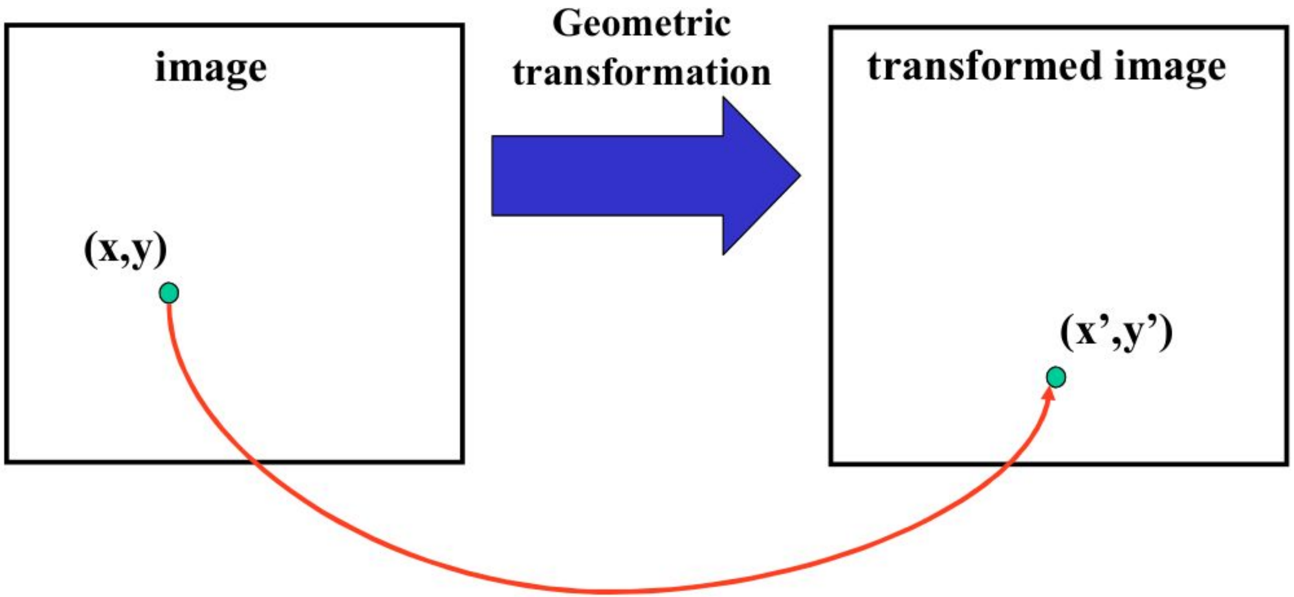
MLRF Lecture 03

J. Chazalon, LRDE/EPITA, 2020

Projective transformations

Lecture 03 part 04

A linear transformation of pixel coordinates



$$x' = f(x, y, \{parameters\})$$
$$y' = g(x, y, \{parameters\})$$

Illustration: Robert Collins

Image Mappings Overview

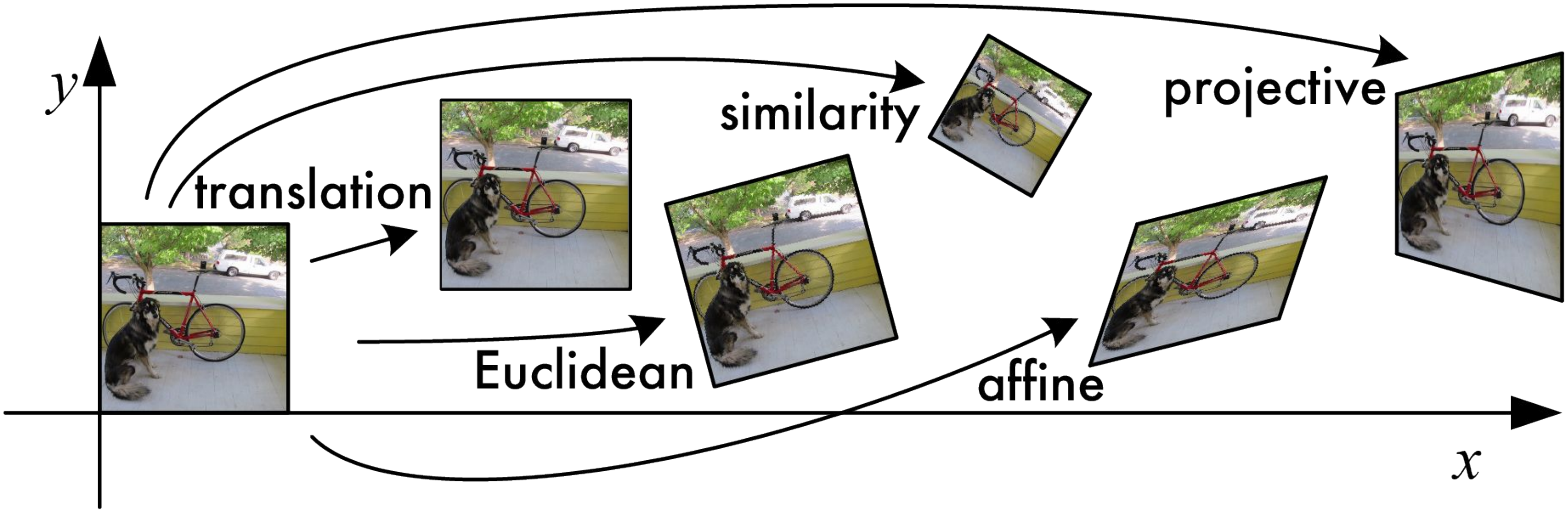


Image adapted from R. Szeliski by JP. Reddie

Math. foundations & assumptions

Homography H
(planar projective
transformation)

