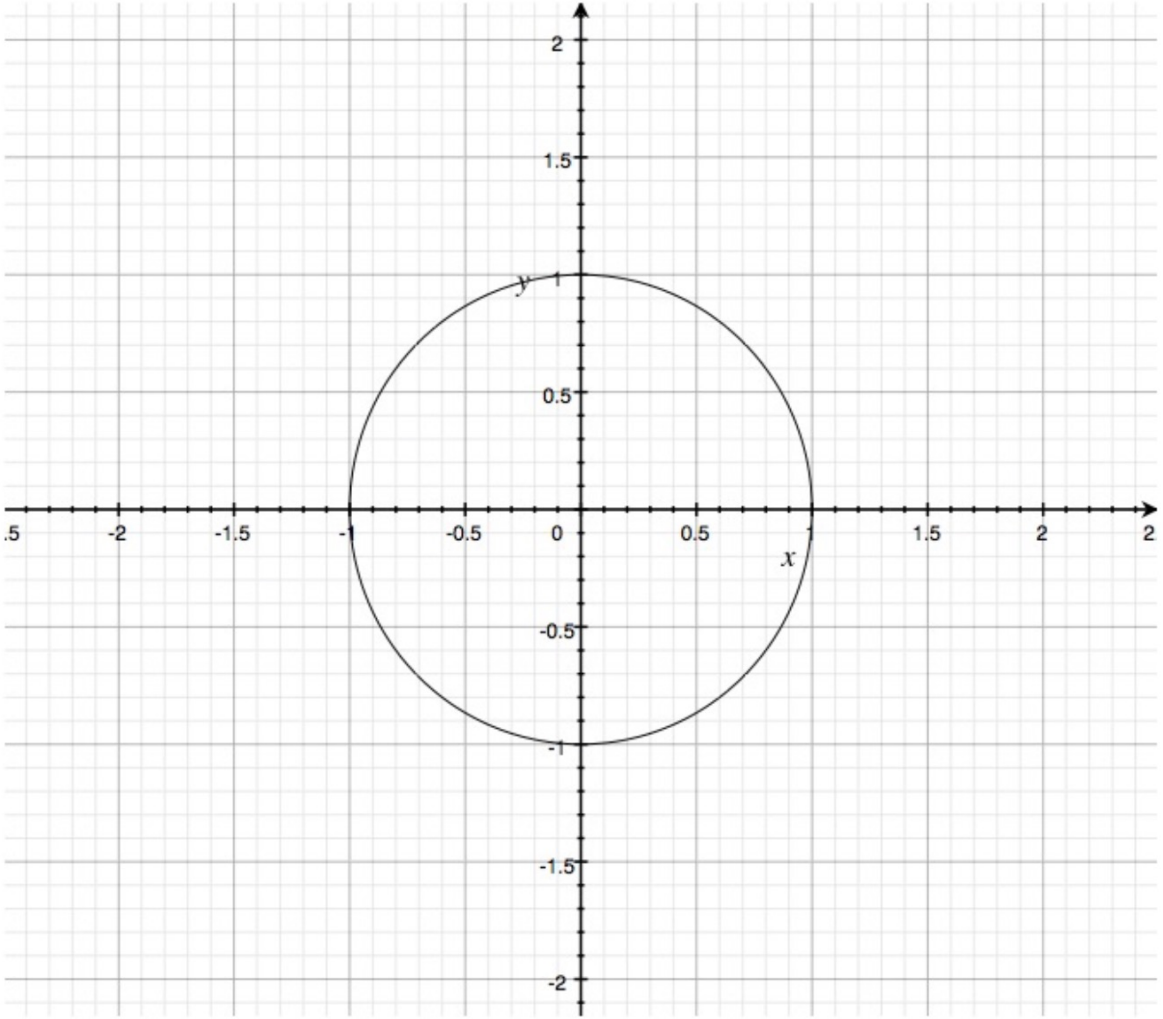
$$= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

if you slice the bowl at $f(x,y) = 1$ what do you get?

Eigen-decomposition

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

'sliced at 1'

