Geometric Transformation

Introduction

- We will study a number of geometric transformations like translation, ,rotation, scaling and others
- Useful in a variety of applications
- They form the basis for image registration algorithms

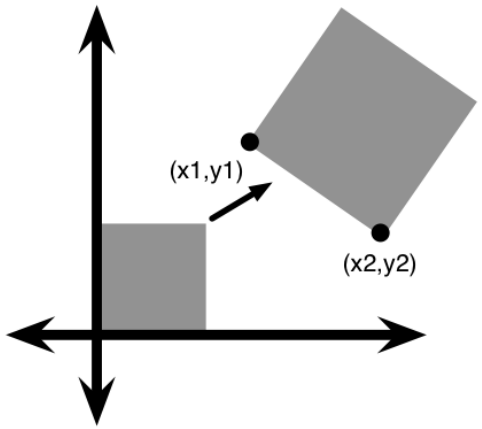
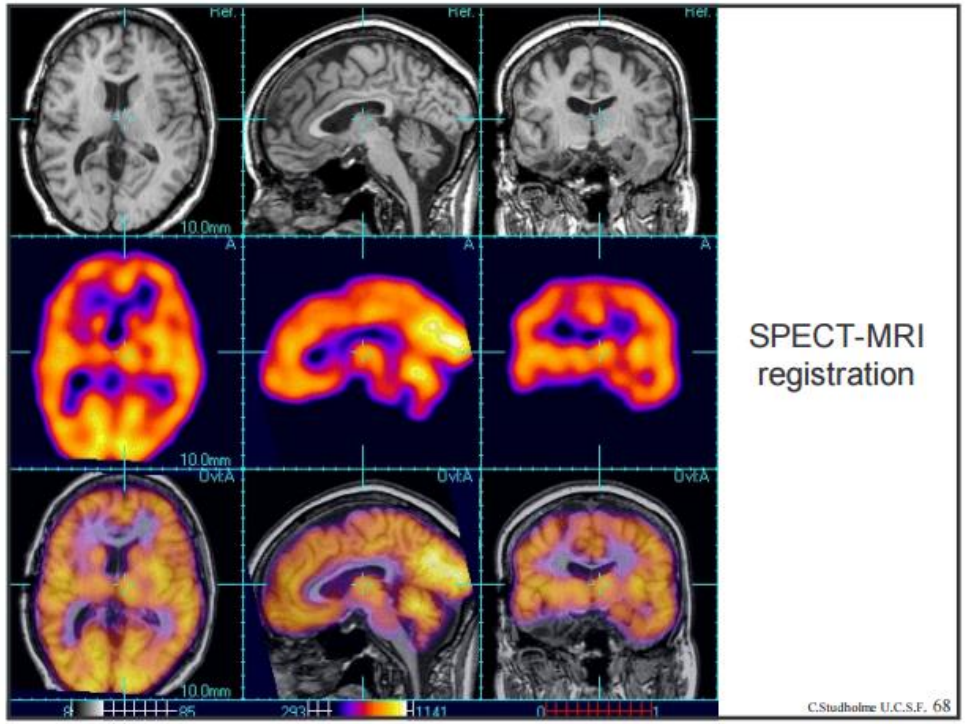


Image registration

- A subfield in image processing where the goal is to align two images.
- Image transformations are applied to an image to align it with the second image.



Geometric transformation

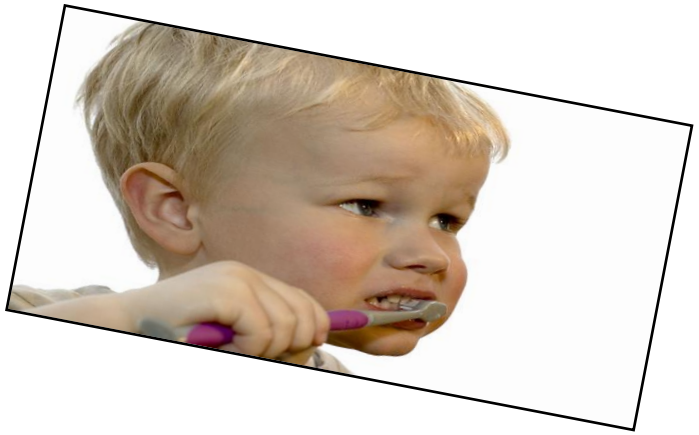
- Transformations can be applied to the whole image or to a region of interest in an image.
- We will assume that our transformation applies to the whole image or coordinates of the region of interest are already determined.

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

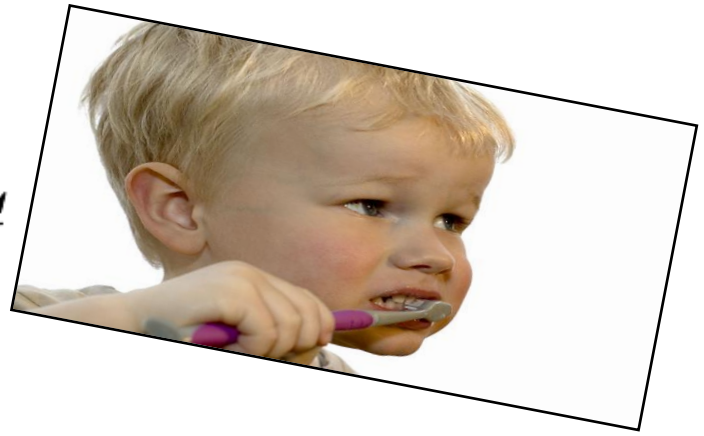
changes *range* of image function

F

Warping

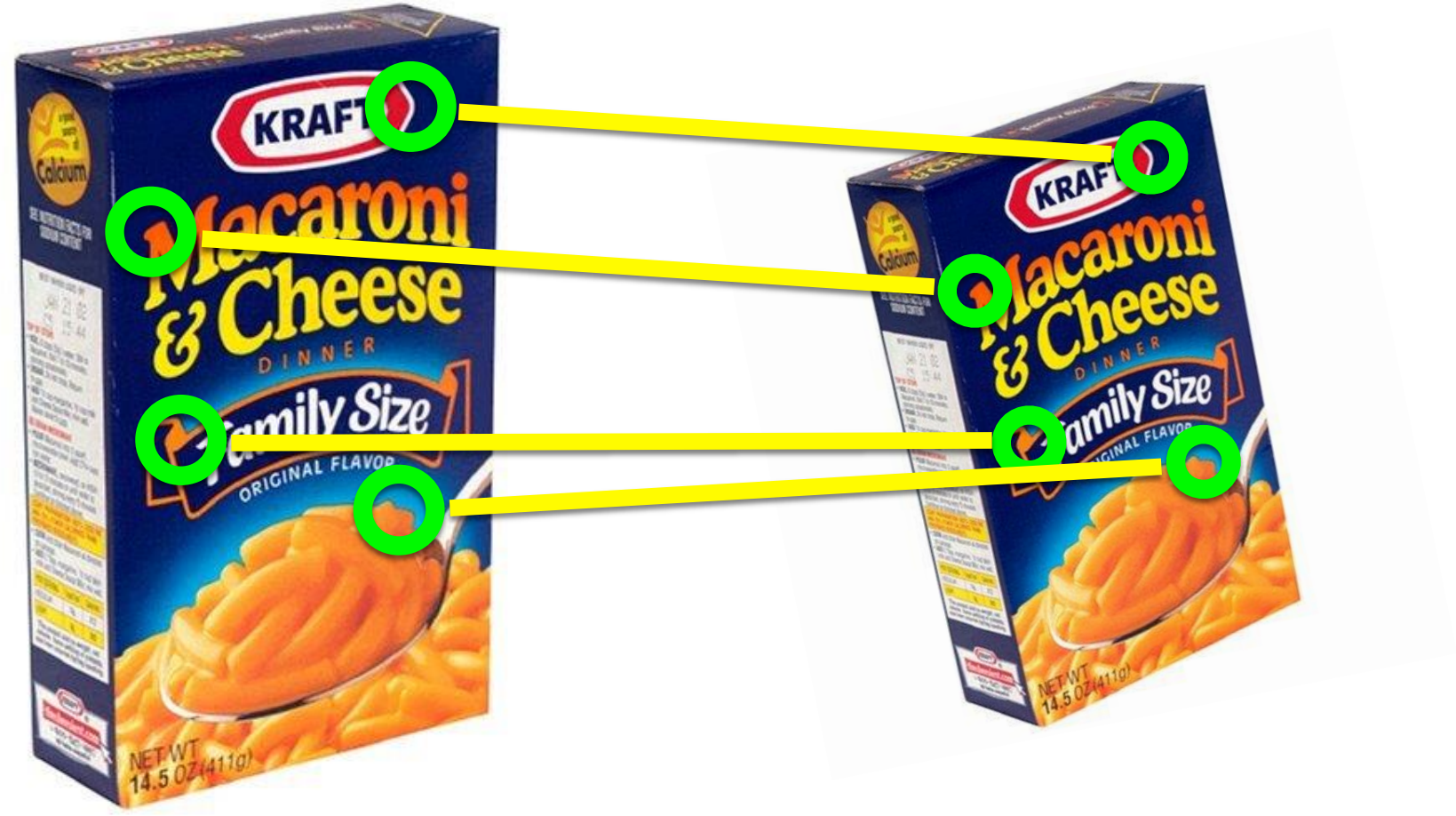
$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

G



changes *domain* of image function

Warping example: feature matching



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:

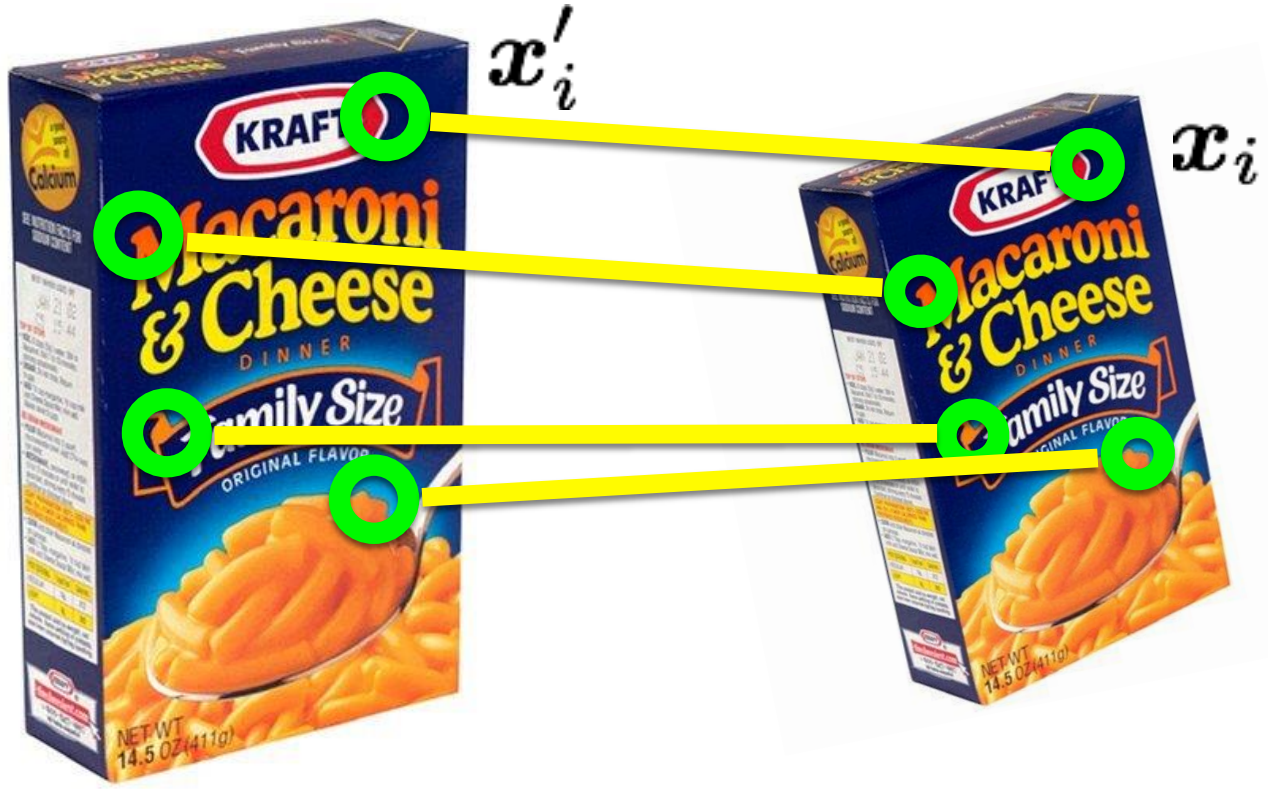
$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$

transformation function

parameters

find the best estimate of the parameters

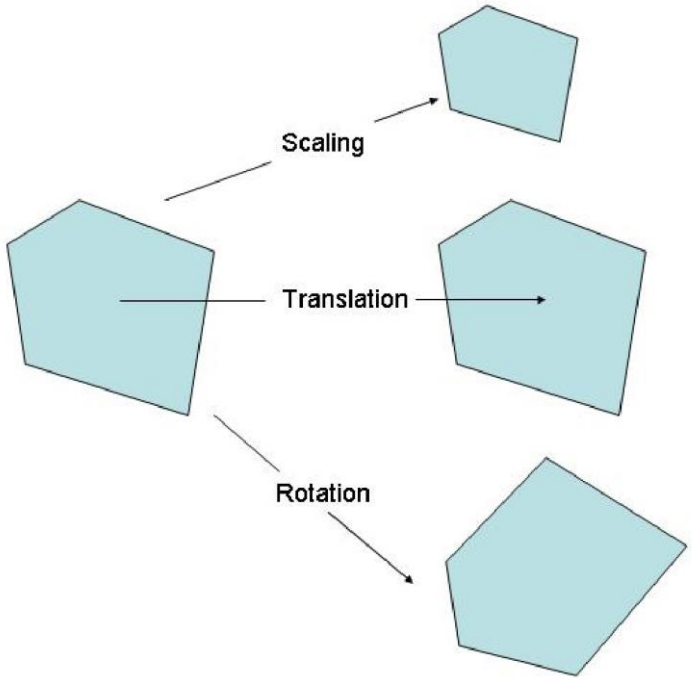
$$\boldsymbol{p}$$



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

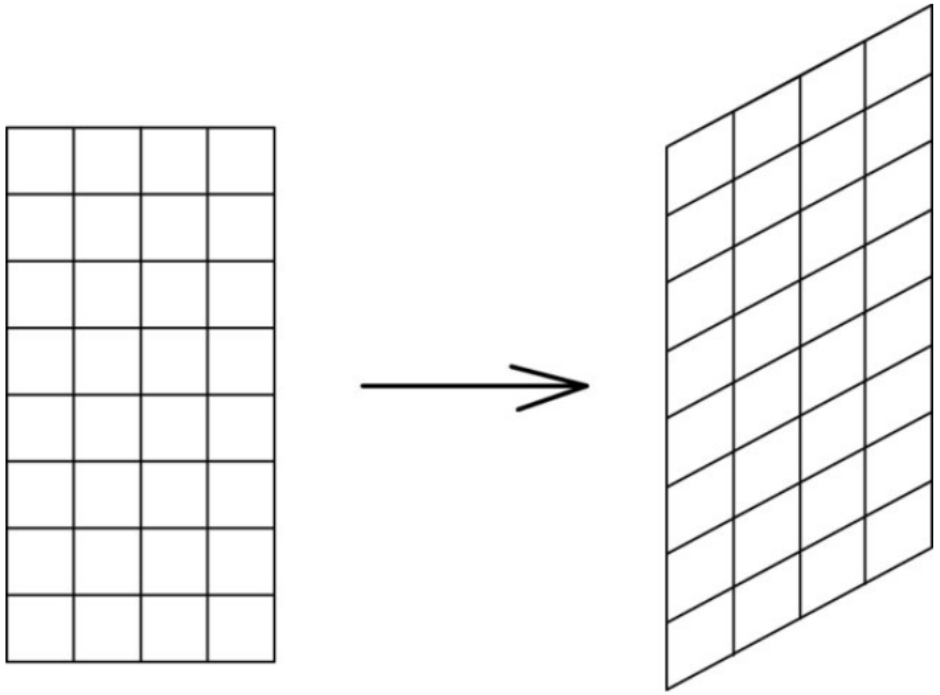
Rigid (Euclidian) and Similarity transformations

- Rigid transformations: Translation, and rotation
- Similarity transformations : Translation, rotation, and uniform scaling
- Preserve shapes

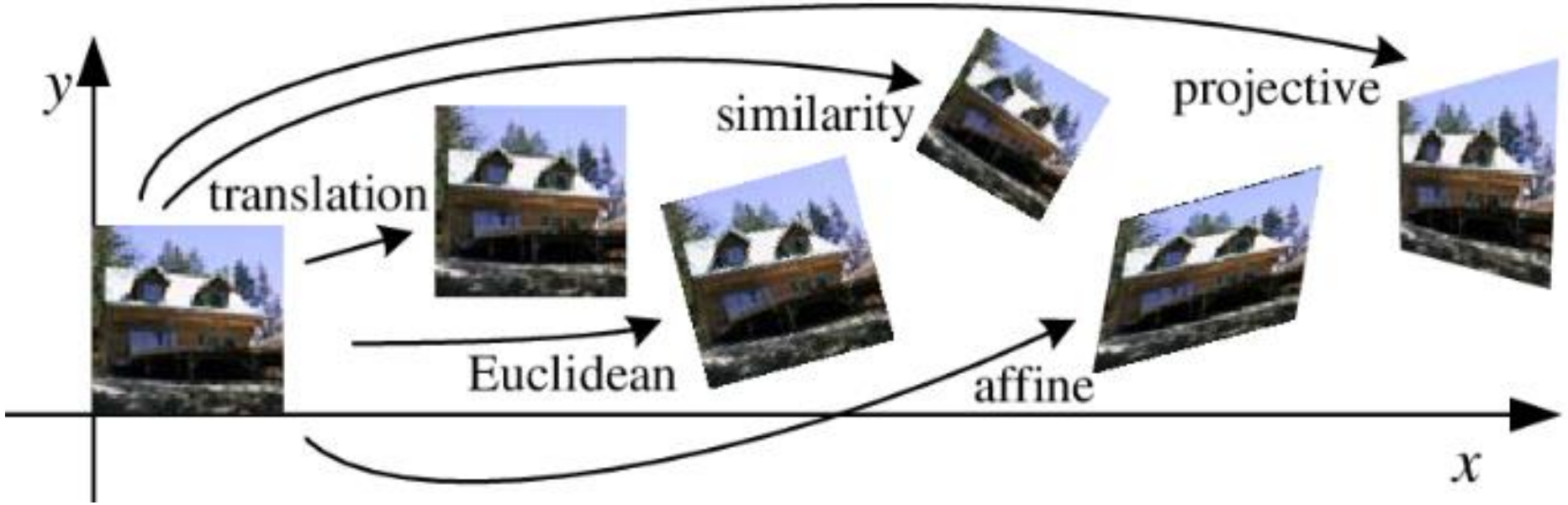


Affine Transformations

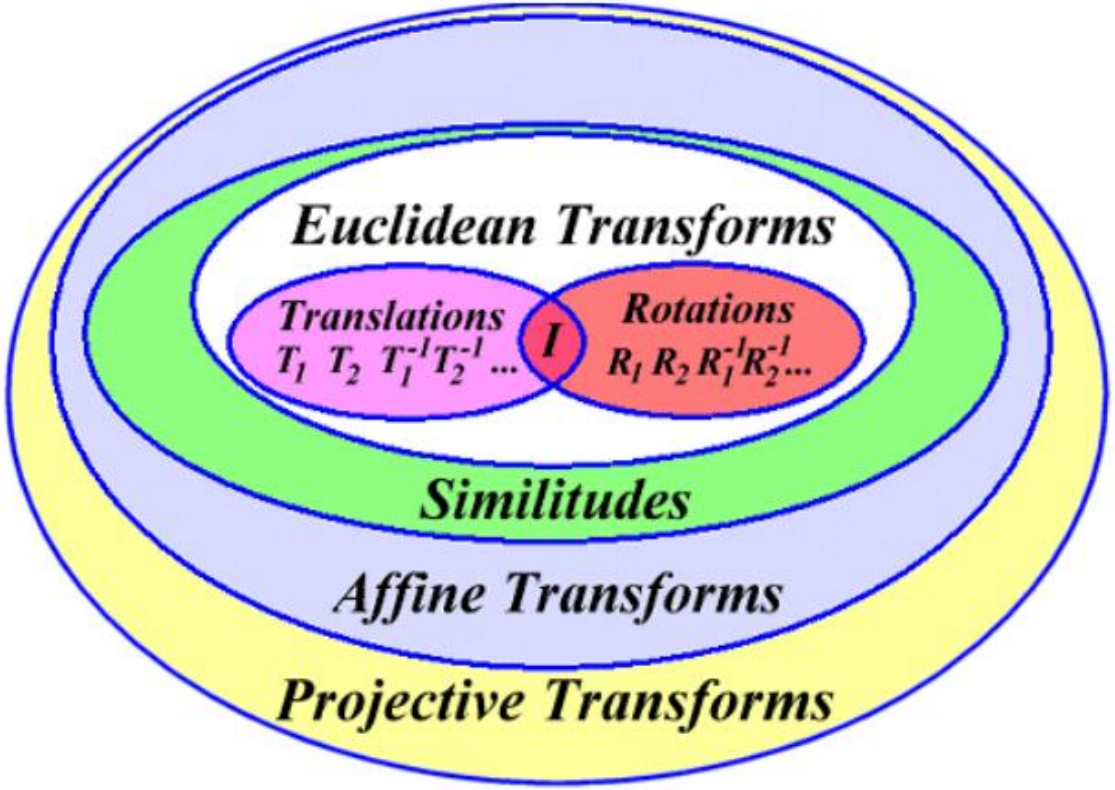
- Includes translation, rotation, scaling, and shearing.
- Straight lines remain straight, parallel lines remains parallel.
- Rectangles becomes parallelograms under shearing operations