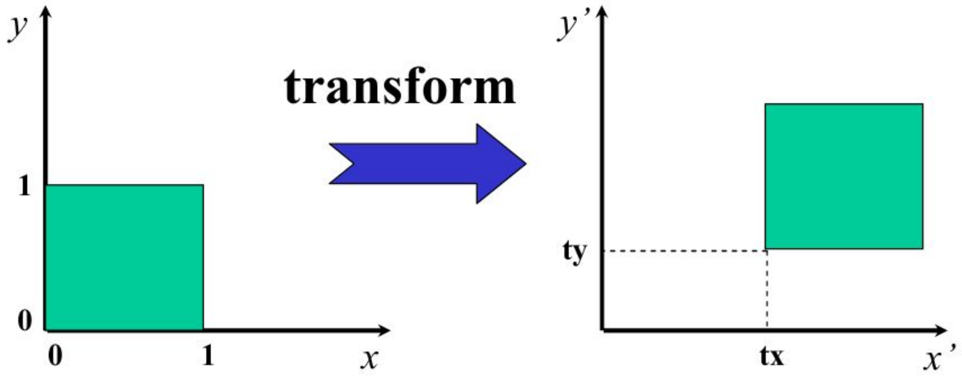
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

For **planar surfaces**, 3D to 2D perspective projection reduces to a 2D to 2D transformation.

This is just a change of coordinate system.

This transformation is **INVERTIBLE**!

Translation



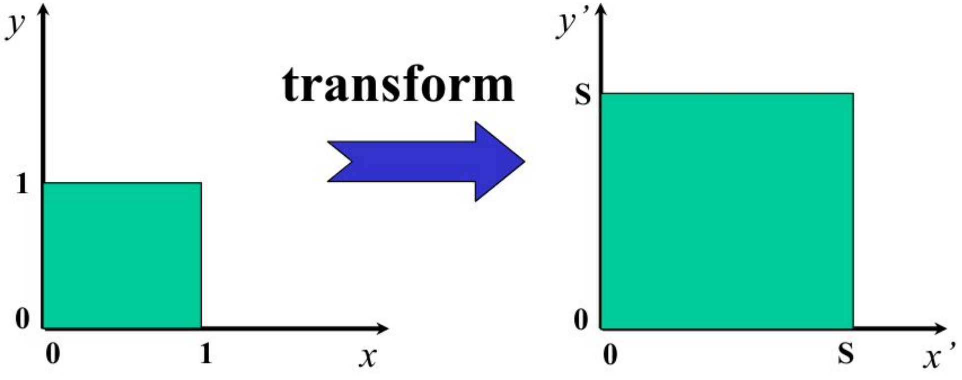
$$x' = x + t_x$$
$$y' = y + t_y$$

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

equations **matrix form**

Illustration: Robert Collins

Scale



$$x' = s\,x_i$$
$$y' = s\,y_i$$

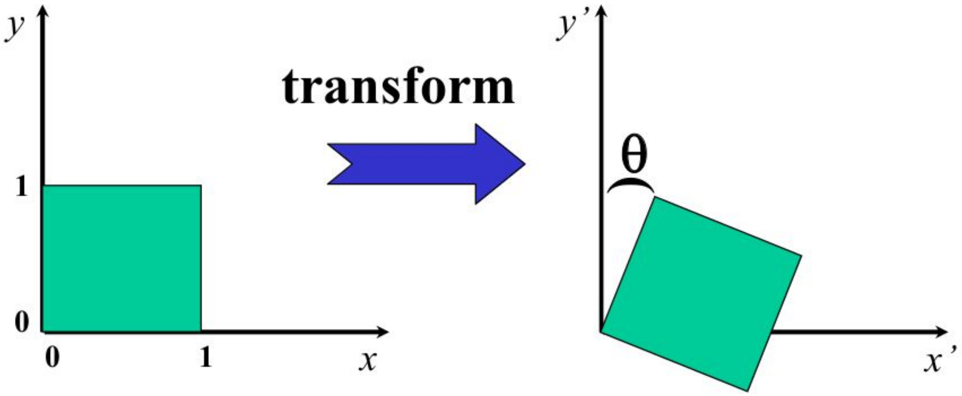
equations

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

matrix form

Illustration: Robert Collins

Rotation



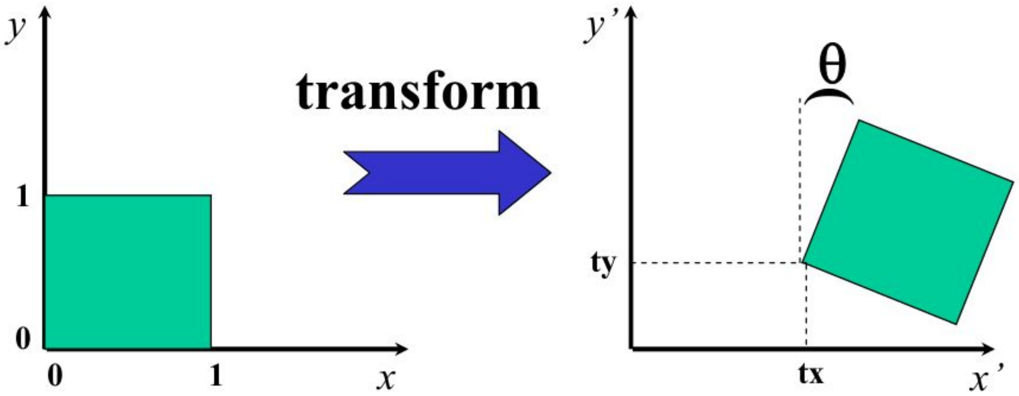
$$
\begin{aligned}
x' &= x_i \cos\theta - y_i \sin\theta \\
y' &= x_i \sin\theta + y_i \cos\theta
\end{aligned}
$$

