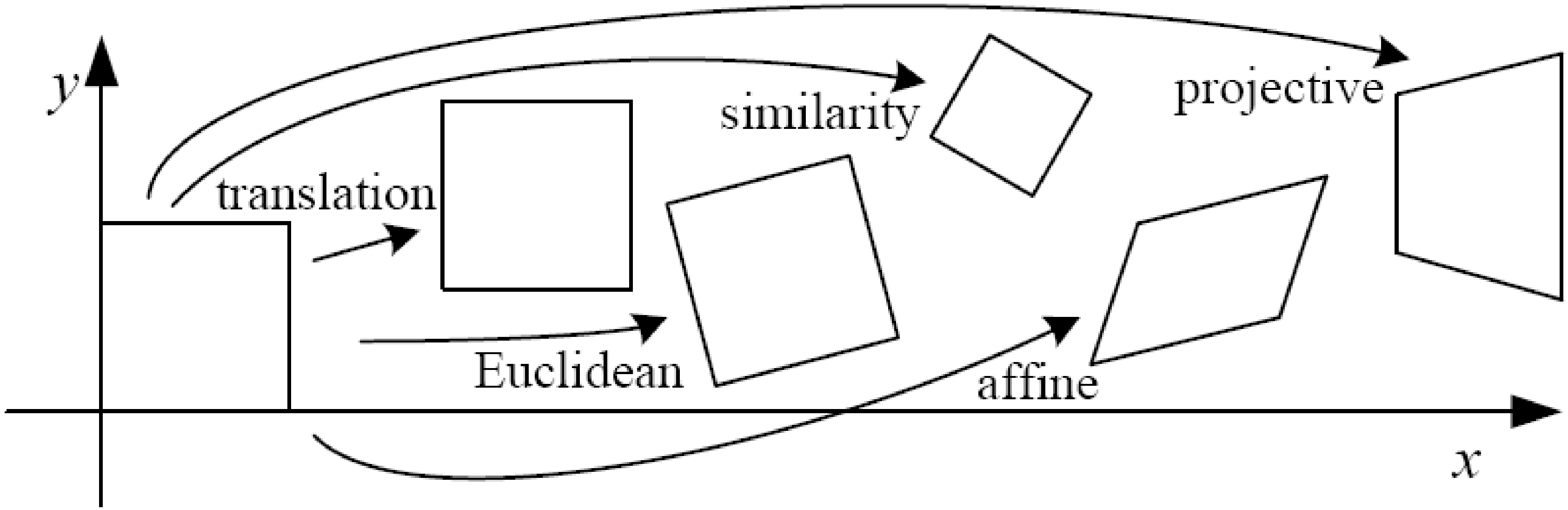
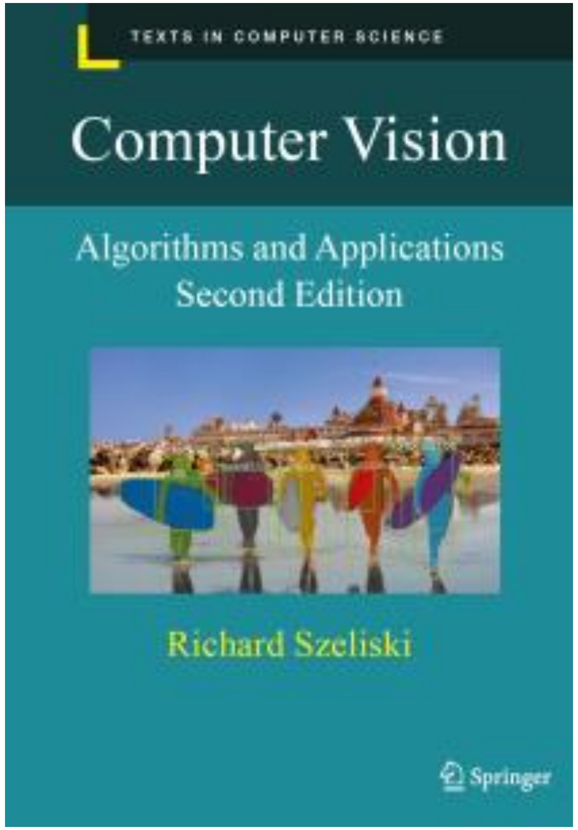
# Computer Vision

## Image transformations and image warping



# Important information




## Textbook

Rick Szeliski, *Computer Vision: Algorithms and Applications* online at: http://szeliski.org/Book/

Many of the slides in this course are modified from the excellent class notes of similar courses offered in other schools by Noah Snavely, Prof Yung-Yu Chuang, Fredo Durand, Alyosha Efros, Bill Freeman, James Hays, Svetlana Lazebnik, Andrej Karpathy, Fei-Fei Li, Srinivasa Narasimhan, Silvio Savarese, Steve Seitz, Richard Szeliski, and Li Zhang. The instructor is extremely thankful to the researchers for making their notes available online. Please feel free to use and modify any of the slides, but acknowledge the original sources where appropriate.

