Recitation 5

2D Transformations

Agenda

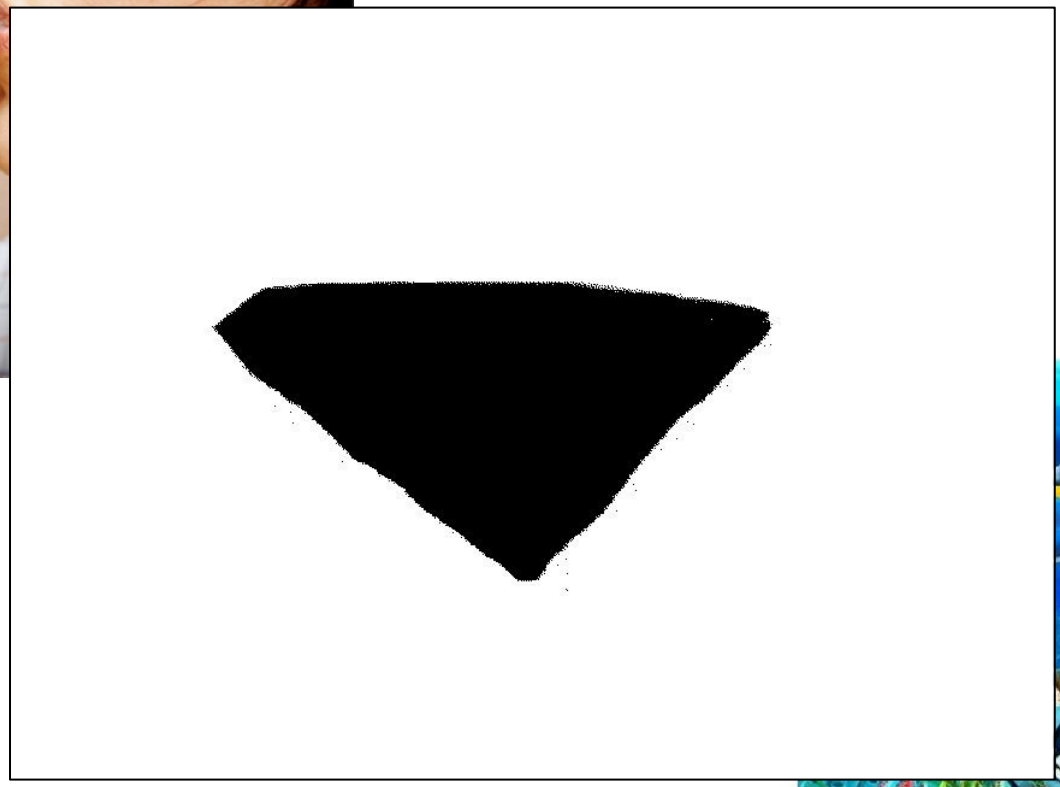
1. Ex 3 - Blending
2. 2D Transformations
3. Warping
4. Feature Points

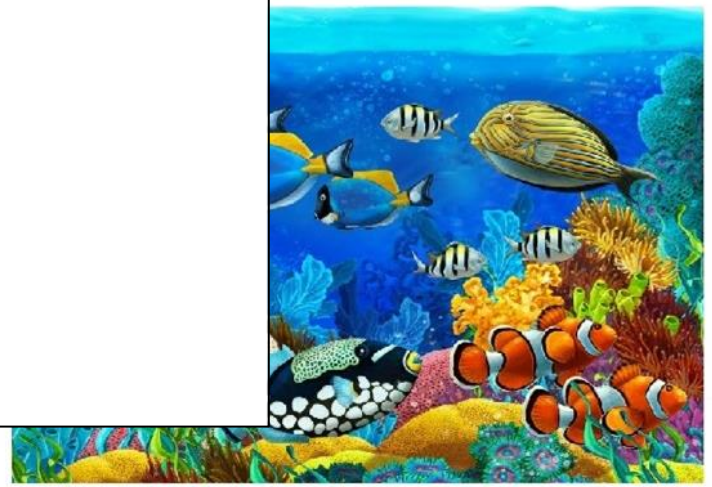
Ex 3 - Blending

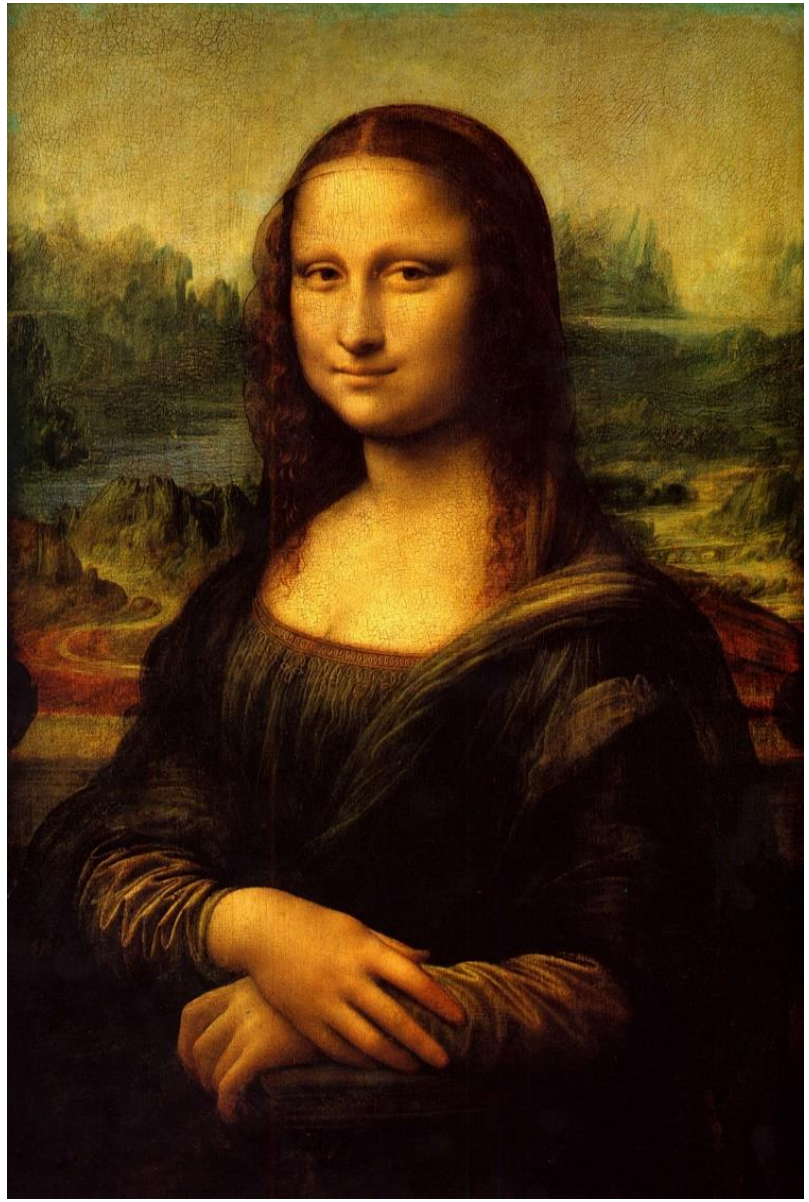
2D Transformations

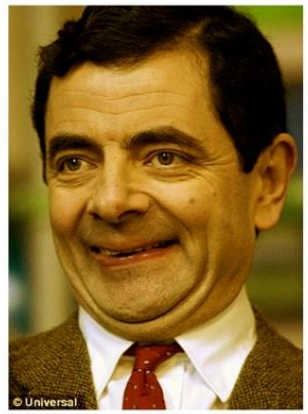
Warping

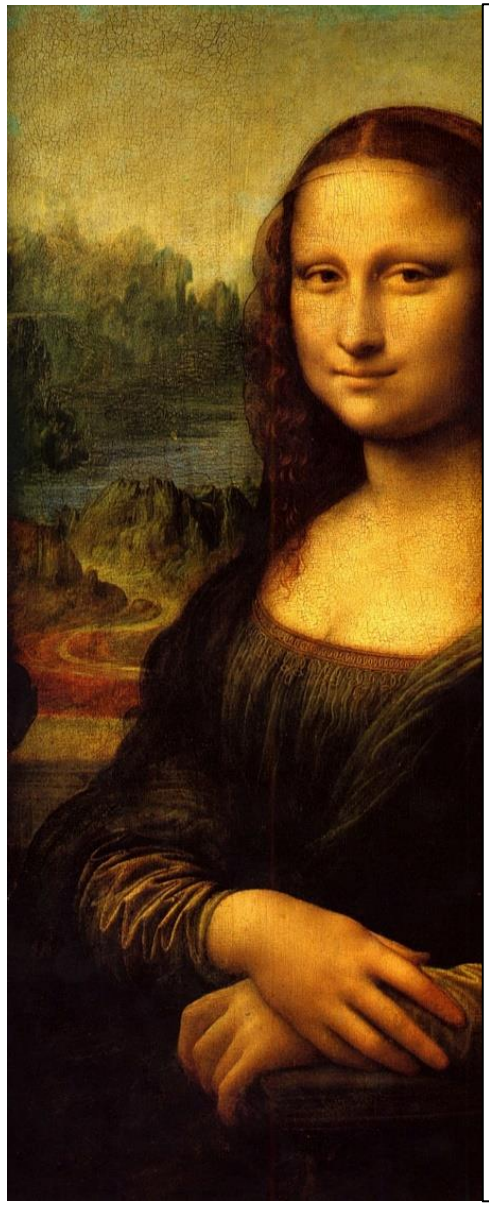
Feature Points

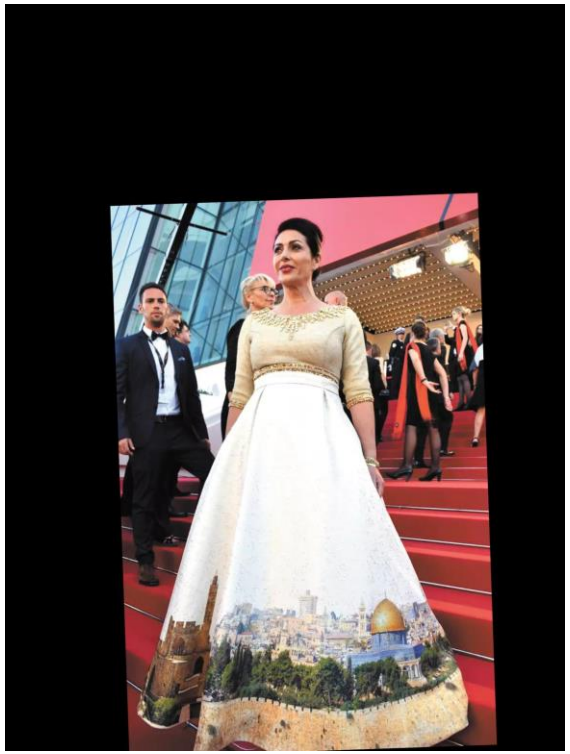
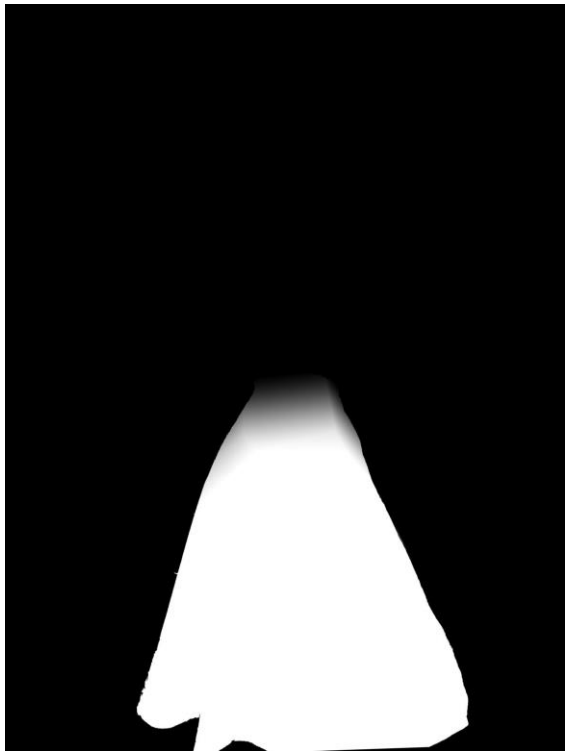







© Universal




Moshe Weitzhandler

Yoni Pushett




Leeyam Gabay

Ex 3 - Blending

2D Transformations

Image Warping

- image filtering: change the ***intensities*** of the image

f 

$$g(x,y) = T\big(f(x,y)\big)$$

$$\longrightarrow \boxed{T} \longrightarrow$$

g 

- image warping: change the ***coordinates*** of the image

f 

$$g(x,y) = f\big(T(x,y)\big)$$

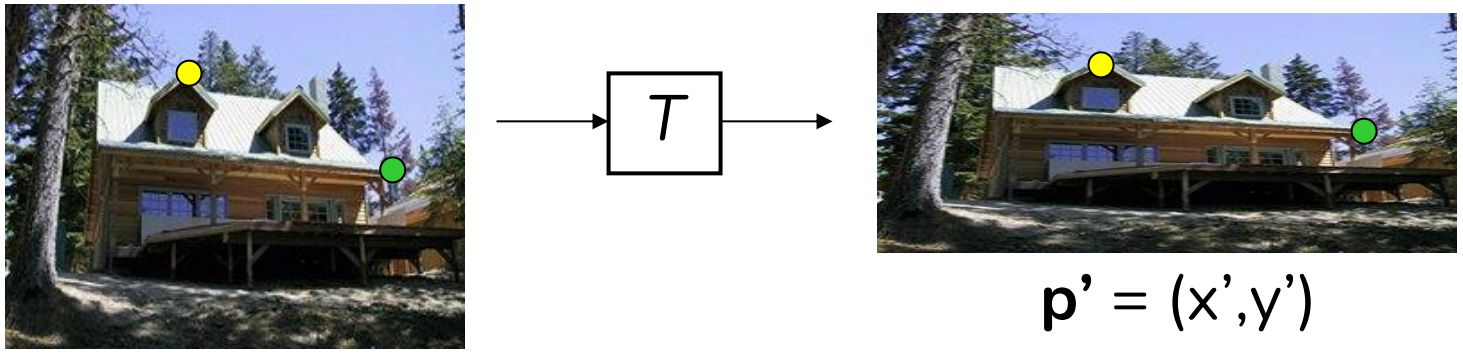
$$\longrightarrow \boxed{T} \longrightarrow$$

g 

Image Warping

Many slides from Steve Seitz and Alexei Efros

Global Parametric Warping



p = (x,y)

p' = (x',y')

- Transformation T is a **coordinate-changing** machine:

$$p' = T(p)$$

- What does it mean that *T* is global?