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

equations **matrix form**

Illustration: Robert Collins

Euclidean (rigid)



$$
\begin{aligned}
x' &= x_i \cos\theta - y_i \sin\theta + t_x \\
y' &= x_i \sin\theta + y_i \cos\theta + t_y
\end{aligned}
$$

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

equations **matrix form**

Illustration: Robert Collins

Notation: Partitioned matrices

$$\left[\begin{array}{c} x' \\ y' \\ \hline 1 \end{array}\right] = \left[\begin{array}{cc|c} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ \hline 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} x \\ y \\ \hline 1 \end{array}\right]$$

$$\begin{bmatrix} \overset{\text{2x1}}{p'} \\ \overset{\text{1x1}}{1} \end{bmatrix} = \begin{bmatrix} \overset{\text{2x2}}{R} & \overset{\text{2x1}}{t} \\ \overset{\text{1x2}}{0} & \overset{\text{1x1}}{1} \end{bmatrix} \begin{bmatrix} \overset{\text{2x1}}{p} \\ \overset{\text{1x1}}{1} \end{bmatrix}$$

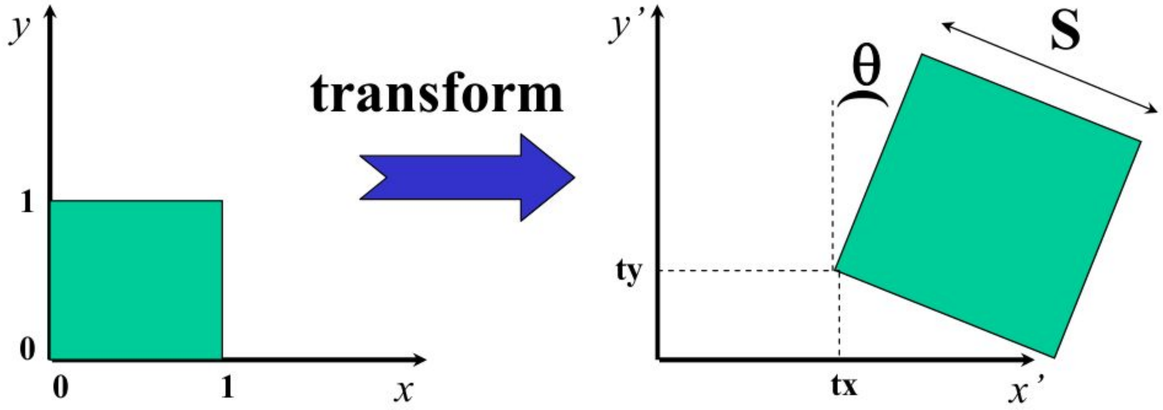
matrix form

$$p' = Rp + t$$

equation form

Illustration: Robert Collins

Similarity (scaled Euclidean)