All readings are from Richard Szeliski, Computer Vision: Algorithms and Applications, 2nd Edition, unless otherwise noted.

# Reading

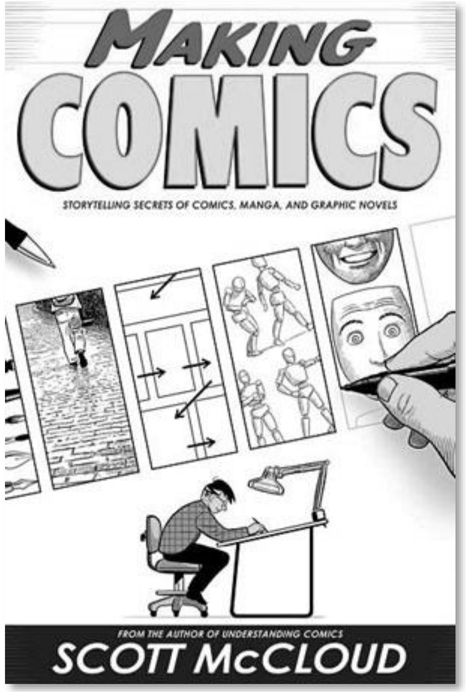
- Szeliski: Chapter 3.6

# Image alignment

# Image alignment

Why don’t these image line up exactly?

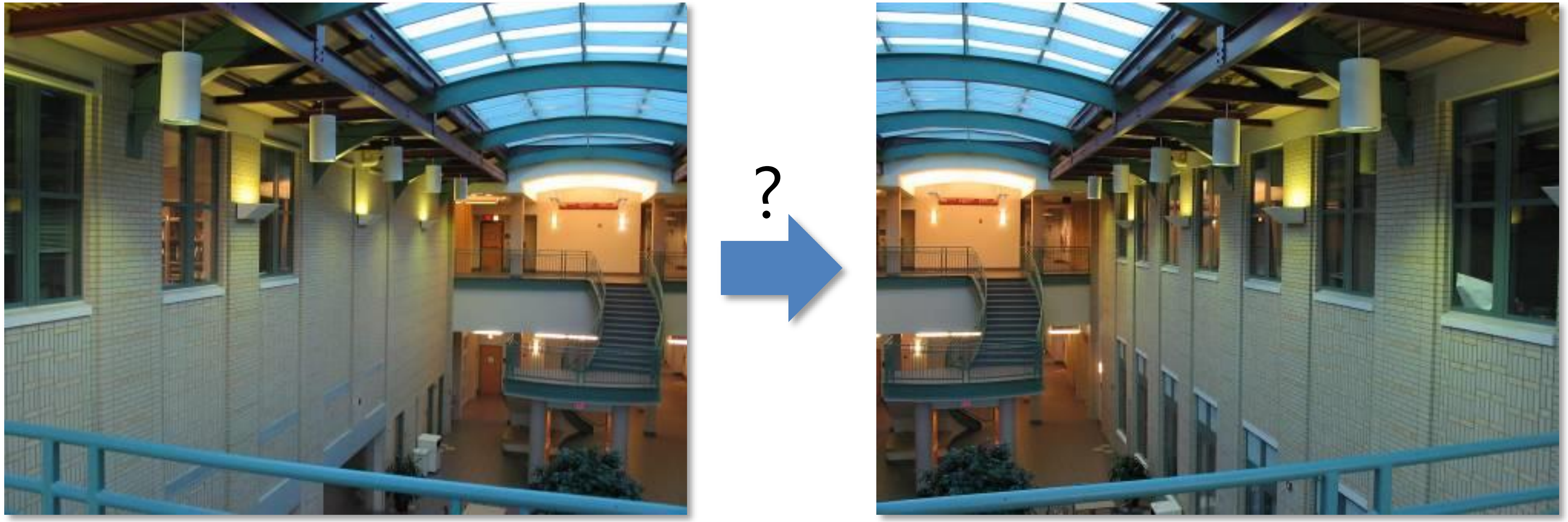
# What is the geometric relationship between these two images?




?


