Компьютерное зрение

Лекция 3.
Ключевые точки. Дескрипторы. Гомография.

04.06.2020

Руслан Рахимов

План

- Ключевые точки (Corners)
- Дескрипторы (Descriptors)
- 2d преобразования (Warping)
- Гомография (Homography)

План

- Ключевые точки (Corners)
- Дескрипторы (Descriptors)
- 2d преобразования (Warping)
- Гомография (Homography)

Вводный пример

Где медведь?

Задача 1

Задача 2

Вводный пример

Где медведь?

Задача 1

Задача 2

1) Почему во втором случае было проще найти?
2) Были видны “характерные” фрагменты медведя

Особенности (features)

- Хорошо различимые “фрагменты” объекта
  - особенности (features)
  - характеристические точки (characteristic points)
  - ключевые точки (keypoints)
  - Локальные особые точки (local feature points)
- Характерные фрагменты позволяют справиться с изменениями ракурса масштаба и перекрытиями

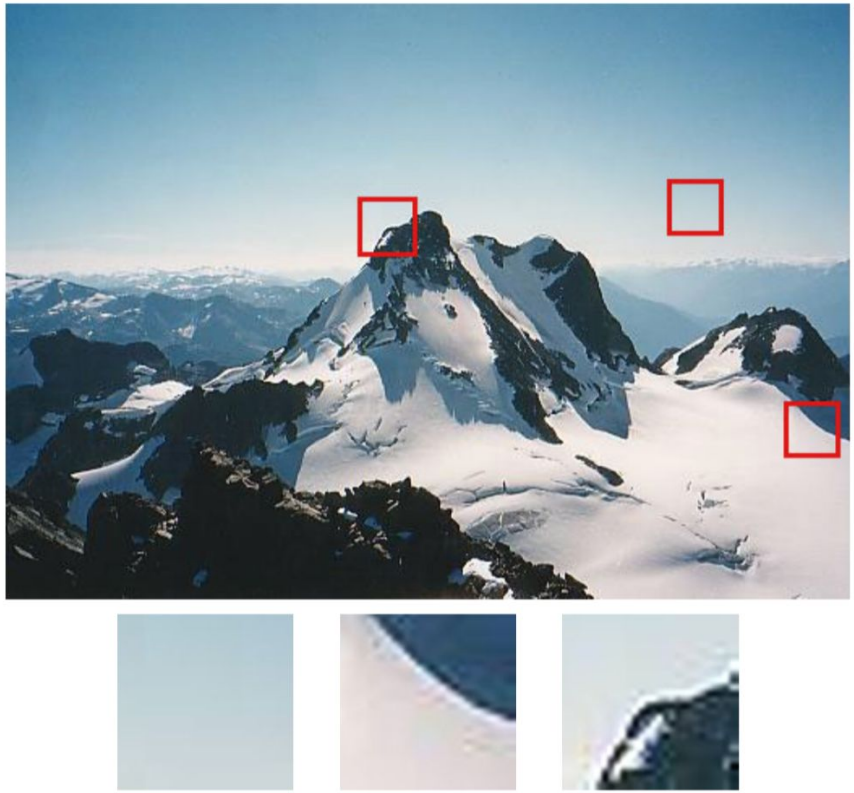
Требования

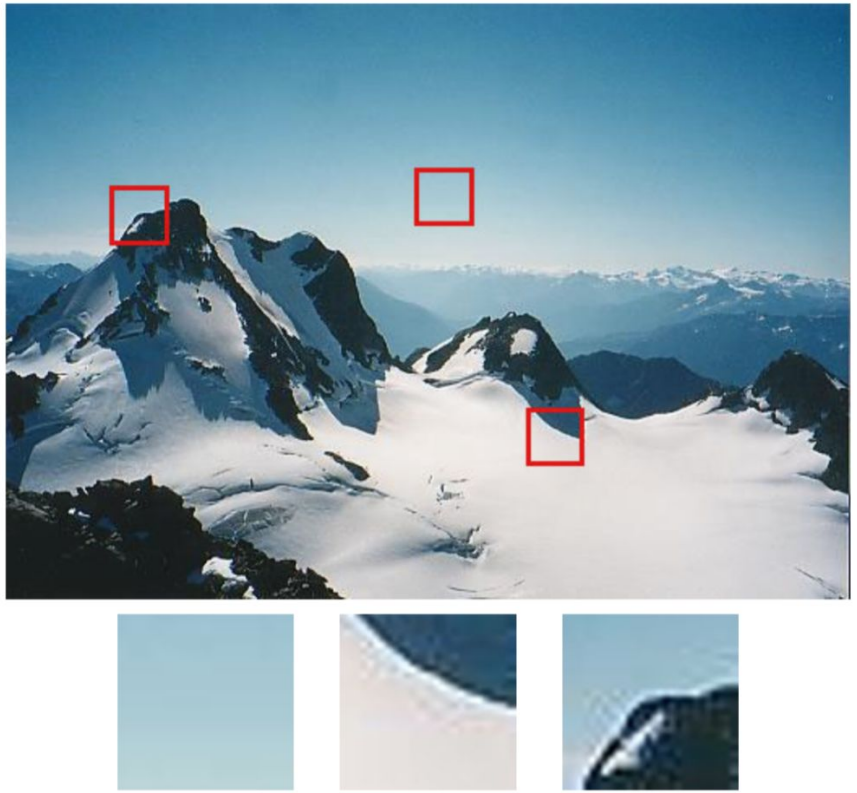
- Как можно сформулировать требования к хорошо различимым фрагментам объекта?

Требования

- Как можно сформулировать требования к хорошо различимым фрагментам объекта?
- Отличаются от большинства других фрагментов объекта
- Инвариантны к изменению освещения
- Инвариантны к изменению ракурса
  - Можно находить одну и ту же точку на измененных изображениях
  - Можем “идентифицировать эту точку”

Характерные точки





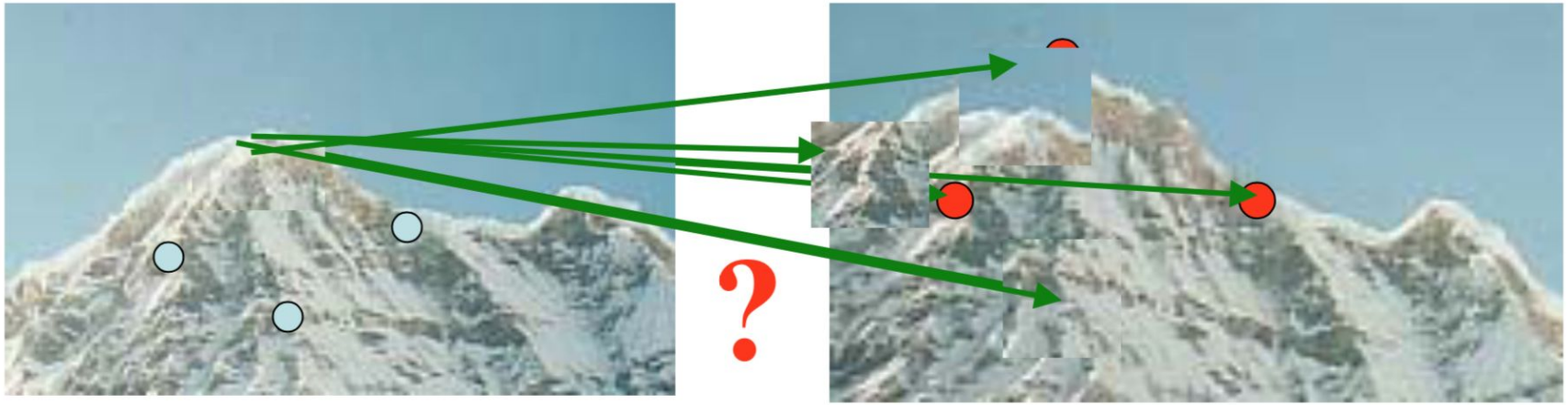
Требования

- Локальность (locality)
  - Особенность занимает маленькую область изображения, поэтому работа с ней не чувствительна к перекрытиям
- Повторимость (repeatability)
  - Особенность находится в том же месте объекта не смотря на изменения масштаба, положения, ракурса и освещения
- Значимость (saliency)
  - Каждая особенность имеет уникальное (distinctive) описание
- Компактность и эффективность
  - Количество особенностей существенно меньше числа пикселей изображения

Характерные точки

- позволяют находить одинаковые области (предметы) на разных изображениях
- используются для склейки панорам и составления карт по спутниковым снимкам

Повторимость



Поиск характерной точки на изображении

- Как понять что выбранная область содержит характерную точку?
- Область вокруг характерной точки должна сильно варьироваться
- В области характерной точки небольшой сдвиг изображения должен приводить к существенному различию по сравнению с исходным изображением

Why detect corners?

Why detect corners?

Image alignment (homography, fundamental matrix)

3D reconstruction

Motion tracking

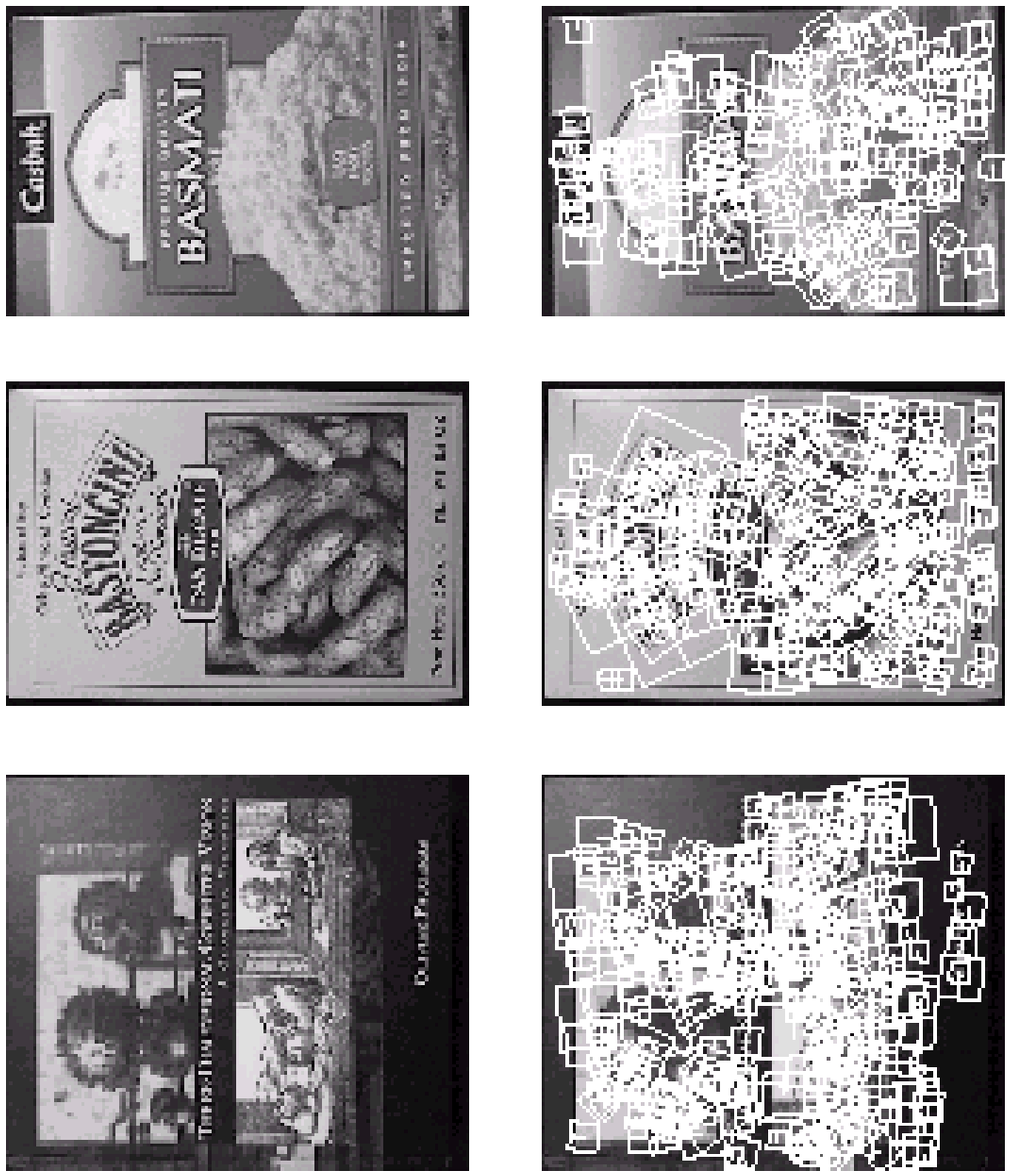
Object recognition

Indexing and database retrieval

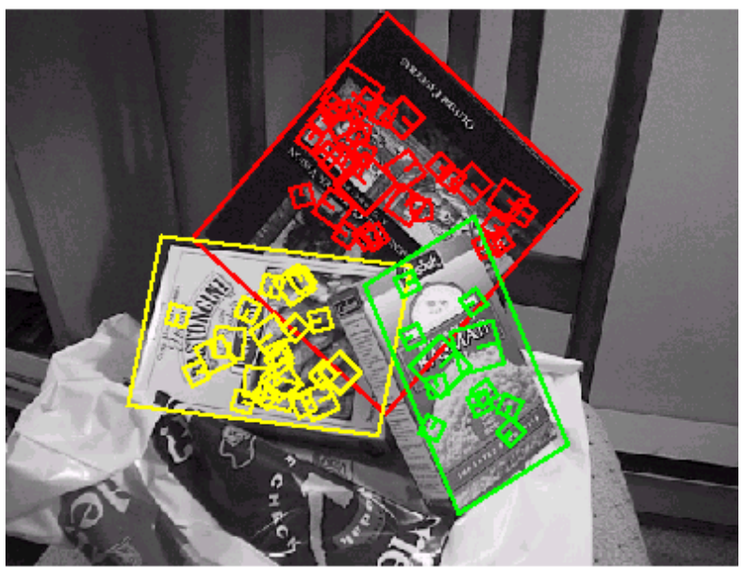
Robot navigation

Planar object instance recognition

Database of planar objects



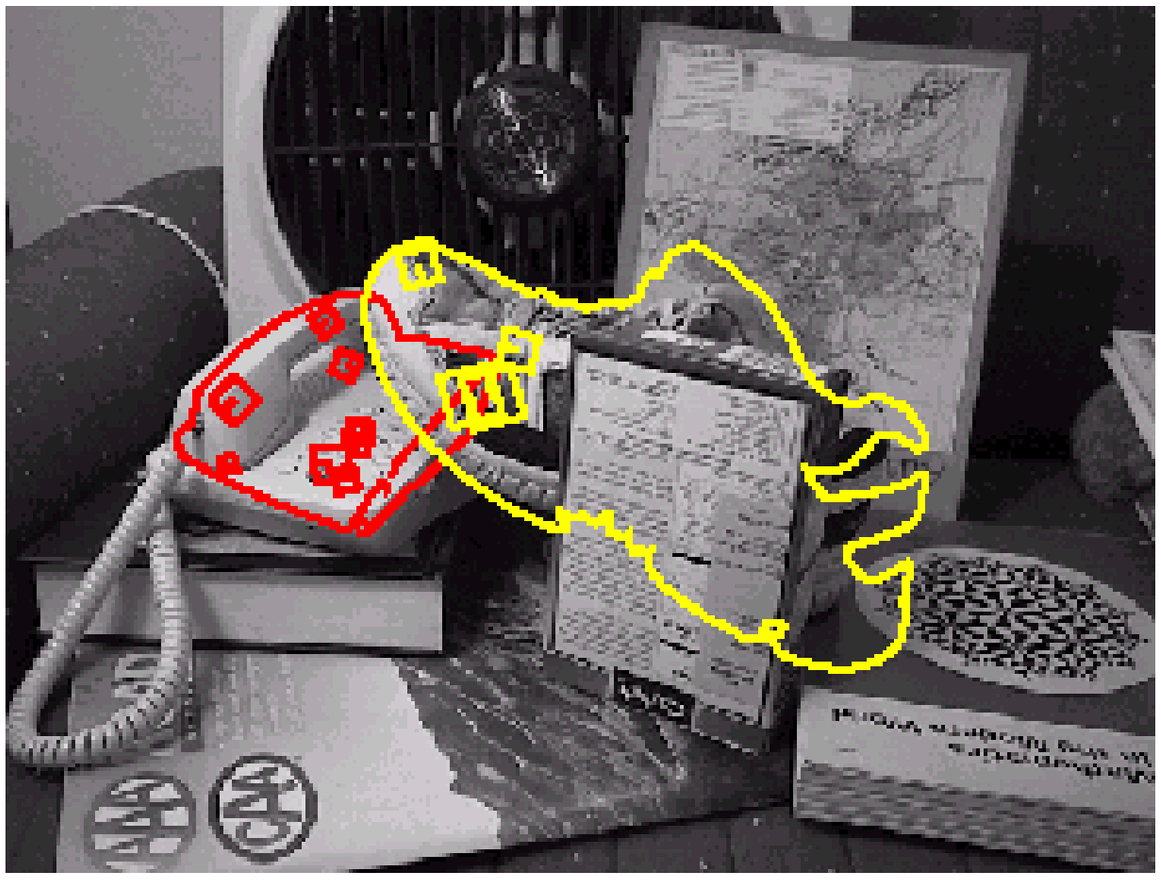
Instance recognition

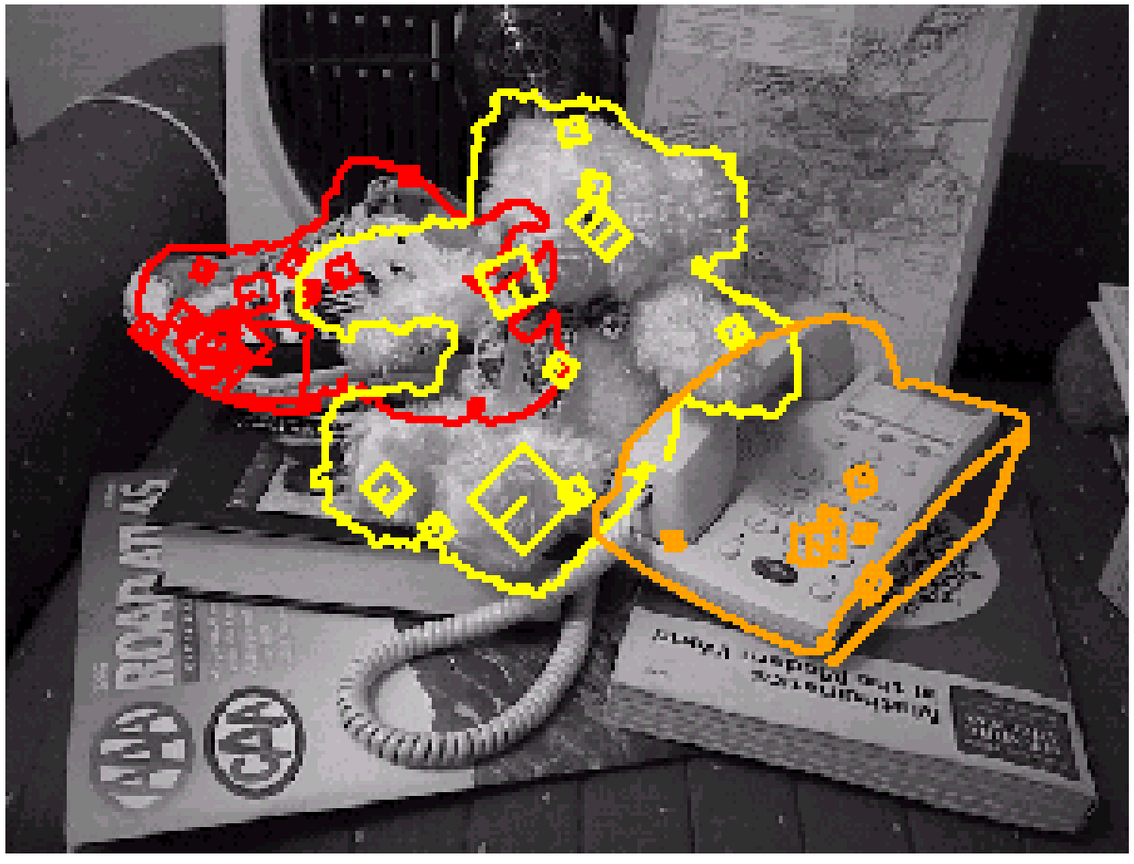


3D object recognition

Database of 3D objects

3D objects recognition





Recognition under occlusion

Location Recognition

Robot Localization

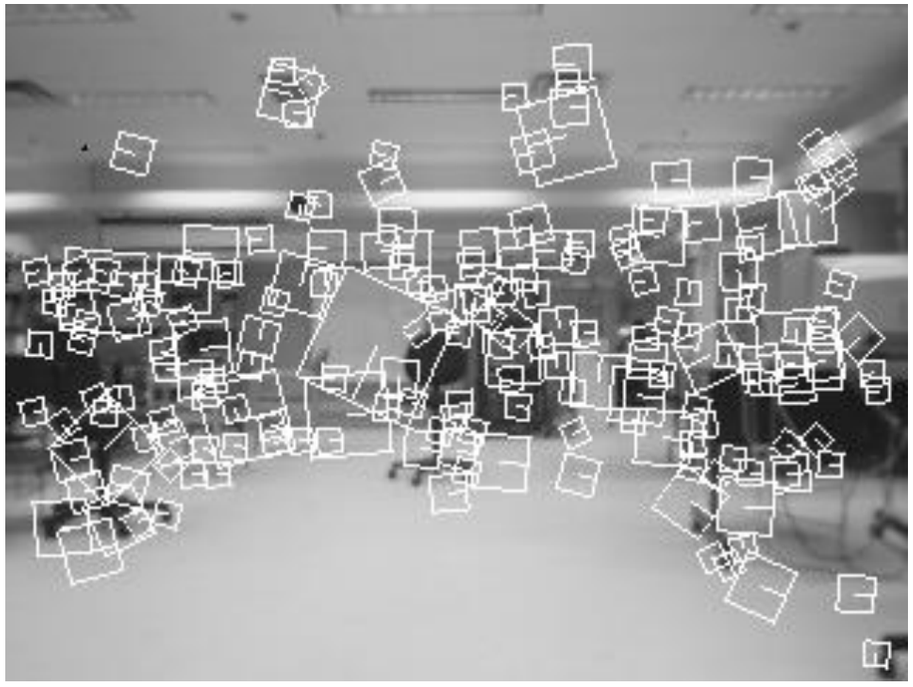
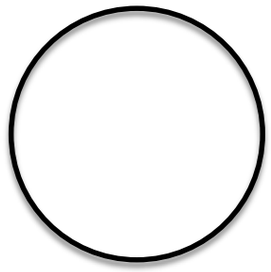


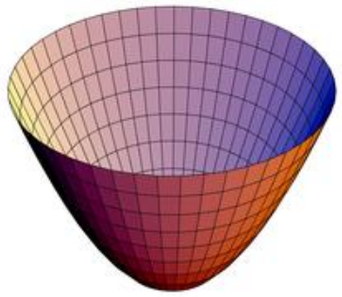
Image matching

Visualizing quadratics



Equation of a circle

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



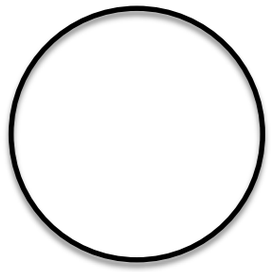
Equation of a ‘bowl’ (paraboloid)

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

If you slice the bowl at

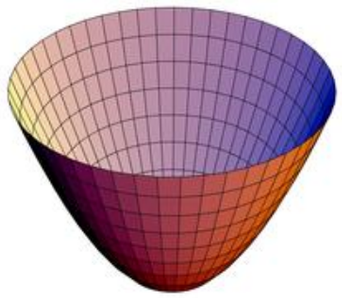
$$f(x, y) = 1$$

what do you get?



Equation of a circle

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



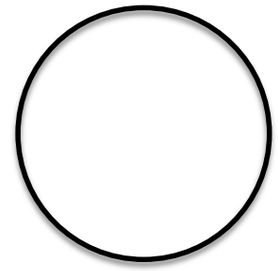
Equation of a ‘bowl’ (paraboloid)

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

If you slice the bowl at

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

what do you get?



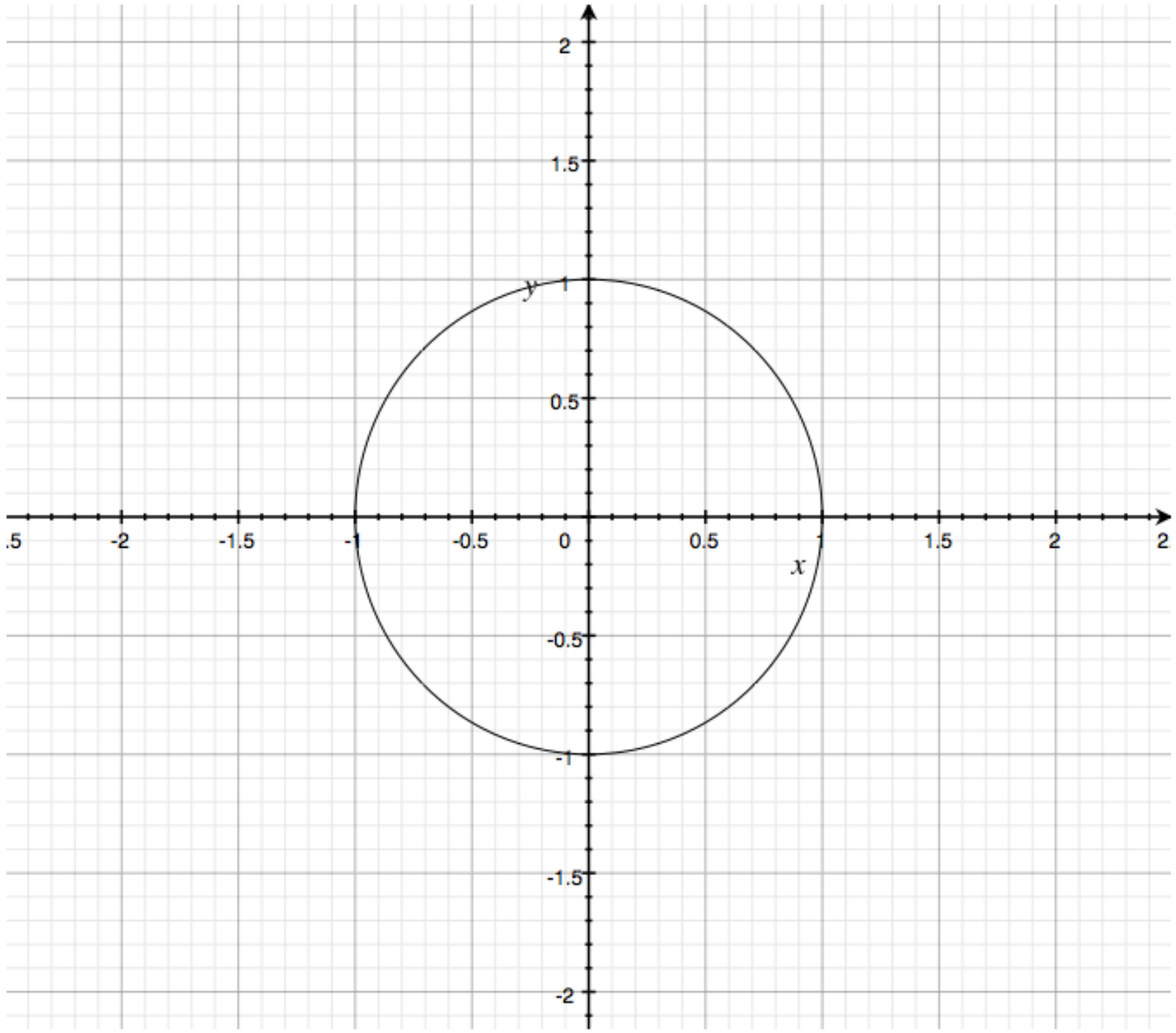
$$f(x,y) = x^2 + y^2$$

can be written in matrix form like this…

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

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

‘sliced at 1’



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

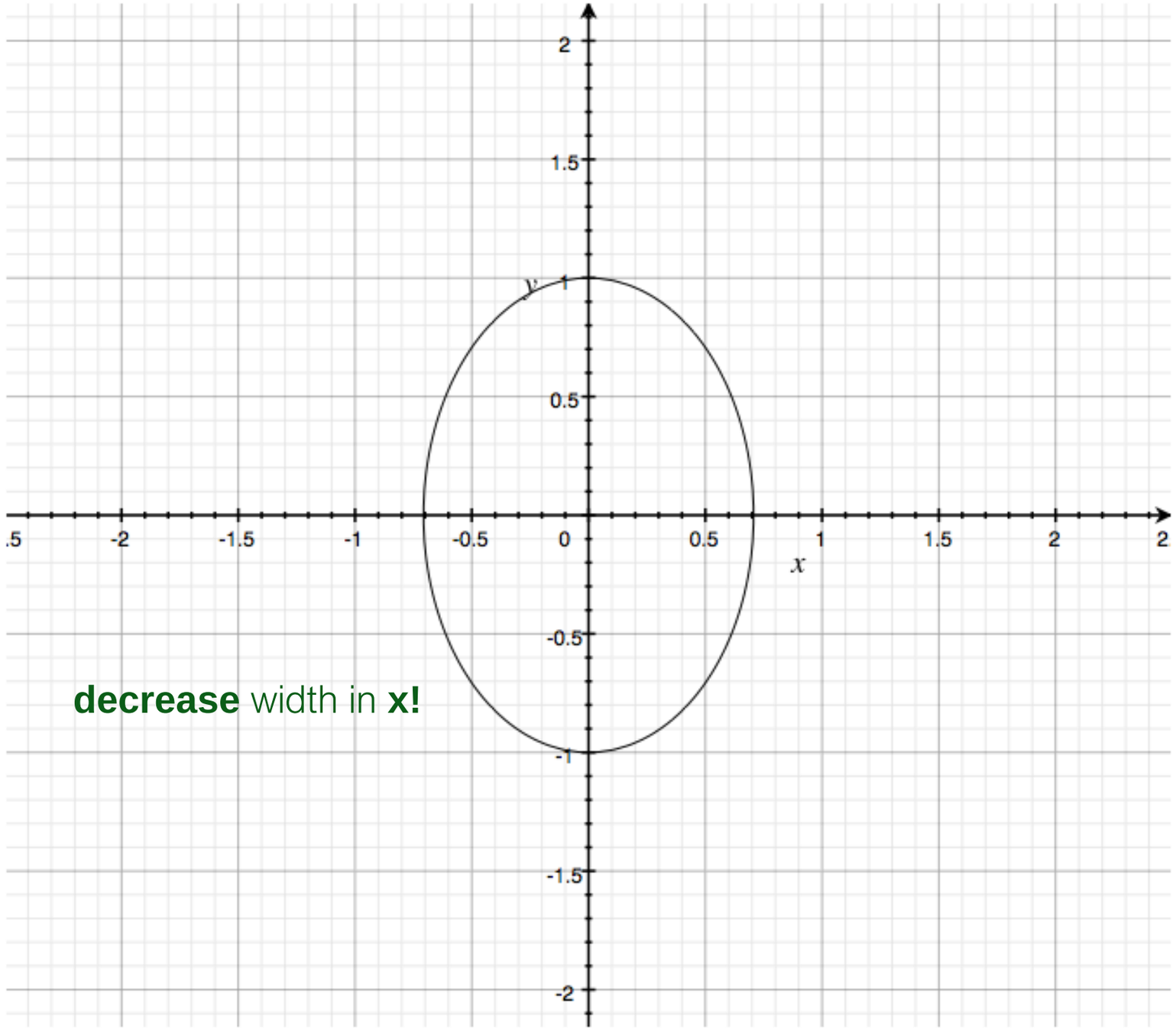
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1

*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!**

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

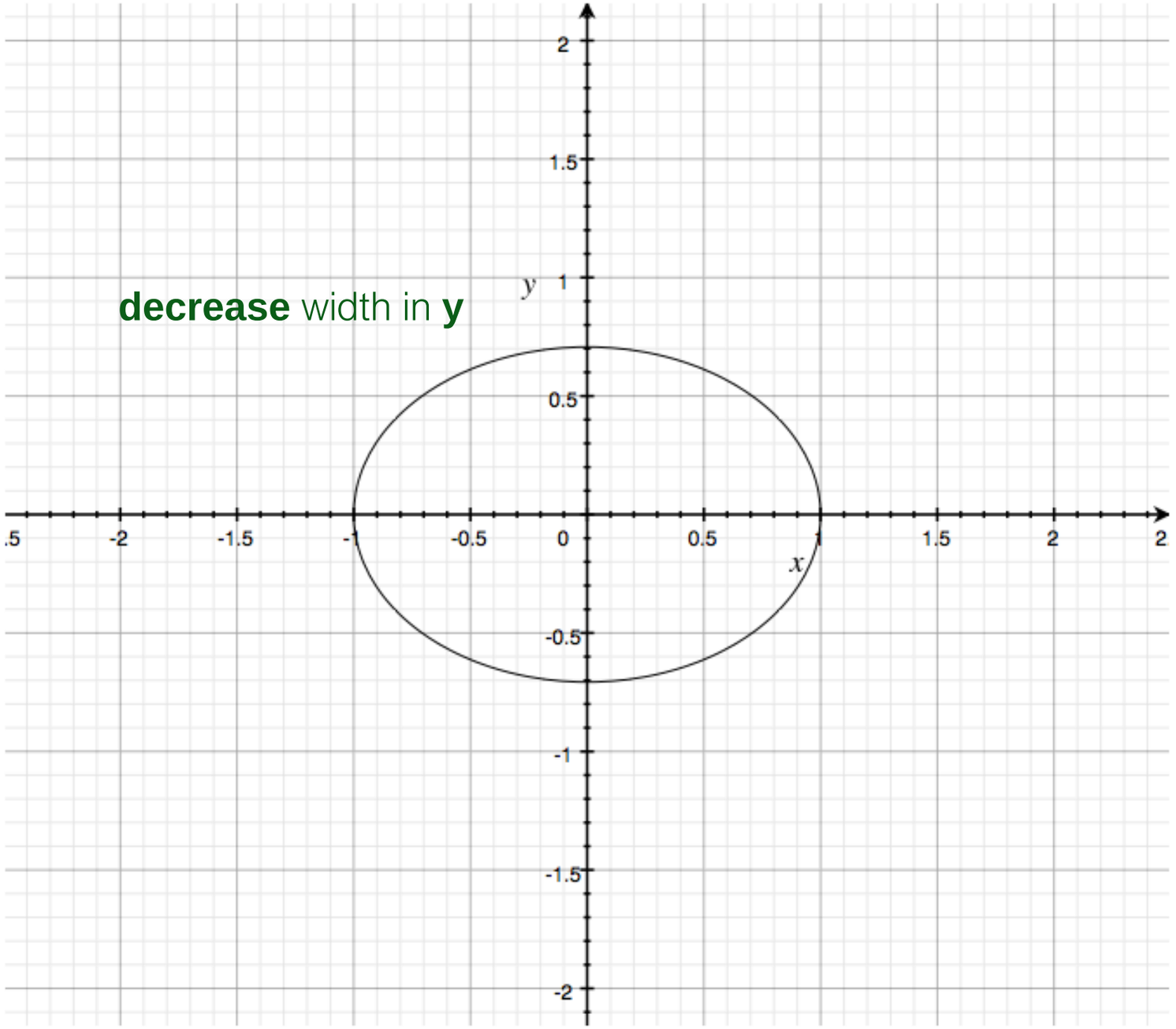
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and slice at 1

*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



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

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?

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

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)

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

eigenvectors

axis of the ‘ellipse slice’

eigenvalues along diagonal

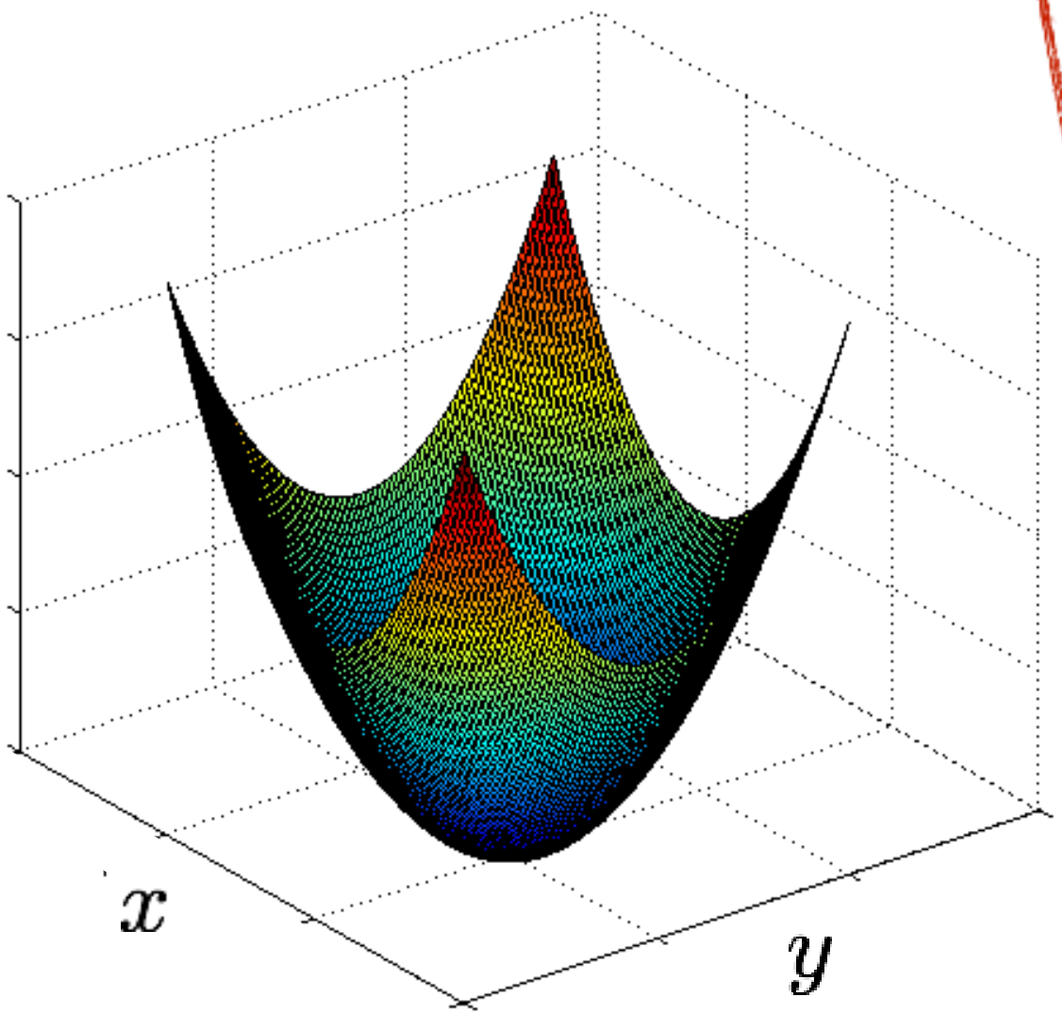
Inverse sqr of length of the quadratic along the axis

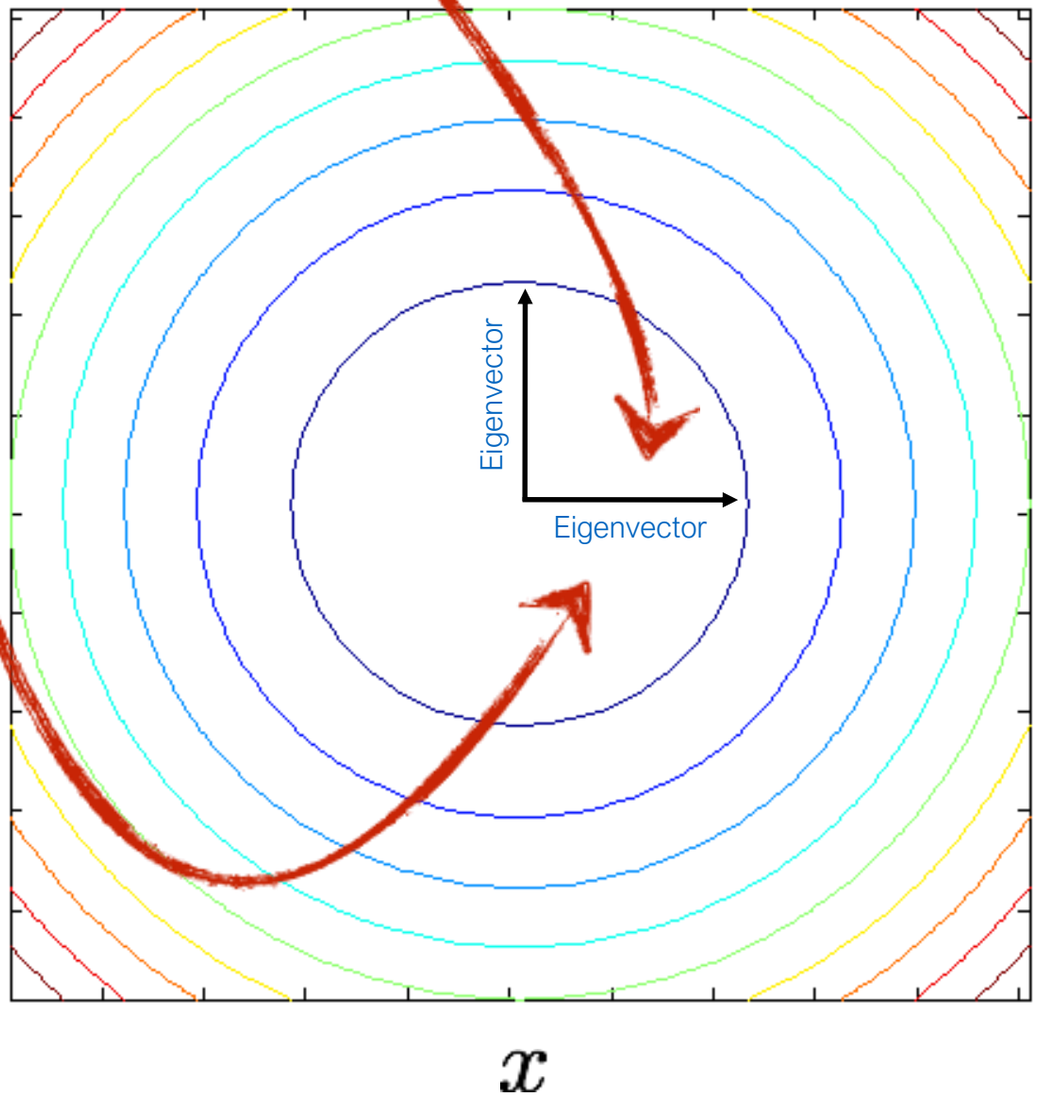
Eigenvectors Eigenvalues

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

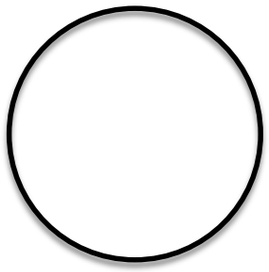
Eigenvectors

Inverse sqr of the size of the axis



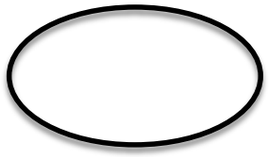


Recall:



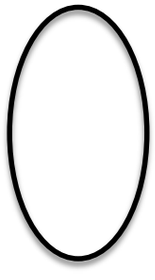
$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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



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

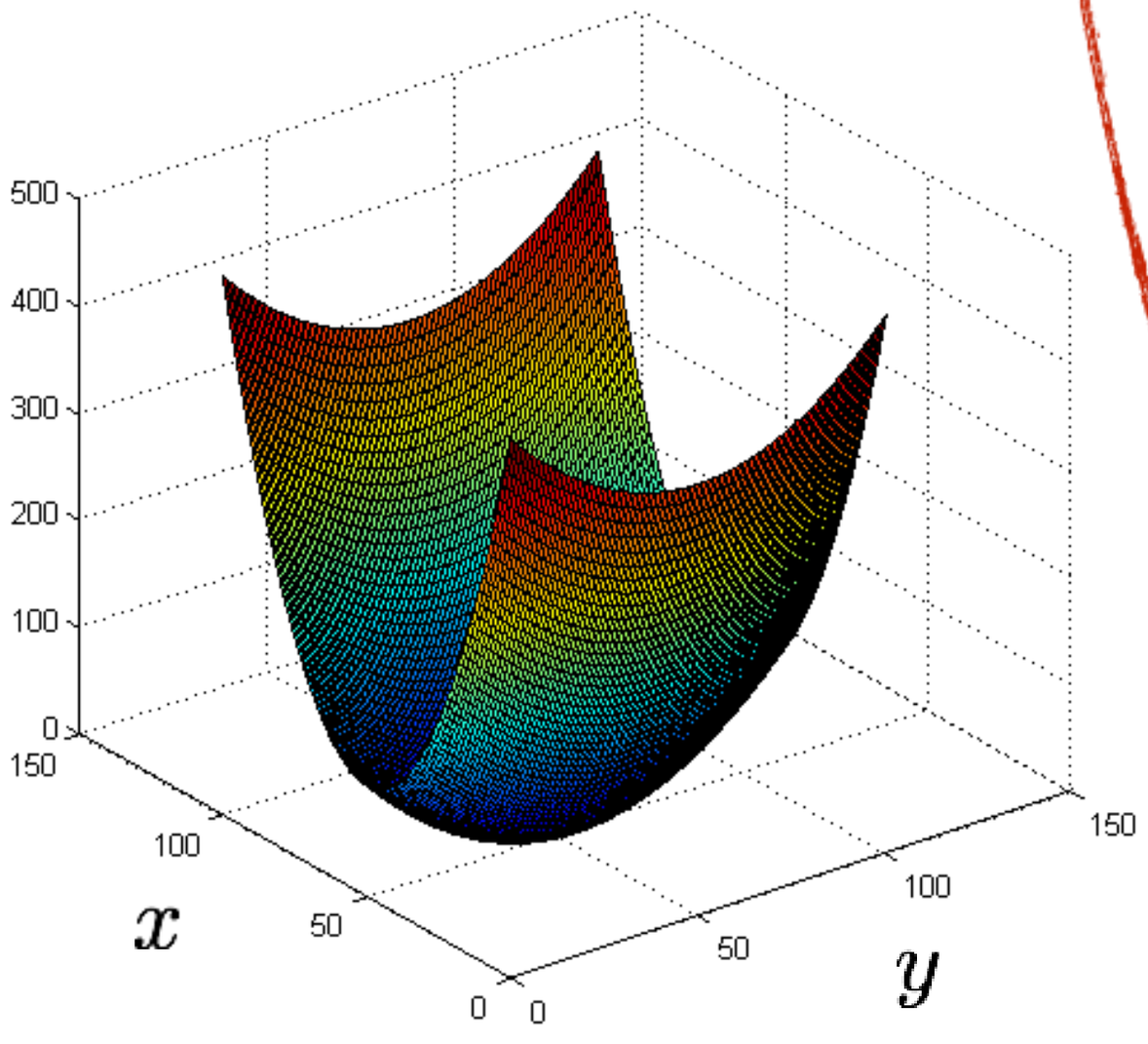
$$\mathbf{A} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T$$