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$

  - Has the same form for any point *p*
  - Determined by just a few numbers (parameters)

Global Parametric Warping

- Examples of parametric warps:

translation

scaling

rotation

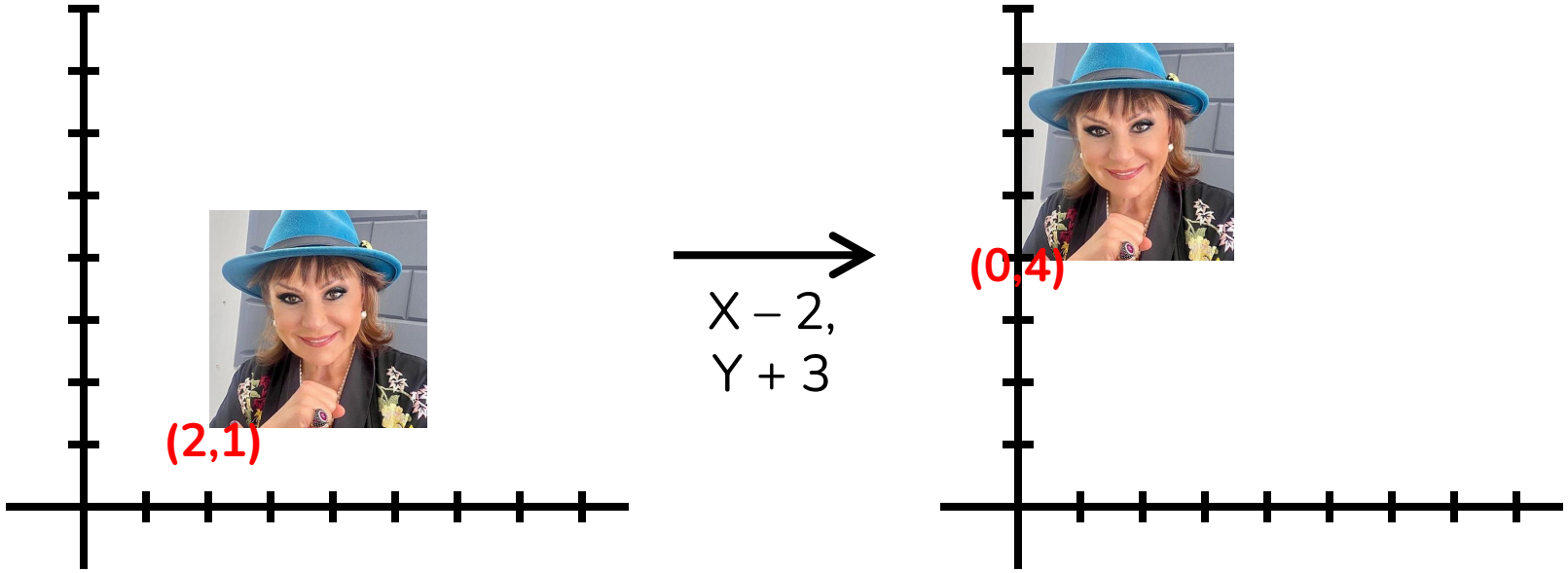
mirror

shear

perspective

Translation

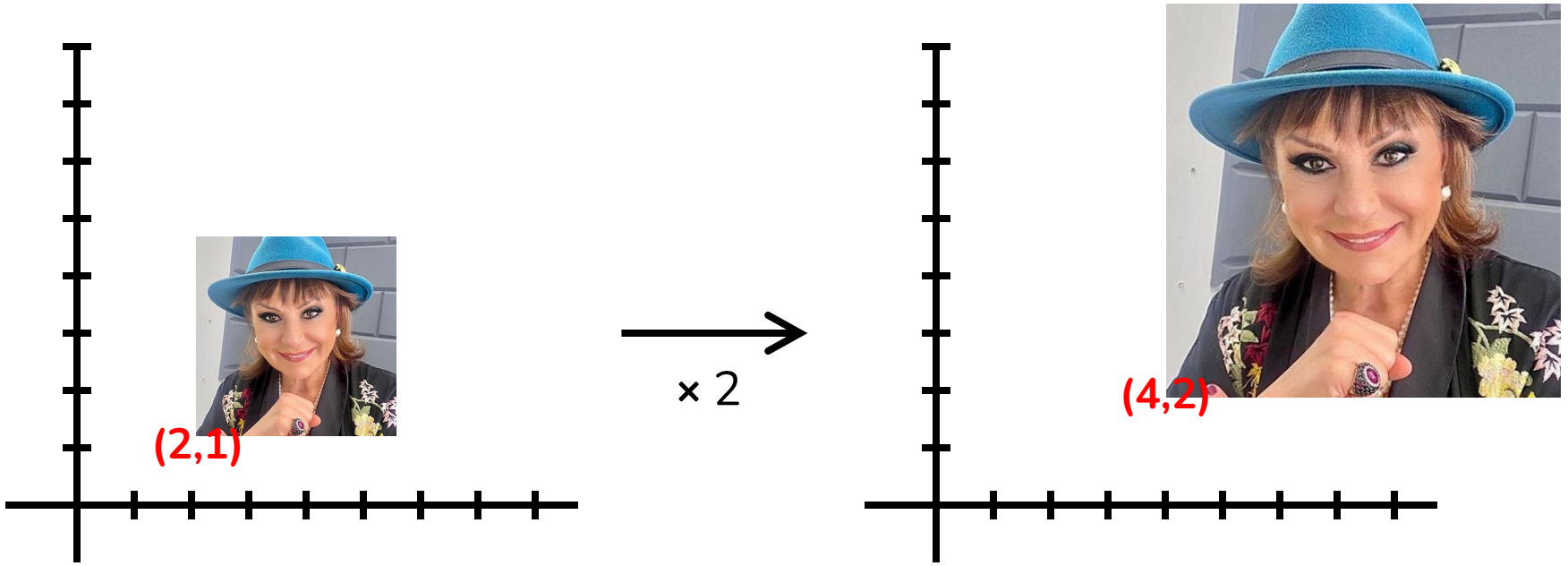
- *Translation* of a coordinate means adding a scalar to each of its components:



Q: How to represent in matrix form?

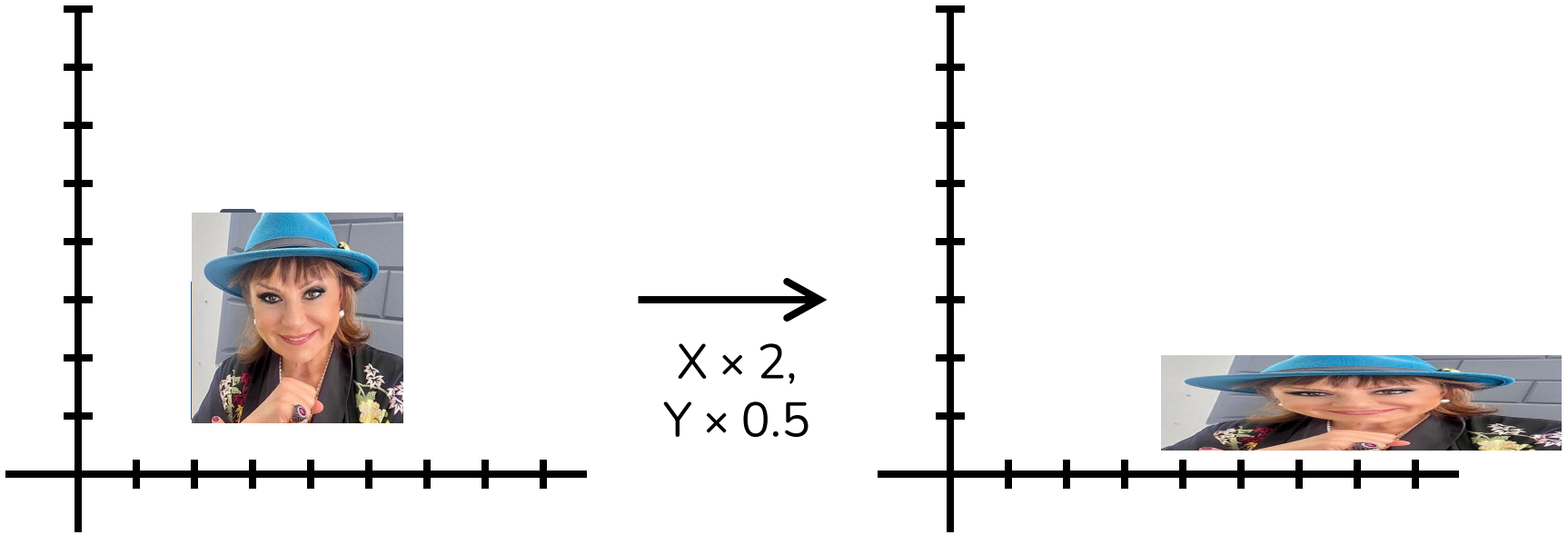
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components (dimensions):



Scaling

- *Non-uniform scaling*: different scalars per component (dimension):



Q: How to represent in matrix form?

Scaling

- The scaling operation:

$$x' = ax$$

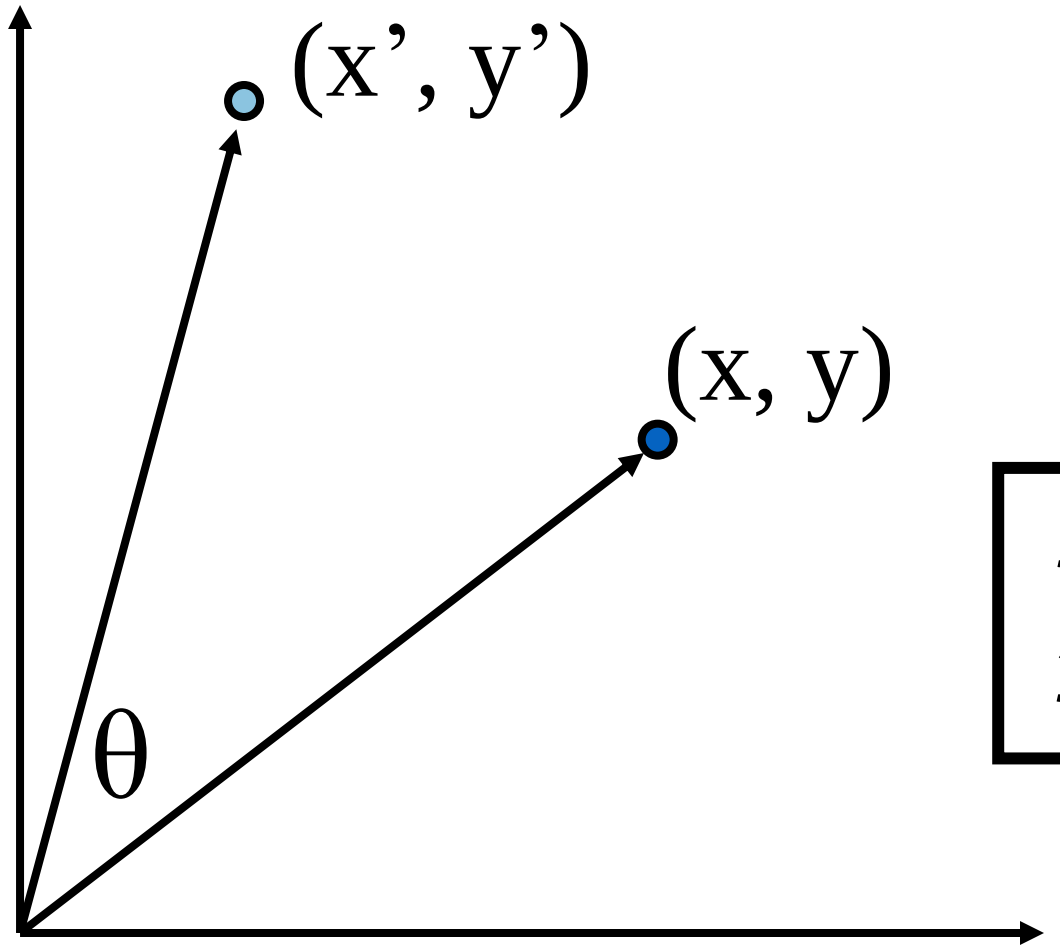
$$y' = by$$

- In matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

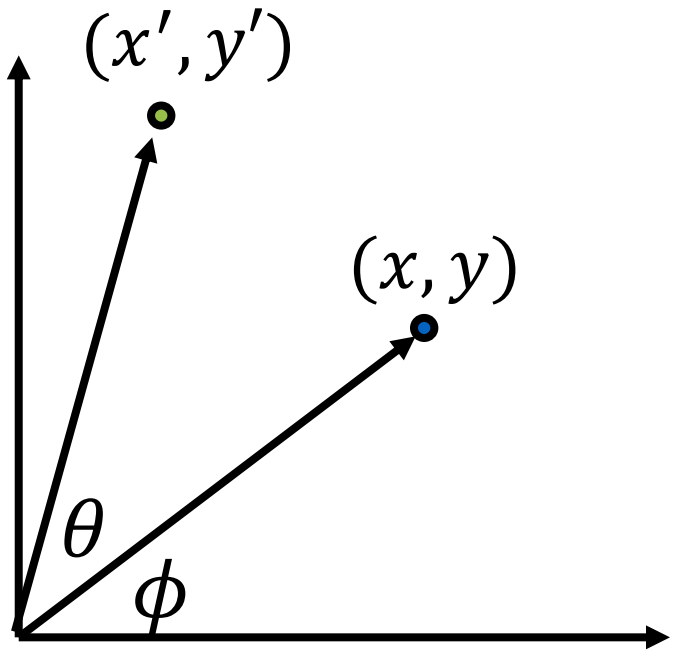
(What is the inverse of S?)

2-D Rotation



$$x' = xcos(\theta) - ysin(\theta)$$
$$y' = xsin(\theta) + ycos(\theta)$$

2-D Rotation



$$x = r\cos(\phi)$$
$$y = r\sin(\phi)$$
$$x' = r\cos(\phi + \theta)$$
$$y' = r\sin(\phi + \theta)$$

Trig Identity…

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

Substitute…

$$x' = xcos(\theta) - ysin(\theta)$$

Q: How to represent in matrix form?

2-D Rotation

- This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Even though sin(θ) and cos(θ) are nonlinear functions of θ,
  - ***x' is a linear combination of x and y***
  - ***y' is a linear combination of x and y***

- What is the inverse transformation?
  - Rotation by –θ
  - For rotation matrices, det(R) = 1 so $\mathbf{R}^{-1} = \mathbf{R}^{T}$

2x2 Matrices

- Which types of transformations can be represented by 2x2 matrices?

2D Identity:

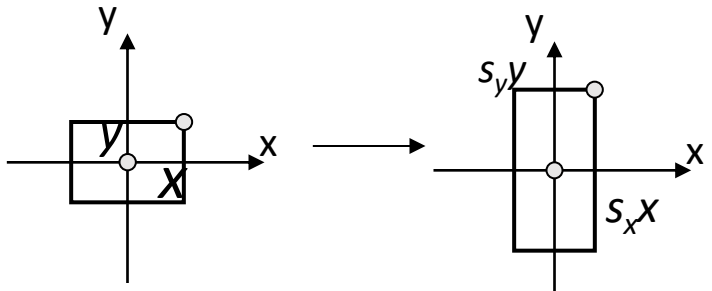
$$x' = x$$
$$y' = y$$

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

2D Scale around (0,0):

$$x' = s_x \cdot x$$
$$y' = s_y \cdot y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



2x2 Matrices

- Which types of transformations can be represented by 2x2 matrices?

2D Rotate around (0,0):

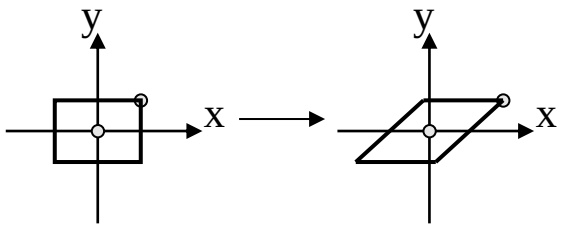
$$x' = \cos\Theta \cdot x - \sin\Theta \cdot y$$
$$y' = \sin\Theta \cdot x + \cos\Theta \cdot y$$

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

2D Shear:

$$x' = x + sh_x \cdot y$$
$$y' = sh_y \cdot x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Shear ;along x axis

2x2 Matrices

- Which types of transformations can be represented by 2x2 matrices?

  2D Mirror over the Y axis:

$$x' = -x$$
$$y' = y$$

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

  2D Mirror over (0,0):

$$x' = -x$$
$$y' = -y$$

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

2x2 Matrices

- Which types of transformations can be represented by 2x2 matrices?

  2D Translation?

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

NO!

2D translation is **NOT** a linear transformation from R^2 to R^2!

$$T(0) \neq 0$$

2D Linear Transformations

- Linear transformations are combinations of scale, rotation and shear.

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

- Properties of linear transformations:
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved along lines
  - Closed under composition:

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

  - Easy to find inverse when nonsingular

Homogeneous Coordinates

- **Q:** How can we represent translation as a 3x3 matrix?