**Answer: Similarity transformation** (translation, rotation, uniform scale)

# What is the geometric relationship between these two images?

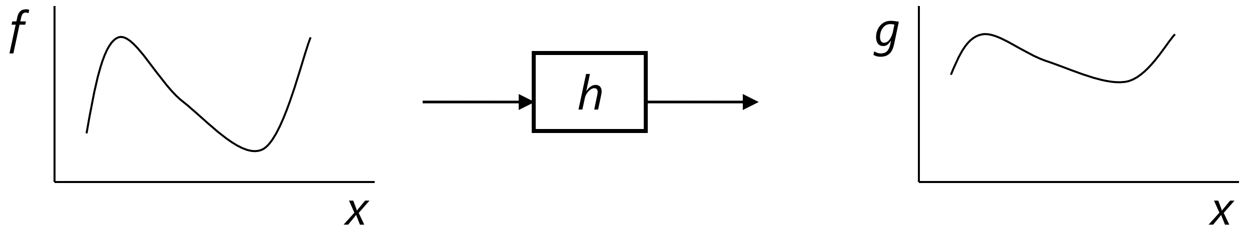


# What is the geometric relationship between these two images?

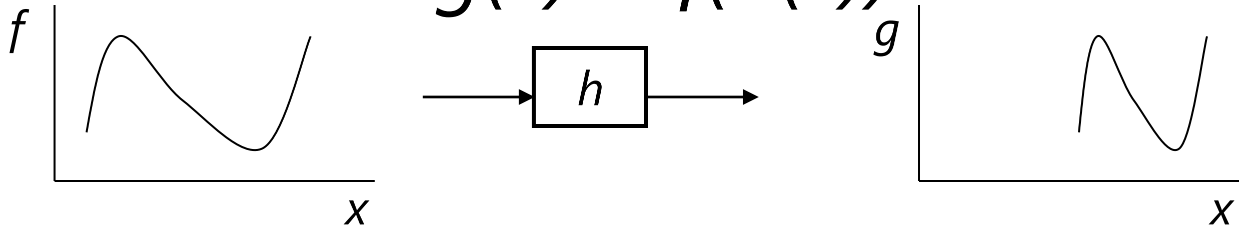
**Very important for creating mosaics!**
First, we need to know what this transformation is.
Second, we need to figure out how to compute it using feature matches.

# Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



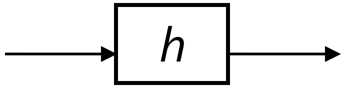

- image warping: change *domain* of image
  - $g(x) = f(h(x))$



# Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$

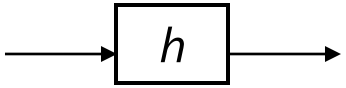
*f*



*g*


- image warping: change *domain* of image
  - $g(x) = f(h(x))$

*f*



*g*


# Parametric (global) warping

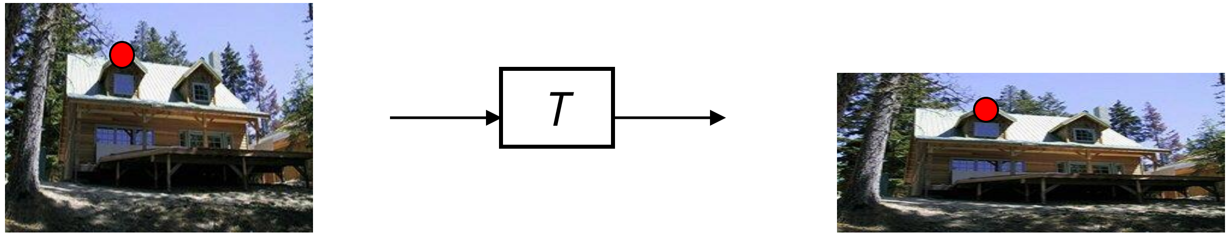
- Examples of parametric warps:

translation

rotation

aspect

# Parametric (global) warping




**p** = (x,y) **p′** = (x′,y′)

- Transformation T is a coordinate-changing machine:

$$\mathbf{p}' = T(\mathbf{p})$$

- What does it mean that *T* is global?
  - Is the same for any point **p**
  - can be described by just a few numbers (parameters)
- Let's consider *linear* xforms (can be represented by a 2x2 matrix):

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

# Common linear transformations

- Uniform scaling by *s*:

(0,0)

(0,0)

$$\mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

What is the inverse?

# Common linear transformations

- Rotation by angle $\theta$ (about the origin)


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

What is the inverse?

For rotations:

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

# 2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D mirror across Y axis?

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned}$$

2D mirror across line y = x?

$$\begin{aligned} x' &= y \\ y' &= x \end{aligned}$$

# 2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D mirror across Y axis?

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D mirror across line y = x?

$$\begin{aligned} x' &= y \\ y' &= x \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

# 2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}$$

# 2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}$$

NO!