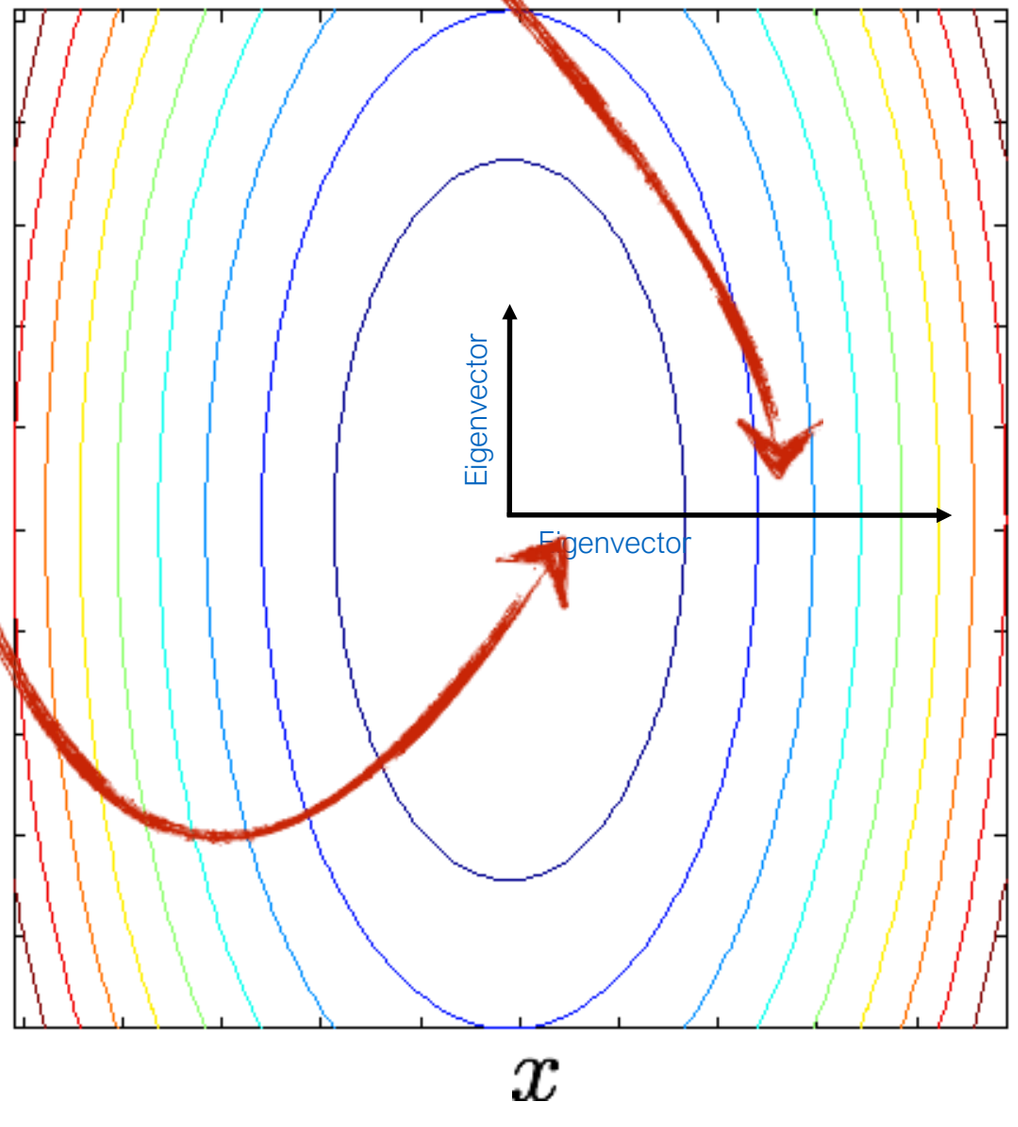
Eigenvalues

Eigenvectors

Eigenvectors

Inverse sqr of length of axis



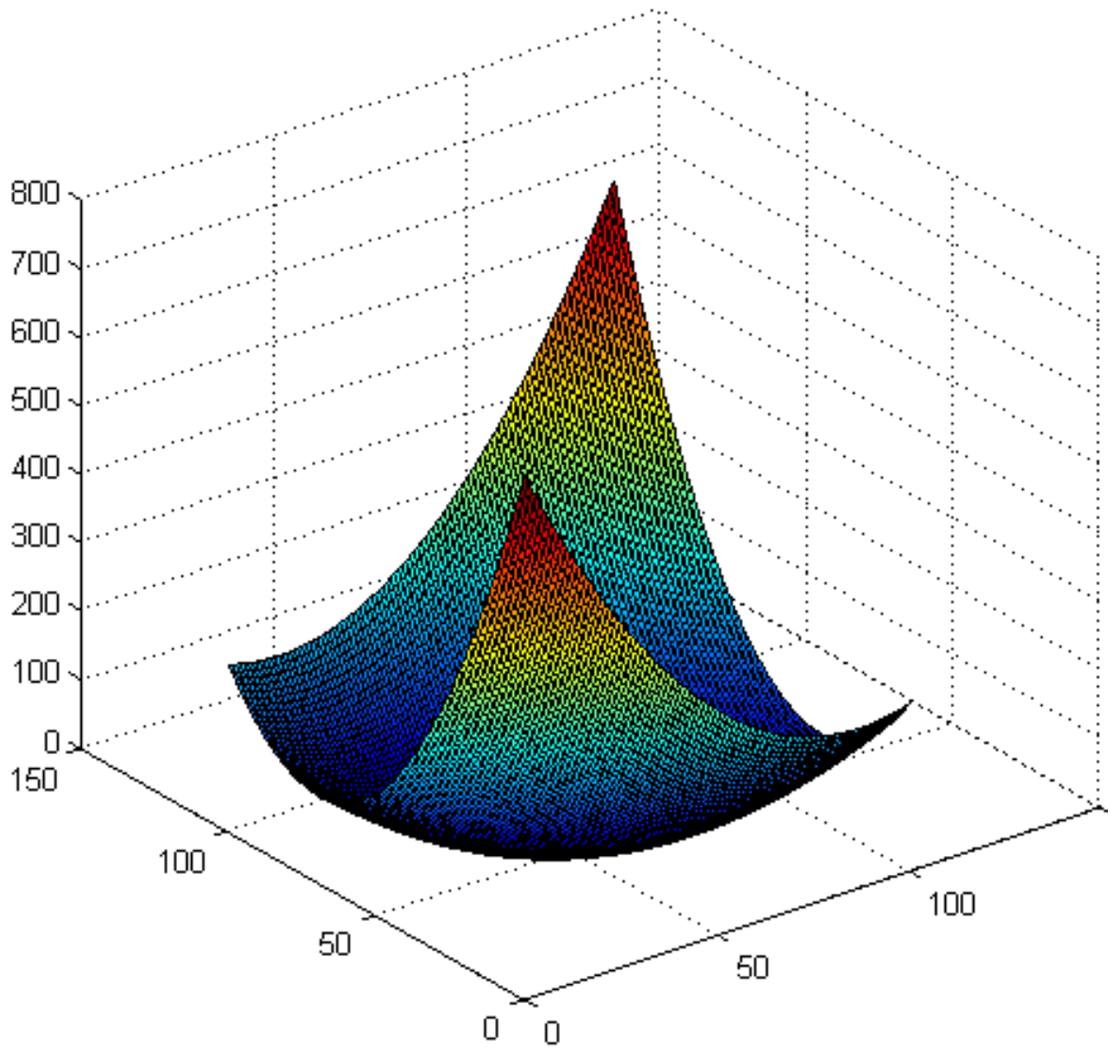


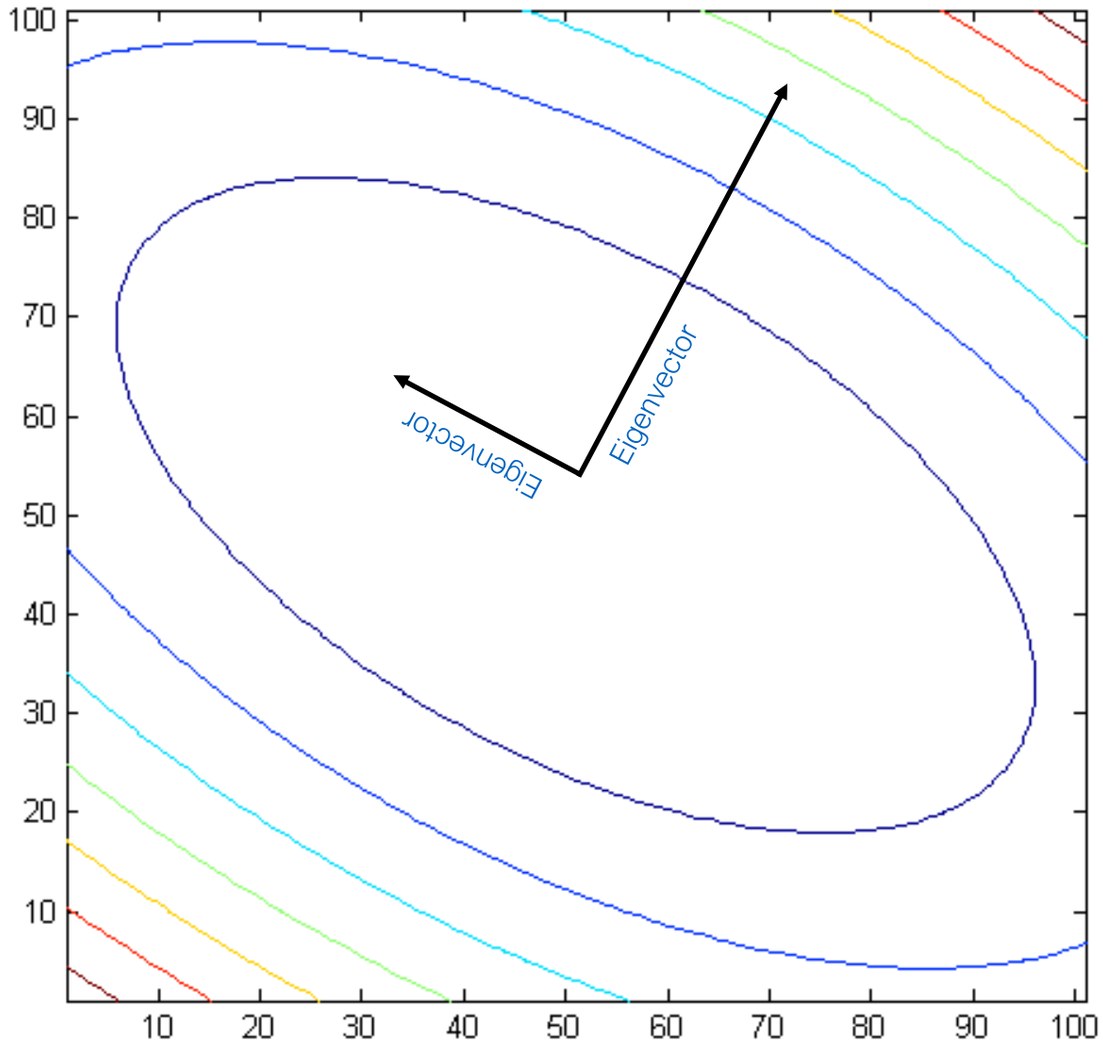
Eigenvalues

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

Eigenvectors

Eigenvectors



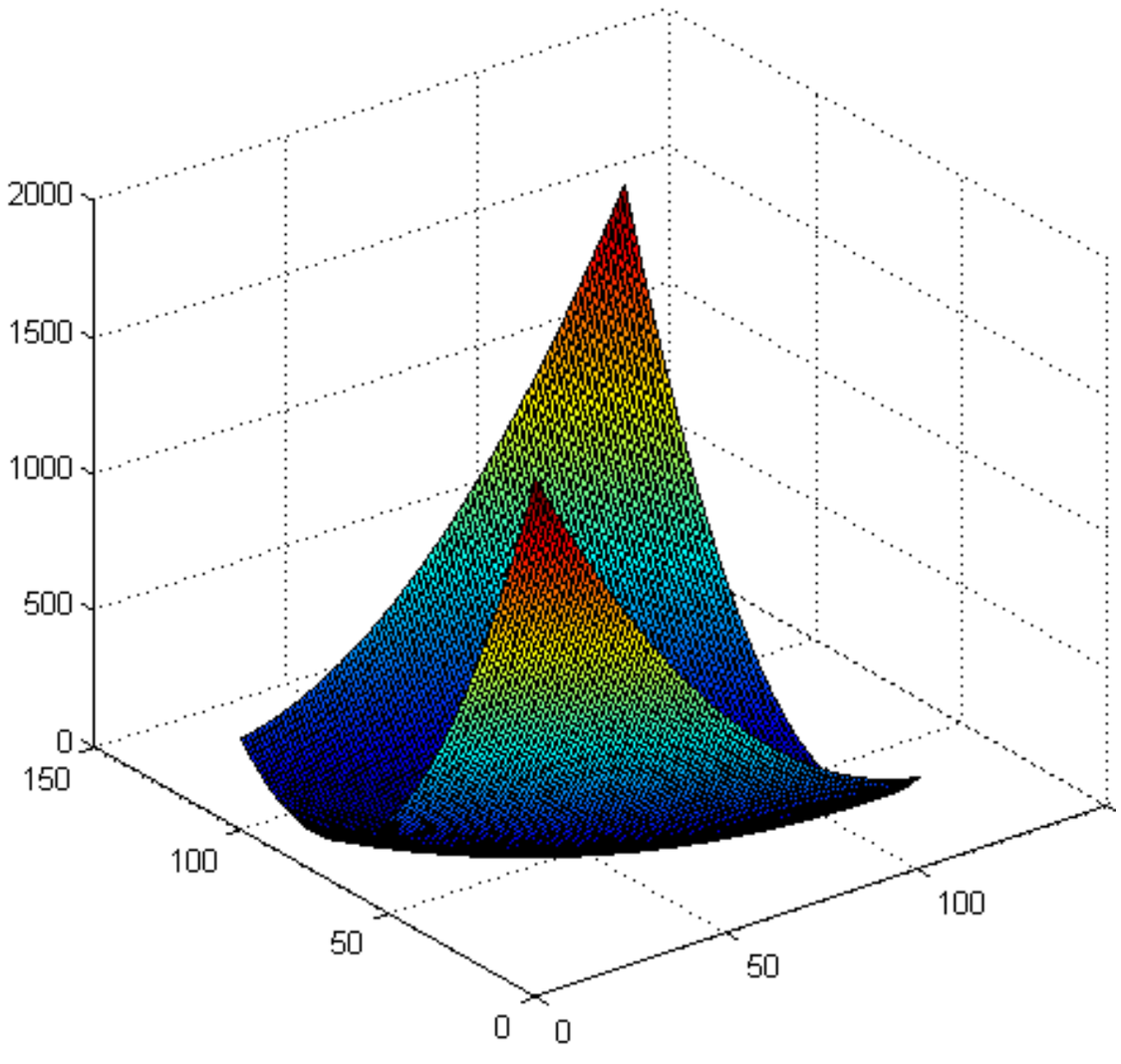


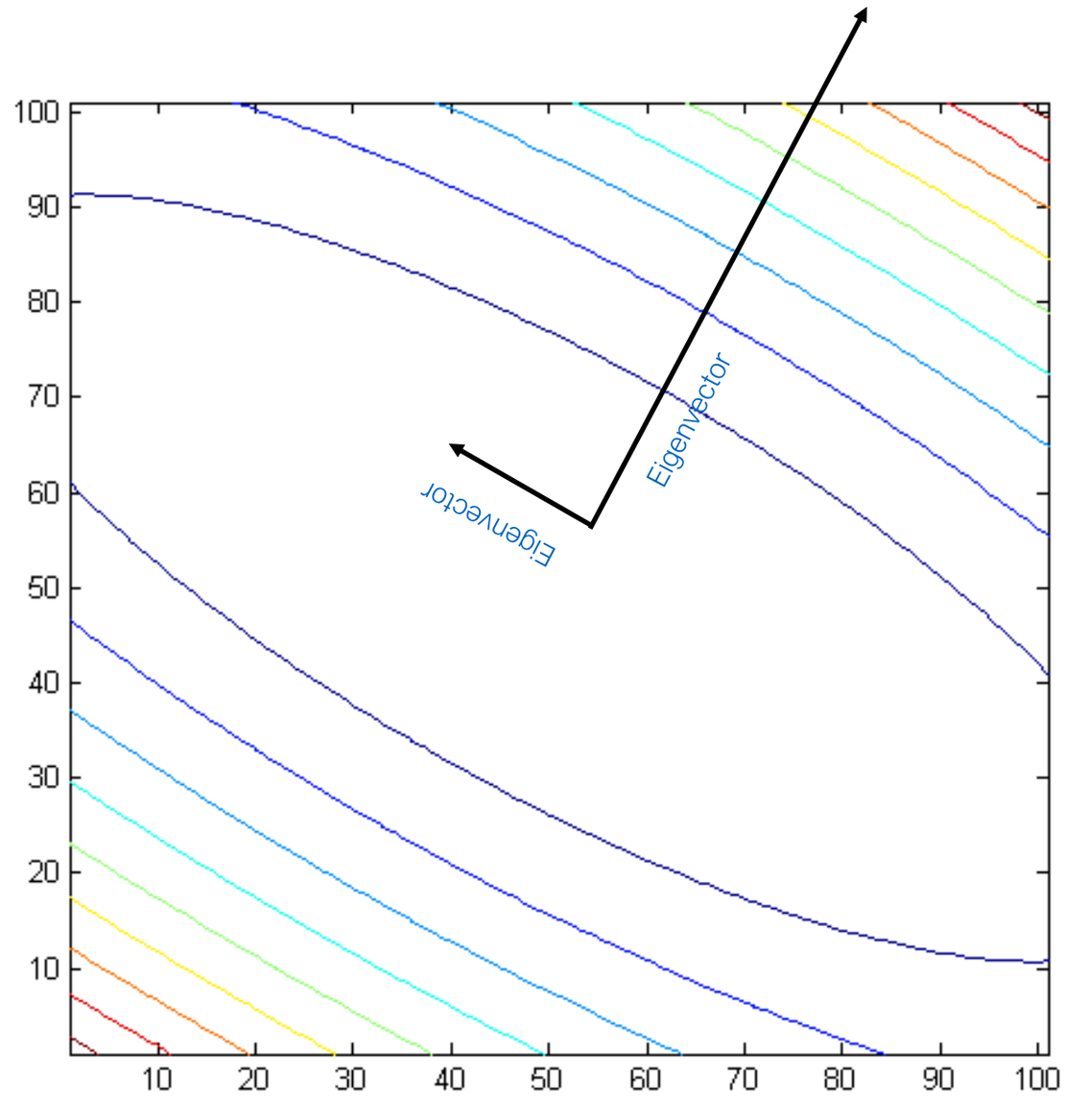
Eigenvalues

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

Eigenvectors

Eigenvectors





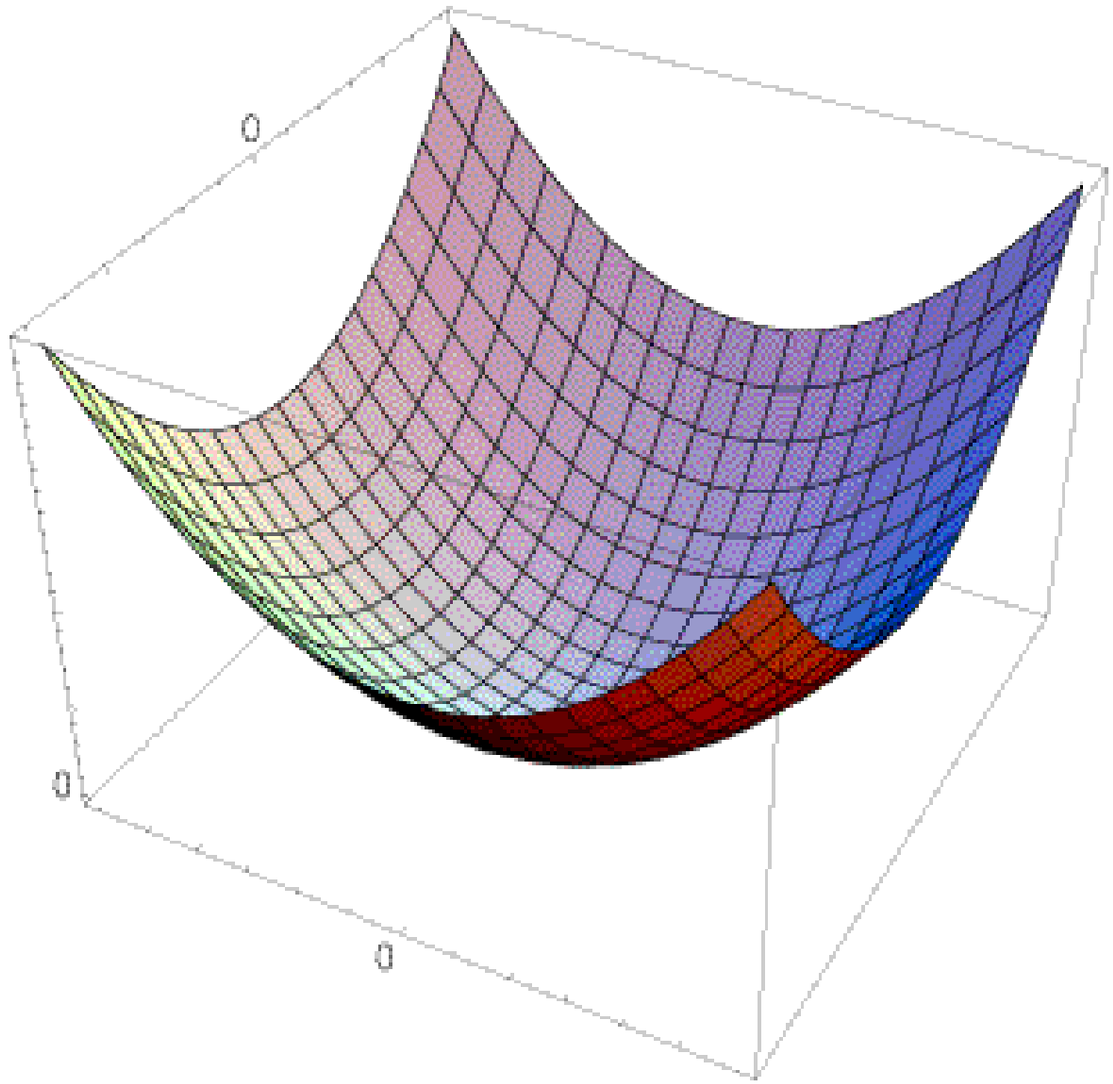
We will need this to understand the…

Error function for Harris Corners

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

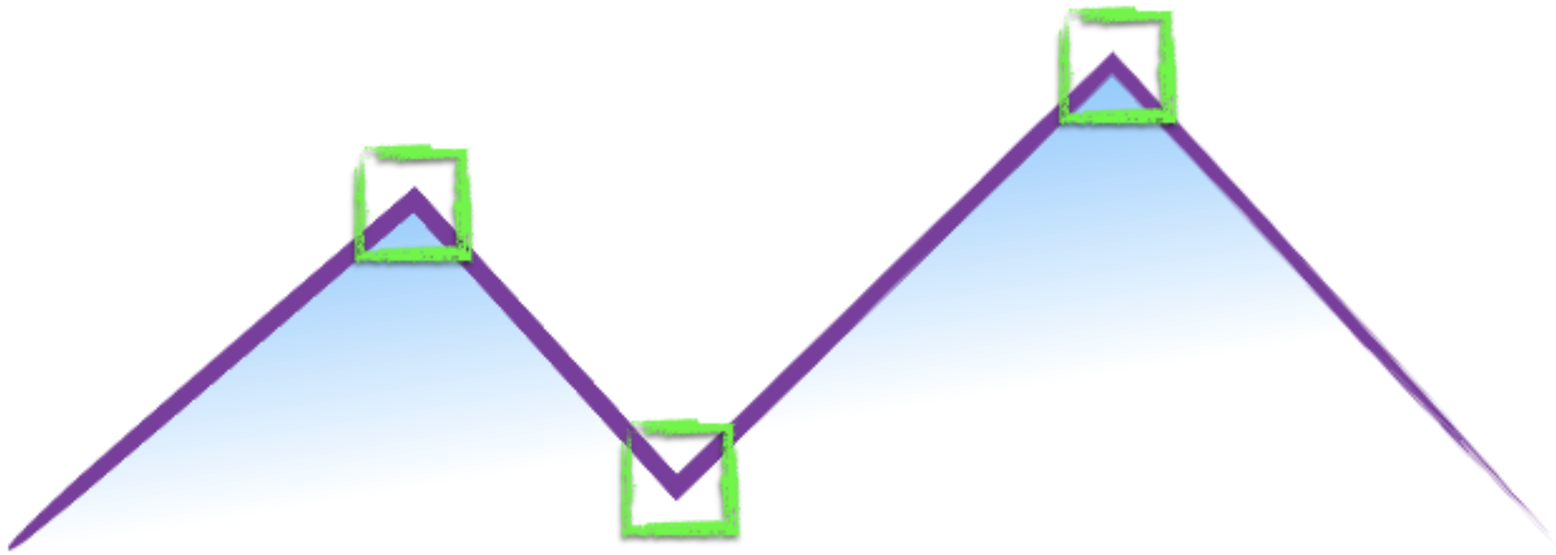
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

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



Harris corner detector

How do you find a corner?

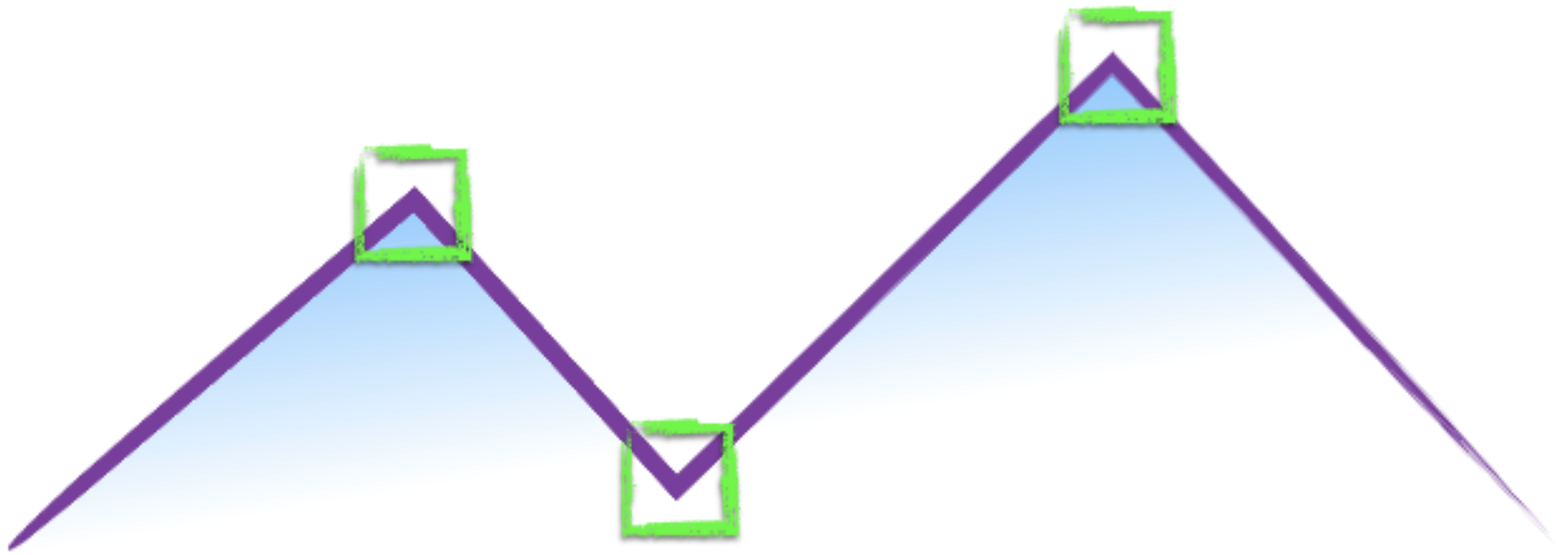


Поиск характерной точки на изображении

- Как понять что выбранная область содержит характерную точку?
- Область вокруг характерной точки должна сильно варьироваться
- В области характерной точки небольшой сдвиг изображения должен приводить к существенному различию по сравнению с исходным изображением

How do you find a corner?

[Moravec 1980]

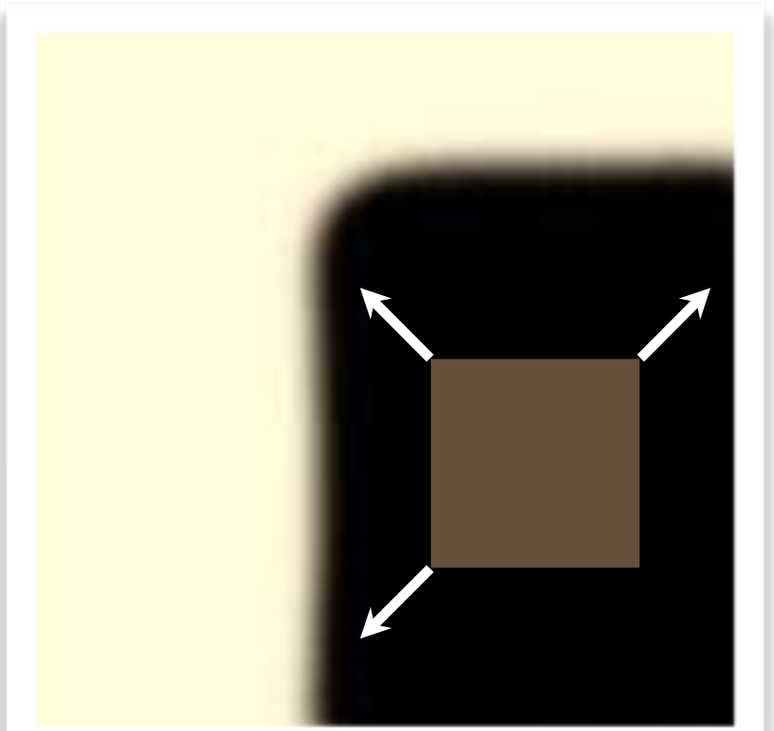


Easily recognized by looking through a small window

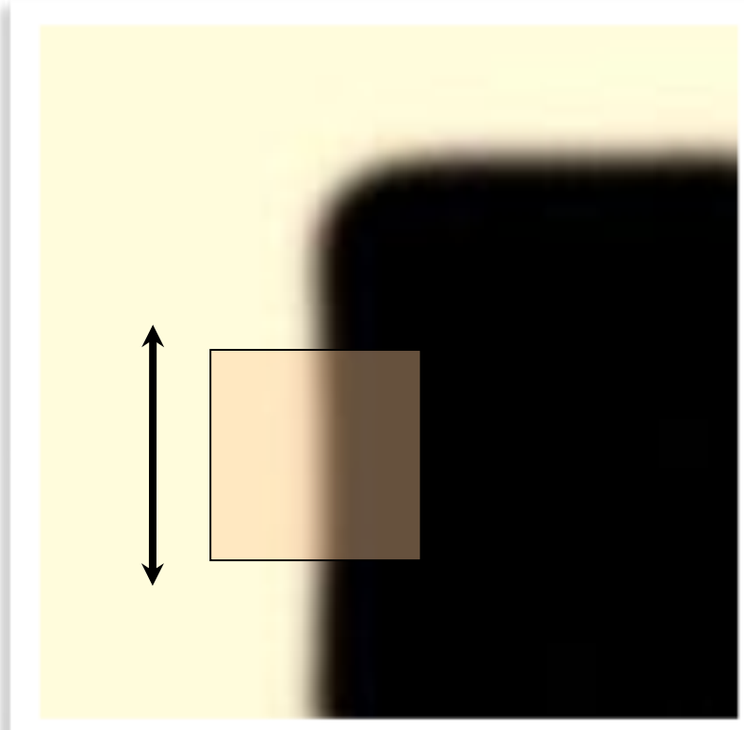
Shifting the window should give large change in intensity

Easily recognized by looking through a small window

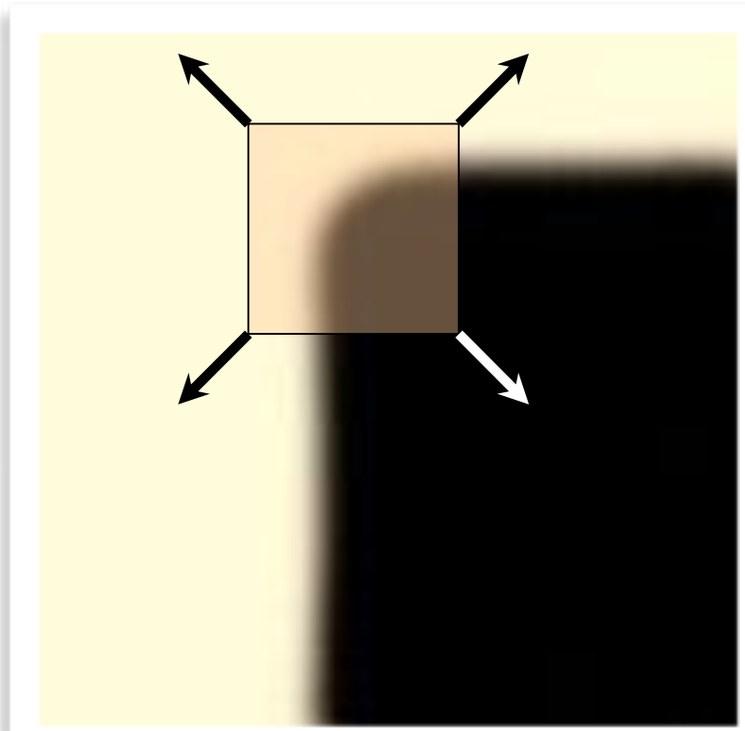
Shifting the window should give large change in intensity



“flat” region:
no change in all directions



“edge”:
no change along the edge direction



“corner”:
significant change in all directions

[Moravec 1980]

Design a program to detect corners
(hint: use image gradients)

Finding corners
(a.k.a. PCA)

1. Compute image gradients over small region

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

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

2. Subtract mean from each image gradient

3. Compute the covariance matrix

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

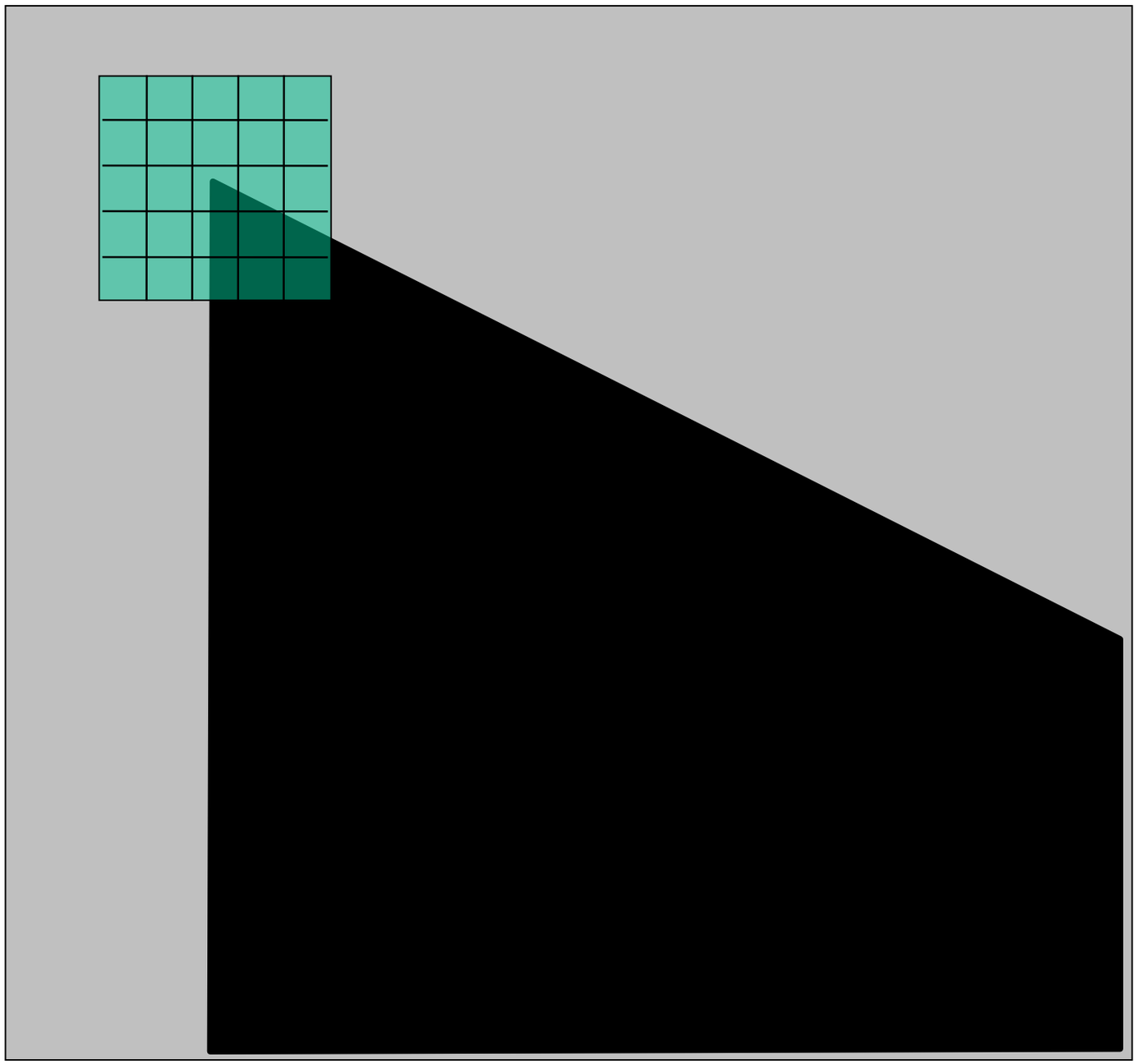
4. Compute eigenvectors and eigenvalues

5. Use threshold on eigenvalues to detect corners

1. Compute image gradients over a small region (not just a single pixel)

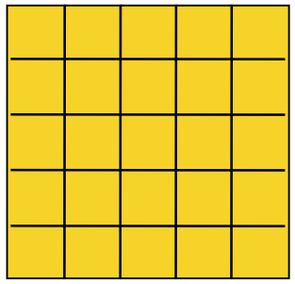
1. Compute image gradients over a small region

(not just a single pixel)



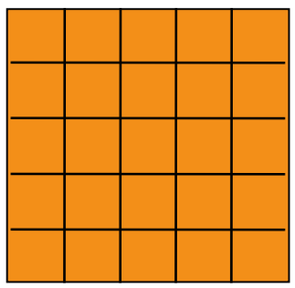
array of x gradients

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



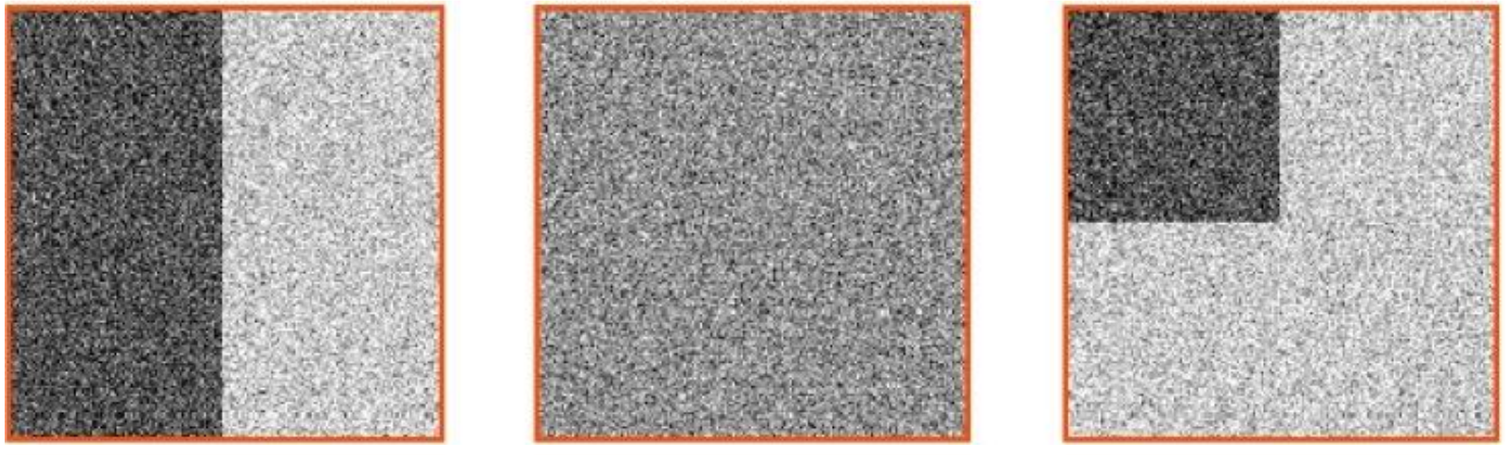
array of y gradients

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

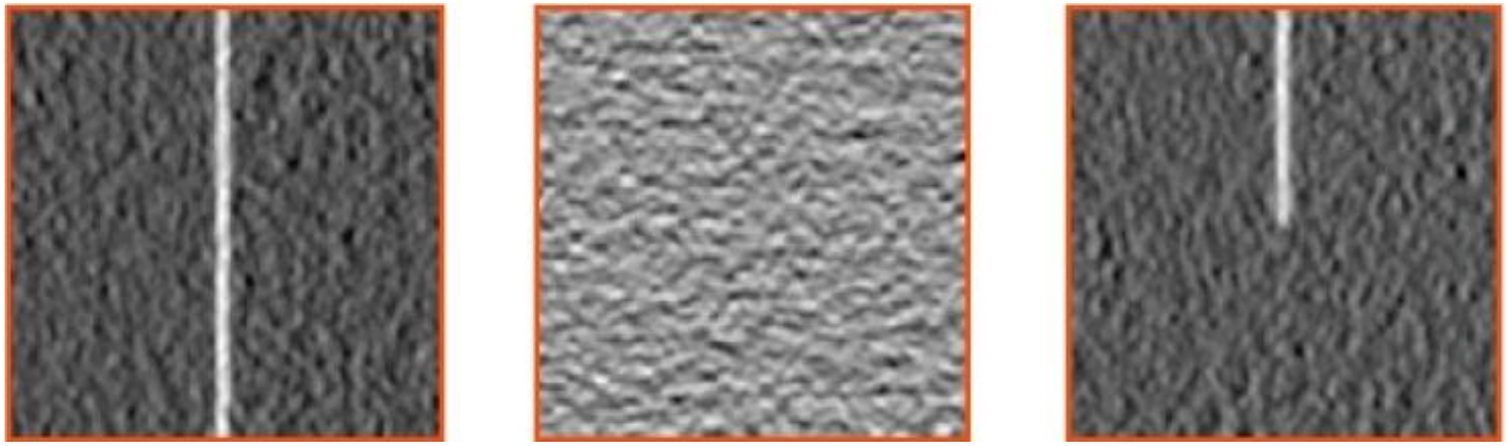


visualization of gradients

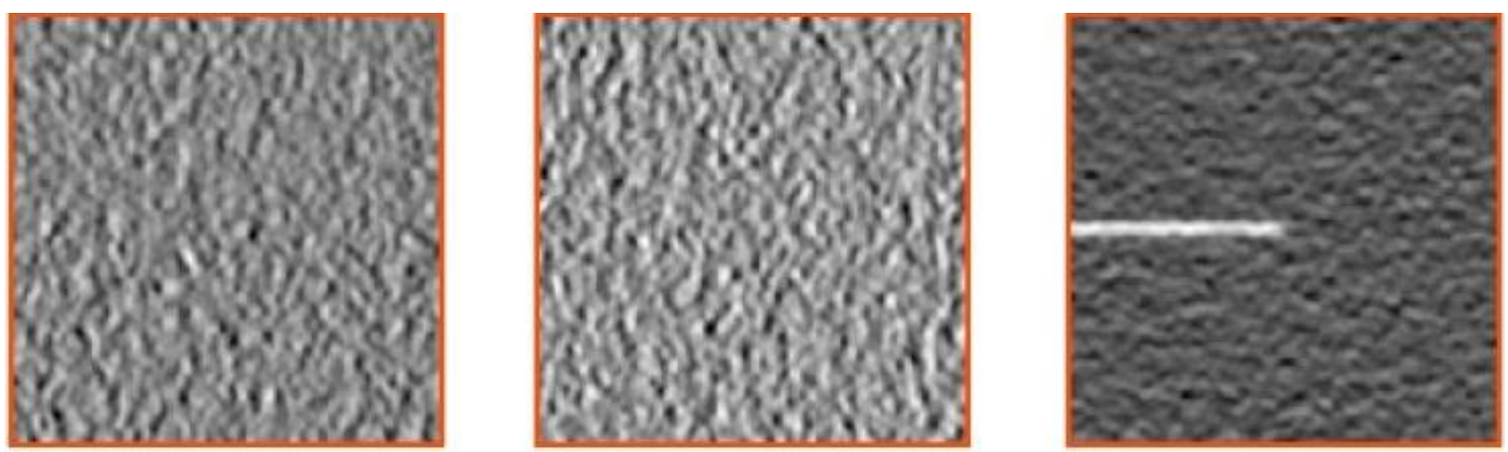
image

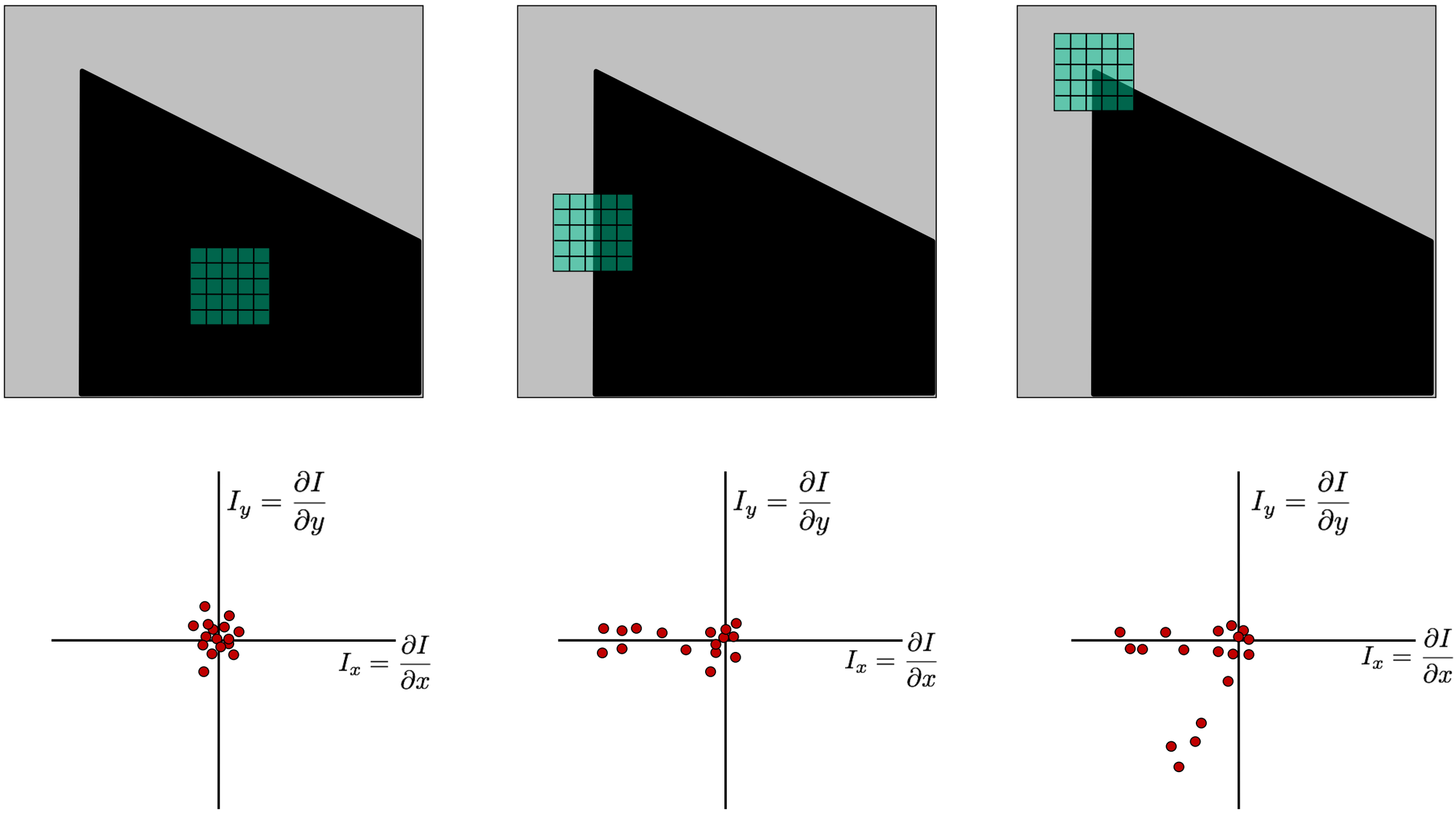


X derivative

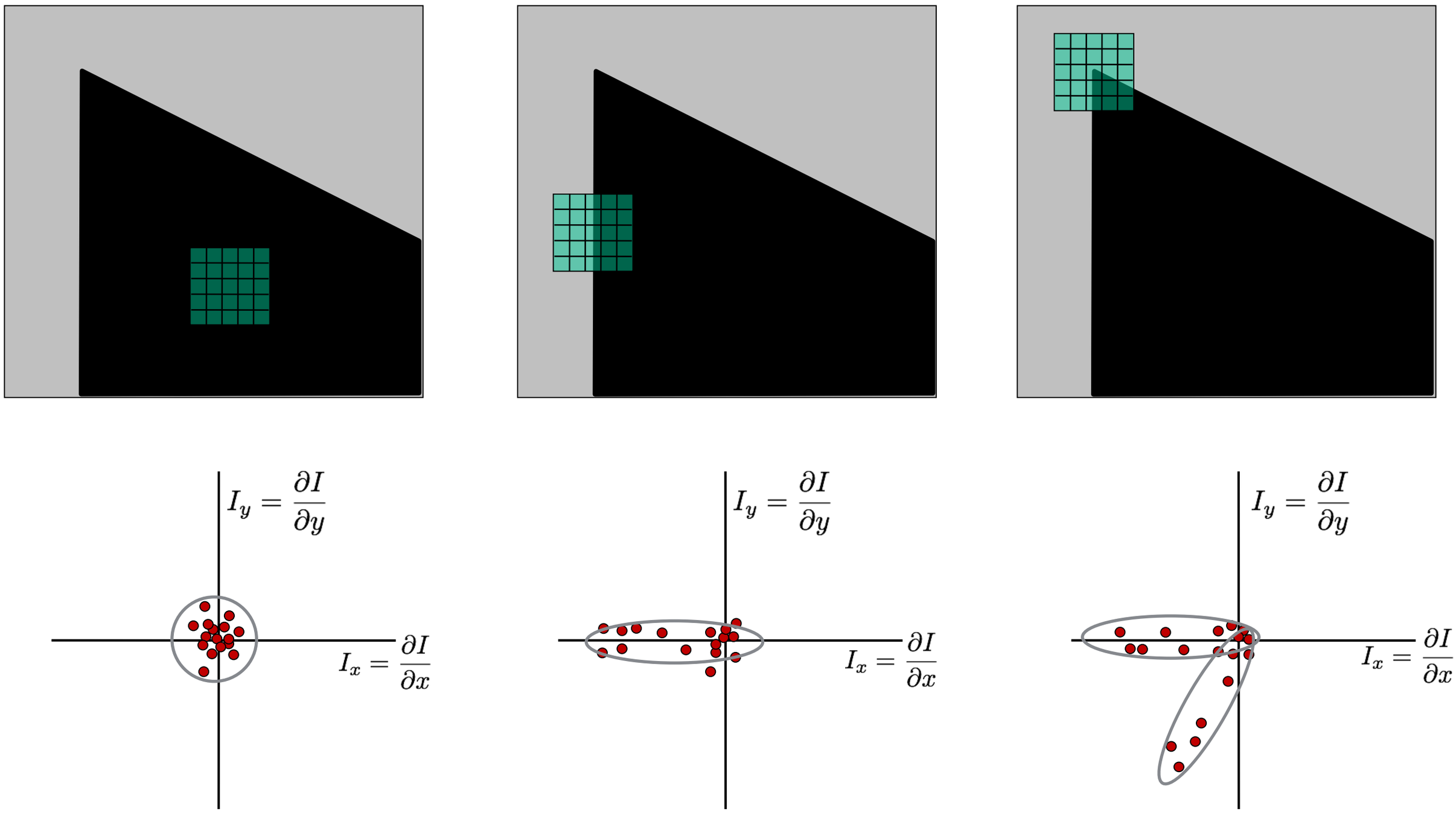


Y derivative

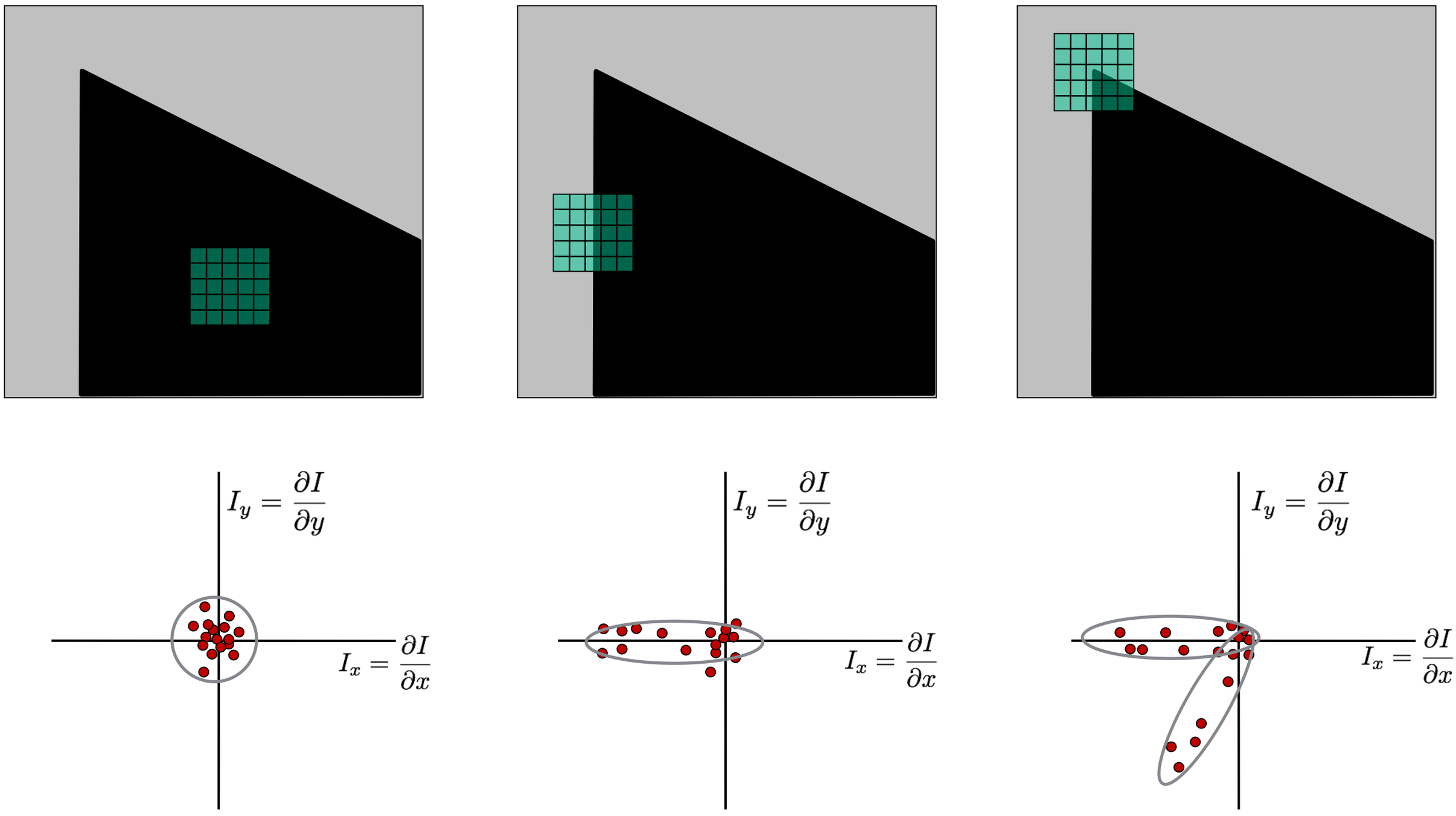




What does the distribution tell you about the region?



distribution reveals edge orientation and magnitude

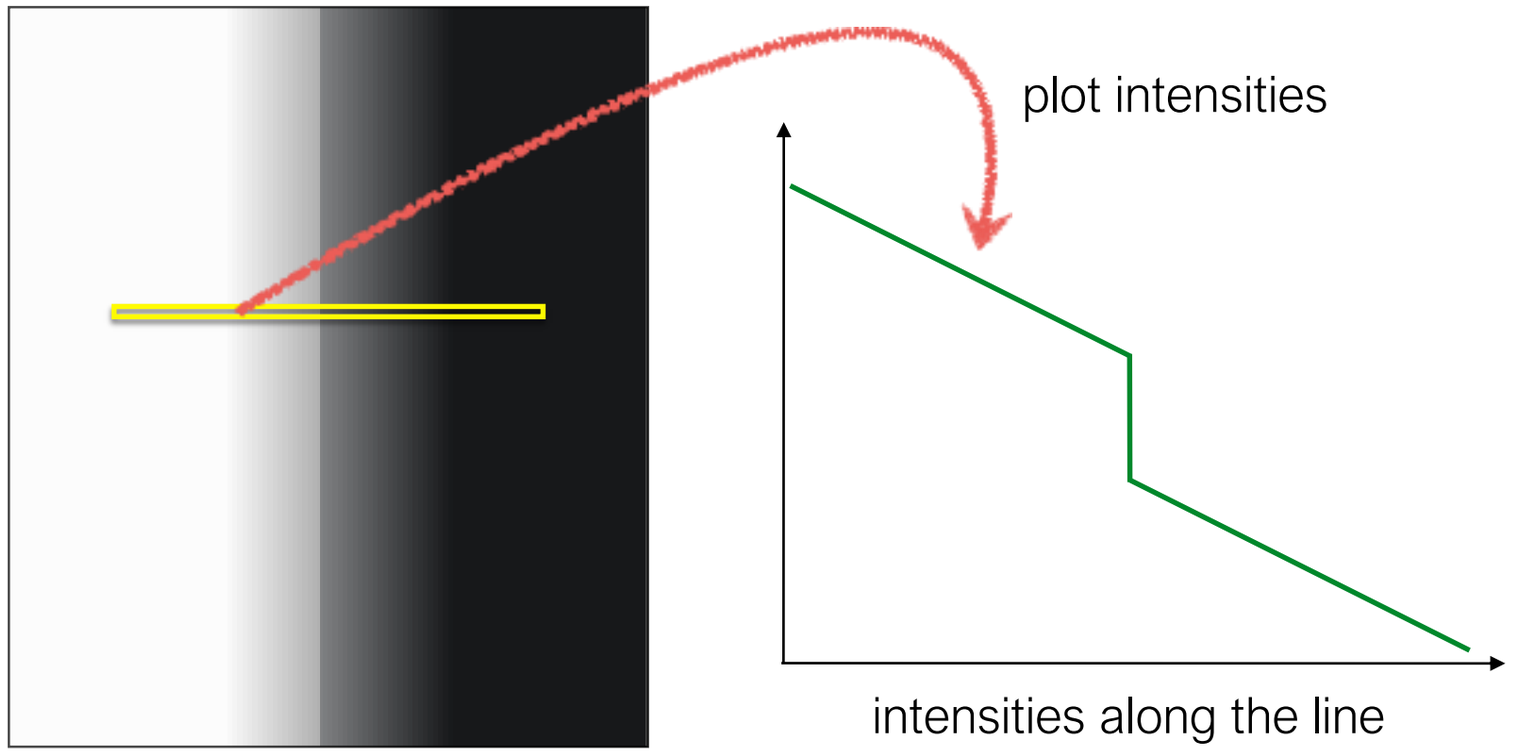


How do you quantify orientation and magnitude?