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

- **A**: Add a 3rd coordinate to every 2D point

Homogeneous Coordinates

represent coordinates in 2 dimensions with a 3-vector

$$\begin{bmatrix} x \\ y \end{bmatrix} \longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ w \end{bmatrix} \longrightarrow \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

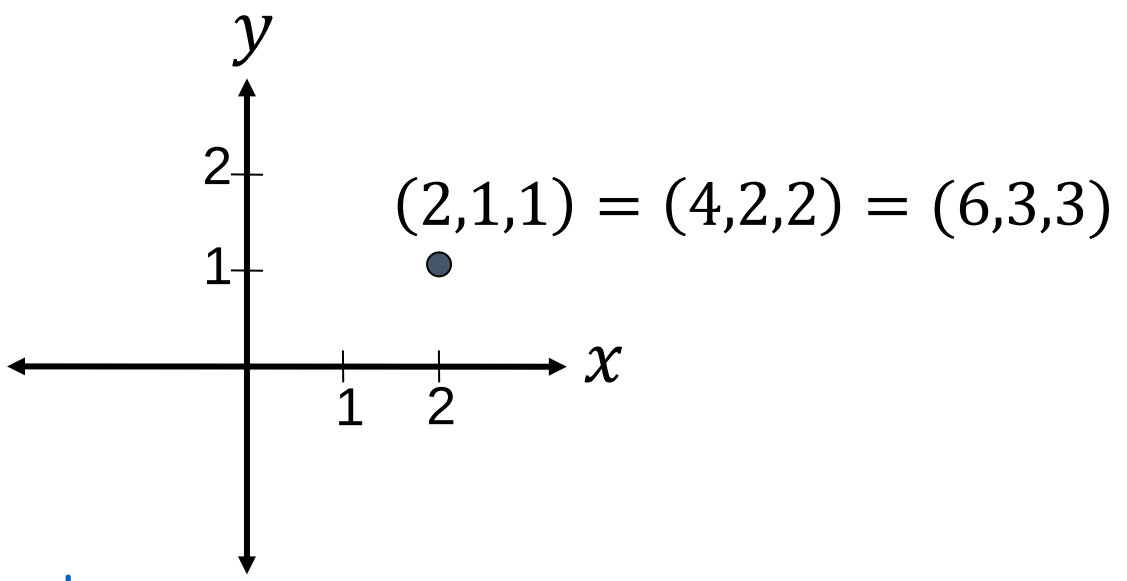
Scalar multiplication does not change the point:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{pmatrix} 2x \\ 2y \\ 2 \end{pmatrix} \cong \begin{pmatrix} 3.5x \\ 3.5y \\ 3.5 \end{pmatrix} \cong \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix}$$

are all equivalent to $\begin{pmatrix} x \\ y \end{pmatrix}$

Homogeneous Coordinates

- Add a 3rd coordinate to every 2D point
  - (x, y, w) represents a point at location (x/w, y/w)
  - (x, y, 0) represents a point at ∞
  - (0, 0, 0) is not allowed



Convenient coordinate system to represent many useful transformations

Homogeneous Coordinates

- **Q:** How can we represent translation as a 3x3 matrix?

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

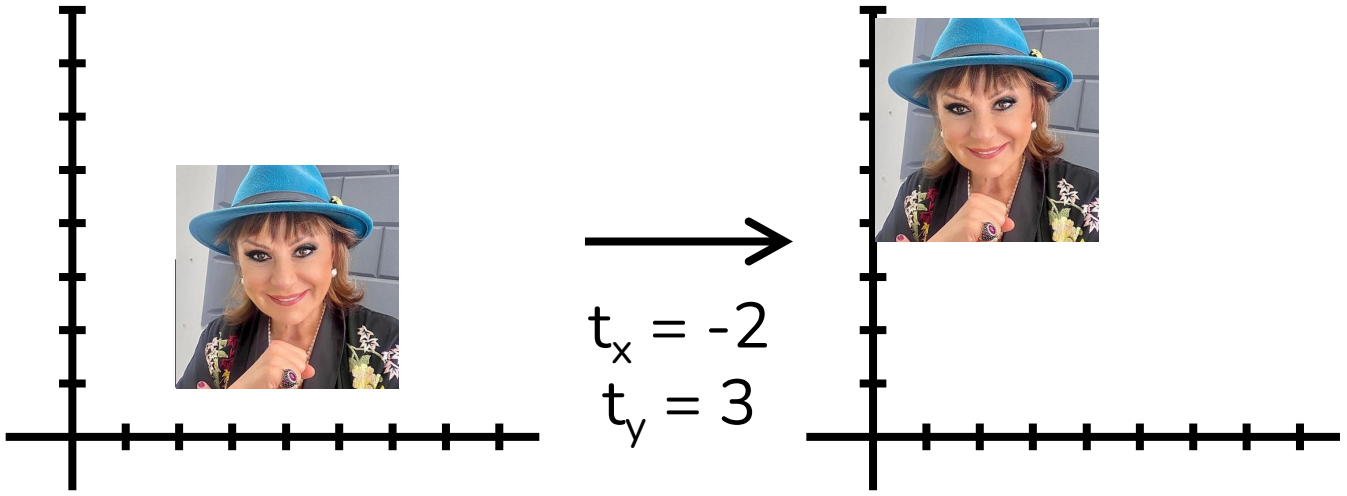
- **A:** Using the rightmost column

$$\mathbf{T}translation = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Translation: Example

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

Homogeneous Coordinates



Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

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

Translate

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

Scale

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

Rotate

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

Shear

Matrix Composition

- Transformations can be combined by matrix multiplication:

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

$\mathbf{p}'$ = $T(t_x,t_y)$ $R(Q)$ $S(s_x,s_y)$ $\mathbf{p}$

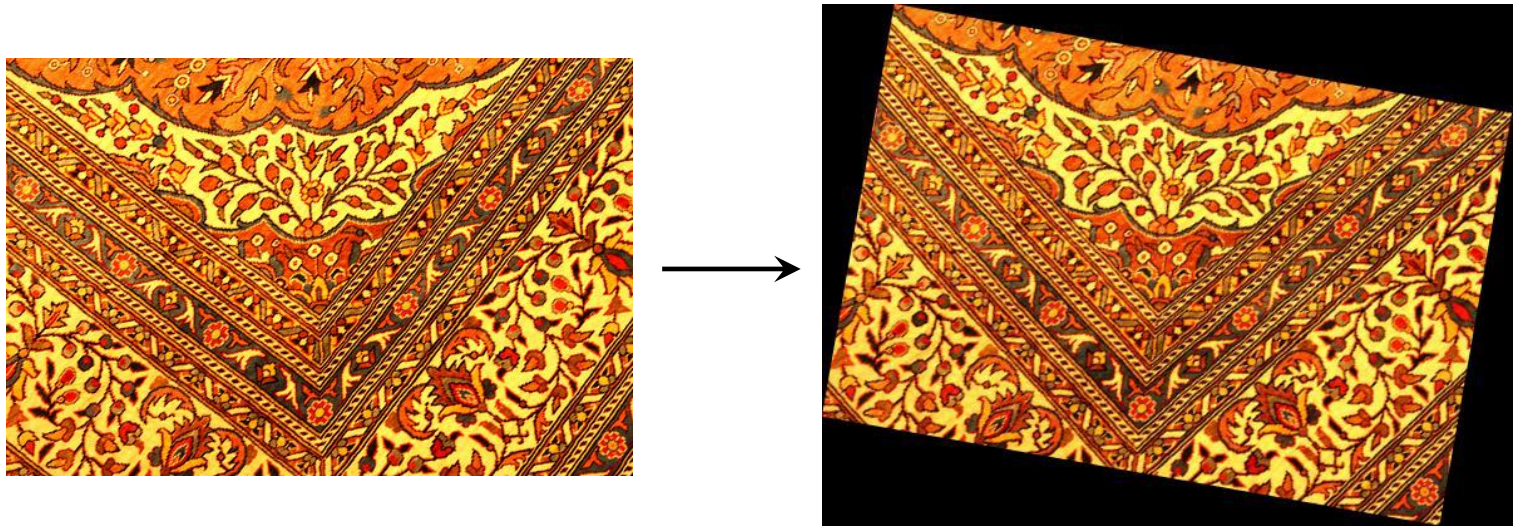
- Advantages:
  - General purpose representation
  - Hardware implementation
- Be aware: the order is important!
  - Matrix multiplication is not commutative

$$\boldsymbol{p}' = TRS\boldsymbol{p}$$

Rigid Transformations

Combination of translation and rotation

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

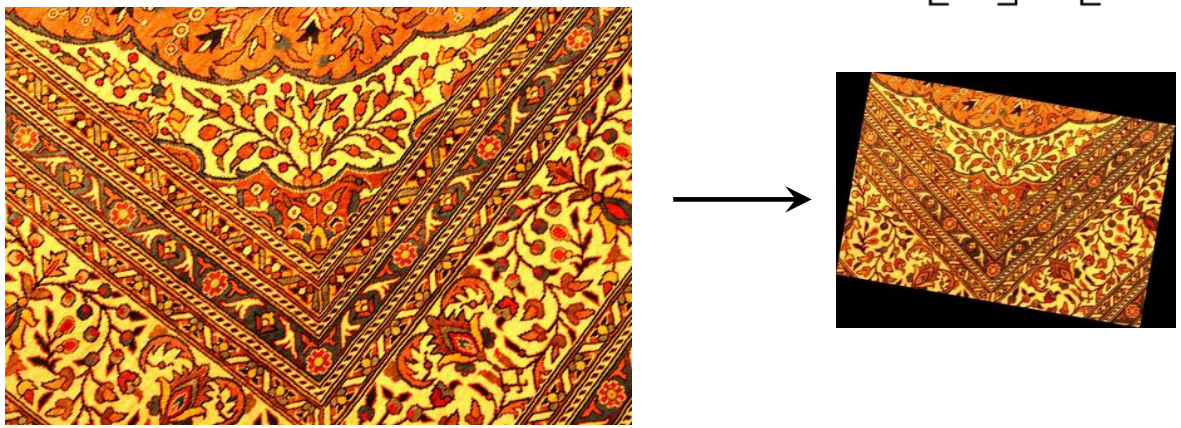


- # Parameters: 3 (t_x, t_y, θ)
- Preserves all information except of orientation (lengths, angles etc.)

Similarity Transformations

Combination of translation, rotation and uniform scaling

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} s \cdot \cos(\theta) & -s \cdot \sin(\theta) & t_x \\ s \cdot \sin(\theta) & s \cdot \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

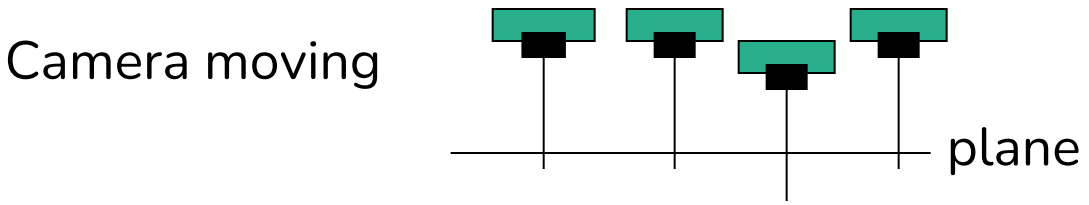


- # Parameters: 4 (s, t_x, t_y, θ)
- Preserves proportions (ratio of two lengths) and angles, but not lengths and orientation

Similarity Transformations

When do we meet them?

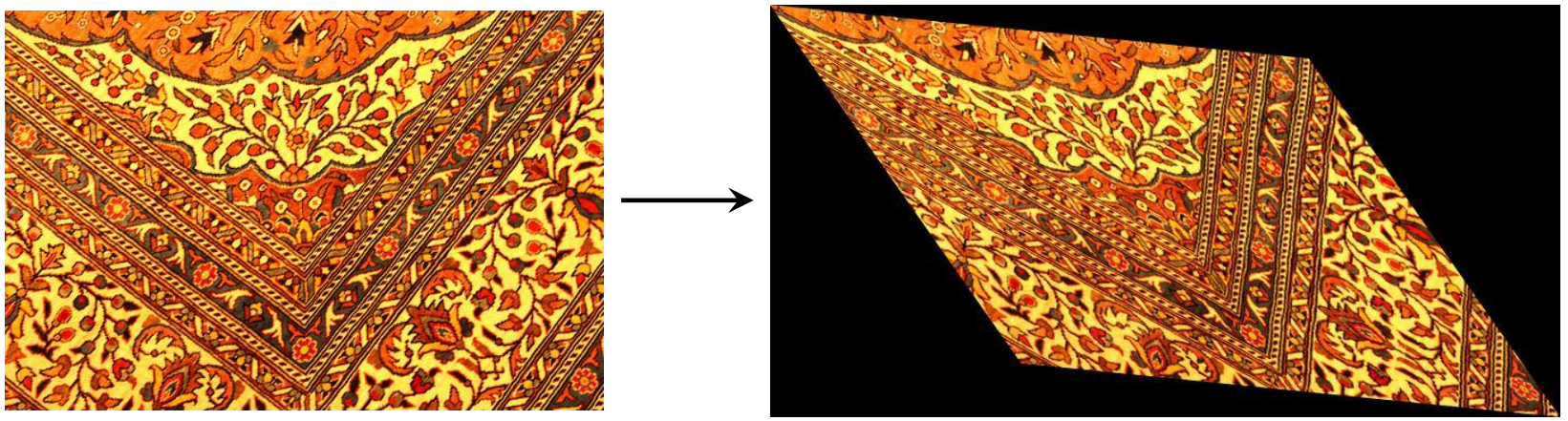
- When the camera is scanning a plane in parallel



Affine Transformations

Combination of translation with all 2D linear transformations (rotation, scaling and shear)

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



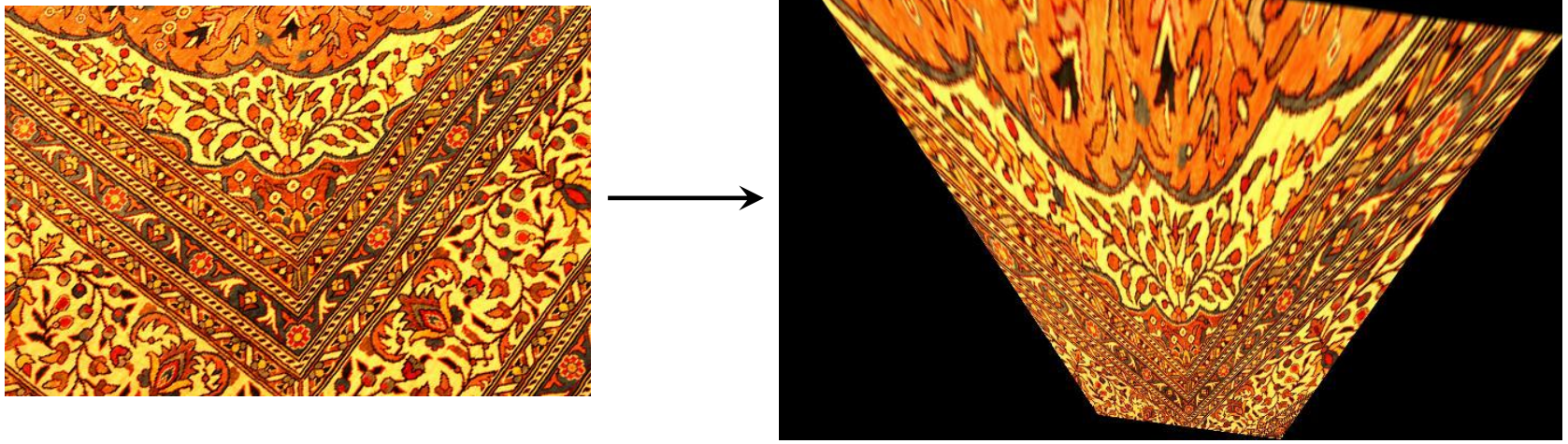
- # Parameters: 6
- Preserves parallelism, ratio of areas.