2D transformations



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

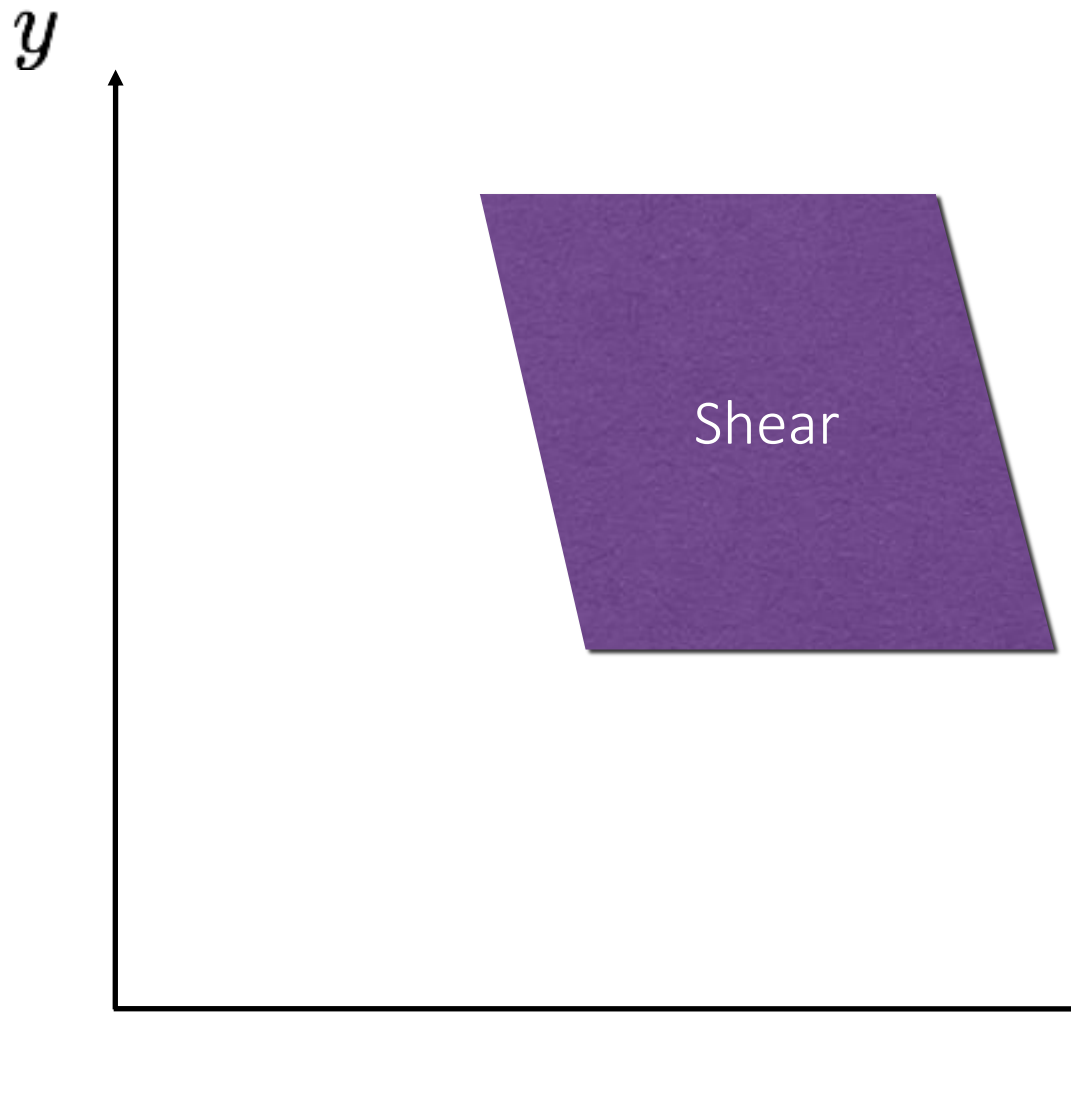
$$x' = ax$$

$$y' = by$$

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



Shear

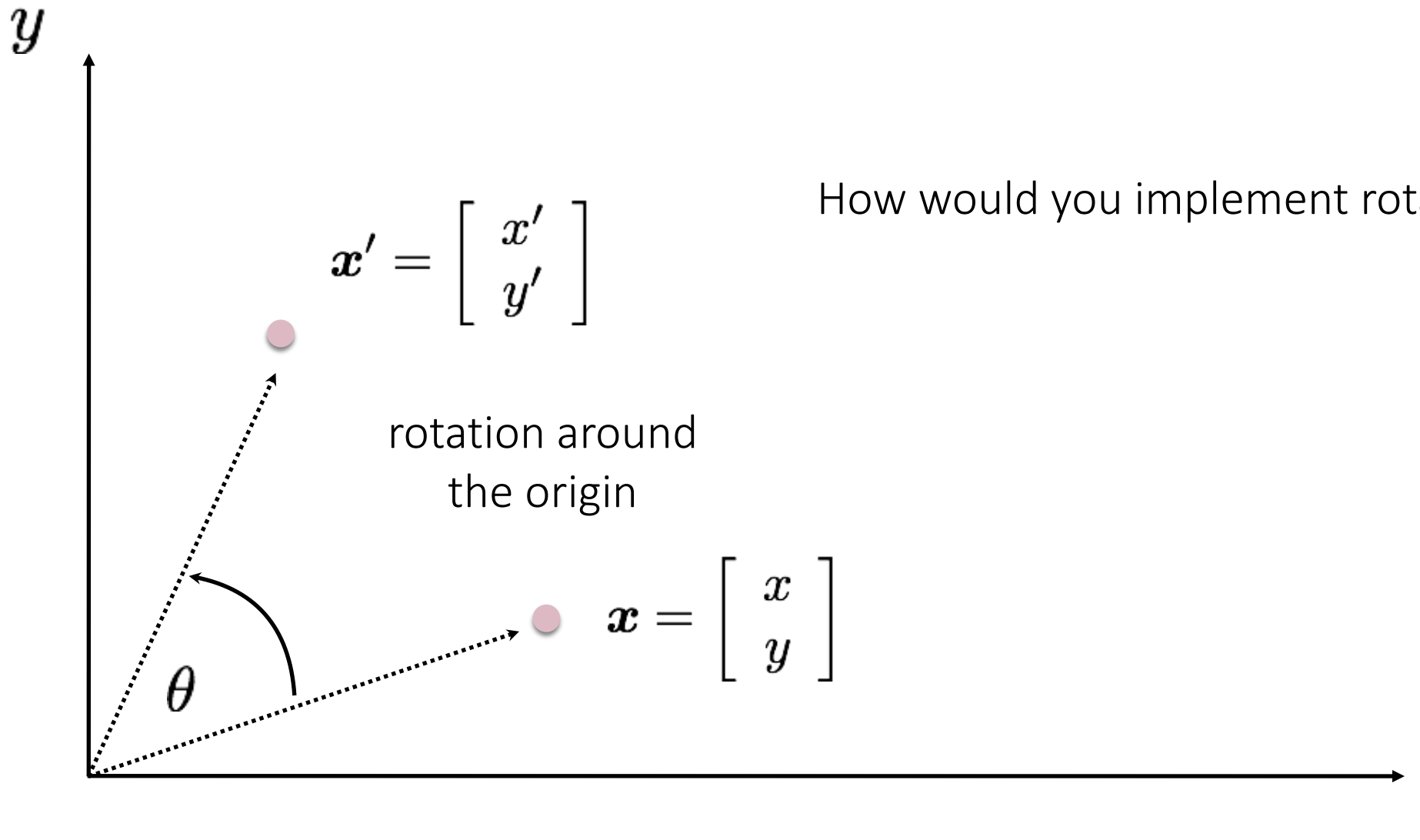
$$x' = x + a \cdot y$$

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

or in matrix form:

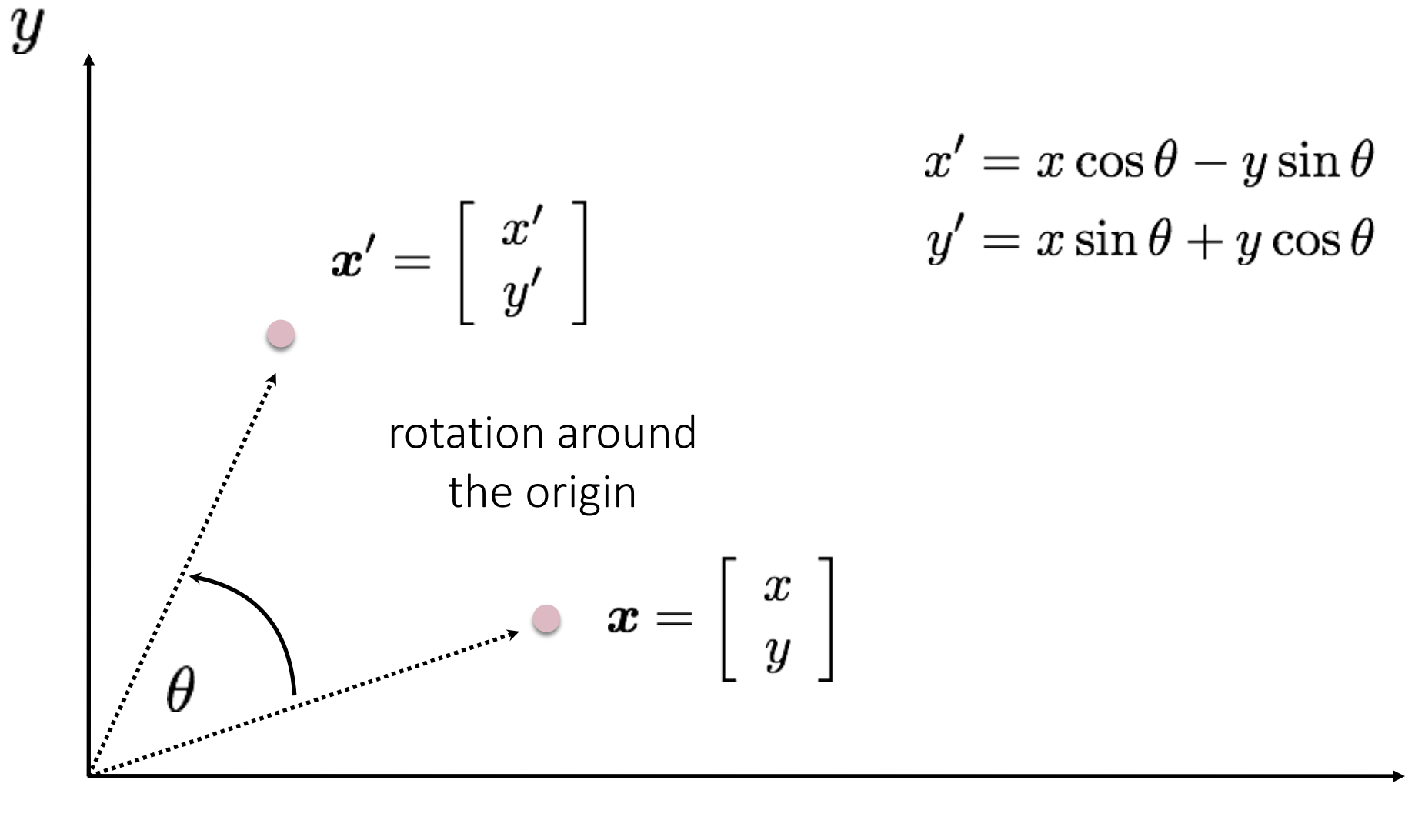
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D planar transformations



How would you implement rotation?

2D planar transformations



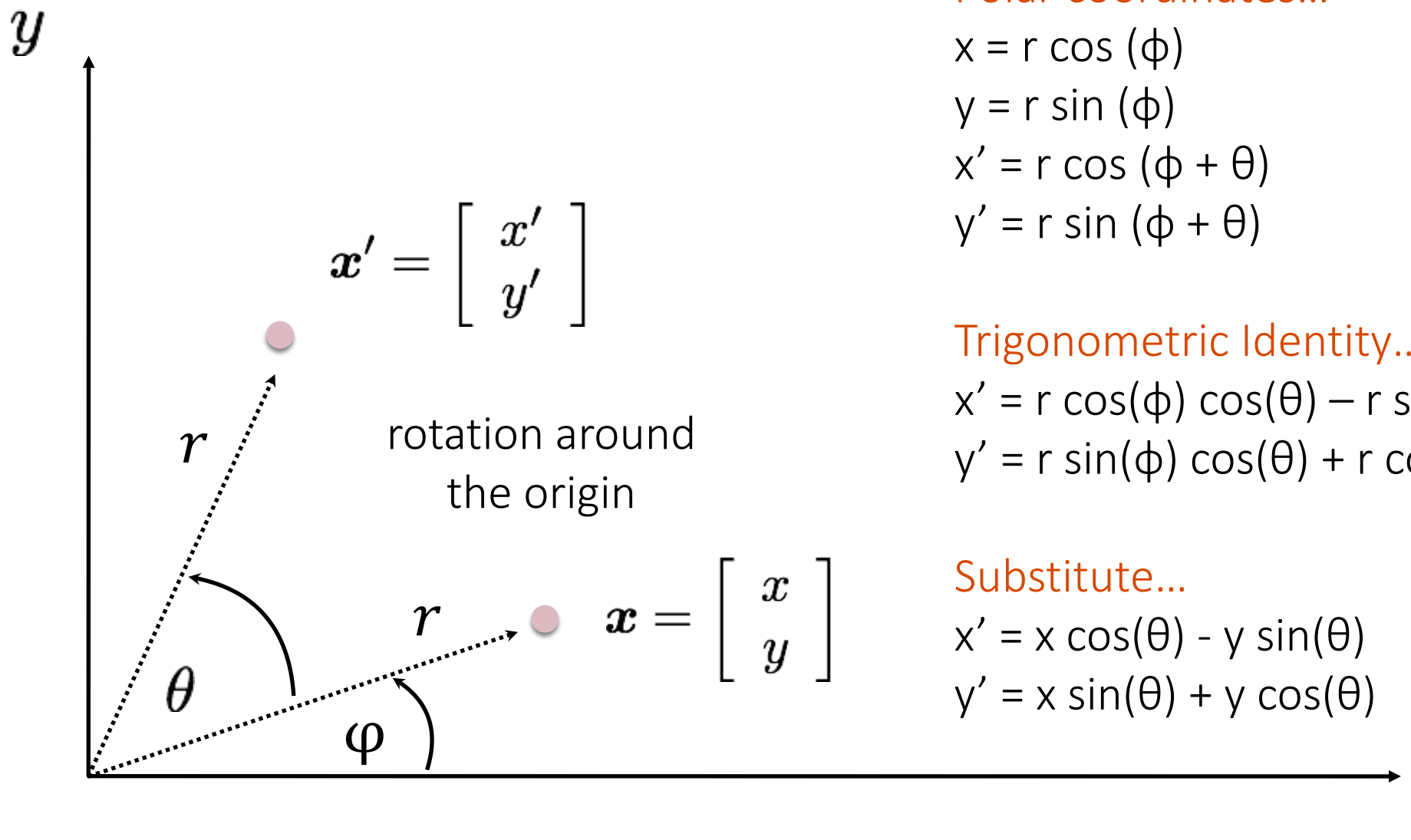
$$\boldsymbol{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

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

rotation around
the origin

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

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)

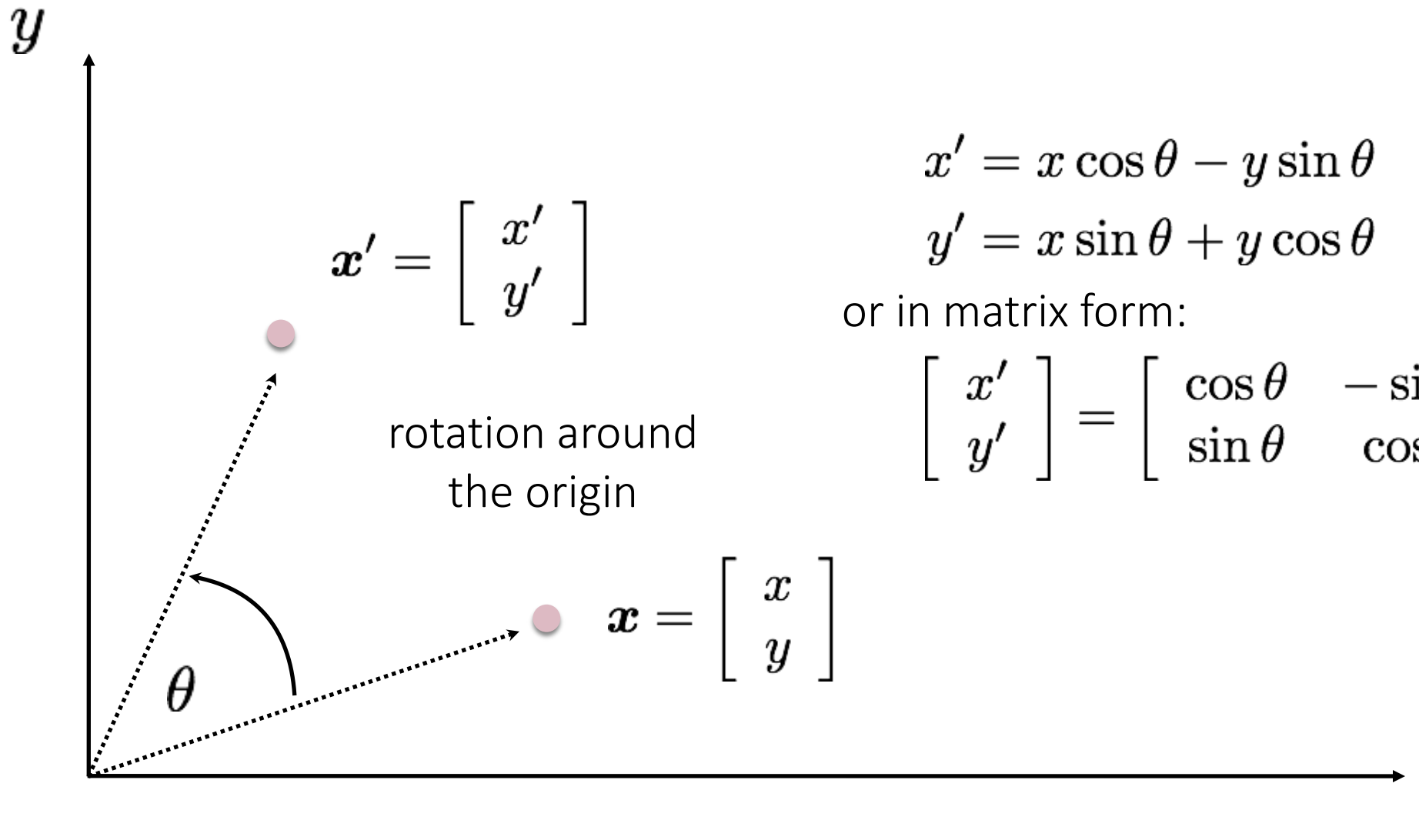
y’ = r sin(ϕ) cos(θ) + r cos(ϕ) sin(θ)