Translation is not a linear operation on 2D coordinates

# All 2D Linear Transformations

- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror

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

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

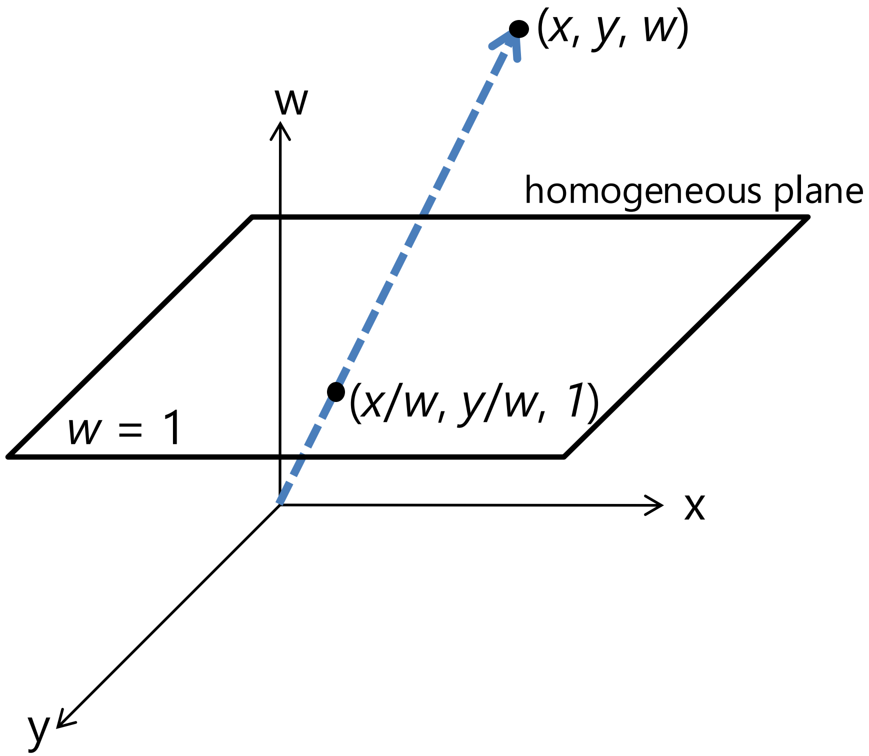
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

# Homogeneous coordinates

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates




Converting *from* homogeneous coordinates

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

# Translation

- Solution: homogeneous coordinates to the rescue

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

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

# Affine transformations

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

any transformation represented by a 3x3 matrix with last row [ 0 0 1 ] we call an *affine* transformation

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

# Basic affine transformations

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

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

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

2D *in-plane* rotation

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

Shear

# Affine transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

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

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

# Where do we go from here?

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

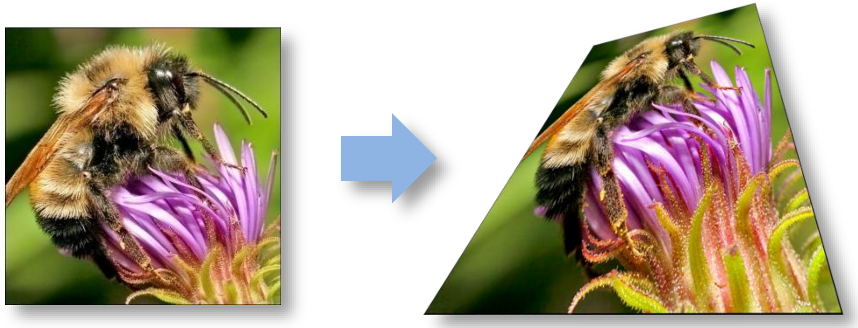
what happens when we mess with this row?