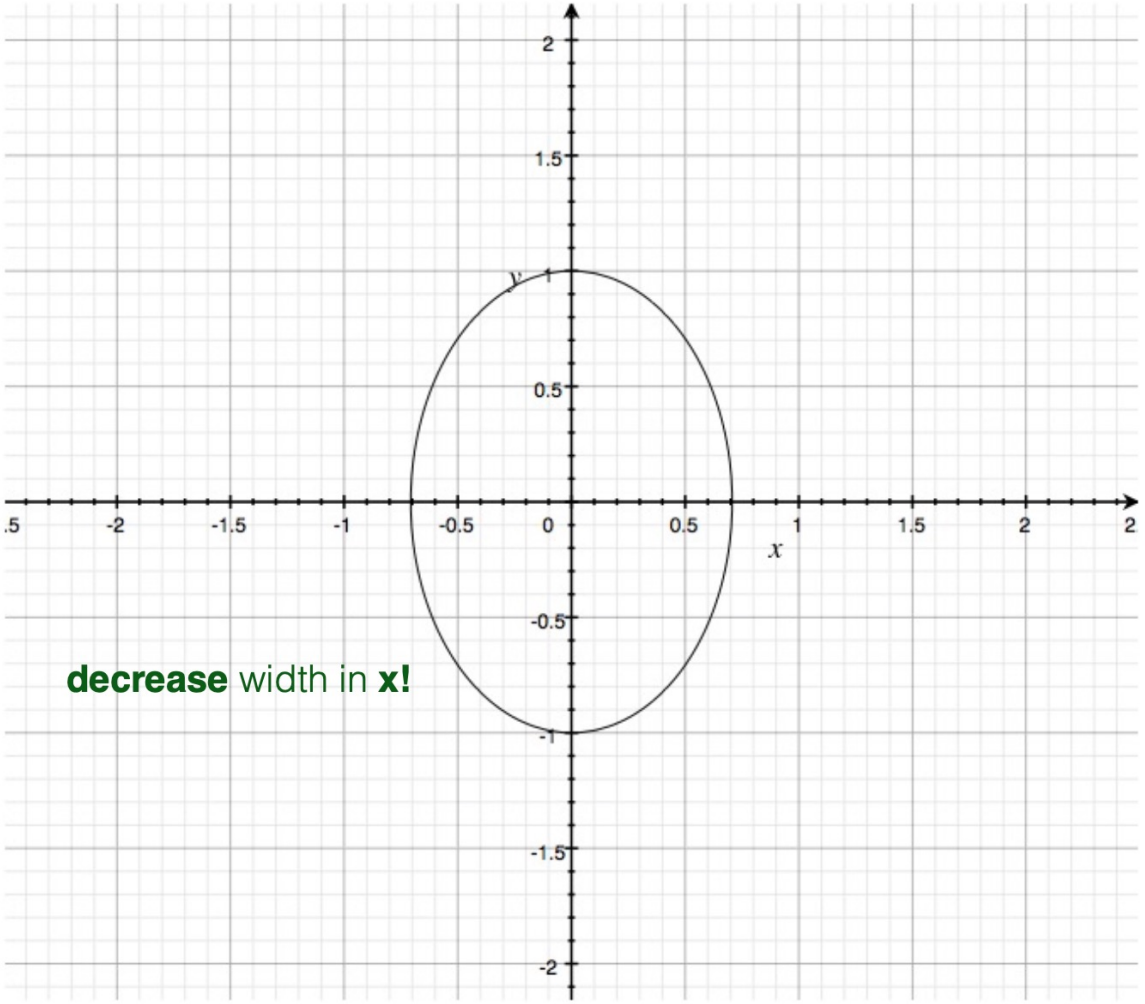


Eigen-decomposition

*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



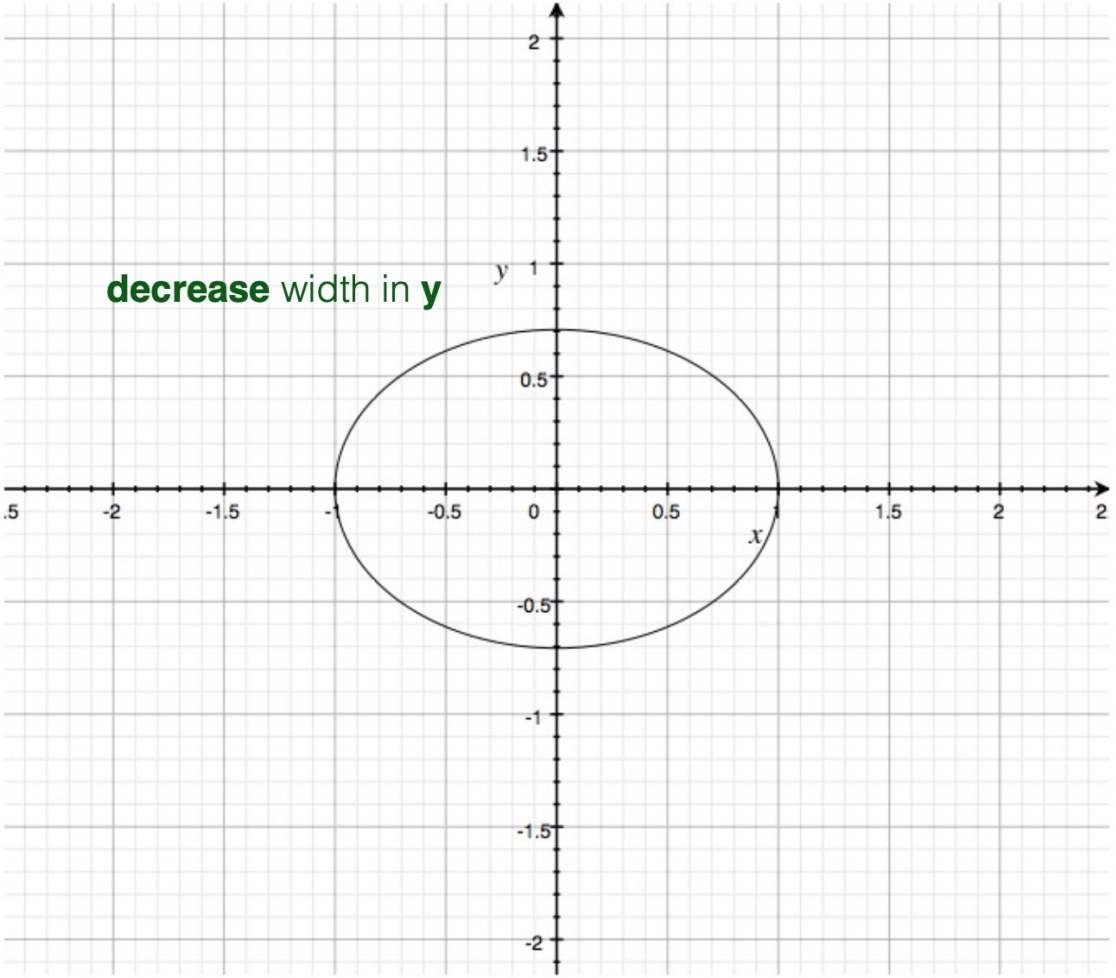
decrease width in **x!**

Eigen-decomposition

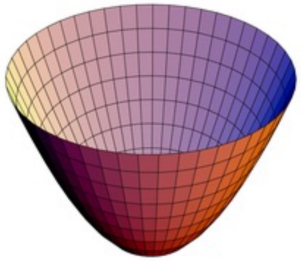
*What happens if you **increase** coefficient on **y**?*

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

and slice at 1



Eigen-decomposition



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

$$= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

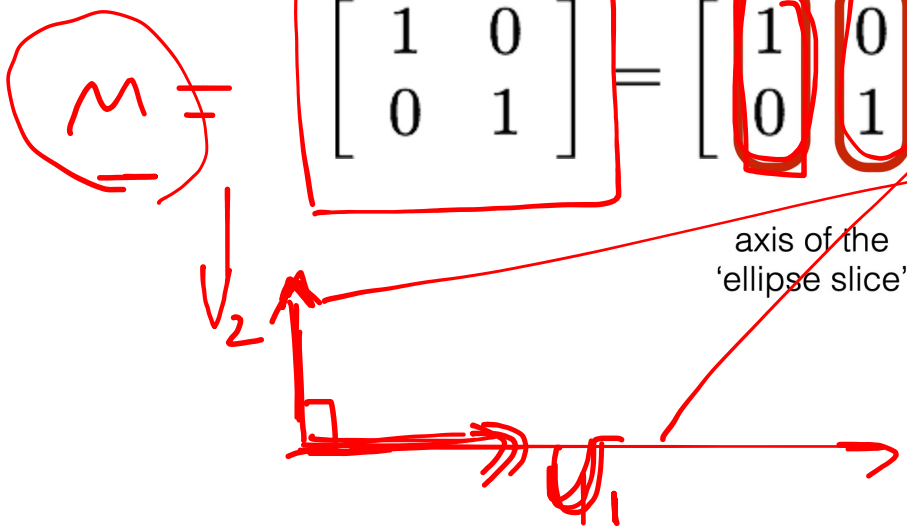
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^\top$$

eigenvectors

eigenvalues along diagonal

axis of the 'ellipse slice'

Inverse sqr of length of the quadratic along the axis



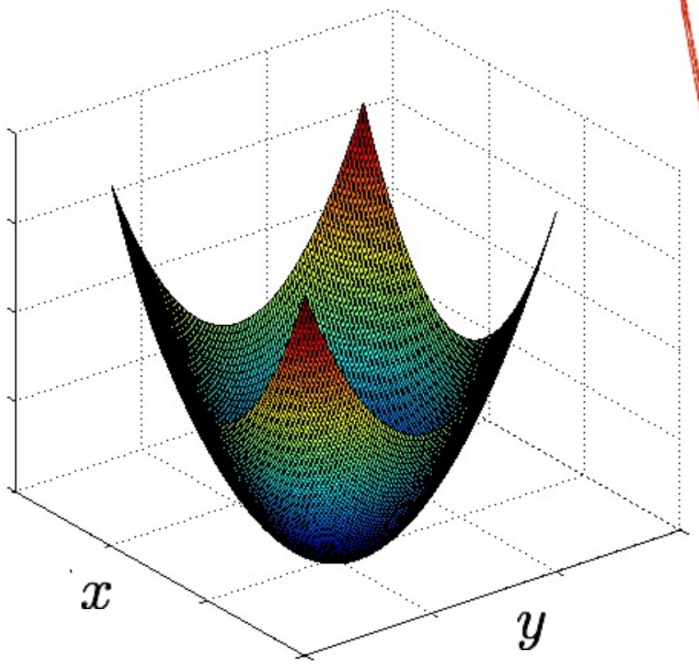
Eigen-decomposition

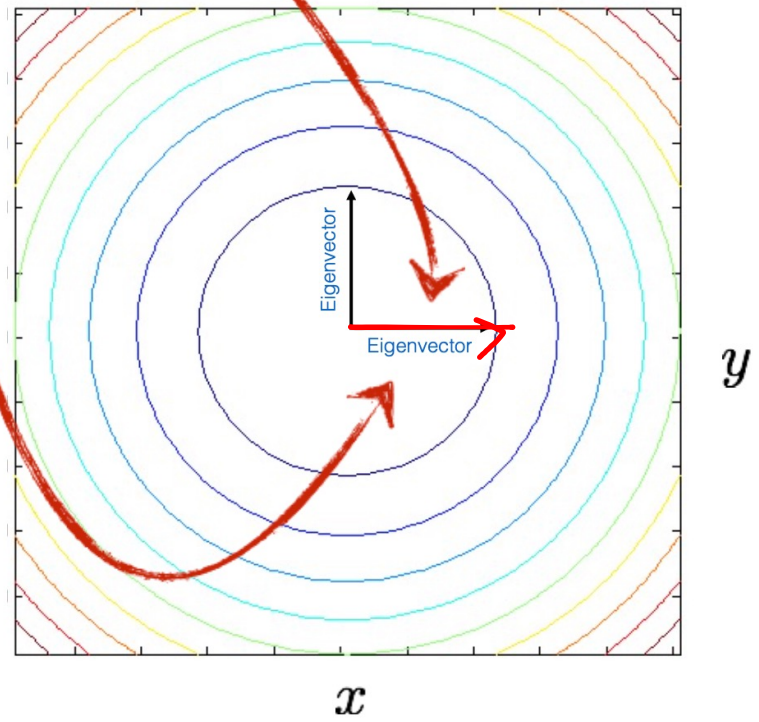
Eigenvectors Eigenvalues

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

Eigenvectors

Inverse sqr of the size of the axis





Eigen-decomposition

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

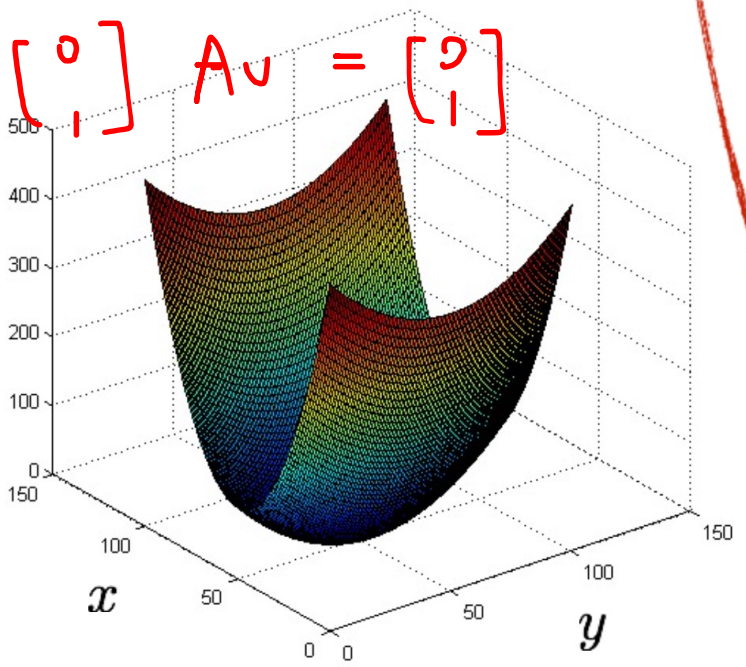
Eigenvalues

Eigenvectors

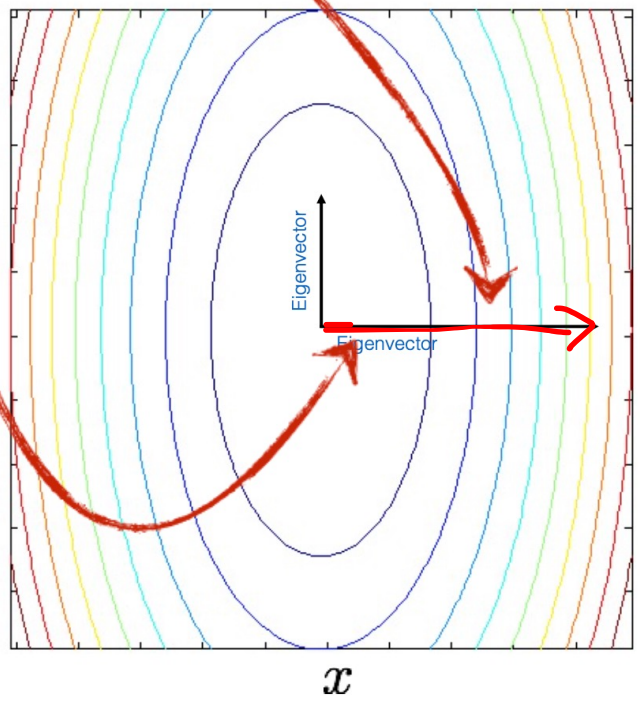
Eigenvectors

$v = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $Av = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$