Substitute…

x’ = x cos(θ) - y sin(θ)

y’ = x sin(θ) + y cos(θ)

2D planar transformations



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

rotation around the origin

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

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

or in matrix form:

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

Transformations operate on coordinates

- Intensities remain fixed (at least ideally)
- We need a description of image coordinates
- Point Distribution Matrix is a method to represent a list of coordinates in an image

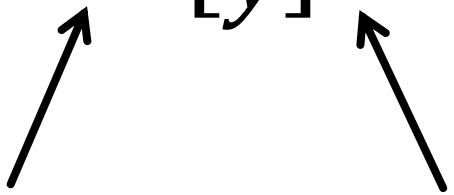
$$S = \begin{bmatrix} x_1 & x_1 & x_1 & \dots & x_M \\ y_1 & y_2 & y_3 & \dots & y_N \end{bmatrix}$$

- Each column in the PDM correspond to the coordinates of one point.
- The pairs $(x_1, y_1), (x_1, y_2), \dots, (x_1, y_N), \dots, (x_2, y_1), (x_2, y_2), \dots (x_2, y_N), \dots, (x_M, y_1), (x_M, y_2), \dots, (x_M, y_N)$, represent all grid coordinates.
- Matlab/Python command meshgrid

2D transformations

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

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



parameters p point $\boldsymbol{x}$

2D 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

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

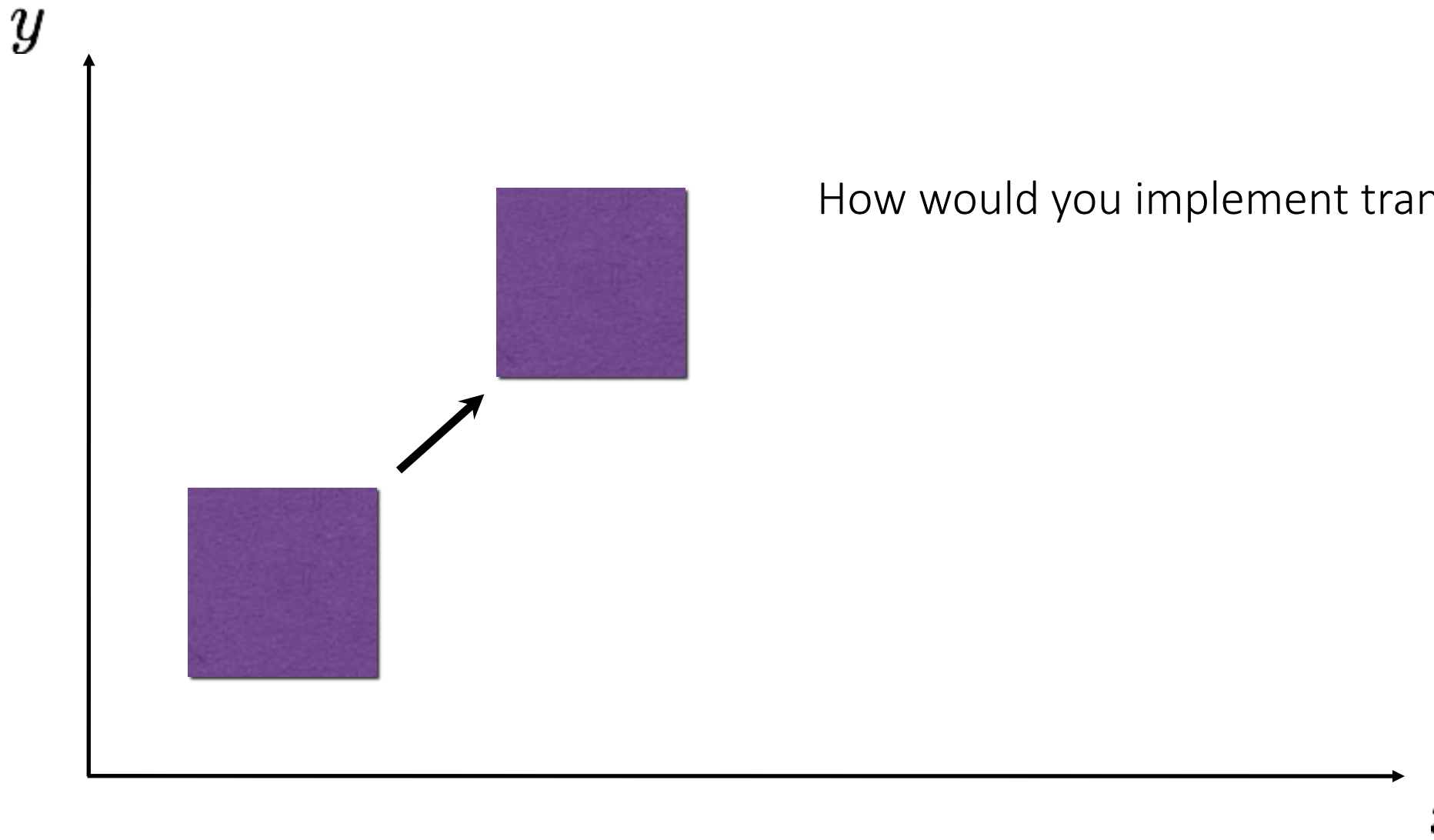
Shear

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

Identity

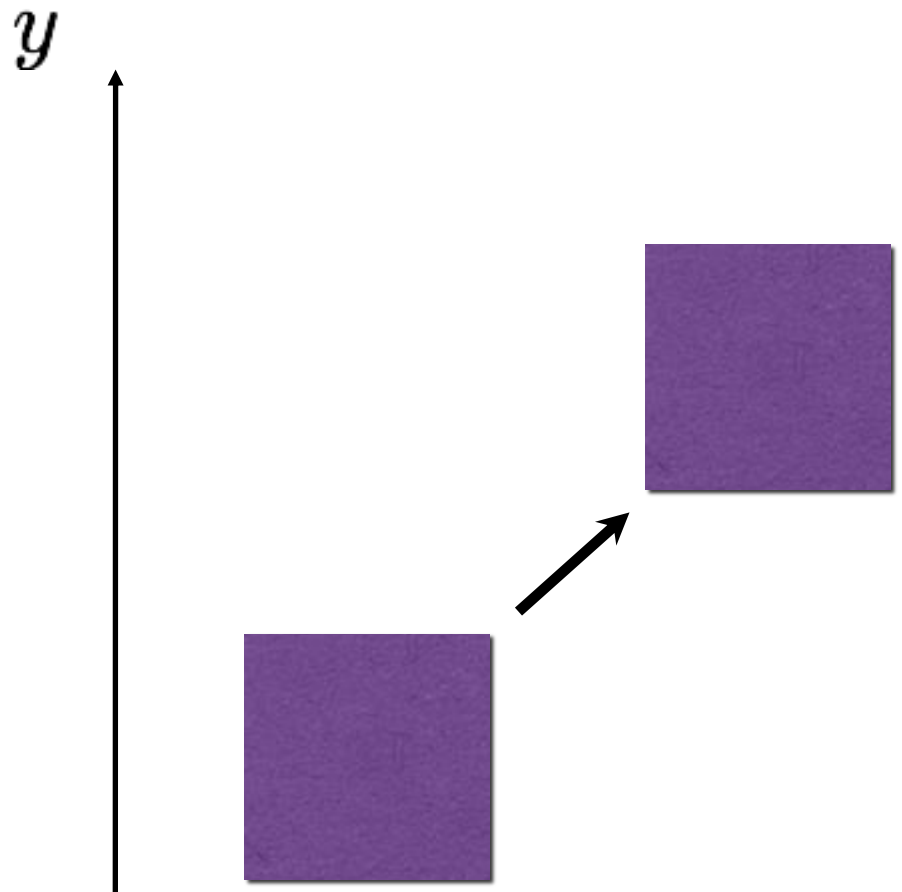
$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D Translation



How would you implement translation?

2D Translation

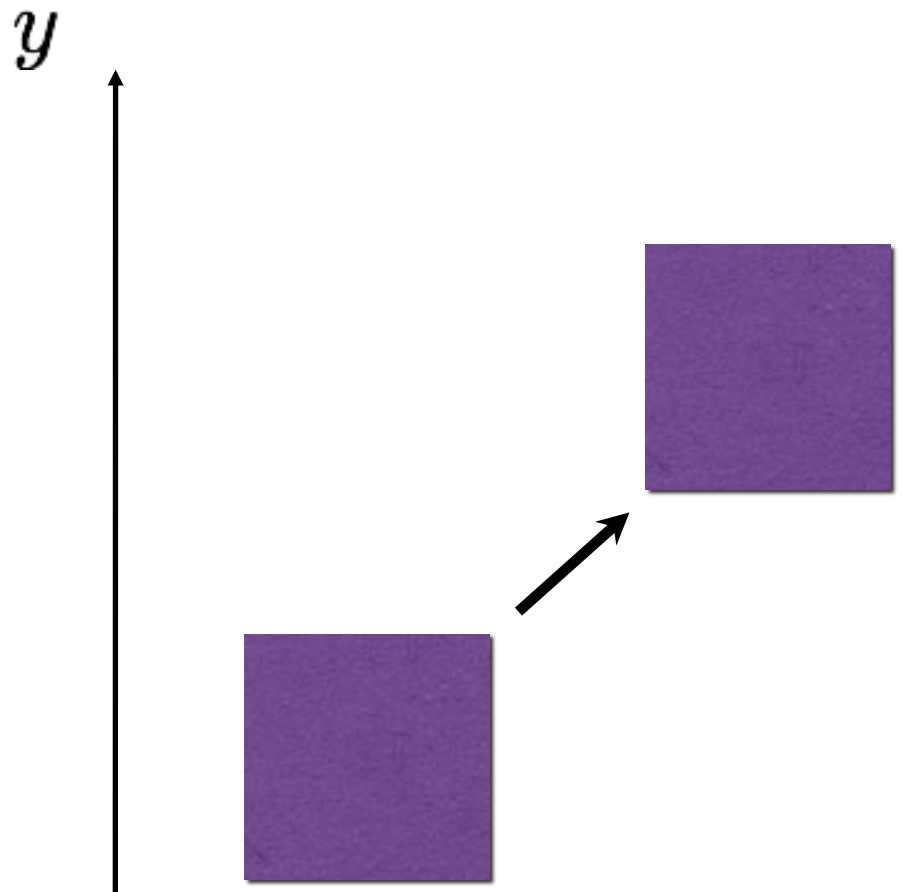


$$x' = x + t_x$$
$$y' = y + t_x$$

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.

Homogeneous coordinates

heterogeneous coordinates

homogeneous coordinates

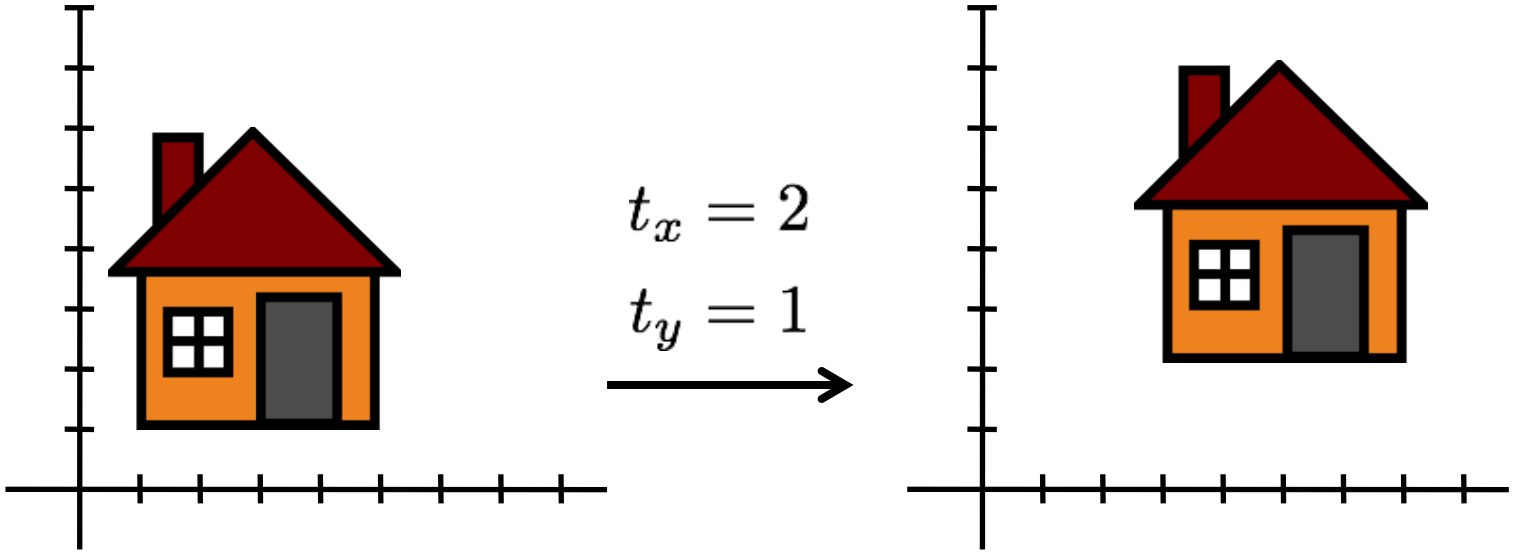
$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

add a 1 here

- Represent 2D point with a 3D vector

2D translation using homogeneous coordinates

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



2D transformations in homogeneous 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 homogeneous 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

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

scaling

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

rotation

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

shearing

2D transformations in homogeneous 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

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

scaling

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

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 homogeneous 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

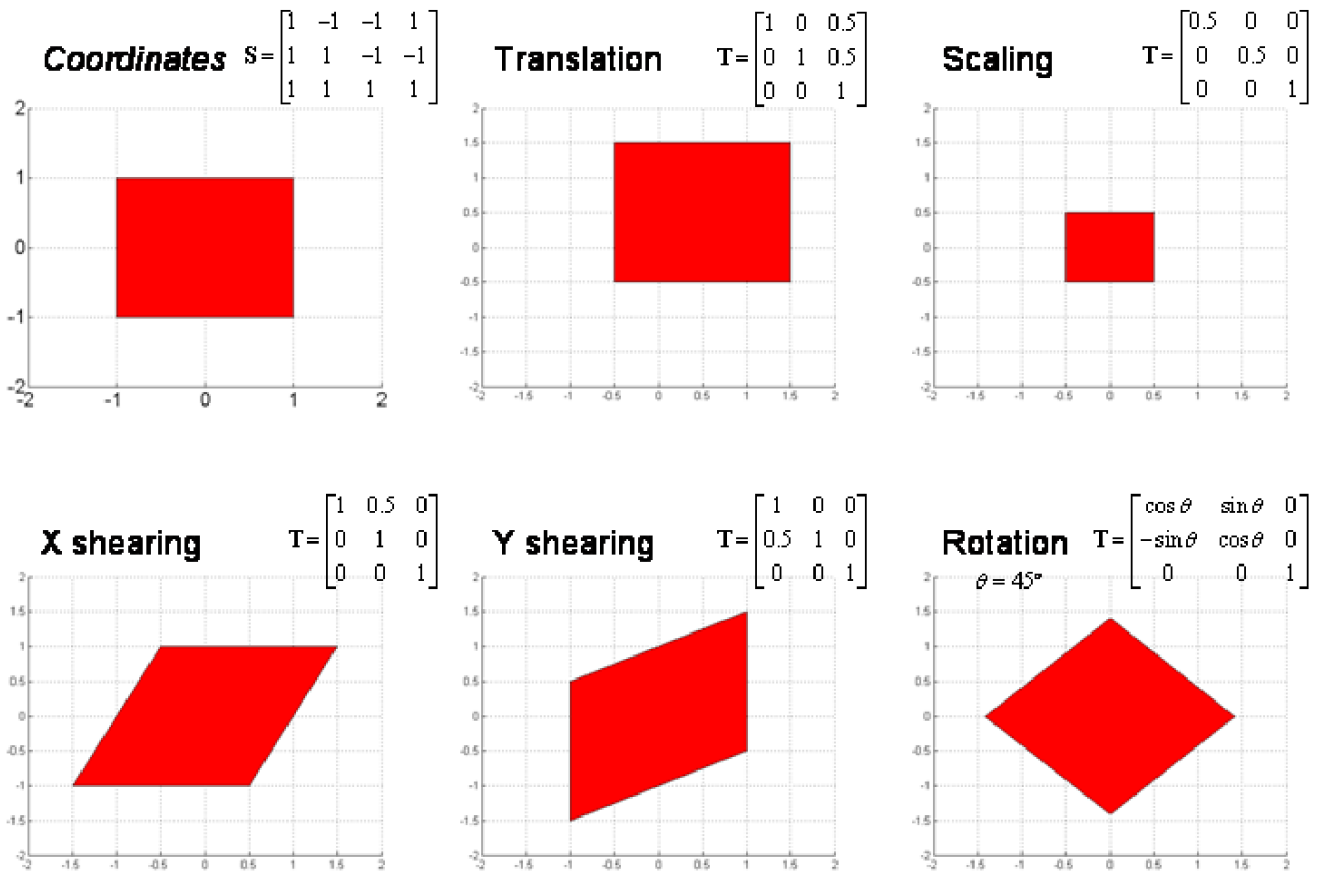
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

Transformation Matrix



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:

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

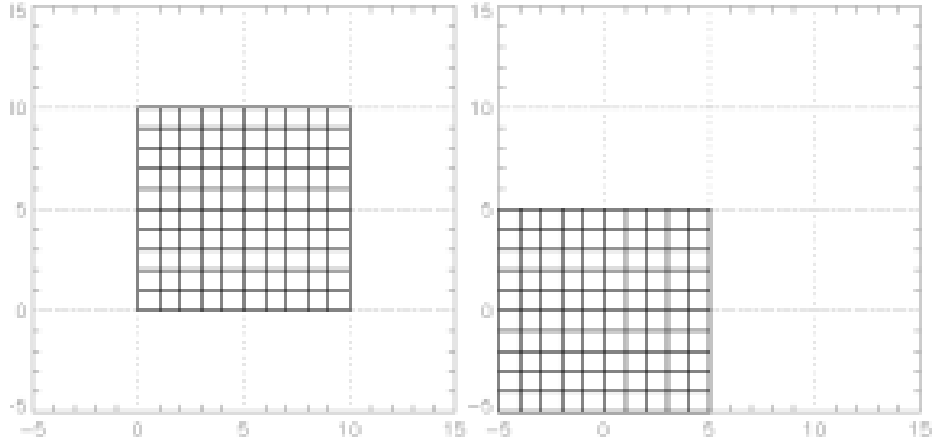
p' = translation(t_x,t_y) rotation(θ) scale(s,s) **p**

Where is the Center?