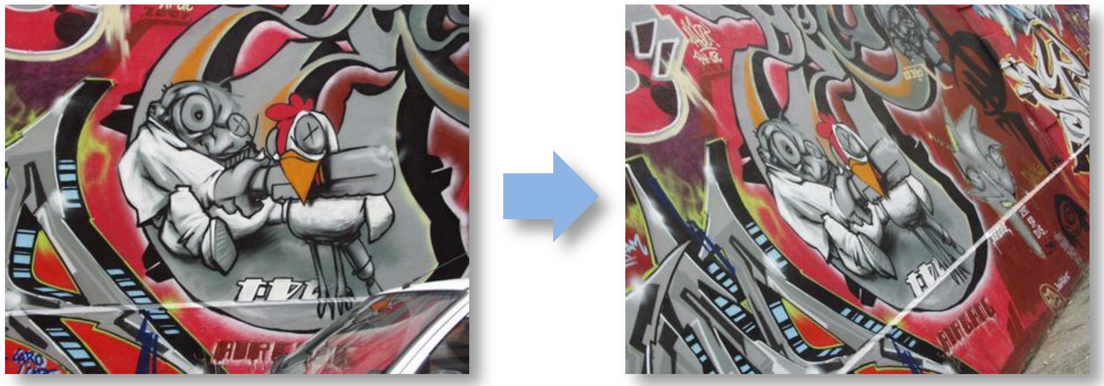
affine transformation

# Projective Transformations *aka* Homographies *aka* Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)





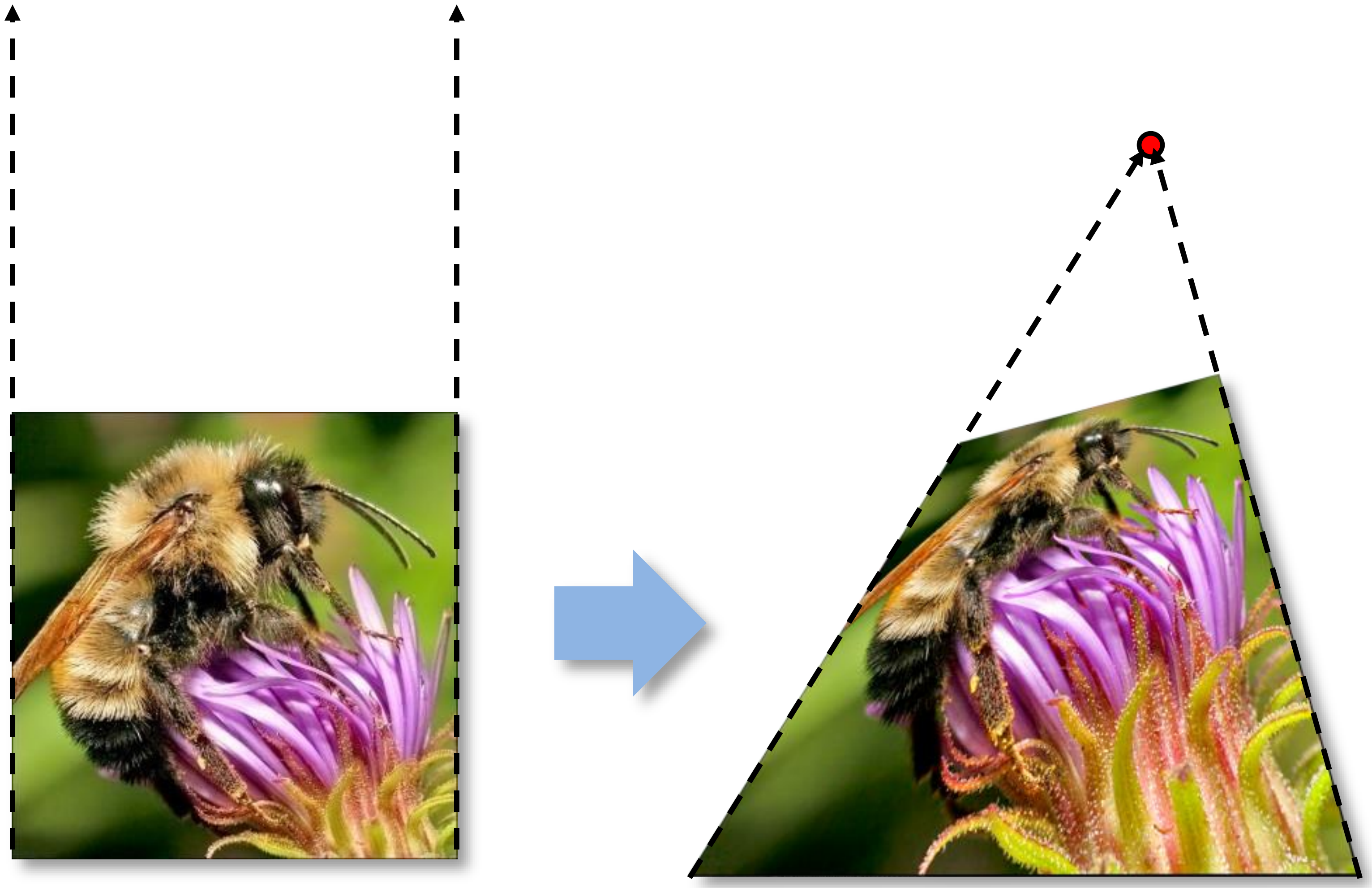
# Homographies

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

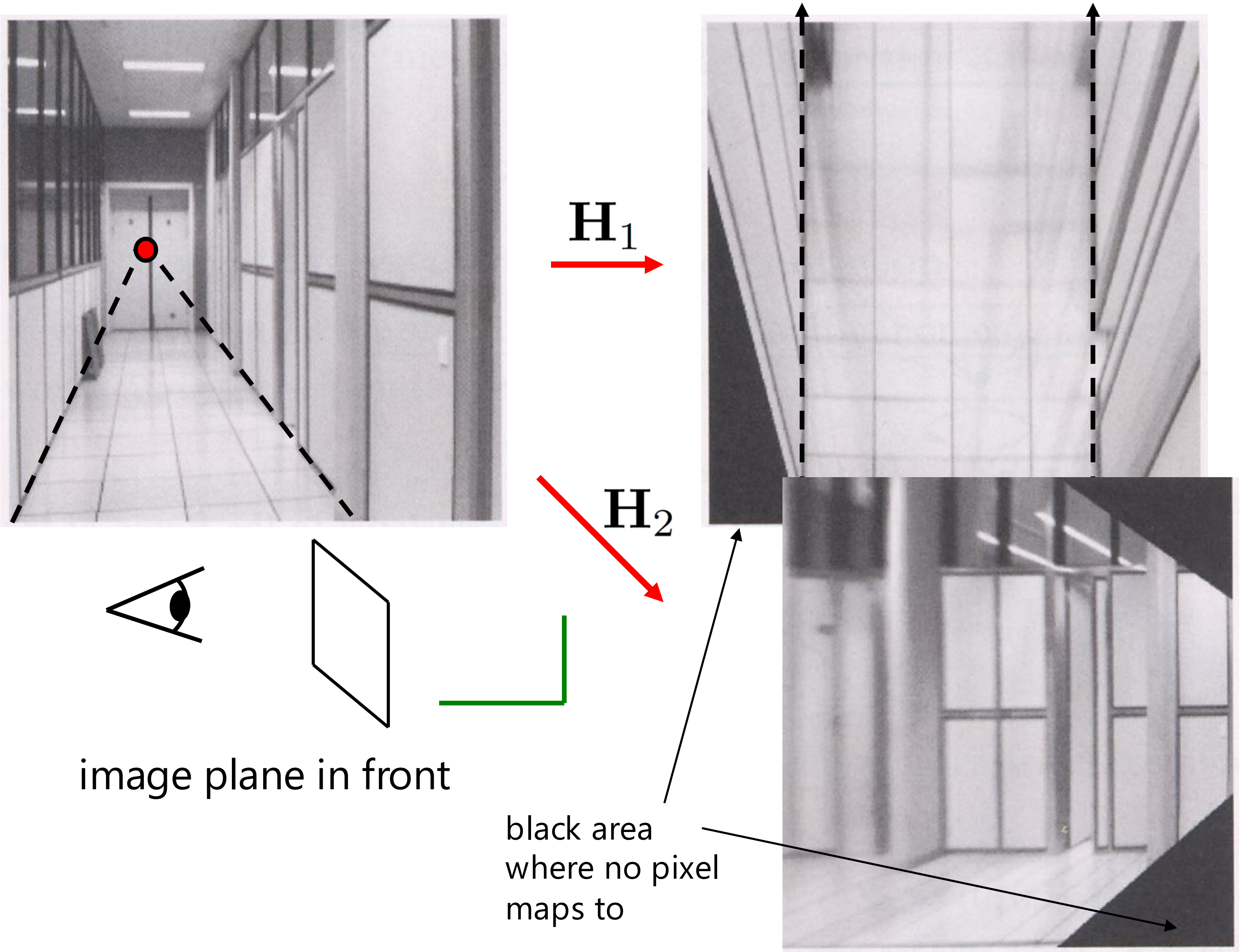
**What happens when the denominator is 0?**

$$\sim \begin{bmatrix} \frac{ax+by+c}{gx+hy+1} \\ \frac{dx+ey+f}{gx+hy+1} \\ 1 \end{bmatrix}$$