2. Subtract the mean from each image gradient

2. Subtract the mean from each image gradient

constant intensity gradient



Поиск характерной точки на изображении

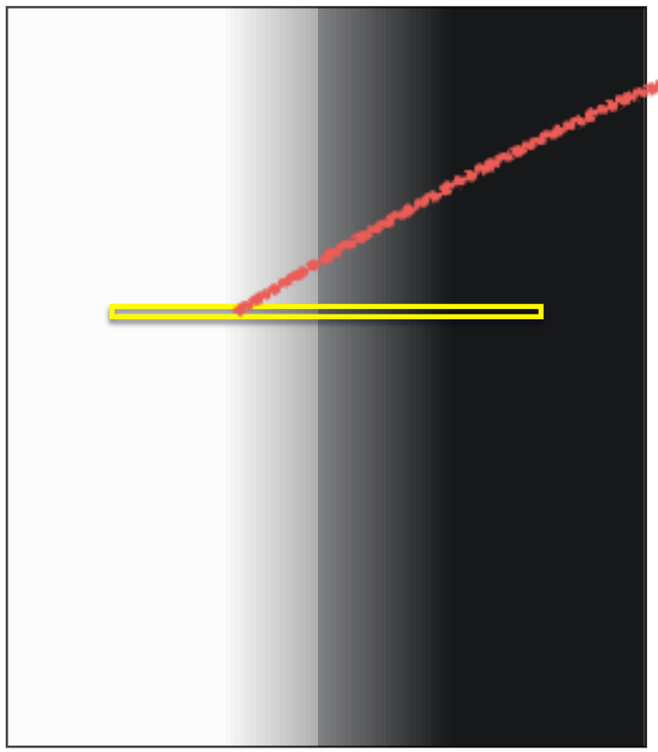
- Как понять что выбранная область содержит характерную точку?
- Область вокруг характерной точки должна сильно варьироваться
- В области характерной точки небольшой сдвиг изображения должен приводить к существенному различию по сравнению с исходным изображением

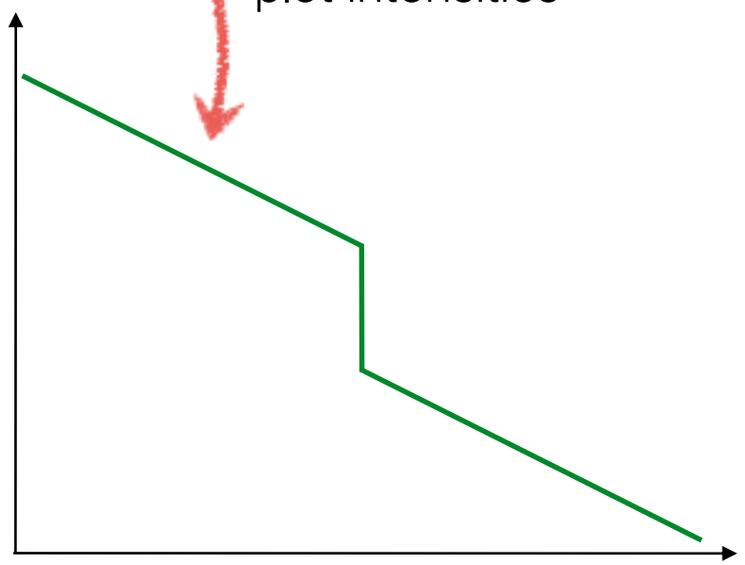
Поиск характерной точки на изображении

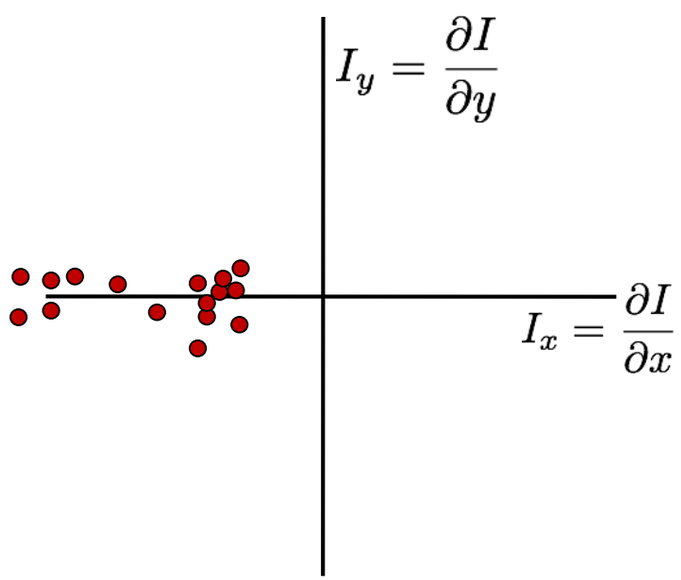
- Как понять что выбранная область содержит характерную точку?
- Область вокруг характерной точки должна сильно варьироваться
- В области характерной точки небольшой сдвиг изображения должен приводить к существенному различию по сравнению с исходным изображением

2. Subtract the mean from each image gradient

constant intensity gradient



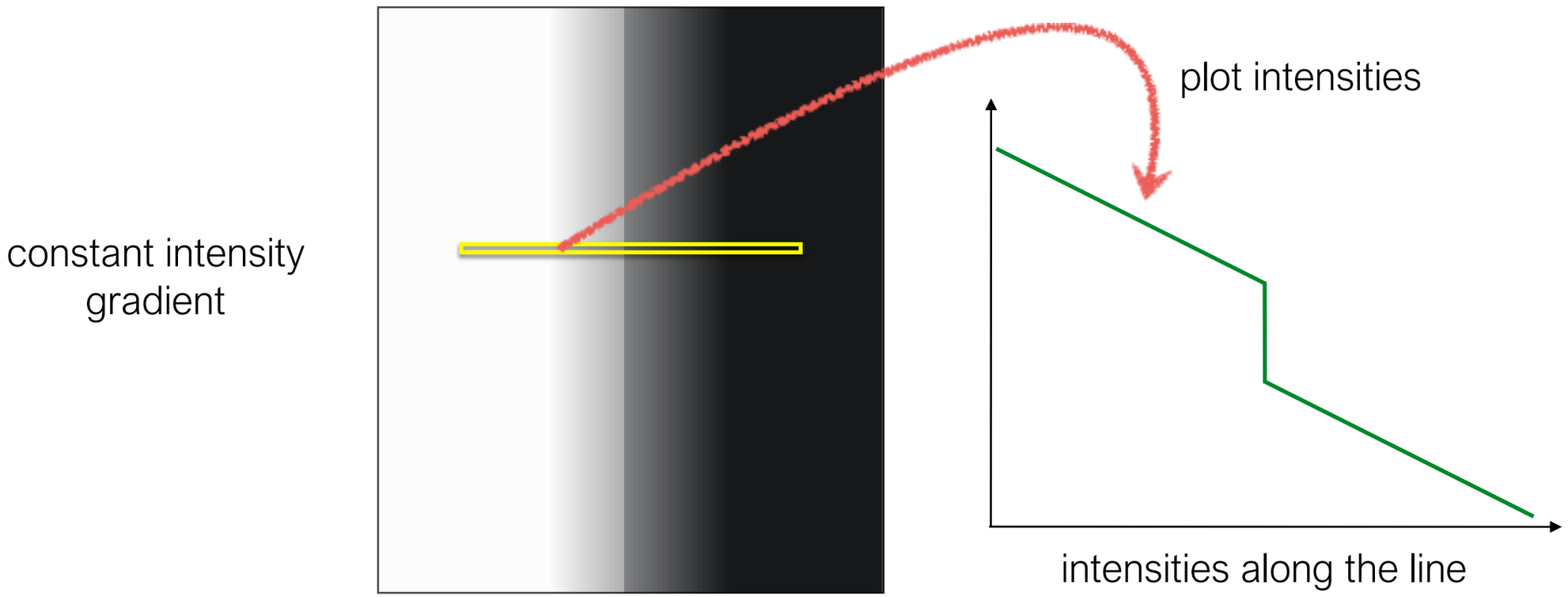


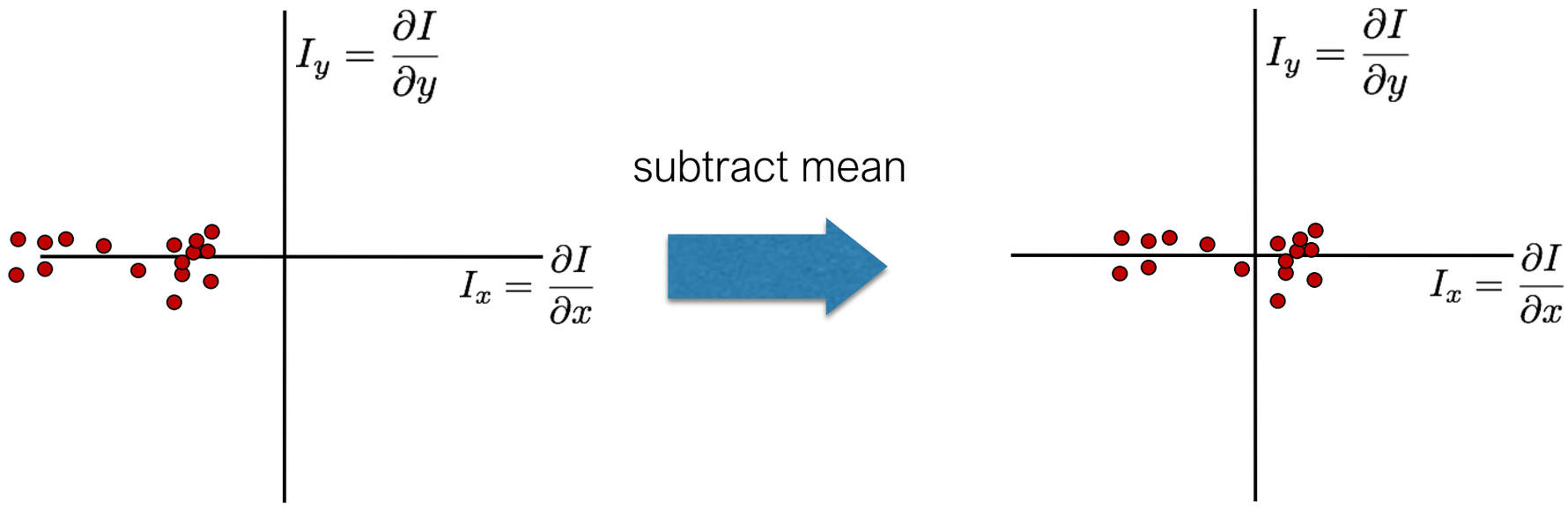


plot of image gradients

subtract mean

2. Subtract the mean from each image gradient





plot of image gradients

data is centered
('DC' offset is removed)

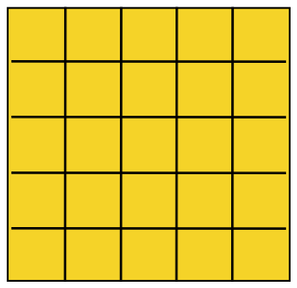
3. Compute the covariance matrix

3. Compute the covariance matrix

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

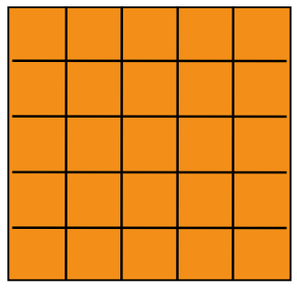
$$\sum_{p \in P} I_x I_y = \text{sum}(\quad .* \quad)$$

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



array of x gradients

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

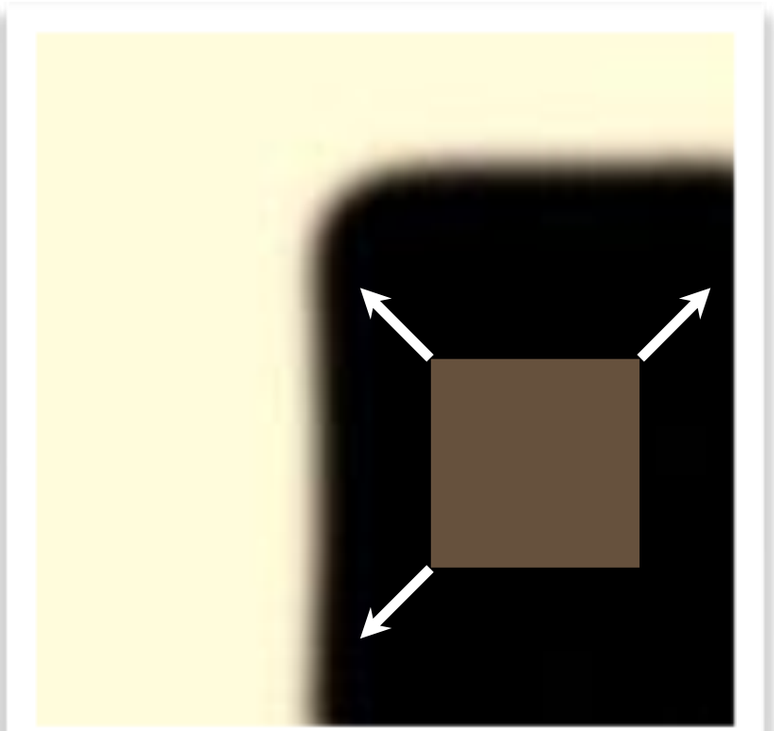


array of y gradients

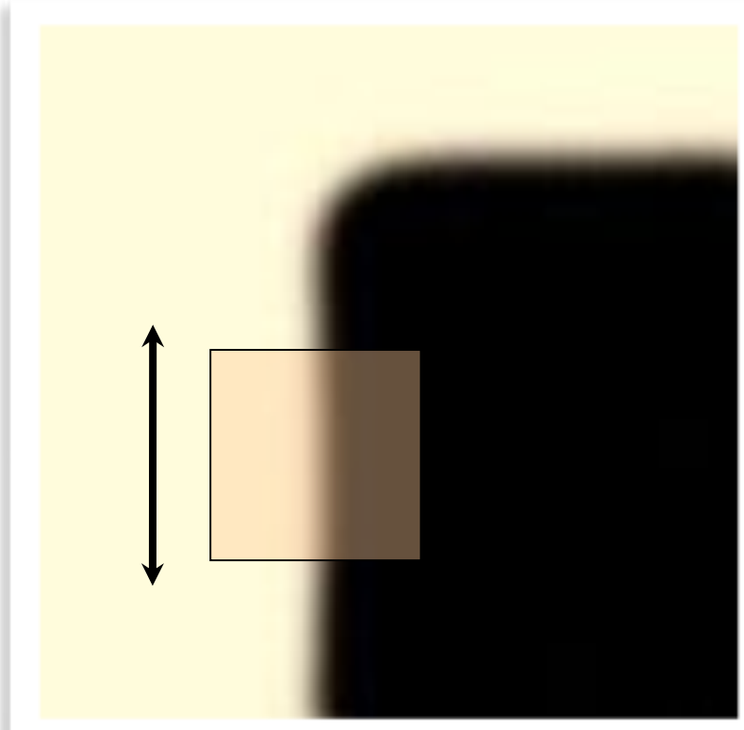
Where does this covariance matrix come from?

Easily recognized by looking through a small window

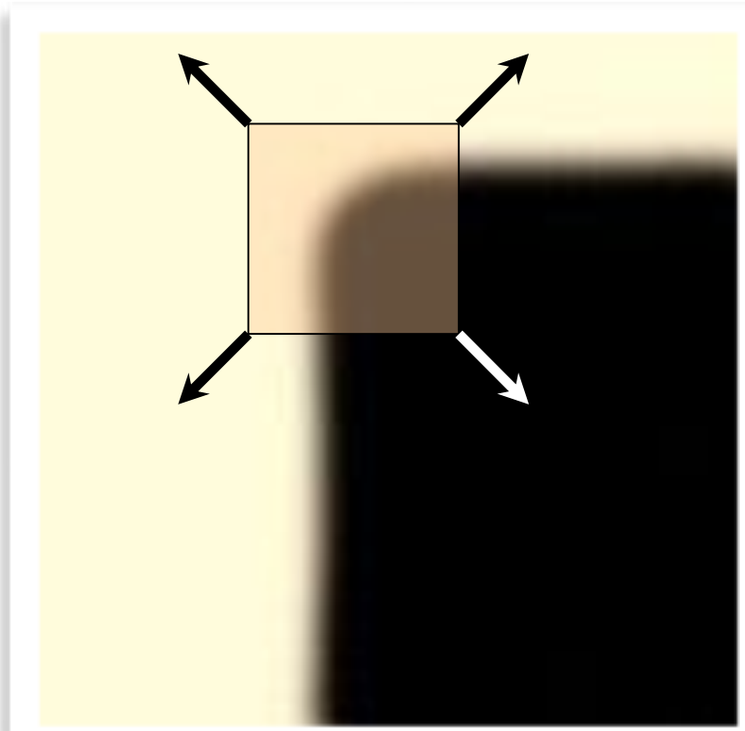
Shifting the window should give large change in intensity



“flat” region:
no change in all directions



“edge”:
no change along the edge direction



“corner”:
significant change in all directions

[Moravec 1980]

Some mathematical background…

Error function

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

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

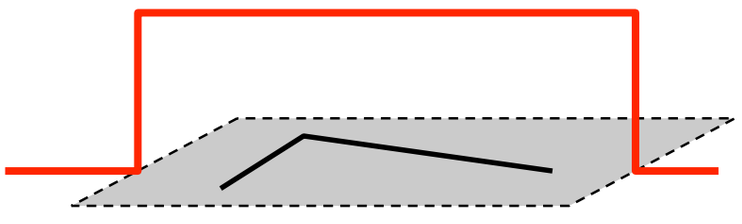
Error function

Window function

Shifted intensity

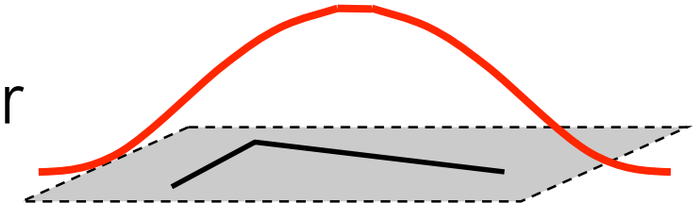
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Error function approximation

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

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

First-order Taylor expansion of I(x,y) about (0,0)
(bilinear approximation for small shifts)

Разложение в ряд Тейлора

Функцию f(x) в точке a можно разложить в ряд:

$$f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots$$

Математика детектора Харриса

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

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

Taylor Series

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

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

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

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

$$E(u,v) = \begin{pmatrix} u & v \end{pmatrix} M \begin{pmatrix} u \\ v \end{pmatrix}$$

Bilinear approximation

For small shifts $[u,v]$ we have a 'bilinear approximation':

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'

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

By computing the gradient covariance matrix…

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

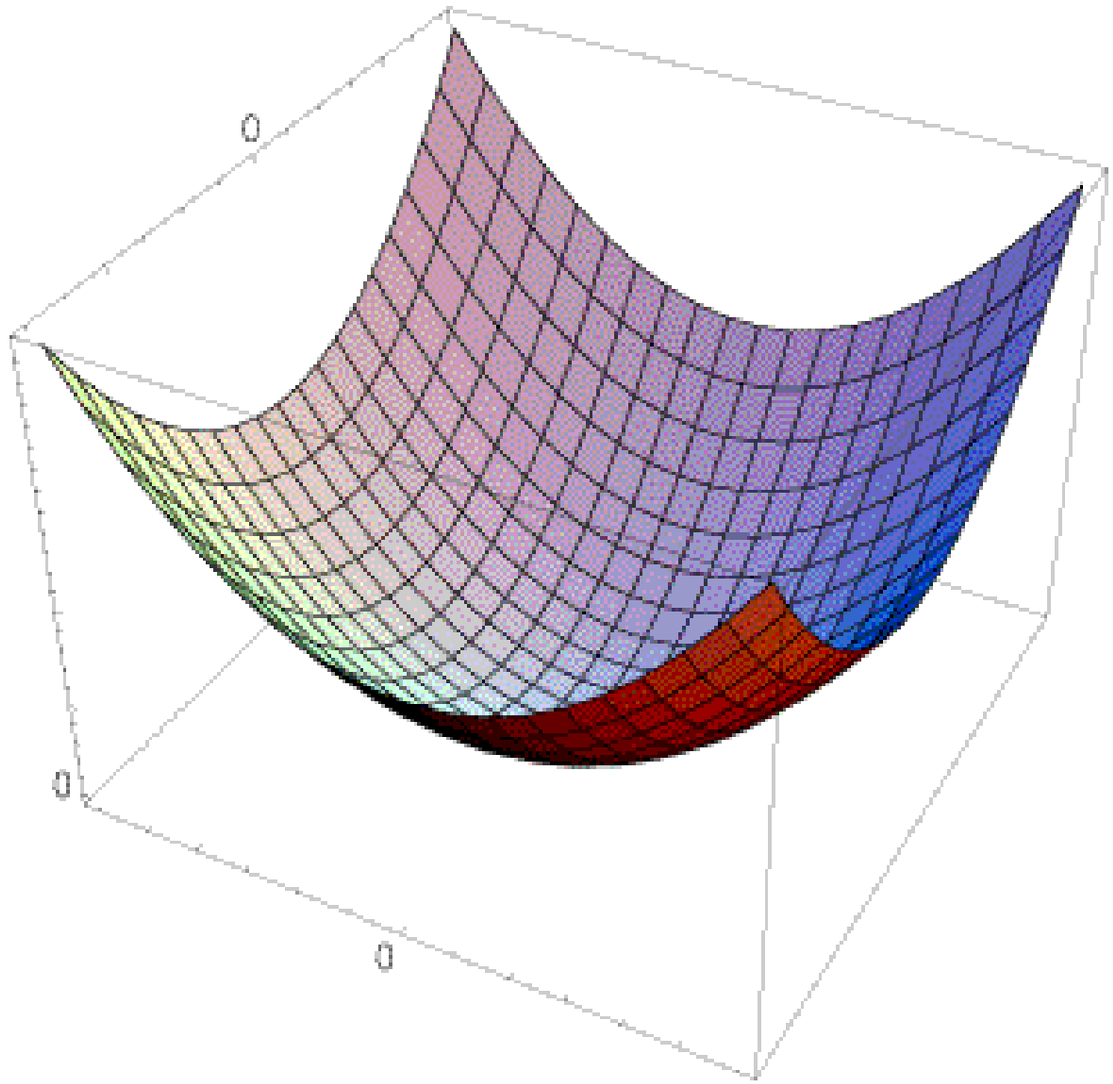
we are fitting a quadratic to the gradients over a small image region

Visualization of a quadratic

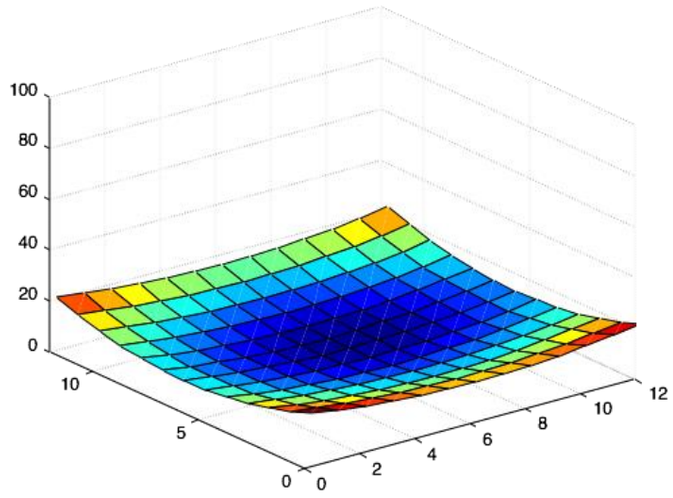
The surface $E(u,v)$ is locally approximated by a quadratic form

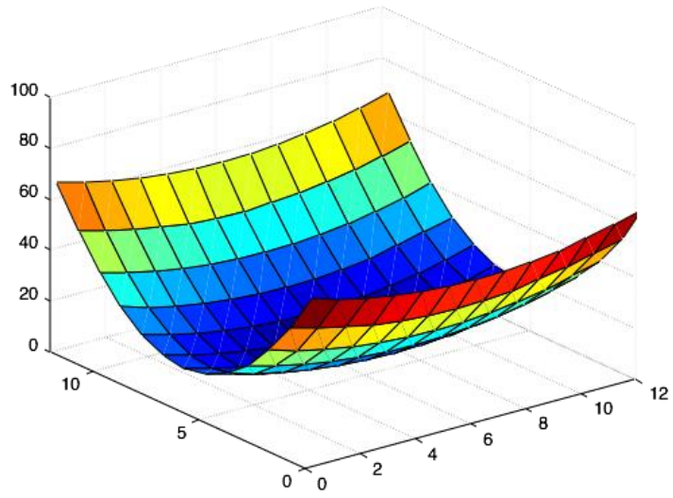
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

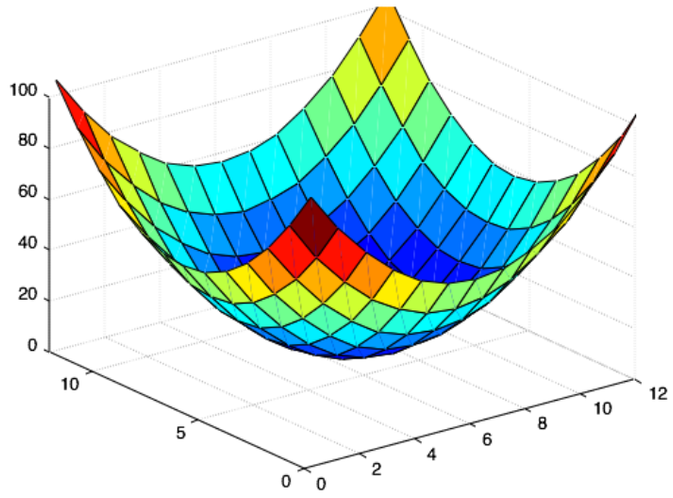
$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Which error surface indicates a good image feature?

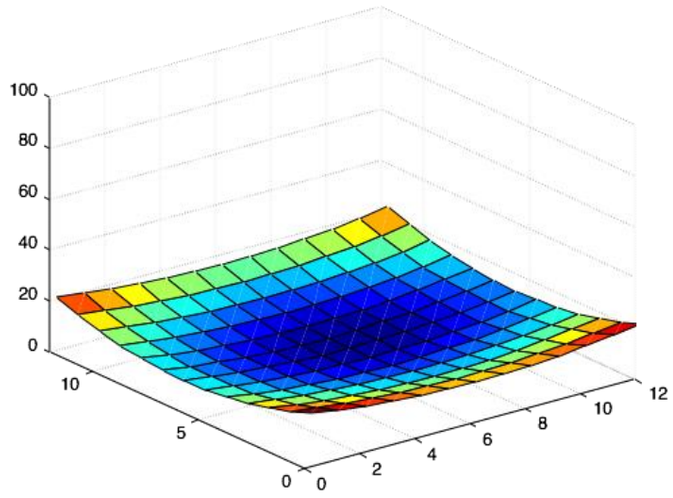




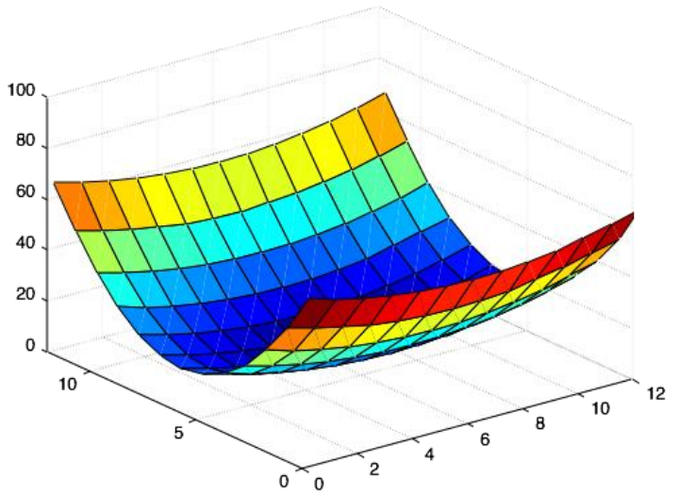


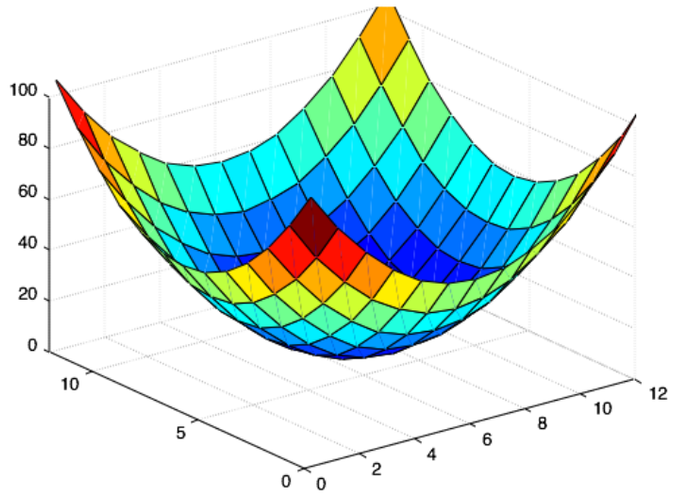
What kind of image patch do these surfaces represent?

Which error surface indicates a good image feature?

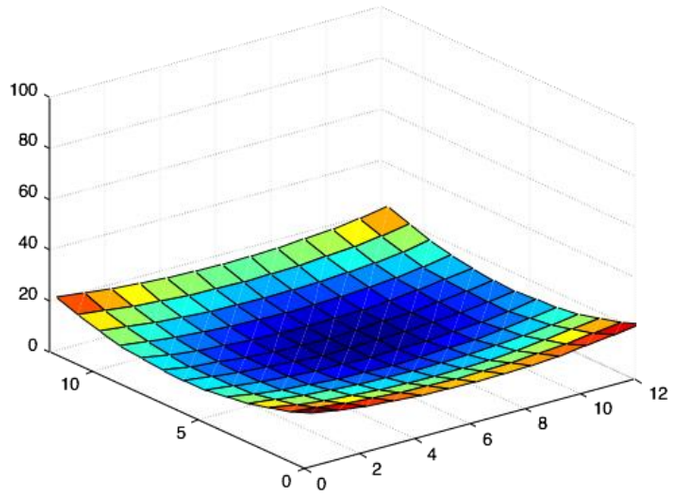


flat

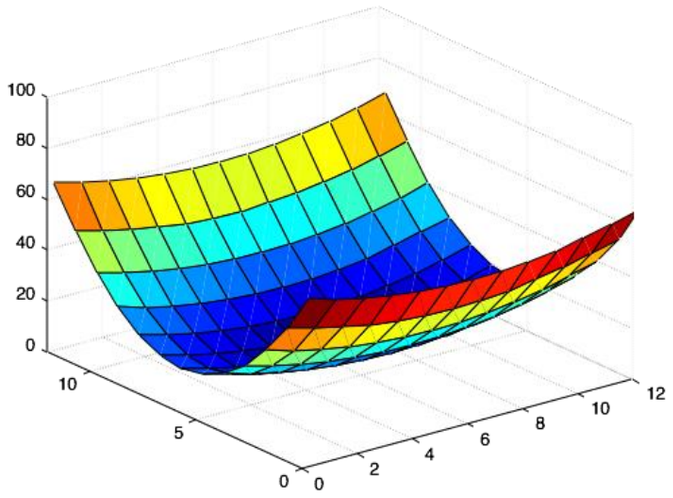




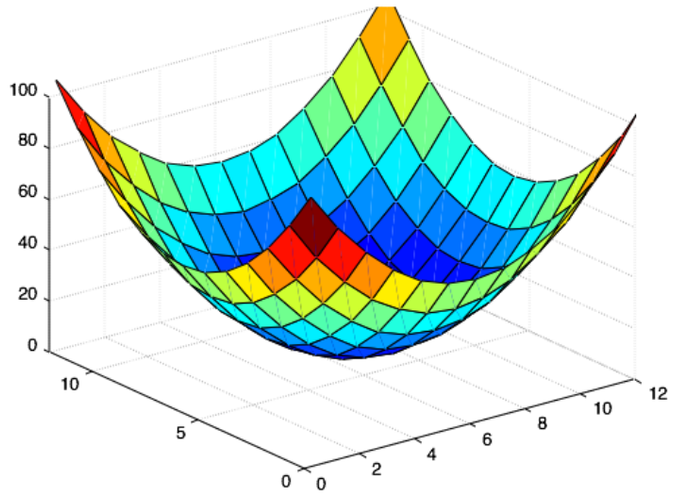
Which error surface indicates a good image feature?



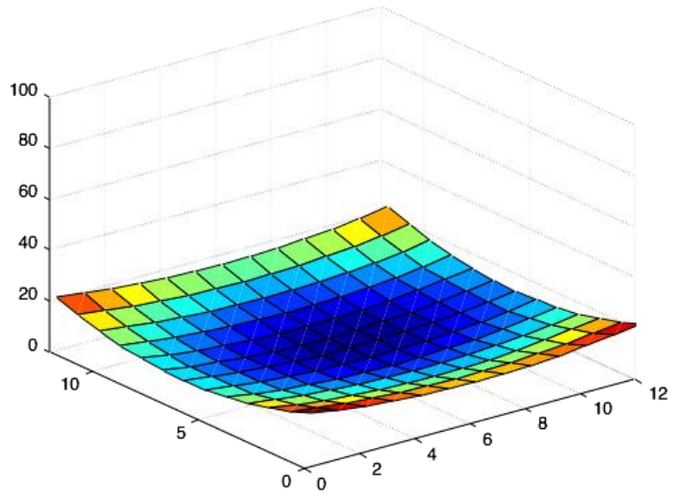
flat



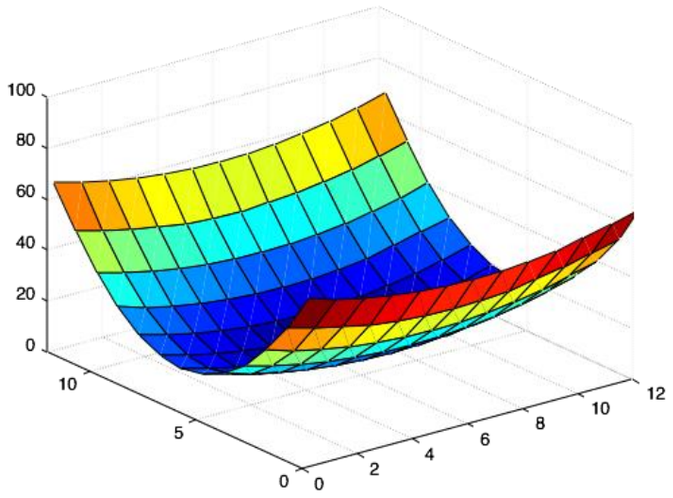
edge
‘line’



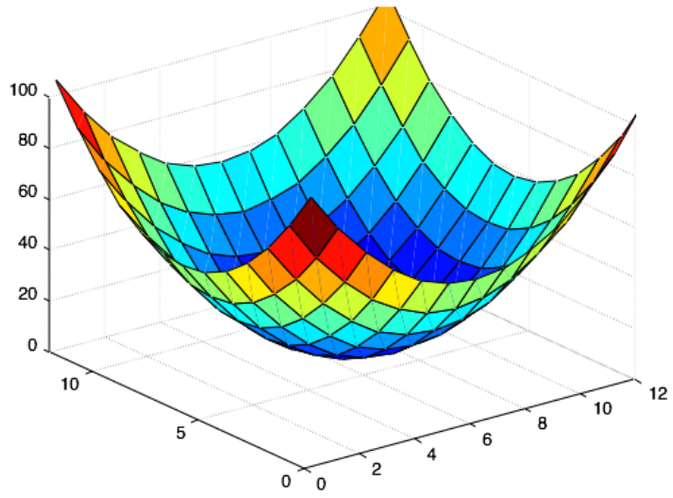
Which error surface indicates a good image feature?



flat



edge
‘line’



corner
‘dot’

4. Compute eigenvalues and eigenvectors

4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\boldsymbol{e} = \lambda \boldsymbol{e}$$

eigenvector

$$(M - \lambda I)\boldsymbol{e} = 0$$

4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\boldsymbol{e} = \lambda \boldsymbol{e}$$

eigenvector

$$(M - \lambda I)\boldsymbol{e} = 0$$

1. Compute the determinant of $M - \lambda I$

(returns a polynomial)

4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\boldsymbol{e} = \lambda \boldsymbol{e}$$

eigenvector

$$(M - \lambda I)\boldsymbol{e} = 0$$

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

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

4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\boldsymbol{e} = \lambda \boldsymbol{e}$$

eigenvector

$$(M - \lambda I)\boldsymbol{e} = 0$$

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 $(M - \lambda I)\boldsymbol{e} = 0$
(returns eigenvectors)

eig(M)

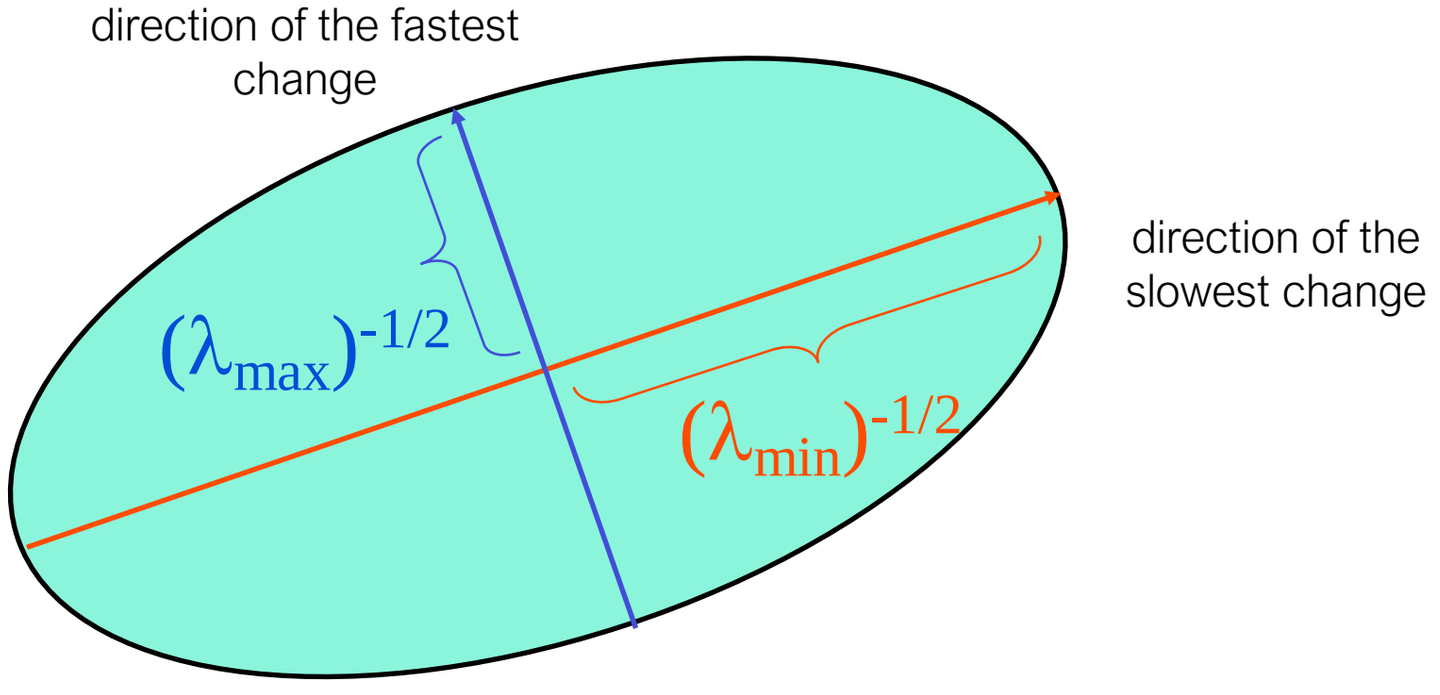
Visualization as an ellipse

Since M is symmetric, we have $M = R^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}R$

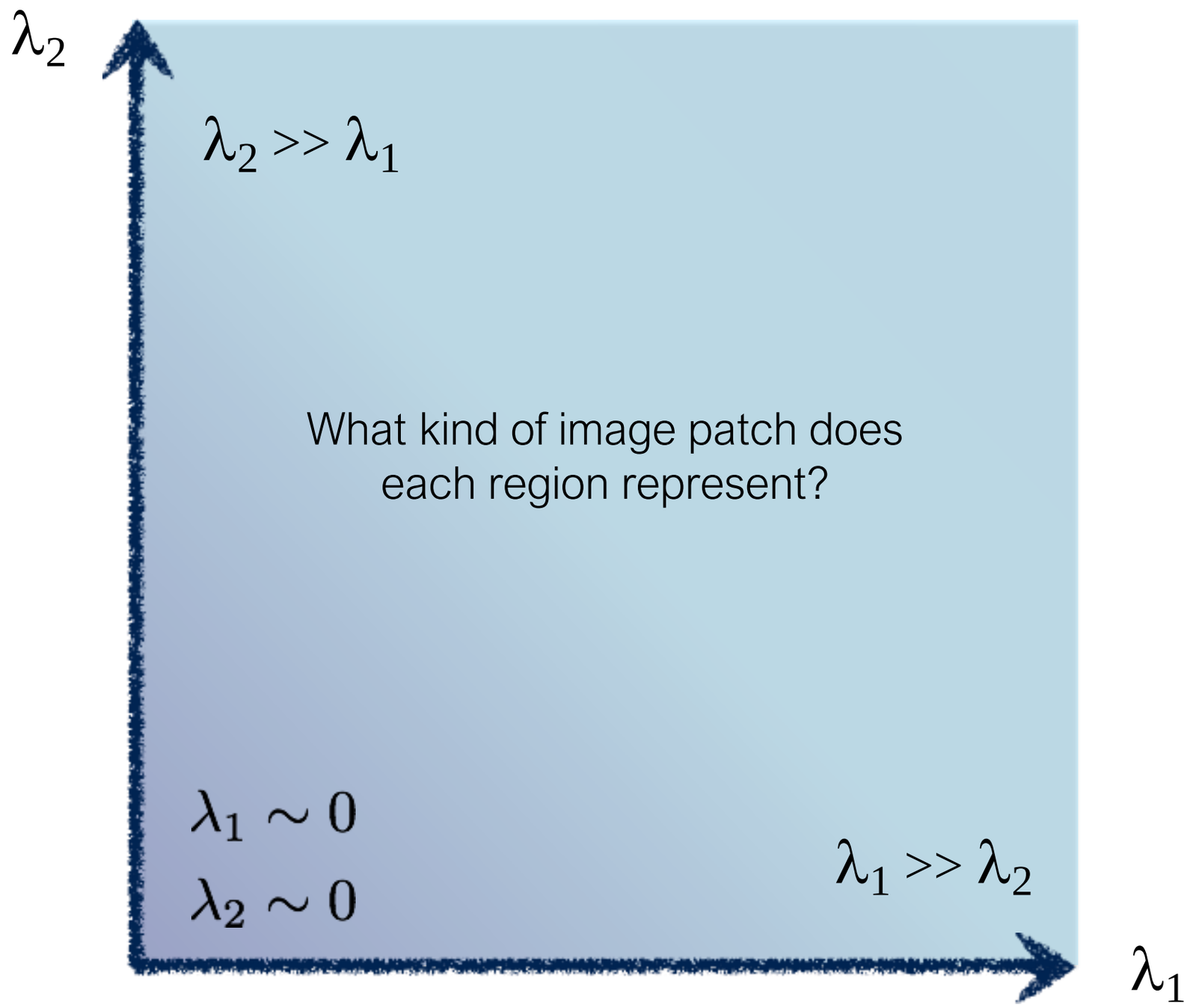
We can visualize M as an ellipse with axis lengths determined by the eigenvalues and orientation determined by R

Ellipse equation:

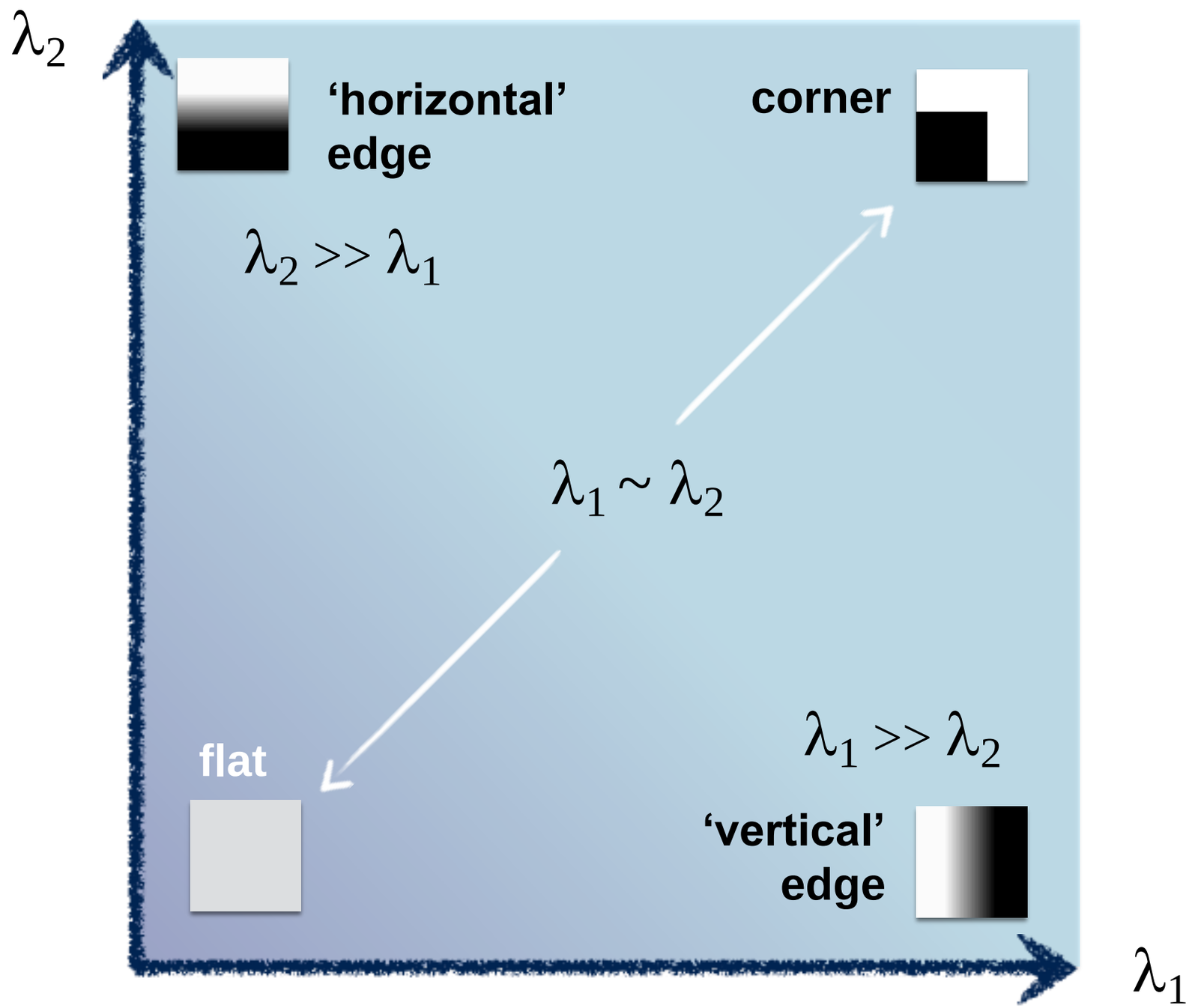
$$[u \quad v]\ M \begin{bmatrix}u\\ v\end{bmatrix} = \text{const}$$



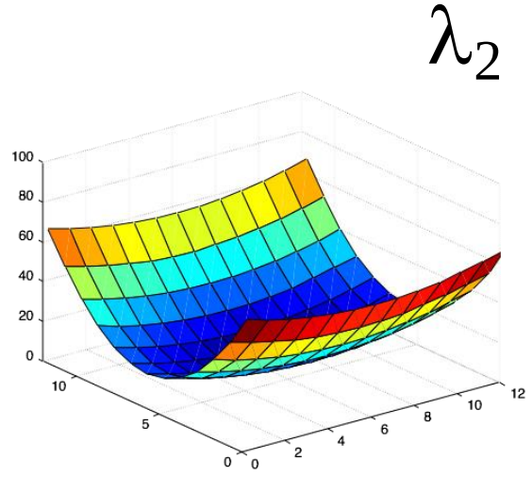
interpreting eigenvalues

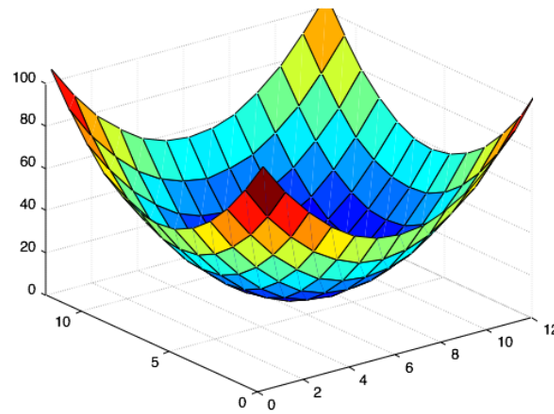


interpreting eigenvalues

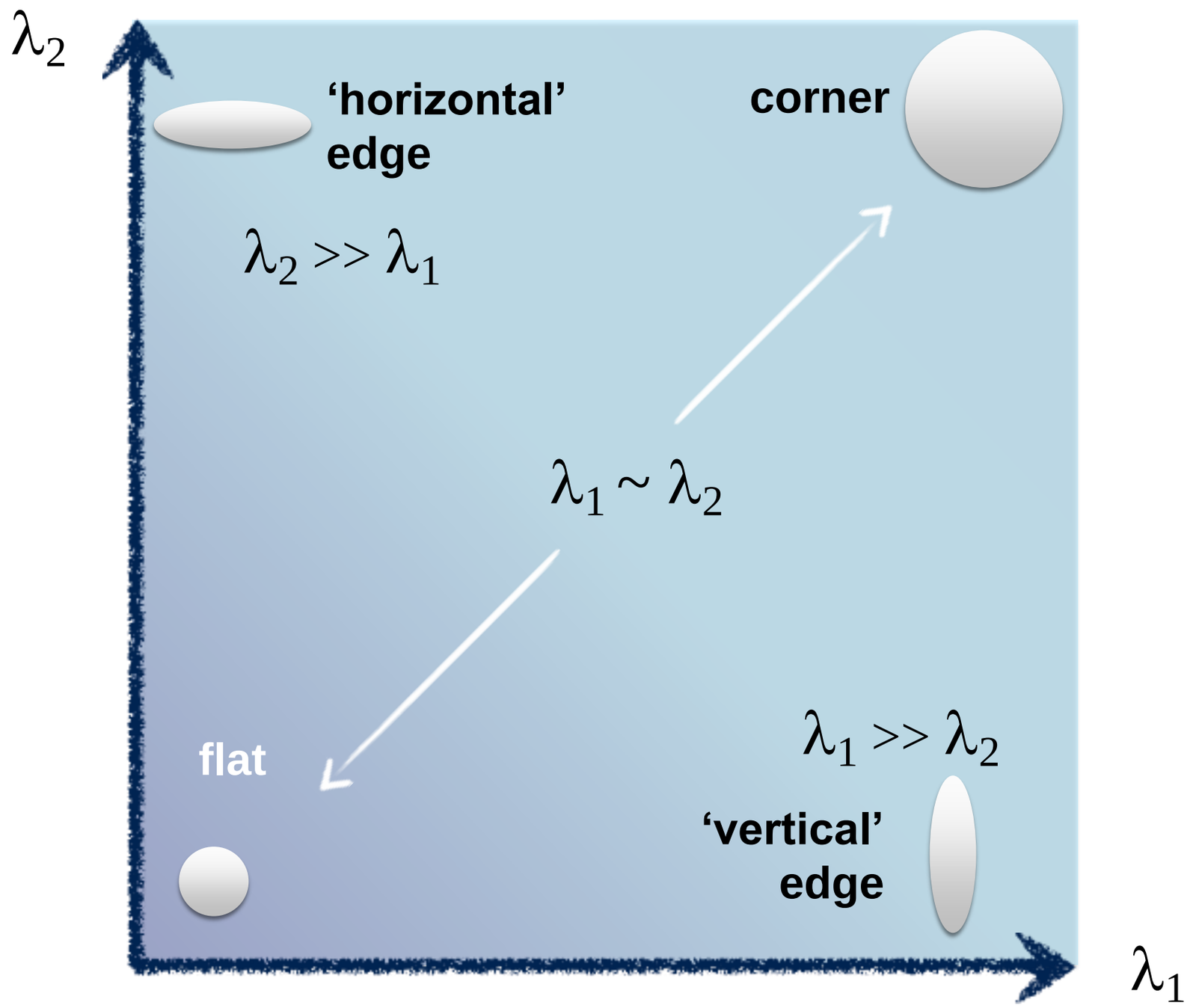


interpreting eigenvalues



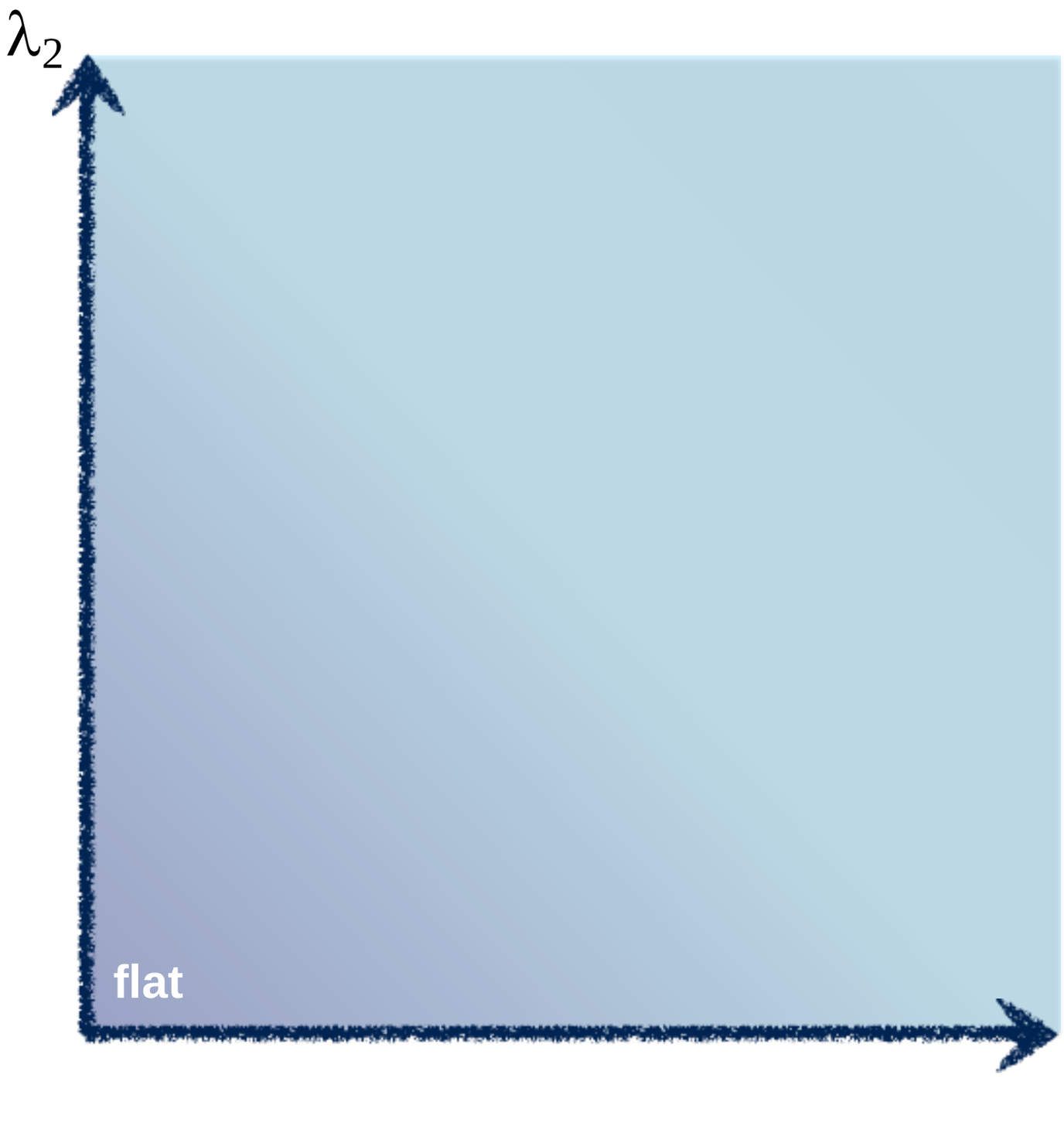


interpreting eigenvalues



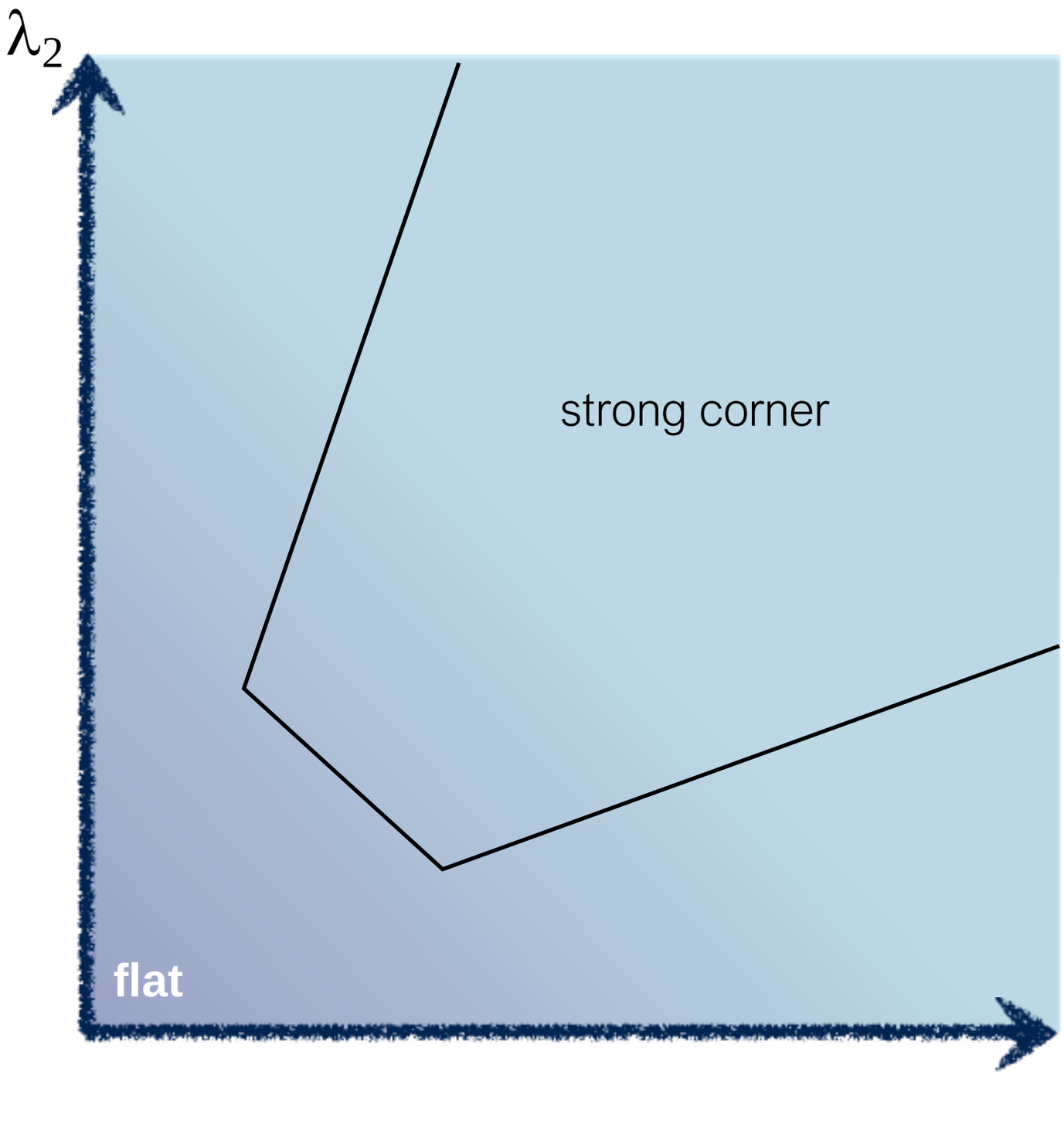
5. Use threshold on eigenvalues to detect corners

5. Use threshold on eigenvalues to detect corners



Think of a function to score ‘cornerness’

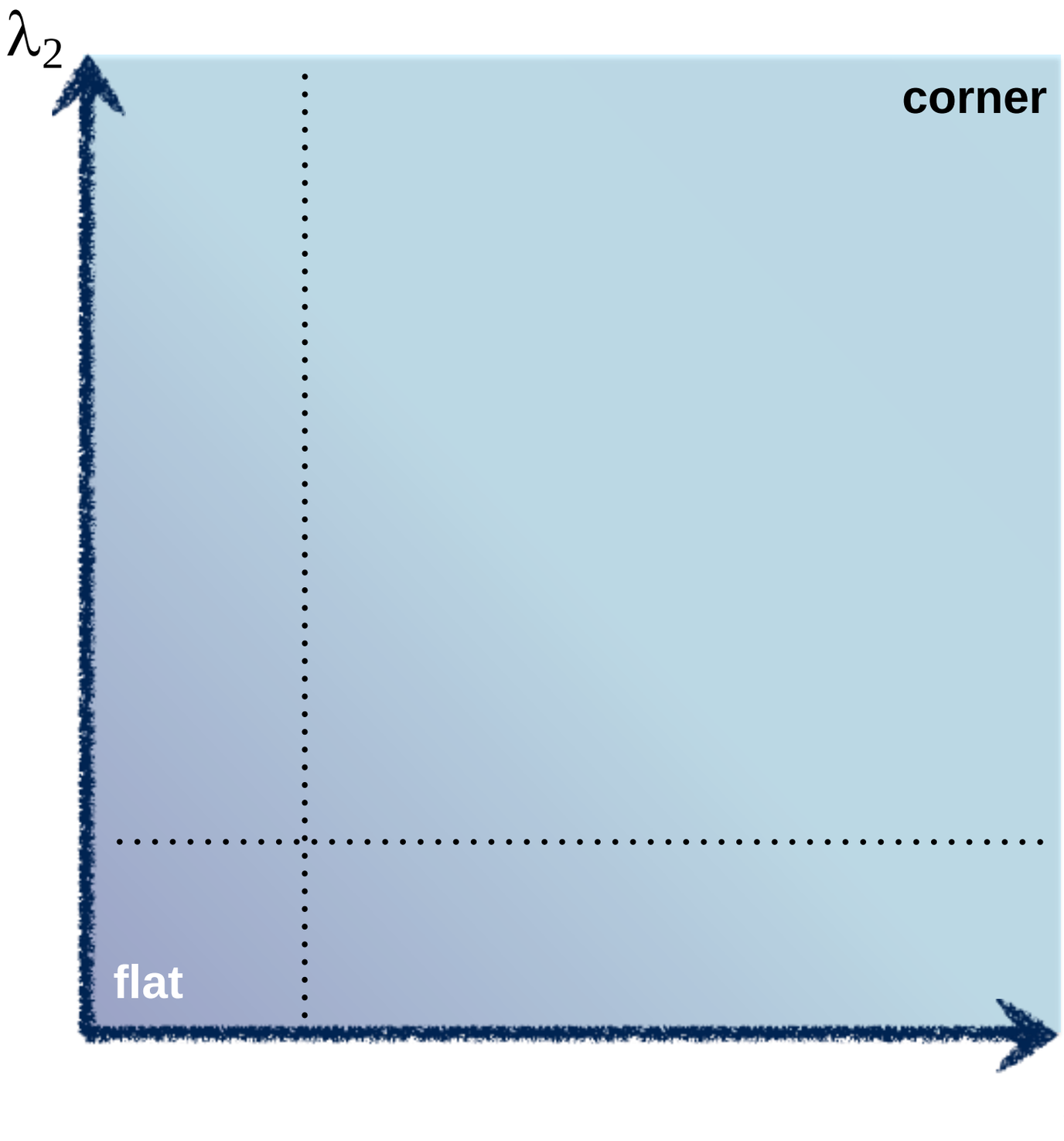
5. Use threshold on eigenvalues to detect corners



Think of a function to score ‘cornerness’

5. Use threshold on eigenvalues to detect corners

(a function of $\hat{}$)

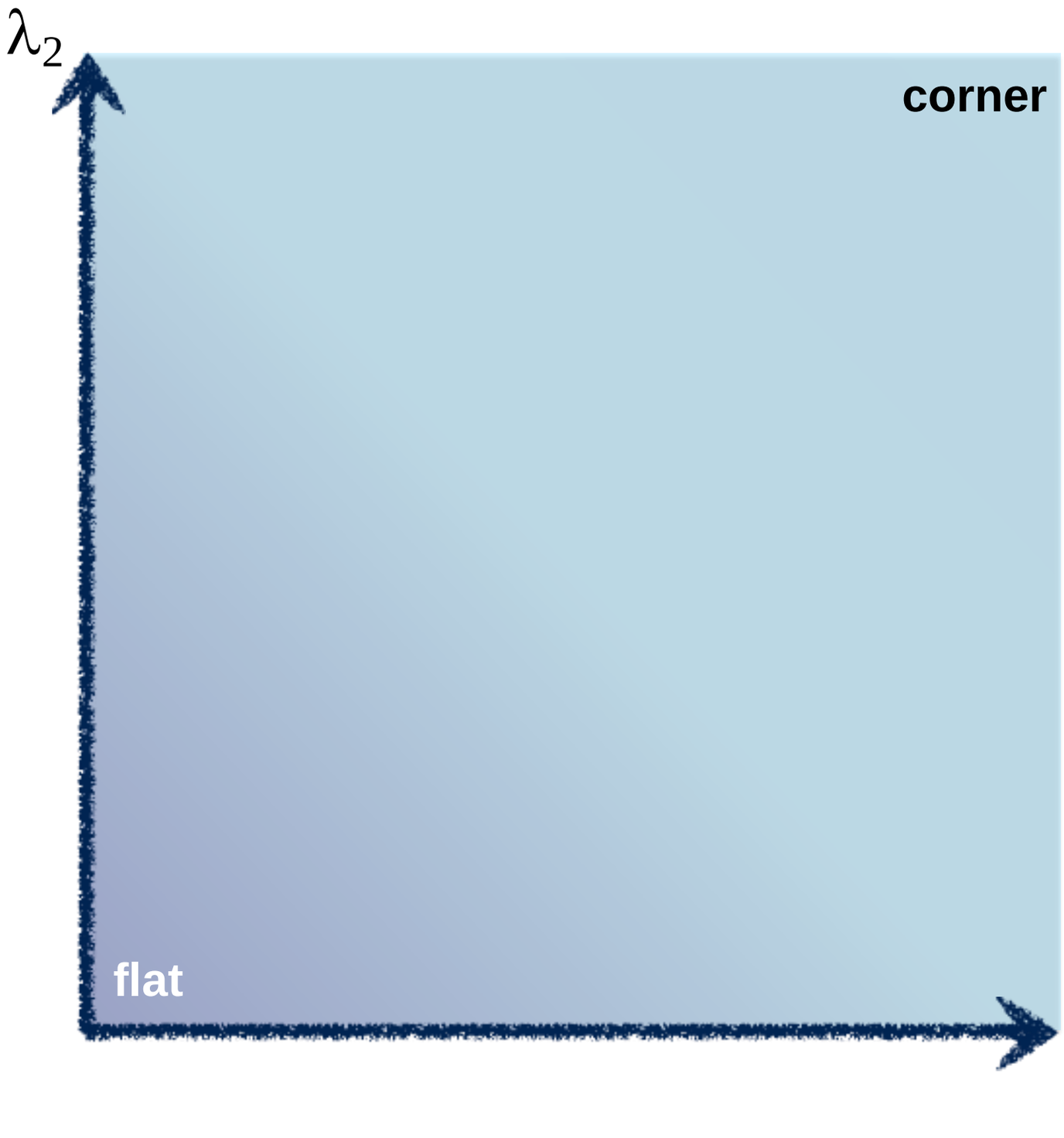


Use the smallest eigenvalue as the response function

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

5. Use threshold on eigenvalues to detect corners

(a function of $\hat{}$)



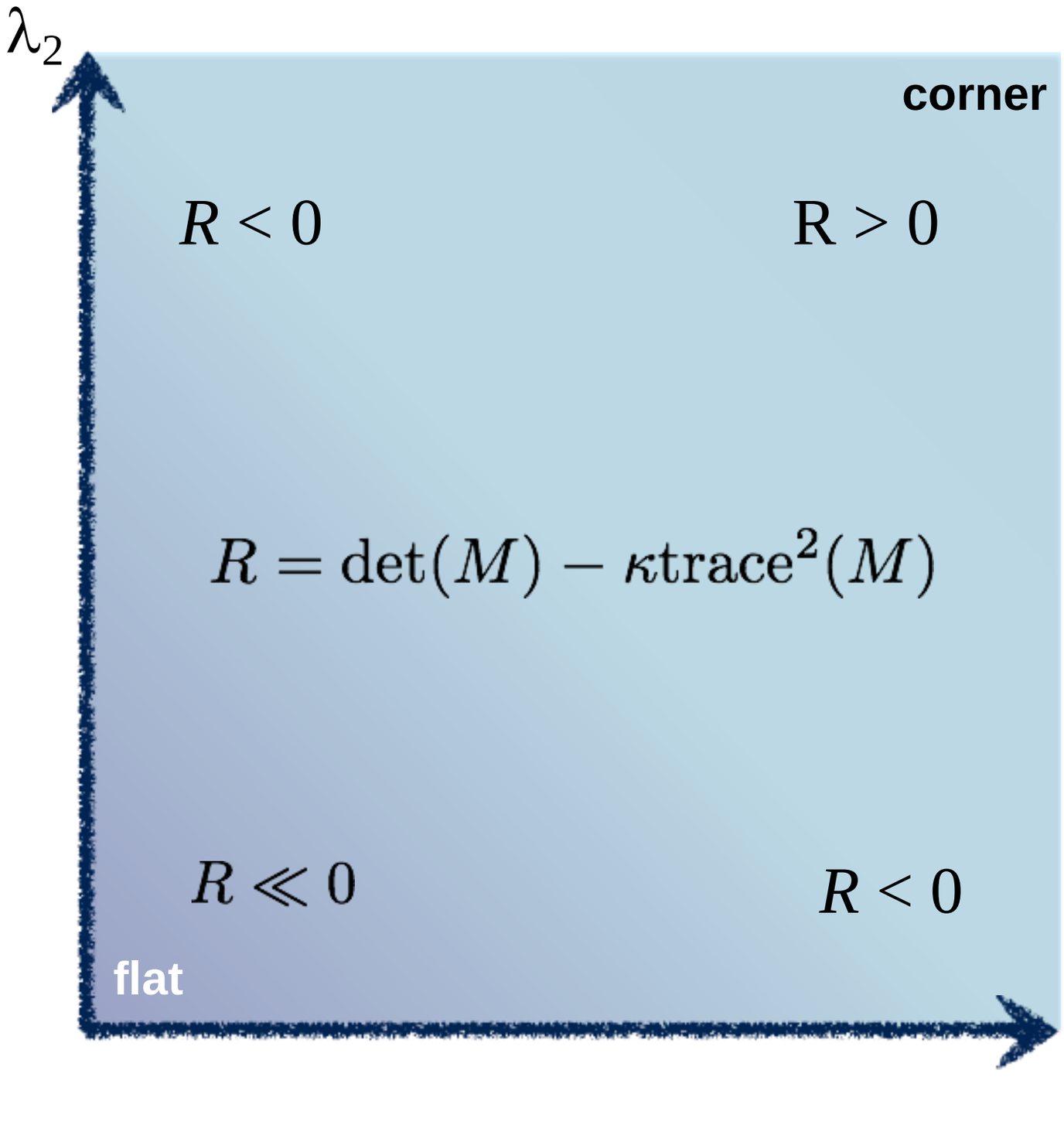
Eigenvalues need to be bigger than one.