$v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $Av = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$



Inverse sqr of length of axis



Eigen-decomposition

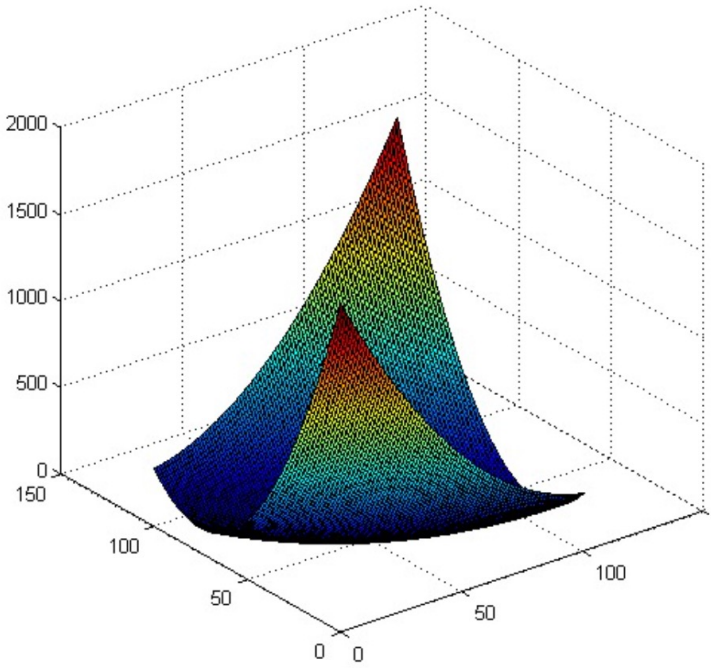
$$\mathbf{A} = \begin{bmatrix} 7.75 & 3.90 \\ 3.90 & 3.25 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T$$

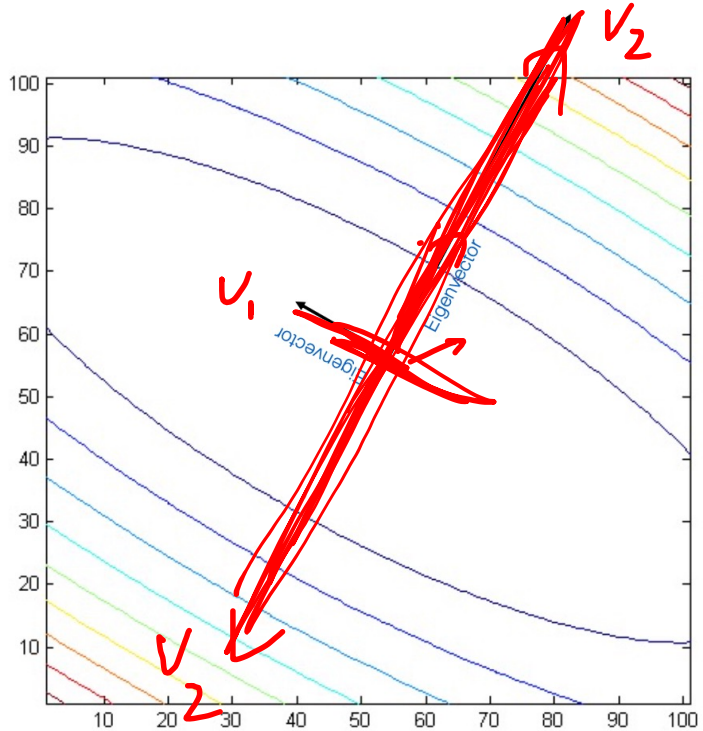
Eigenvalues

Eigenvectors

Eigenvectors

v_1 v_2



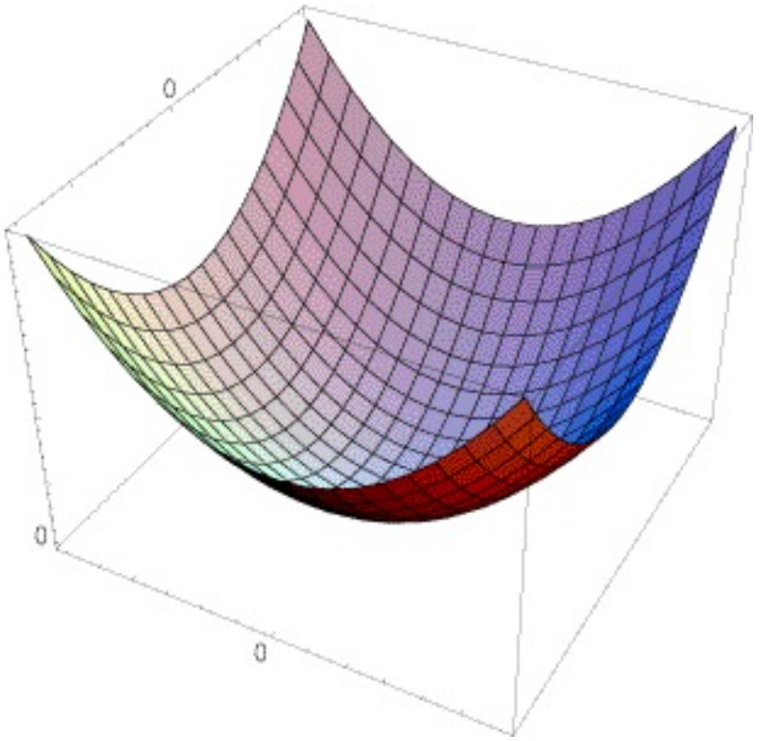


Harris corner detector

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

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

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



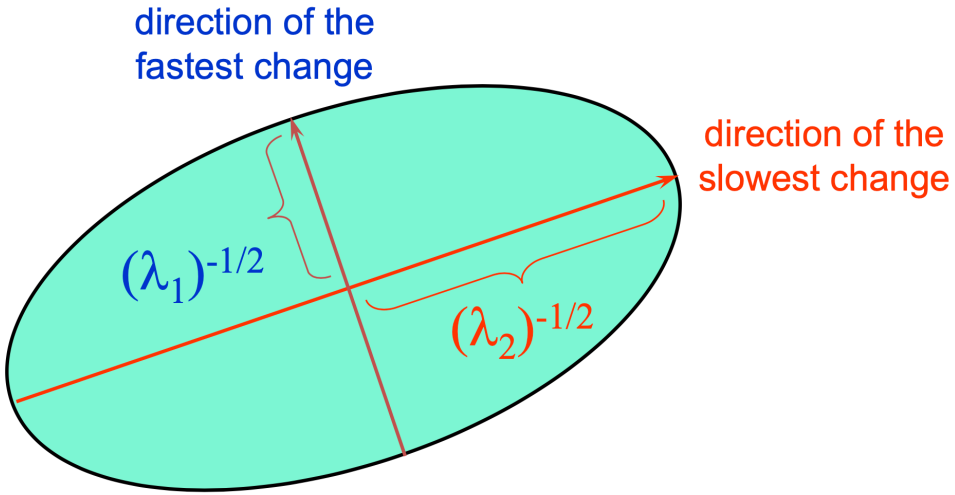
Harris corner detector

$$\boldsymbol{M} = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix} = \boldsymbol{R^{-1}} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \boldsymbol{R}$$

eigen-decomposition

Directions

Magnitude



Harris corner detector

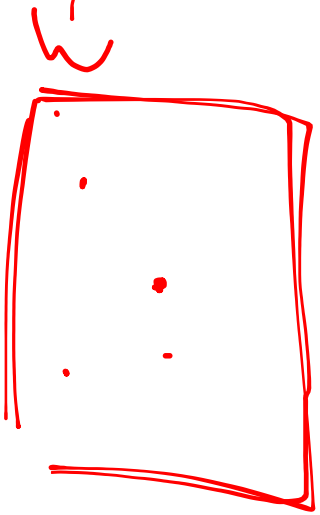
$$M = \begin{bmatrix} \sum_{x,y \in W} w(x,y) I_x^2 & \sum_{x,y \in W} w(x,y) I_x I_y \\ \sum_{x,y \in W} w(x,y) I_x I_y & \sum_{x,y \in W} w(x,y) I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