# Points at infinity



# Image warping with homographies



# Homographies

# Homographies

- Homographies …
  - Affine transformations, and
  - Projective warps

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

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition

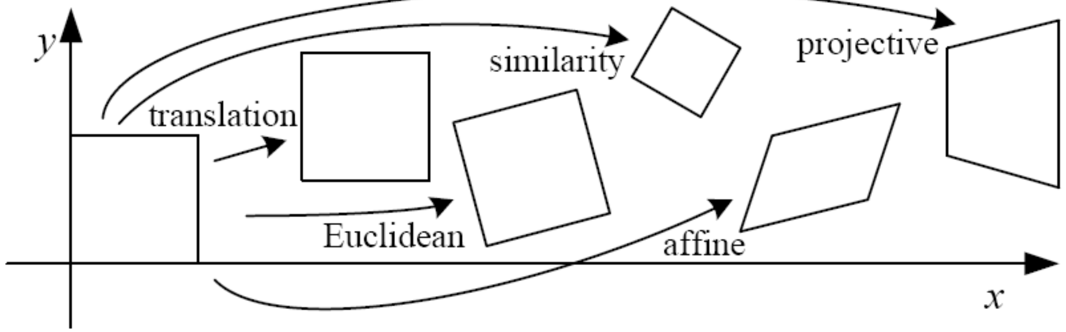
- Key fact: homographies are only defined up to a scale factor (e.g., ***H*** and 2***H*** are equivalent homographies)