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

Can compute this more efficiently…

5. Use threshold on eigenvalues to detect corners

(a function of $\hat{}$)



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

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

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

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

Harris & Stephens (1988)

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

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.

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \qquad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x^2} = I_x \cdot I_x \qquad I_{y^2} = I_y \cdot I_y \qquad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x^2} = G_{\sigma'} * I_{x^2} \qquad S_{y^2} = G_{\sigma'} * I_{y^2} \qquad S_{xy} = G_{\sigma'} * I_{xy}$$

Harris Detector

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

4. Define the matrix at each pixel

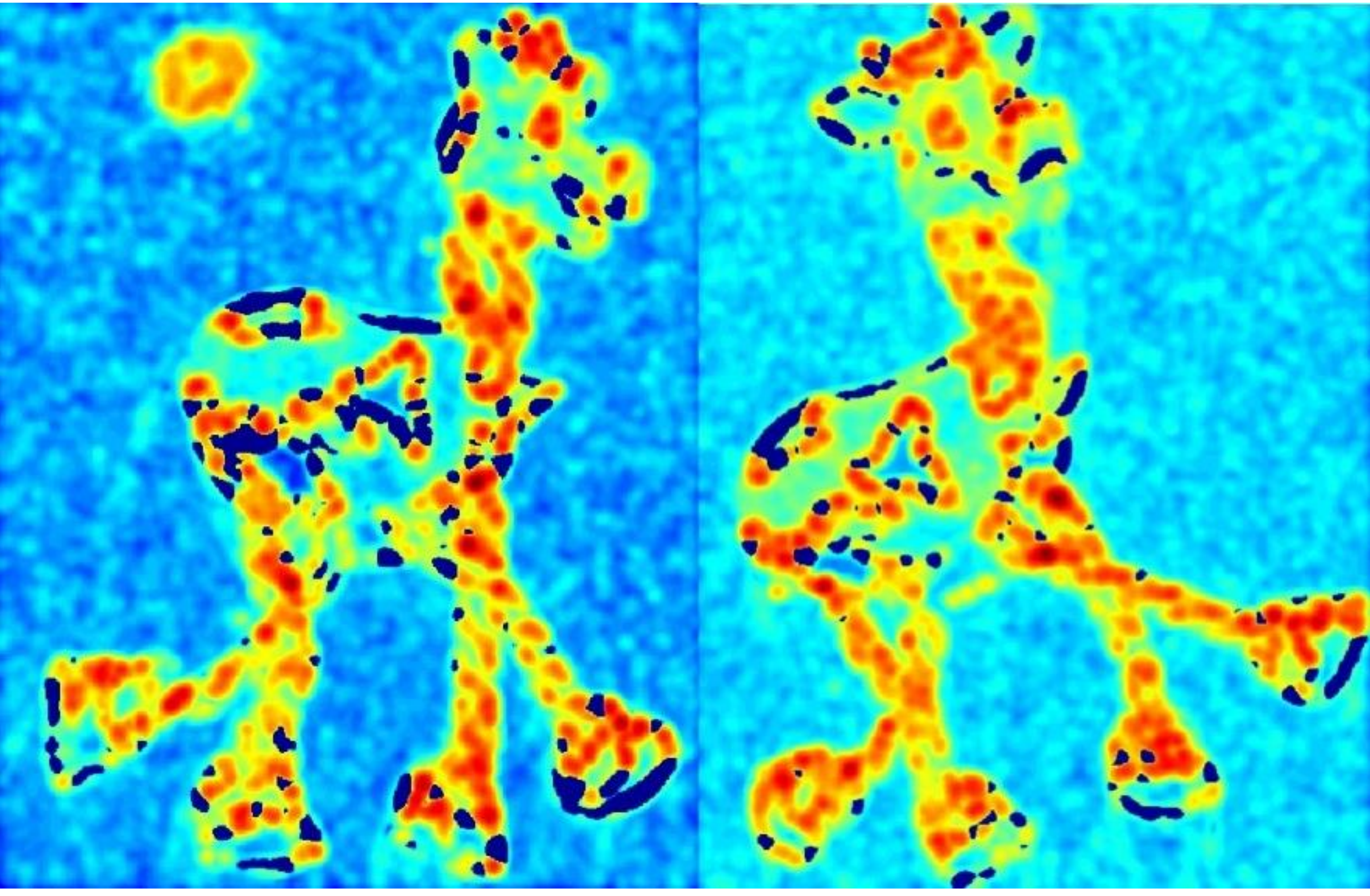
$$M(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

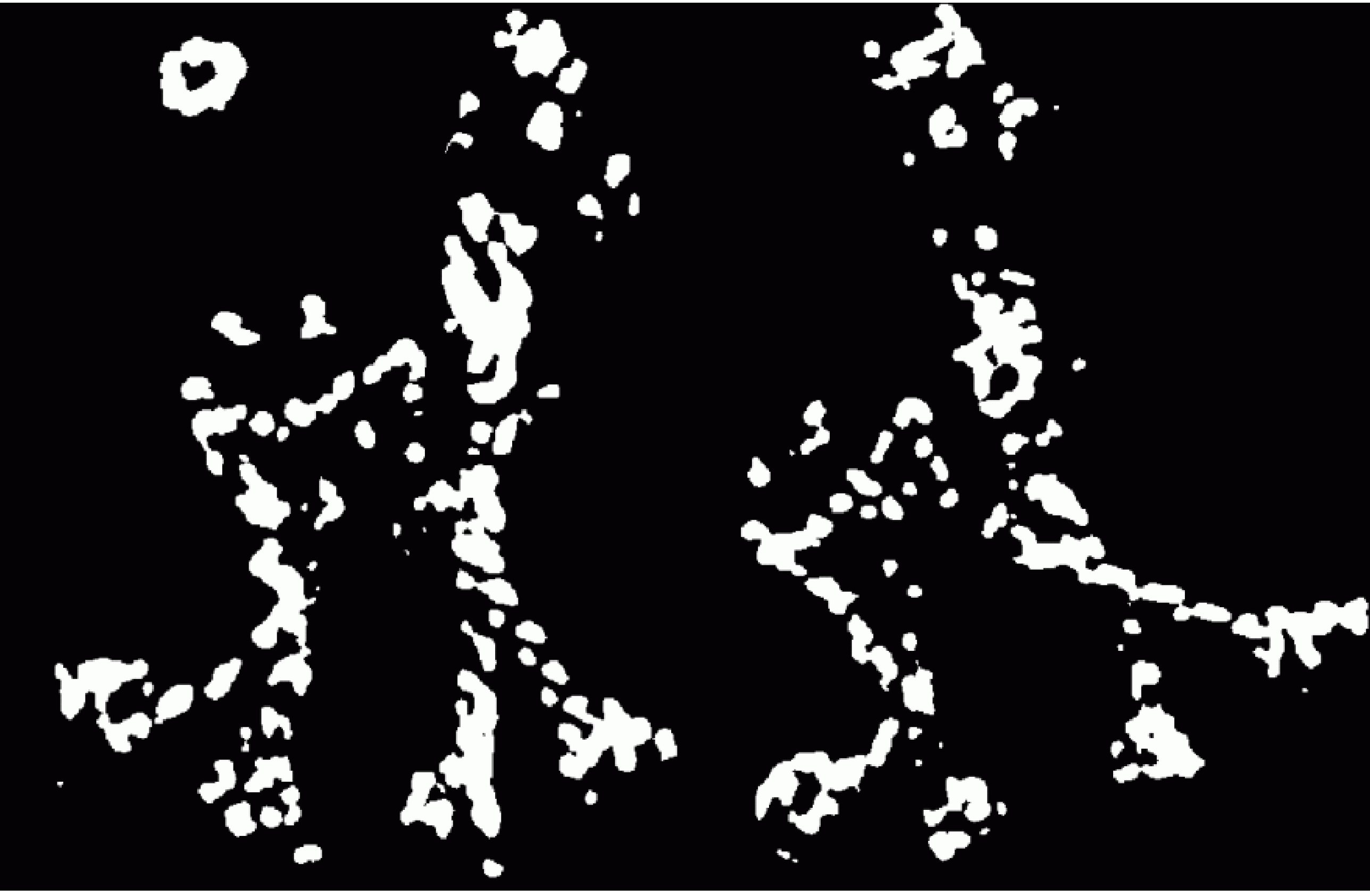
$$R = \det M - k\left(\text{trace}M\right)^2$$

6. Threshold on value of R; compute non-max suppression.

Corner response

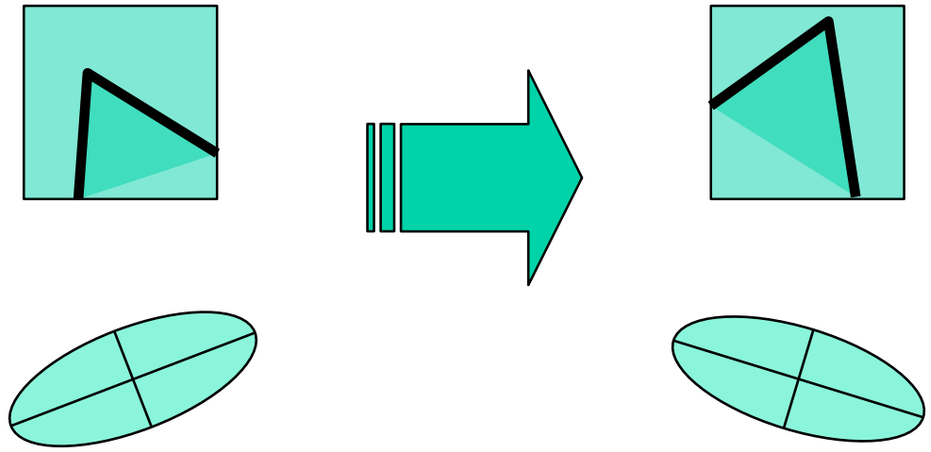


Thresholded corner response



Non-maximal suppression

Harris corner response is invariant to rotation



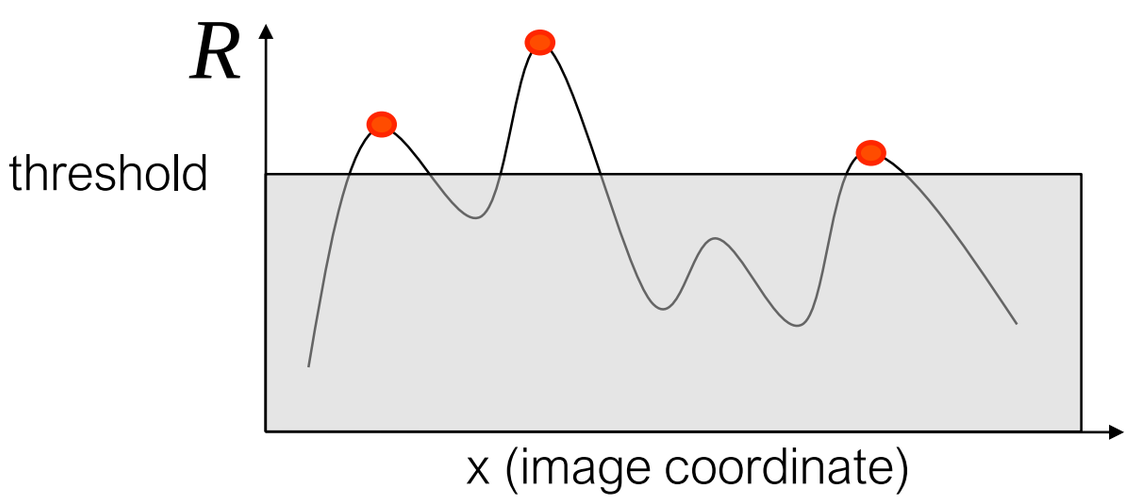
Ellipse rotates but its shape
(**eigenvalues**) remains the same

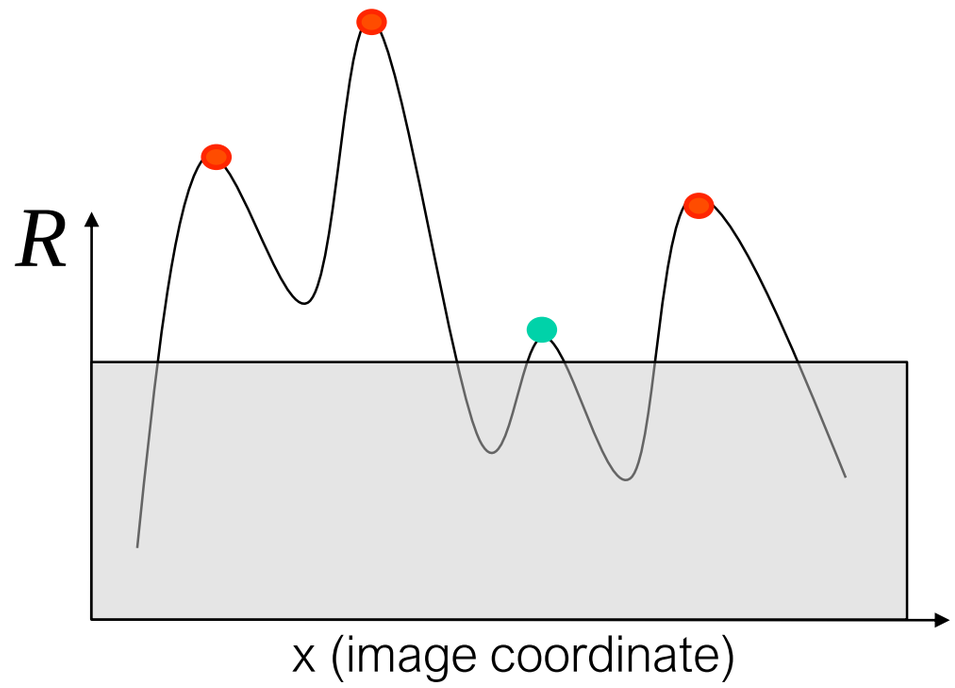
Corner response R is invariant to image rotation

Harris corner response is invariant to intensity changes

Partial invariance to *affine intensity* change

- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scale: $I \rightarrow a\, I$

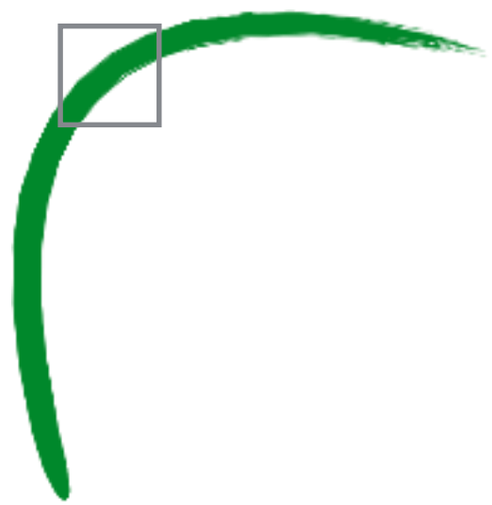




The Harris detector is not invariant to changes in …

The Harris corner detector is <u>not</u> invariant to scale

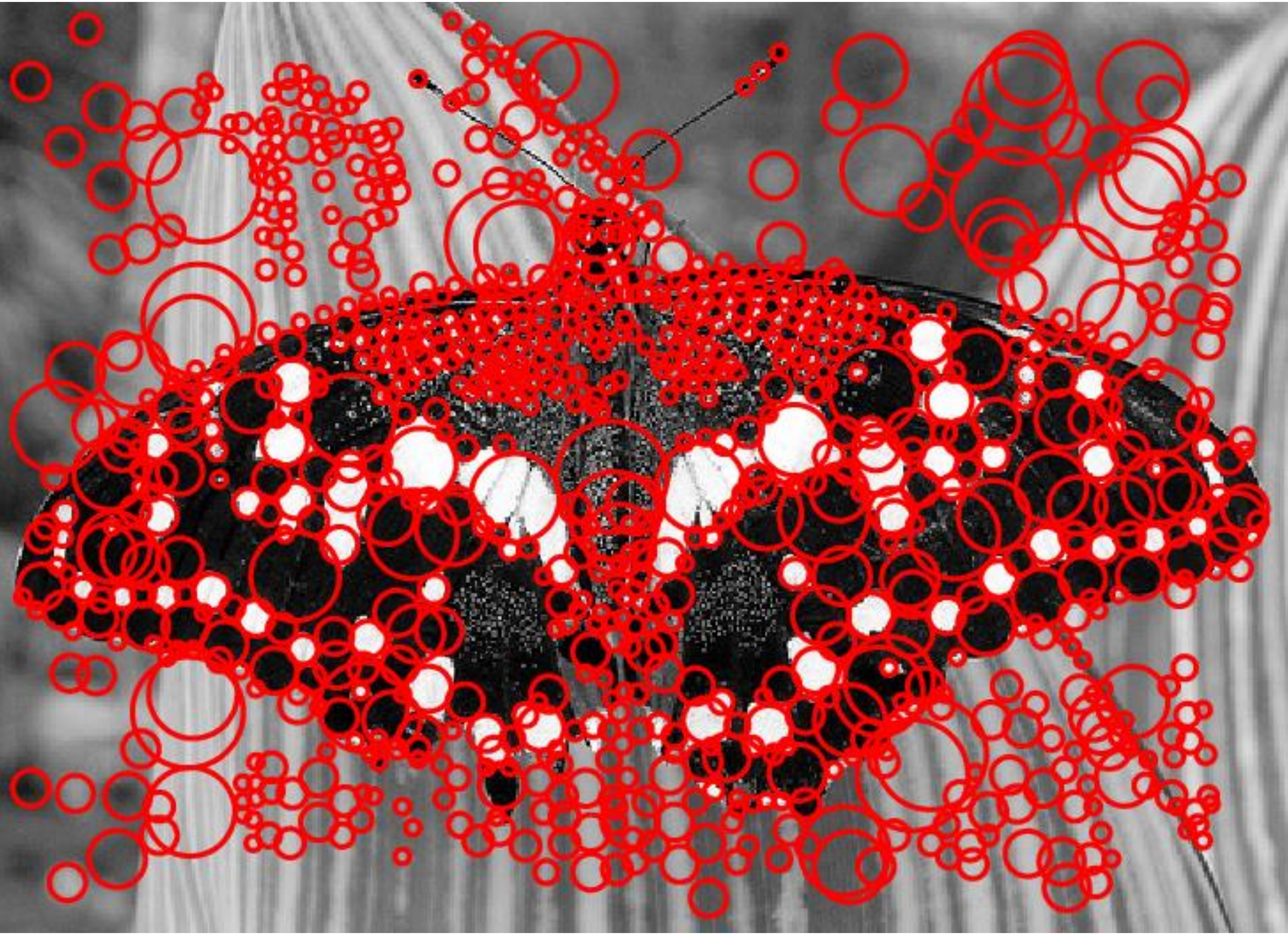
edge!



corner!

План

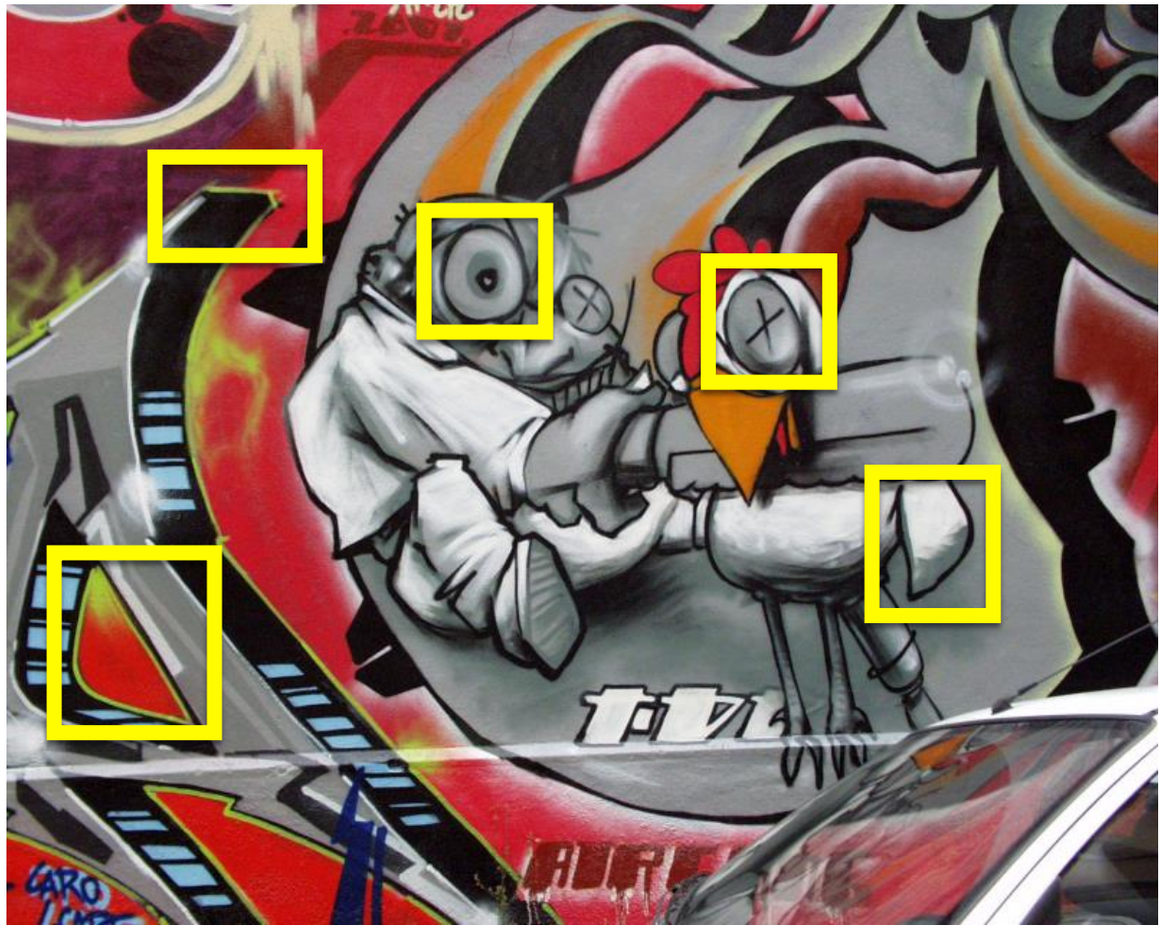
- Ключевые точки (Corners)
- Дескрипторы (Descriptors)
- 2d преобразования (Warping)
- Гомография (Homography)

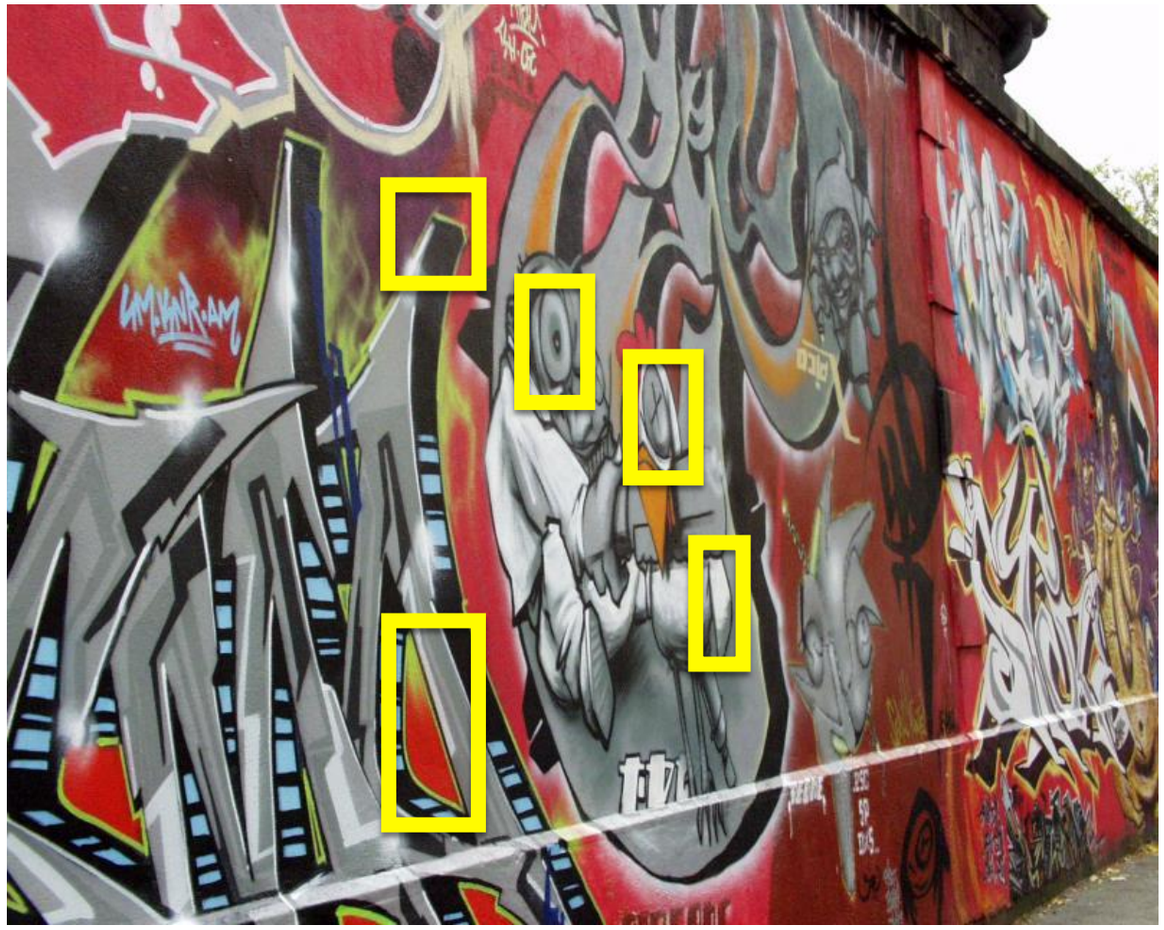


План

- Ключевые точки (Corners)
- Дескрипторы (Descriptors)
- 2d преобразования (Warping)
- Гомография (Homography)

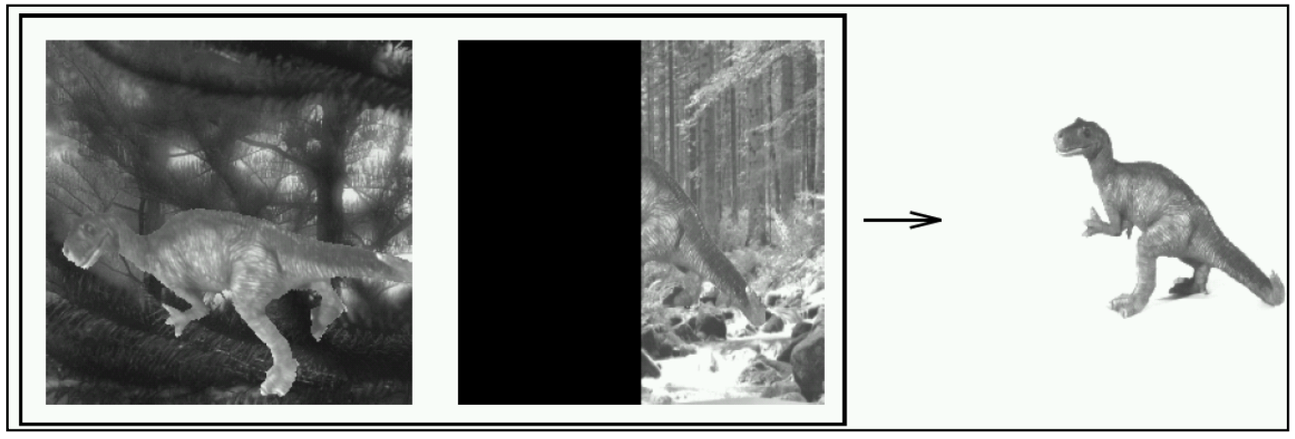
Why do we need feature descriptors?





If we know where the good features are,
how do we match them?

Object instance recognition



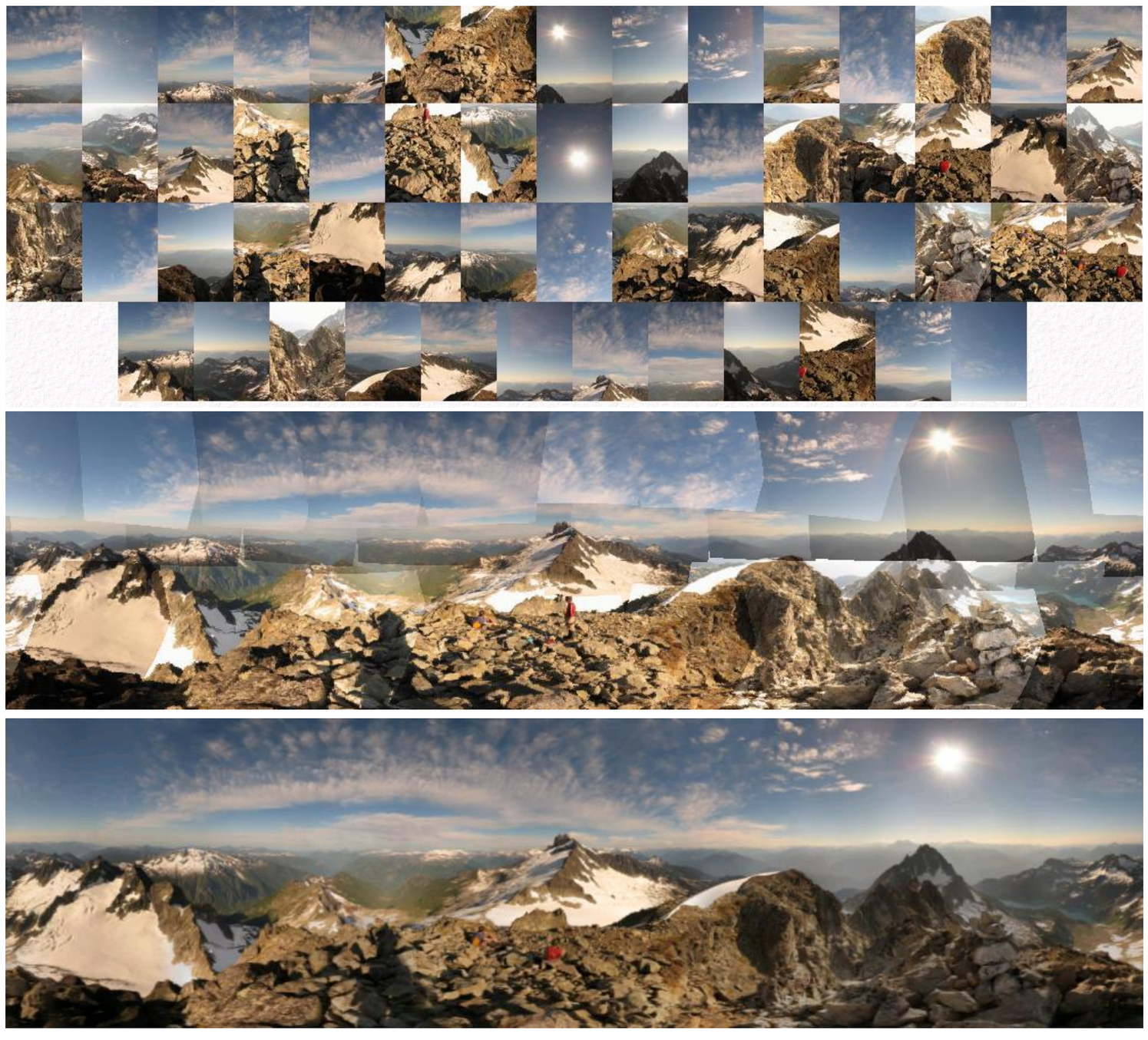
Schmid and Mohr 1997

Sivic and Zisserman, 2003

Rothganger et al. 2003

Lowe 2002

Image mosaicing



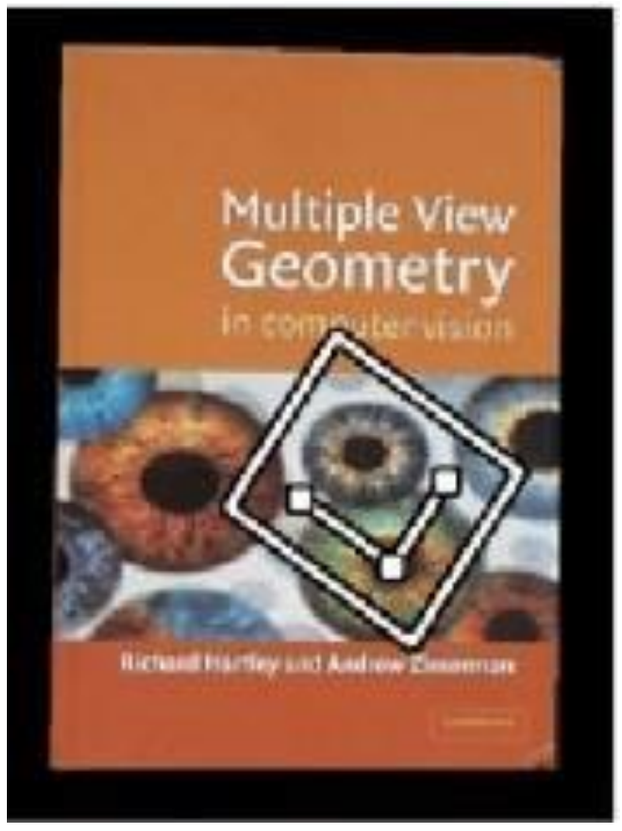
How do we describe an image patch?

Patches with similar content should have similar descriptors.

Designing feature descriptors

Photometric transformations

Geometric transformations



objects will appear at different scales, translation and rotation

What is the best descriptor for an image feature?

Image patch

Just use the pixel values of the patch

Perfectly fine if geometry and appearance is unchanged

(a.k.a. template matching)

Tiny Images

Just down-sample it!
Simple, fast, robust to small affine transforms.

Image patch

Just use the pixel values of the patch

Perfectly fine if geometry and appearance is unchanged

(a.k.a. template matching)

What are the problems?

Image patch

Just use the pixel values of the patch

Perfectly fine if geometry and appearance is unchanged

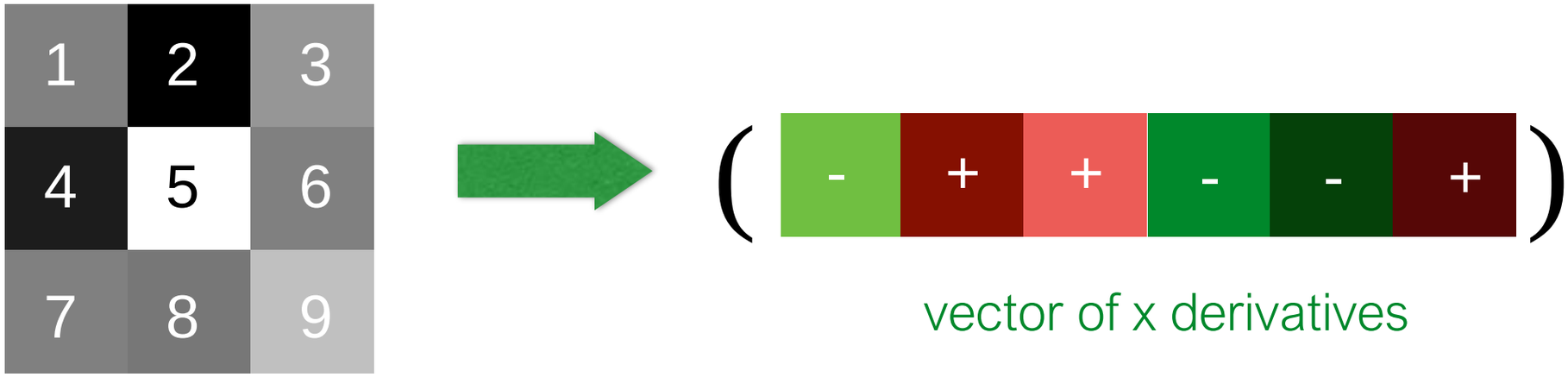
(a.k.a. template matching)

What are the problems?

How can you be less sensitive to absolute intensity values?

Image gradients

Use pixel differences



vector of x derivatives

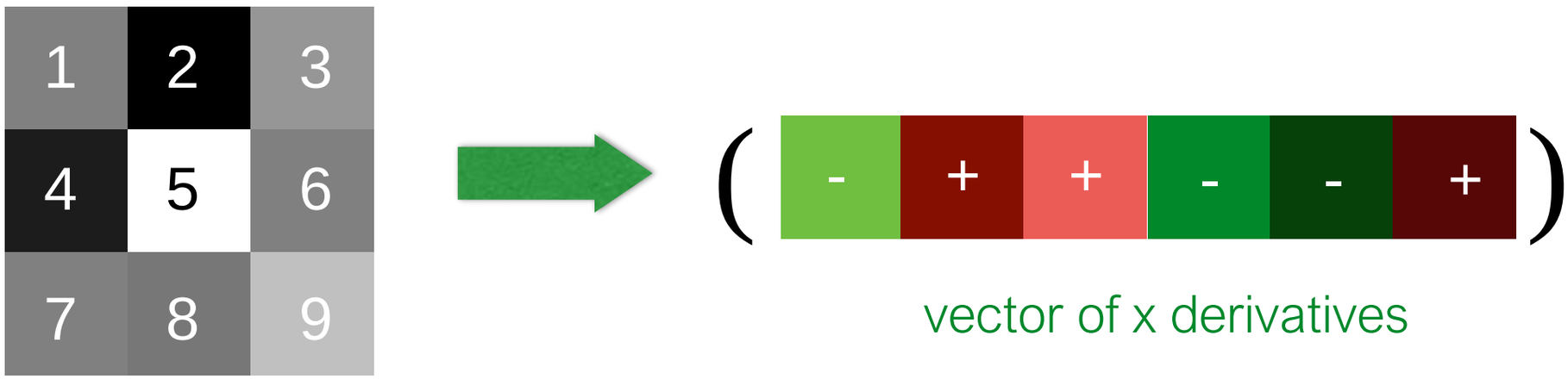
‘binary descriptor’

Feature is invariant to absolute intensity values

What are the problems?

Image gradients

Use pixel differences



vector of x derivatives

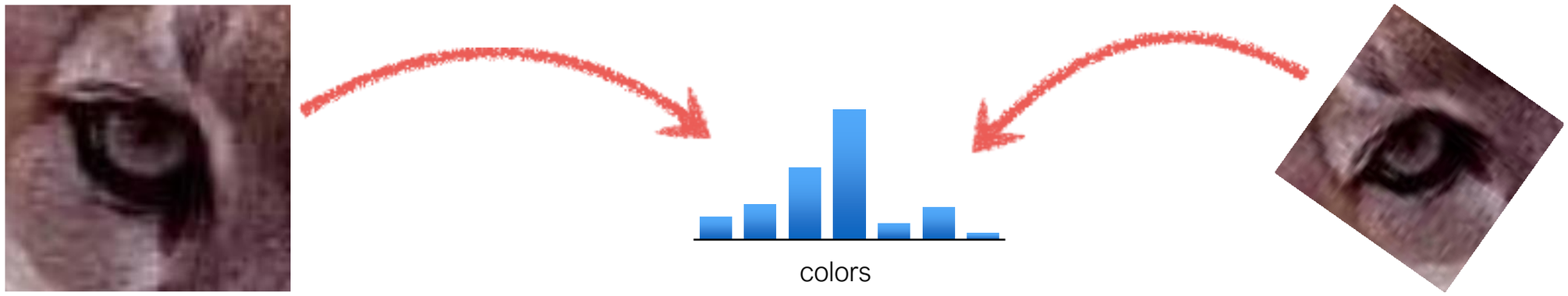
Feature is invariant to absolute intensity values

What are the problems?

How can you be less sensitive to deformations?

Color histogram

Count the colors in the image using a histogram

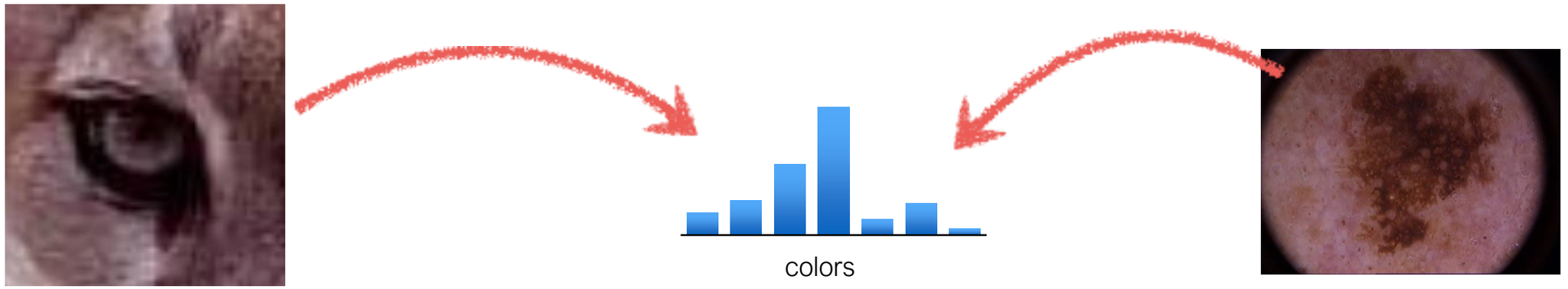


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram

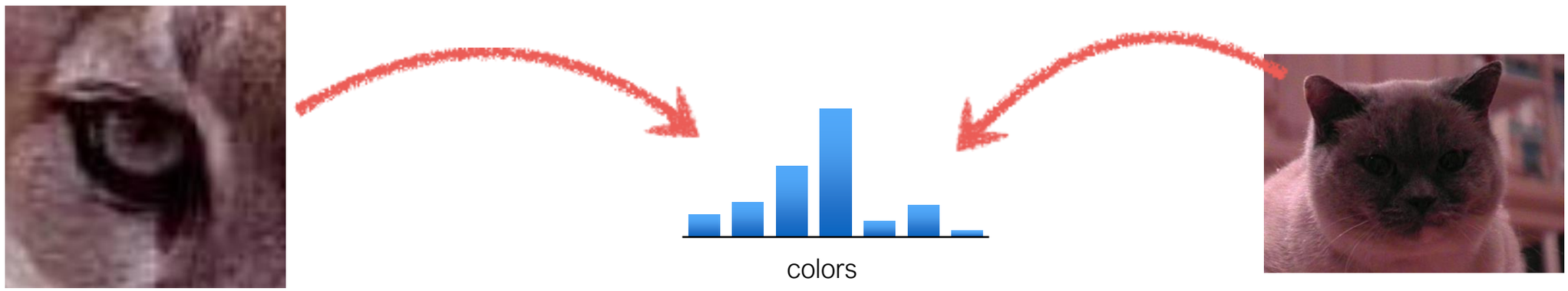


Invariant to changes in scale and rotation

What are the problems?

Color histogram

Count the colors in the image using a histogram



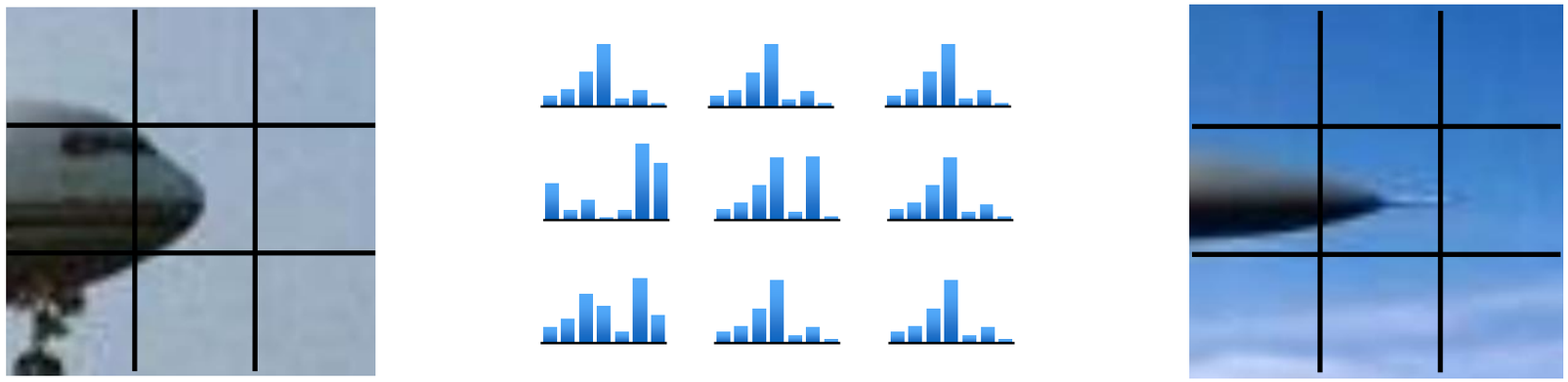
Invariant to changes in scale and rotation

What are the problems?

How can you be more sensitive to spatial layout?

Spatial histograms

Compute histograms over spatial ‘cells’



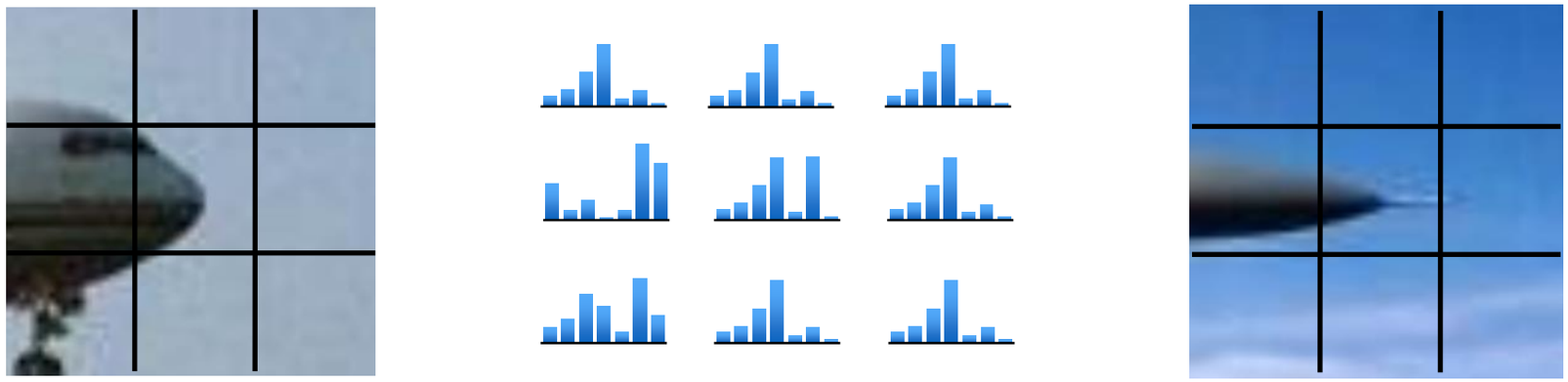
Retains rough spatial layout

Some invariance to deformations

What are the problems?

Spatial histograms

Compute histograms over spatial ‘cells’



Retains rough spatial layout

Some invariance to deformations

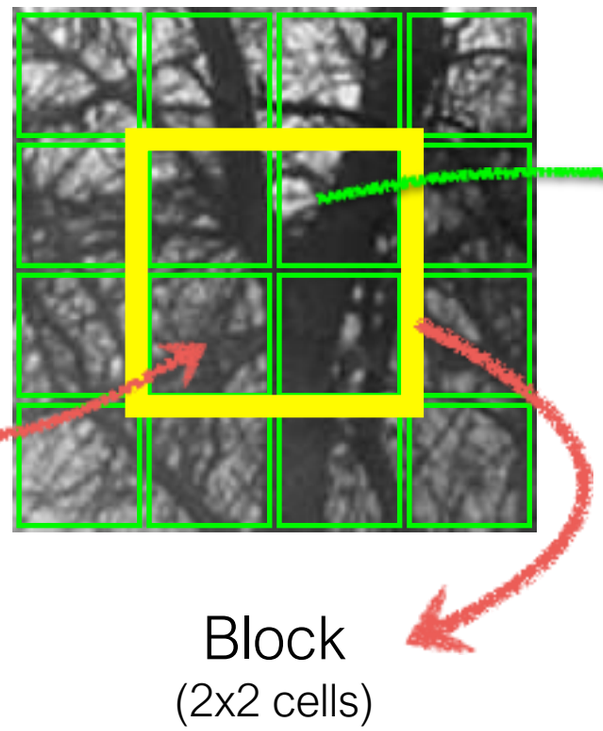
What are the problems?

How can you be completely invariant to rotation?