$$p' = sRp + t$$

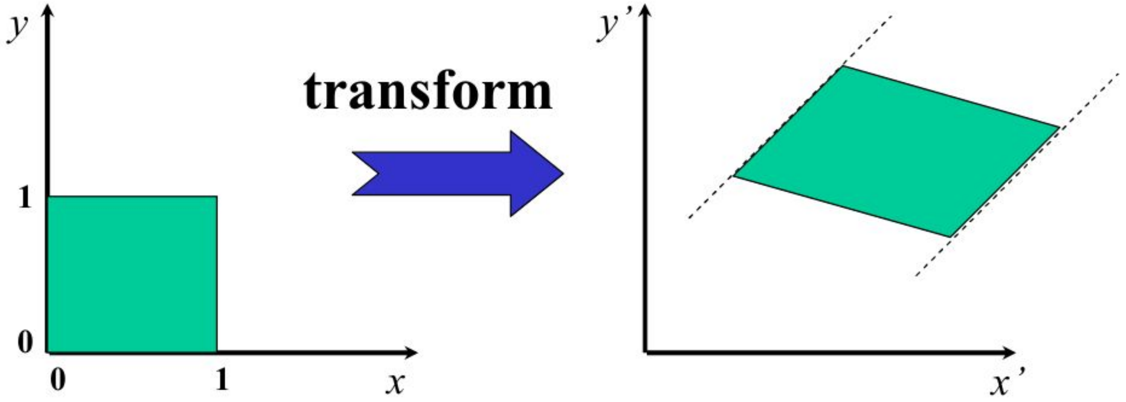
equations

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

matrix form

Illustration: Robert Collins

Affine



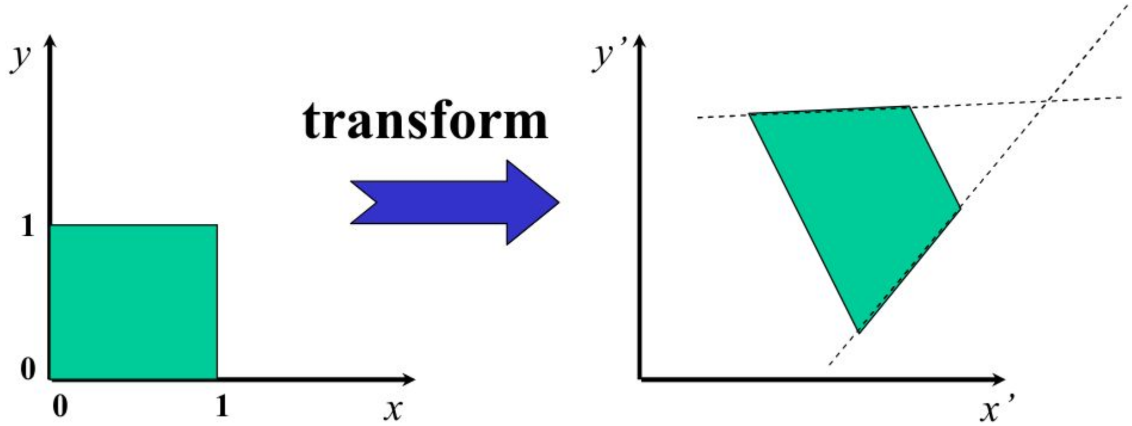
$$p' = Ap + b$$

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

equations **matrix form**

Illustration: Robert Collins

Projective



Note!

$$p' = \frac{Ap + b}{c^T p + 1}$$

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} \sim \begin{bmatrix} A & b \\ c^T & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

equations **matrix form**

Illustration: Robert Collins

More on projective transform

Each point in 2D is actually a vector in 3D

Equivalent up to scaling factor

$3*\mathbf{H} \sim \mathbf{H}$

Have to normalize to get back to 2D

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} \qquad \bar{\mathbf{x}} = \tilde{\mathbf{x}} \,/\, \tilde{w}$$

Why does this make sense?

Pinhole camera model:

- Every point in 3D projects onto our viewing plane through our aperture
- Points along a vector are indistinguishable

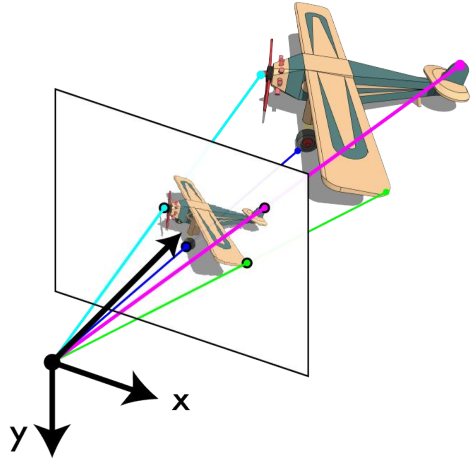


Illustration by JP. Reddie

More on projective transform

Using homography to project point

Multiply $\tilde{x}$ by $\tilde{H}$ to get $\tilde{x'}$

Convert to $\tilde{x'}$ by dividing by $\tilde{w'}$

$$\mathbf{\tilde{x}'} = \mathbf{\tilde{H}}\ \mathbf{\tilde{x}}$$

$$\mathbf{\bar{x}} = \mathbf{\tilde{x}} / \tilde{w}$$

$$\begin{bmatrix} \tilde{x}' \\ \tilde{y}' \\ \tilde{w}' \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix}$$