# Alternate formulation for homographies

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

where the length of the vector [$h_{00}$ $h_{01}$ … $h_{22}$] is 1

# 2D image transformations



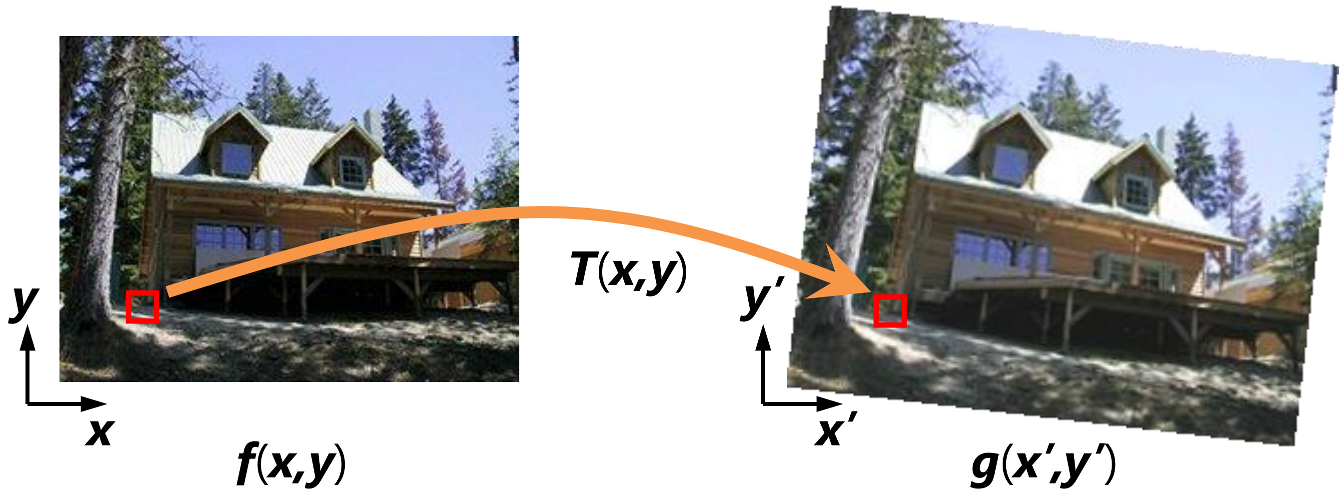

| Name | Matrix | # D.O.F. | Preserves: | Icon |
|---|---|---|---|---|
| translation | $\begin{bmatrix} I \mid t \end{bmatrix}_{2\times 3}$ | 2 | orientation $+ \cdots$ | |
| rigid (Euclidean) | $\begin{bmatrix} R \mid t \end{bmatrix}_{2\times 3}$ | 3 | lengths $+ \cdots$ | |
| similarity | $\begin{bmatrix}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

These transformations are a nested set of groups

- Closed under composition and inverse is a member

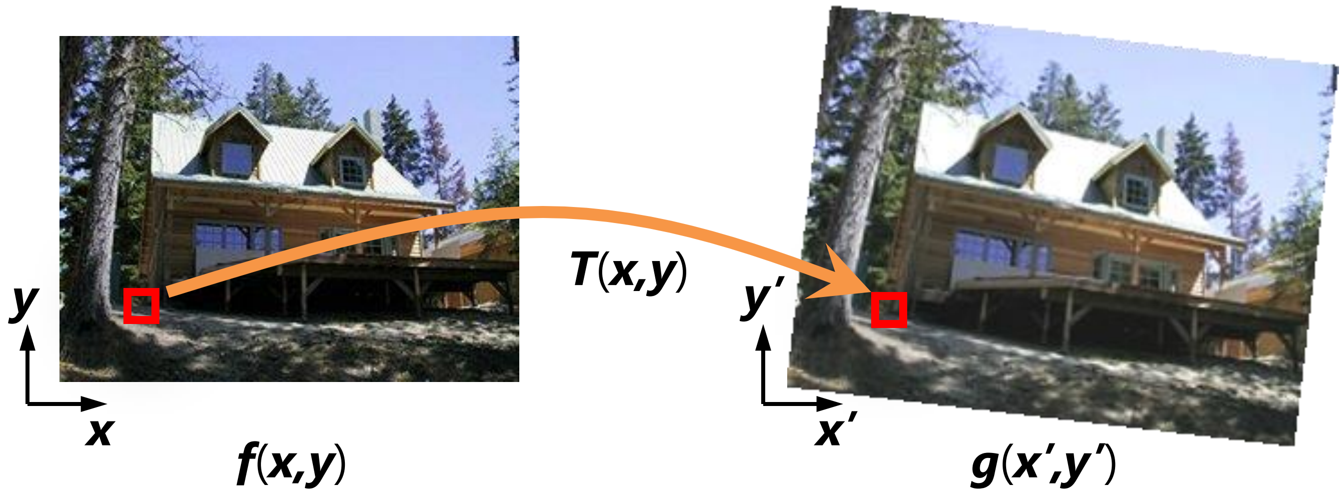
# Implementing image warping

- Given a coordinate xform $(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y})$ and a source image $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{f}(\boldsymbol{T}(\boldsymbol{x},\boldsymbol{y}))$?