HOG descriptor

HOG

Dalal, Triggs. **Histograms of Oriented Gradients** for Human Detection. CVPR, 2005



Cell
(8x8 pixels)

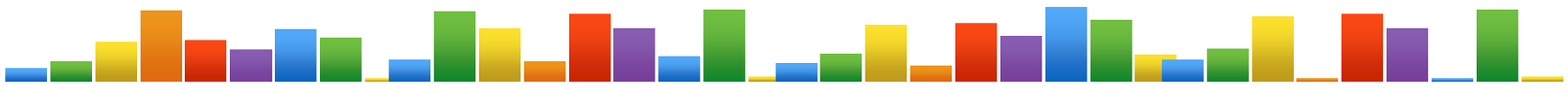
Block
(2x2 cells)

histogram of ‘unsigned’ gradients

soft binning

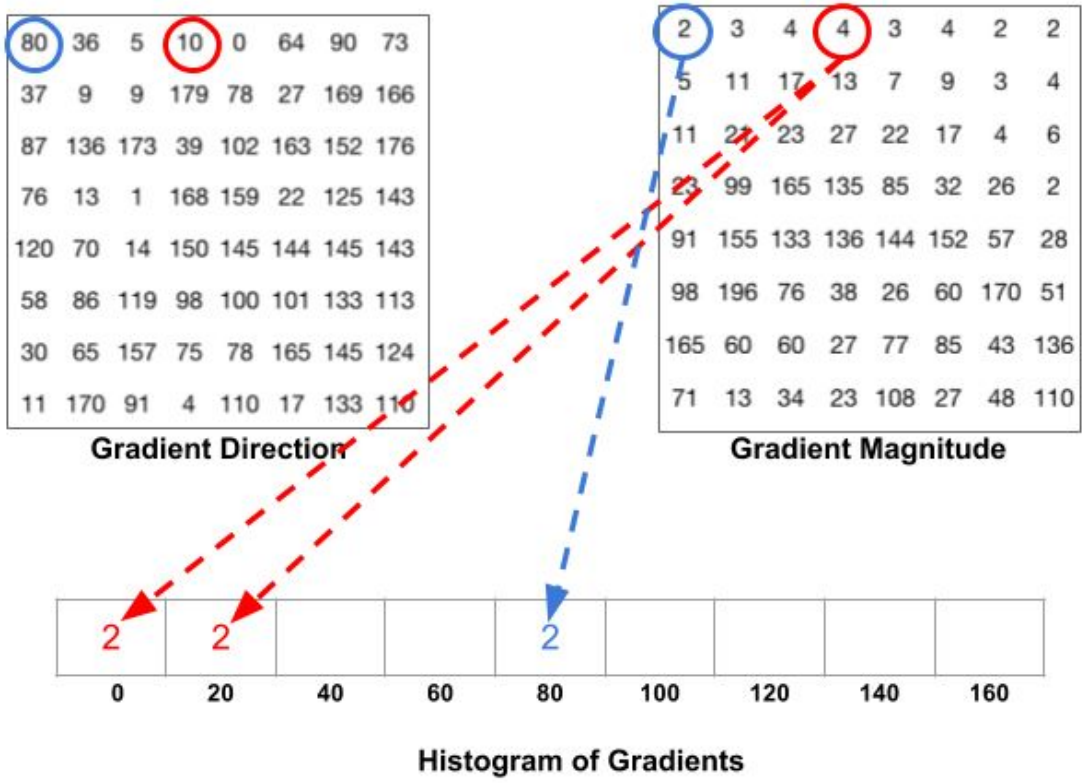
gradient magnitude histogram
(one for each cell)

Concatenate and L-2 normalization



Single scale, no dominant orientation

Гистограммы градиентов (HOG)



Pedestrian detection

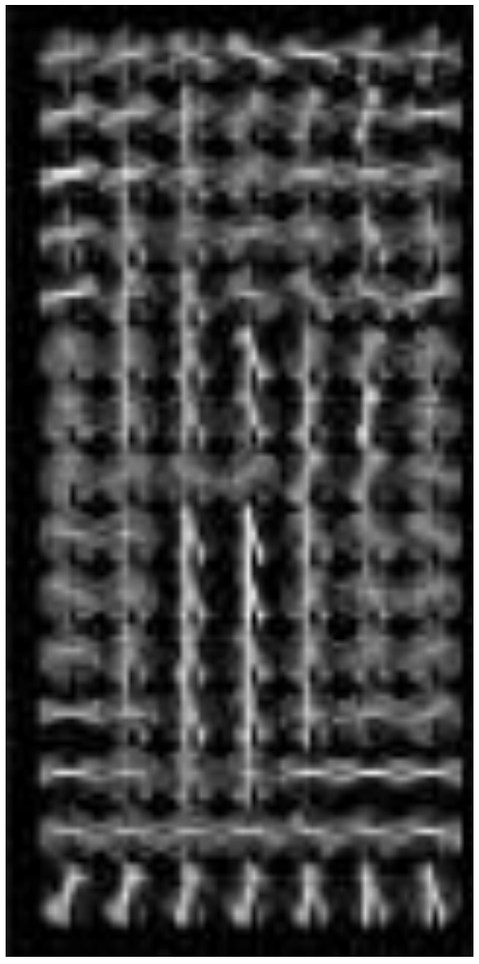
128 pixels
16 cells
15 blocks

1 cell step size

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 36 =
3780

visualization



Redundant representation due to overlapping blocks

How many times is each inner cell encoded?





http://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

SIFT

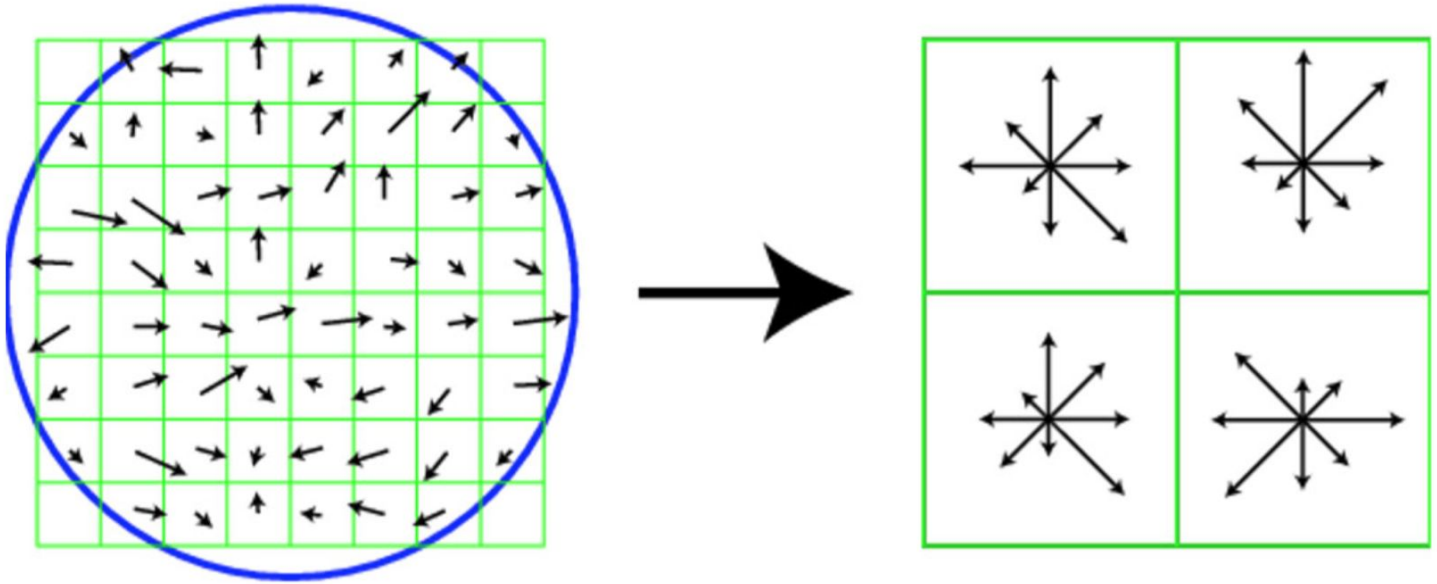
SIFT - Scale Invariant Feature Transform

- дескриптор основан на построении гистограммы градиентов (HOG)
- в окрестности характерной точки выделяется область размером 16x16 пикселей
- для каждого пикселя оценивается вектор градиента
- длина вектора градиента взвешивается гуассовским фильтром, таким образом, чтобы пиксели удаленные от характерной точки имели меньший вес

SIFT - Scale Invariant Feature Transform

- исходная область 16x16 разбивается на части размера 4x4
- для каждой части строится гистограмма градиентов с 8 ячейками
- в результате получается вектор из 128 признаков
- полученный вектор нормируется до единичной длины

SIFT - Scale Invariant Feature Transform

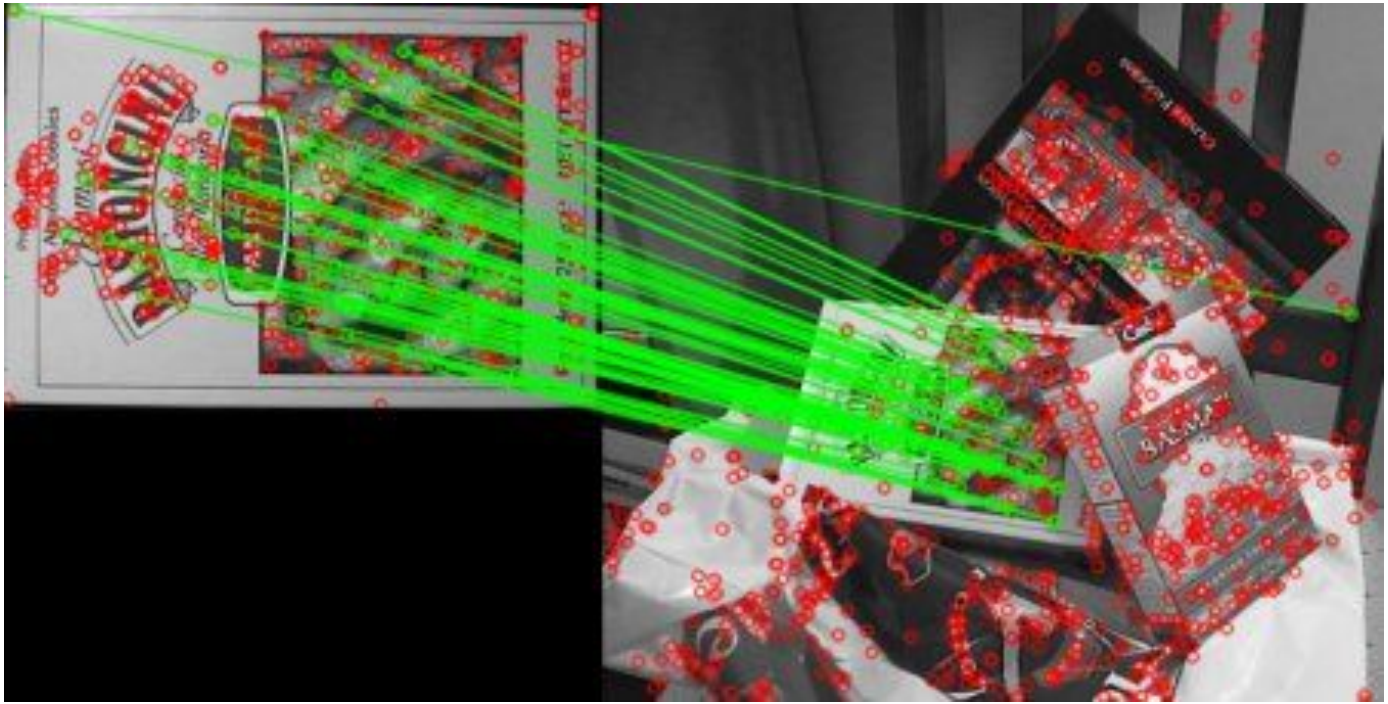


Матчинг характерных точек

Матчинг характерных точек

- выбрать меру расстояния для дескрипторов - евклидова мера (L2), L1, Hamming
- попарное сравнение всех точек - полный перебор, долго
- индексация перед поиском и поиск по индексу
  - поиск точек в окрестности - kdtree
  - хеширование точек таким образом, чтобы точки с похожими дескрипторами оказывались рядом - locality sensitive hashing

Матчинг характерных точек



План

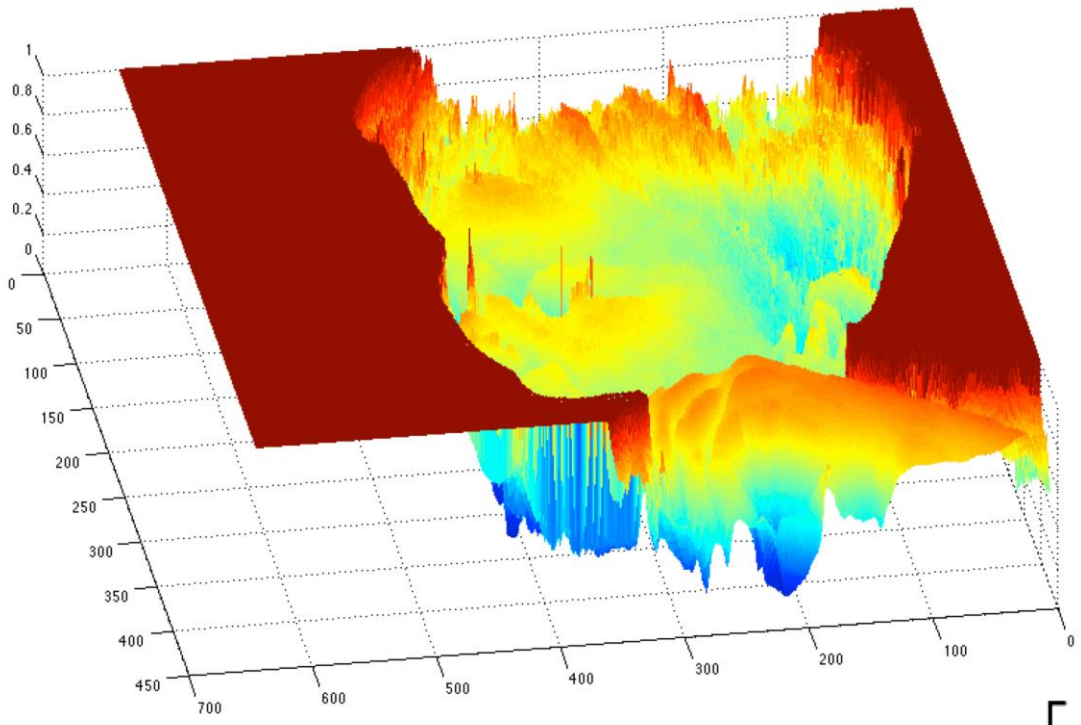
- Ключевые точки (Corners)
- Дескрипторы (Descriptors)
- 2d преобразования (Warping)
- Гомография (Homography)

What is an image?

grayscale image

What is the range of the image function f?

$f(\boldsymbol{x})$



domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

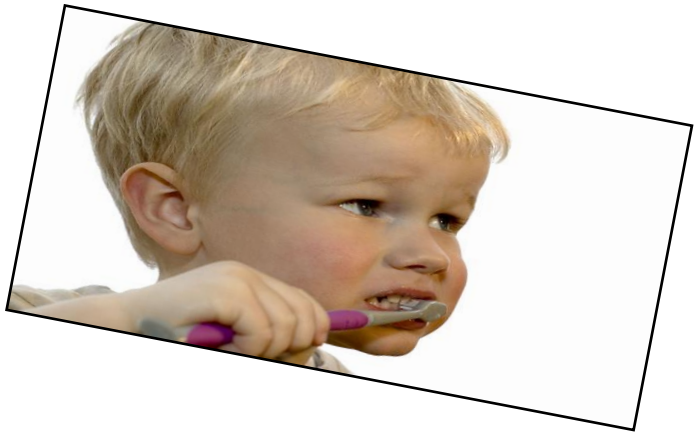
A (grayscale) image is a 2D function.

What types of image transformations can we do?

Filtering

changes pixel *values*

Warping



changes pixel *locations*

What types of image transformations can we do?

F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

G

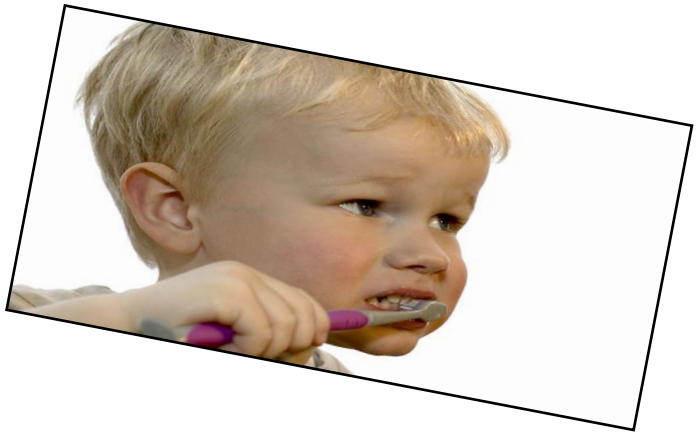
changes *range* of image function

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

G

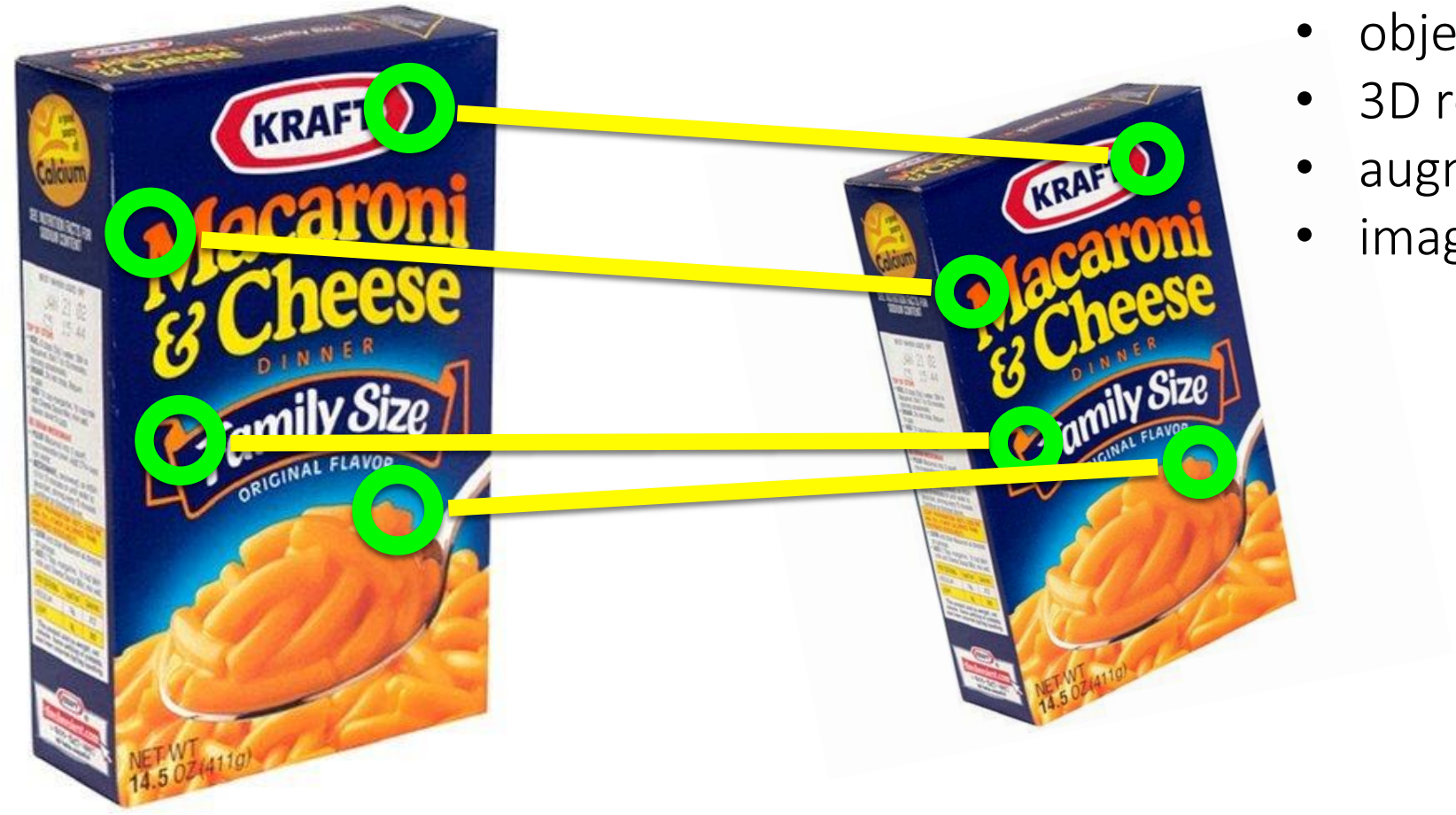


changes *domain* of image function

Warping example: feature matching

Warping example: feature matching

Warping example: feature matching



- object recognition
- 3D reconstruction
- augmented reality
- image stitching

How do you compute the transformation?

Warping example: feature matching

Given a set of matched feature points:

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

point in one image

point in the other image

and a transformation:

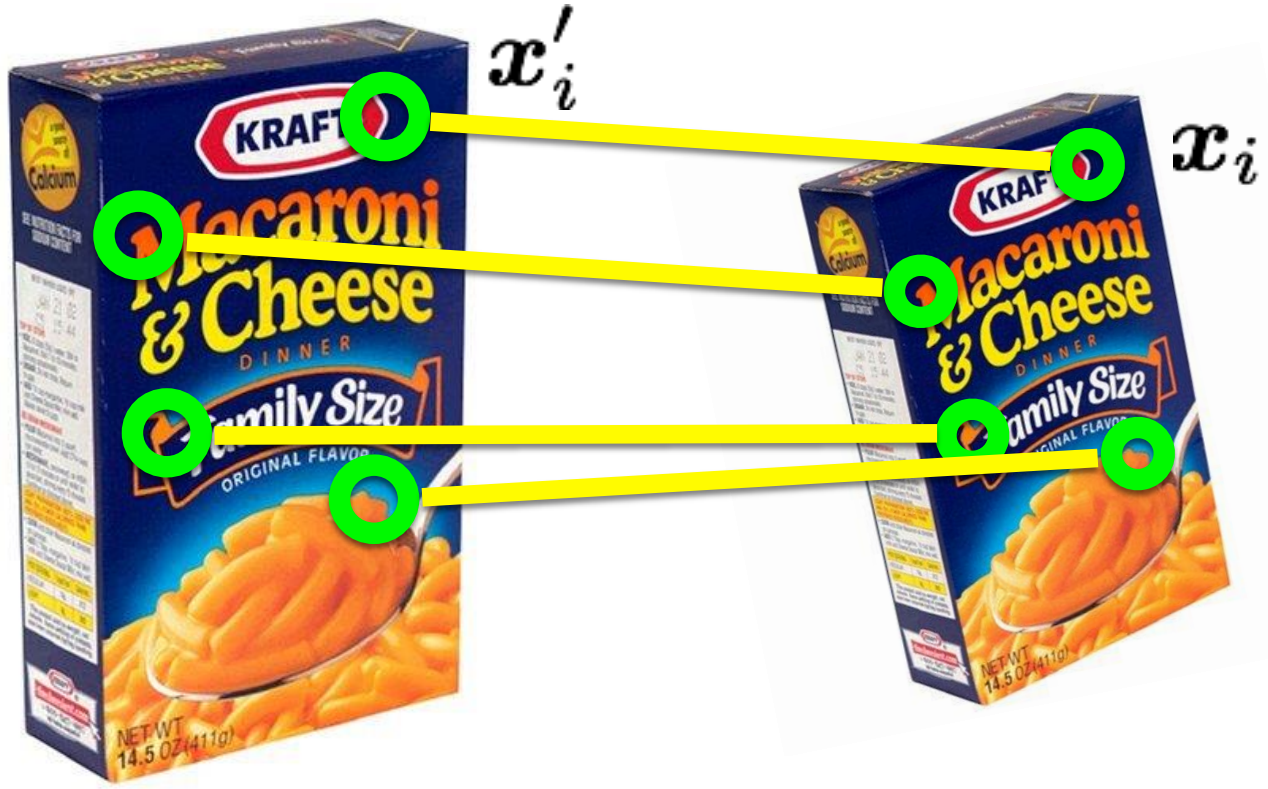
$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$

transformation function

parameters

find the best estimate of the parameters

$$\boldsymbol{p}$$



What kind of transformation functions $\boldsymbol{f}$ are there?

2D transformations

2D transformations

translation

rotation

aspect

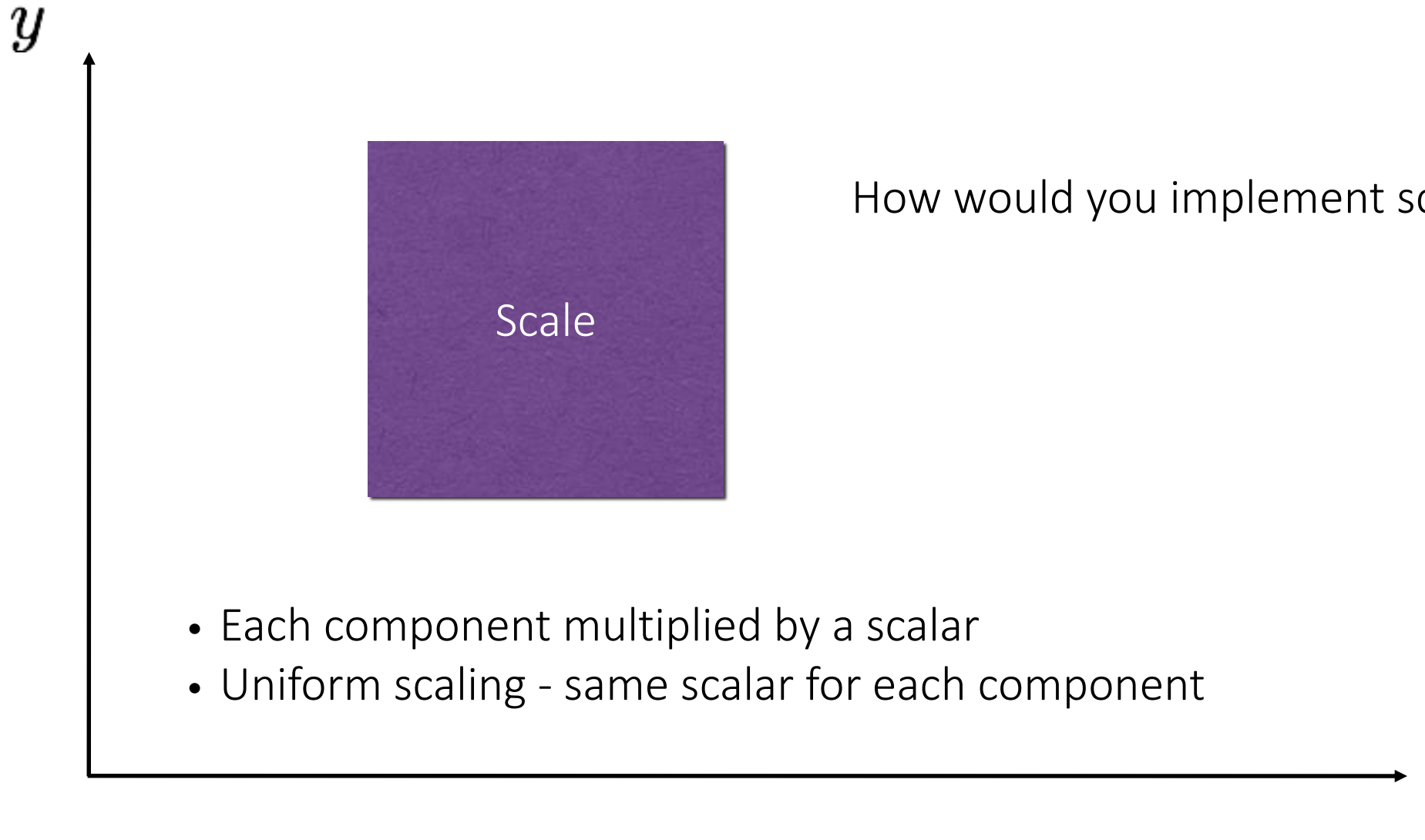
affine

perspective

cylindrical

2D planar transformations

2D planar transformations



How would you implement scaling?

- Each component multiplied by a scalar
- Uniform scaling - same scalar for each component

2D planar transformations

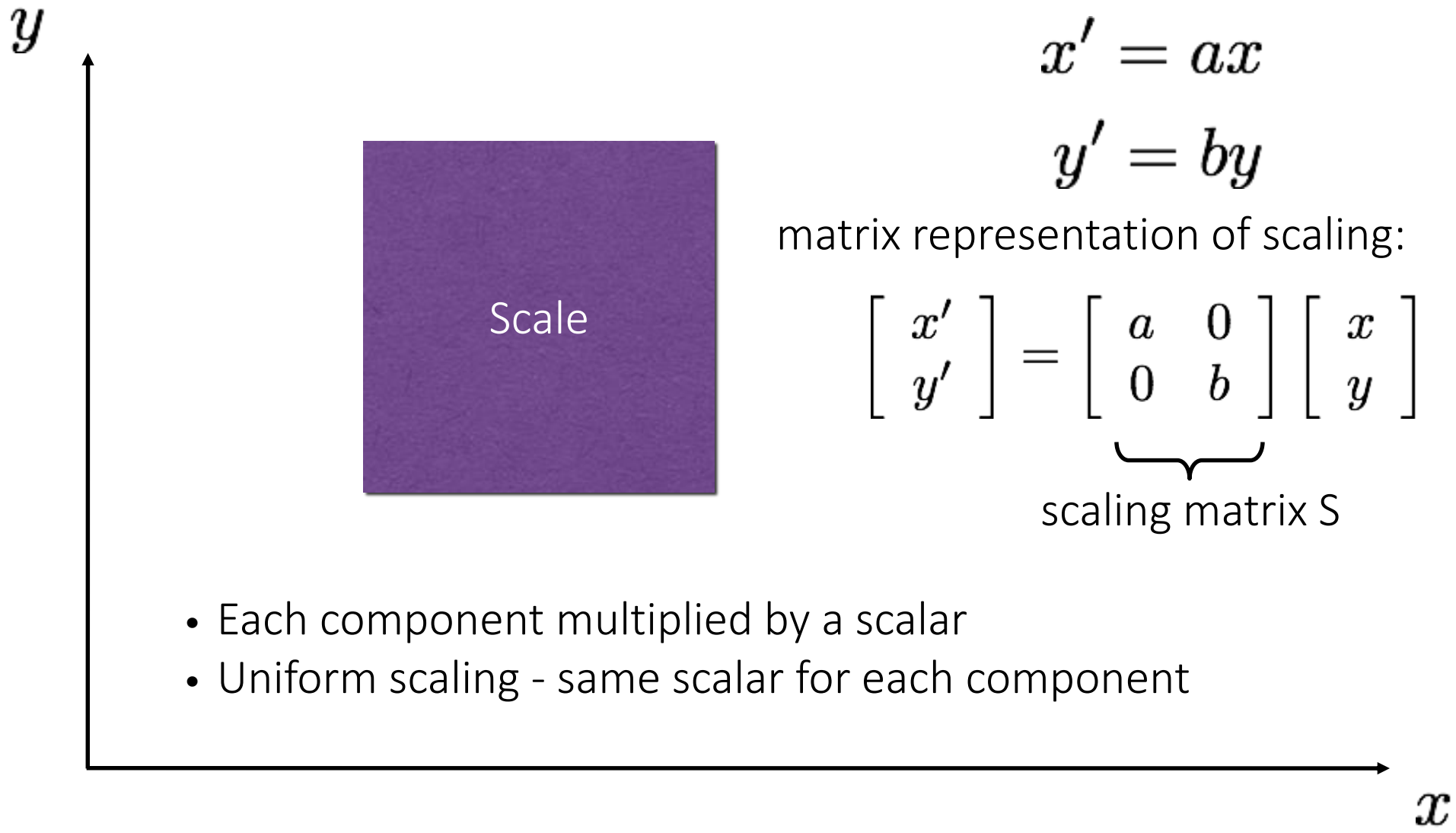
$$x' = ax$$

$$y' = by$$

What's the effect of using different scale factors?

- Each component multiplied by a scalar
- Uniform scaling - same scalar for each component

2D planar transformations



Scale

$$x' = ax$$

$$y' = by$$

matrix representation of scaling:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

scaling matrix S

- Each component multiplied by a scalar
- Uniform scaling - same scalar for each component

2D planar transformations

How would you implement shearing?

2D planar transformations

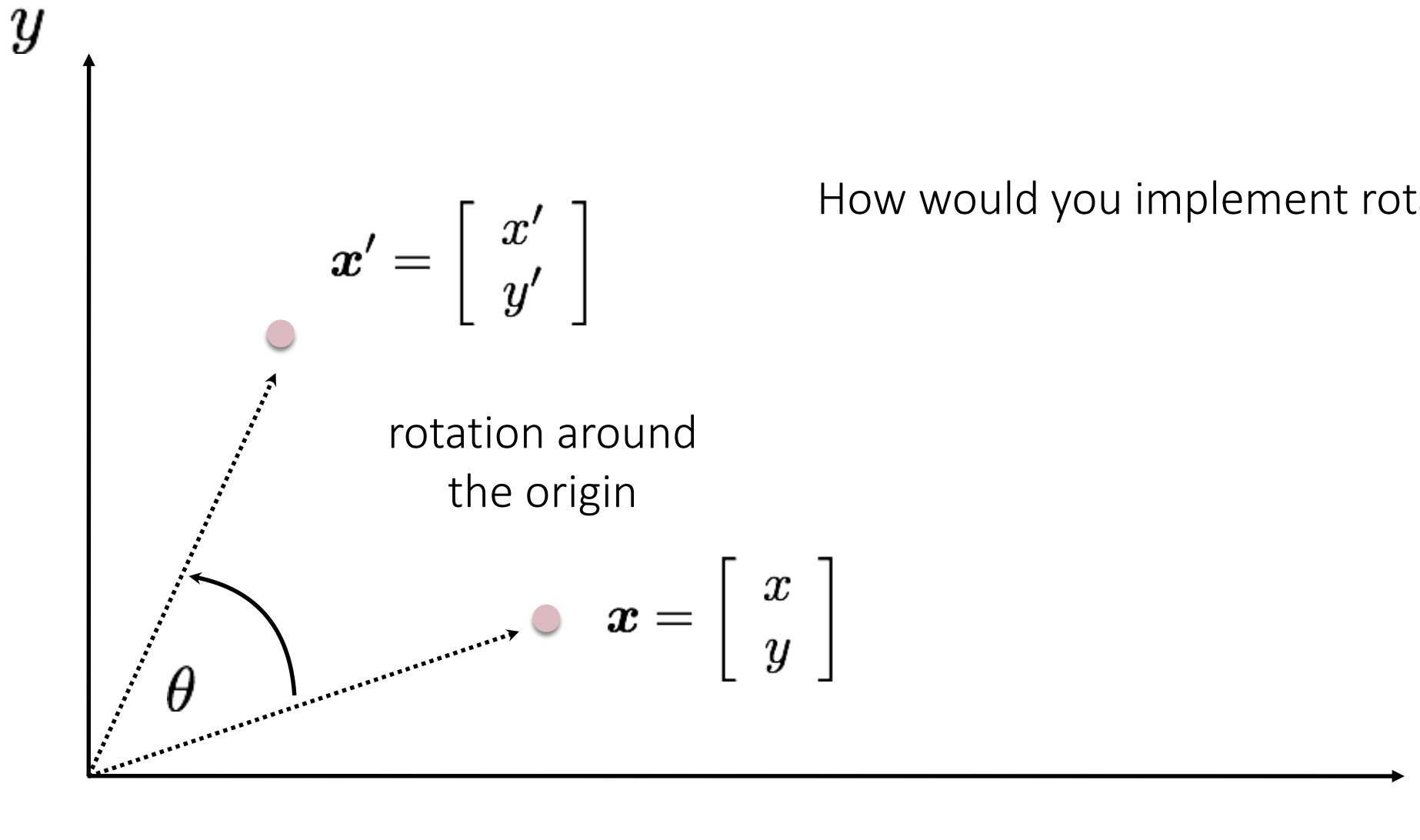
$$x' = x + a \cdot y$$

$$y' = b \cdot x + y$$

or in matrix form:

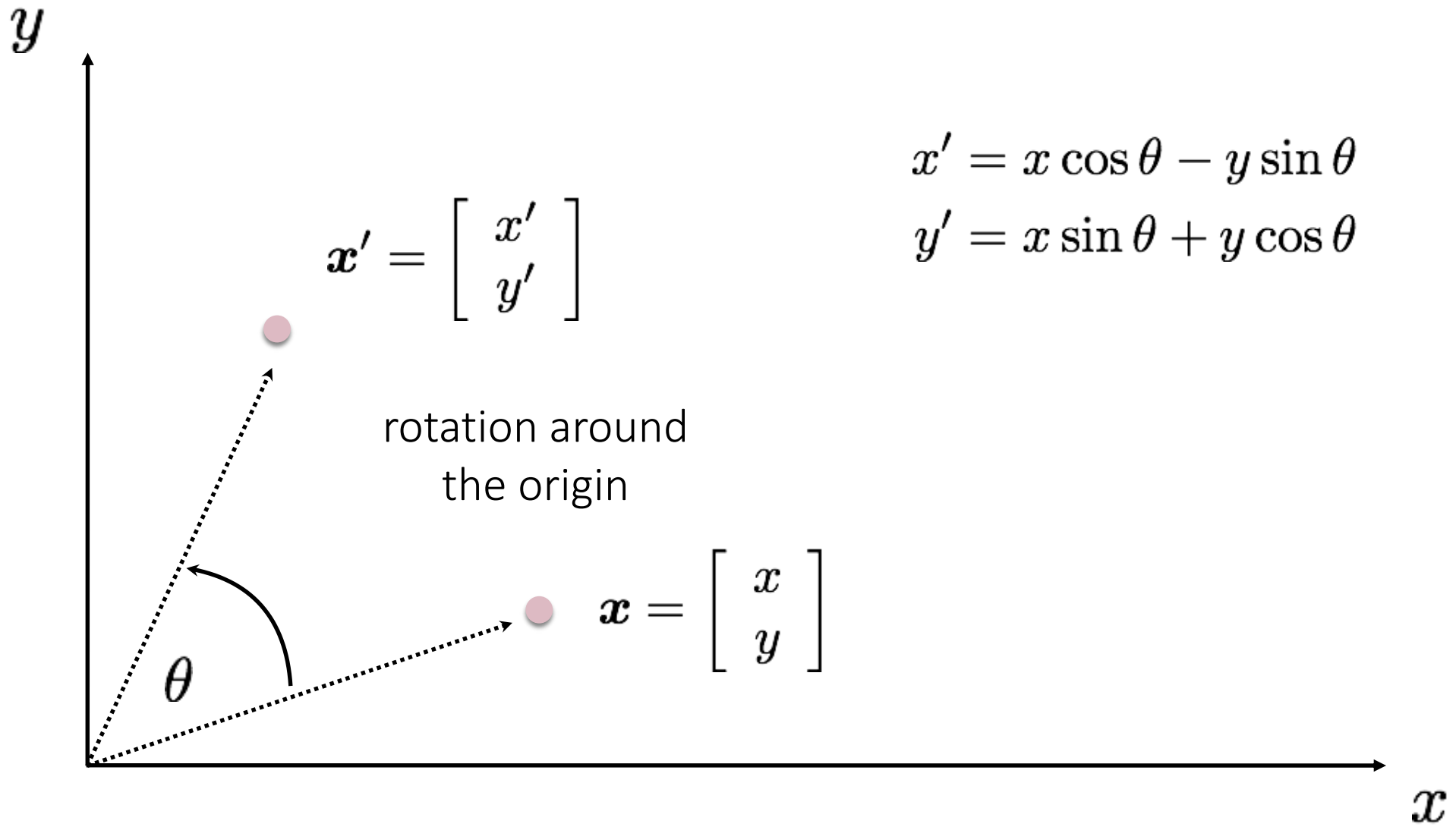
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D planar transformations



How would you implement rotation?

2D planar transformations

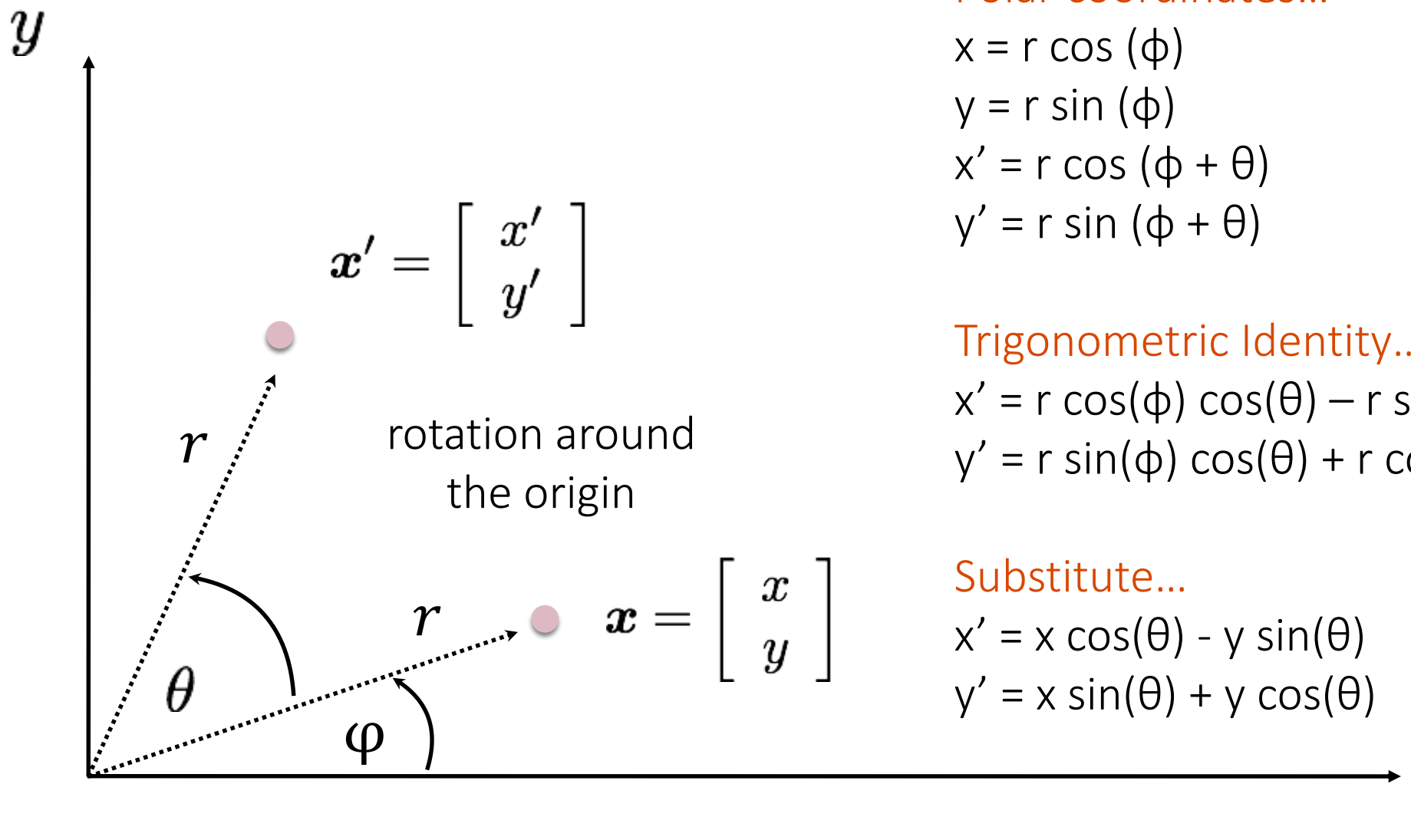


$$\boldsymbol{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

2D planar transformations



Polar coordinates…

x = r cos (ϕ)

y = r sin (ϕ)

x’ = r cos (ϕ + θ)

y’ = r sin (ϕ + θ)

Trigonometric Identity…

x’ = r cos(ϕ) cos(θ) – r sin(ϕ) sin(θ)

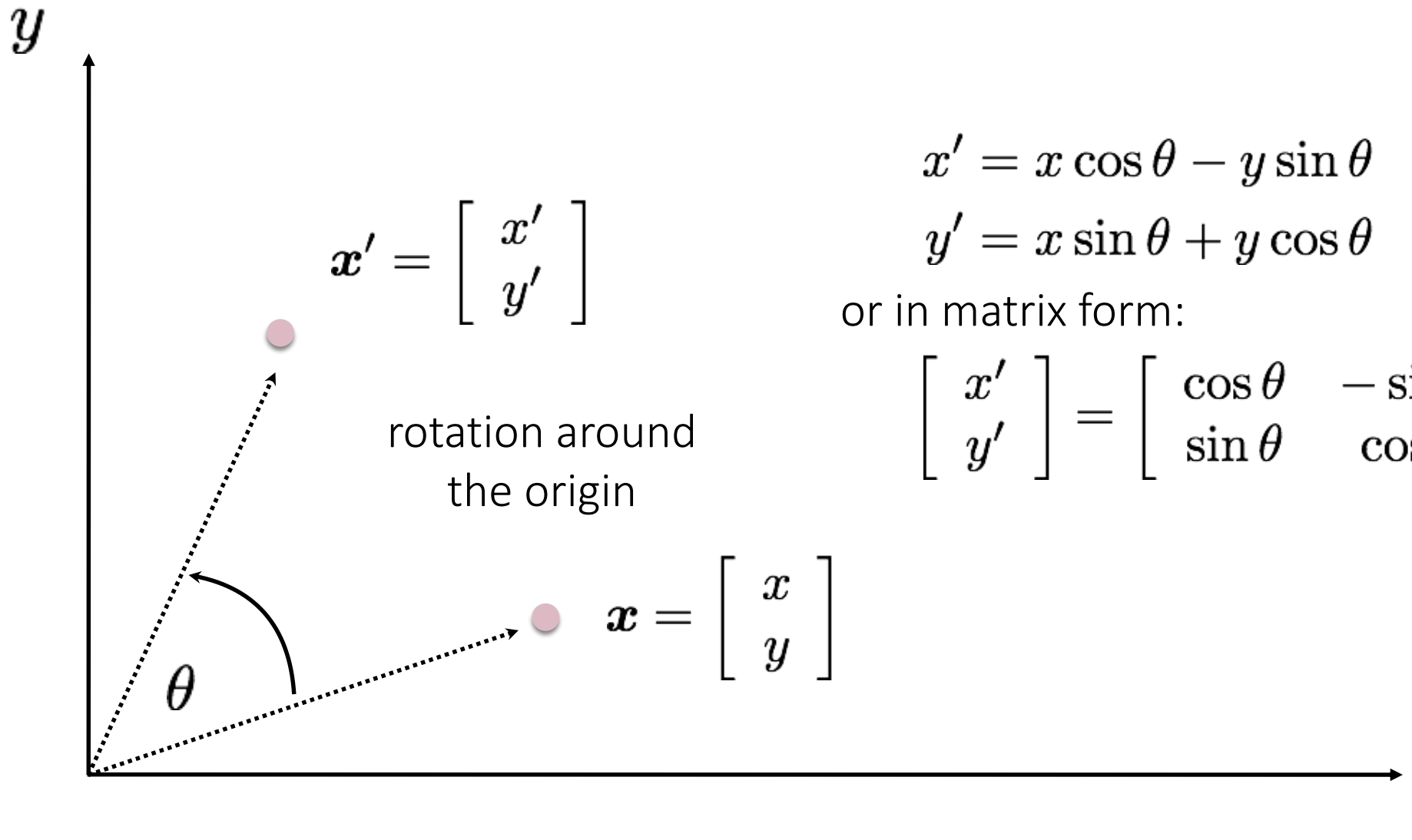
y’ = r sin(ϕ) cos(θ) + r cos(ϕ) sin(θ)

Substitute…

x’ = x cos(θ) - y sin(θ)

y’ = x sin(θ) + y cos(θ)

2D planar transformations



$$\boldsymbol{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

rotation around the origin

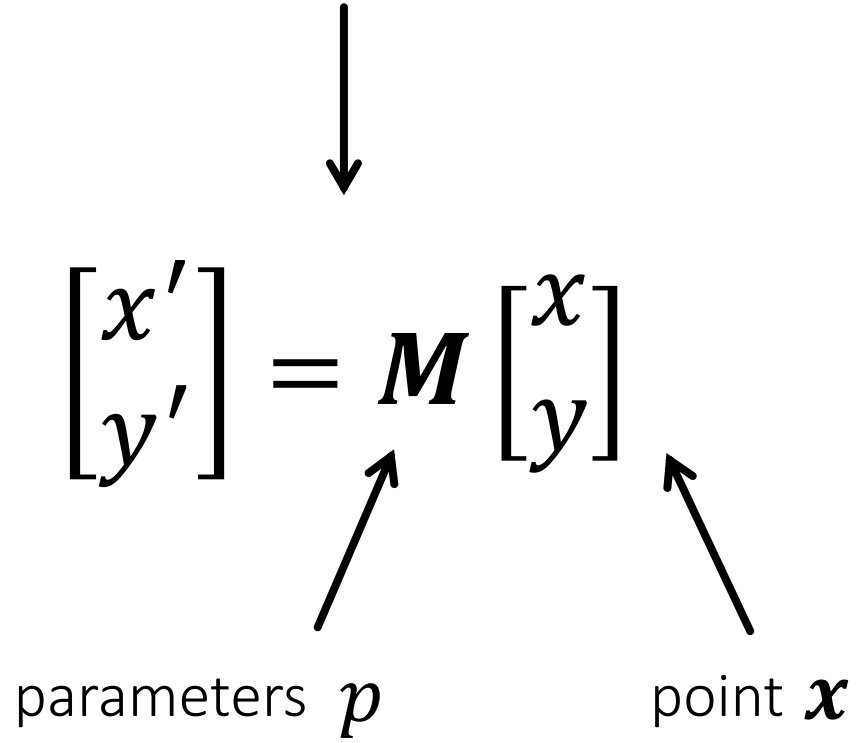
$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

or in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D planar and linear transformations

$$\boldsymbol{x}' = f(\boldsymbol{x}; p)$$



2D planar and linear transformations

Scale

$$\mathbf{M} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Flip across y

$$\mathbf{M} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Rotate

$$\mathbf{M} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Flip across origin

$$\mathbf{M} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

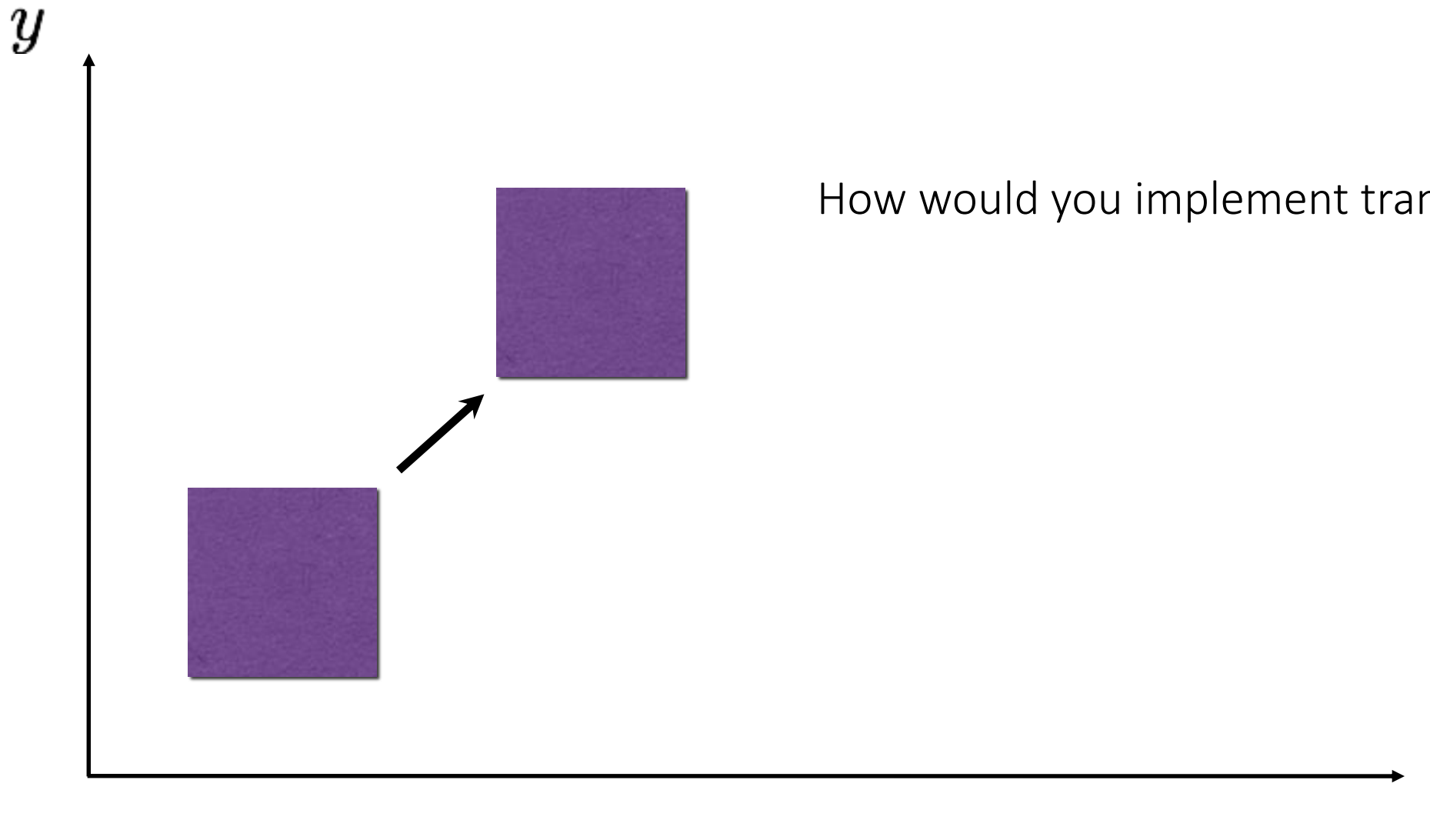
Shear

$$\mathbf{M} = \begin{bmatrix} 1 & s_x \\ s_y & 1 \end{bmatrix}$$

Identity

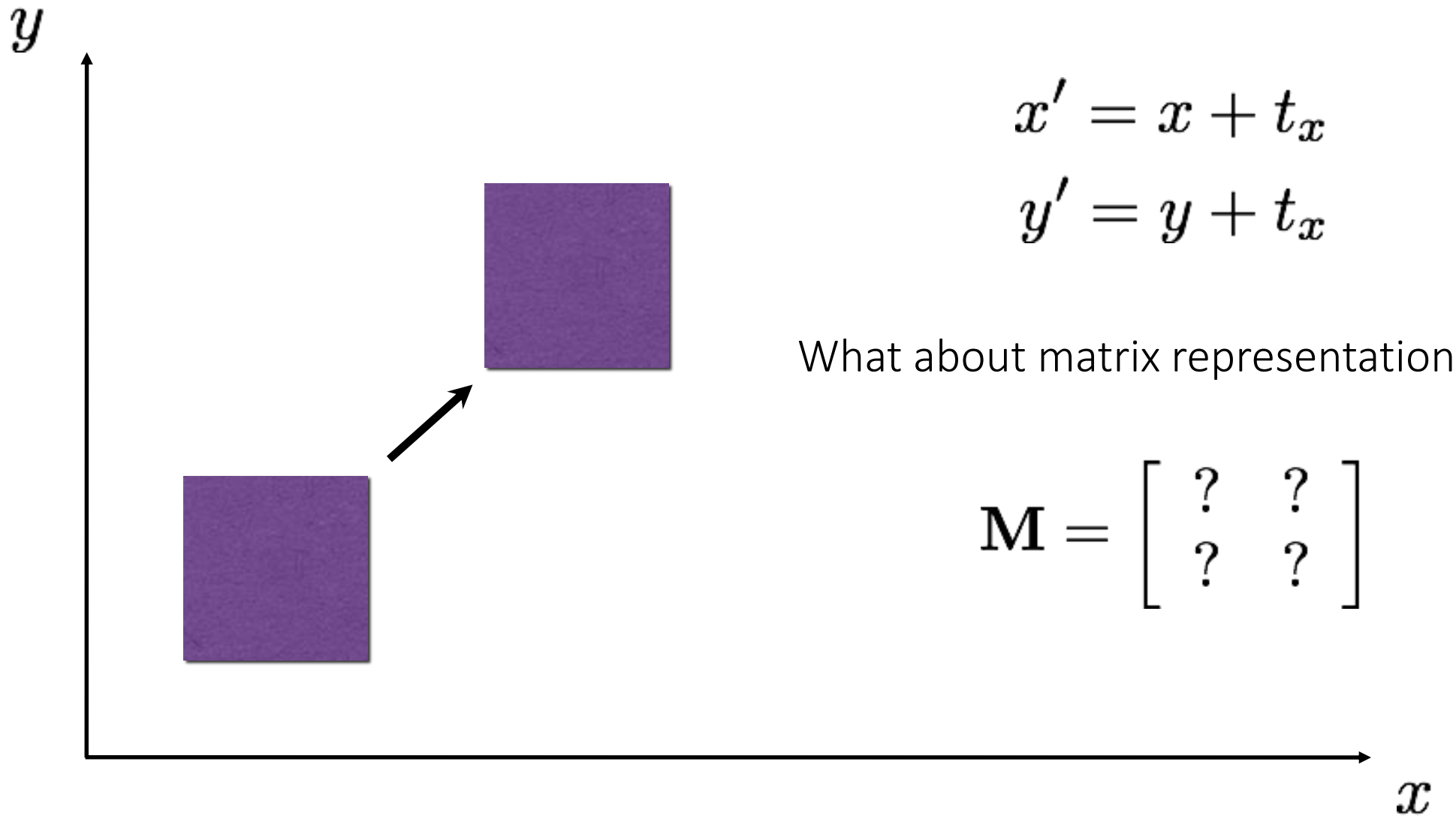
$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D translation



How would you implement translation?