Summary of 2D transformations

Euclidean

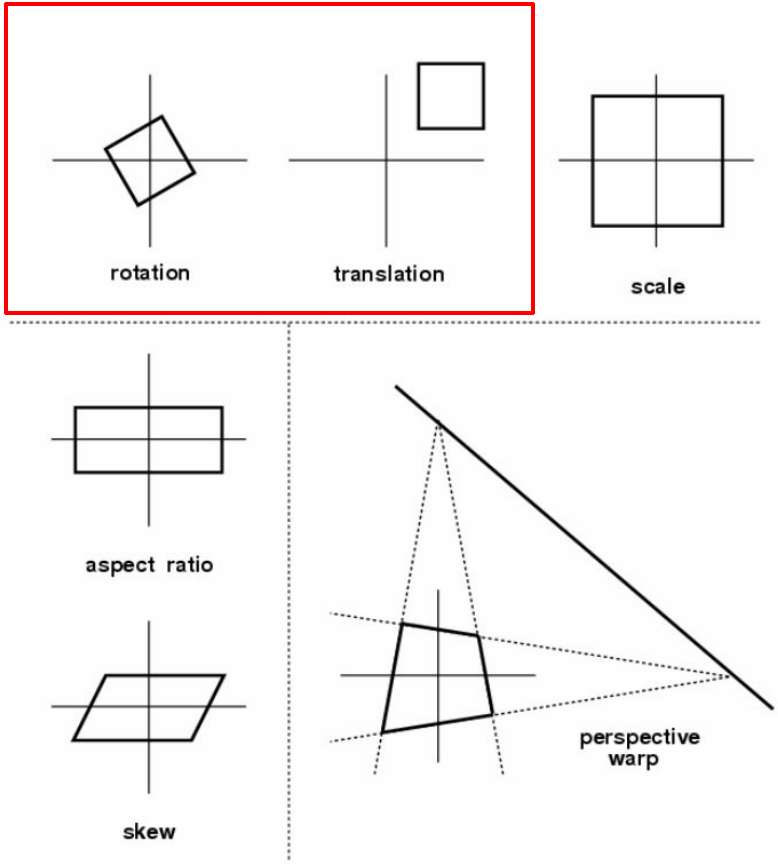


Illustration: Robert Collins

Summary of 2D transformations

Similarity

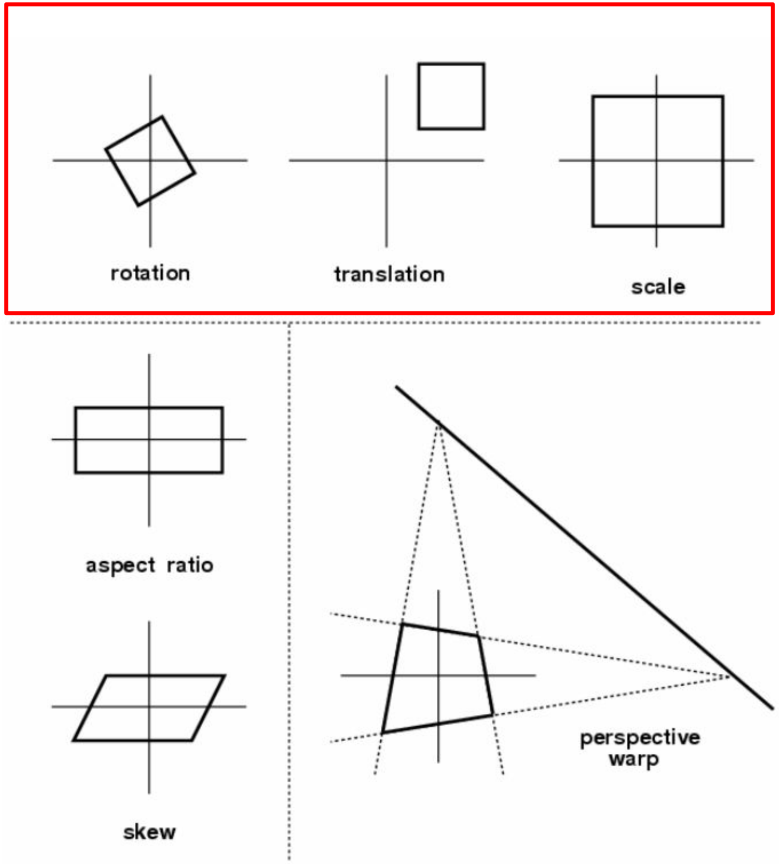


Illustration: Robert Collins

Summary of 2D transformations

Affine

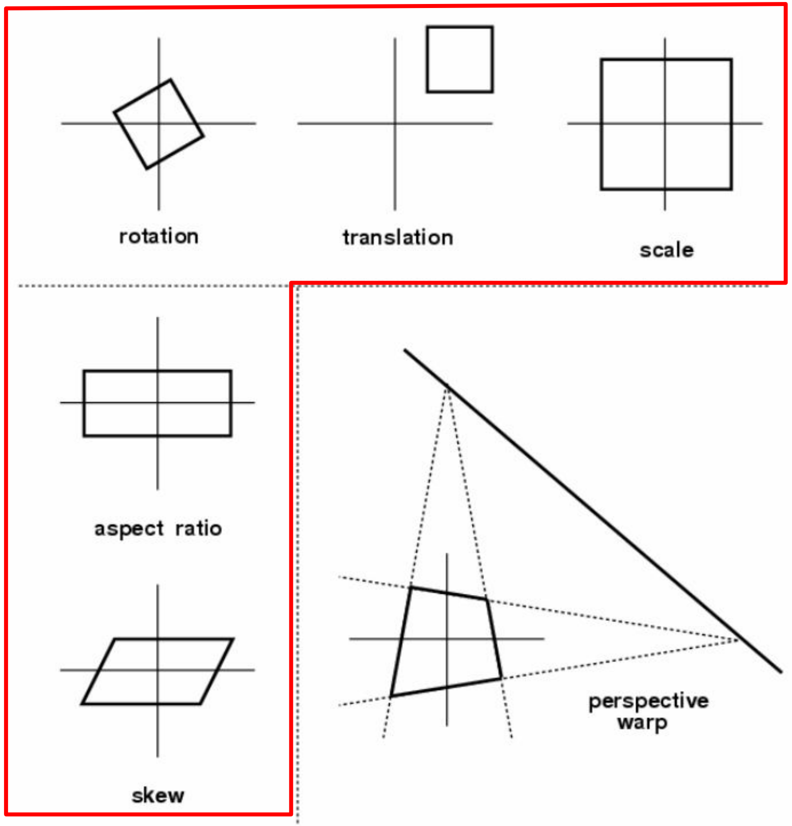


Illustration: Robert Collins