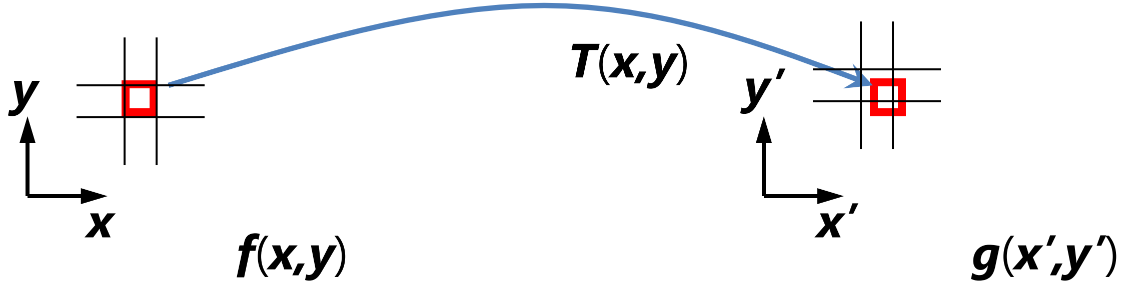
# Forward Warping

- Send each pixel $(\boldsymbol{x},\boldsymbol{y})$ to its corresponding location $(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y})$ in $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'})$
  - What if pixel lands “between” two pixels?



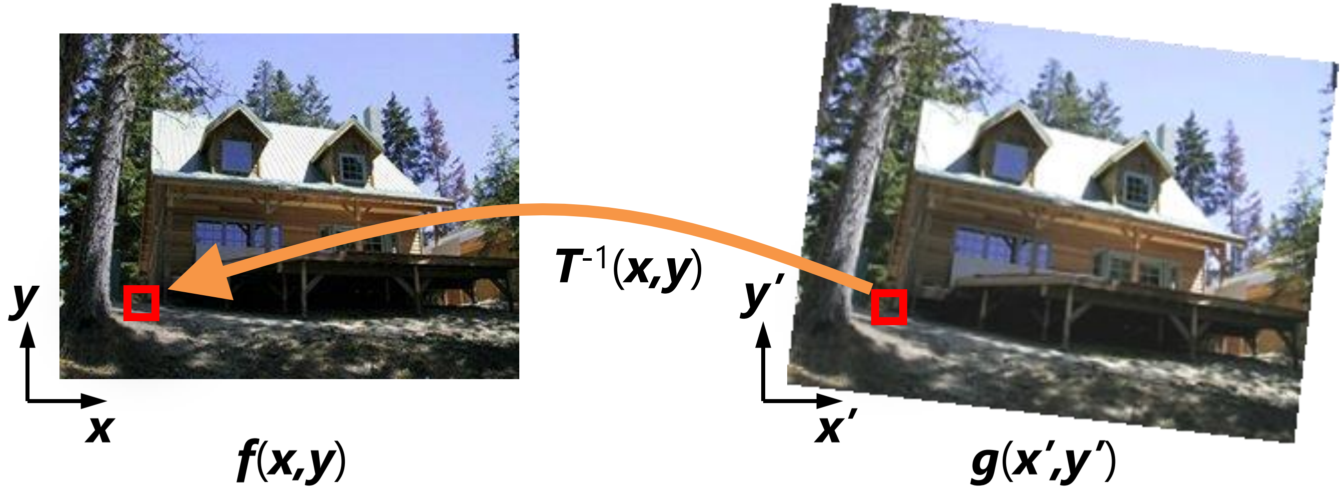
# Forward Warping

- Send each pixel (***x,y***) to its corresponding location (***x′,y′***) = ***T***(***x,y***) in ***g***(***x′,y′***)
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)
  - Can still result in holes



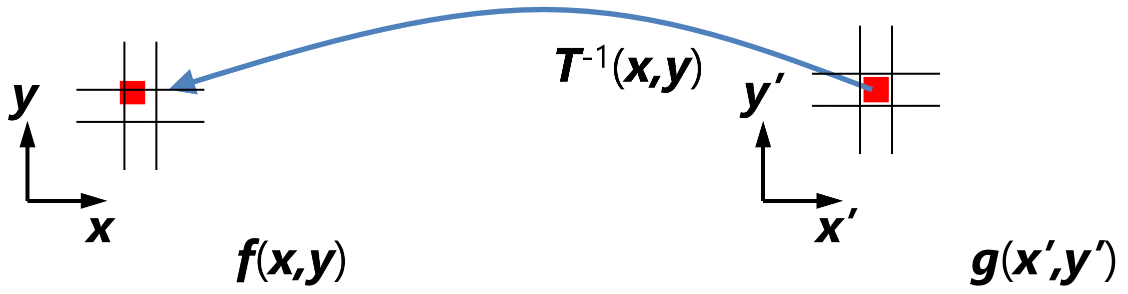
# Inverse Warping

- Get each pixel $\boldsymbol{g(x',y')}$ from its corresponding location $(\boldsymbol{x,y}) = \boldsymbol{T}^{-1}(\boldsymbol{x,y})$ in $\boldsymbol{f(x,y)}$
    - Requires taking the inverse of the transform
    - What if pixel comes from “between” two pixels?



# Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'})$ from its corresponding location $(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{T}^{-1}(\boldsymbol{x'},\boldsymbol{y'})$ in $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



# Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic
  - sinc
- Needed to prevent "jaggies" and "texture crawl"

  (with prefiltering)

# Questions?