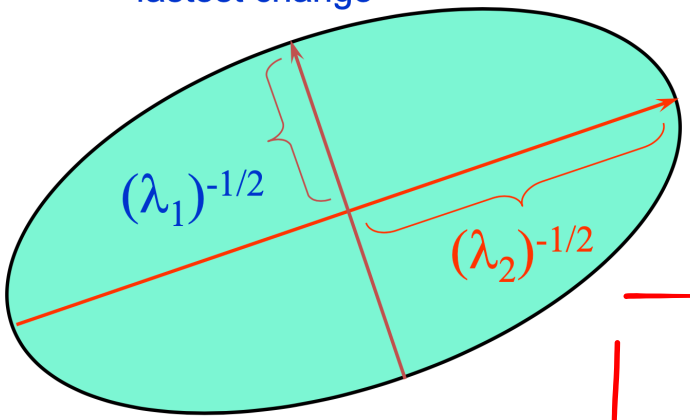
eigen-decomposition

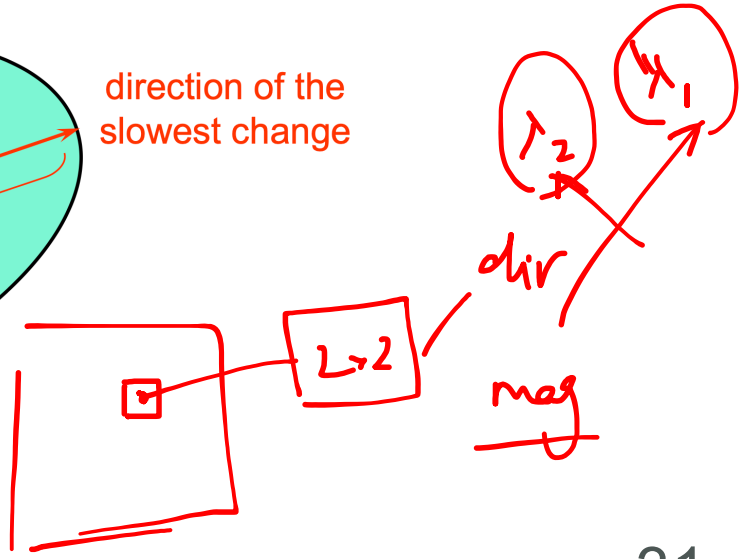
Directions

Magnitude

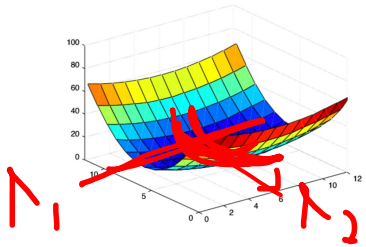
Extended form

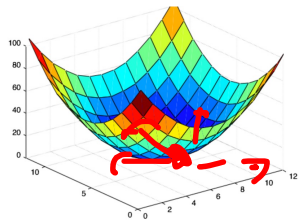


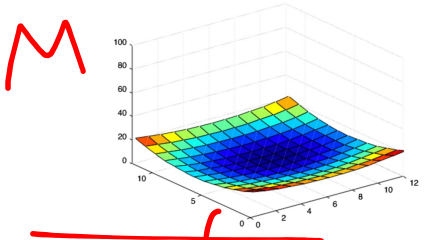


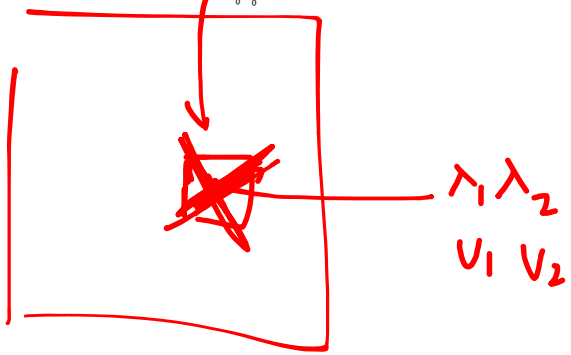


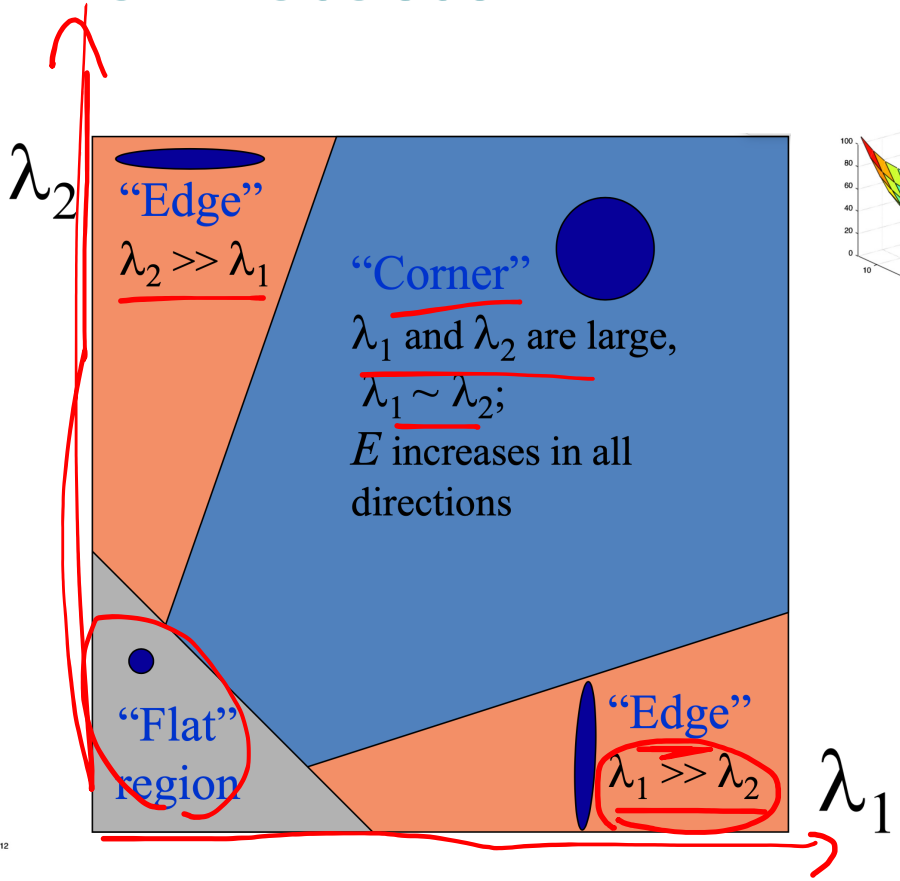
Harris corner detector



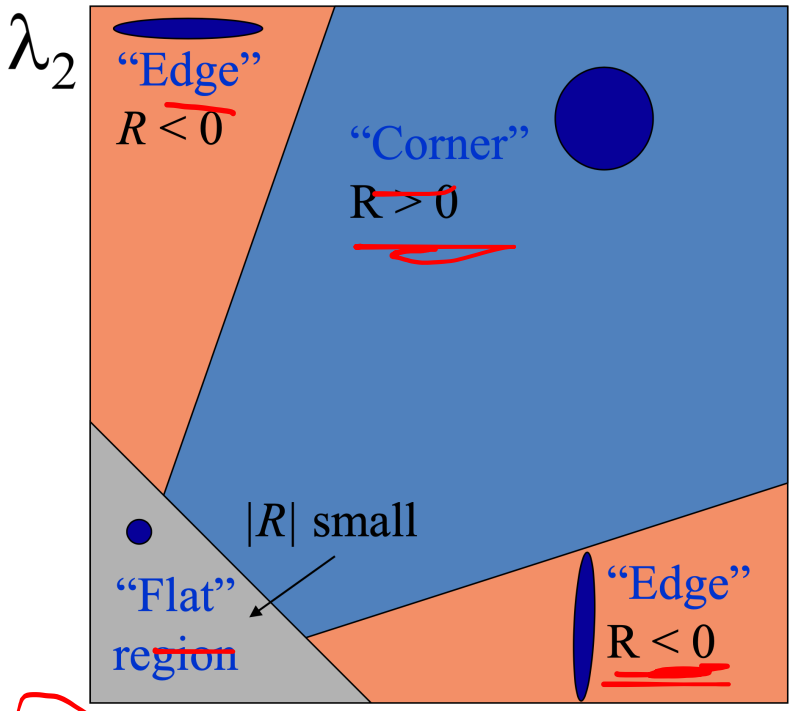


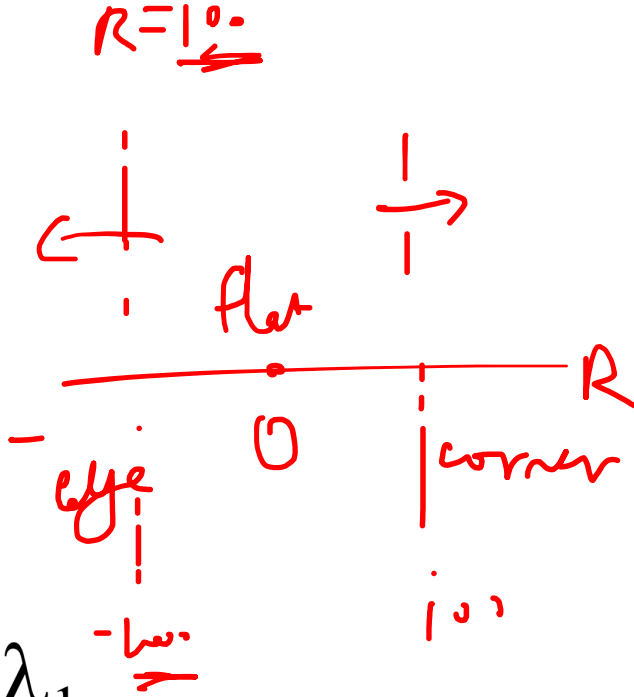






Harris corner detector





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

(0.04 to 0.06)