Summary of 2D transformations

Projective

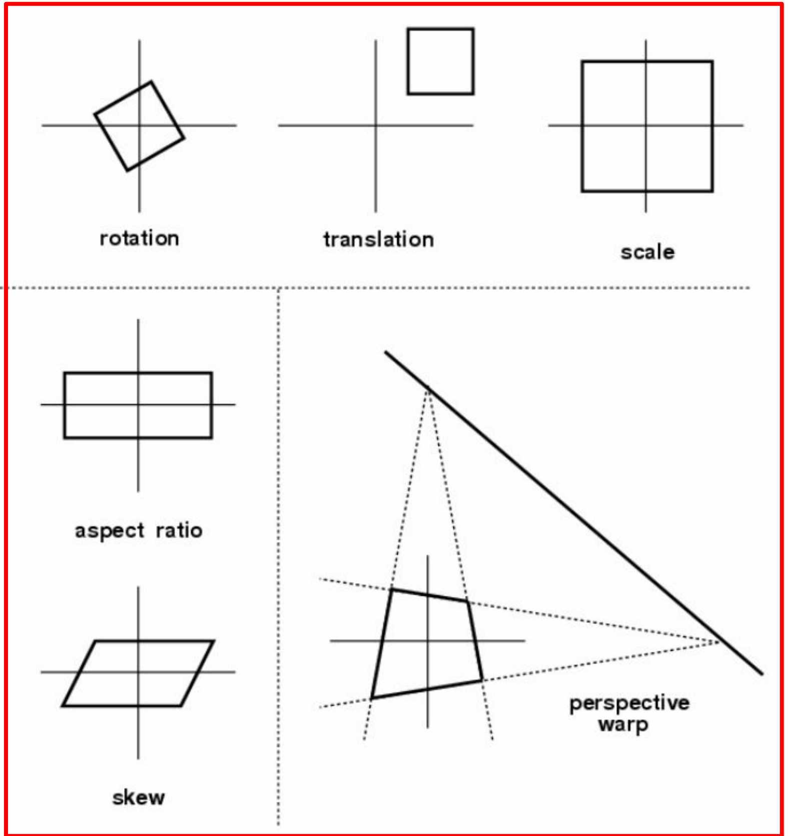


Illustration: Robert Collins

Summary of 2D transformations

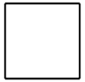
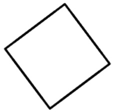
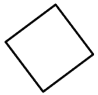
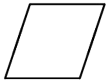
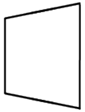