2D translation

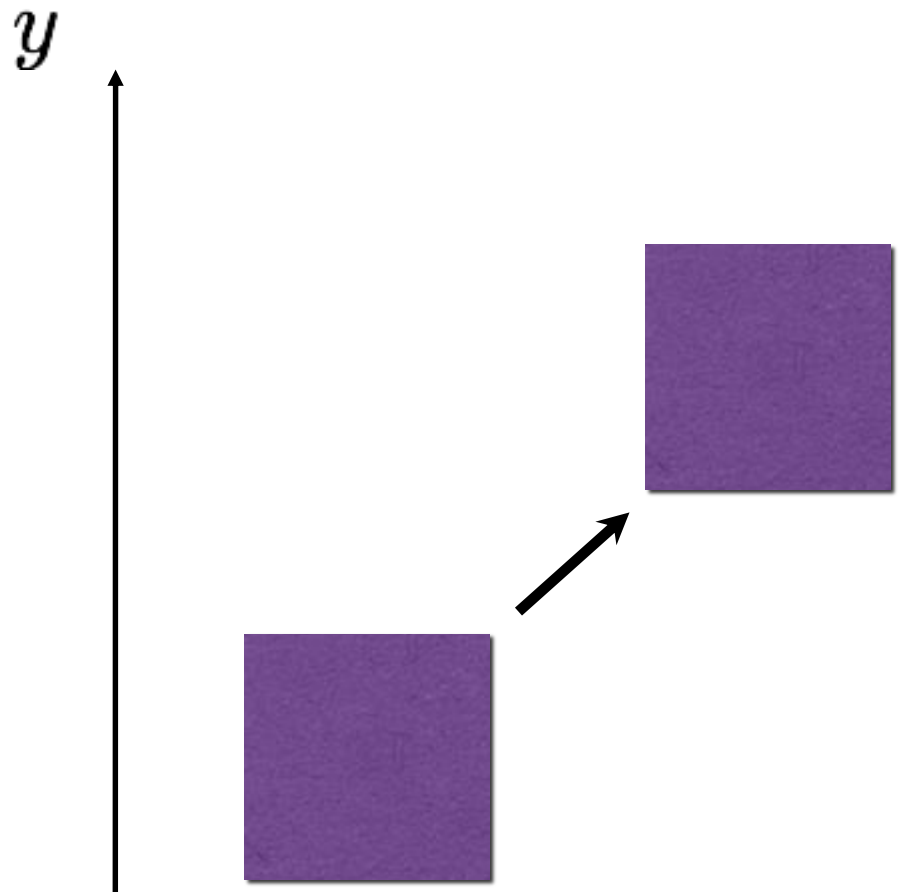


$$x' = x + t_x$$
$$y' = y + t_x$$

What about matrix representation?

$$\mathbf{M} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$$

2D translation



$$x' = x + t_x$$
$$y' = y + t_x$$

What about matrix representation?

Not possible.

Projective geometry 101

Homogeneous coordinates

heterogeneous coordinates

homogeneous coordinates

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

add a 1 here

- Represent 2D point with a 3D vector

Homogeneous coordinates

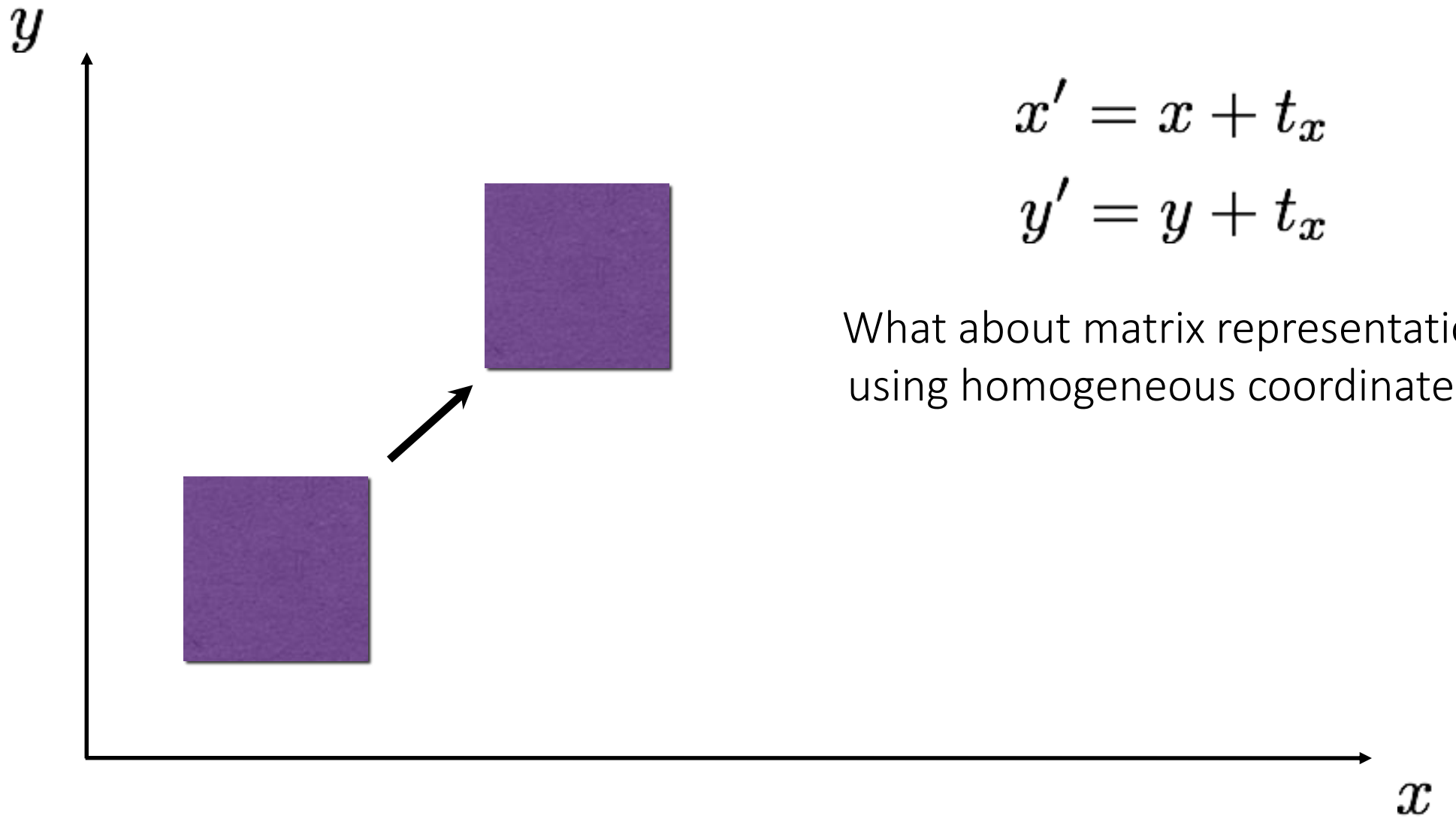
heterogeneous coordinates

homogeneous coordinates

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \overset{\text{def}}{=\!=} \begin{bmatrix} ax \\ ay \\ a \end{bmatrix}$$

- Represent 2D point with a 3D vector
- 3D vectors are only defined up to scale

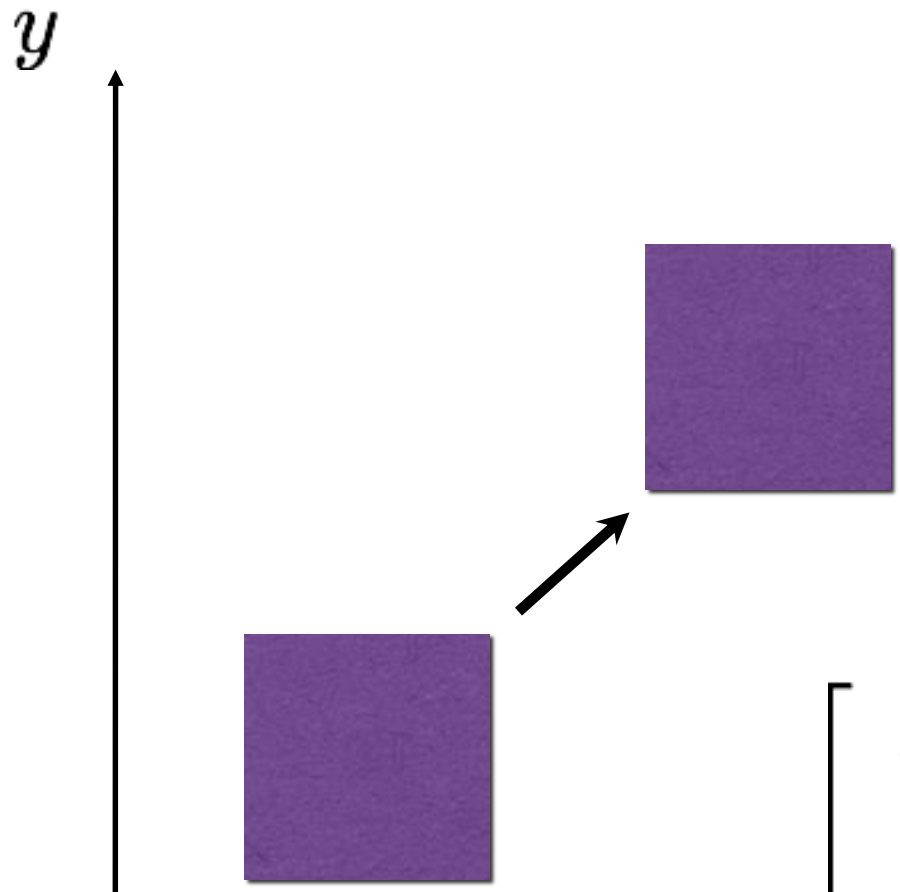
2D translation



$$x' = x + t_x$$
$$y' = y + t_x$$

What about matrix representation using homogeneous coordinates?

2D translation



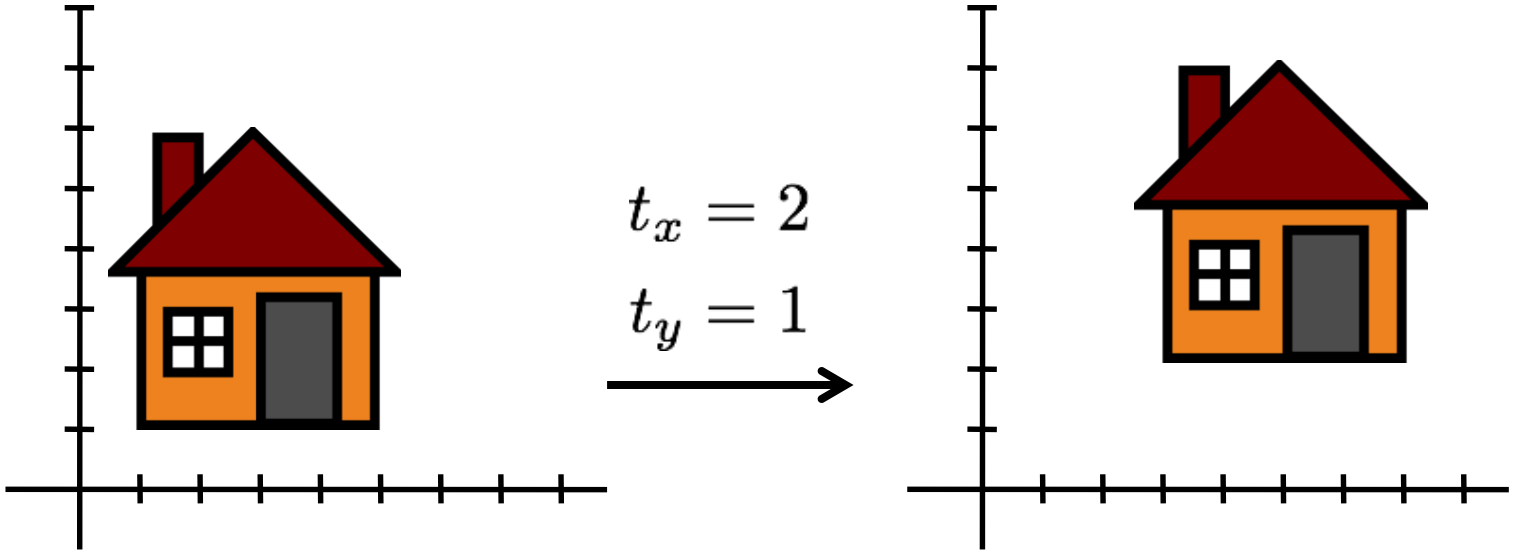
$$x' = x + t_x$$
$$y' = y + t_x$$

What about matrix representation using heterogeneous coordinates?

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \mathbf{M} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2D translation using homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$



Homogeneous coordinates

Conversion:

- heterogeneous → homogeneous

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- homogeneous → heterogeneous

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

- scale invariance

$$\begin{bmatrix} x & y & w \end{bmatrix}^\top = \lambda \begin{bmatrix} x & y & w \end{bmatrix}^\top$$

Special points:

- point at infinity

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

- undefined

$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Projective geometry

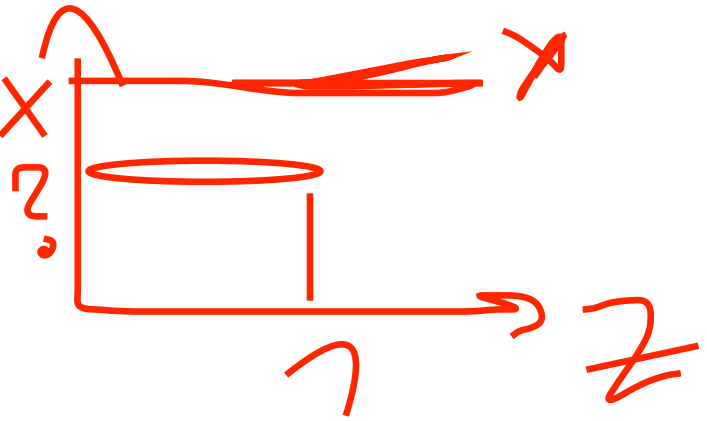


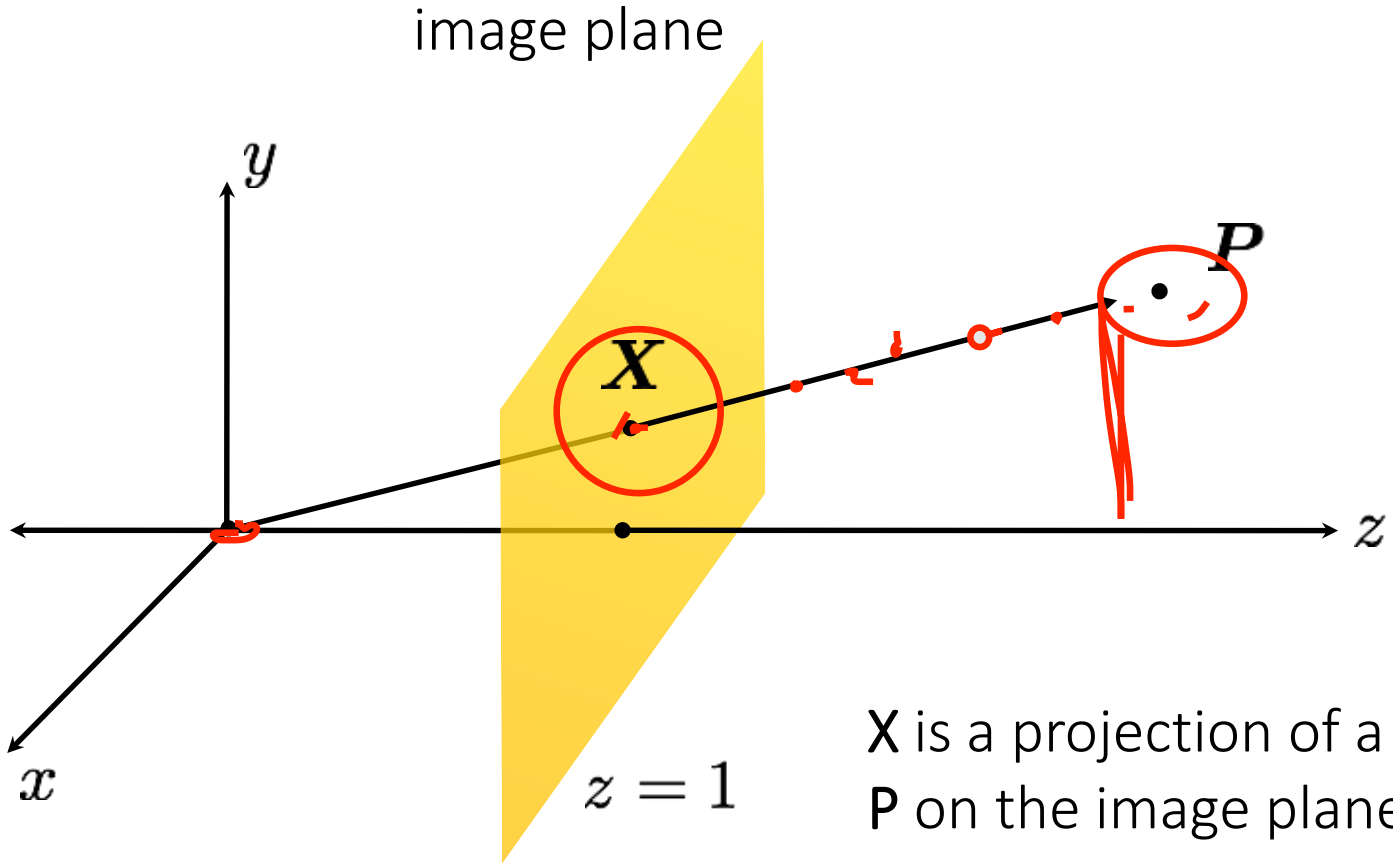
image point in pixel coordinates

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$\Updownarrow$

image point in homogeneous coordinates

$$\boldsymbol{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



X is a projection of a point P on the image plane

What does scaling X correspond to?

Transformations in projective geometry

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \quad ? \quad \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \quad ? \quad \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \quad ? \quad \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

shearing

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \quad ? \quad \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \quad ? \quad \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

shearing

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \quad & ? & \quad \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotation

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

shearing

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotation

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

shearing

Matrix composition

Transformations can be combined by matrix multiplication:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

p’ = ? ? ? p

Matrix composition

Transformations can be combined by matrix multiplication:

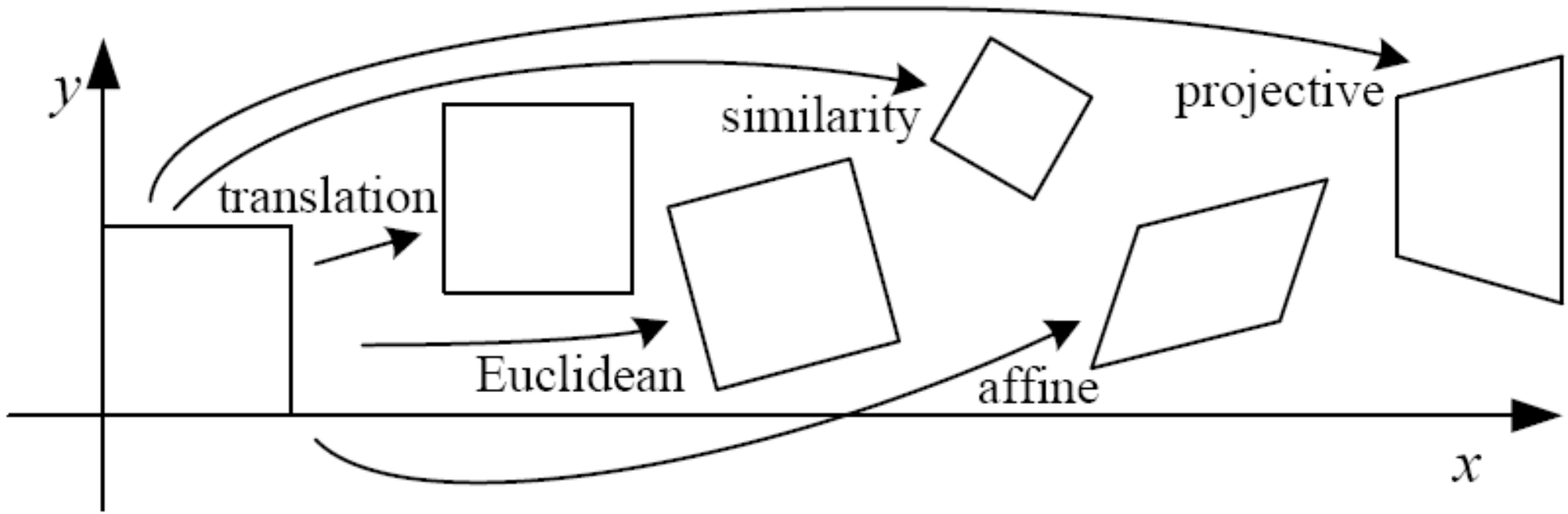
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$\mathbf{p}'$ = translation(t_x,t_y) rotation(θ) scale(s,s) $\mathbf{p}$

Does the multiplication order matter?

Classification of 2D transformations

Classification of 2D transformations



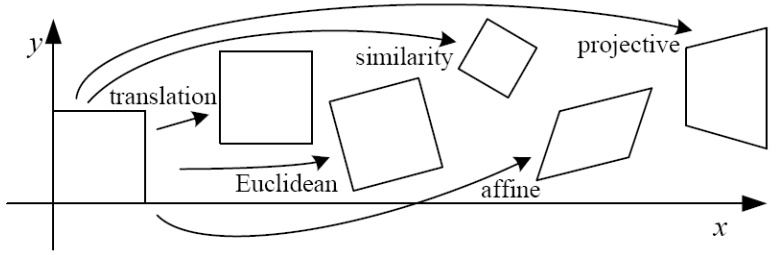
Classification of 2D transformations

Name	Matrix	# D.O.F.
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}$	?
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}$	?
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}$	?
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}$	?
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}$	?

Classification of 2D transformations

Translation: $\begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$

How many degrees of freedom?

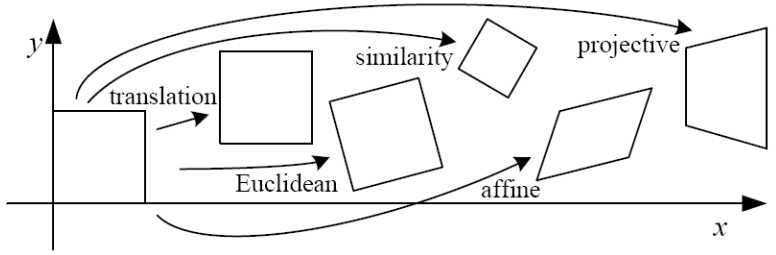


Classification of 2D transformations

Euclidean (rigid):
rotation + translation

$$\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?

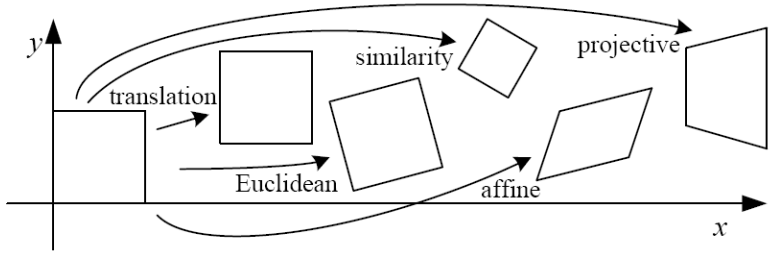


Classification of 2D transformations

Euclidean (rigid):
rotation + translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & r_3 \\ \sin\theta & \cos\theta & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom?

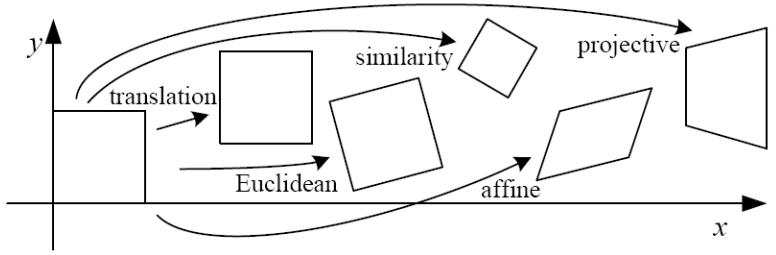


Classification of 2D transformations

which other matrix values will change if this increases?

Euclidean (rigid): rotation + translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & r_3 \\ \sin\theta & \cos\theta & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

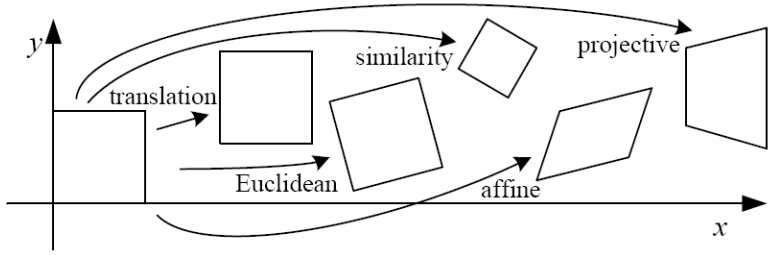


Classification of 2D transformations

what will happen to the image if this increases?

Euclidean (rigid): rotation + translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & r_3 \\ \sin\theta & \cos\theta & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

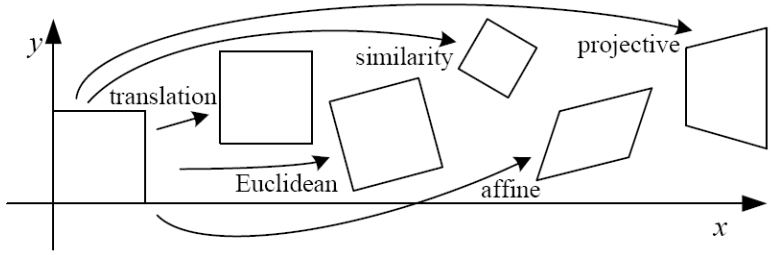


Classification of 2D transformations

what will happen to the image if this increases?

Euclidean (rigid): rotation + translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & r_3 \\ \sin\theta & \cos\theta & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

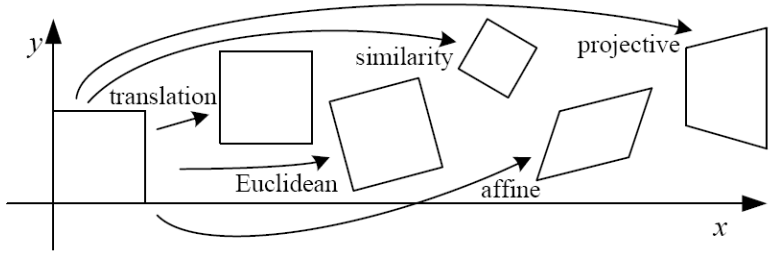


Classification of 2D transformations

Similarity:
uniform scaling + rotation
+ translation

$$\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?



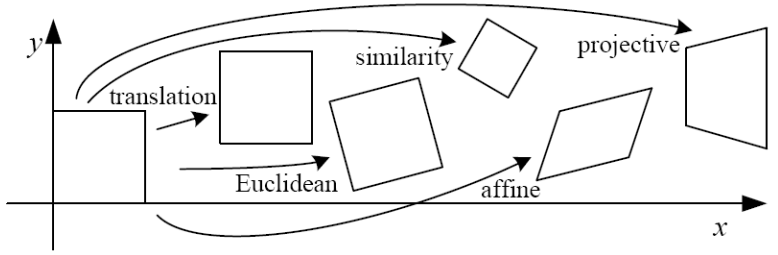
Classification of 2D transformations

multiply these four by scale **s**

Similarity:
uniform scaling + rotation
+ translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & r_3 \\ \sin\theta & \cos\theta & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom?

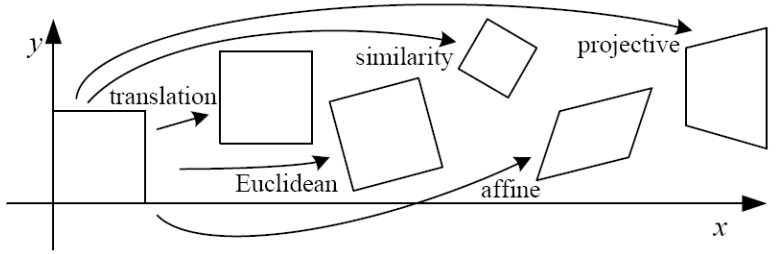


Classification of 2D transformations

what will happen to the image if this increases?

Similarity:
uniform scaling + rotation
+ translation

$$\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

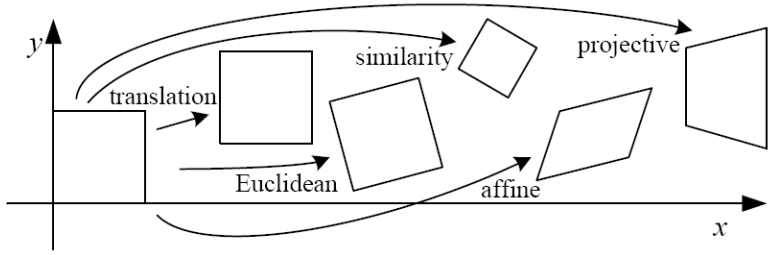


Classification of 2D transformations

Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?



Classification of 2D transformations

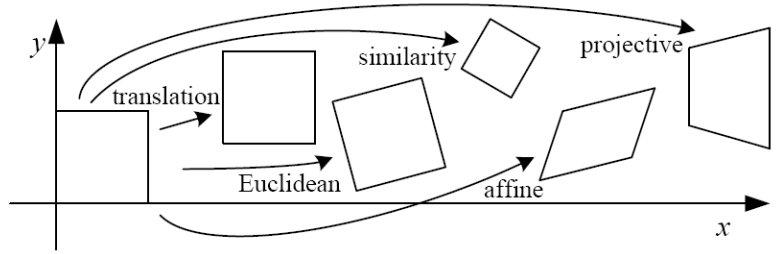
Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?

similarity shear

$$\begin{bmatrix} sr_1 & sr_2 \\ sr_3 & sr_4 \end{bmatrix} \begin{bmatrix} 1 & h_1 \\ h_2 & 1 \end{bmatrix} = \begin{bmatrix} sr_1 + h_2 sr_2 & sr_2 + h_1 sr_1 \\ sr_3 + h_2 sr_4 & sr_4 + h_1 sr_3 \end{bmatrix}$$



Classification of 2D transformations

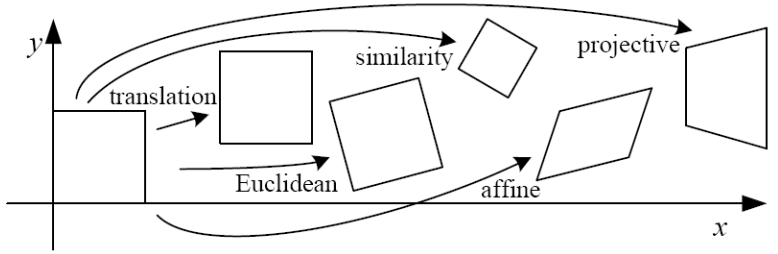
Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom?

similarity shear

$$\begin{bmatrix} sr_1 & sr_2 \\ sr_3 & sr_4 \end{bmatrix} \begin{bmatrix} 1 & h_1 \\ h_2 & 1 \end{bmatrix} = \begin{bmatrix} sr_1 + h_2 sr_2 & sr_2 + h_1 sr_1 \\ sr_3 + h_2 sr_4 & sr_4 + h_1 sr_3 \end{bmatrix}$$



Affine transformations

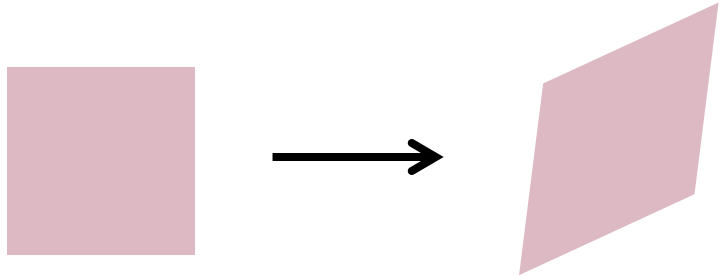
Affine transformations are combinations of

- arbitrary (4-DOF) linear transformations; and
- translations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of affine transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines map to parallel lines
- ratios are preserved
- compositions of affine transforms are also affine transforms



Does the last coordinate w ever change?

Affine transformations

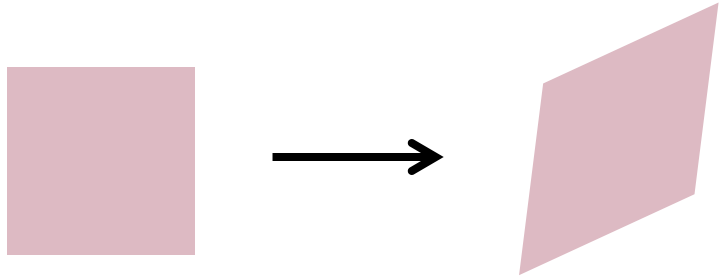
Affine transformations are combinations of

- arbitrary (4-DOF) linear transformations; and
- translations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of affine transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines map to parallel lines
- ratios are preserved
- compositions of affine transforms are also affine transforms



Nope! But what does that mean?

How to interpret affine transformations here?

image point in pixel coordinates

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

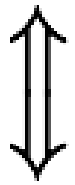
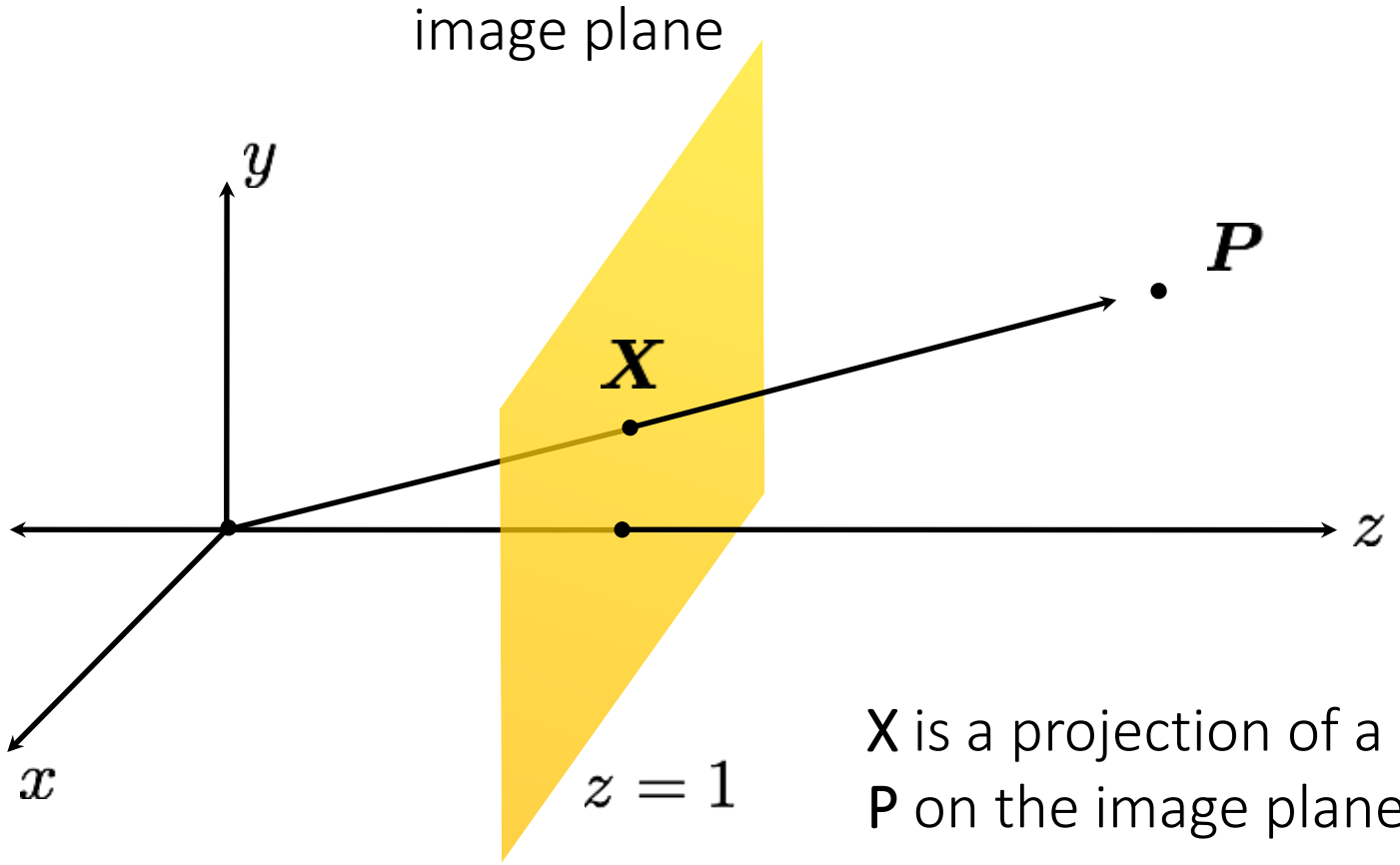


image point in heterogeneous coordinates

$$\boldsymbol{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



X is a projection of a point P on the image plane

Projective transformations

Projective transformations are combinations of

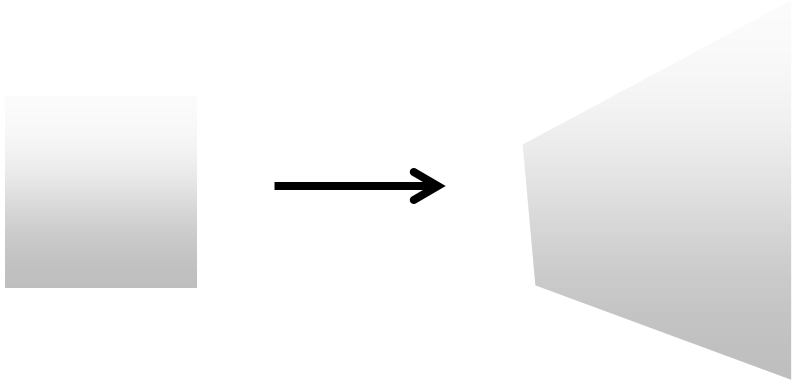
- affine transformations; and
- projective wraps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

How many degrees of freedom?

Properties of projective transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms



Projective transformations

Projective transformations are combinations of

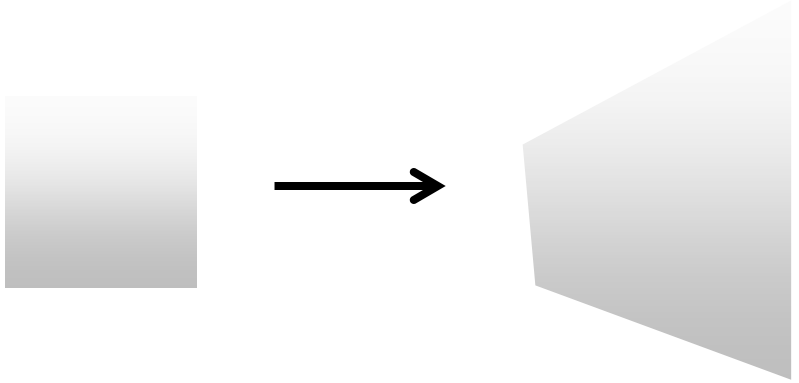
- affine transformations; and
- projective wraps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

8 DOF: vectors (and therefore matrices) are defined up to scale)

Properties of projective transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms



How to interpret projective transformations here?

image point in pixel coordinates

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

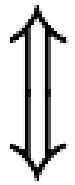
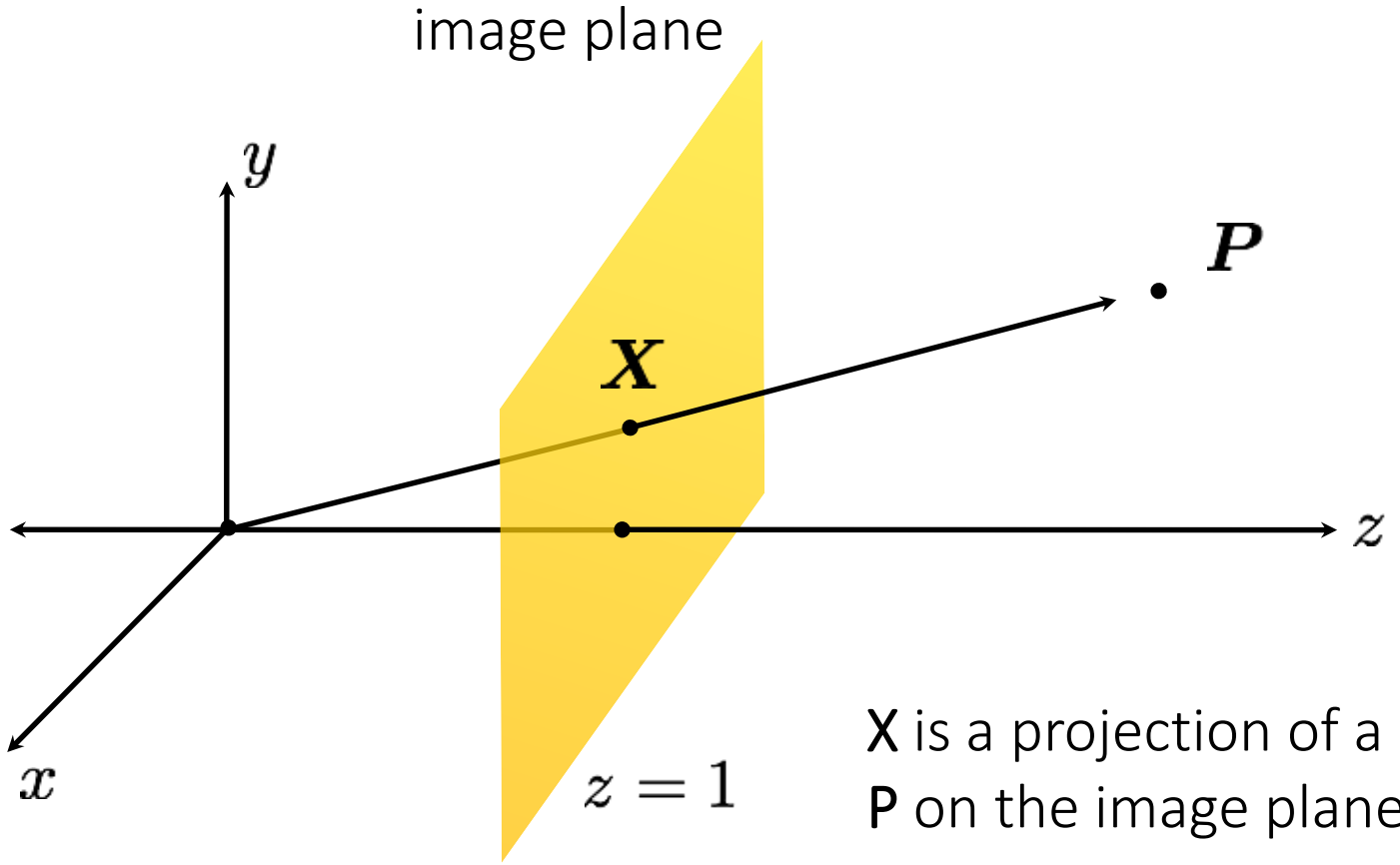


image point in heterogeneous coordinates

$$\boldsymbol{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

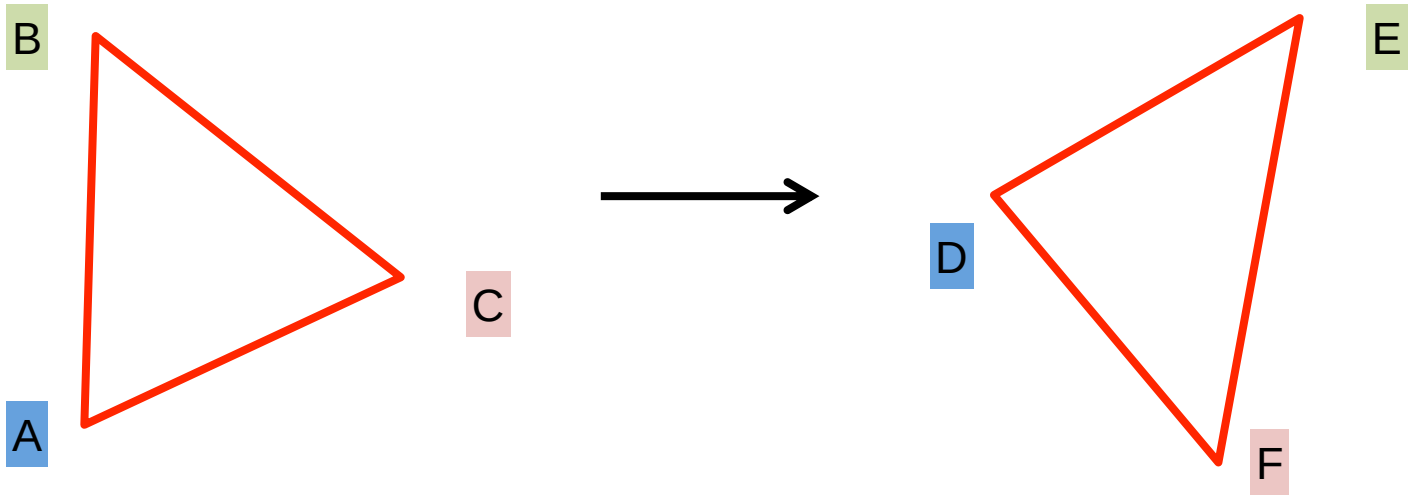


X is a projection of a point P on the image plane

Determining unknown 2D transformations

Determining unknown transformations

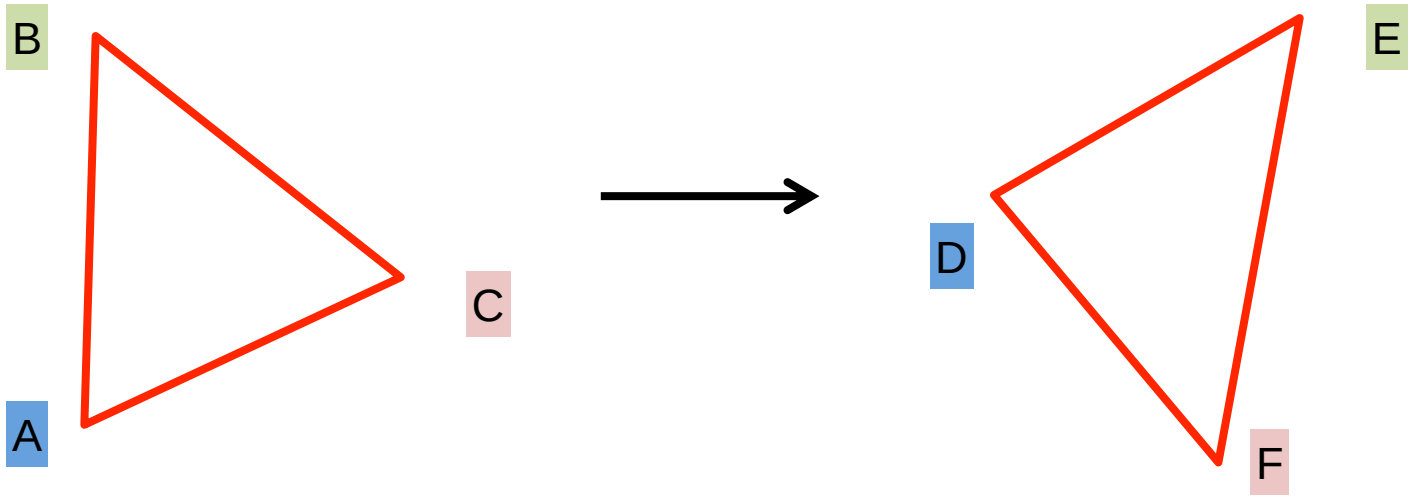
Suppose we have two triangles: ABC and DEF.

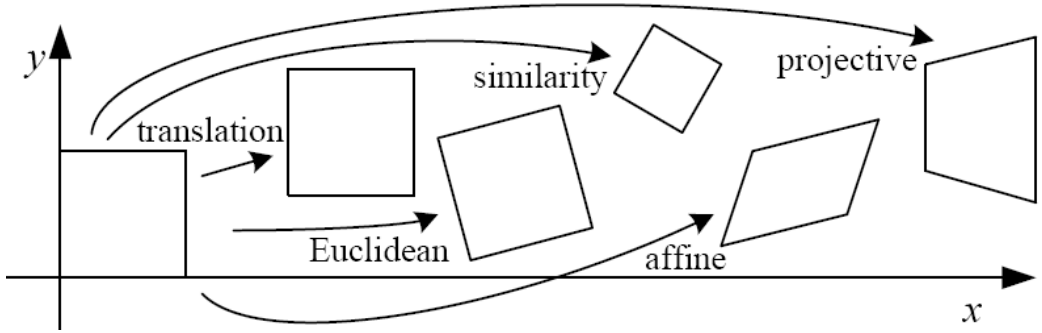


Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?

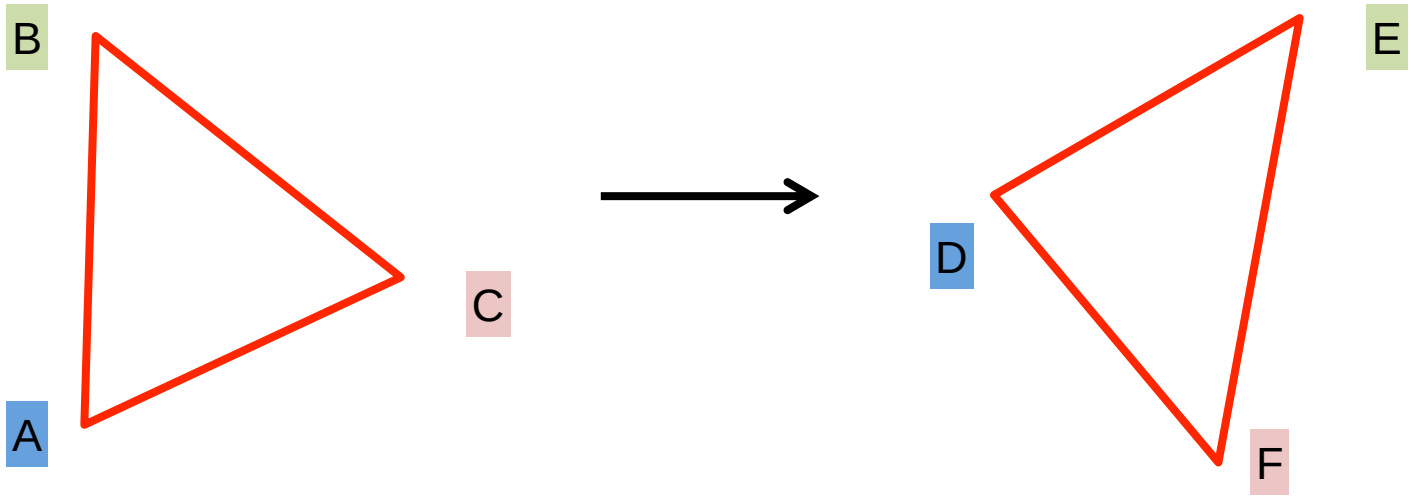




Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
- How do we determine the unknown parameters?



Affine transform:
uniform scaling + shearing
+ rotation + translation

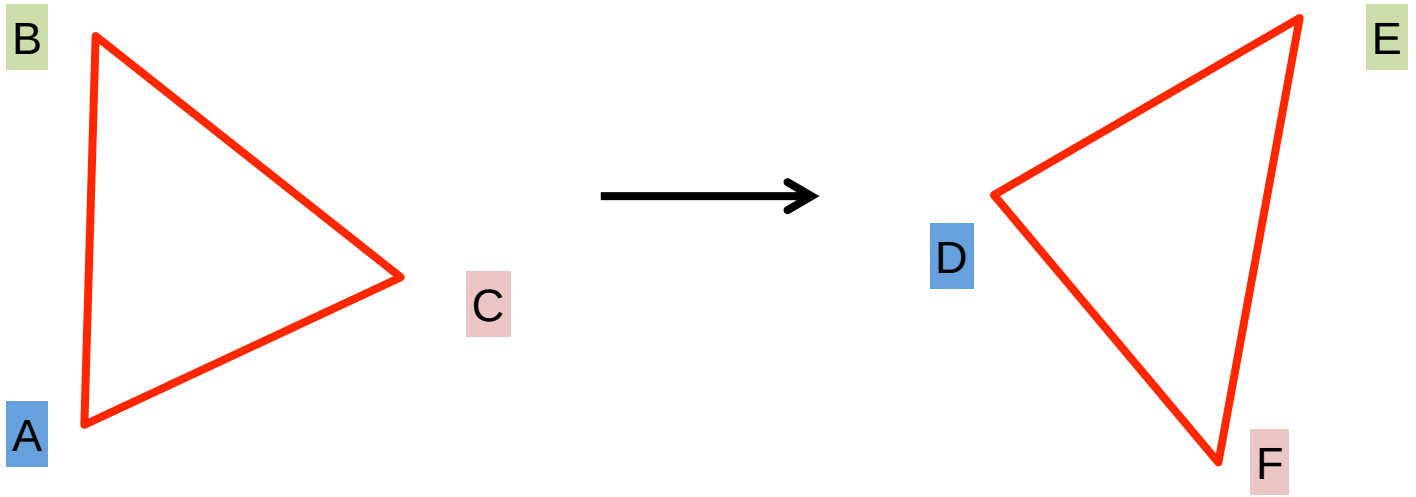
$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom do we have?

Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
- How do we determine the unknown parameters?



unknowns

$$x' = \mathbf{M}x$$

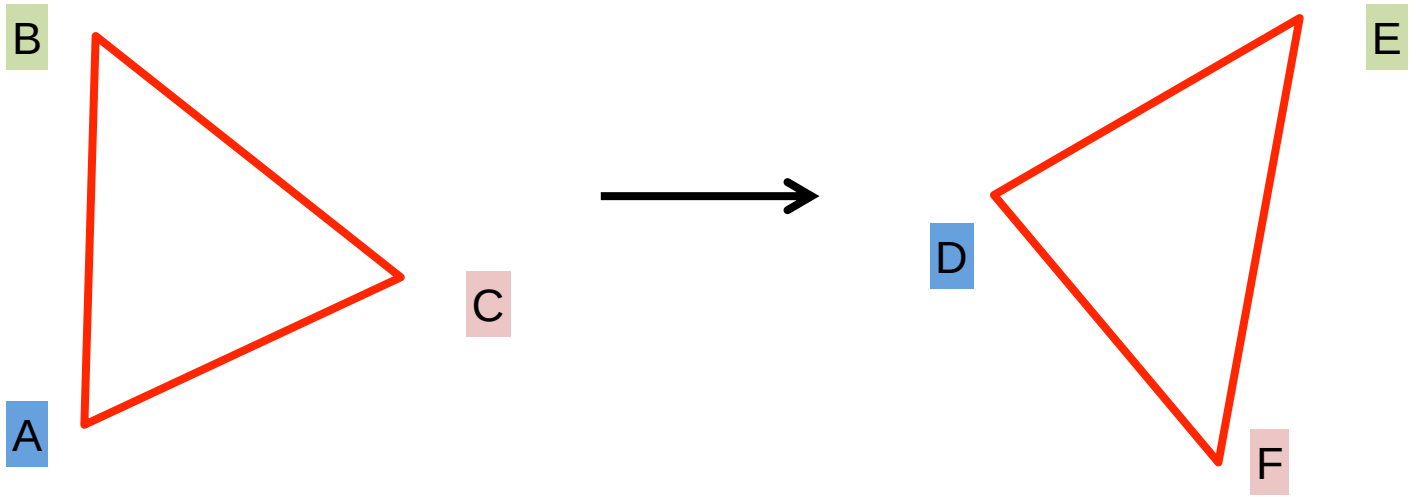
point correspondences

- One point correspondence gives how many equations?
- How many point correspondences do we need?

Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
- How do we determine the unknown parameters?