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

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

Harris corner detector

Harris & Stephens (1988)

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

Kanade & Tomasi (1994)

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

Nobel (1998)

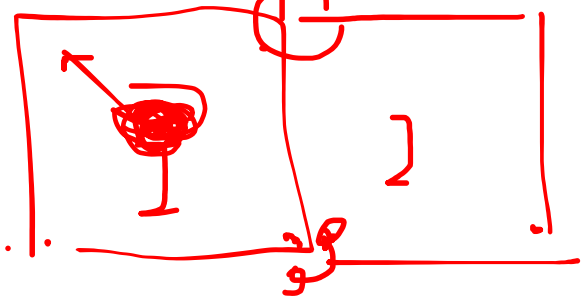
$$R = \frac{\det(M)}{\text{trace}(M) + \epsilon}$$

Harris corner detector

Algorithm

- Compute partial derivatives I_x, I_y per pixel
- Compute $\boldsymbol{M}$ at each pixel, using Gaussian weighting $\boldsymbol{W}$

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



Harris corner detector

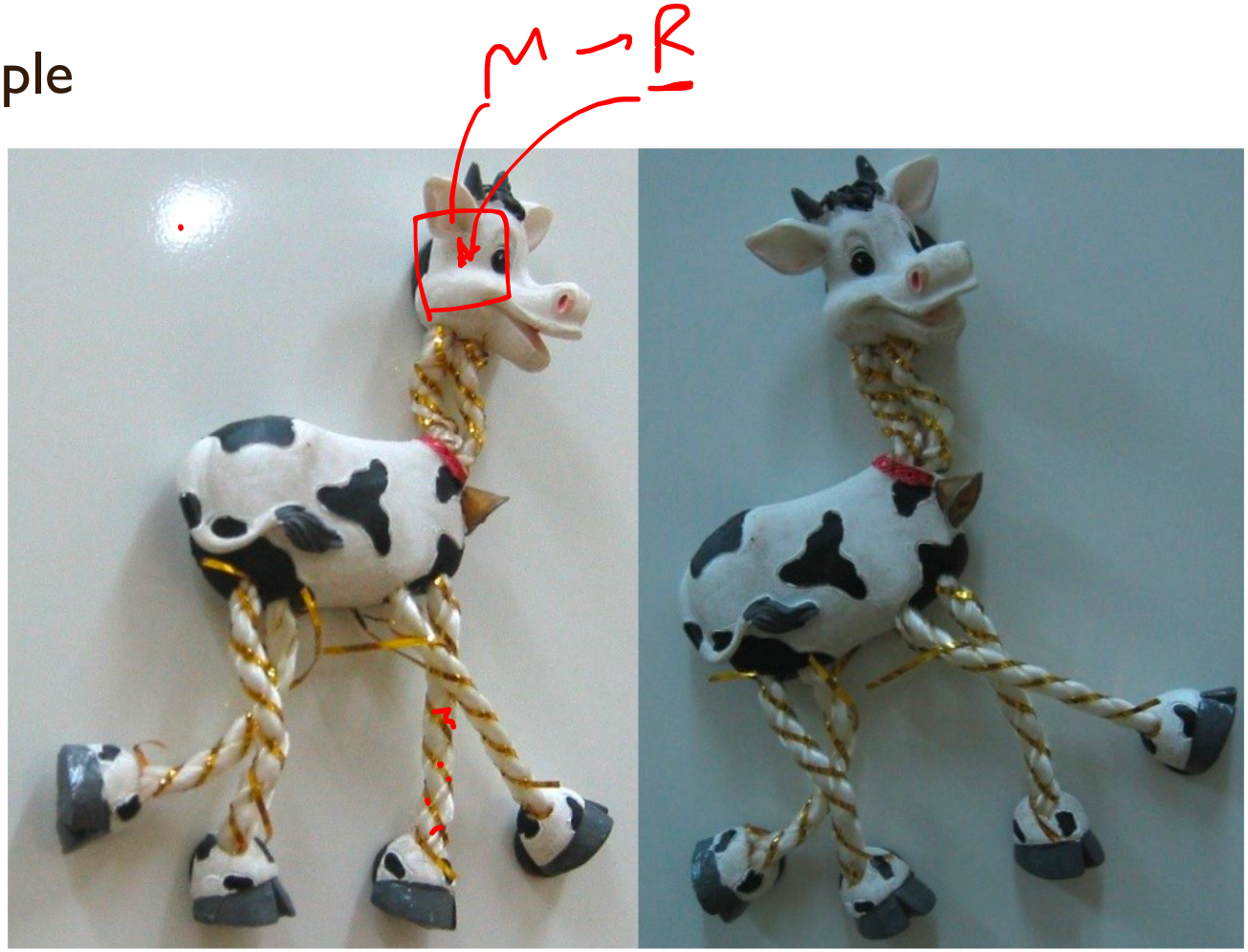
Algorithm

- Compute partial derivatives I_x, I_y per pixel
- Compute $\boldsymbol{M}$ at each pixel, using Gaussian weighting $\boldsymbol{W}$
- Compute response function R