Projective Transformations

Affine transformations + projective warps

Maps points from a 2D plane to another 2D plane

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



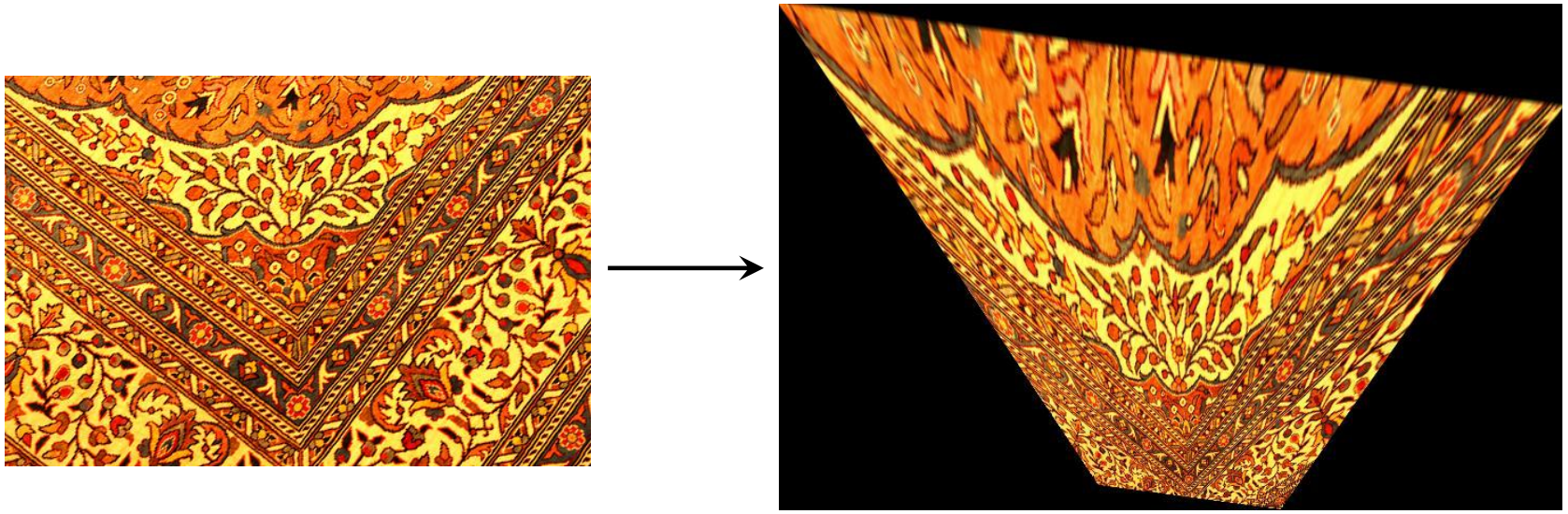
- # Parameters: 8 (the 9th is redundant, since we use homogenous coordinates)

Projective Transformations

Affine transformations + projective warps

Maps points from a 2D plane to another 2D plane

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



- # Parameters: 8 (the 9th is redundant, since we use homogenous coordinates)
- **Preserves only straight lines** (not parallelism, angles etc.)

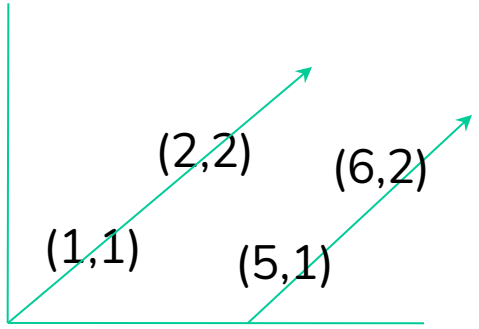
Projective Transformations

Affine transformations + projective warps

Maps points from a 2D plane to another 2D plane

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

$$\frac{x'}{w} = \frac{a \cdot x + b \cdot y + c}{x \cdot g + y \cdot h + 1} \qquad \frac{y'}{w} = \frac{d \cdot x + e \cdot y + f}{x \cdot g + y \cdot h + 1}$$



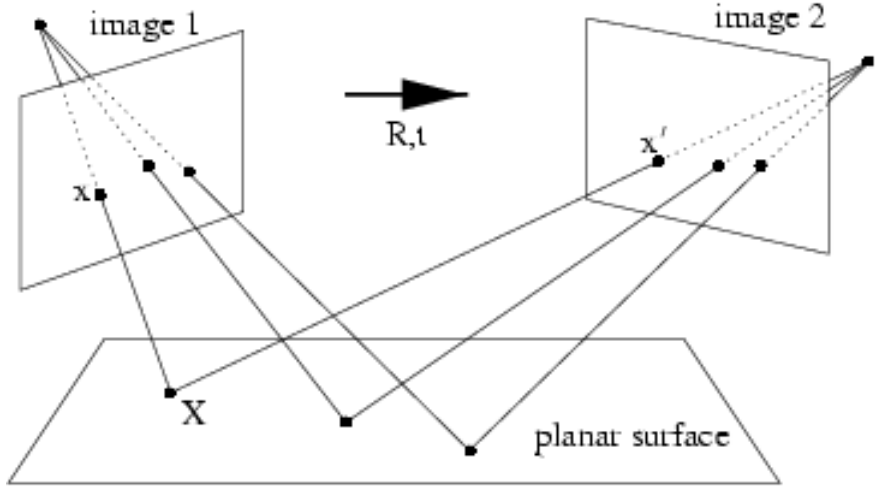
(1/3,1/3) (2/5,2/5)

(5/7,1/7) (6/9,2/9)

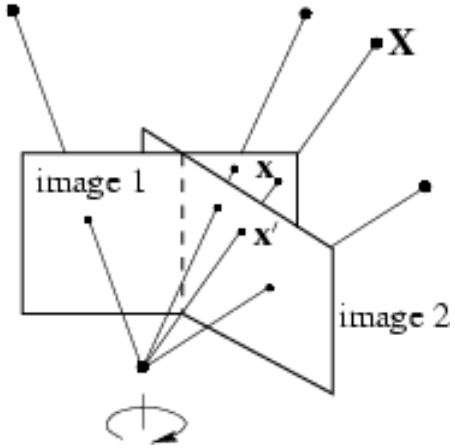
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Projective Transformations

Examples of projective Transformations :



Transformation between 2 images induced by a world plane



Camera rotation (same center of projection) / camera varying the focal length

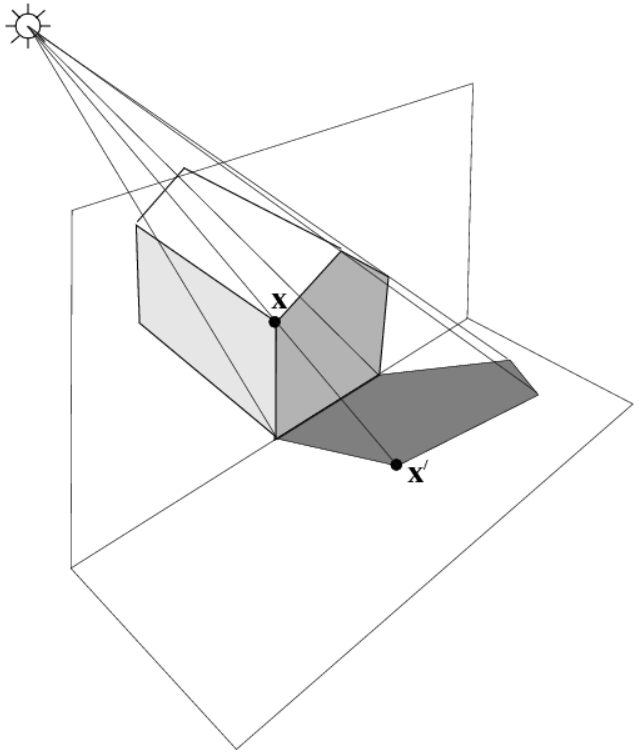


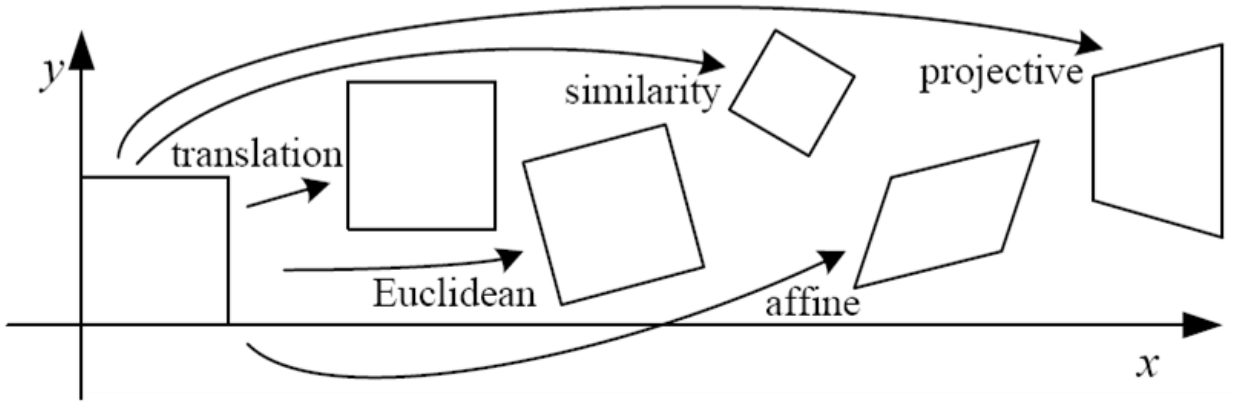
Image of a plane and its shadow on another plane

Removing Perspective Distortion

Original image

Warped image

2D Image Transformations

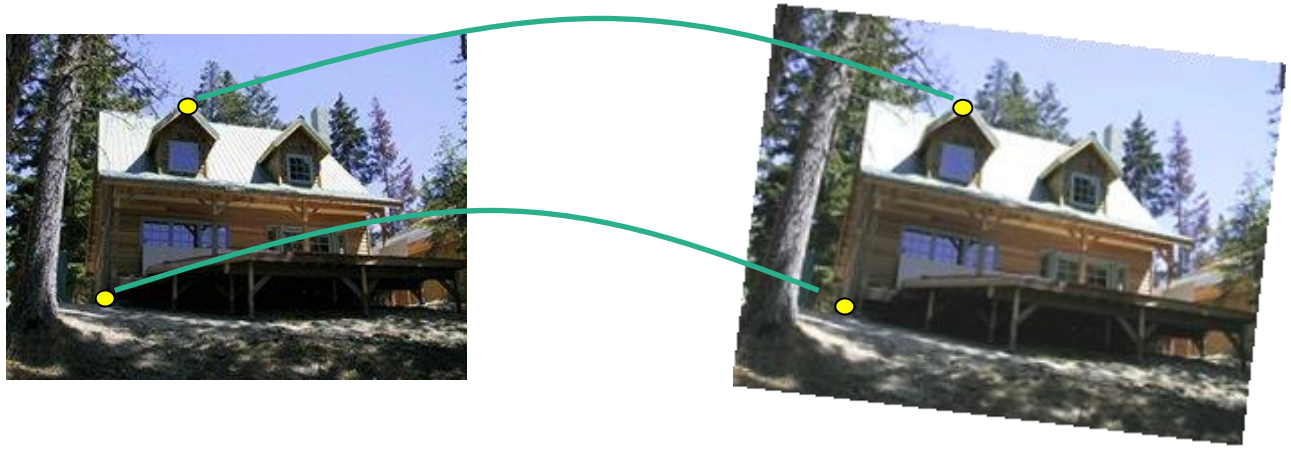


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

- These transformations are a nested set of groups
- Each of them is closed under composition and is invertible

Computing 2D Transformations

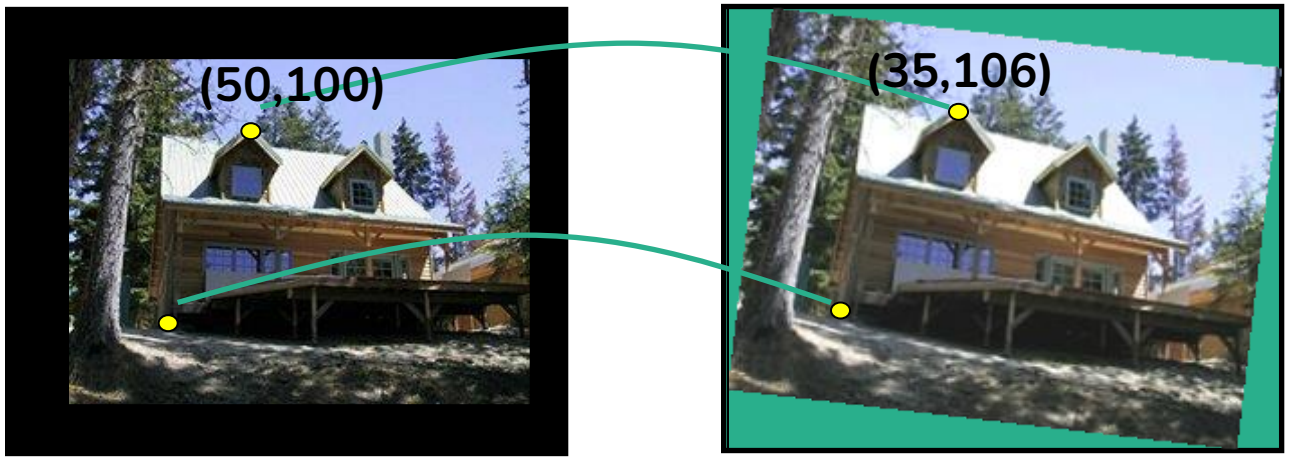
Corresponding sets of points enable computing the transformation matrix



- Each corresponding pair contributes 2 equations
- How many points are required for each different transformation (translation, rigid, similarity, affine, projective)?

Computing 2D Transformations

Example: rigid transformation



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 106 \\ 35 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 100 \\ 50 \\ 1 \end{bmatrix}$$