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \vert & \mathbf{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \vert & \mathbf{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \vert & \mathbf{t} \end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

Image Mappings Overview

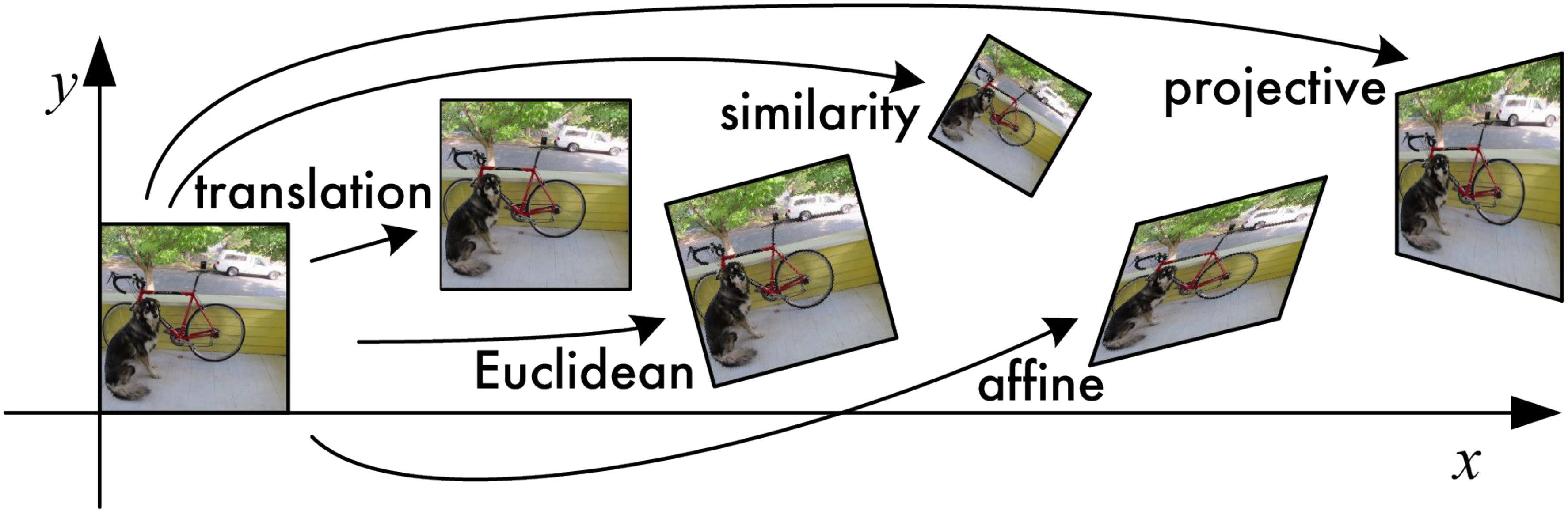
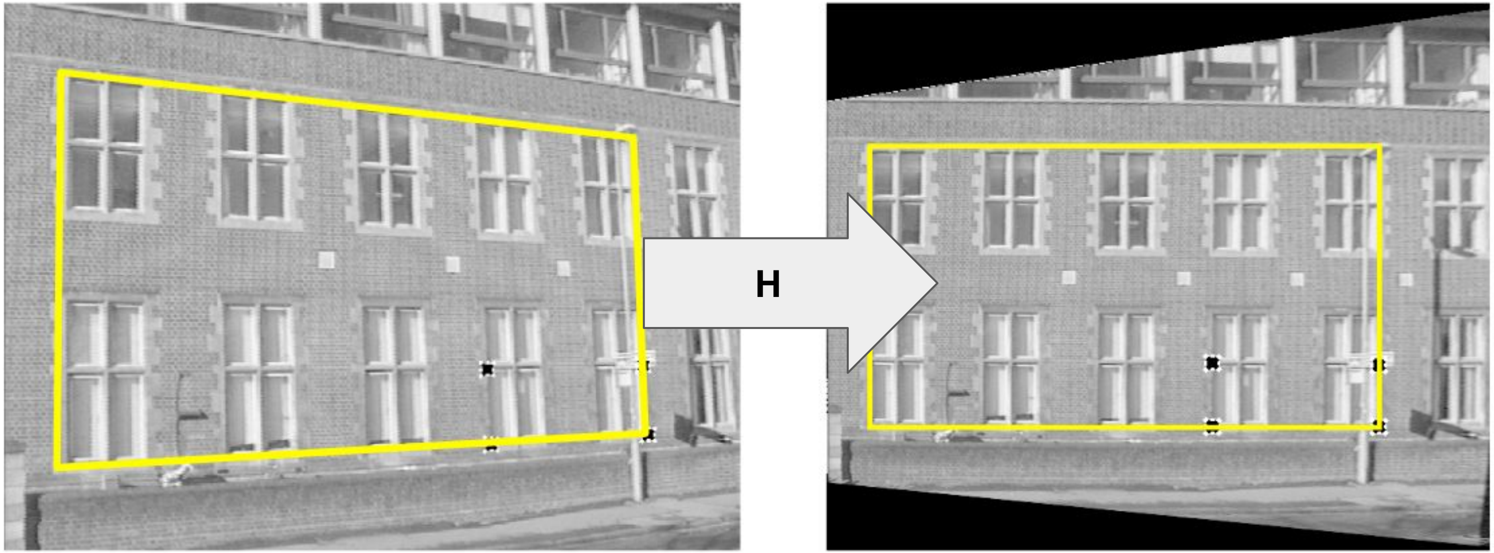