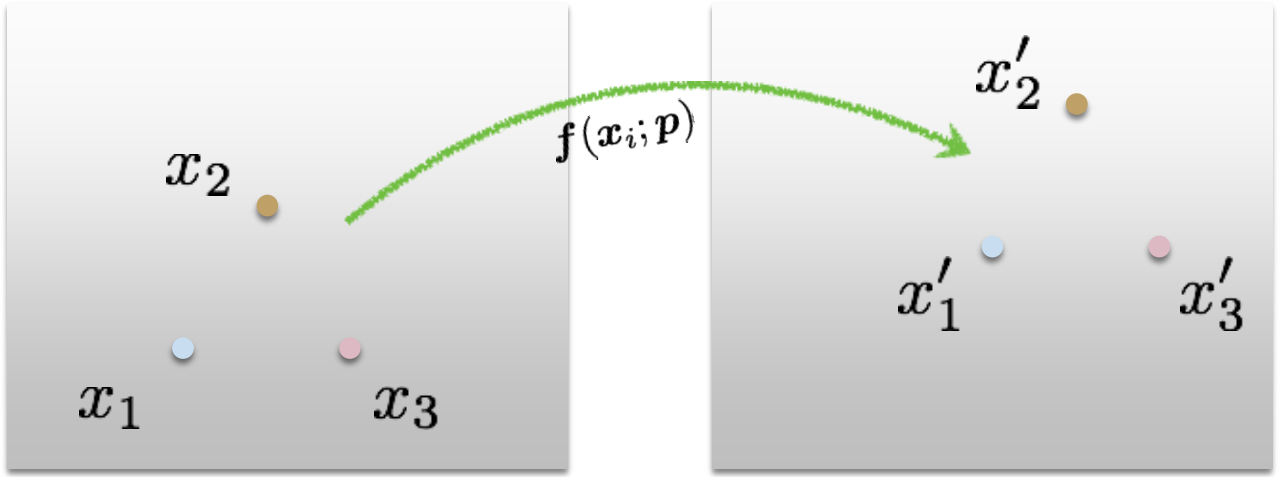


unknowns

$$x' = \mathbf{M}x$$

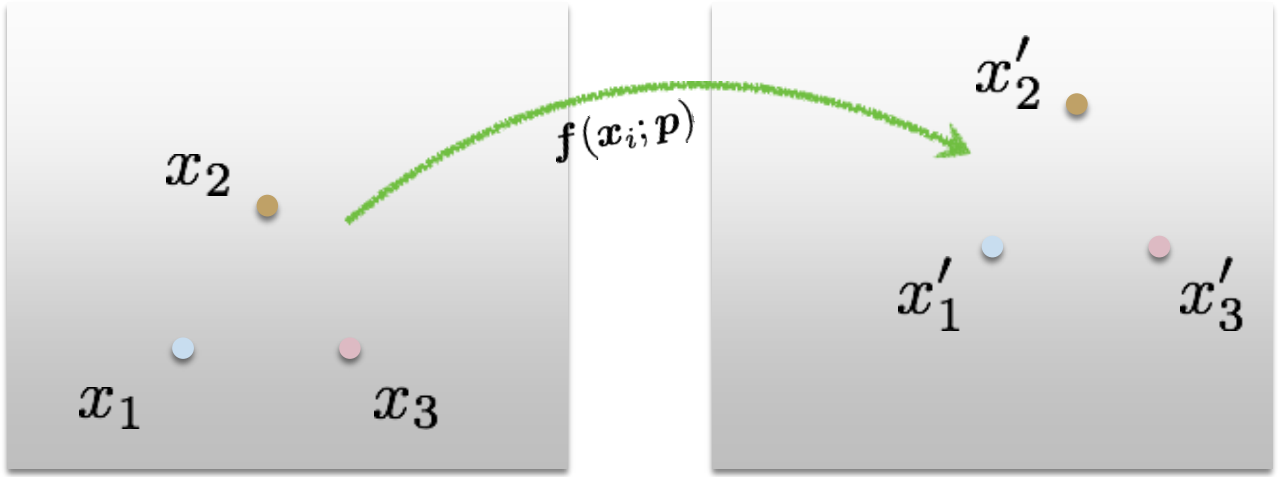
point correspondences

How do we solve this for **M**?



Least Squares Error

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$



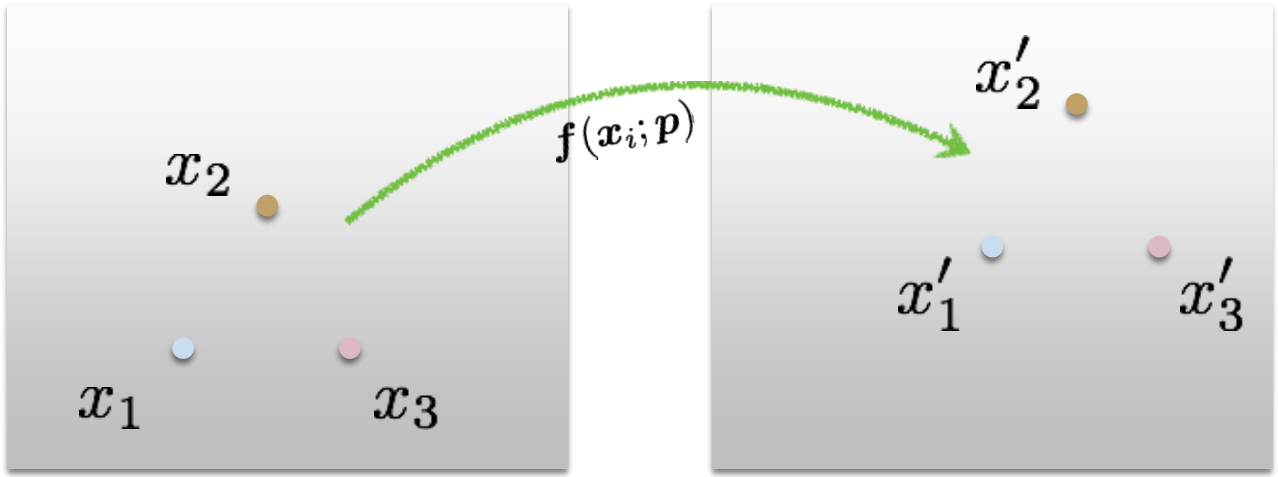
Least Squares Error

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

What is this?

What is this?

What is this?



$$||\boldsymbol{x}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Least Squares Error

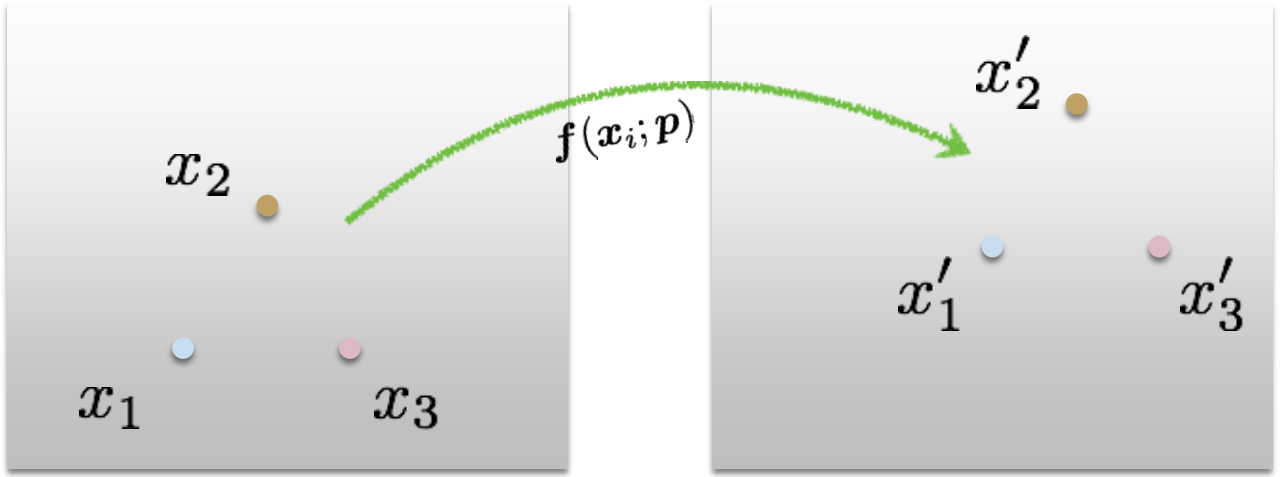
Euclidean (L2) norm

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

squared!

predicted location

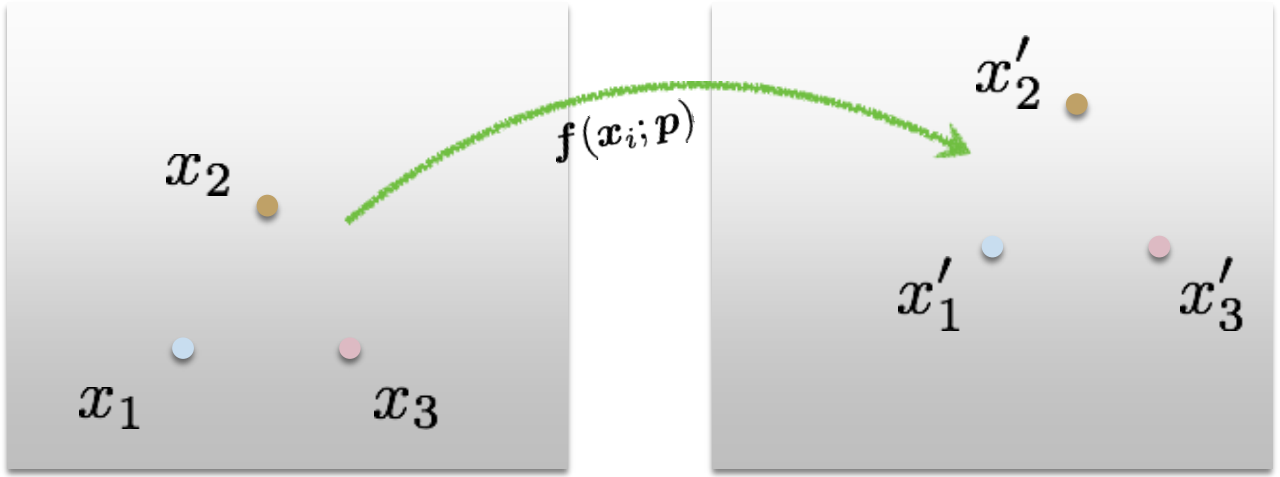
measured location



Least Squares Error

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

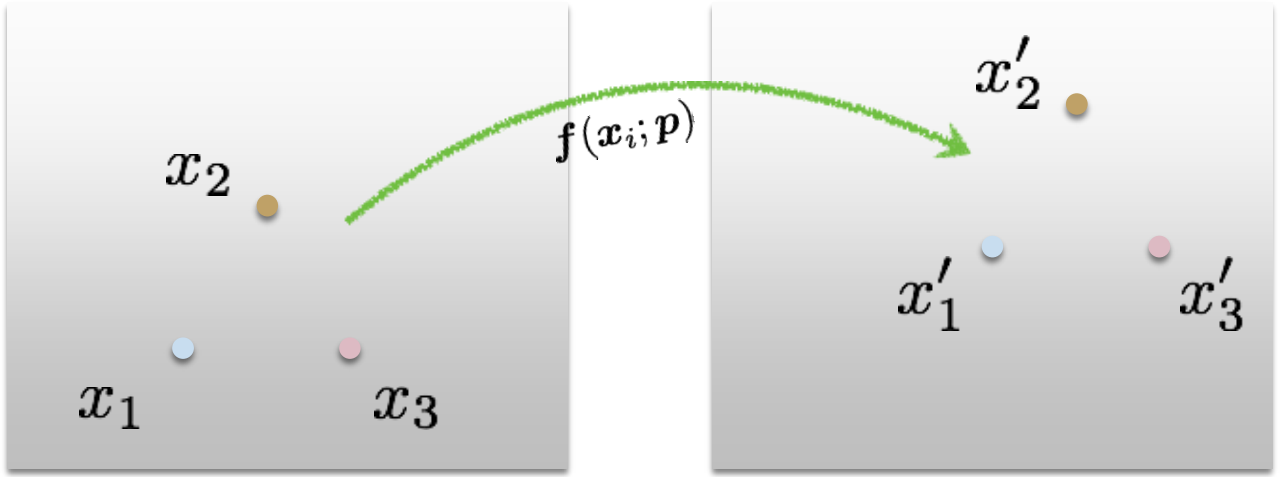
Residual (projection error)



Least Squares Error

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

What is the free variable?
What do we want to optimize?



Find parameters that minimize squared error

$$\hat{\boldsymbol{p}} = \arg\min_{\boldsymbol{p}} \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

General form of linear least squares

(**Warning:** change of notation. x is a vector of parameters!)

$$E_{\mathrm{LLS}} = \sum_i \left| \boldsymbol{a}_i \boldsymbol{x} - \boldsymbol{b}_i \right|^2$$

$$= \left\| \mathbf{A}\boldsymbol{x} - \boldsymbol{b} \right\|^2$$ (matrix form)

Determining unknown transformations

Affine transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Why can we drop the last line?

Vectorize transformation parameters:

Stack equations from point correspondences:

$$\begin{bmatrix} x' \\ y' \\ x' \\ y' \\ \vdots \\ x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ & & & \vdots & & \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}$$

$$\boldsymbol{b} \qquad \mathbf{A} \qquad \boldsymbol{x}$$

Notation in system form:

General form of linear least squares

(**Warning:** change of notation. x is a vector of parameters!)

$$E_{\mathrm{LLS}} = \sum_i \left|\boldsymbol{a}_i \boldsymbol{x} - \boldsymbol{b}_i\right|^2$$

$$= \left\|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\right\|^2 \qquad \text{(matrix form)}$$

This function is quadratic.

How do you find the root of a quadratic?

Solving the linear system

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = \boldsymbol{x}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} - 2\boldsymbol{x}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $\boldsymbol{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$ ⟵

Note: You almost <u>never</u> want to compute the inverse of a matrix.

In Matlab:

```
x = A \ b
```

Linear least squares estimation only works when the transform function is ?

Linear least squares estimation only works when the transform function is **linear! (duh)**

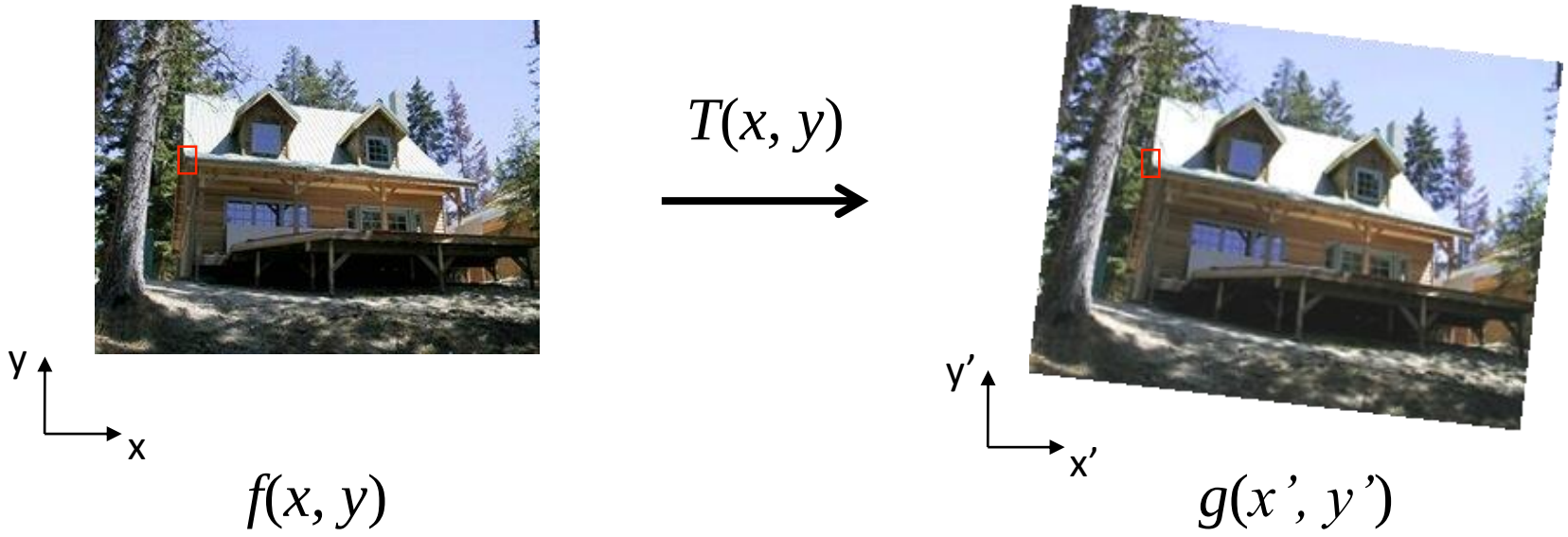
Also doesn’t deal well with outliers

Determining unknown image warps

Determining unknown image warps

Suppose we have two images.

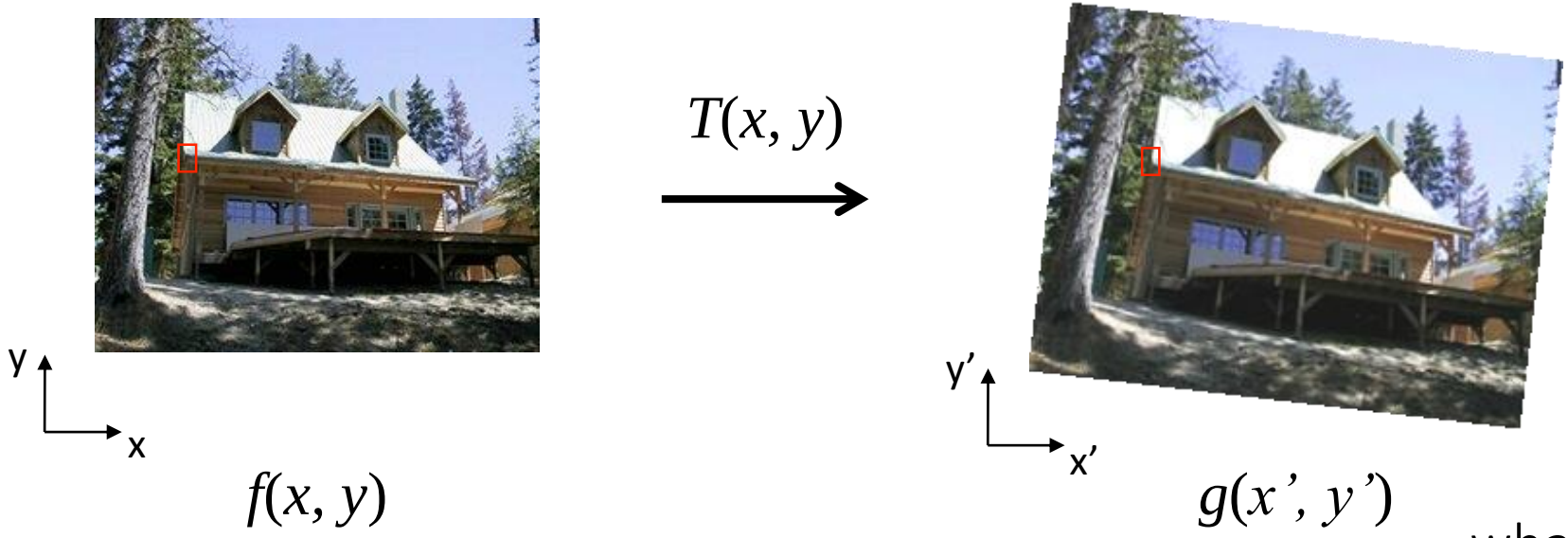
- How do we compute the transform that takes one to the other?



Forward warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?



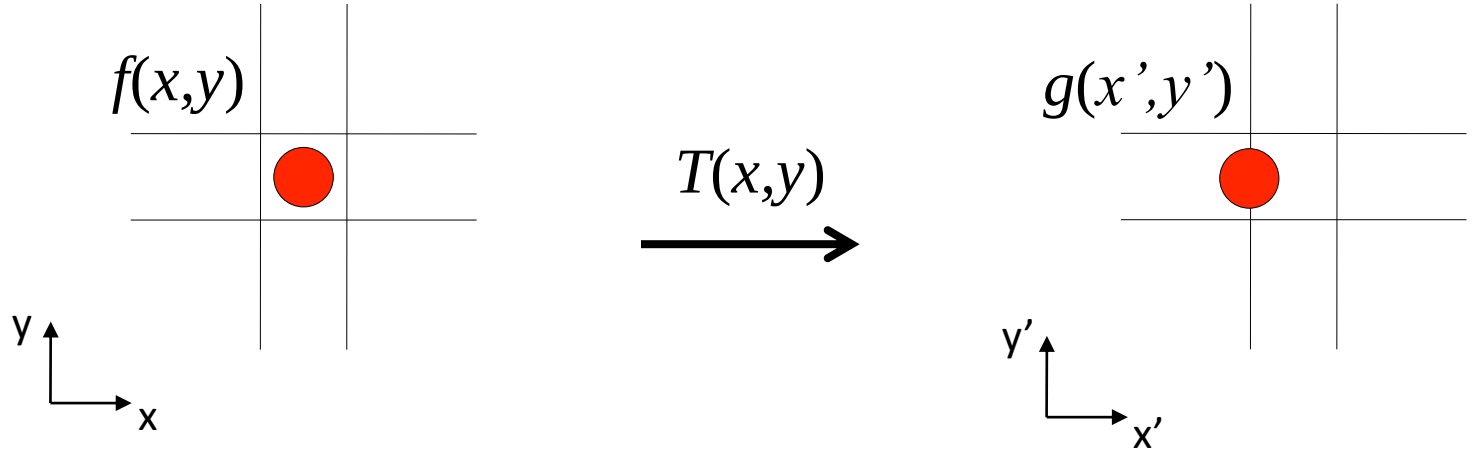
what is the problem with this?

1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before
3. Send intensities $f(x,y)$ in first image to their corresponding location in the second image

Forward warping

Pixels may end up between two points

- How do we determine the intensity of each point?

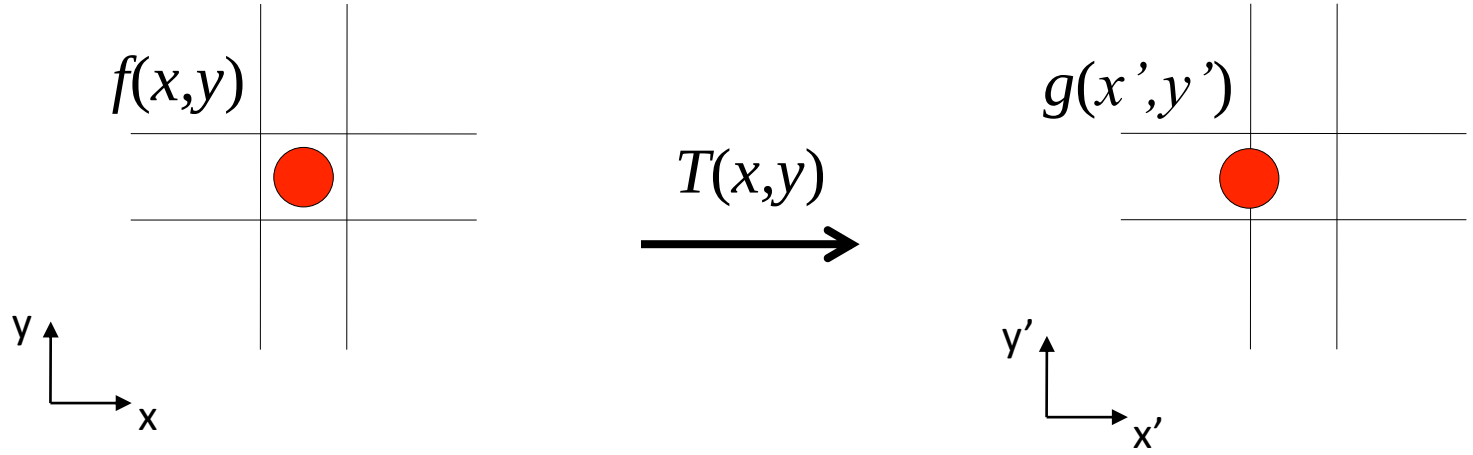


Forward warping

Pixels may end up between two points

- How do we determine the intensity of each point?

✓ We distribute color among neighboring pixels (x',y') ("splatting")



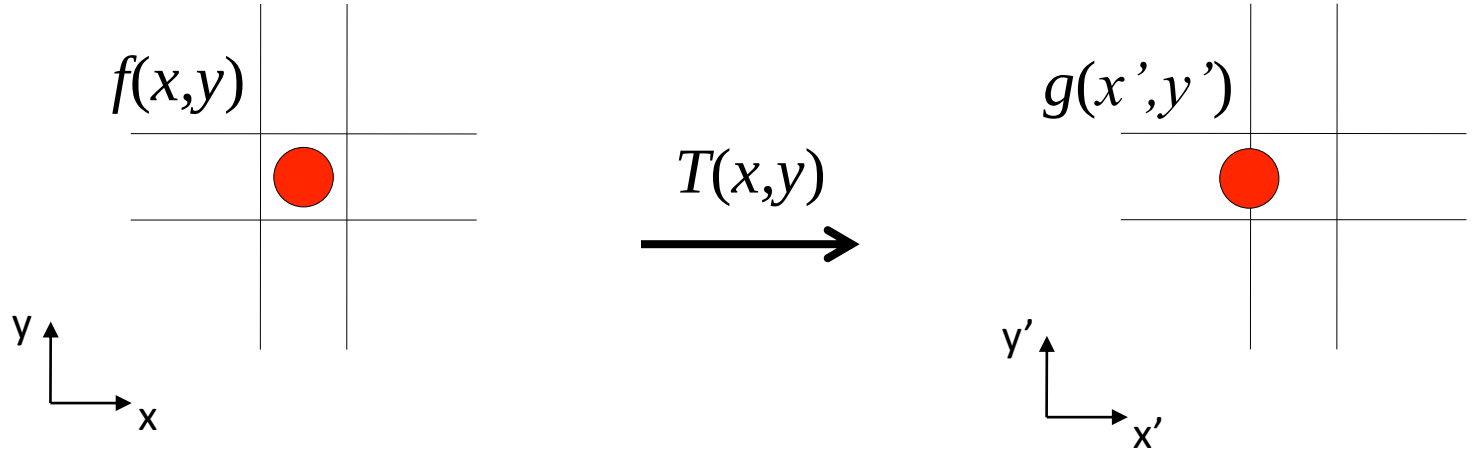
- What if a pixel (x',y') receives intensity from more than one pixels (x,y)?

Forward warping

Pixels may end up between two points

- How do we determine the intensity of each point?

✓ We distribute color among neighboring pixels (x',y') ("splatting")



- What if a pixel (x',y') receives intensity from more than one pixels (x,y)?

✓ We average their intensity contributions.

Inverse warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?

$f(x, y)$

$T^{-1}(x, y)$

$g(x', y')$

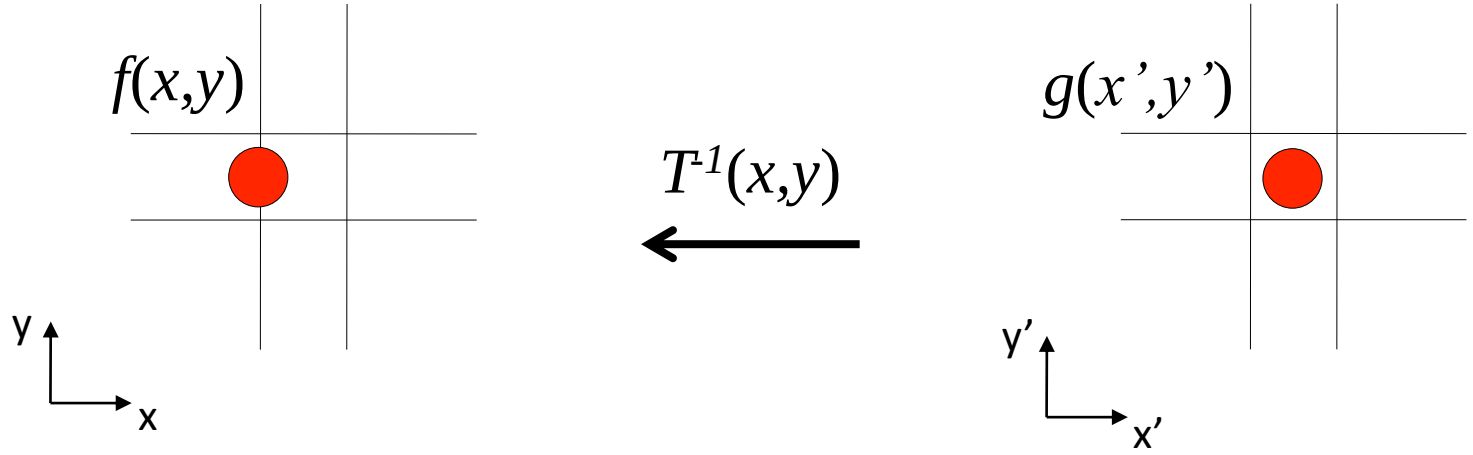
what is the problem with this?

1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before, then compute its inverse
3. Get intensities $g(x',y')$ in in the second image from point $(x,y) = T^{-1}(x',y')$ in first image

Inverse warping

Pixel may come from between two points

- How do we determine its intensity?

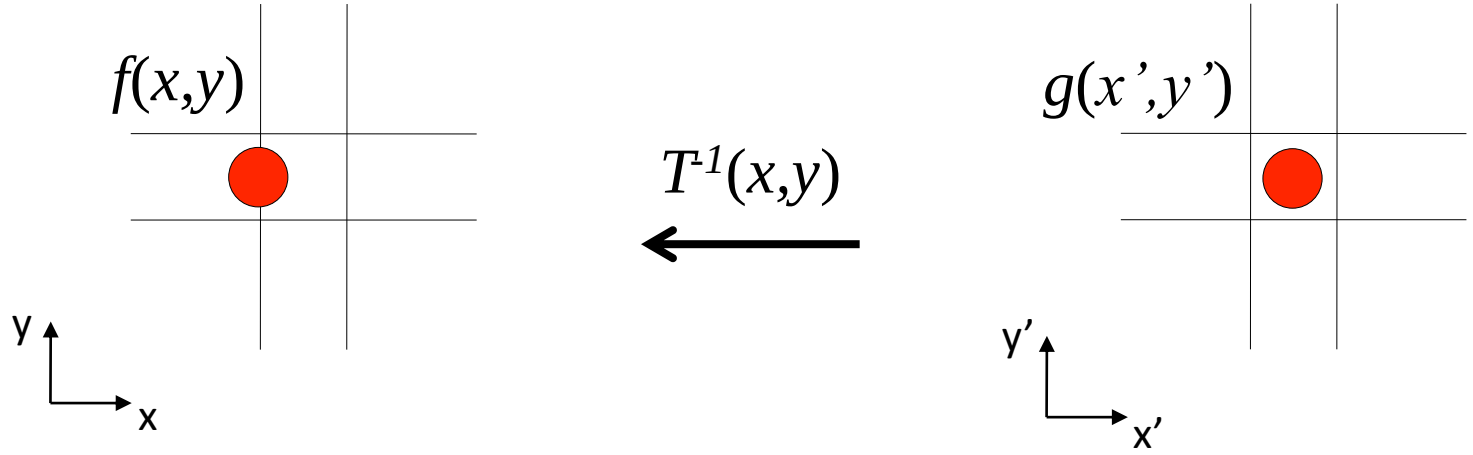


Inverse warping

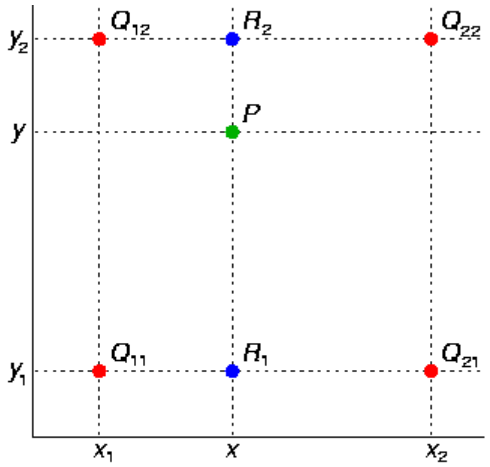
Pixel may come from between two points

- How do we determine its intensity?

✓ Use interpolation

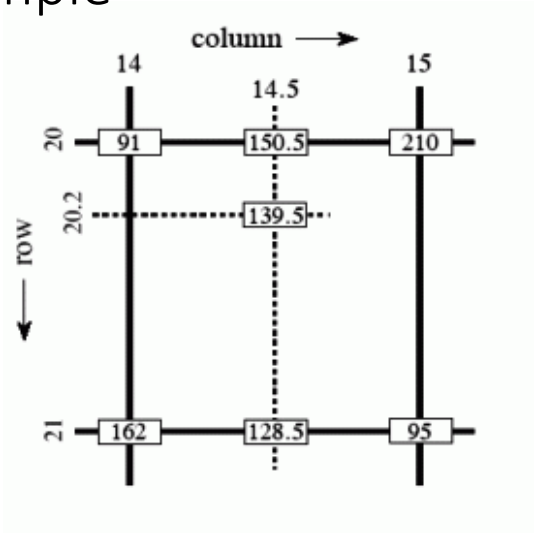


Bilinear interpolation



1. Interpolate to find R2
2. Interpolate to find R1
3. Interpolate to find P

Grayscale example



In matrix form (with adjusted coordinates)

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

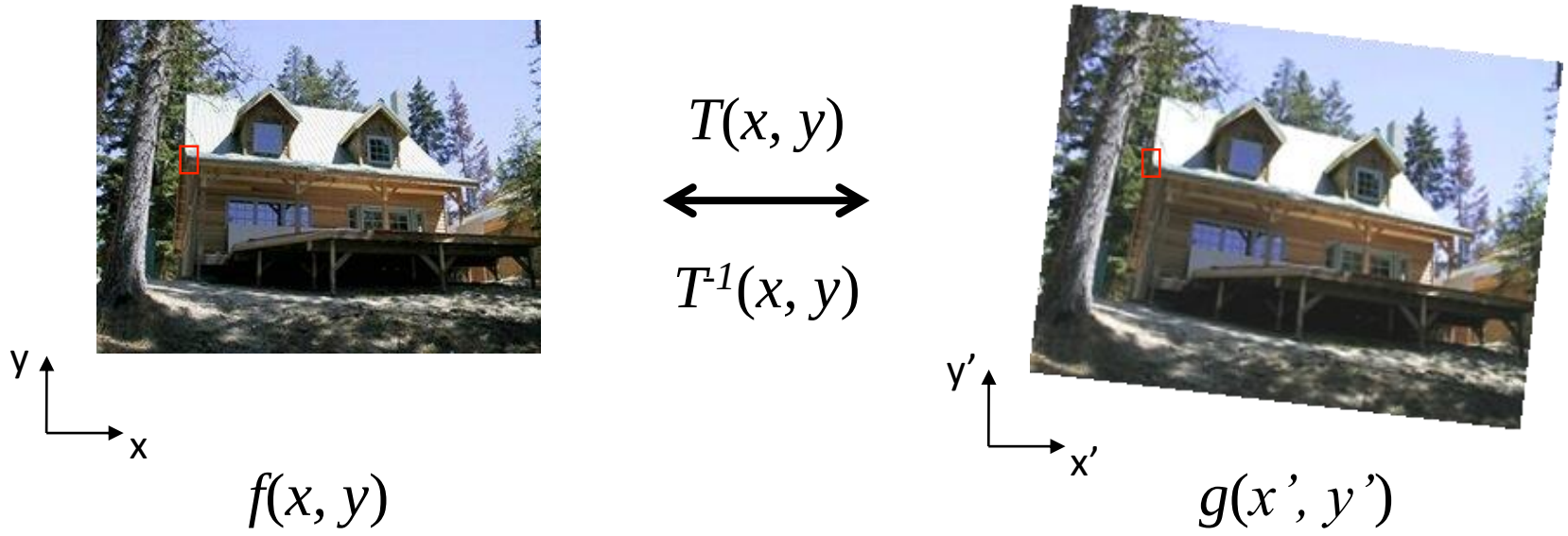
In Matlab:

call `interp2`

Forward vs inverse warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?



- Inverse warping eliminates holes in target image
- Forward warping does not require existence of inverse transform

План

- Ключевые точки (Corners)
- Дескрипторы (Descriptors)
- 2d преобразования (Warping)
- Гомография (Homography)

Motivation for image alignment: panoramas.

How do you create a panorama?

Panorama: an image of (near) 360^{o} field of view.

How do you create a panorama?

Panorama: an image of (near) 360^{o} field of view.

1. Use a very wide-angle lens.

Wide-angle lenses

Fish-eye lens: can produce (near) hemispherical field of view.

What are the pros and cons of this?

How do you create a panorama?

Panorama: an image of (near) 360^{o} field of view.

1. Use a very wide-angle lens.

- Pros: Everything is done optically, single capture.
- Cons: Lens is super expensive and bulky, lots of distortion (can be dealt-with in post).

Any alternative to this?

How do you create a panorama?

Panorama: an image of (near) 360^{o} field of view.

1. Use a very wide-angle lens.
- Pros: Everything is done optically, single capture.
- Cons: Lens is super expensive and bulky, lots of distortion (can be dealt-with in post).

2. Capture multiple images and combine them.

Panoramas from image stitching

1. Capture multiple images from different viewpoints.

2. Stitch them together into a virtual wide-angle image.

How do we stitch images from different viewpoints?

Will standard stitching work?

1. Translate one image relative to another.
2. (Optionally) find an optimal seam.

How do we stitch images from different viewpoints?

Will standard stitching work?

1. Translate one image relative to another.
2. (Optionally) find an optimal seam.

left on top

right on top

Translation-only stitching is not enough to mosaic these images.

How do we stitch images from different viewpoints?

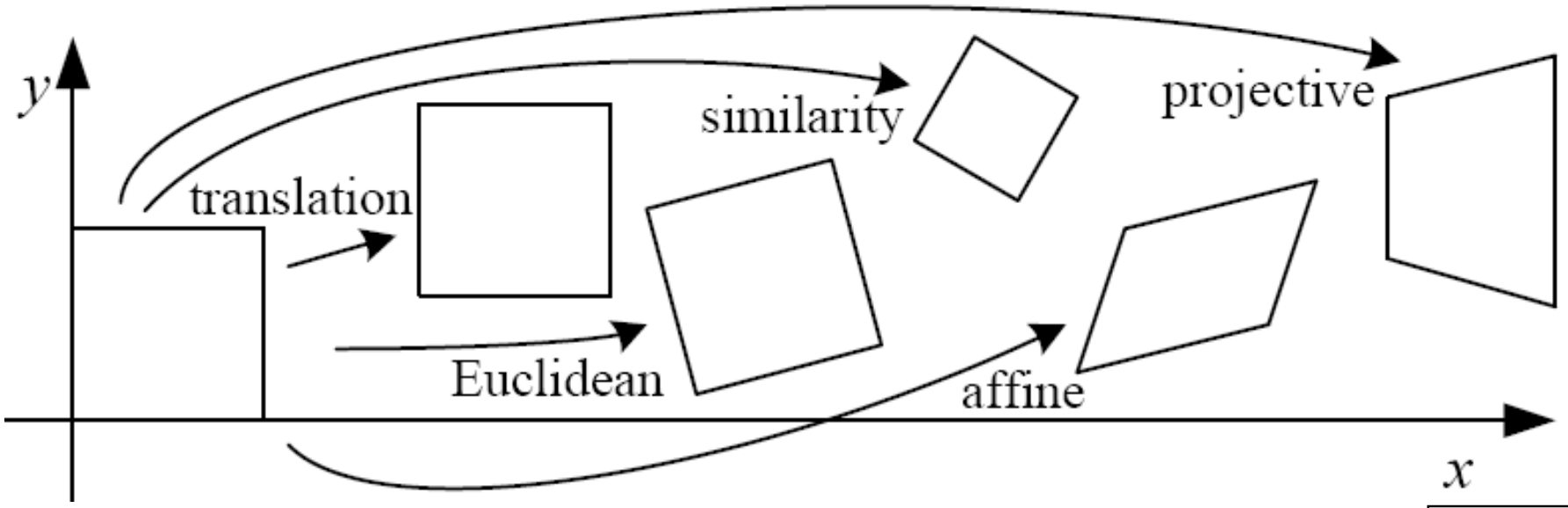
What else can we try?

How do we stitch images from different viewpoints?

Use image homographies.

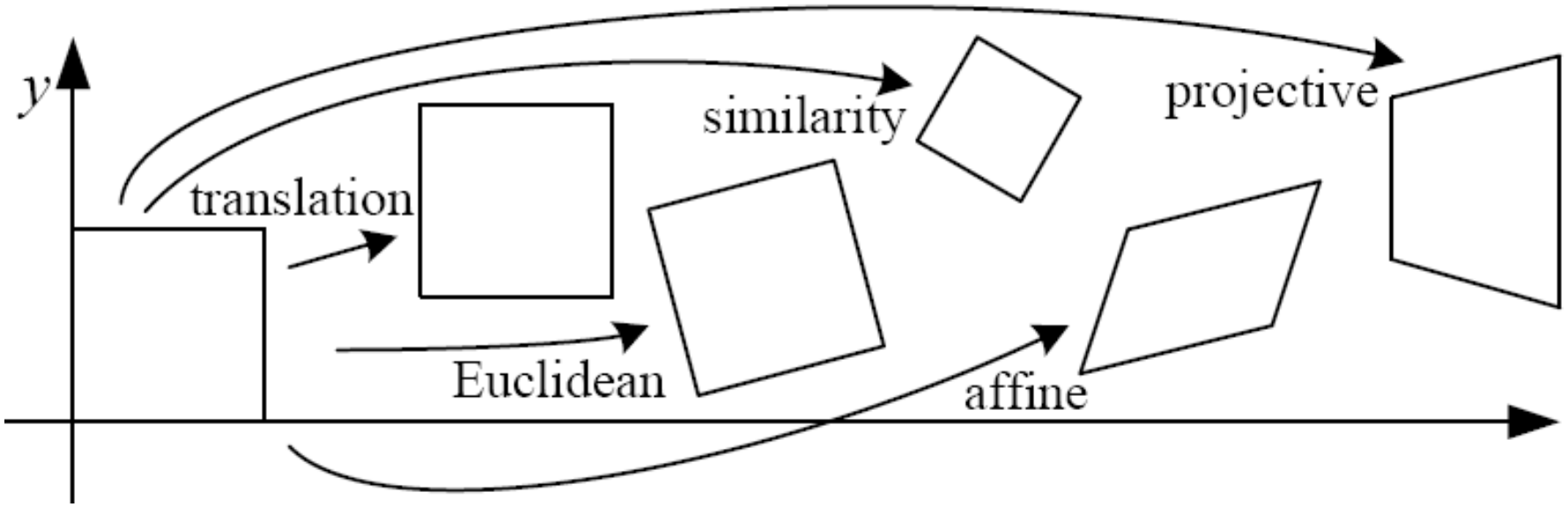
Back to warping: image homographies

Classification of 2D transformations

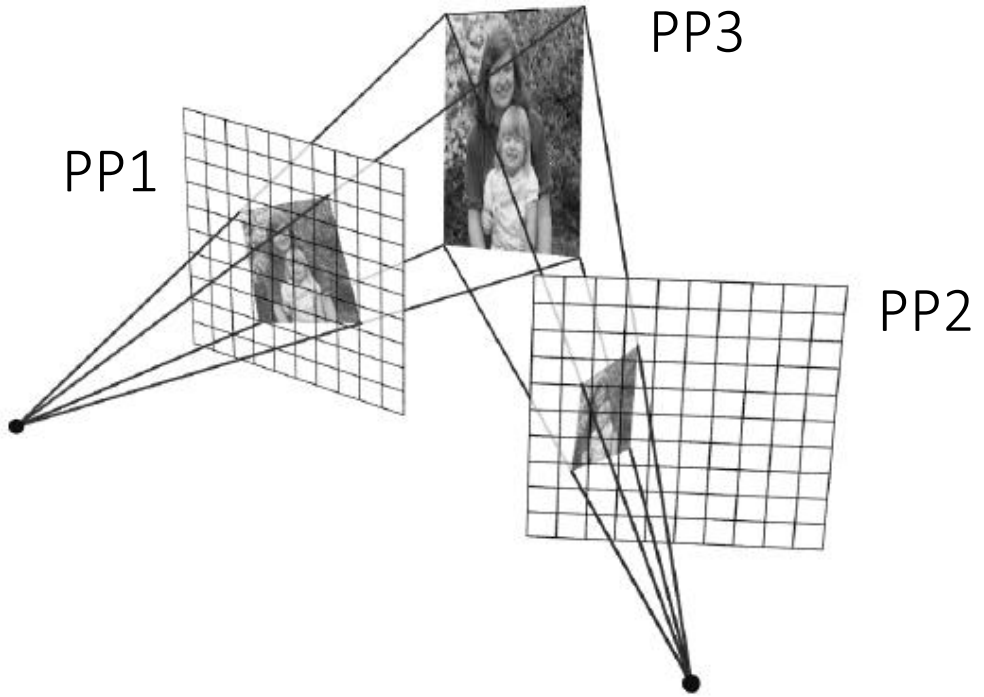


Name	Matrix	# D.O.F.
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4
affine	$\left[\boldsymbol{A}\right]_{2\times 3}$	6
projective	$\left[\tilde{\boldsymbol{H}}\right]_{3\times 3}$	8

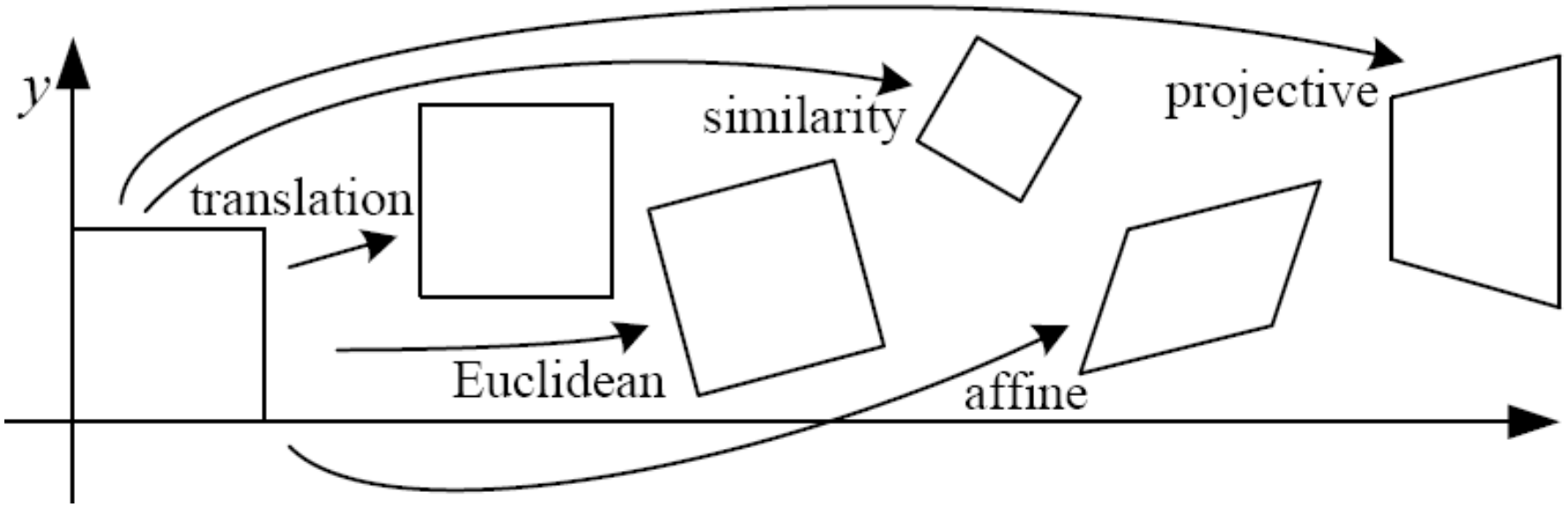
Classification of 2D transformations

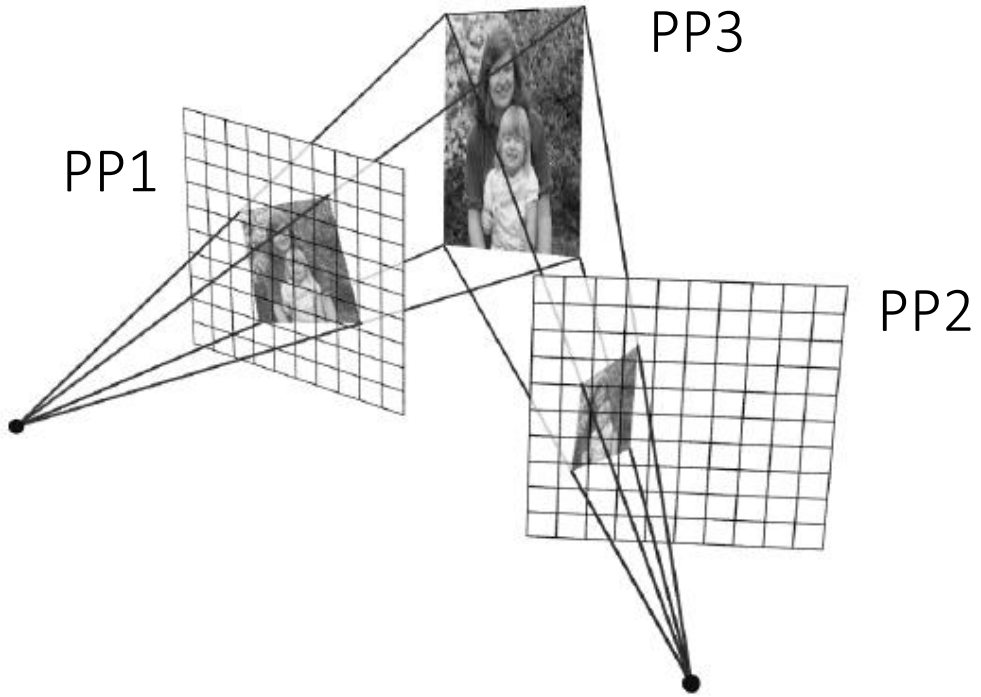


Which kind transformation is needed to warp projective plane 1 into projective plane 2?



Classification of 2D transformations





Which kind transformation is needed to warp projective plane 1 into projective plane 2?

- A projective transformation (a.k.a. a homography).

Image rectification

two original images

rectified and stitched

Street art

A weird drawing

Holbein, “The Ambassadors”

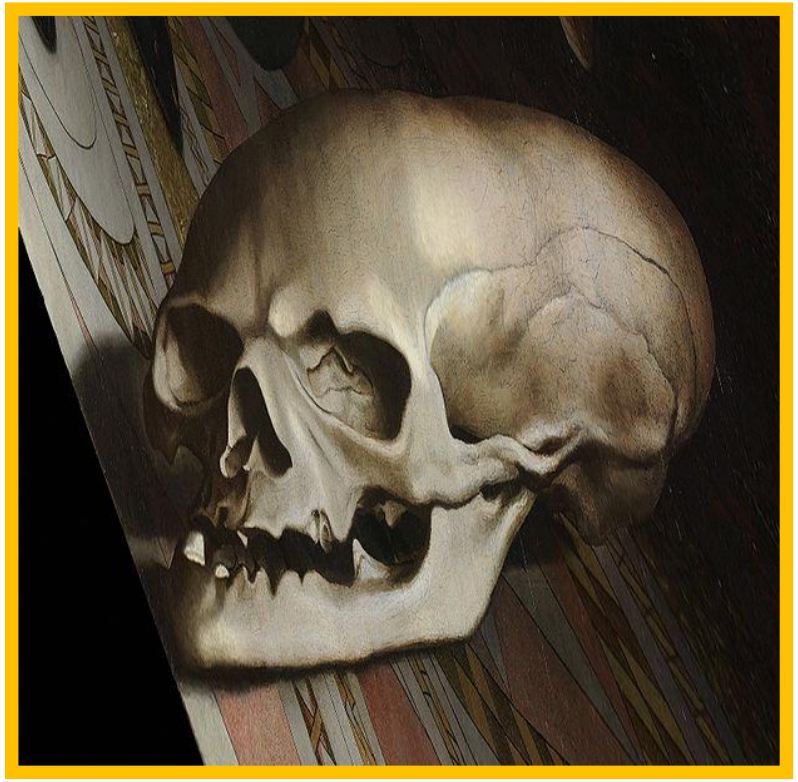
A weird drawing

Holbein, “The Ambassadors”

What’s this???

A weird drawing

Holbein, “The Ambassadors”



rectified view

skull under anamorphic perspective

A weird drawing

Holbein, “The Ambassadors”

DIY: use a polished spoon to see the skull

Panoramas from image stitching

1. Capture multiple images from different viewpoints.

2. Stitch them together into a virtual wide-angle image.

When can we use homographies?

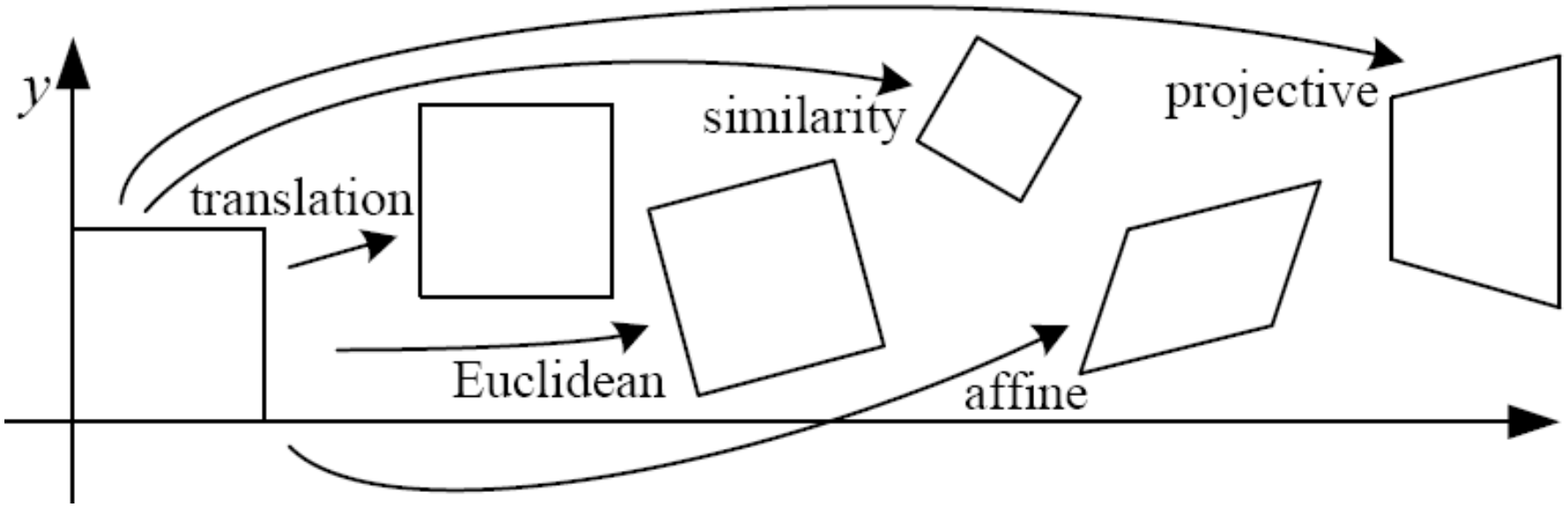
We can use homographies when…

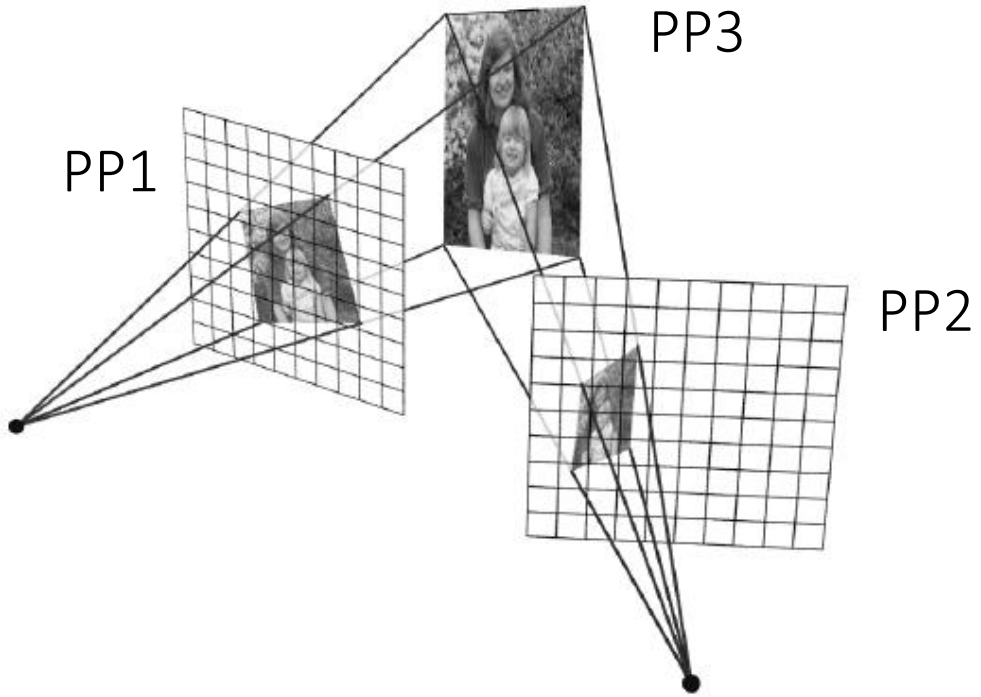
1. … the scene is planar; or

2. … the scene is very far or has small (relative) depth variation
   → scene is approximately planar

Computing with homographies

Classification of 2D transformations





Which kind transformation is needed to warp projective plane 1 into projective plane 2?

- A projective transformation (a.k.a. a homography).

Applying a homography

1. Convert to homogeneous coordinates: $p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

What is the size of the homography matrix?

2. Multiply by the homography matrix: $P' = H \cdot P$

3. Convert back to heterogeneous coordinates: $P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$

Applying a homography

1. Convert to homogeneous coordinates: $p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

What is the size of the homography matrix? Answer: 3 x 3

2. Multiply by the homography matrix: $P' = H \cdot P$

How many degrees of freedom does the homography matrix have?