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

Harris corner detector

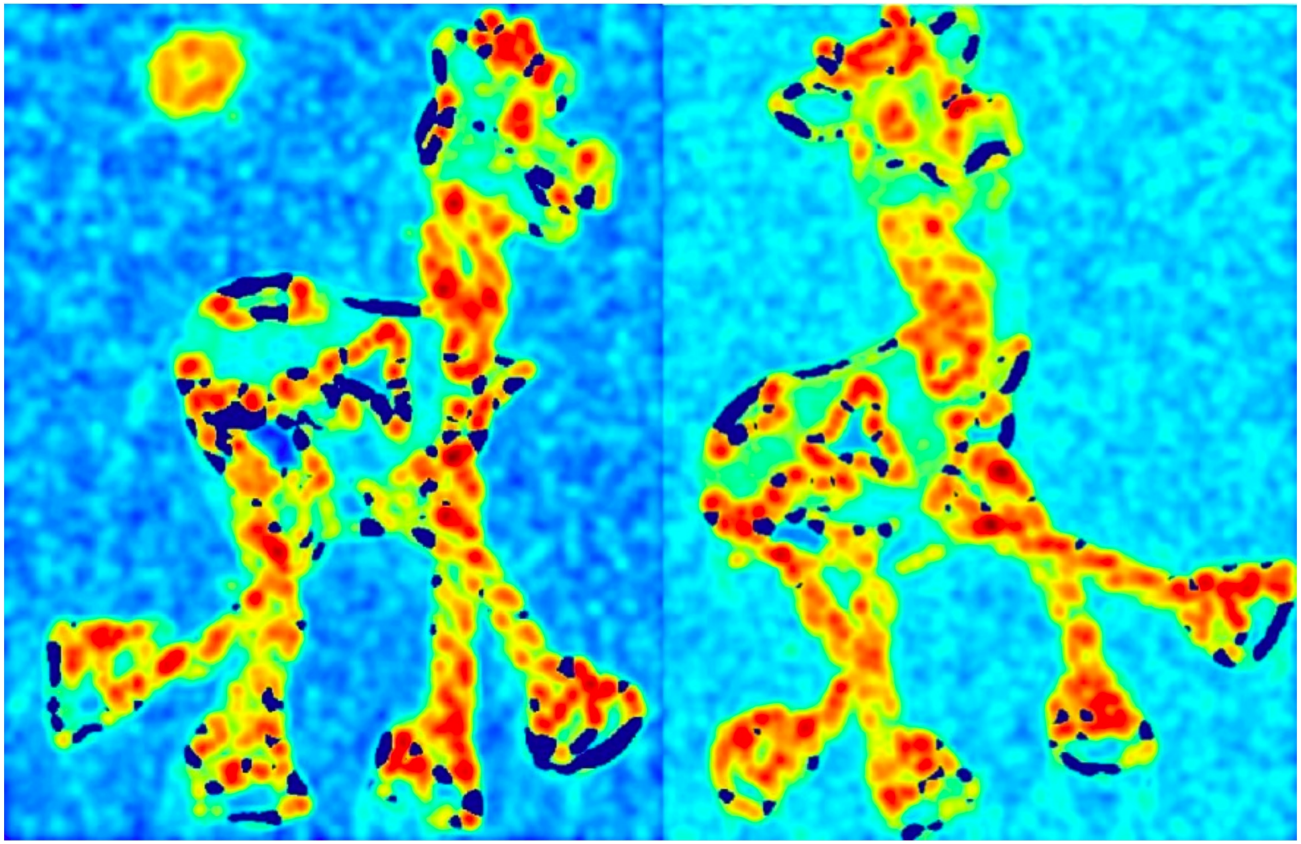
Example



Harris corner detector

Example

Compute R



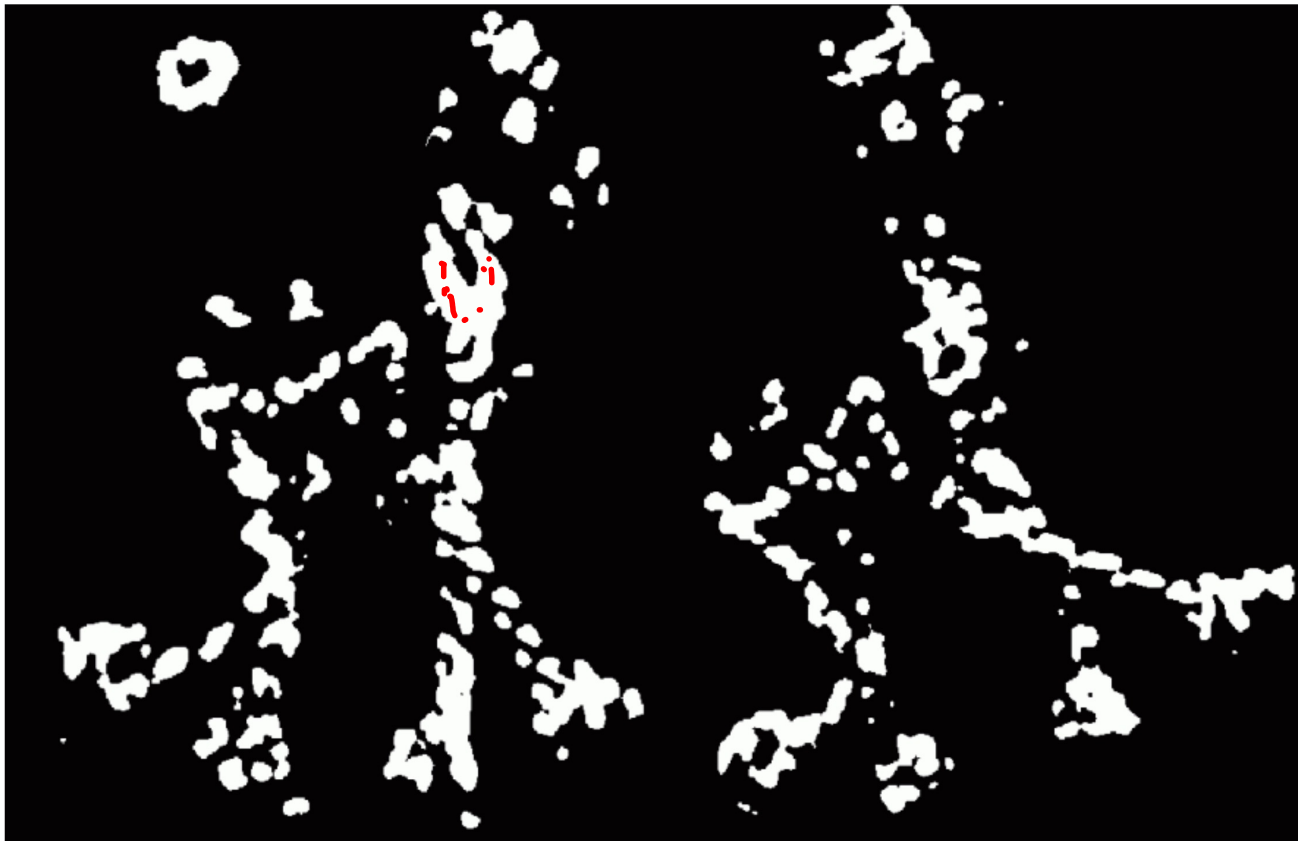
Harris corner detector

Algorithm

- Compute partial derivatives I_x, I_y per pixel
- Compute $\boldsymbol{M}$ at each pixel, using Gaussian weighting $\boldsymbol{w}$
- Compute response function R
- Threshold R

Harris corner detector

Example Threshold R



Harris corner detector

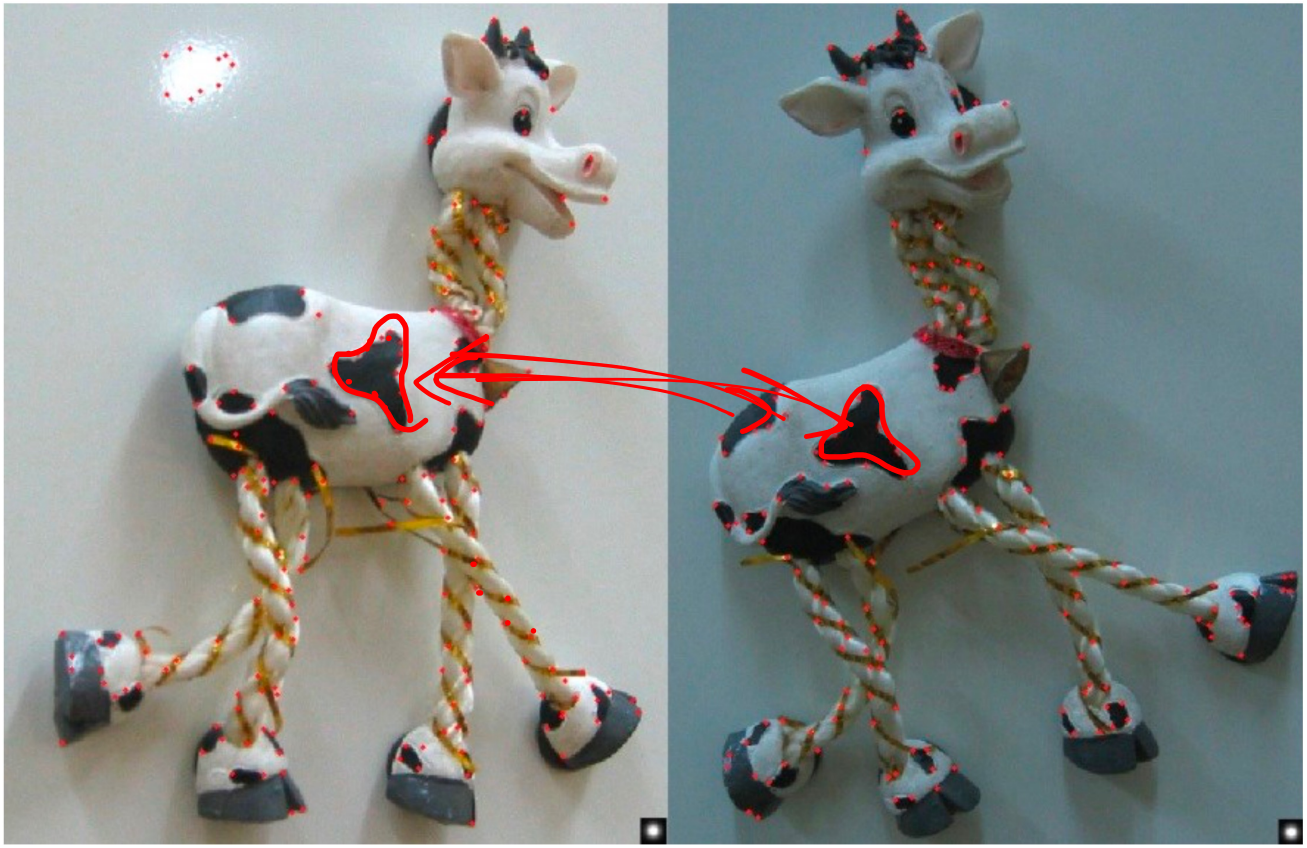
Algorithm

- Compute partial derivatives I_x, I_y per pixel
- Compute $\boldsymbol{M}$ at each pixel, using Gaussian weighting $\boldsymbol{w}$
- Compute response function R
- Threshold R
- Take only local maxima (called non-maxima suppression)

Harris corner detector

Example

Final results



Harris corner detector

Robustness

- Lighting

Harris corner detector

Robustness

- Lighting (*affine intensity* change)

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

Would the affine transformation effect $\boldsymbol{M}$?

$$\boldsymbol{M} = a^2 \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix}$$

a will effect the scale

~~*b* does not effect~~

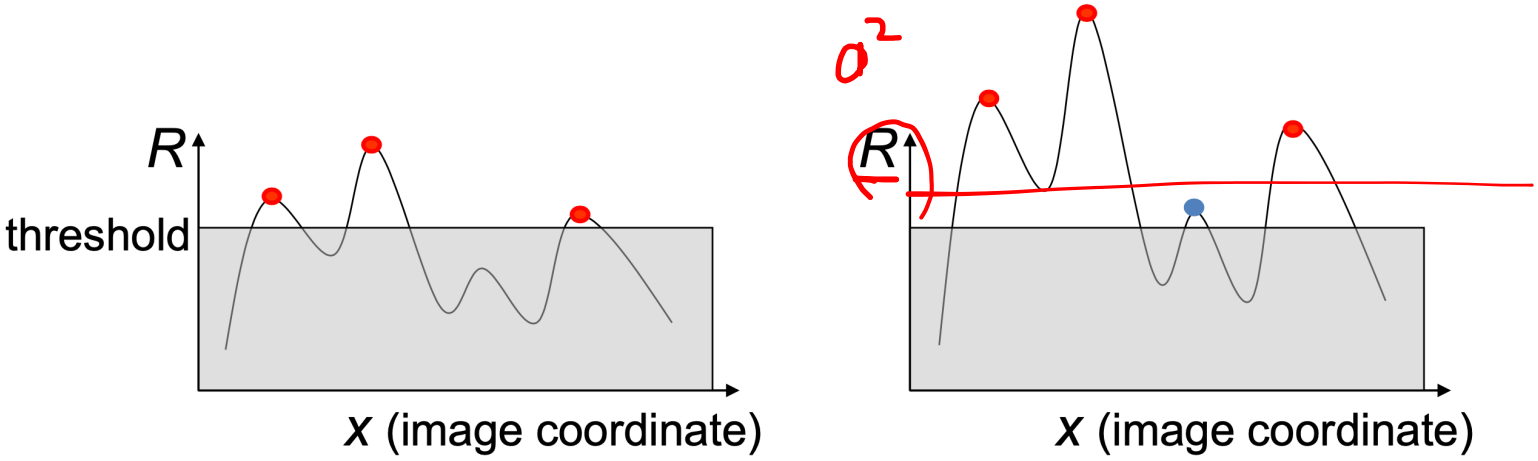
Harris corner detector

Robustness

- Lighting (*affine intensity* change)

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

Would the affine transformation effect $\boldsymbol{M}$?