- Previous transformation matrices assumes that the center of the object of interest is the coordinate (0, 0).
- The origin can be shifted to the center by subtracting the center coordinate from the PDM matrix
- Or

  a. translate the image to the center first

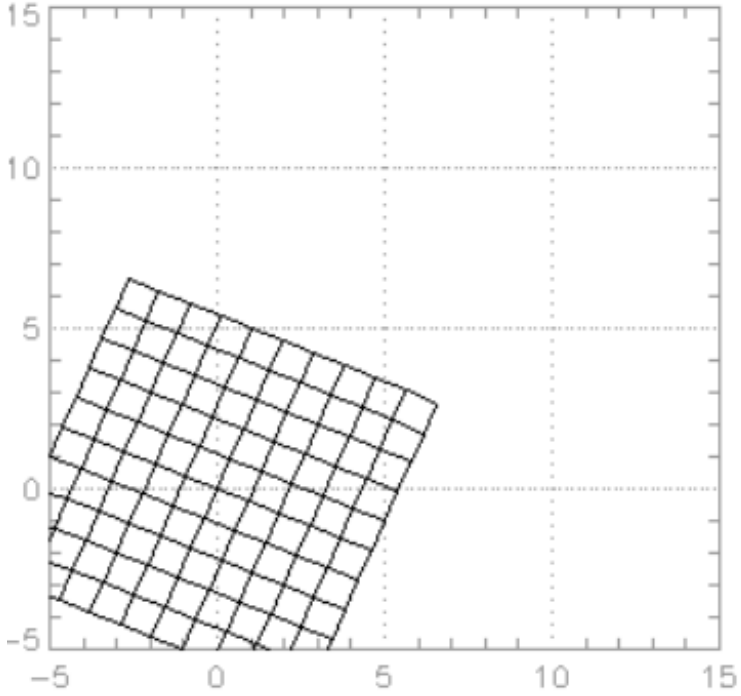
$$\mathbf{T}_1 = \begin{bmatrix} 1.00 & 0.00 & -5.00 \\ 0.00 & 1.00 & -5.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$



Where is the Center?

b. Apply desired function e.g. rotation

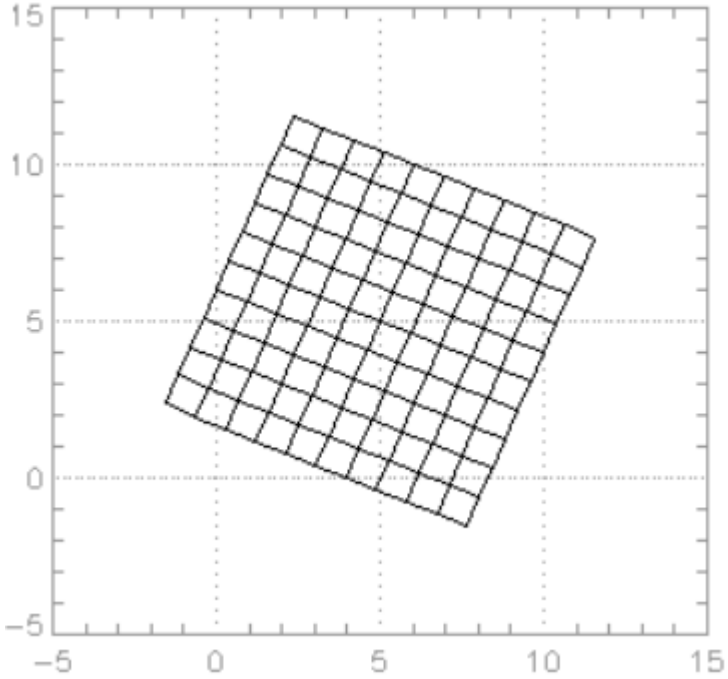
$$\mathbf{T}_2 = \begin{bmatrix} 0.92 & 0.39 & 0.00 \\ -0.39 & 0.92 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$



Where is the Center?

c. Translate back to the original location

$$\mathbf{T}_3 = \begin{bmatrix} 1.00 & 0.00 & 5.00 \\ 0.00 & 1.00 & 5.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

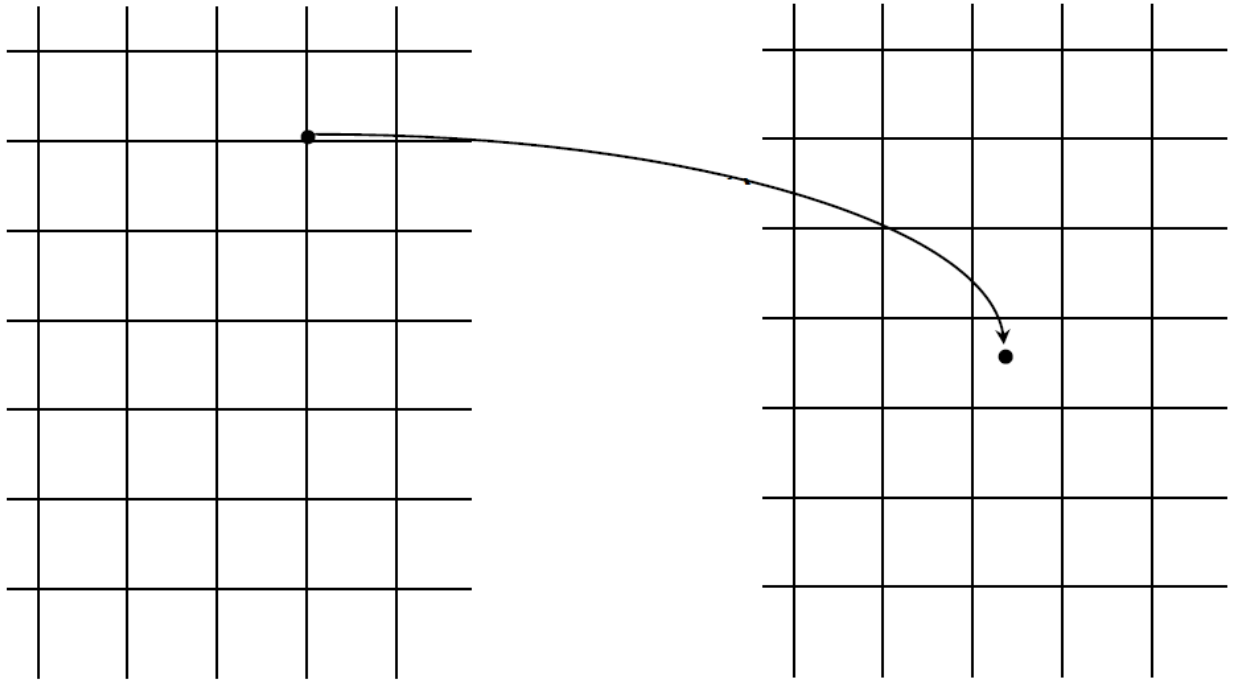


Moving intensities

- So far, we have seen different methods for updating coordinate values to correspond to different geometrical transformations (translation, rotation, scaling….)
- In this setting we have a list of coordinates **S** and a list of transformed coordinates $\mathbf{S}^{\mathbf{Transform}}$.
- We should next, move the intensities from the older coordinates in **S** to the new coordinates in $\mathbf{S}^{\mathbf{Transform}}$.
- Interpolation is often used to compute the new intensities in $\mathbf{S}^{\mathbf{Transform}}$

Interpolation

- Estimating intensities at unknown points using available data points.



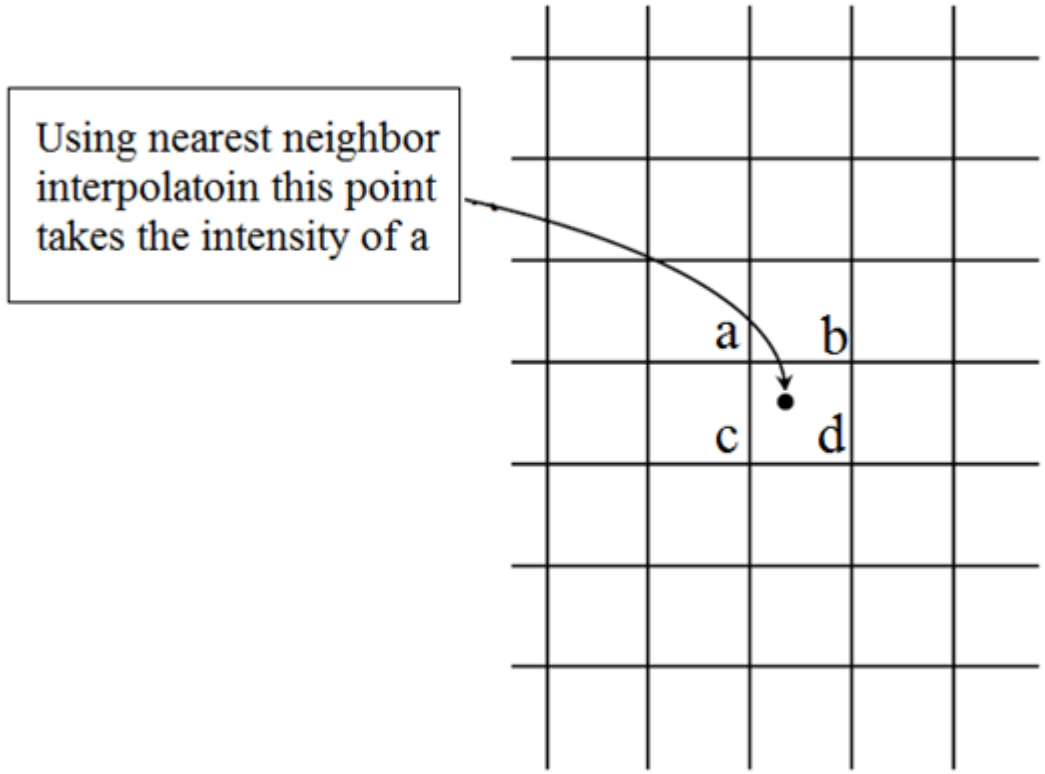
Interpolation - Nearest Neighbor Interpolation

- Nearest neighbor interpolation

$$\hat{I}(x, y) \equiv I(\text{ROUND}(x), \text{ROUND}(y))$$

- The equation assumes that a bound checking process is implemented to make sure the round operation does not generate coordinates outsides image boundaries.

Interpolation - Nearest Neighbor Interpolation



Bilinear interpolation

- A more accurate approach is **bilinear interpolation**.
  - It is a 2D extension of 1D linear interpolation.

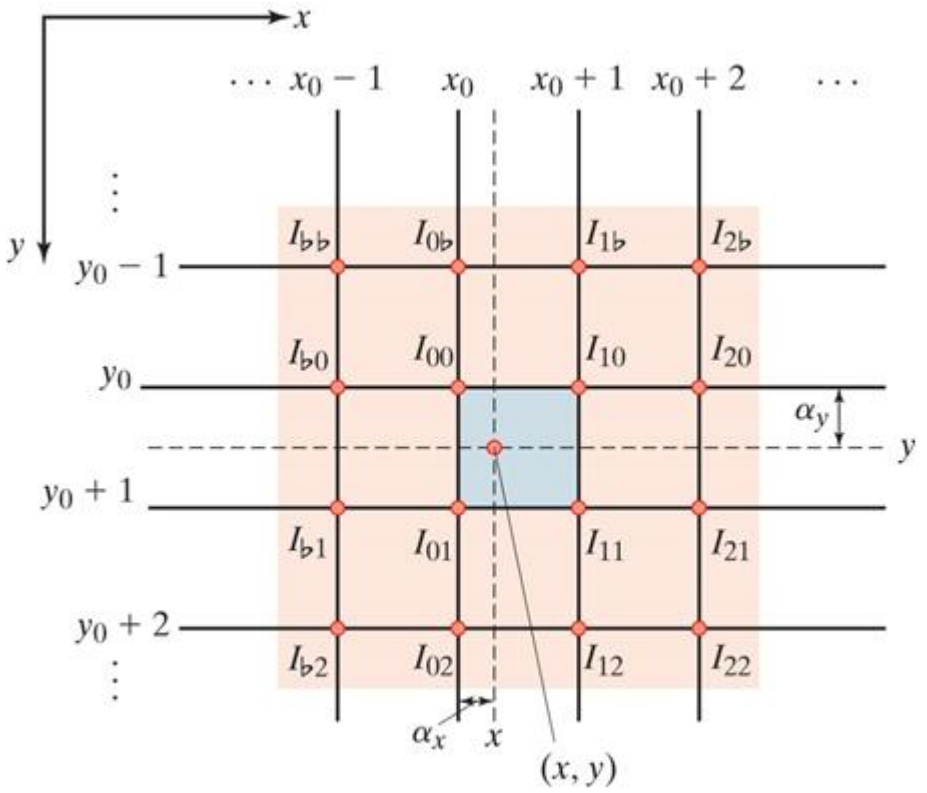
- The interpolated value is the weighted average of the four nearby pixels:

$$\hat{I}(x,y) = \overline{\alpha}_x\overline{\alpha}_y I_{00} + \alpha_x\overline{\alpha}_y I_{10} + \overline{\alpha}_x\alpha_y I_{01} + \alpha_x\alpha_y I_{11}$$

Bilinear interpolation

$$\hat{I}(x, y) = \overline{\alpha}_x \overline{\alpha}_y I_{00} + \alpha_x \overline{\alpha}_y I_{10} + \overline{\alpha}_x \alpha_y I_{01} + \alpha_x \alpha_y I_{11}$$

- *where* $\overline{\alpha_x} = 1 - \alpha_x$ and $\overline{\alpha_y} = 1 - \alpha_y$

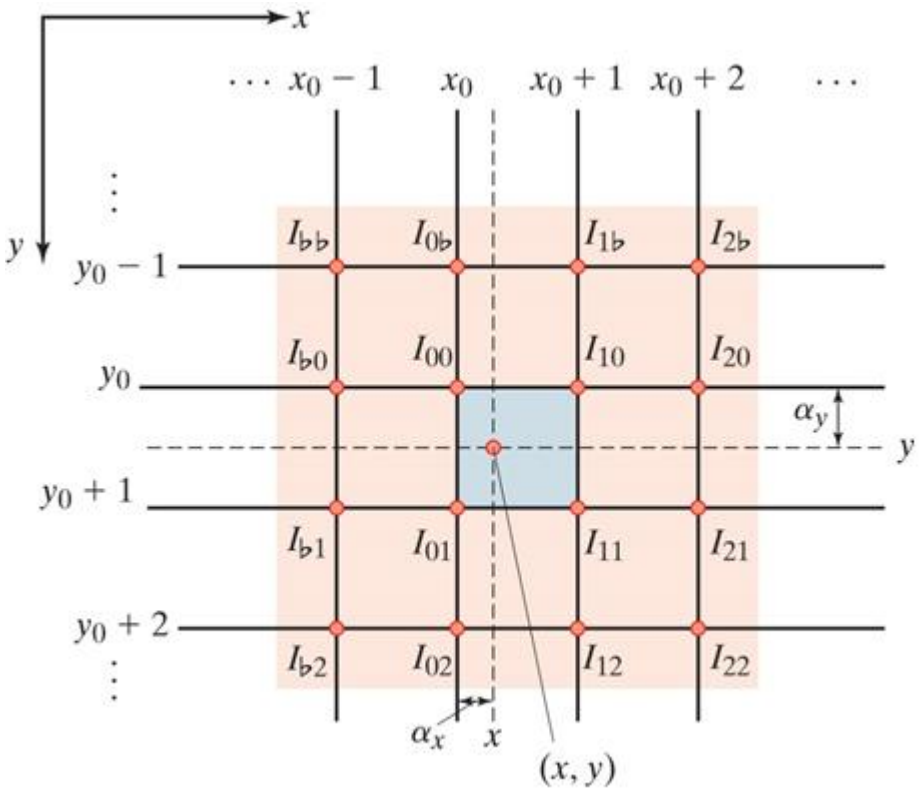


Bilinear interpolation

Given the following image, use bilinear interpolation to compute the value at (1.2, 0.5). Assume that the Upper left coordinate is (0,0).

$$I = \begin{bmatrix} 232 & 177 & 82 & 7 \\ 241 & 18 & 152 & 140 \\ 156 & 221 & 67 & 3 \end{bmatrix}$$

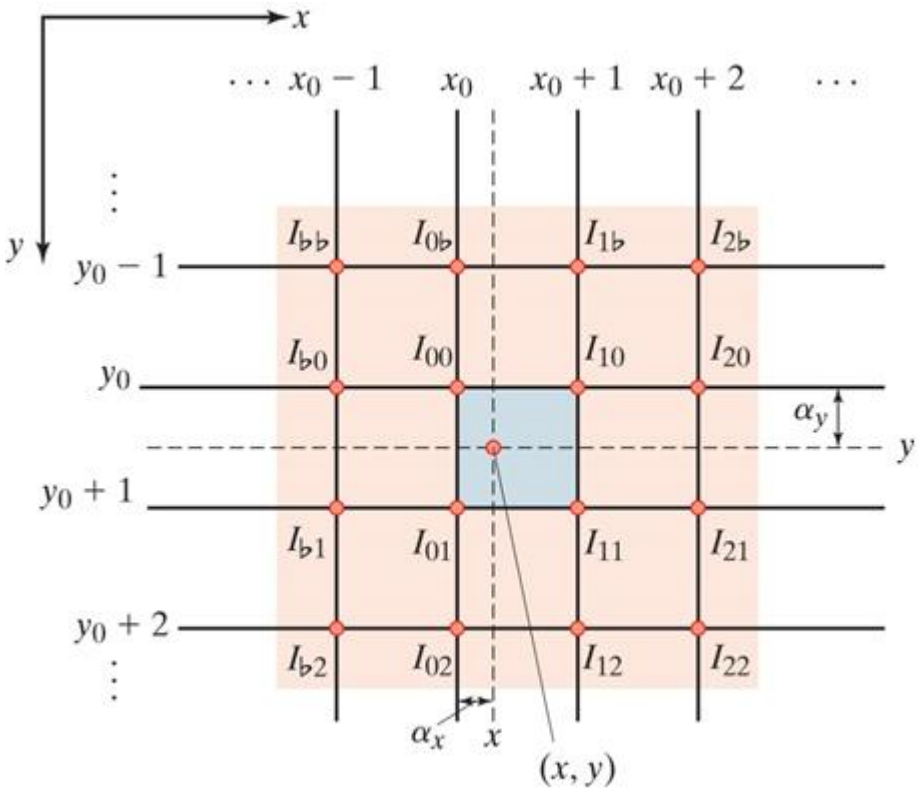
$$\hat{I}(x,y) = \overline{\alpha}_x\overline{\alpha}_y I_{00} + \alpha_x\overline{\alpha}_y I_{10} + \overline{\alpha}_x\alpha_y I_{01} + \alpha_x\alpha_y I_{11}$$



Bilinear interpolation

Given the following image, use bilinear interpolation to compute the value at (1.2, 0.5). Assume that the Upper left coordinate is (0,0).

$$I = \begin{bmatrix} 232 & 177 & 82 & 7 \\ 241 & 18 & 152 & 140 \\ 156 & 221 & 67 & 3 \end{bmatrix}$$



Solution:

The interpolated point is surrounded by the points

$$\begin{matrix} 177 & 82 \\ 18 & 152 \end{matrix}$$

I_{00} =177, I_{10} =82, I_{01} =18, I_{11} =152, α_x=0.2, α_y=0.5, $\overline{\alpha_x} = 0.8$, $\overline{\alpha_y} = 0.5$.