Image adapted from R. Szeliski by JP. Reddie

Warping images

Warping Example



Source Image

Destination image

from Hartley & Zisserman

Warping & Bilinear Interpolation

Given a transformation between two images (coordinate systems)
we want to **“warp” one image** into the **coordinate system** of the **other**.

We will call the coordinate system
where we are **mapping from**
the **“source”** image.

We will call the coordinate system
we are **mapping to**
the **“destination”** image.

Forward Warping

For each pixel x in the source image

Determine where it goes as H(x)

Color the destination pixel

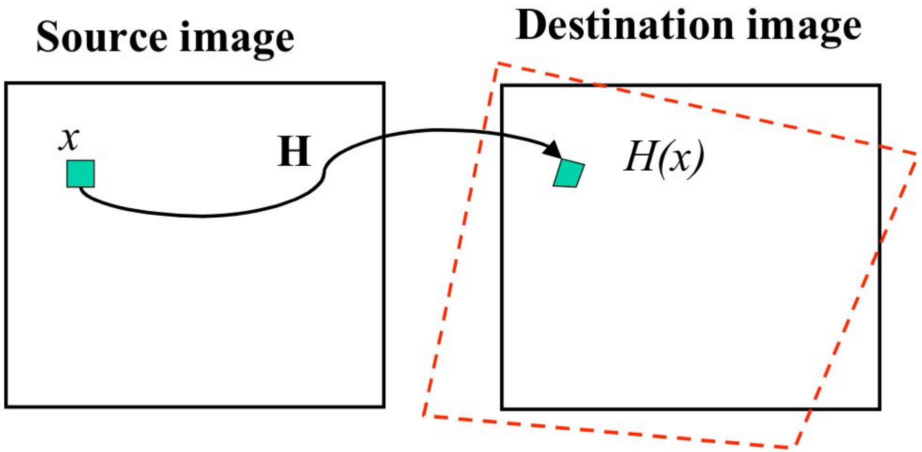


Illustration: Robert Collins

Forward Warping

For each pixel x in the source image

Determine where it

Color the

PROBLEM

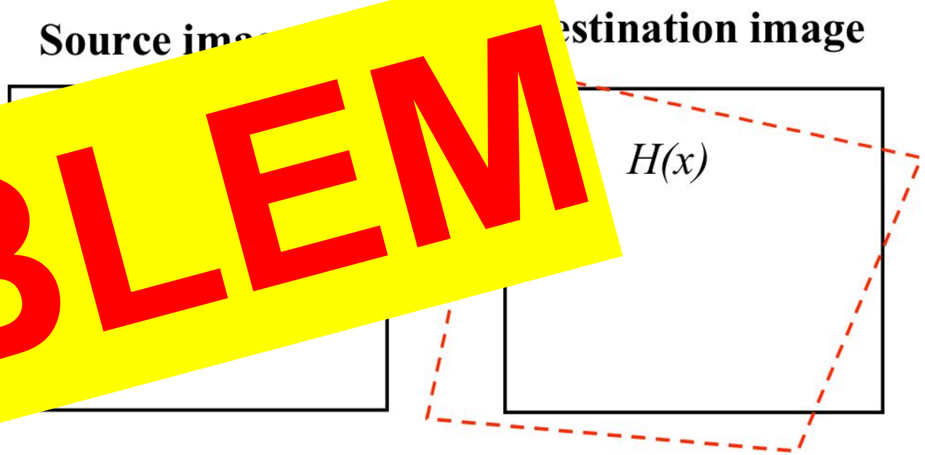


Illustration: Robert Collins

Forward Warping Problem

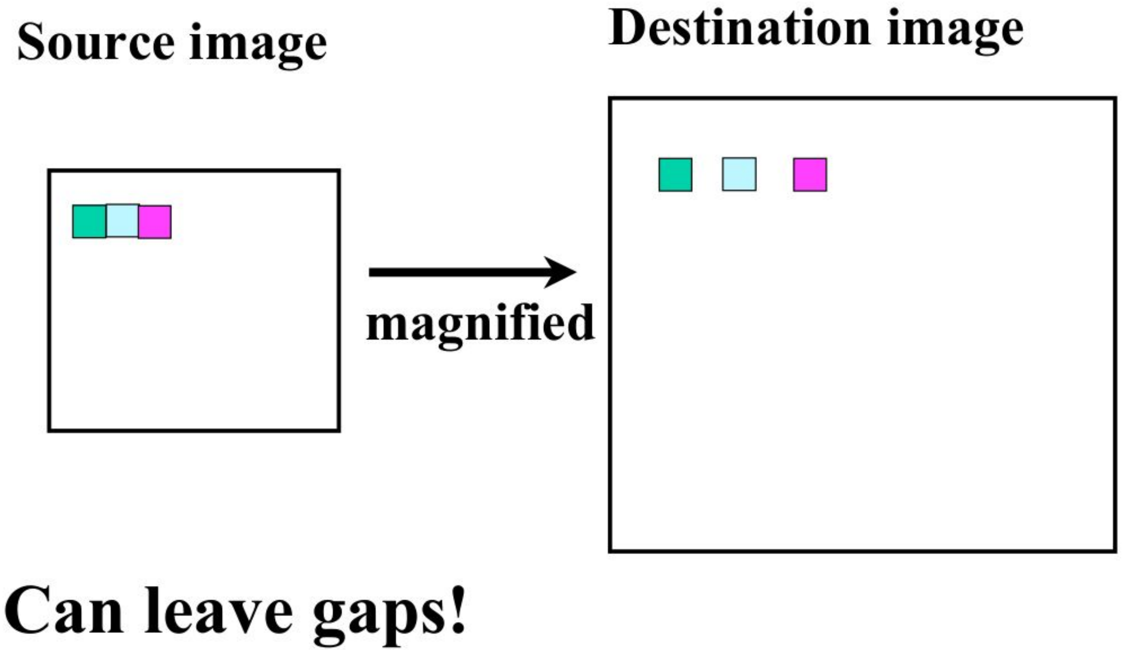
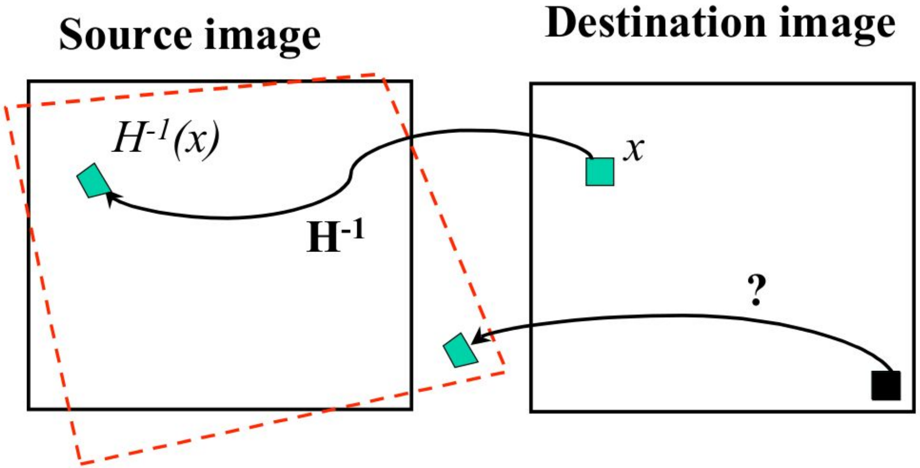


Illustration: Robert Collins

Backward Warping — No gap



For each pixel x in the destination image

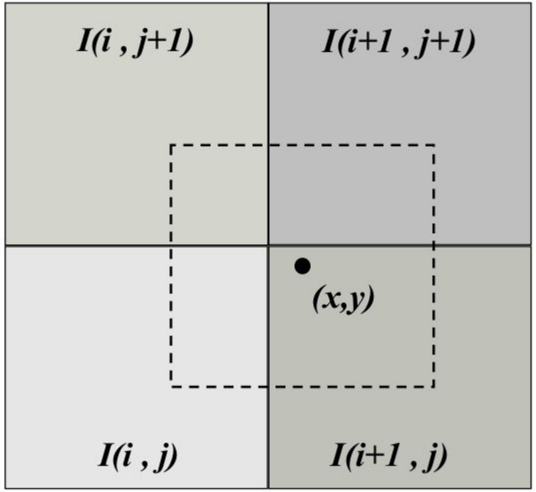
Determine where it comes from as $\mathbf{H}^{-1}$ **(x)**

Get color from that location

Interpolation

What do we mean by “get color from that location”?

Consider grey values. What is intensity at (x,y)?



Need interpolation