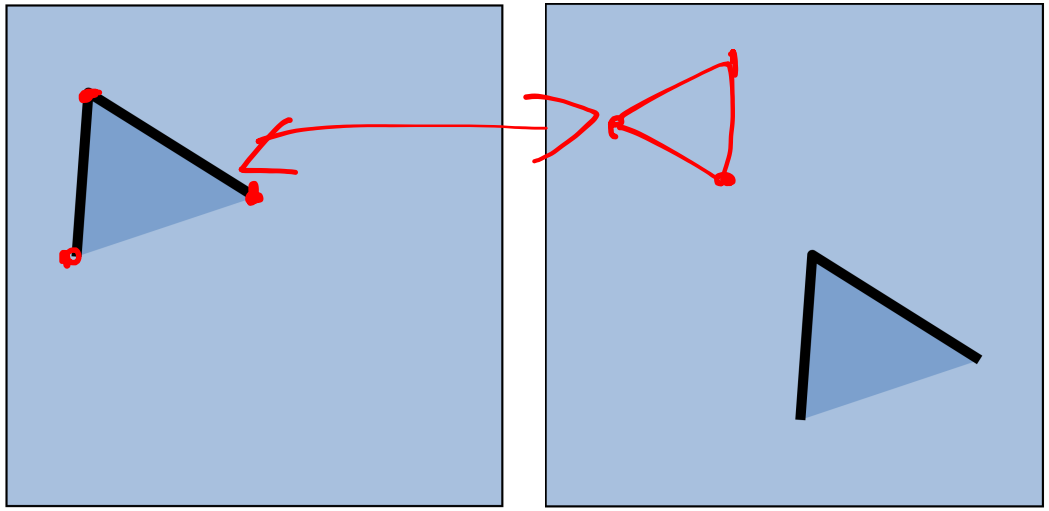


adaptive strategy to choose a threshold

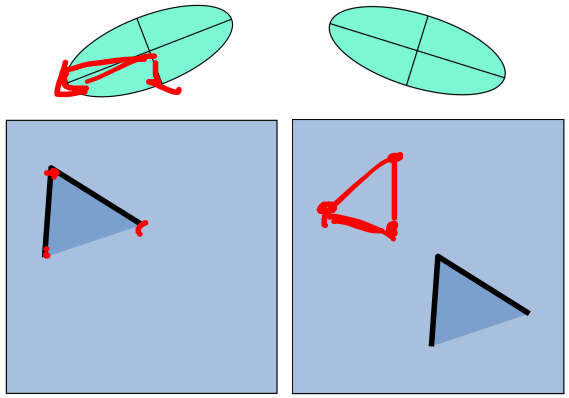
Harris corner detector

Robustness

- ✓ Lighting (*affine intensity* change)
- Rotation



Harris corner detector



Algorithm

- Compute partial derivatives I_x, I_y per pixel
- Compute $\boldsymbol{M}$ at each pixel, using Gaussian weighting $\boldsymbol{w}$
- Compute response function R

Will M change?

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

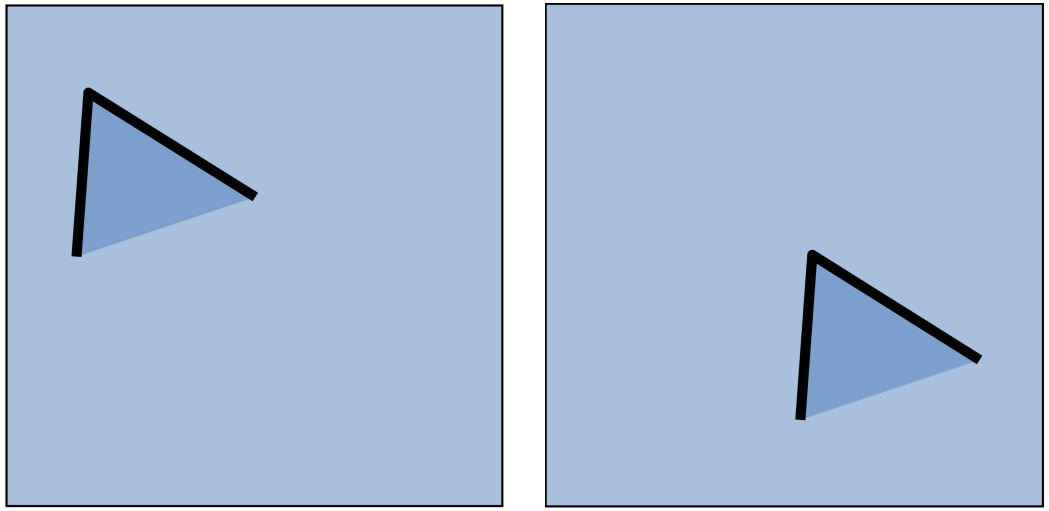
Will R change?

Rotations just cause the corner rotation to change.
Eigenvalues remain the same

Harris corner detector

Robustness

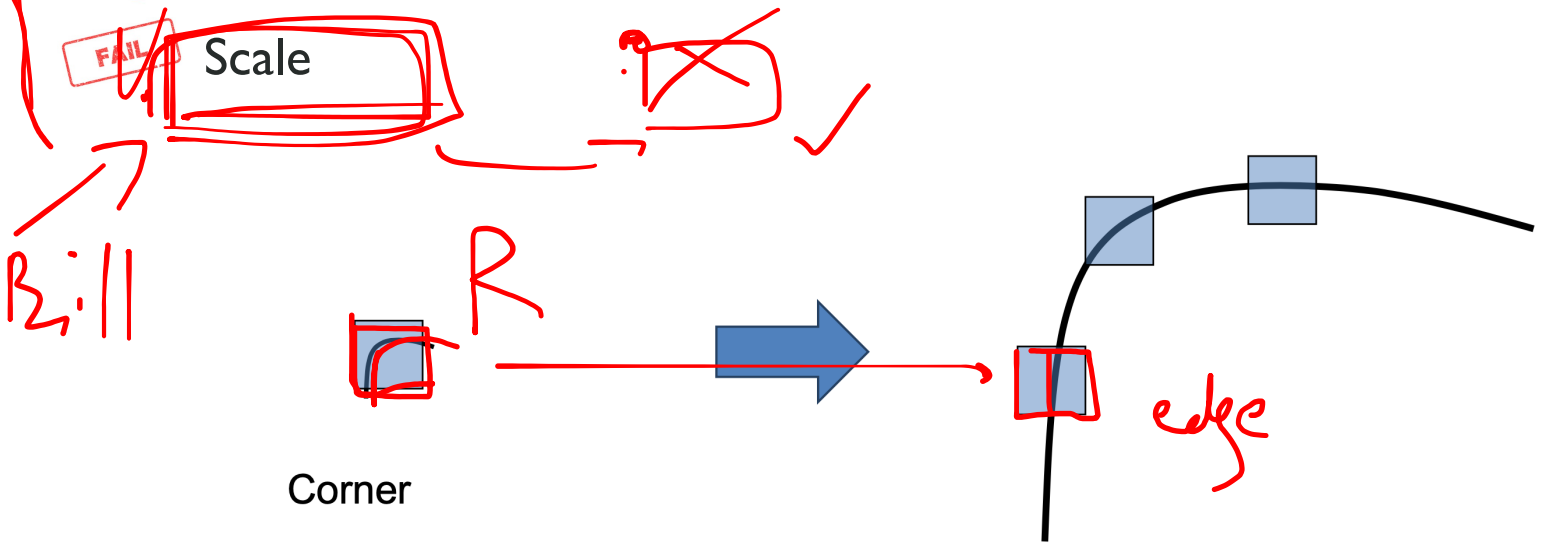
- Lighting (*affine intensity* change)
- Rotation