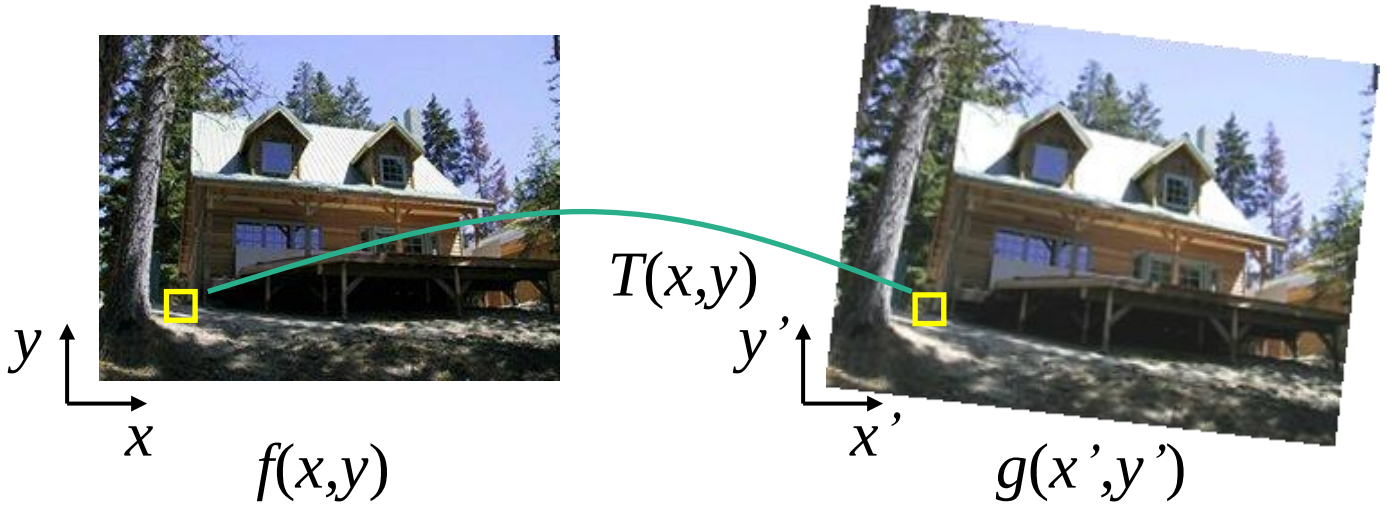
$\longrightarrow$ Extract θ

2D Transformations

Warping

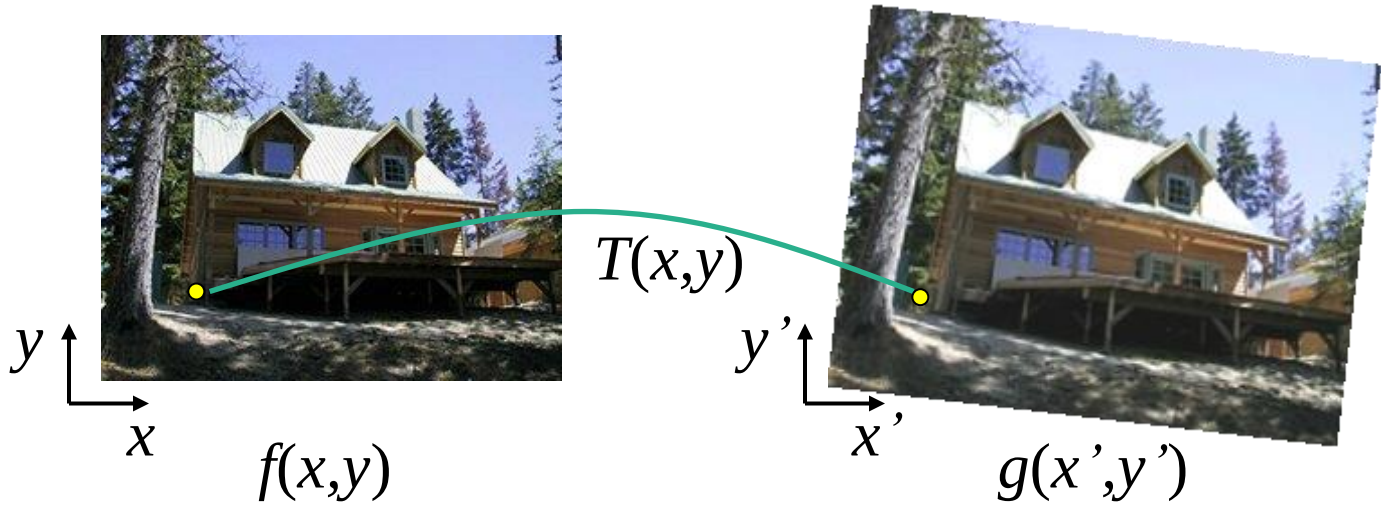
Feature Points

Image warping



Given a coordinate transform *(x',y')* = *T*(*x*,*y*) and a source image *f*(*x*,*y*), how do we compute a transformed image *g(x',y')* = *f*(*x*,*y*)?

Forward warping

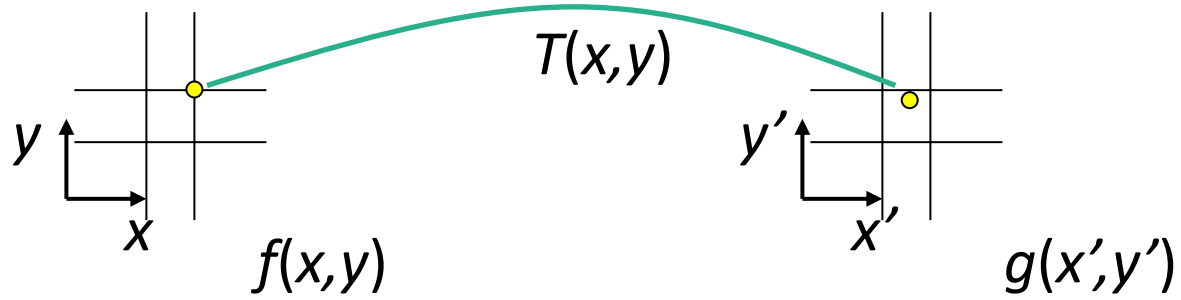


Send each pixel (x,y) to its corresponding location

$(x',y') = T(x,y)$ in the second image

Q: What is pixel lands "between" two pixels?

Forward warping



Send each pixel (x,y) to its corresponding location
$(x',y') = T(x,y)$ in the second image

Q: What is pixel lands "between" two pixels?

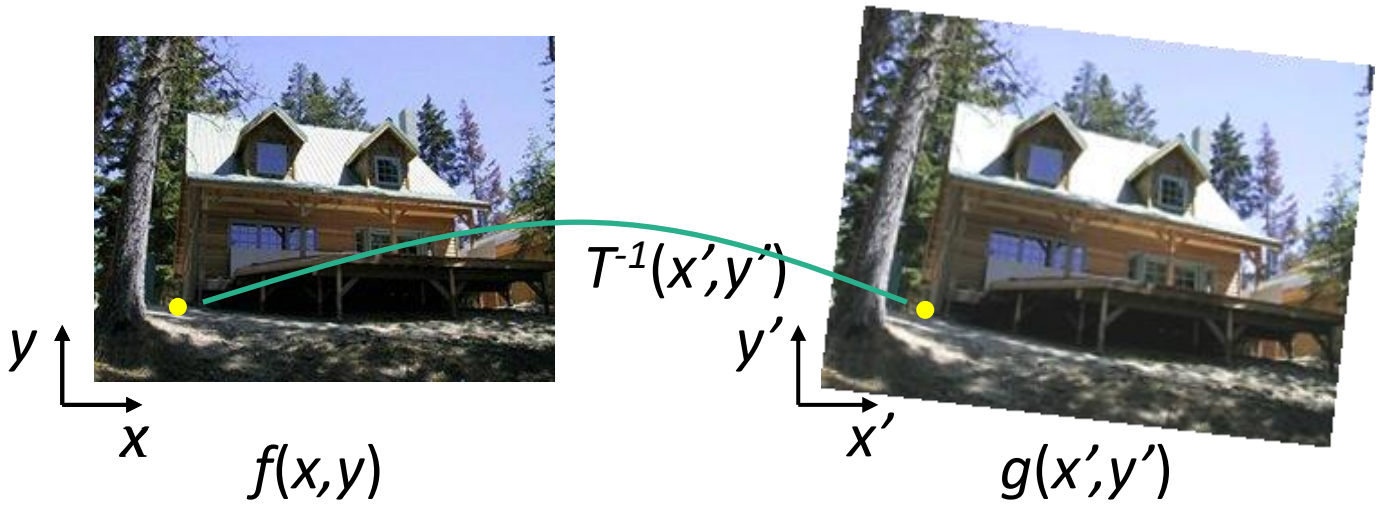
A: distribute its color among the neighboring pixels of (x',y')

- known as "splatting"

Q: What happens when we scale an image by a factor of 2?

A: holes every second pixel

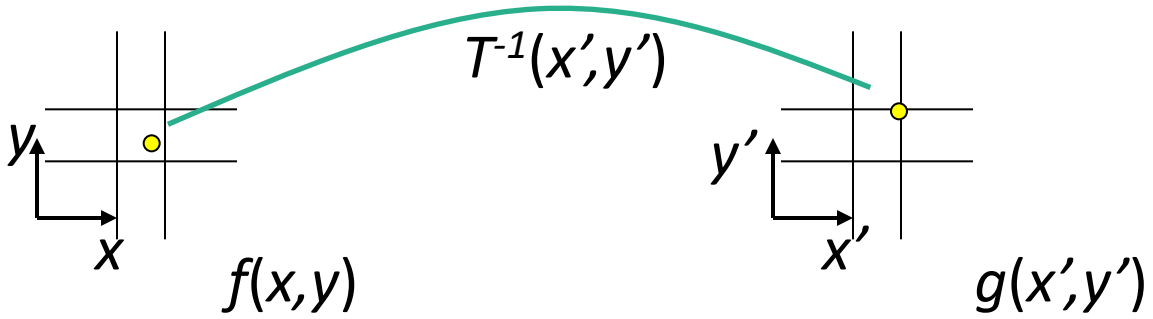
Backward (inverse) warping



Get each pixel (x',y') from its corresponding location
$(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

Inverse warping



Get each pixel g(x',y') from its corresponding location (x,y) = T-1(x',y') in the first image

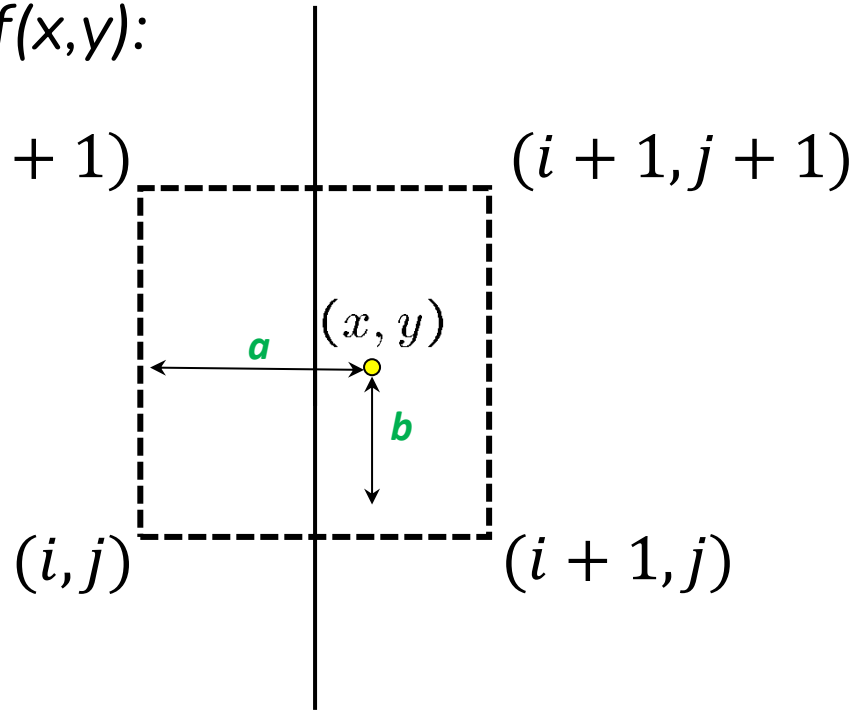
Q: what if pixel comes from "between" two pixels?

A: ***Interpolate*** the color value from its neighbors

- nearest neighbor, bilinear, bicubic, Gaussian

Bilinear interpolation

Sampling at *f(x,y)*:



$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

Forward vs. inverse warping

Q: which is better?

A: usually inverse – eliminates holes

- however, it requires an invertible warp function, which isn't always available.

Warping using Python

- How do we avoid loops ?
  - Use np.meshgrid() to represent image indices

- Transform coordinates using homography

- Use map_coordinates() from scipy.ndimage to perform the interpolation

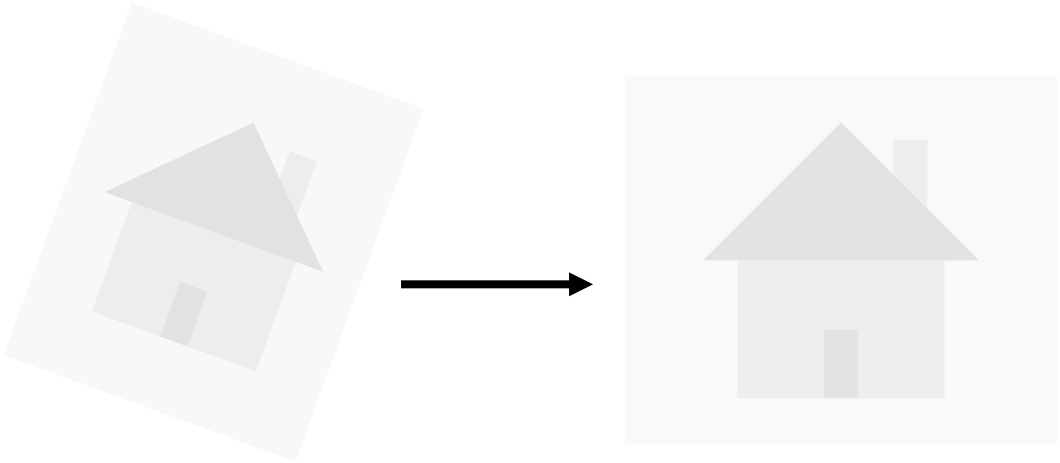
2D Transformations
Warping

Feature Points

Outline

- Motivation for using image features
  - Feature based alignment
- Desired Properties of features
- Feature Detectors
  - Harris corner detector
  - Scale invariant Harris detector
- Feature Descriptors
  - MOPS: Multi-scale oriented patches
  - SIFT: orientation histograms (relative to key-point orientations)

Motivation: Image Alignment



How do we align two images automatically?

Two broad approaches:

- **Direct (pixel-based) alignment**
  - Search for alignment where most pixels agree
  - (Cross correlation, LK)

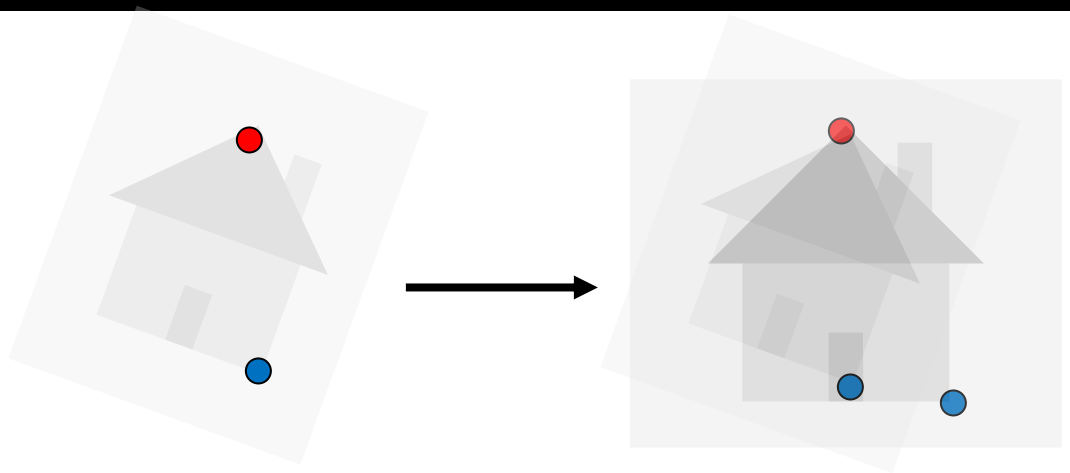
Direct Methods Limitations

Less suitable for partially overlapping images:

photo based panorama

- Limited range of convergence

Motivation: Image Alignment



How do we align two images automatically?