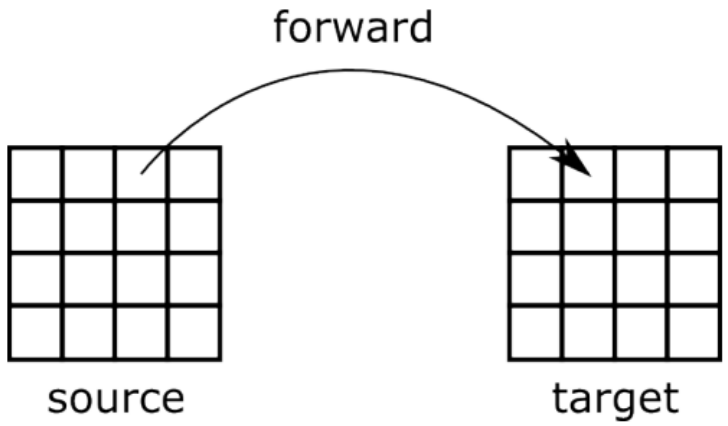
$$\hat{I}(x,y) = \overline{\alpha}_x\overline{\alpha}_y I_{00} + \alpha_x\overline{\alpha}_y I_{10} + \overline{\alpha}_x\alpha_y I_{01} + \alpha_x\alpha_y I_{11}$$ = 70.8 + 8.2 + 7.2 + 15.2= 101.4

Bicubic interpolation

- More accurate than bilinear.
- Uses derivatives to obtain a smoother estimate
- Slightly improves results over bilinear interpolation but computationally expensive

Forward and Inverse Warping

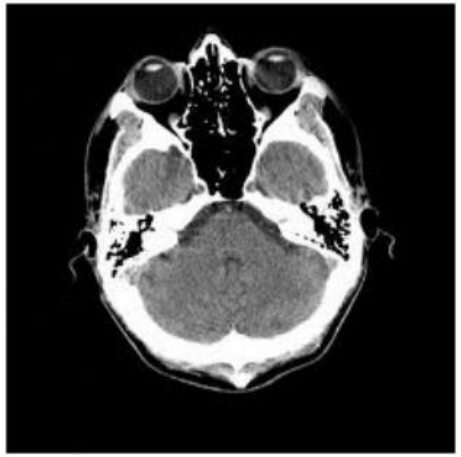
- The straightforward approach to applying transformations is to apply the transformation matrix to the input coordinates

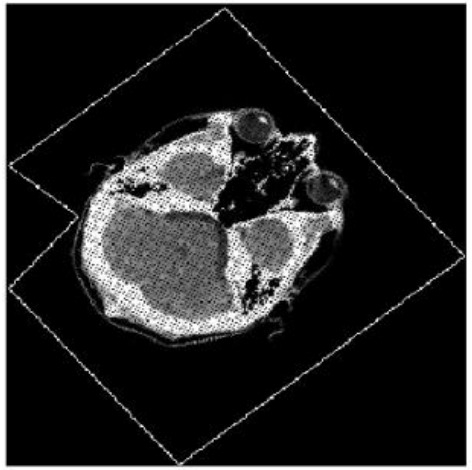


- This approach often leads to artifacts and is not preferred

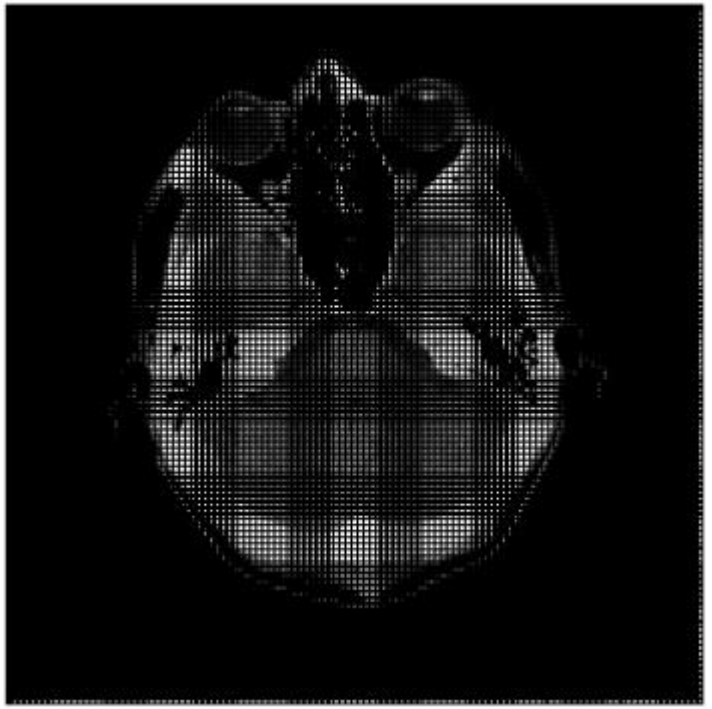
Forward and Inverse Warping

- Forward warping can generate holes in the output image.





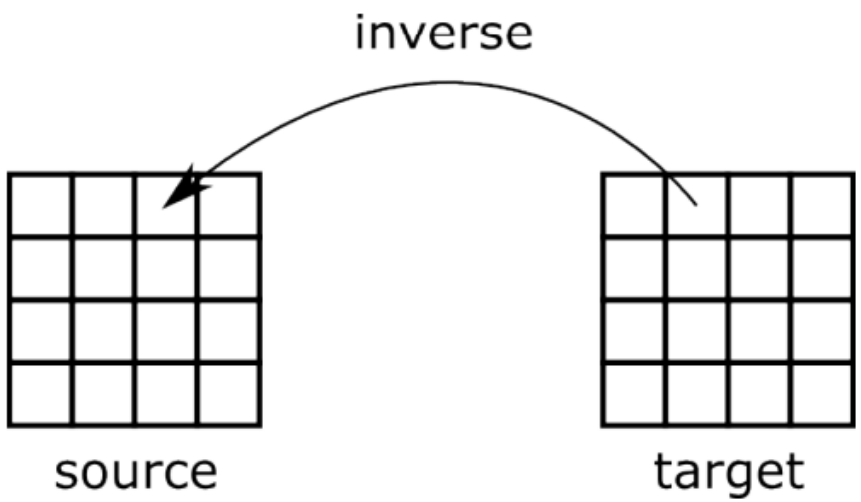
Rotation

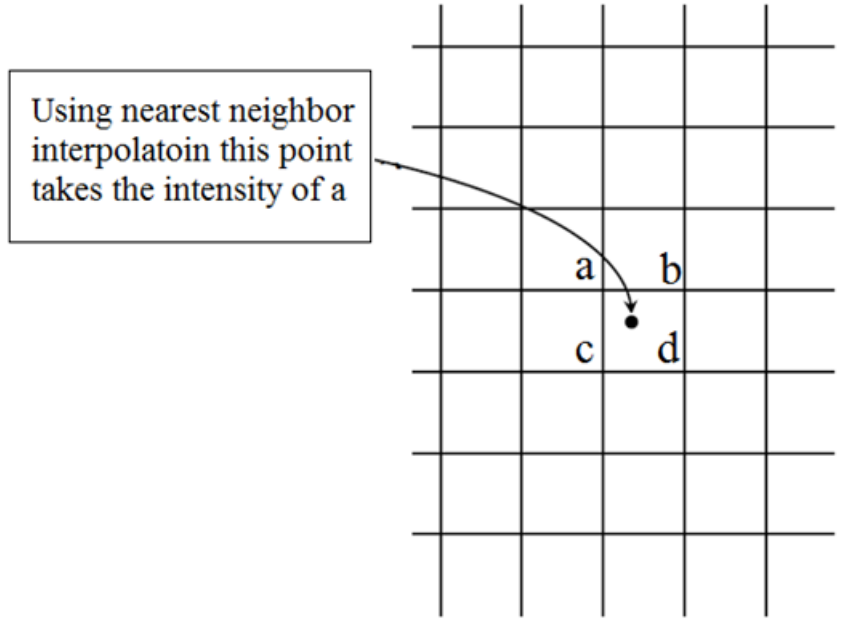


Scaling

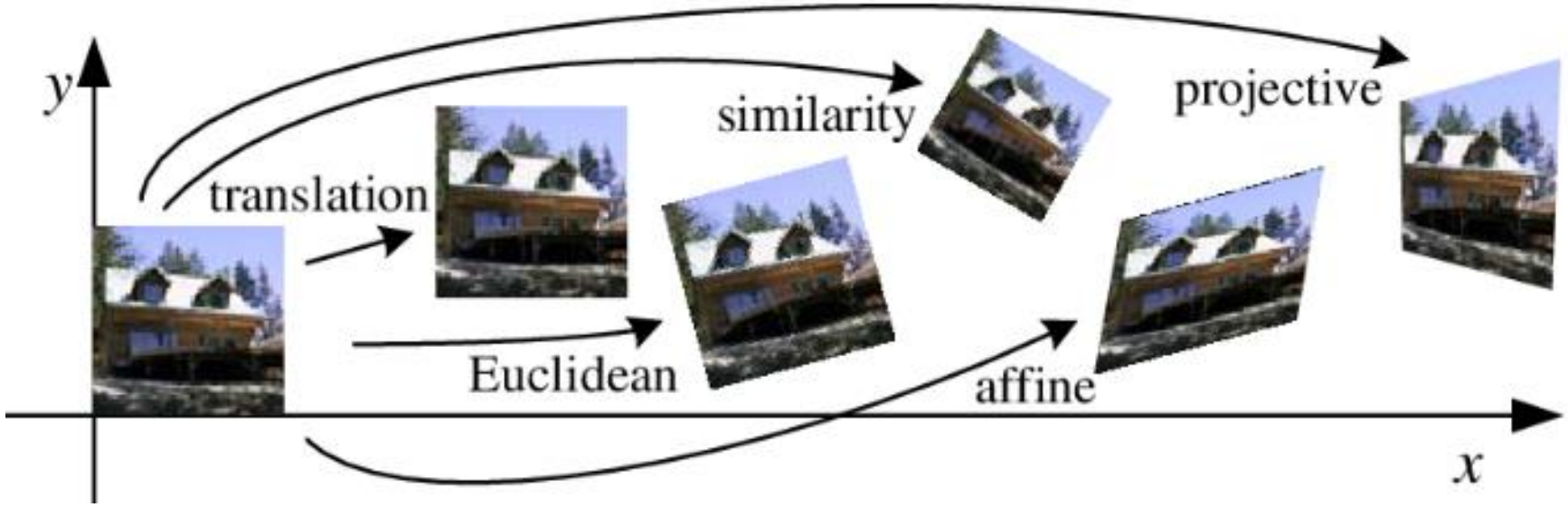
Inverse Warping

- Start by looping over the target image pixels. Compute the inverse transform (i.e., invert the transformation matrix). If you inverse relationship gives a non integer output, perform interpolation.





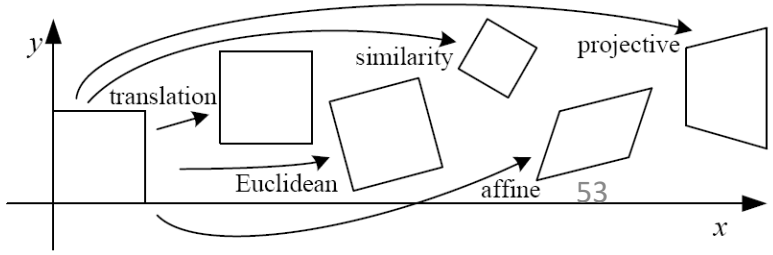
2D transformations



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?

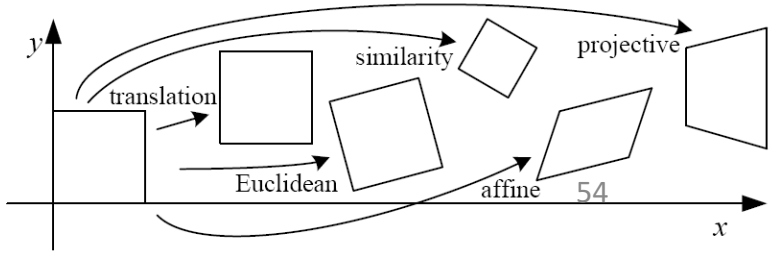


Classification of 2D transformations

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?

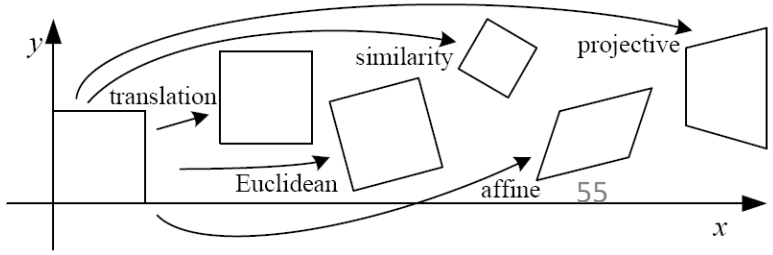


Classification of 2D transformations

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?

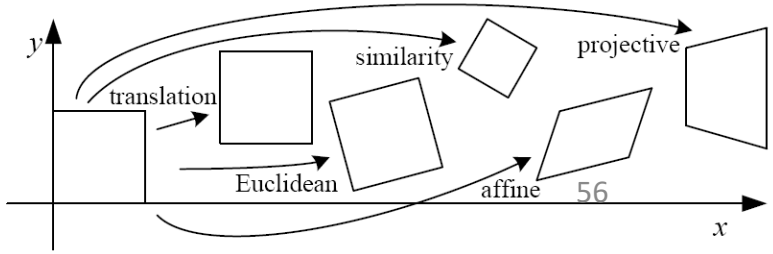


Classification of 2D transformations

what will happen to the image if this increases?

Euclidean (rigid): rotation + translation

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

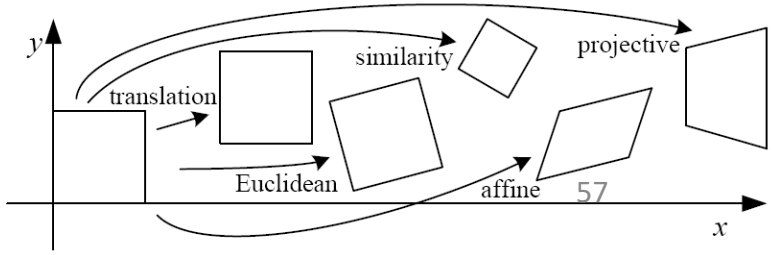


Classification of 2D transformations

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?

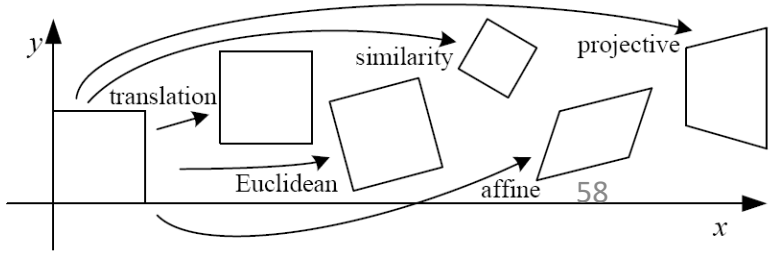


Classification of 2D transformations

Similarity:
uniform scaling + rotation
+ translation

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

How many degrees of freedom?

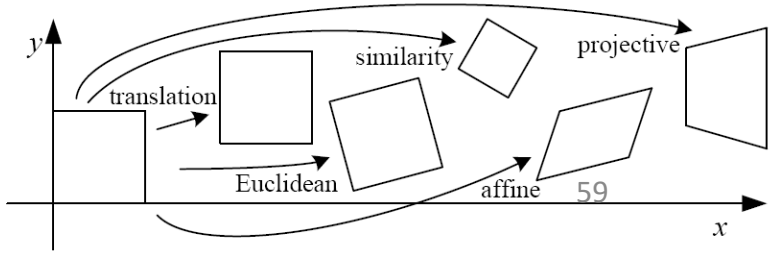


Classification of 2D transformations

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?



Affine transformations

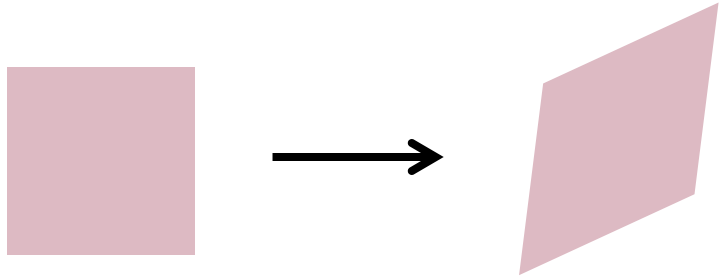
Affine transformations are combinations of

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

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

Properties of affine transformations:

- 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

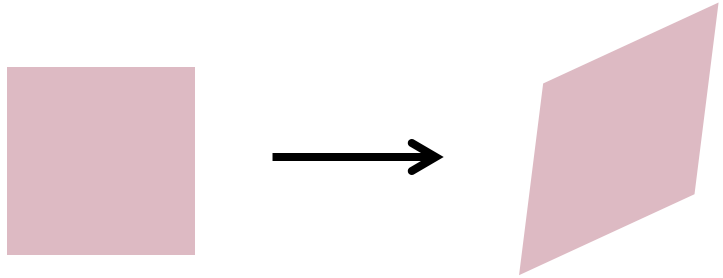
Affine transformations are combinations of

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

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

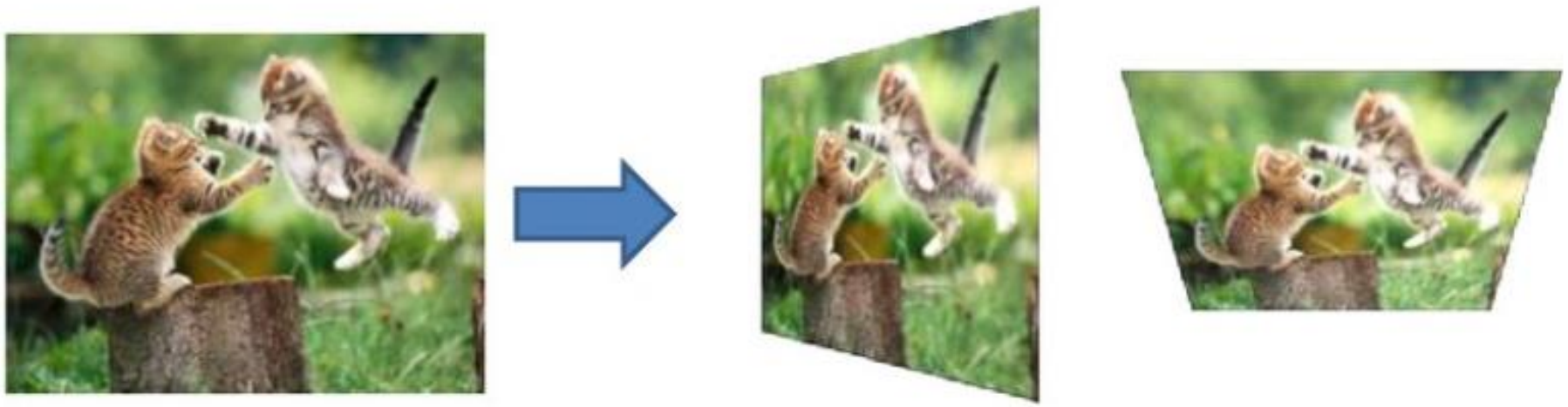
Properties of affine transformations:

- 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? Nope! But what does that mean?