Harris corner detector

Robustness

- Lighting (*affine intensity* change)
- Rotation
- Scale

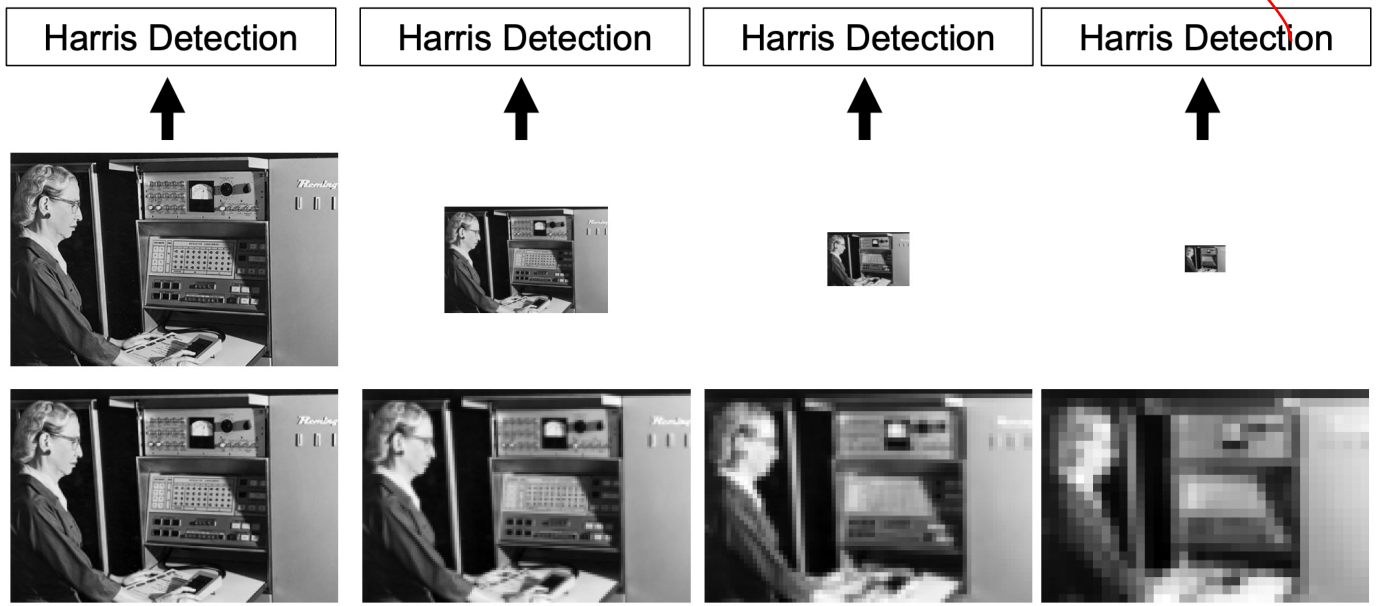


One pixel can become many pixels and vice- versa

Harris corner detector

Robustness

- ✓ Lighting (*affine intensity* change)
- ✓ Rotation
- Scale

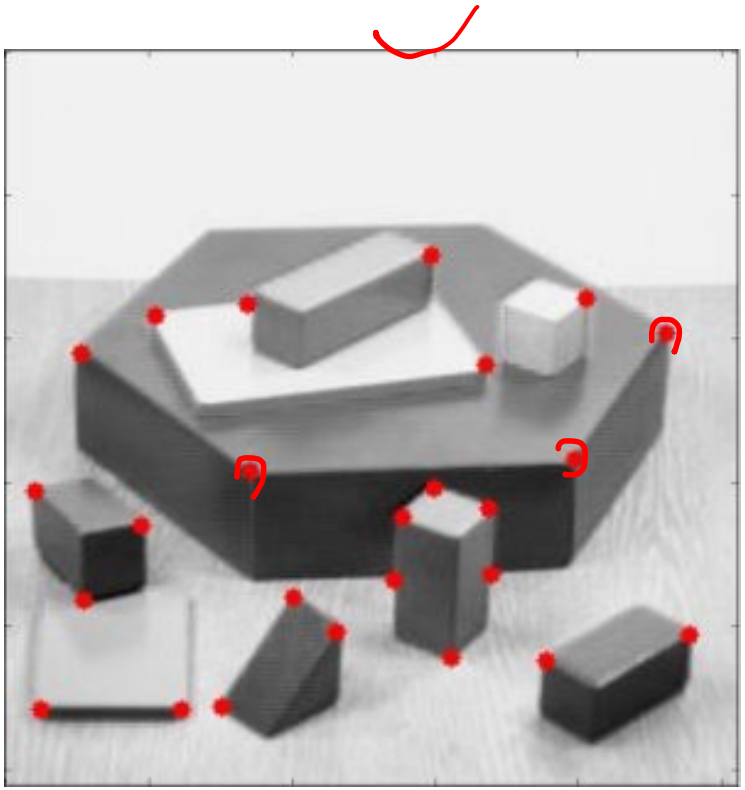


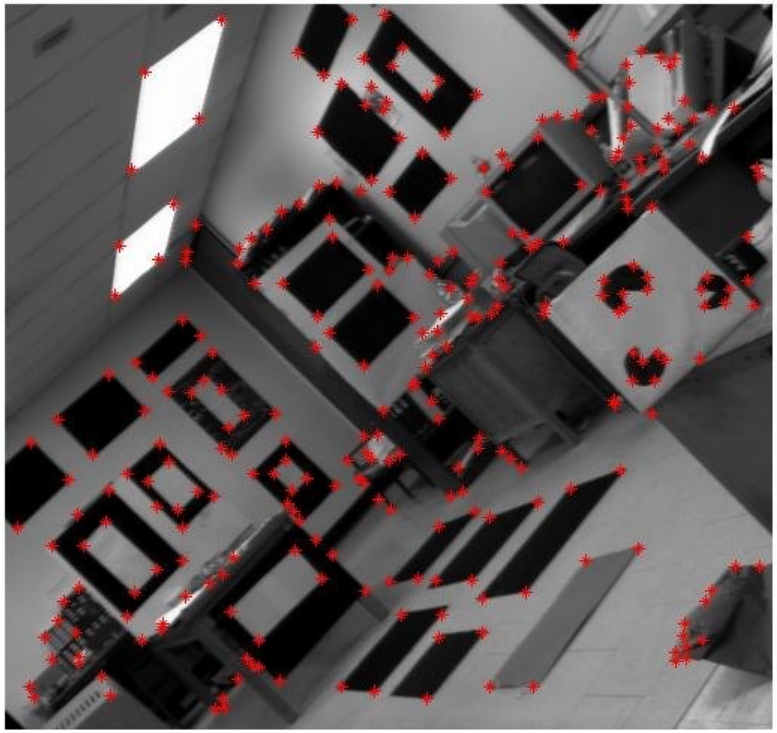
Harris corner detector

Experiments

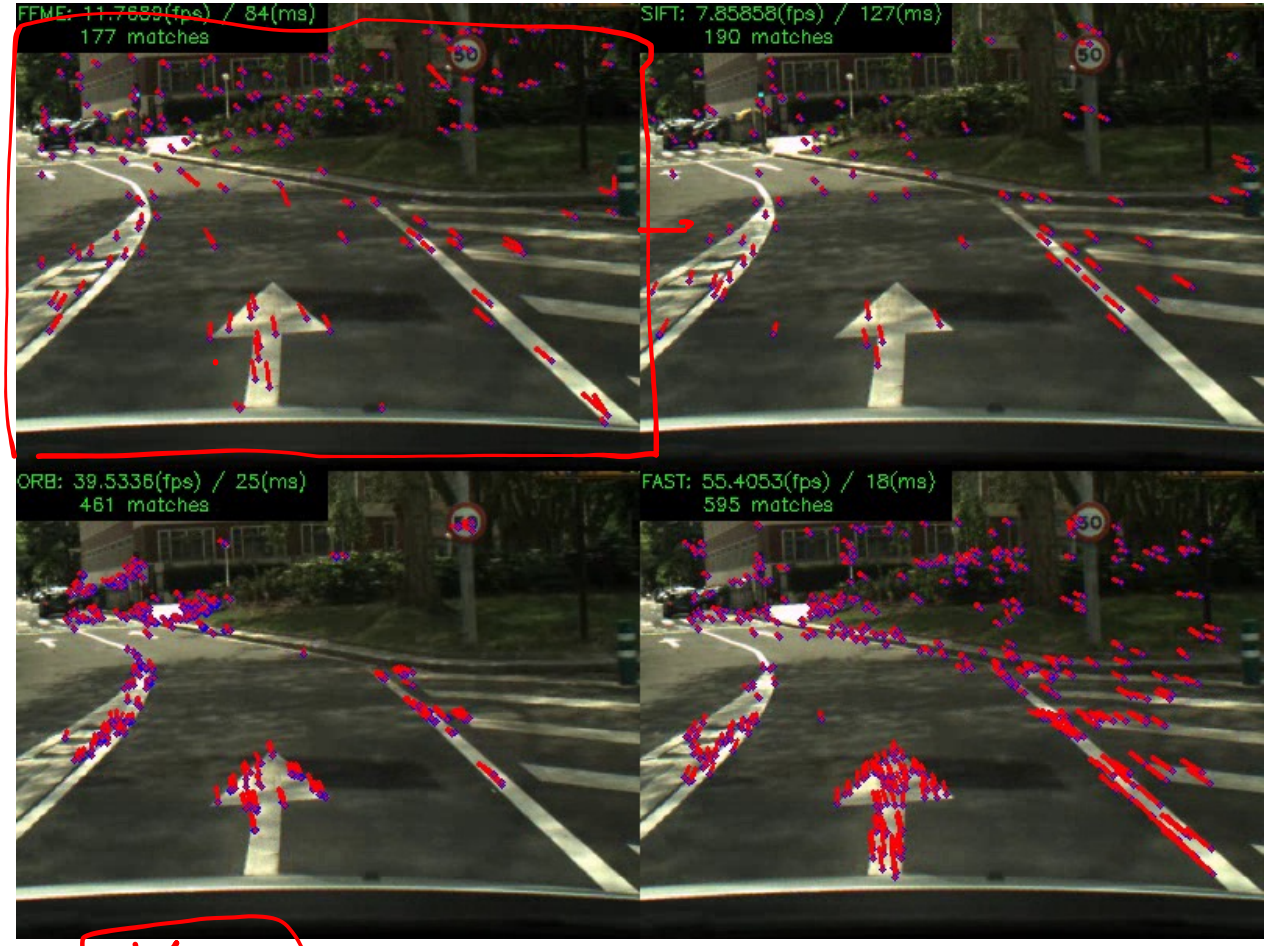
Harris corner detector

Experiments





Next lecture: Feature descriptor



Thank you very much!

sihengc@sjtu.edu.cn