Two broad approaches:

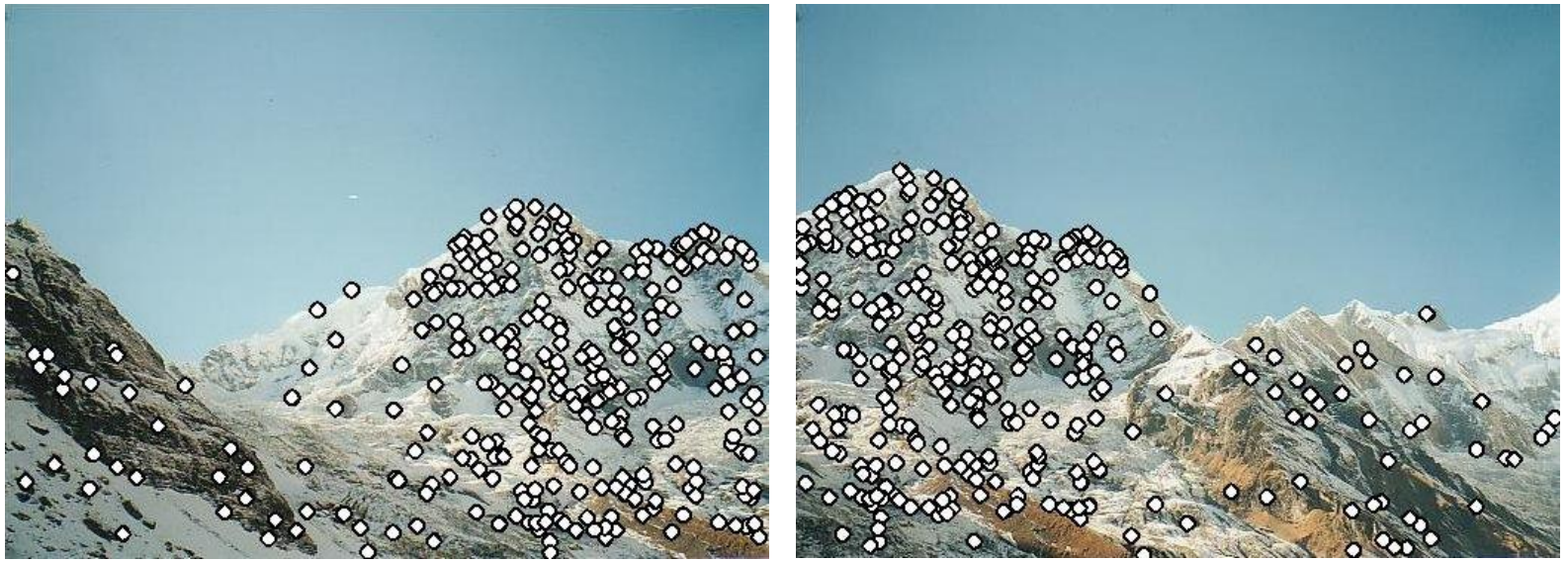
- **Direct (pixel-based) alignment**
  - Search for alignment where most pixels agree
  - (Cross correlation, LK)
- **Feature-based alignment**
  - Find a few matching features in both images
  - compute alignment

Example For Feature Based Alignment

M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003

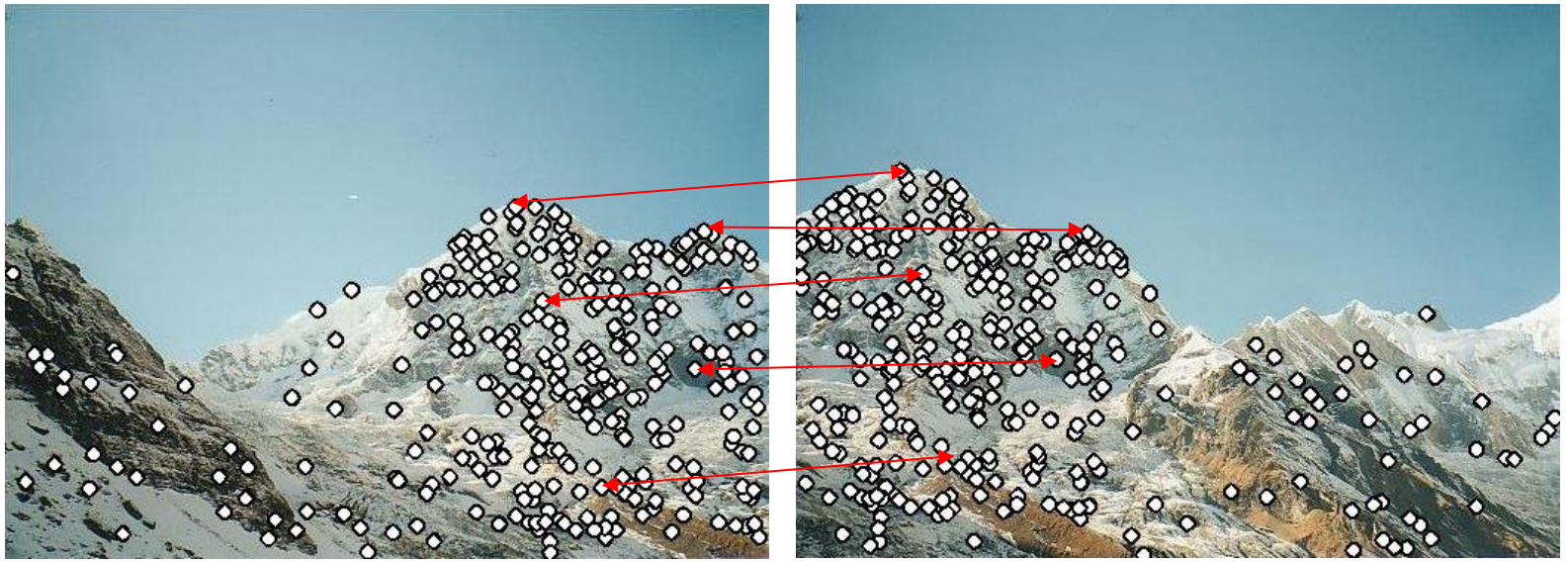
Matching with Features

1. **Detect** feature points in both images
2. Build a **descriptor** from each point



Matching with Features

1. **Detect** feature points in both images
2. Build a **descriptor** from each point
3. Find corresponding pairs



Matching with Features

1. **Detect** feature points in both images
2. Build a **descriptor** from each point
3. Find corresponding pairs
4. ✅ Use these pairs to align images

Matching with Features

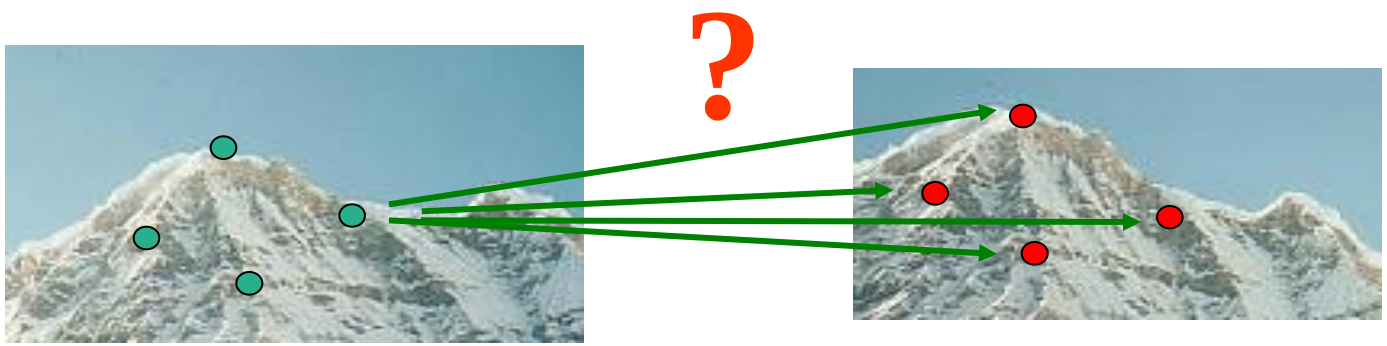
Problem 1: Detect the *same* points *independently* in both images

no chance to match!

We need a repeatable detector

Matching with Features

Problem 2: For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

Matching with Features: Desired Properties

- Easy to extract
- Invariance: tolerate image changes in
  - Image noise
  - Illumination
  - Scaling
  - Translation, Rotation
  - Viewing direction
- Easy to match against large database of object features

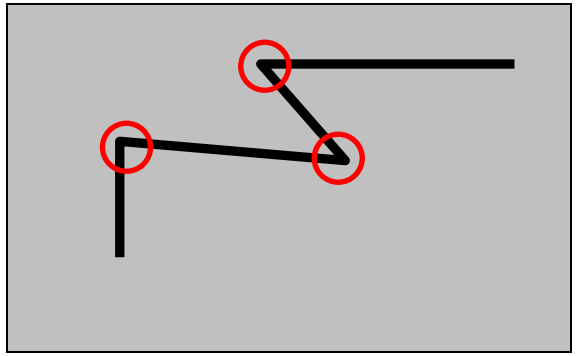
Matching with Features

1. **Detect** feature points in both images
2. Build a **descriptor** from each point
3. Find corresponding pairs
4. ✅ Use these pairs to align images

Harris corner detector

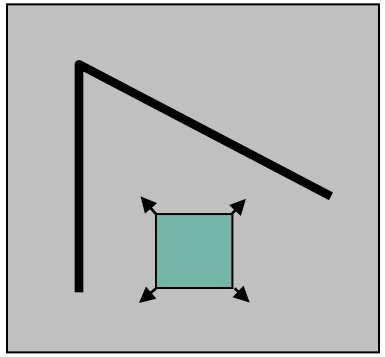
Harris corner detector

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

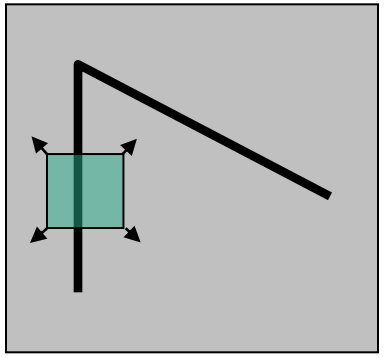


Harris Detector: Basic Idea

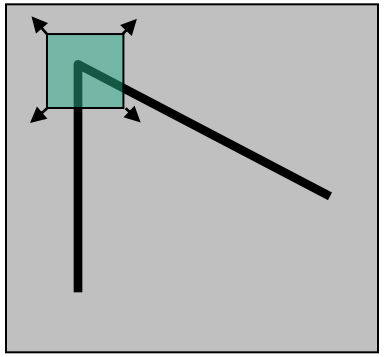
We should easily recognize the point by looking through a small window and shifting it:



"flat" region: no change in all directions



"edge": no change along the edge direction



"corner": significant change in all directions

Harris Detector: Mathematics

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

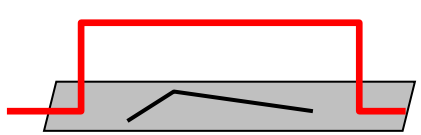
$$E(u,v) = \sum_{x,y} w(x,y) \cdot [I(x+u, y+v) - I(x,y)]^2$$

Window function

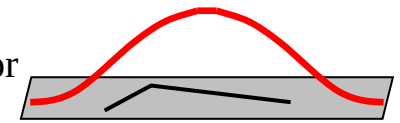
Shifted intensity

Intensity

Window function $w(x,y)$ =



or

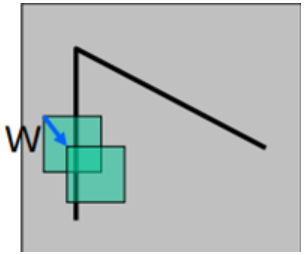


1 in window, 0 outside

Gaussian

Harris Detector: Mathematics

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



Local Taylor approximation: $f(x+h) \approx f(x) + f'(x) \cdot h$

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

$$[I(x+u, y+v) - I(x,y)] \approx I(x,y) + I_x \cdot u + I_y \cdot v - I(x,y) = I_x \cdot u + I_y \cdot v$$

$$[I(x+u, y+v) - I(x,y)]^2 \approx [(\mathrm{u}\ \mathrm{v}) \begin{pmatrix} I_x \\ I_y \end{pmatrix}]^2$$

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

Harris Detector: Mathematics

For small shifts [u,v] we have an approximation:

$$E(u,v) \approx \sum_{(x,y)\in W} \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

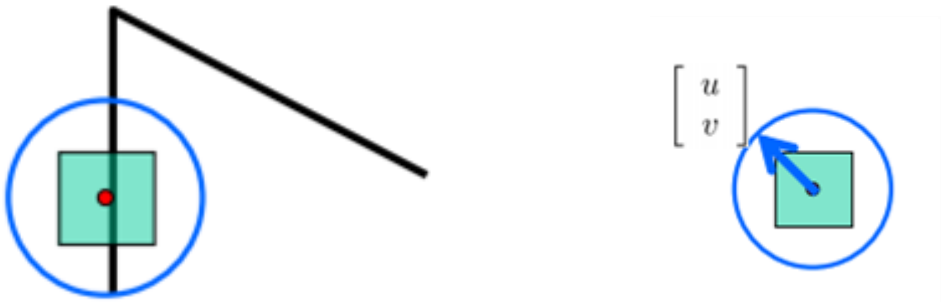
$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_y I_x & \sum_{(x,y)\in W} I_y^2 \end{bmatrix}$$

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

We use the simple rectangular window for simplicity, for other windows we get:

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

Harris Detector: Mathematics



- Which directions will result in the largest and smallest E values?
- **We can find these directions by looking at the eigenvectors of M**

Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis of M

$$E(u,v) \cong [u,v] \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

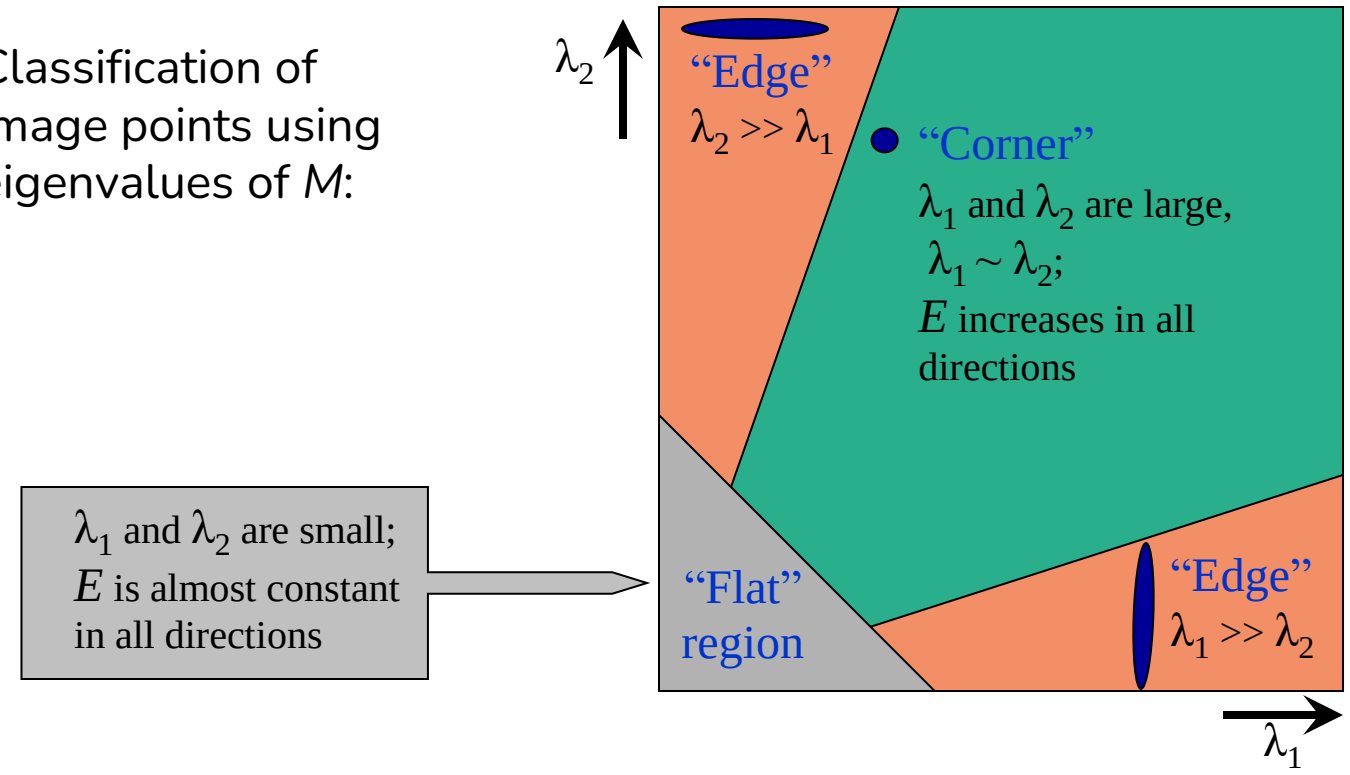
(u_1,v_1) , (u_2,v_2) – the corresponding eigenvectors

The major axes is in the direction of (u_1,v_1)
(proportional to λ_1)

The minor axes is in the direction of (u_2,v_2)
(proportional to λ_2)