Projective Transformations



Projective Transformations

Projective transformations are combinations of

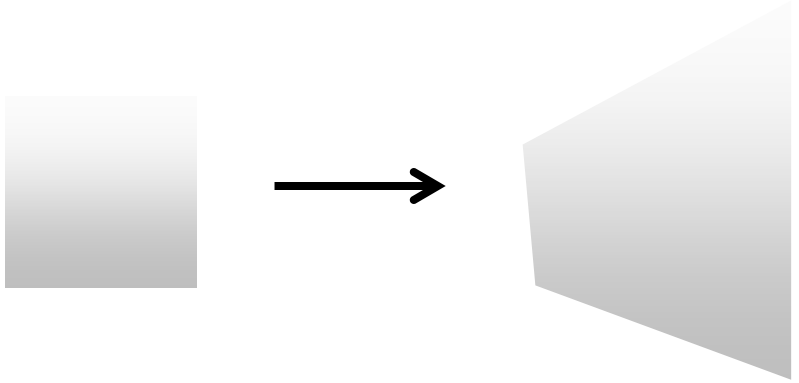
- affine transformations; and
- projective warps

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

8 DOF and 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



Moving back from the homogenous coordinates

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

$$(x', y', w') \longrightarrow \left(\frac{x'}{w'}, \frac{y'}{w'}\right)$$

Projective Transformations

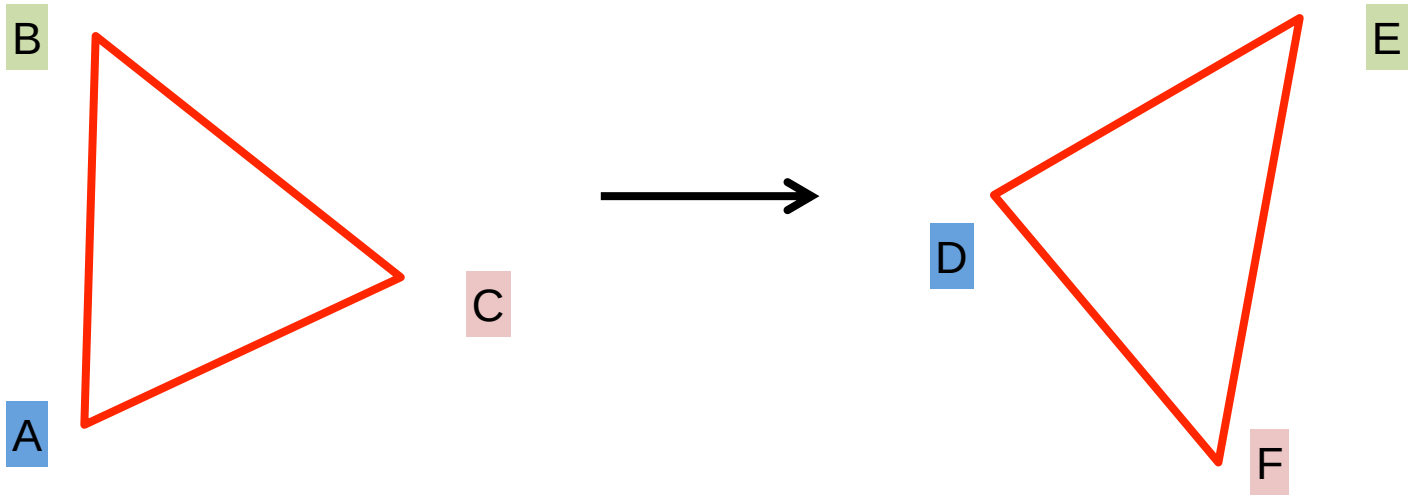
Image Stitching: Two images of a scene taken by a camera. How can we combine these?
We need to estimate the parameters of the projection matrix.

Projective Transformations

by overlapping them:

Determining unknown transformations

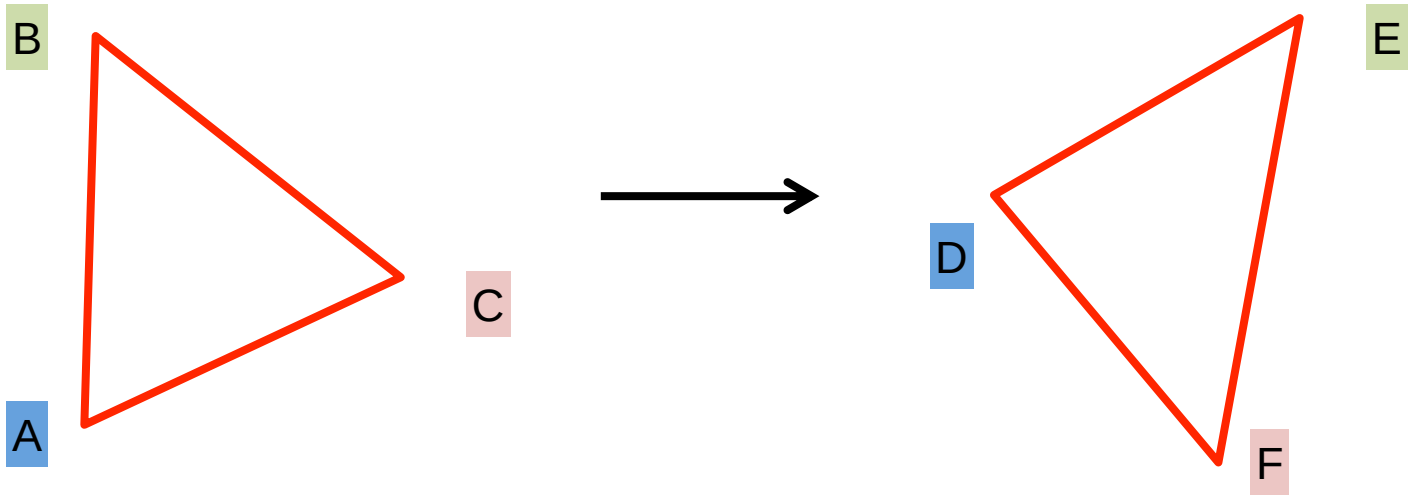
Suppose we have two triangles: ABC and DEF.

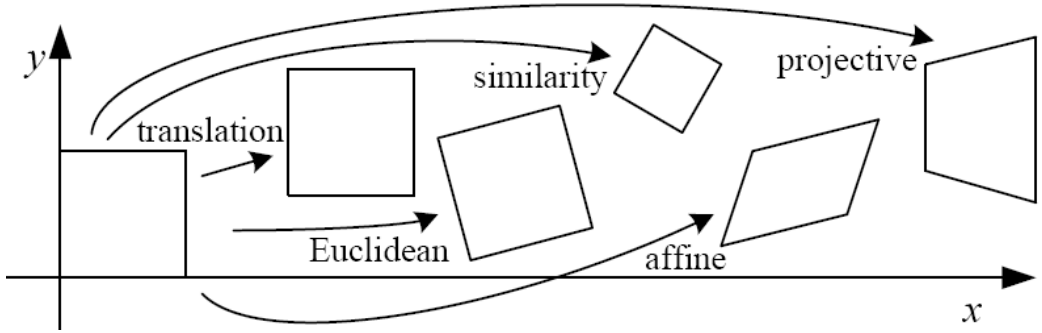


Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

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

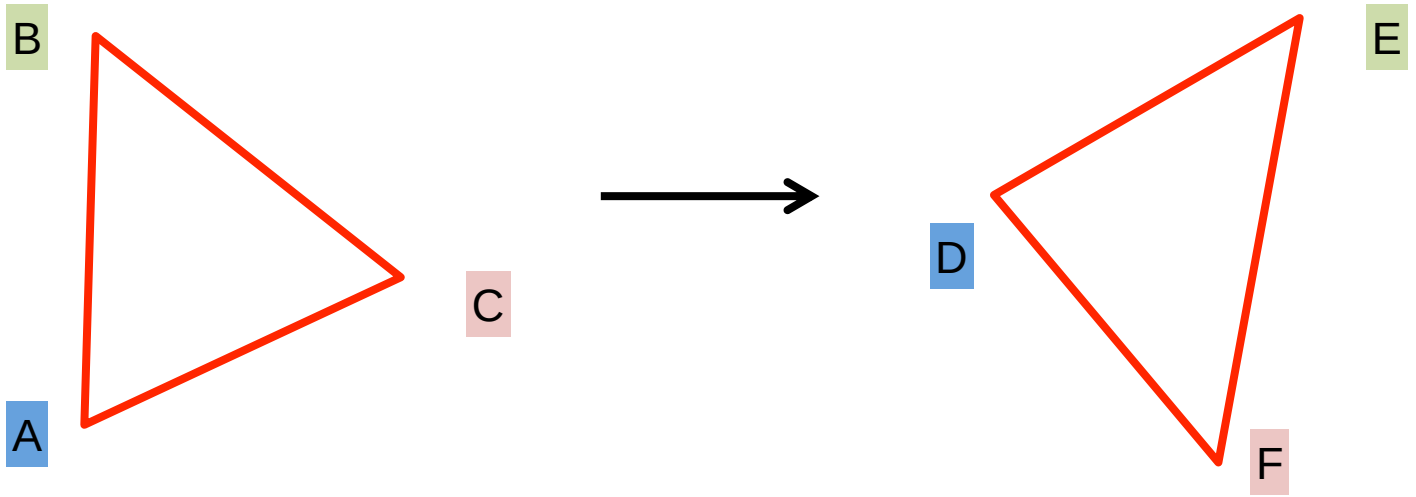




Determining unknown transformations

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

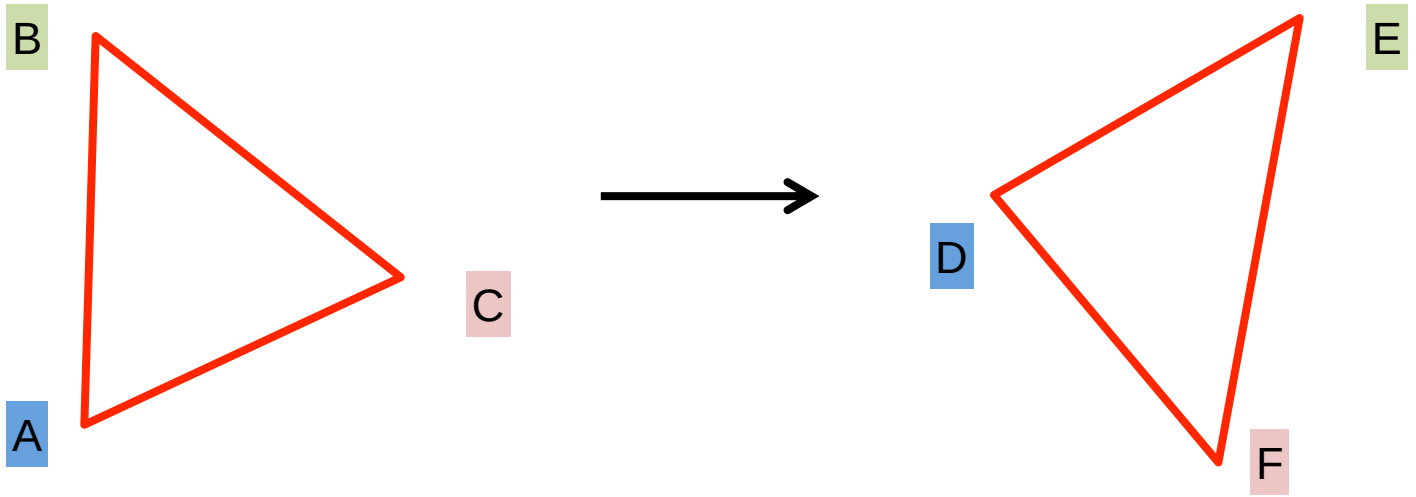
$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

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?



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

unknowns

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

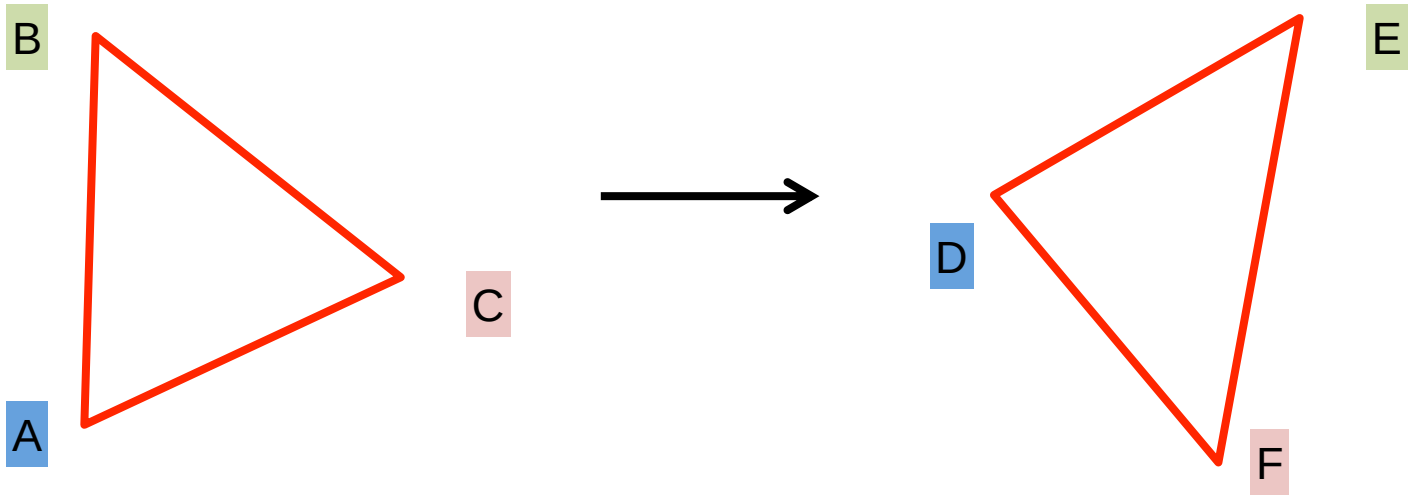
point correspondences

- 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?



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

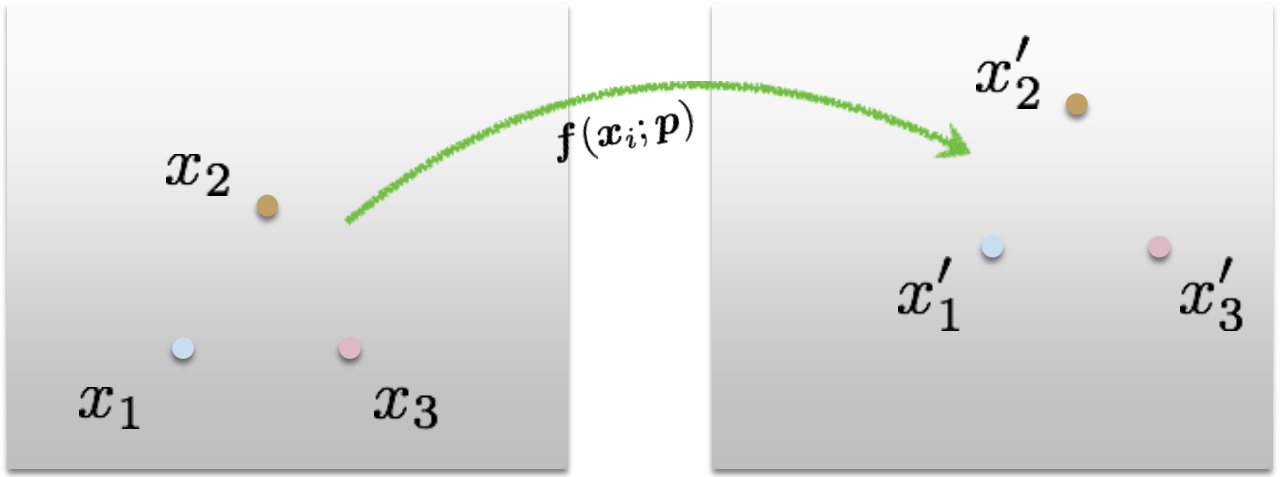
M

unknowns

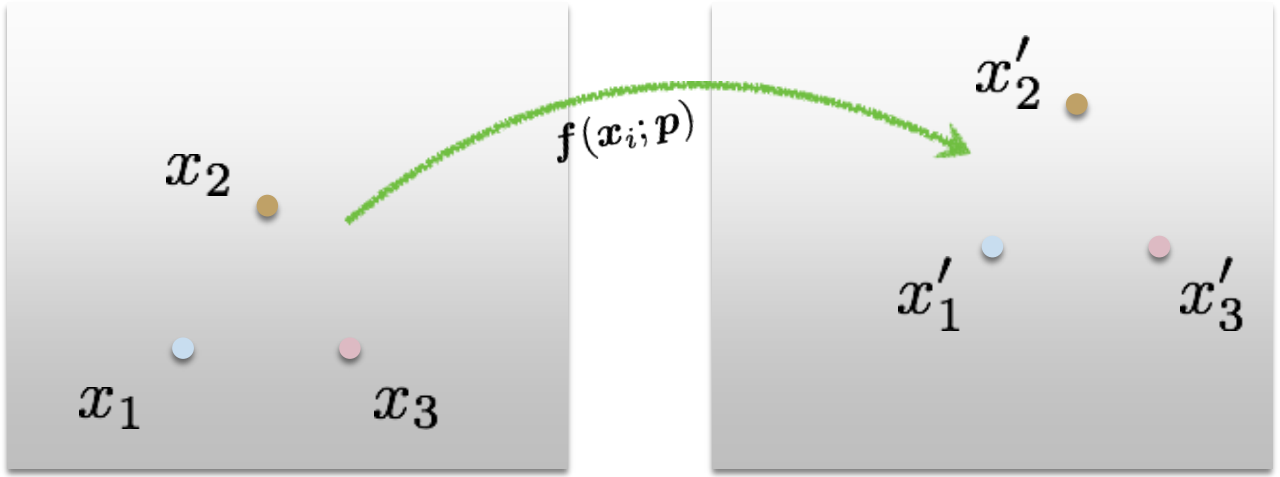
$$\boldsymbol{x}' = \mathbf{M}\boldsymbol{x}$$

point correspondences

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



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



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

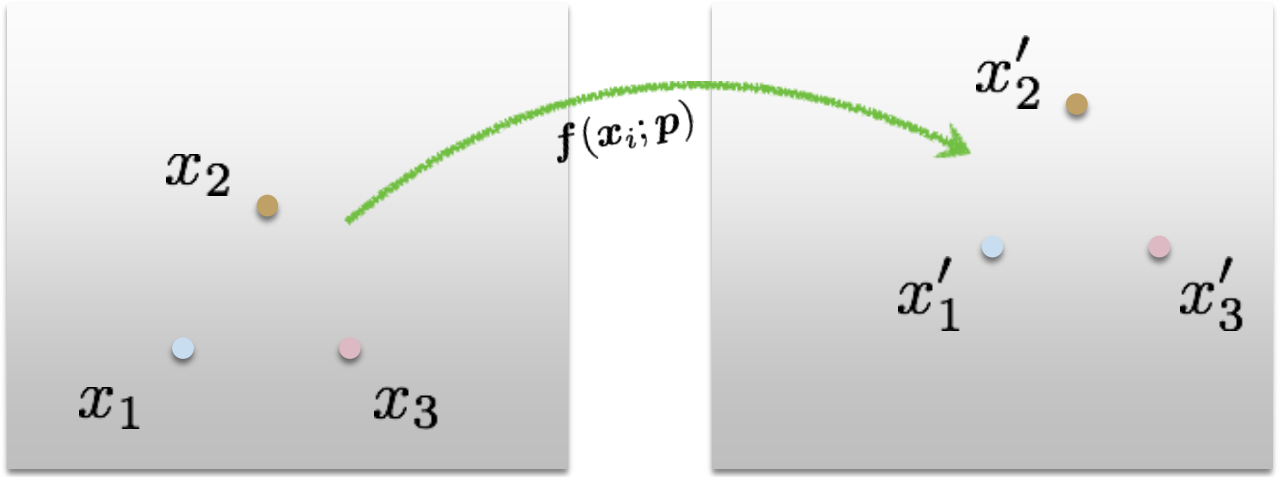
Euclidean (L2) norm

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

squared!