3. Convert back to heterogeneous coordinates: $P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$

Applying a homography

1. Convert to homogeneous coordinates: $p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

What is the size of the homography matrix? Answer: 3 x 3

2. Multiply by the homography matrix: $P' = H \cdot P$

How many degrees of freedom does the homography matrix have? Answer: 8

3. Convert back to heterogeneous coordinates: $P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$

Applying a homography

What is the size of the homography matrix? Answer: 3 x 3

$$P' = H \cdot P$$

How many degrees of freedom does the homography matrix have? Answer: 8

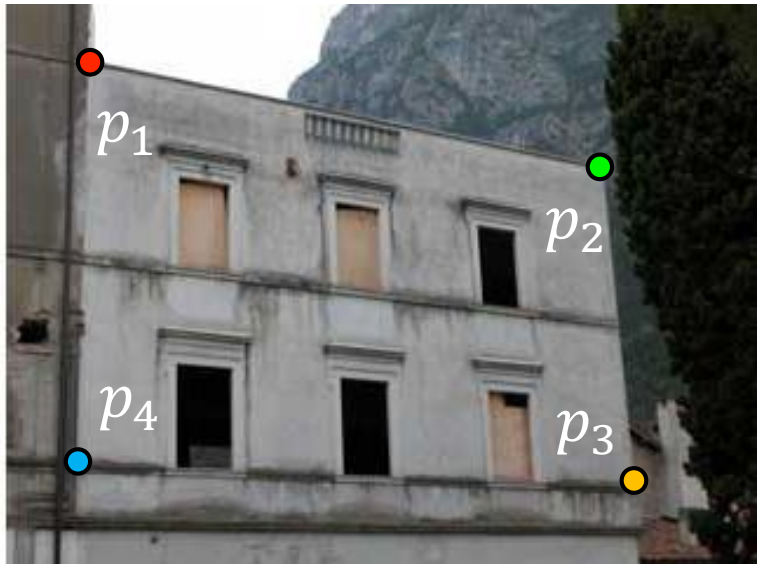
How do we compute the homography matrix?

The direct linear transform (DLT)

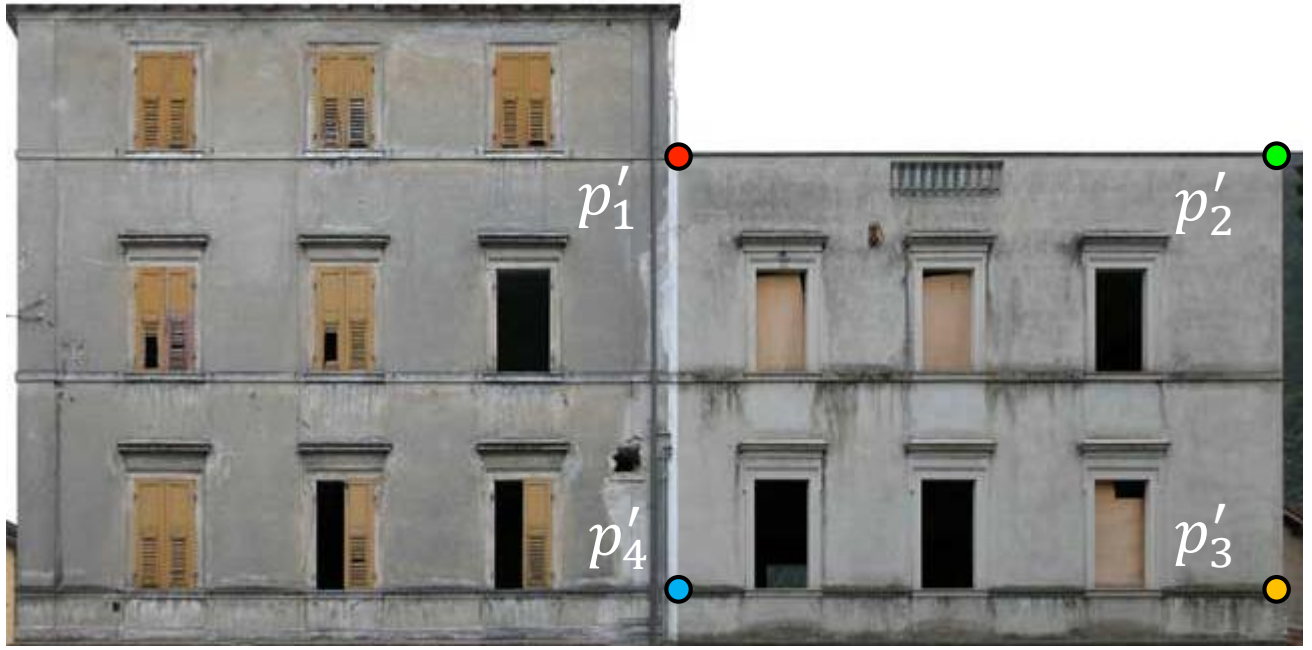
Create point correspondences

Given a set of matched feature points $\{p_i, p_i'\}$ find the best estimate of H such that

$$P' = H \cdot P$$



original image



target image

How many correspondences do we need?

Determining the homography matrix

Write out linear equation for each correspondence:

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$\begin{aligned} x' &= \alpha(h_1 x + h_2 y + h_3) \\ y' &= \alpha(h_4 x + h_5 y + h_6) \\ 1 &= \alpha(h_7 x + h_8 y + h_9) \end{aligned}$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$\begin{aligned} x' &= \alpha(h_1 x + h_2 y + h_3) \\ y' &= \alpha(h_4 x + h_5 y + h_6) \\ 1 &= \alpha(h_7 x + h_8 y + h_9) \end{aligned}$$

Divide out unknown scale factor:

$$\begin{aligned} x'(h_7 x + h_8 y + h_9) &= (h_1 x + h_2 y + h_3) \\ y'(h_7 x + h_8 y + h_9) &= (h_4 x + h_5 y + h_6) \end{aligned}$$

How do you rearrange terms to make it a linear system?

$$x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3)$$
$$y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6)$$

Just rearrange the terms

$$h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0$$
$$h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0$$

Determining the homography matrix

Re-arrange terms:

$$h_7 x x' + h_8 y x' + h_9 x' - h_1 x - h_2 y - h_3 = 0$$
$$h_7 x y' + h_8 y y' + h_9 y' - h_4 x - h_5 y - h_6 = 0$$

Re-write in matrix form:

$$\mathbf{A}_i \boldsymbol{h} = \mathbf{0}$$

How many equations from one point correspondence?

$$\mathbf{A}_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\boldsymbol{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^\top$$

Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

Reminder: Determining affine transformations

Affine transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Vectorize transformation parameters:

Stack equations from point correspondences:

$$\underbrace{\begin{bmatrix} x' \\ y' \\ x' \\ y' \\ \vdots \\ x' \\ y' \end{bmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ & & & \vdots & & \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}}_{\boldsymbol{x}}$$

Notation in system form:

$$\boxed{Ax = b}$$

Reminder: Determining affine transformations

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = \boldsymbol{x}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} - 2\boldsymbol{x}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $\boldsymbol{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$ ⟵

In Matlab:

```
x = A \ b
```

Note: You almost <u>never</u> want to compute the inverse of a matrix.

Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

- How do we solve this?

Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

- Solve with SVD

Solving for H using DLT

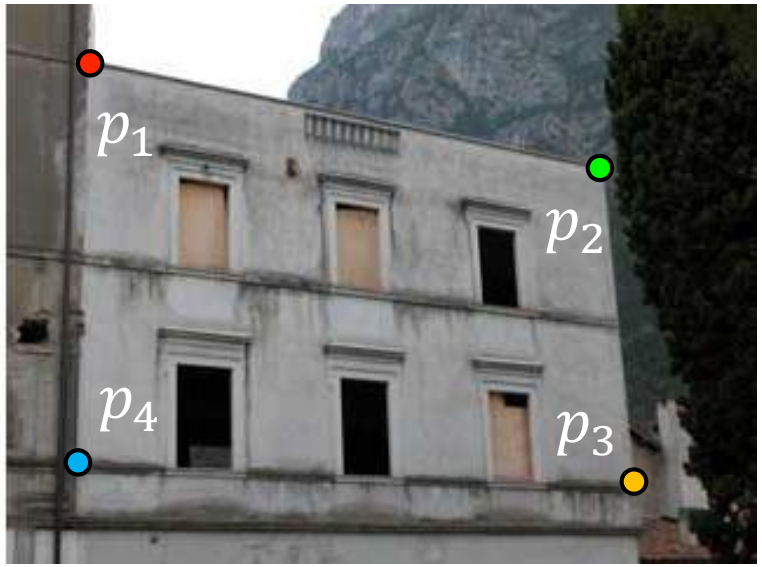
Given $\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$ solve for H such that $\boldsymbol{x}' = \mathbf{H}\boldsymbol{x}$

1. For each correspondence, create 2x9 matrix $\mathbf{A}_i$
2. Concatenate into single 2n x 9 matrix $\mathbf{A}$
3. Compute SVD of $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$
4. Store singular vector of the smallest singular value $\boldsymbol{h} = \boldsymbol{v}_{\hat{i}}$
5. Reshape to get $\mathbf{H}$

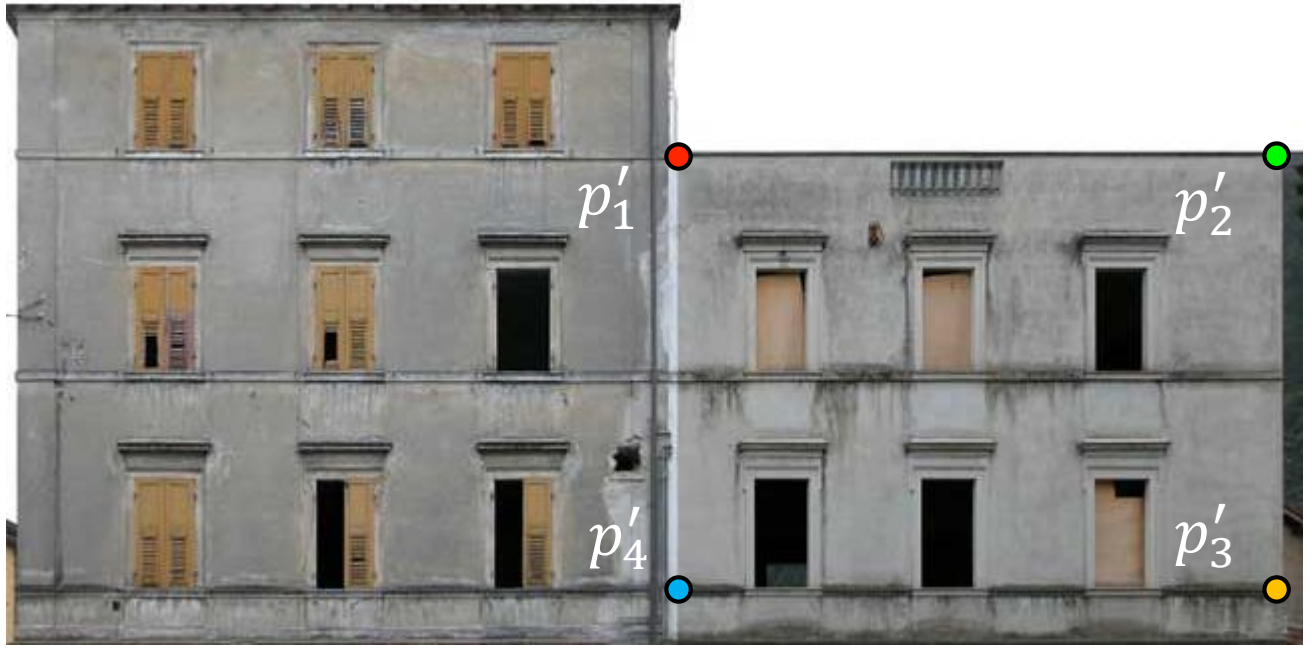
Linear least squares estimation only works when the transform function is **linear! (duh)**

Also doesn’t deal well with **outliers.**

Create point correspondences



original image



target image

How do we automate this step?

The image correspondence pipeline

1. Feature point detection
   - Detect corners using the Harris corner detector.

2. Feature point description
   - Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching

The image correspondence pipeline

1. Feature point detection
   - Detect corners using the Harris corner detector.

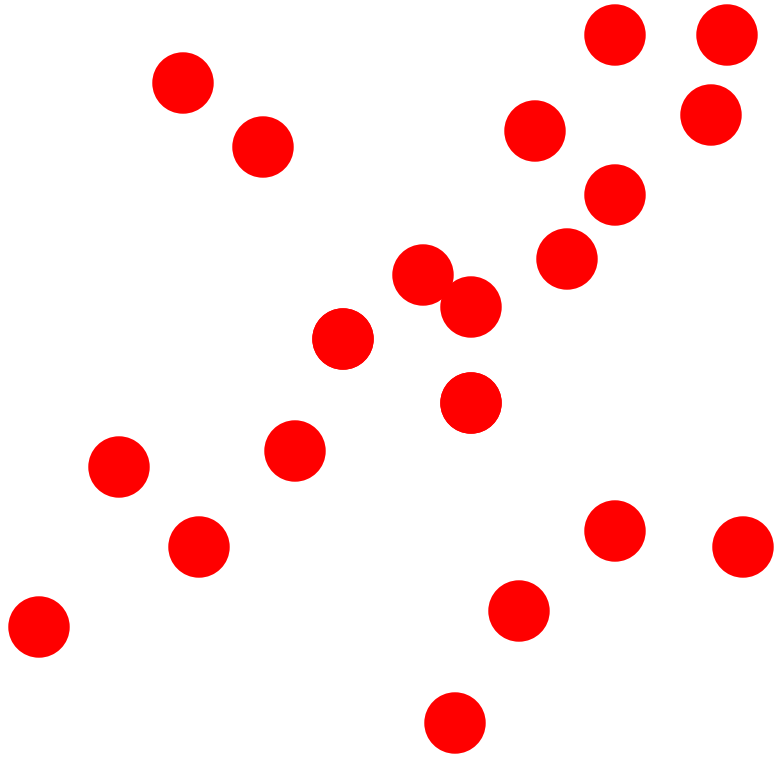
2. Feature point description
   - Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching


bad correspondence
good correspondence

Random Sample Consensus (RANSAC)

Fitting lines
(with outliers)

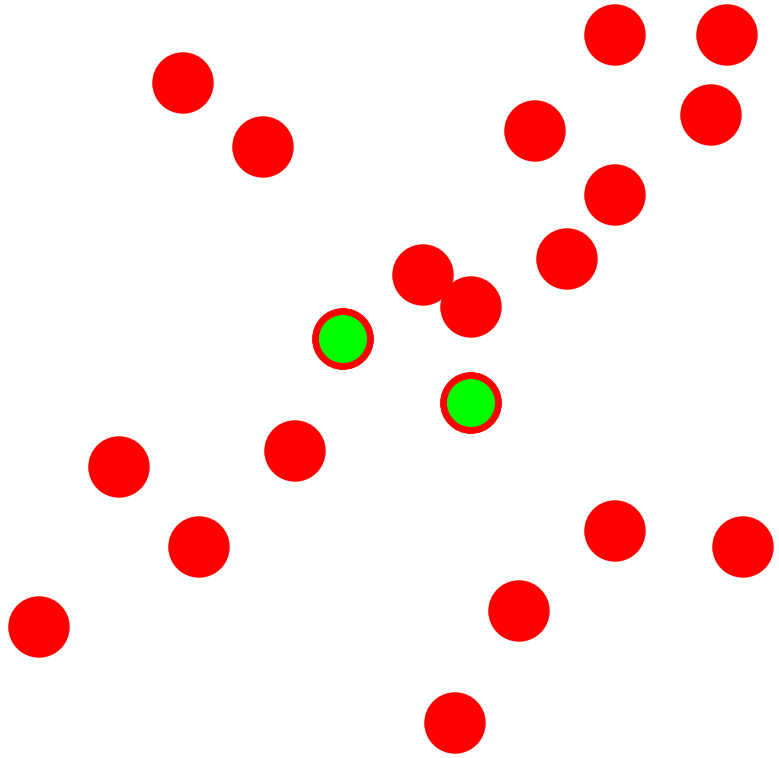


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

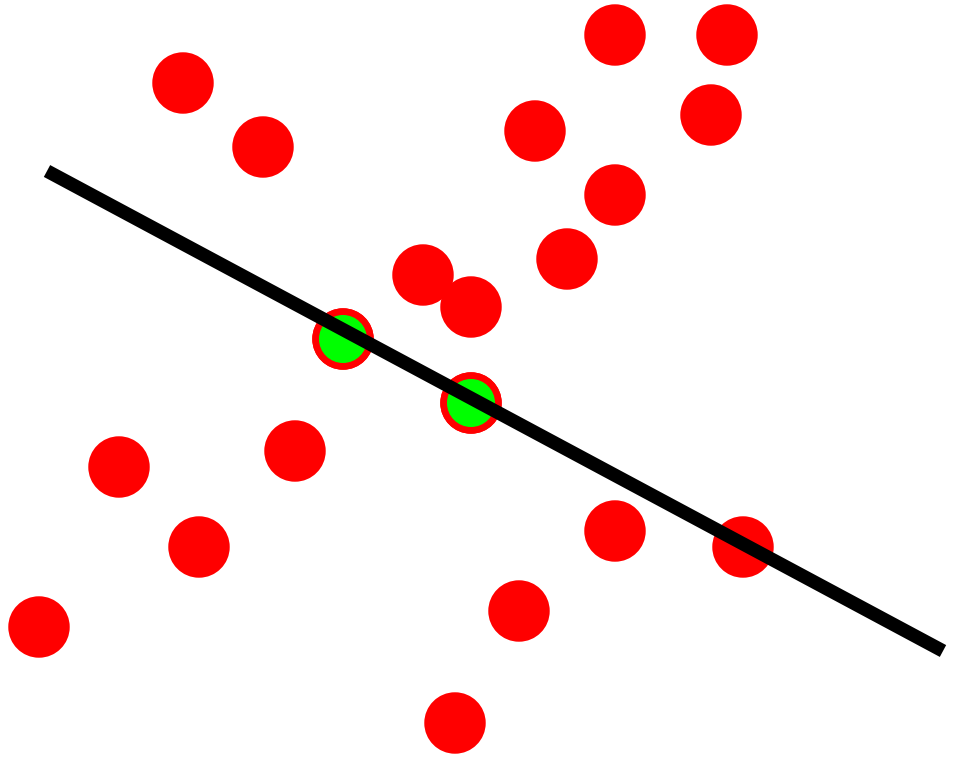


Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



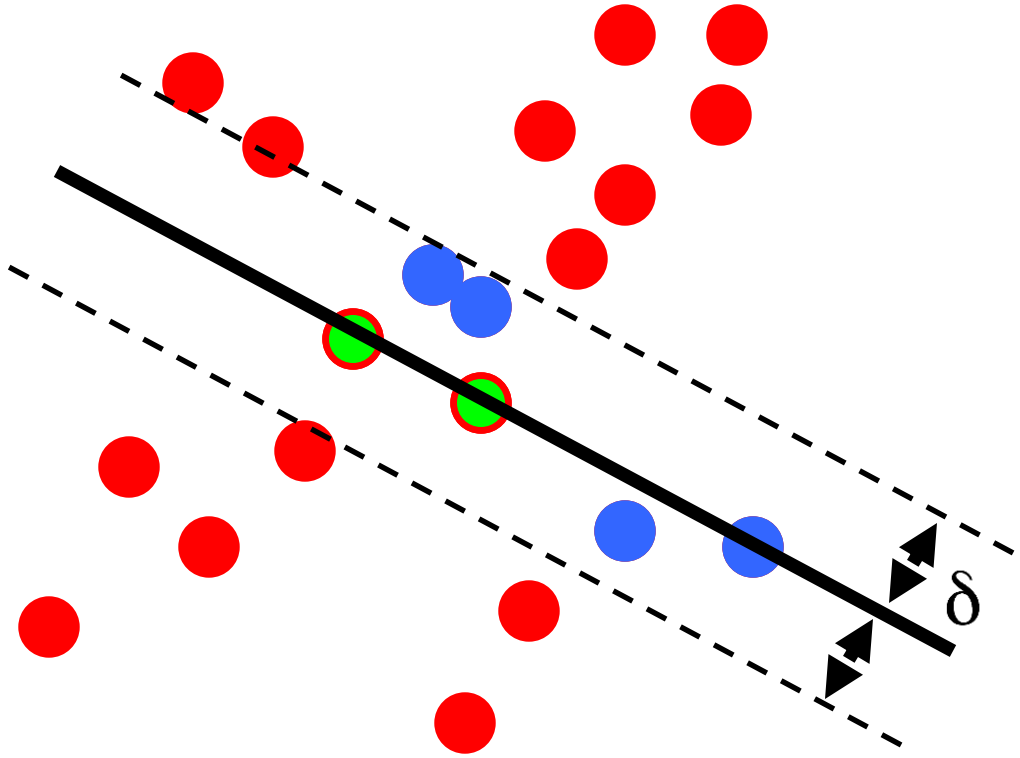
Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

$N_I = 6$

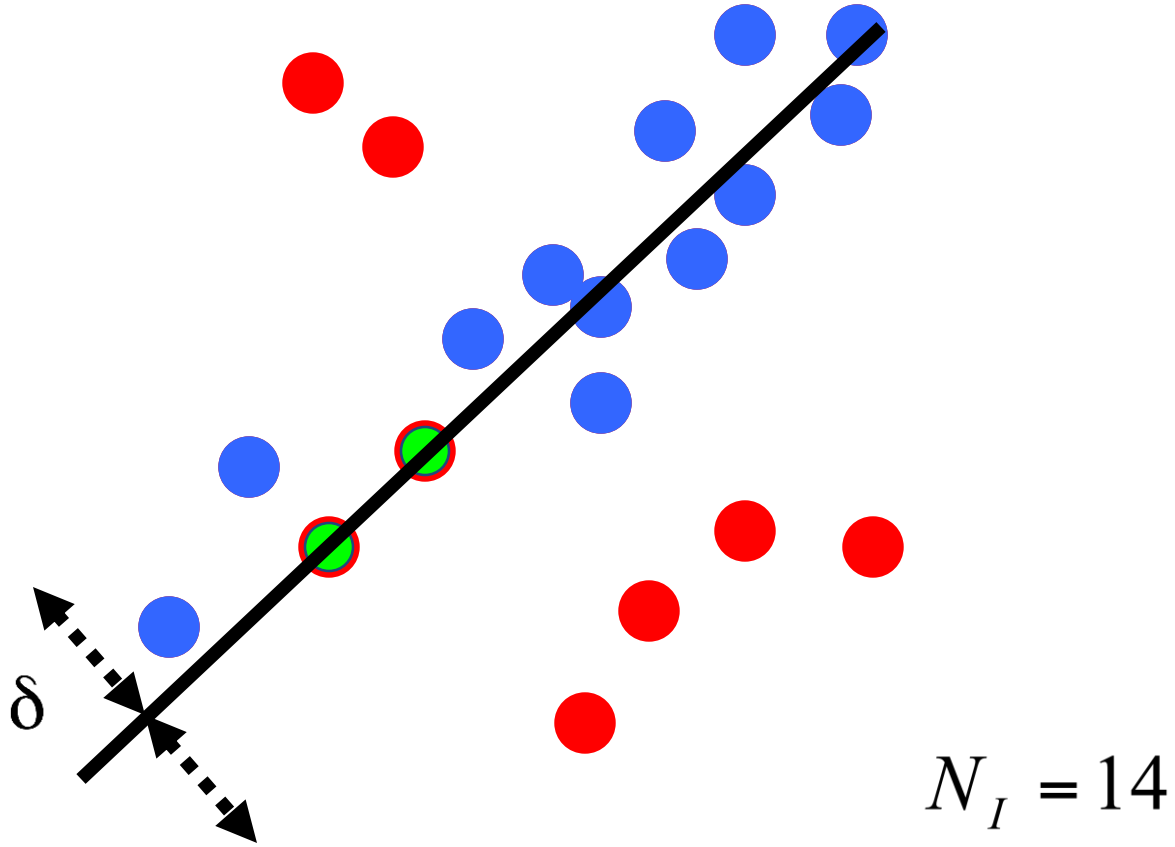


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e)
- Number of sampled points s
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \frac{\log(1-p)}{\log\left(1-(1-e)^s\right)}$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

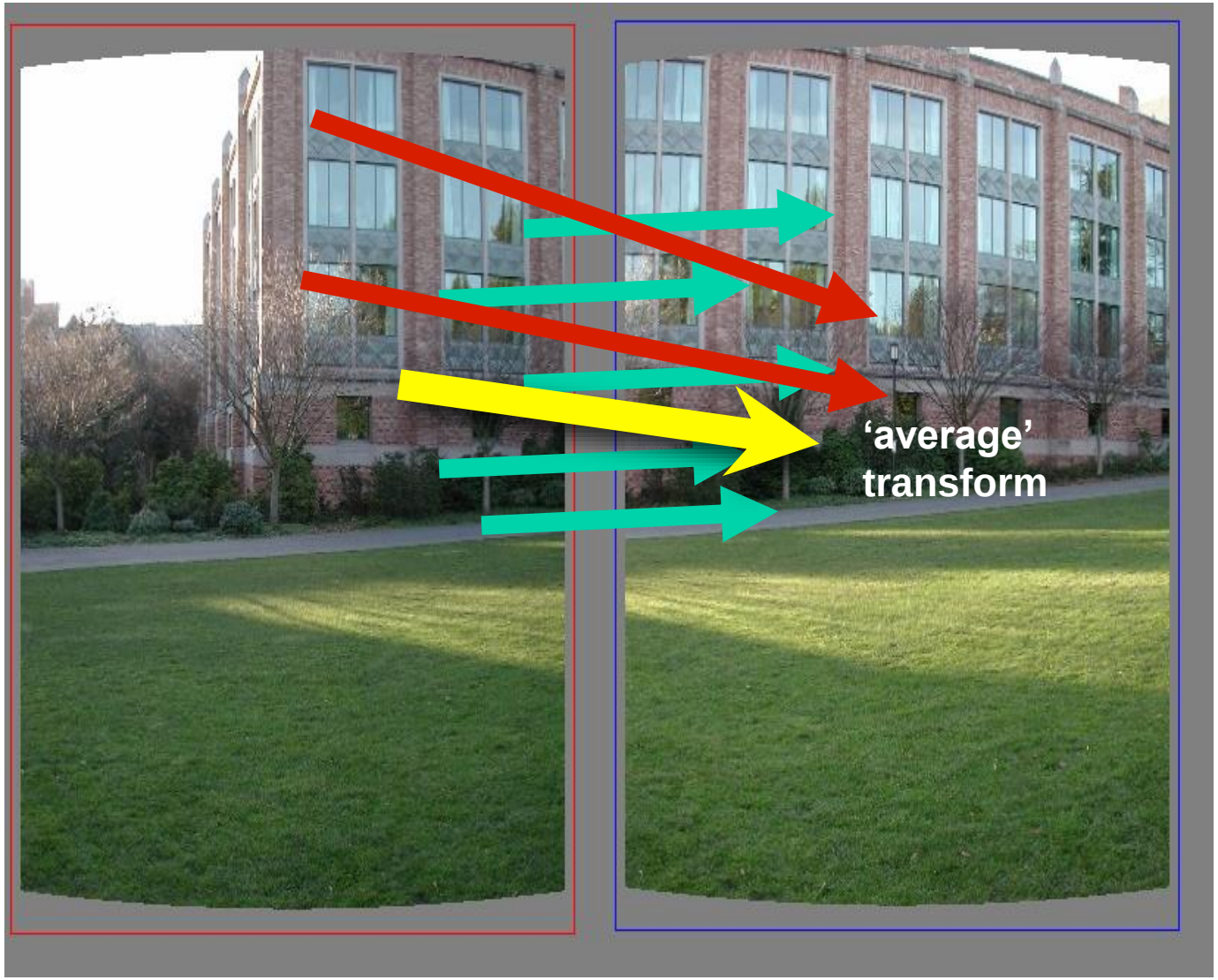
Given two images…

find matching features (e.g., SIFT) and a translation transform

Matched points will usually contain bad correspondences

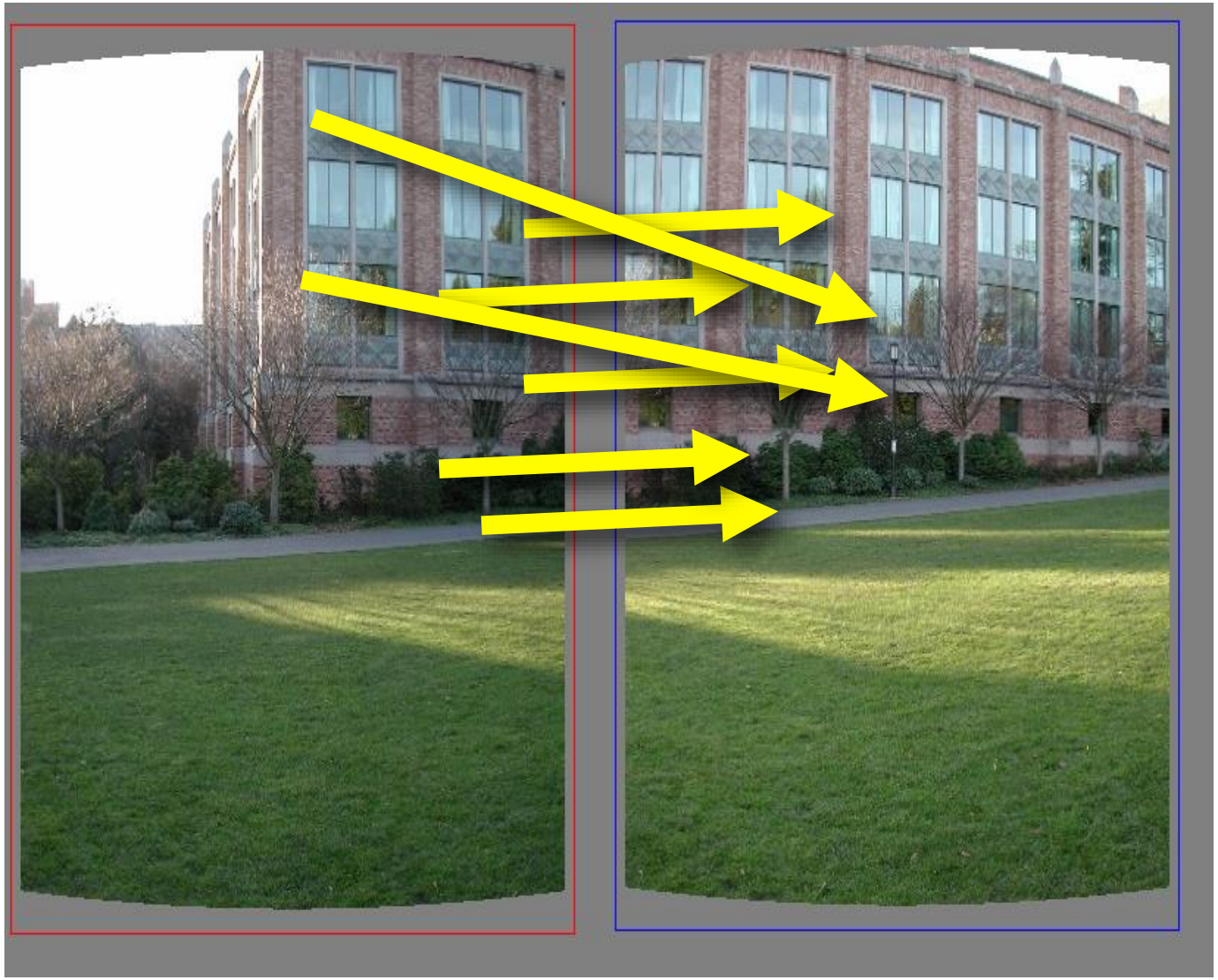
how should we estimate the transform?

LLS will find the ‘average’ transform

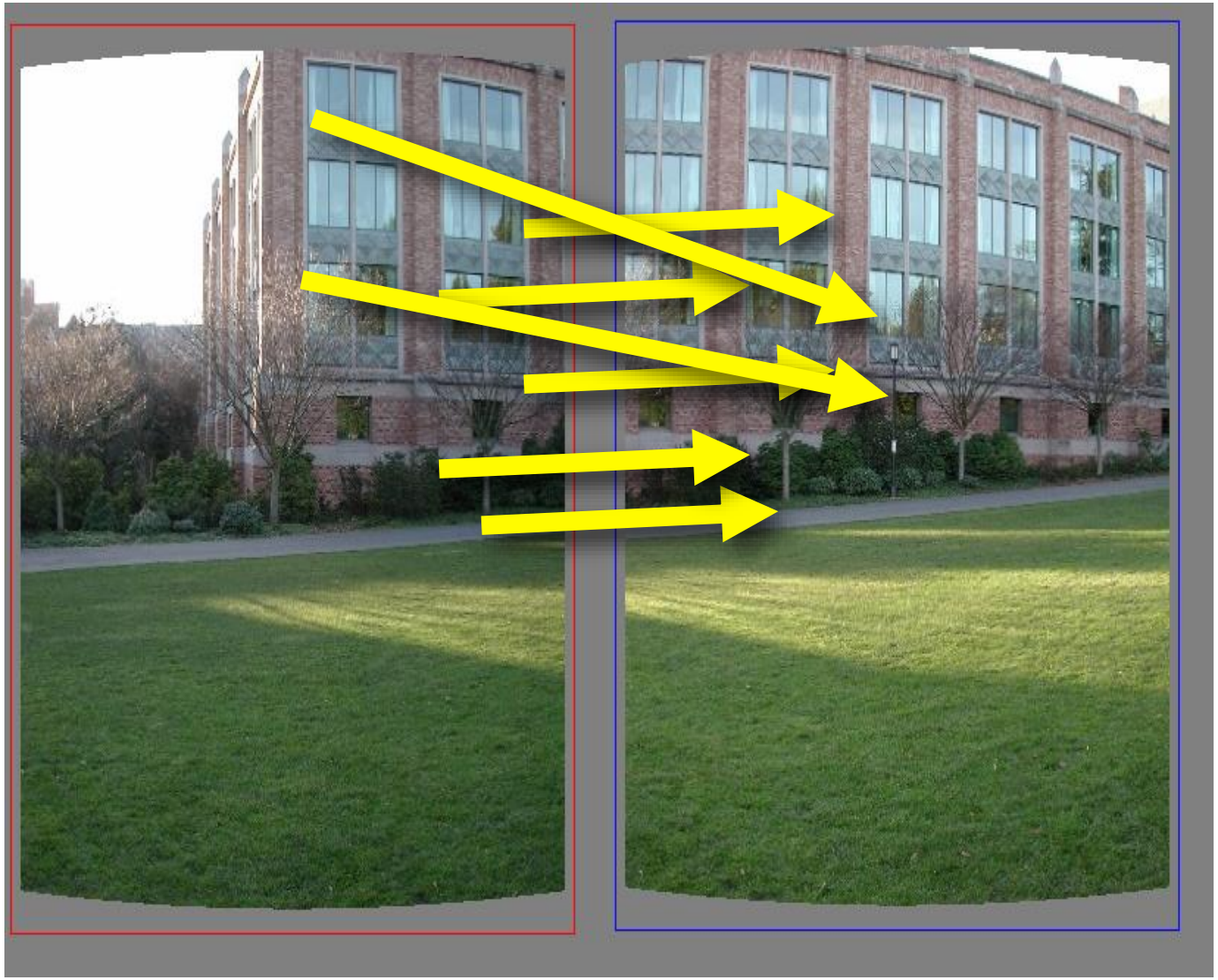


solution is corrupted by bad correspondences

Use RANSAC

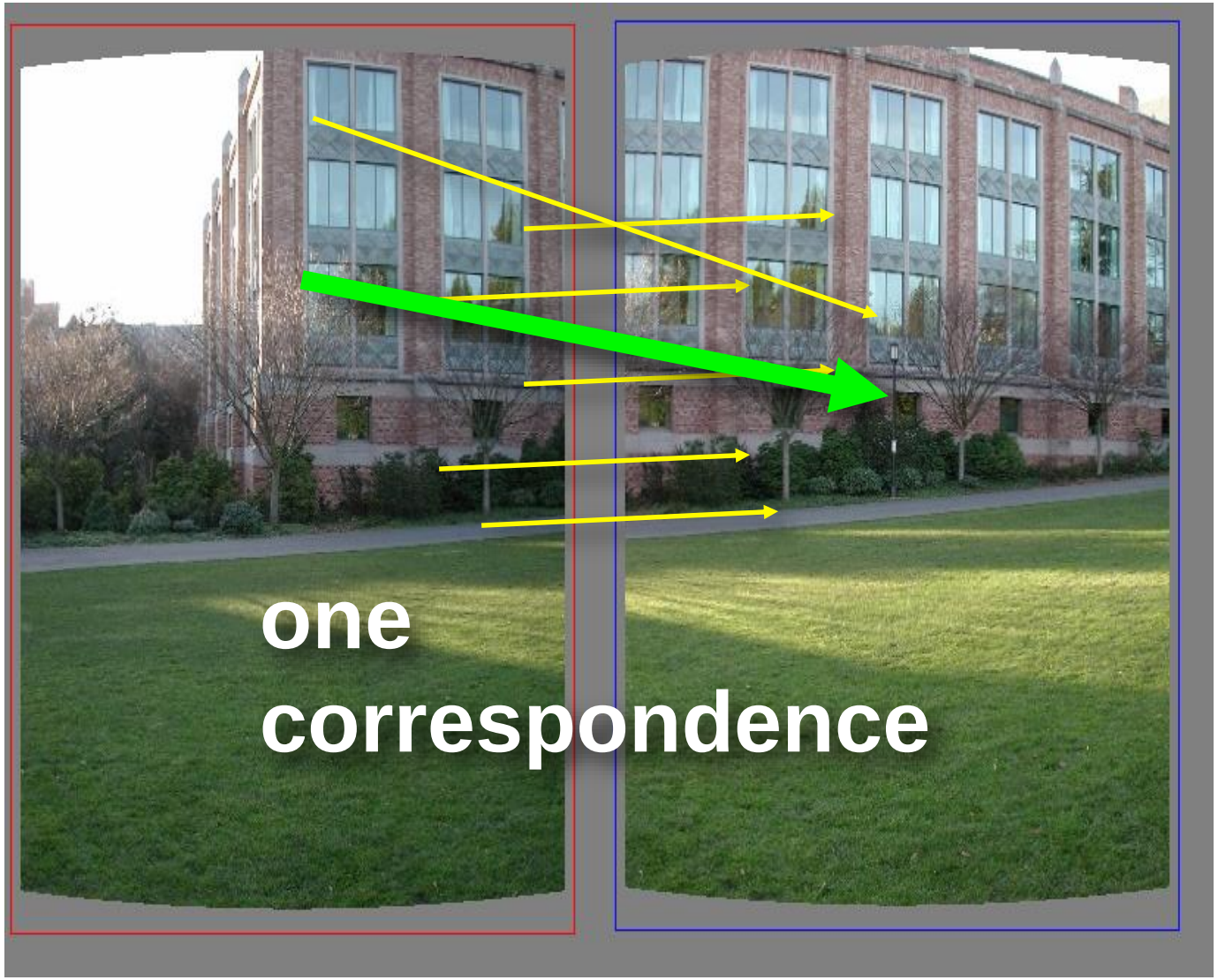


How many correspondences to compute translation transform?

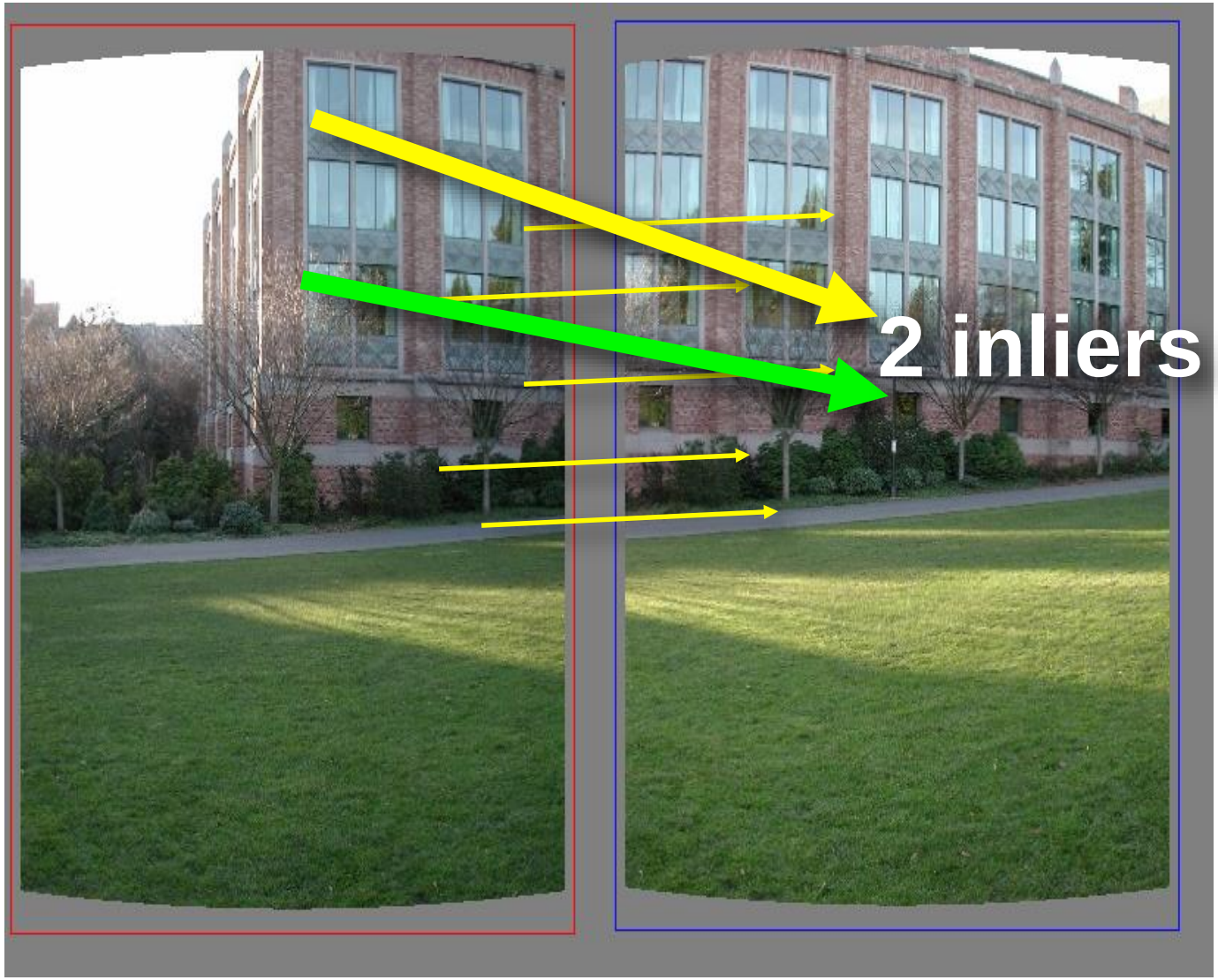


Need only **one correspondence**, to find translation model

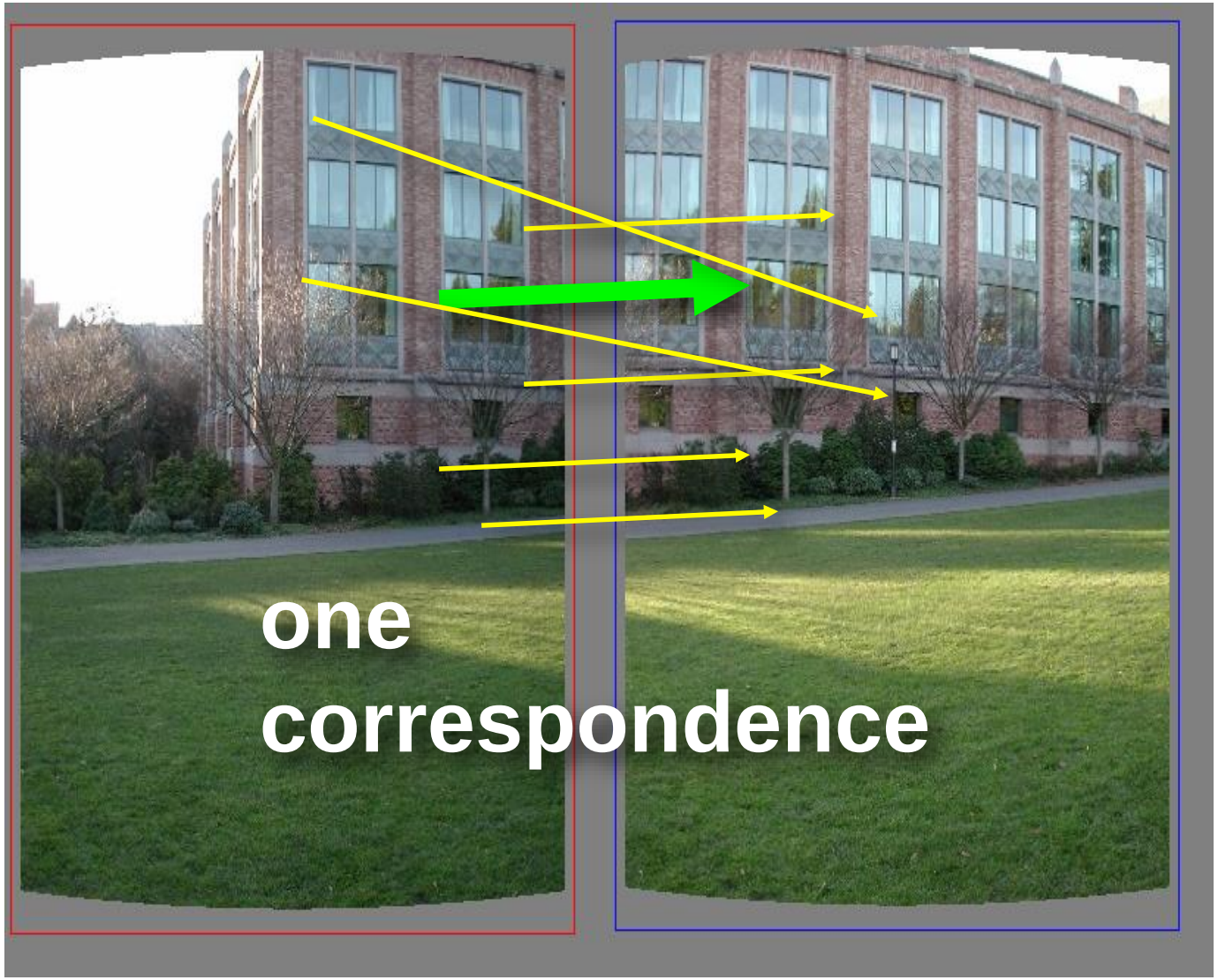
Pick one correspondence, count inliers



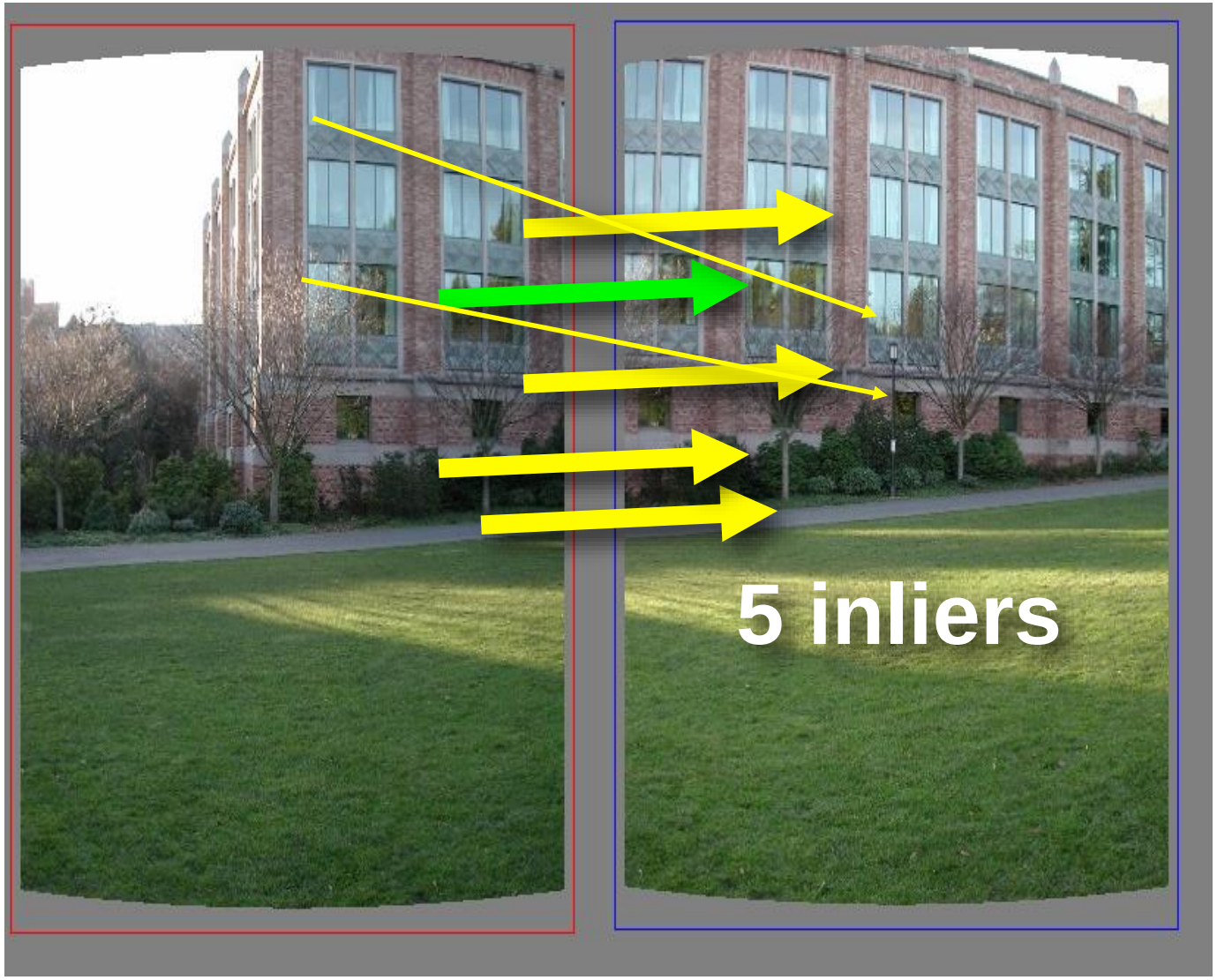
Pick one correspondence, count inliers



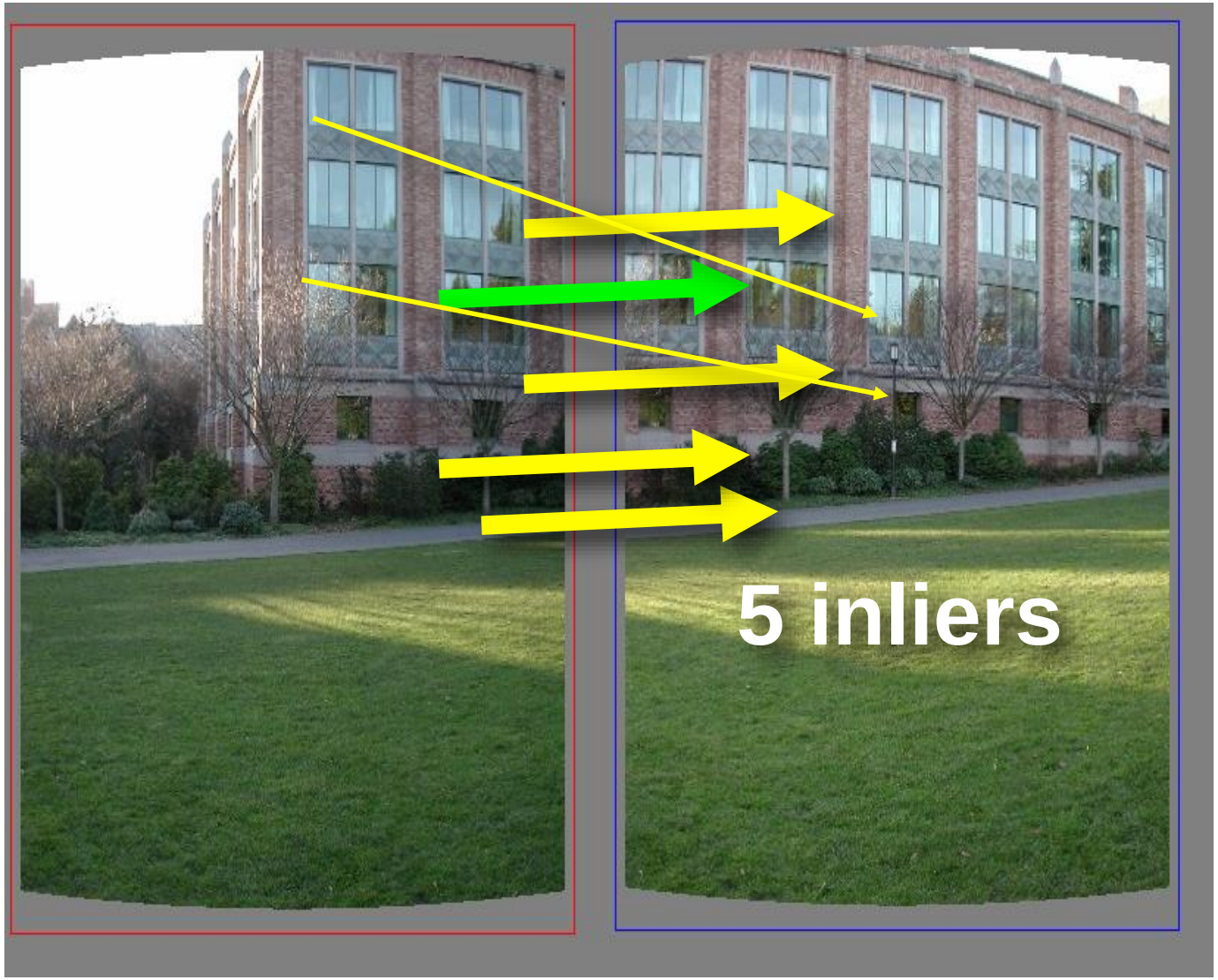
Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick the model with the highest number of inliers!

Estimating homography using RANSAC

- RANSAC loop
  1. Get [redacted] point correspondences (randomly)

Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using

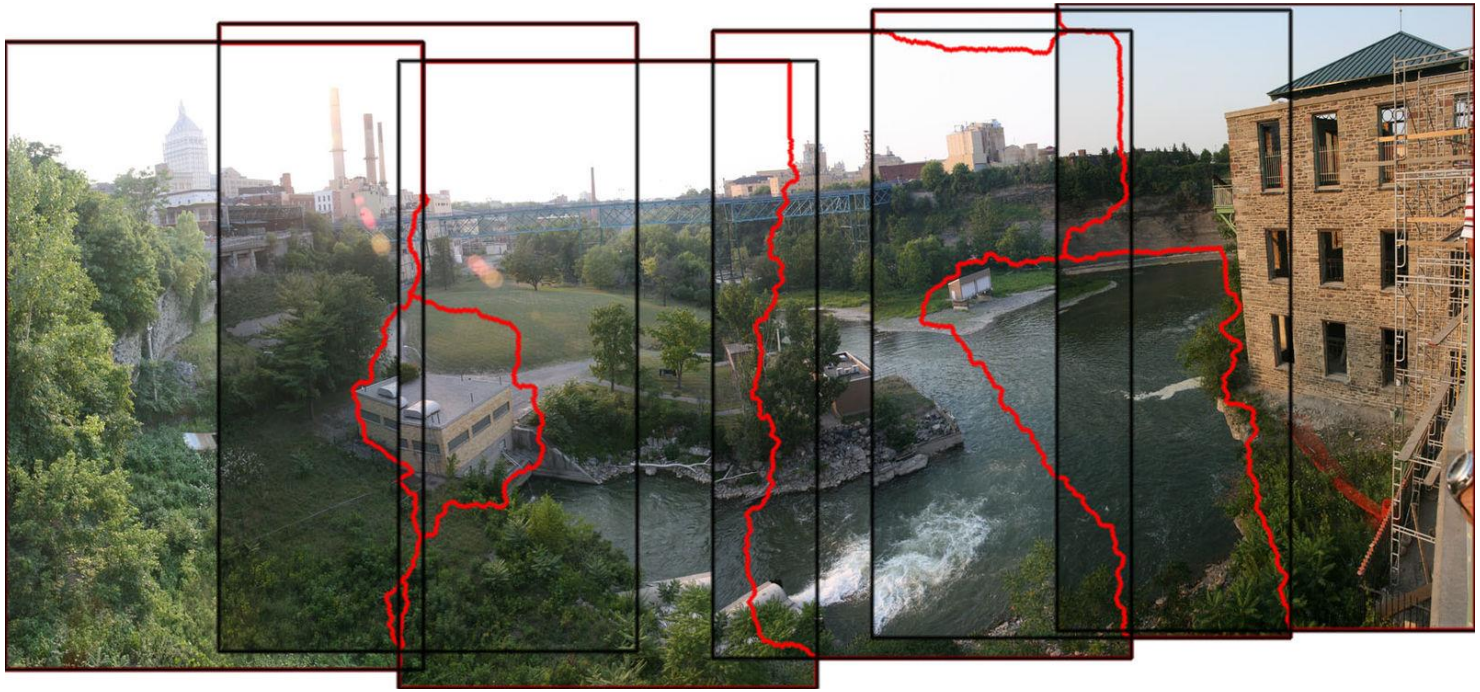
Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers
  4. Keep H if [redacted]

Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers
  4. Keep H if largest number of inliers
- Recompute H using all inliers

Useful for…




IEEE MultiMedia
IEEE COMPUTER SOCIETY

The image correspondence pipeline

1. Feature point detection
   - Detect corners using the Harris corner detector.

2. Feature point description
   - Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching *and* homography estimation
   - Do both simultaneously using RANSAC.