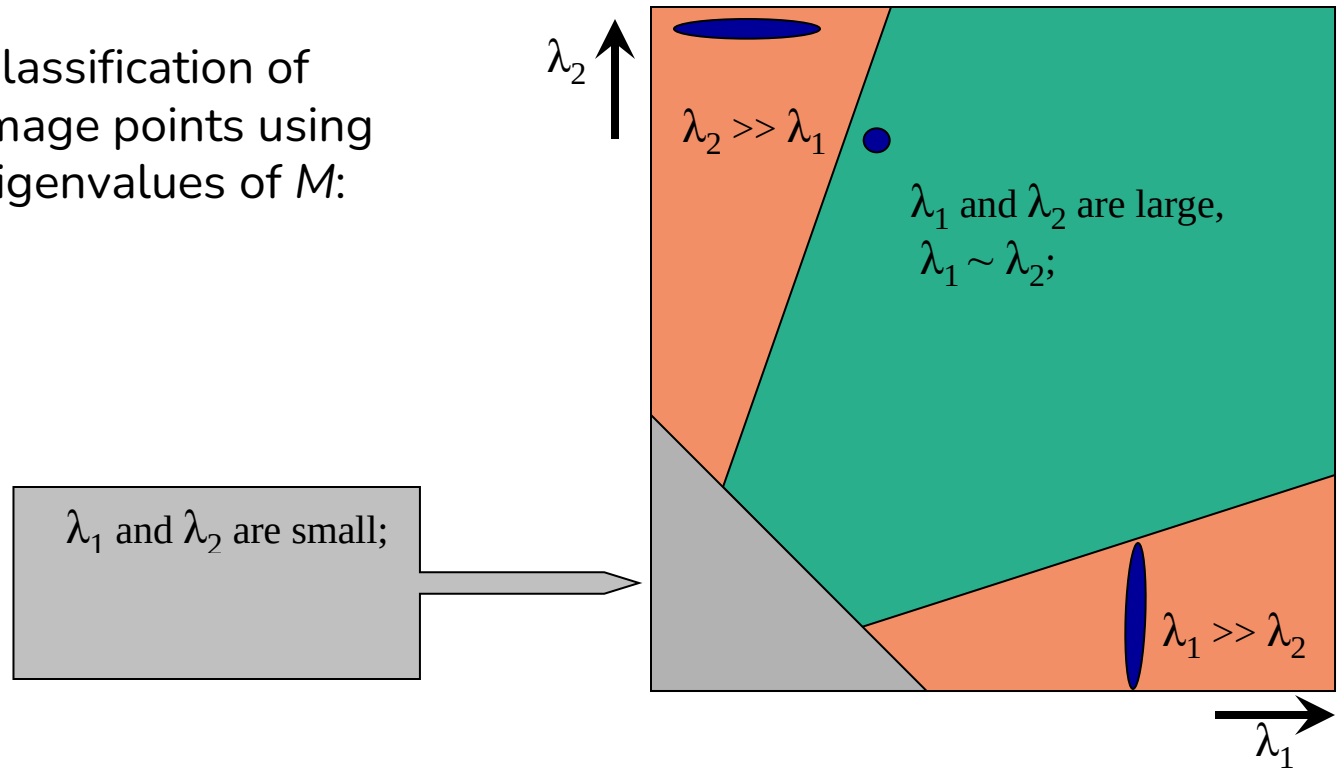
Harris Detector: Mathematics

Classification of image points using eigenvalues of *M*:



Harris Detector: Mathematics

Classification of image points using eigenvalues of *M*:

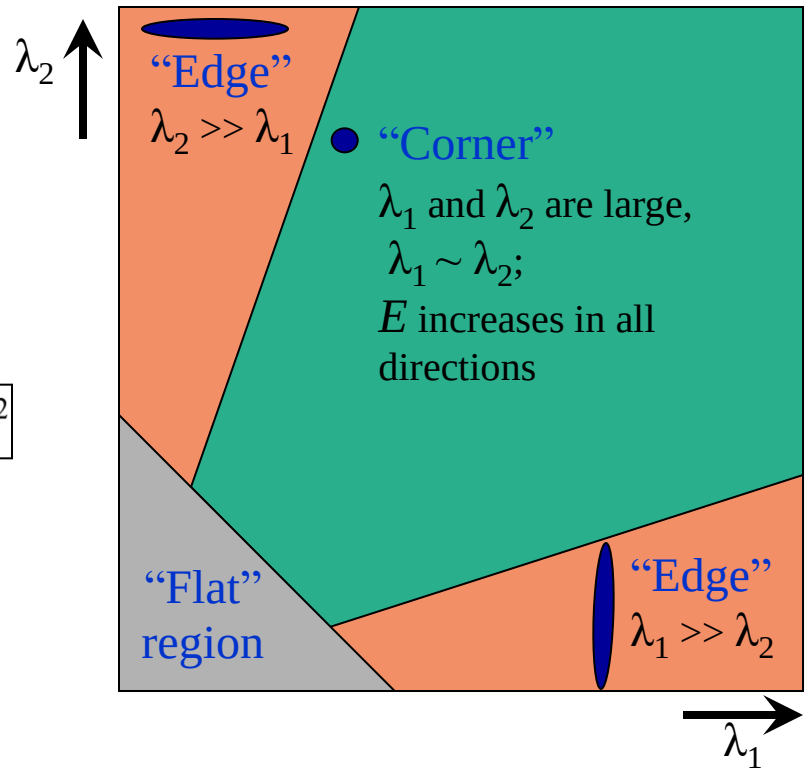


Harris Detector: Mathematics

Commonly used measures of corner response:

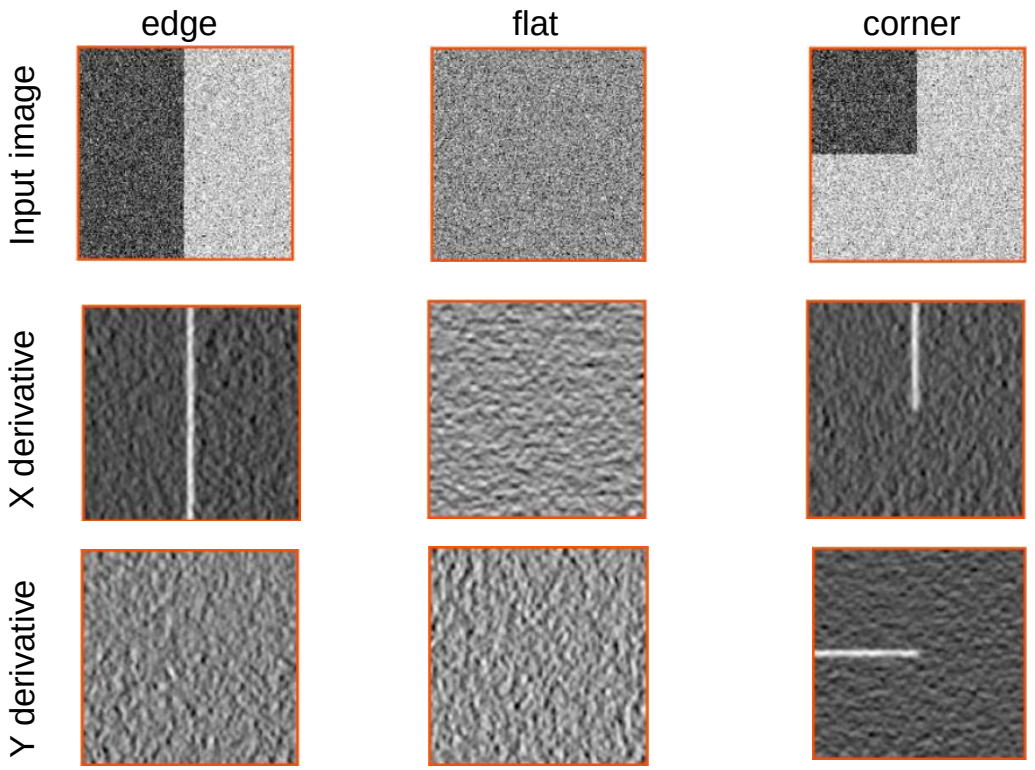
$$R_N = \frac{\text{Det } M}{\text{Trace } M} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$R_H = \text{Det } M - k(\text{Trace } M)^2$$

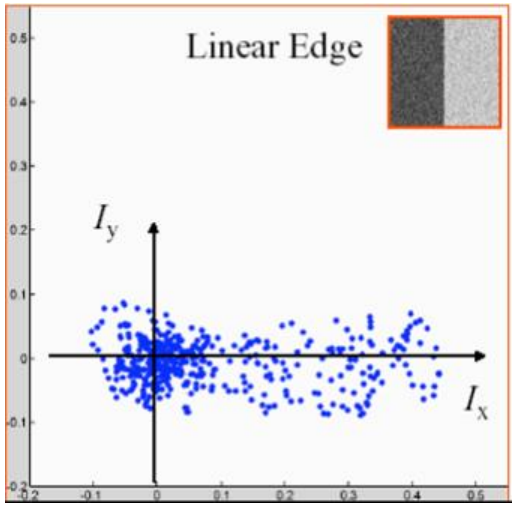


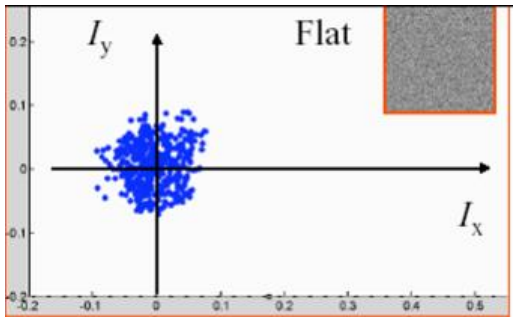
Harris Detector: intuitive way…

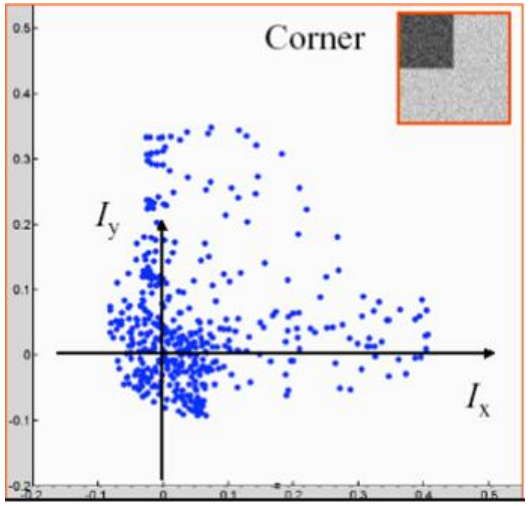
M is composed of derivatives combinations



Harris Detector: intuitive way…







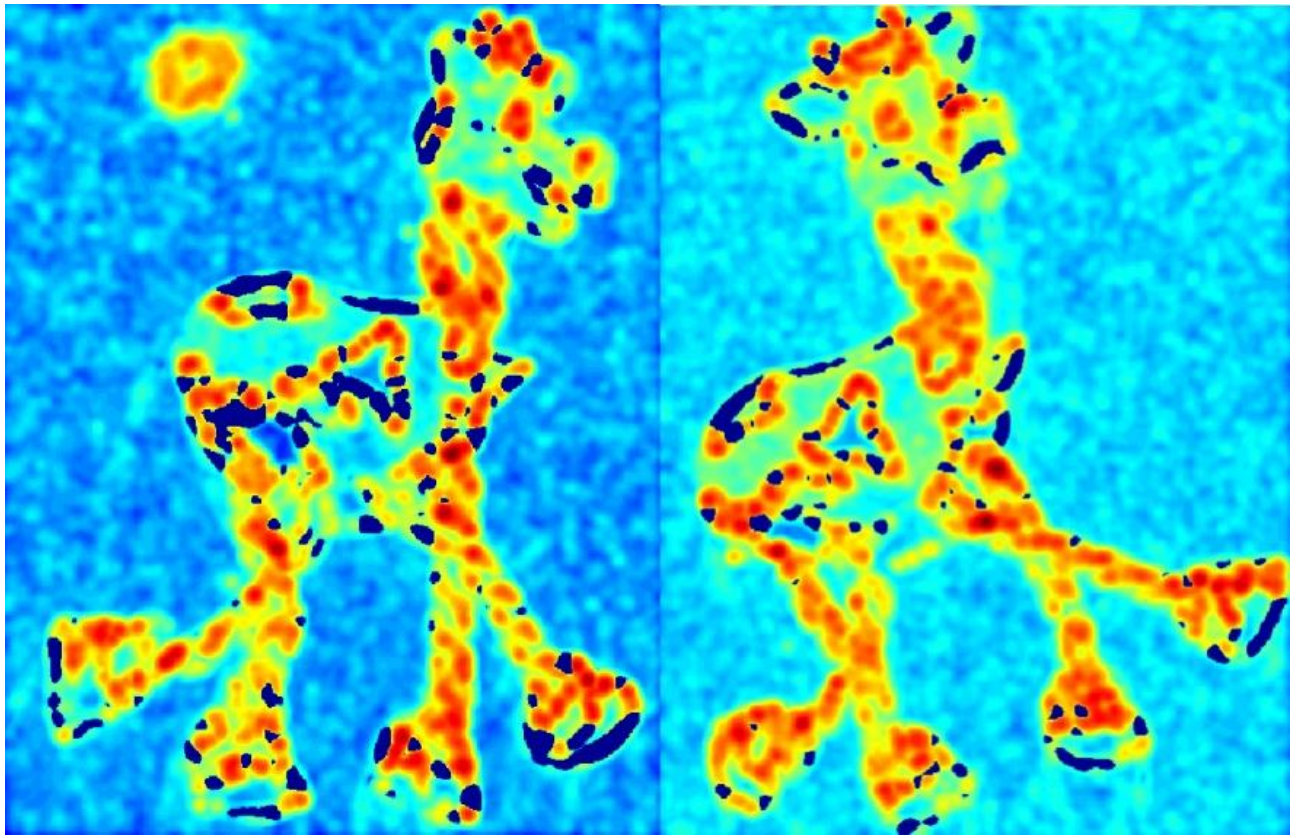
Harris Detector: Algorithm

- Compute corner response R
- Find points with large corner response: $R >$ threshold
- Take only the points of local maxima of R