predicted location

measured location



Find parameters that minimize squared error

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

Determining unknown transformations

Convert the system to a linear least-squares problem:

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:

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

Notation in system form:

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

Least Squares Method

$$A\,x = b$$

$$x = (A^\top A)^{-1} A^\top b$$

Pseudoinverse or the Moore–Penrose inverse

The solution that minimizes mean squared error

Solving for $\boldsymbol{x} = \begin{bmatrix} m \\ b \end{bmatrix}$

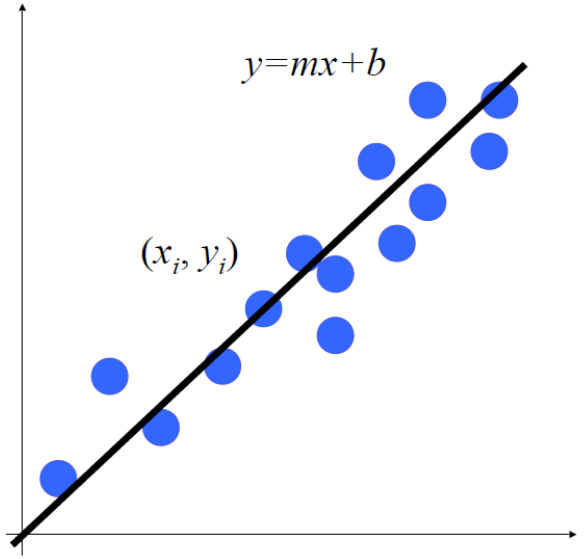


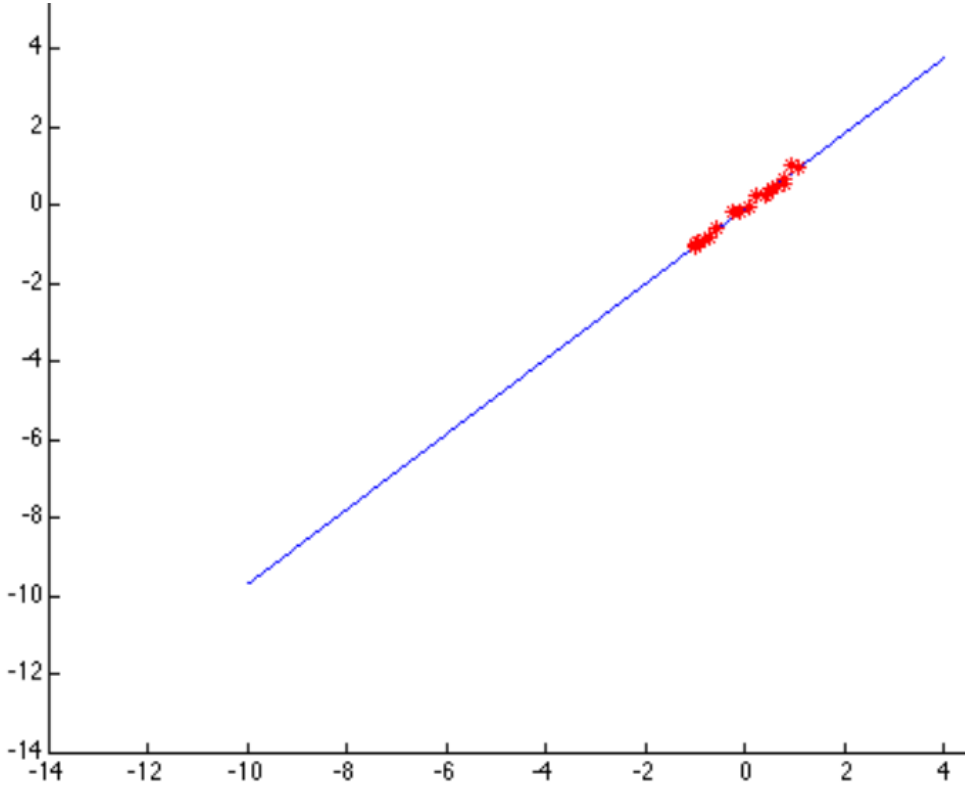
Image Holography Using Projective Transforms

Example: two pictures taken by rotating the camera:

With an homography you can map both images into a single camera:

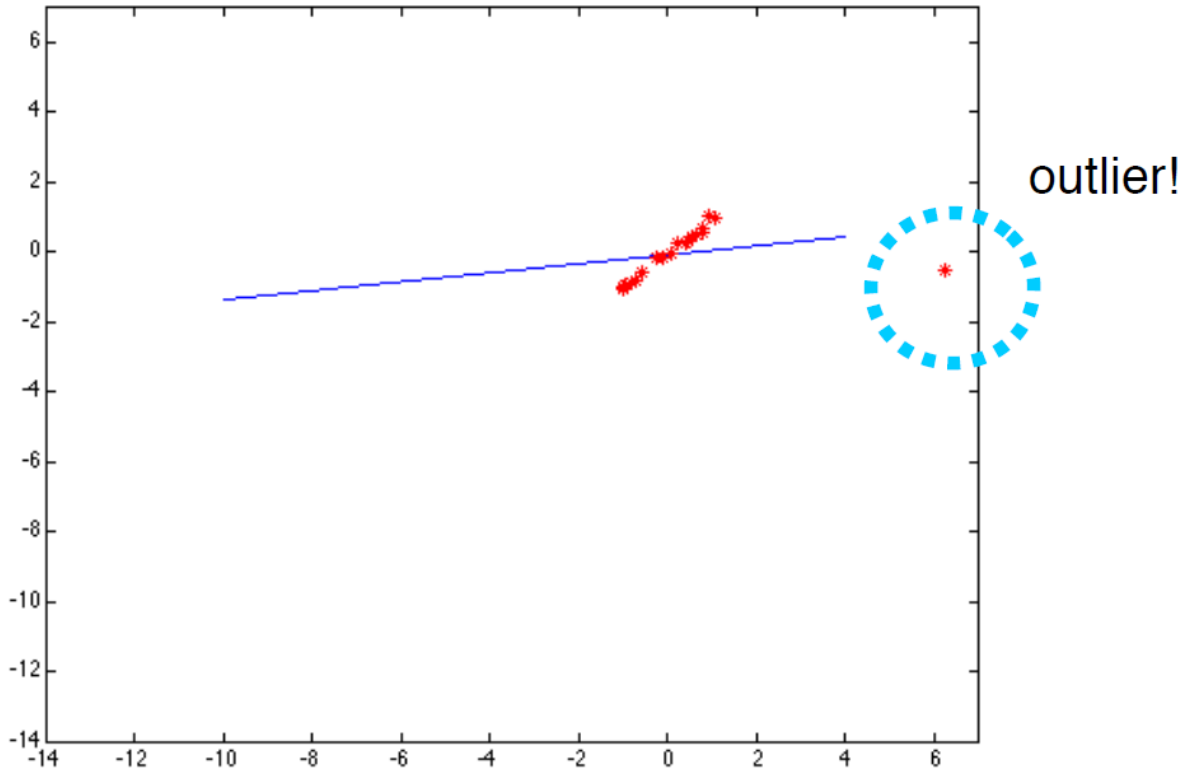
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Problem with Least Squares

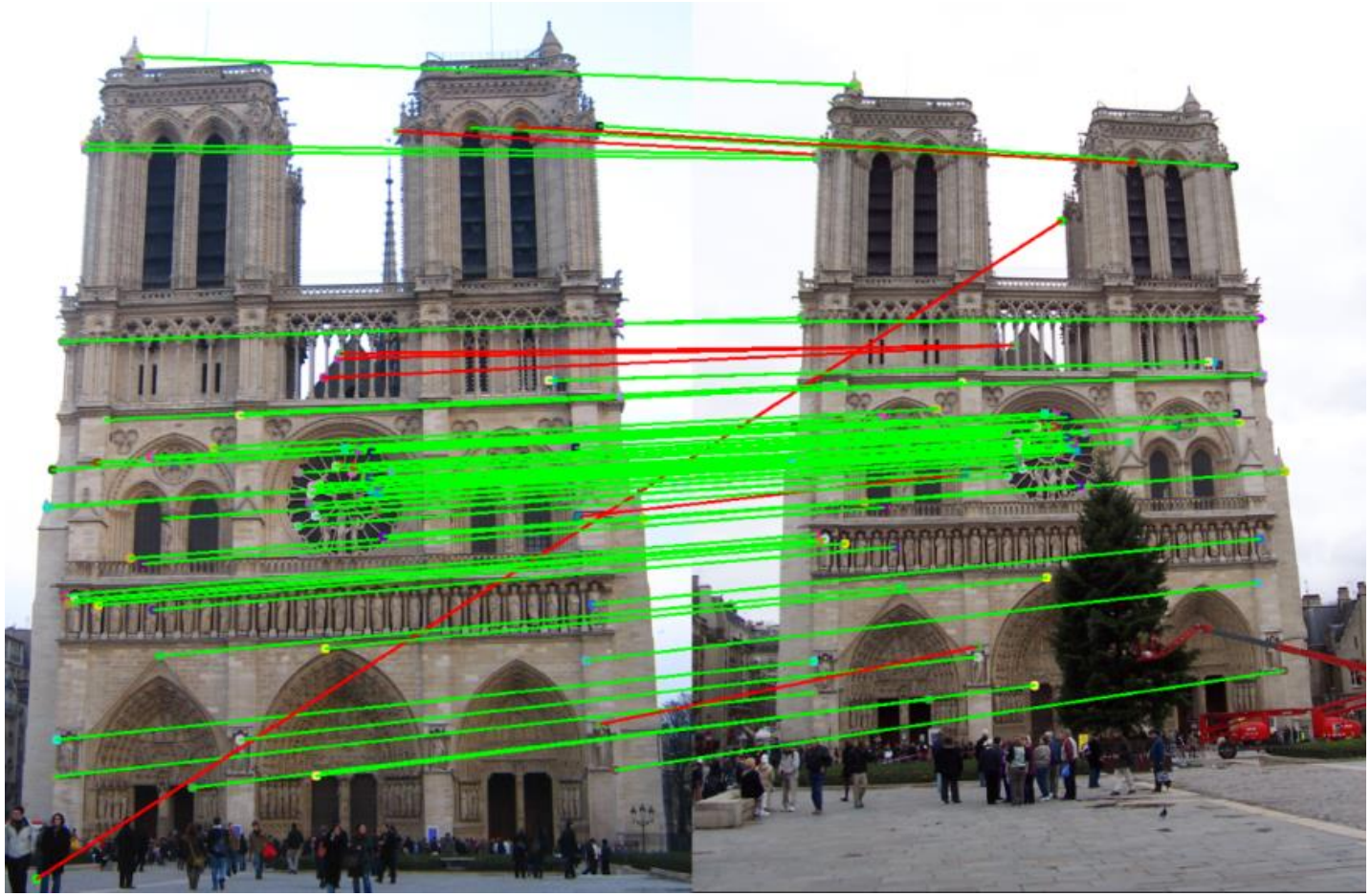


Problem with Least Squares

- Sensitivity to outliers



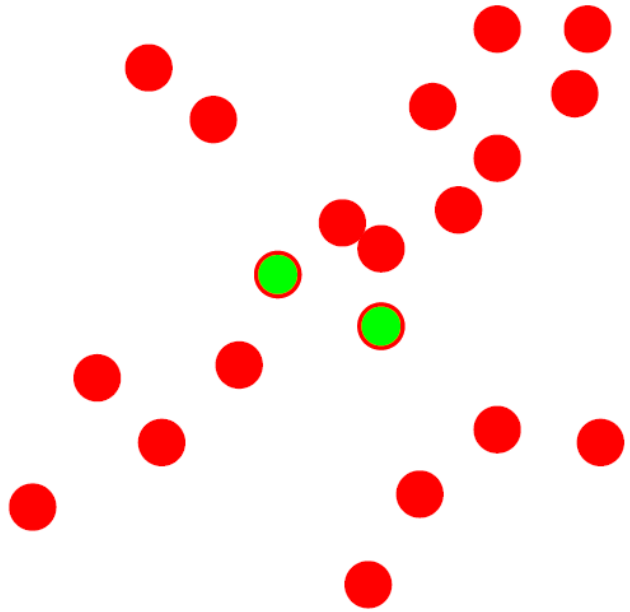
Not all matches can be correct



RANSAC (RANdom SAmple Consensus)

- Main idea:
- Select a random subset of points and fit a line through these points.
- Use the computed line to compute the number of point for which this line is a good fit (inliers).
- Repeat the process until you get the line with largest number of inliers.

RANSAC

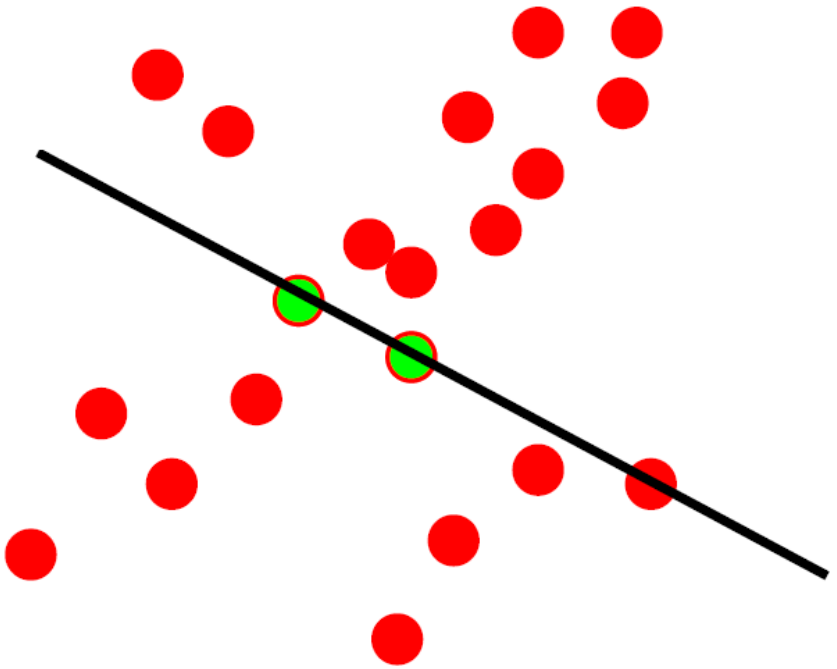


Algorithm:

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

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

RANSAC

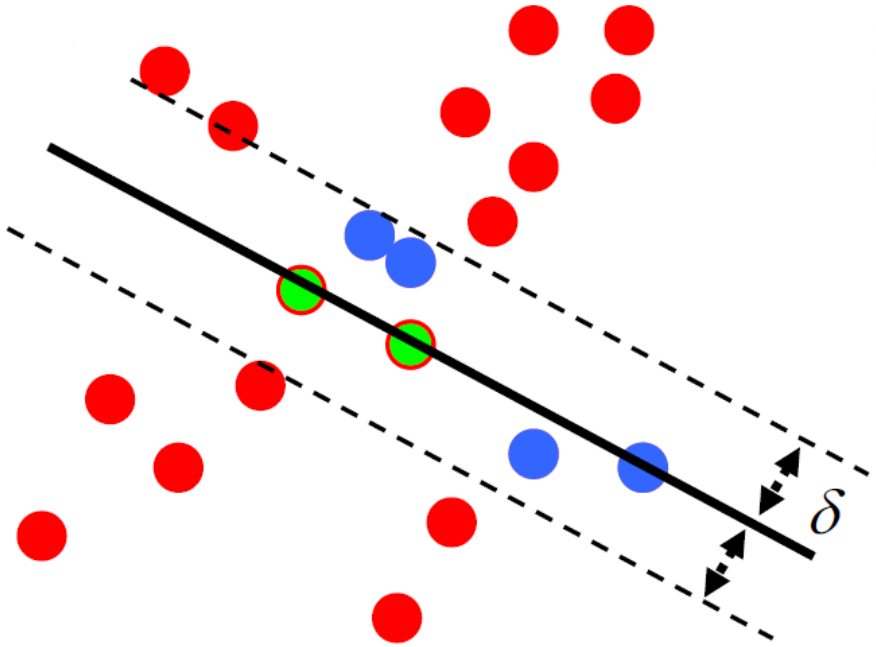


Algorithm:

1. **Sample** (randomly) the number of correspondences required to find the model parameters (#=2).
2. **Solve** for model parameters using the samples.
3. **Score** by the fraction of inliers within a preset δ threshold of the mode

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

RANSAC



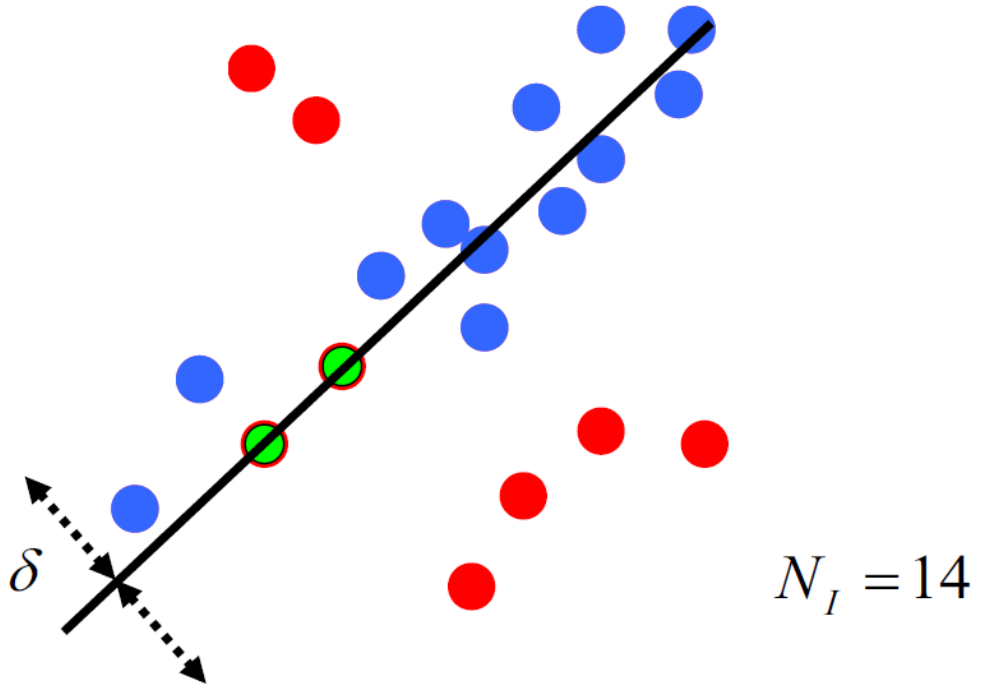
$N_I = 6$

Algorithm:

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

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

RANSAC



Algorithm:

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

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

RANSAC

- RANSAC offers a method for excluding outliners
- If we have a large number of inliers we can average solutions:
  - Once you find the best model (minimum outliers, maximum inliers): Compute the transformation parameters for each set of correct points.
  - Correct solution is the average of all solutions obtained from inliers

RANSAC

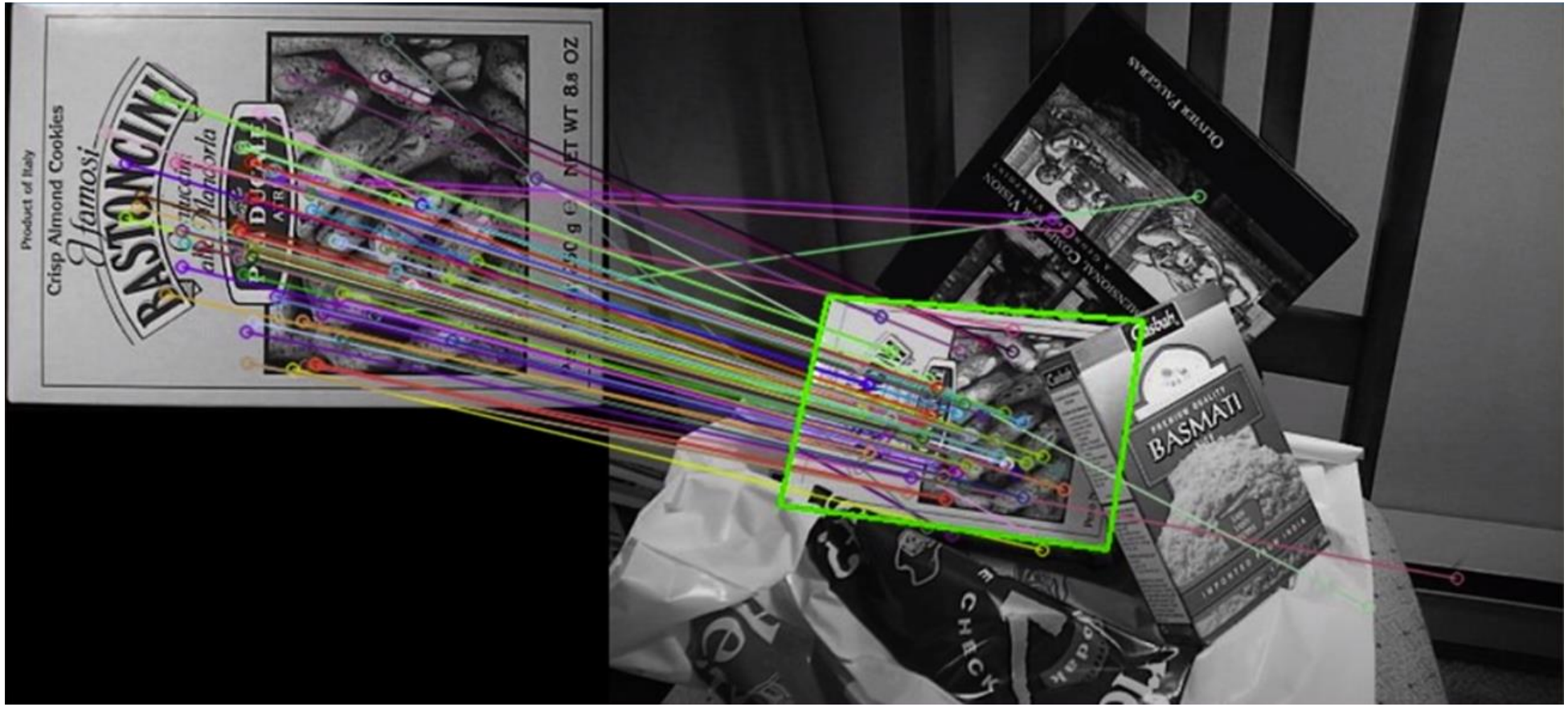
Pros

- Robust to outliers

Cons

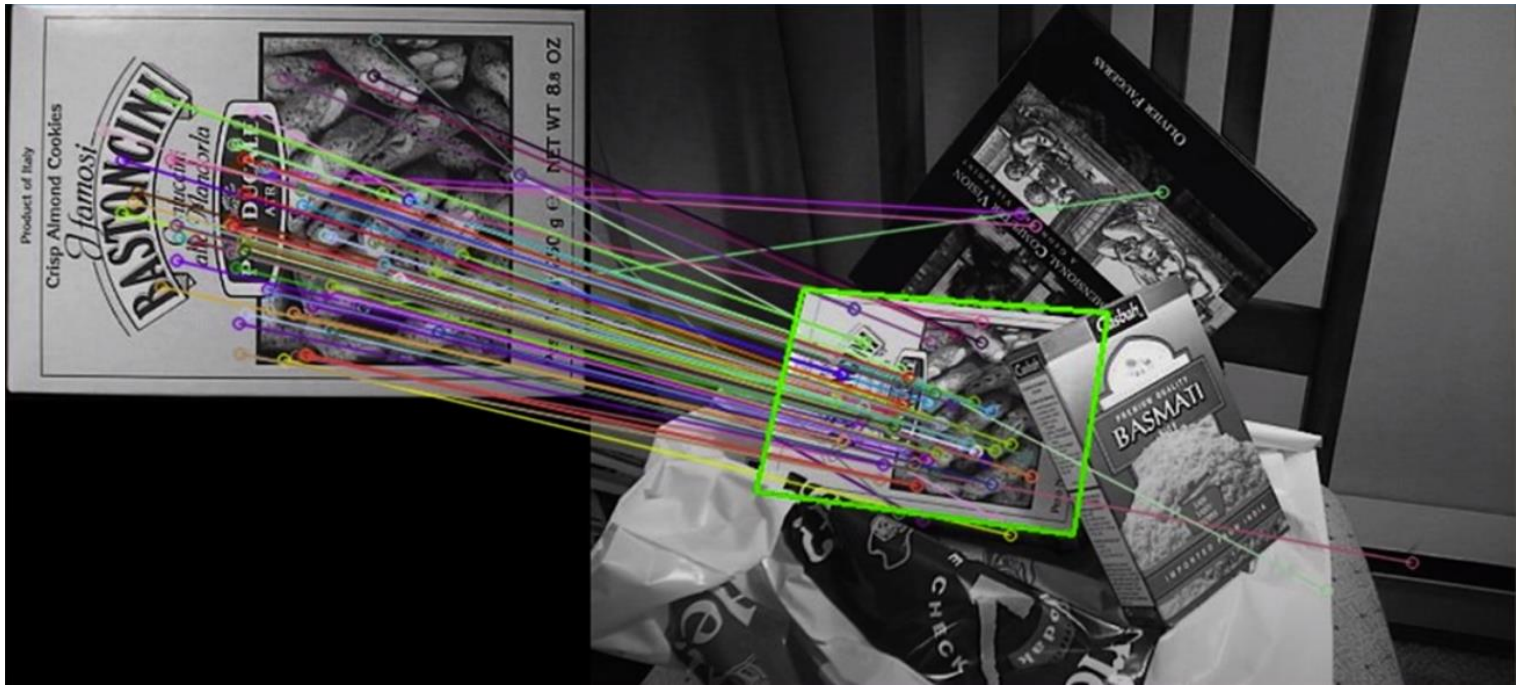
- Computational time grows quickly with the fraction of outliers and number of parameters.
- In practice, works only if outliers are not more that 30% of all correspondences.
- Not good for getting multiple fits (e.g., several similar, but differently located, objects in matched images).

RANSAC example

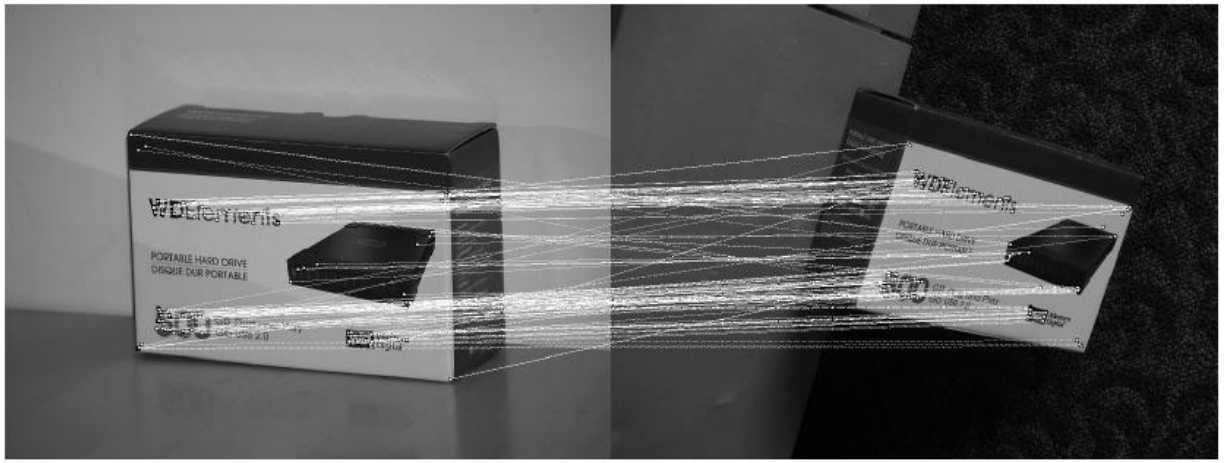


RANSAC example

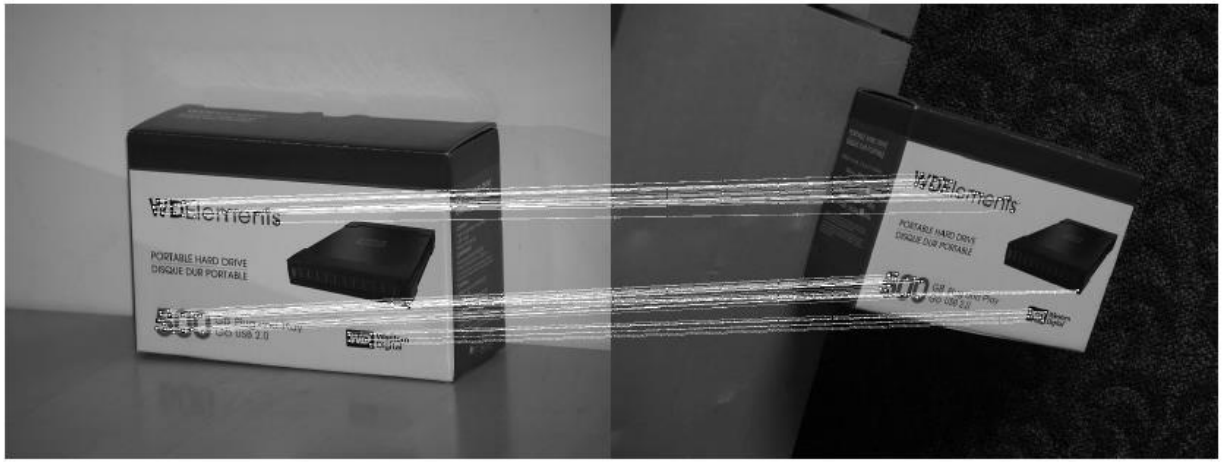
- Using RANSAC we compute the solution for transform parameters using only the inlier points A x = b, A and b are composed of inlier points.



RANSAC example



before RANSAC



after RANSAC

Image Registration

- Image registration (image alignment) is the process of determining the spatial transform that maps points from one image to homologous points on an object in the second image.
- Registration algorithms can be
  - Point based: Matching a set of points between two images
  - Surface based: Matching surfaces
  - Intensity based: rely only on the pixel intensity values in the images

Image Registration

- A subfield in image processing where the goal is to align two images.
- Image transformations are applied to an image to align it with the second image.

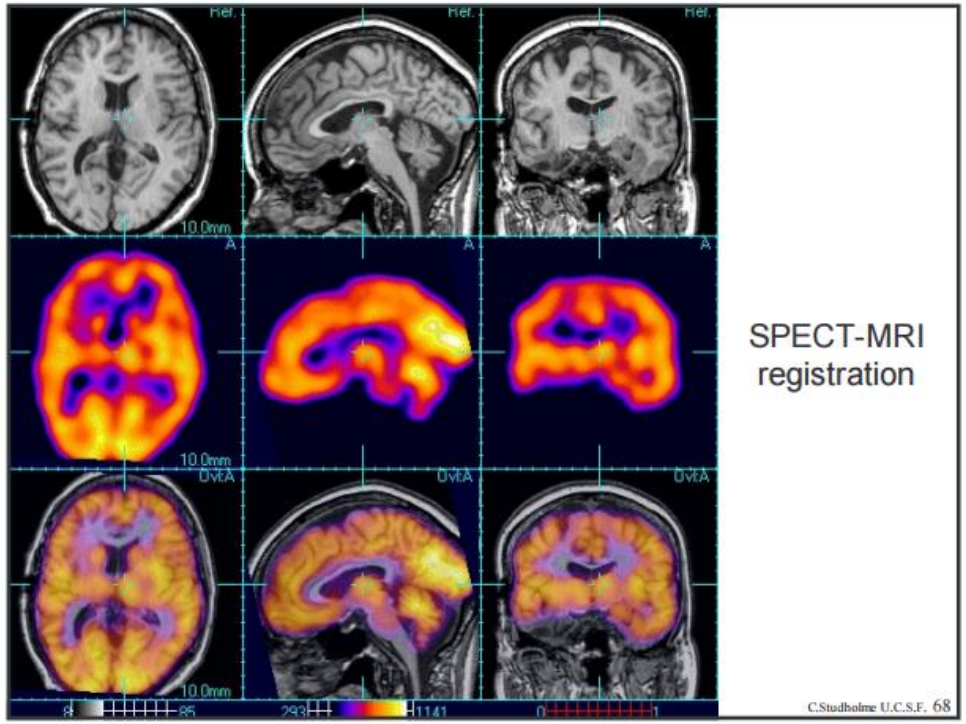


Image Registration

- The general framework for intensity-based registration follows the following steps.
- Assume that we are given a source image (S) and our goal is to compute the transform that best matches this image with the target image (T).

1. Start with a random transform H
2. Apply the transform to image S
3. Compute the distance between image T'=HS and T using a *similarity measure*
4. Using a suitable *optimization algorithm* update transform parameters H and go back to step 2.
5. Stop when the similarity between image HS and the target image is below a desired threshold or after a fixed number of iterations.

Image Registration

- Examples of *similarity measures*
- Sum of Squared Difference (SSD)

$$\mathrm{SSD} = \frac{1}{N}\sum_i [A(i) - B^T(i)]^2$$

- Correlation Coefficient

$$\mathrm{CC} = \frac{1}{N} \frac{\sum_i (A(i) - \overline{A}) \times (B^T(i) - \overline{B^T})}{\{ \sum_i (A(i) - \overline{A})^2 \times \sum_i (B^T(i) - \overline{B^T})^2 \}^{1/2}}$$

- While SSD targets is minimizing the difference between the two images, a linear relationship between the two images is sufficient for CC

Image Registration

- Previous similarity measures will not work in the case where the two images come from different sources (e.g., MRI vs CT)
- Mutual information, a concept from information theory, was found to be a better method for use in image registration

Entropy

- The entropy of *X* is given by the negative of the weighted average of the logarithm of the probabilities

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

- Entropy is often measured in the unit of bits and the used log. is with base 2
- The convention is to define 0 log 0 = 0.

Entropy

- Entropy can be viewed as the amount of new information that can be obtained (surprise).

Example: Find the entropy of

(a) Flipping a fair coin.

$$H(X) = -(\tfrac{1}{2} \log \tfrac{1}{2} + \tfrac{1}{2} \log \tfrac{1}{2}) = 1 \text{ bit.}$$

(b) Flipping a coin that has heads on both sides

$$H(X) = - (1 \log 1 + 0) = 0 \text{ bit.}$$

(c) Flipping a fair 6-sided die

$$H(X) = -6 (-1/6 \log 1/6) \approx 2.585 \text{ bits.}$$

MI and joint entorpy

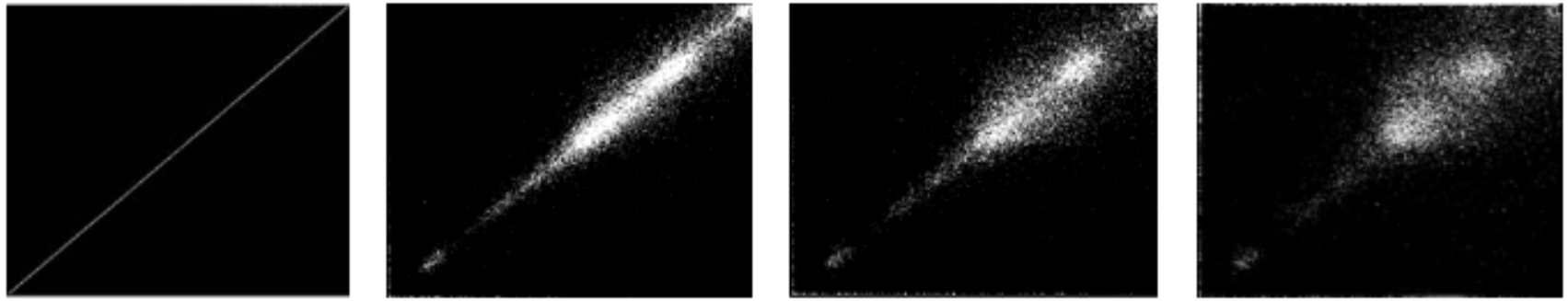
- Mutual information is a popular similarity measure. It's a concept derived from entropy

$$H = -\sum_{s} p(s) \log p(s)$$

- Joint entropy between two images is given by

$$H(A,B) = -\sum_{i,j} p_{AB}(i,j) \log p_{AB}(i,j)$$

Joint Entropy



(a) H=5.9 (b) H=10.5 (c) H=11.2 (d) H=12.1

Figure 8: Join histogram and joint entropy are shown between (a) image A and itself. (b), (c) and (d) image A and image A rotated by 1, 3 and 10 degrees respectively.

$$H(A,B) = -\sum_{i,j} p_{AB}(i,j) \log p_{AB}(i,j)$$

MI based Image Registration

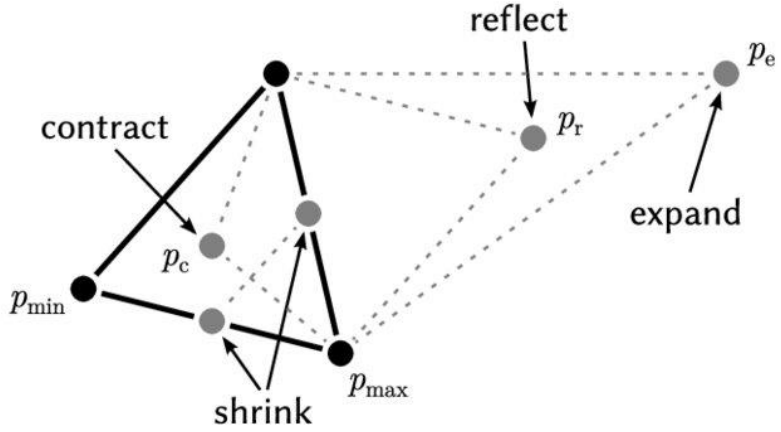
- Maximizing Mutual Information

$$MI(A,B) = H(A) + H(B) - H(A,B)$$

- Computing the derivative is difficult. Often used with derivative free optimization algorithms

Derivative Free Optimization algorithms

- Nelder-Mead's method:
  Evaluate the cost function at a number of points equal to the degrees of freedom.
  Move the worst point according to a given set of rules.



- Powell's direction set
- Particle Swam Optimization
- Evolutionary (genetic) algorithms