Harris Detector: Workflow

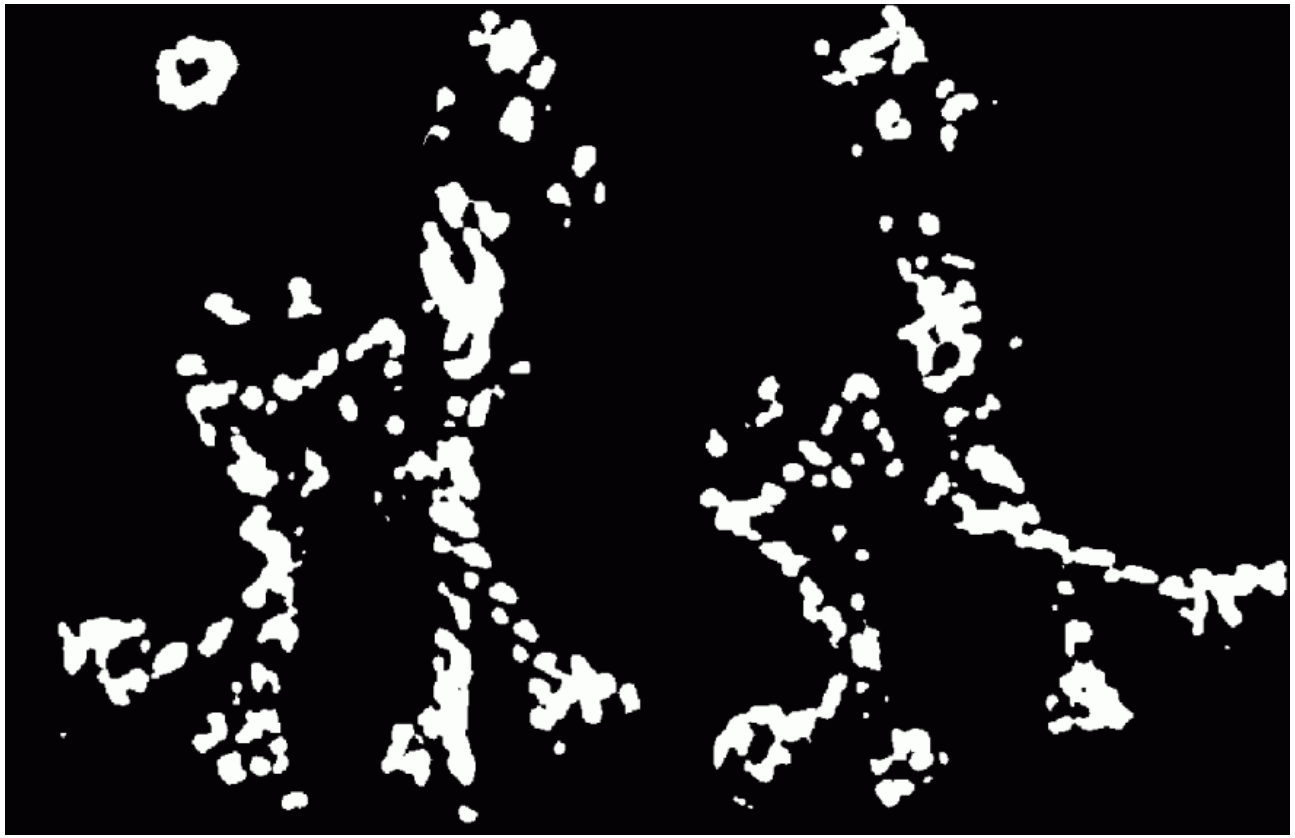
Harris Detector: Workflow

Compute corner response R



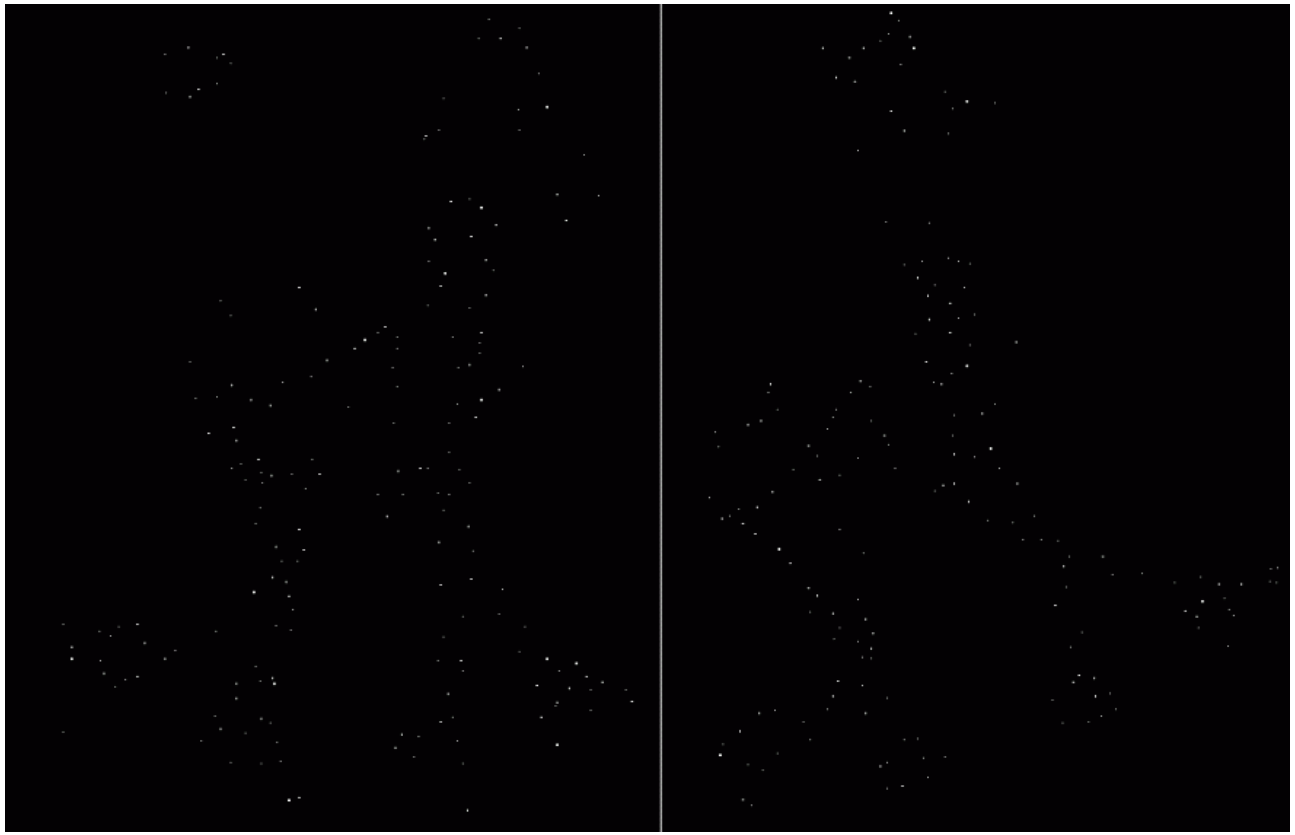
Harris Detector: Workflow

Find points with large corner response: R>threshold



Harris Detector: Workflow

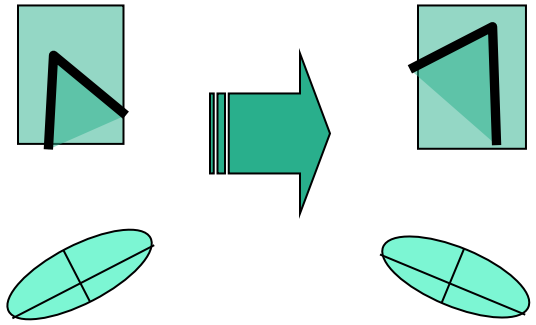
Take only the points of local maxima of R



Harris Detector: Workflow

Harris Detector: Some Properties

- Rotation invariance



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

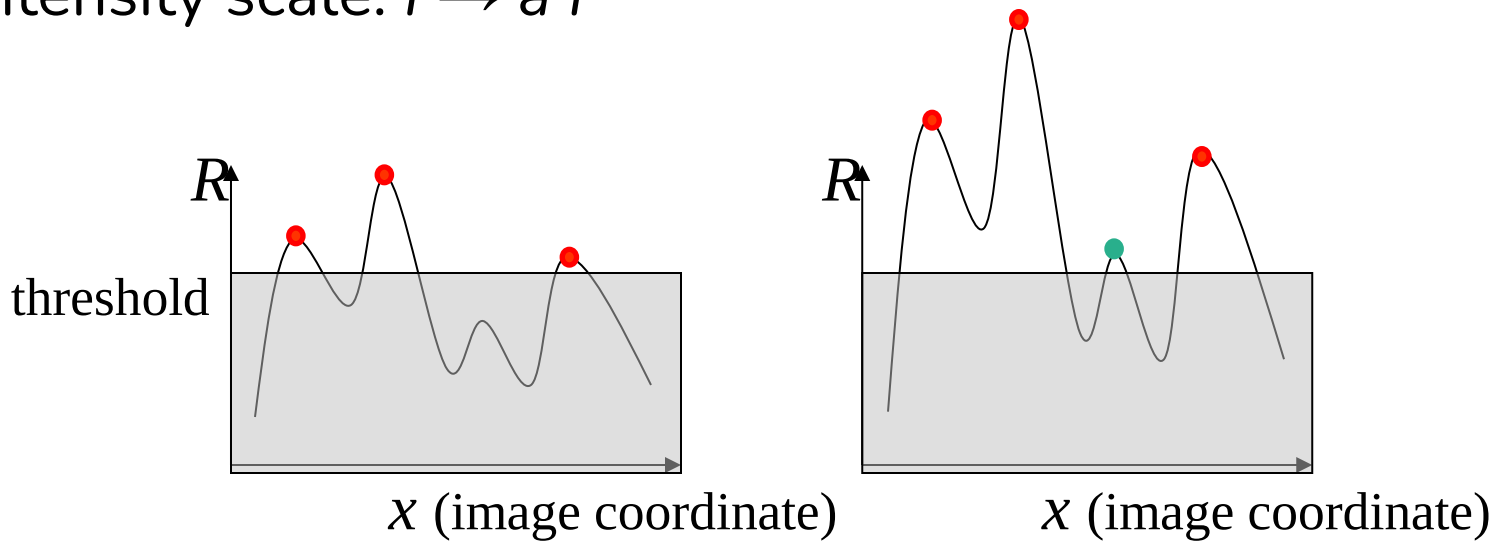
Corner response R is invariant to image rotation

Harris Detector: Some Properties

- Partial invariance to *change in* intensity

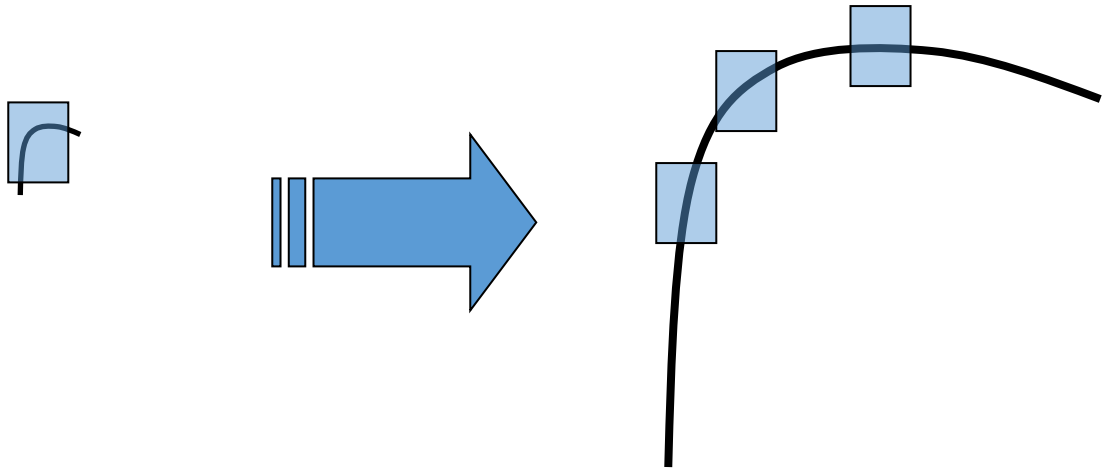
☑ Only derivatives are used
=> invariance to intensity shift $I \rightarrow I + b$

☒ Intensity scale: $I \rightarrow a\, I$



Harris Detector: Some Properties

- But: non-invariant to *geometric scale*!

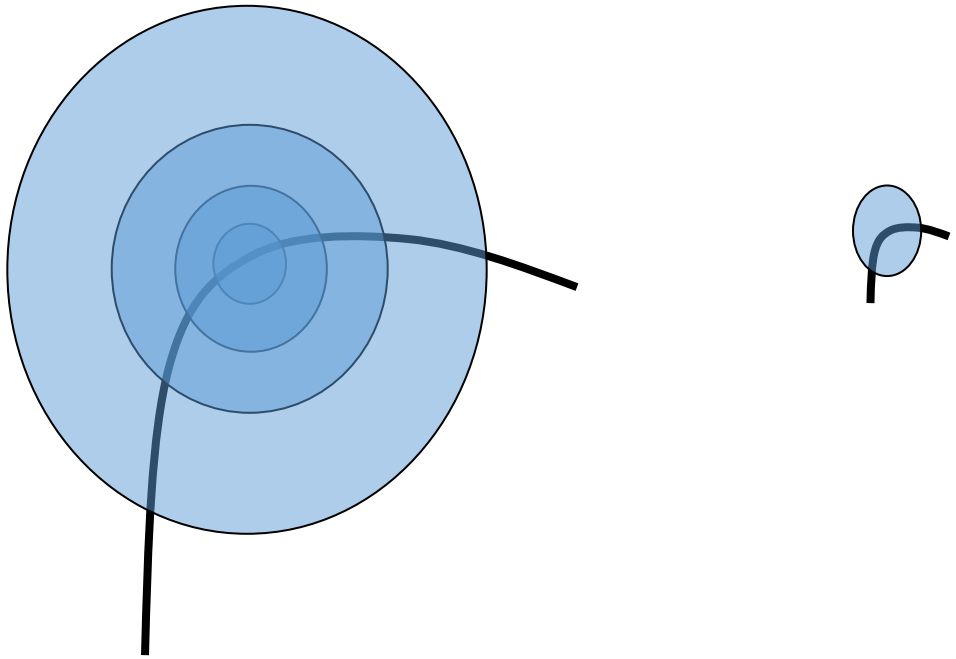


Corner

All points will be classified as edges!

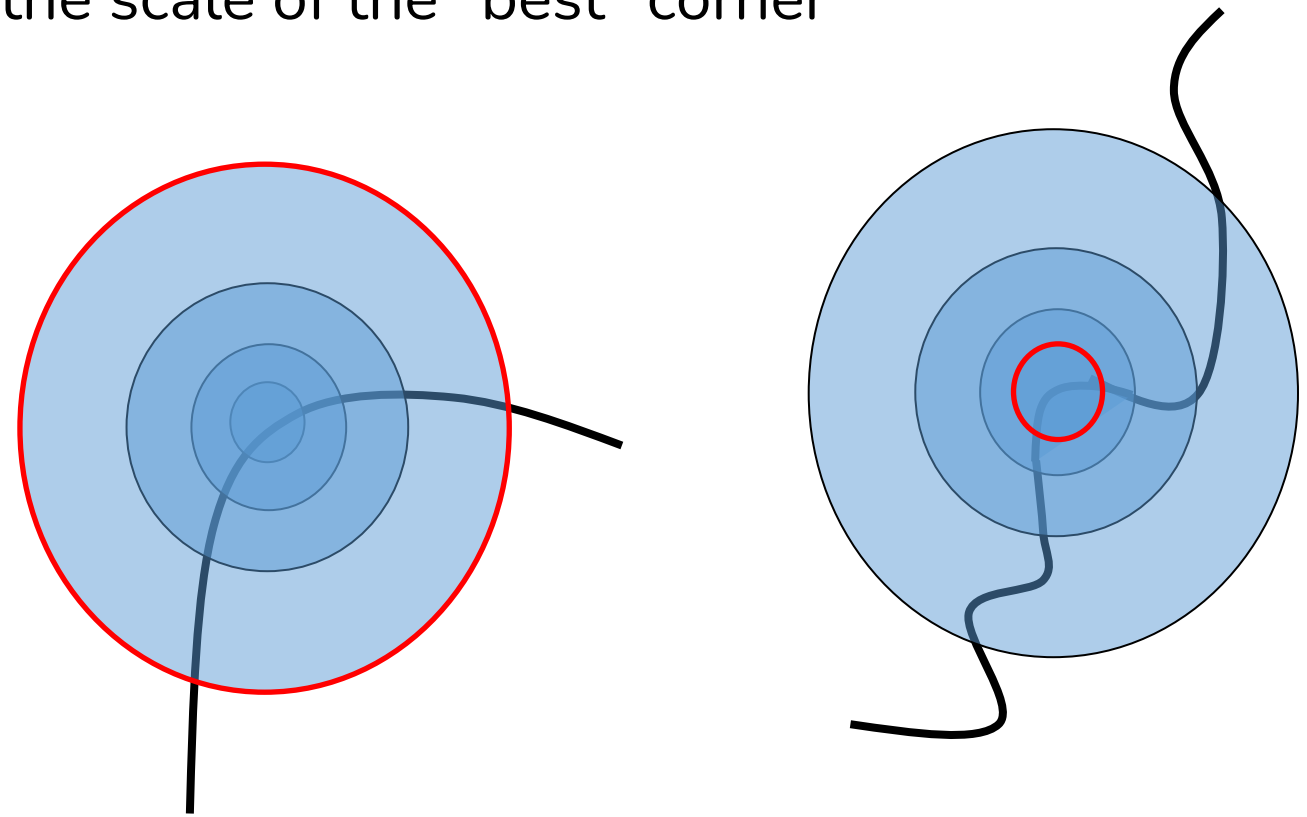
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point.
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

- **The problem**: how do we choose corresponding circles ***independently*** in each image?
- Choose the scale of the “best” corner



Affine transformations in 5 minutes

Next week:
MOPS & SIFT