CS-GY 6643
Computer Vision

Lecture 6: Image warping and stitching

Prof. Erdem Varol

Today's menu

Announcements

Image transformations

Fitting transformations

Image warping

Coding lab

Announcements

Final project proposal due date changed

Now due at October 11, 11:59pm

Additional office hours to discuss project directions

Friday 10/11 - 10a-1p

Appointment booking @ https://calendar.app.google/GHgsCYBdyBJmw2m97

20% of your final project grade (4 out of 20 pts)

Proposal template here:

https://www.overleaf.com/read/rhdrqnmsyzpv#c1b706

(Please copy this template in your own account)

Must complete as a team - one person may submit on brightspace.

What we are looking for

1. Well motivated problem statement + background literature review (1pt)
2. You have an appropriate dataset in hand and prove it through visualization (1pt)
3. Baseline method on the data (1pt)
   a. Can be a related paper with working code
   b. or a classic algorithm if you show that there is no such paper
4. Plan for the final project, proposed method idea / architecture and evaluation metrics (1pt)

Final project - time management

I highly recommend setting up regular short office hour visits every week or 2 weeks to discuss project updates with the prof (or TAs) + teammates

Appointment booking @ https://calendar.app.google/GHgsCYBdyBJmw2m97

Don’t cram final project to last week or even last month. New datasets and baselines take A LOT of time and will benefit from regular progress by everyone.

Homework 2 will be out tomorrow by evening

Due October 18, 11:59pm

Will cover:

Feature detectors and descriptors

Convolutions

Robust model fitting

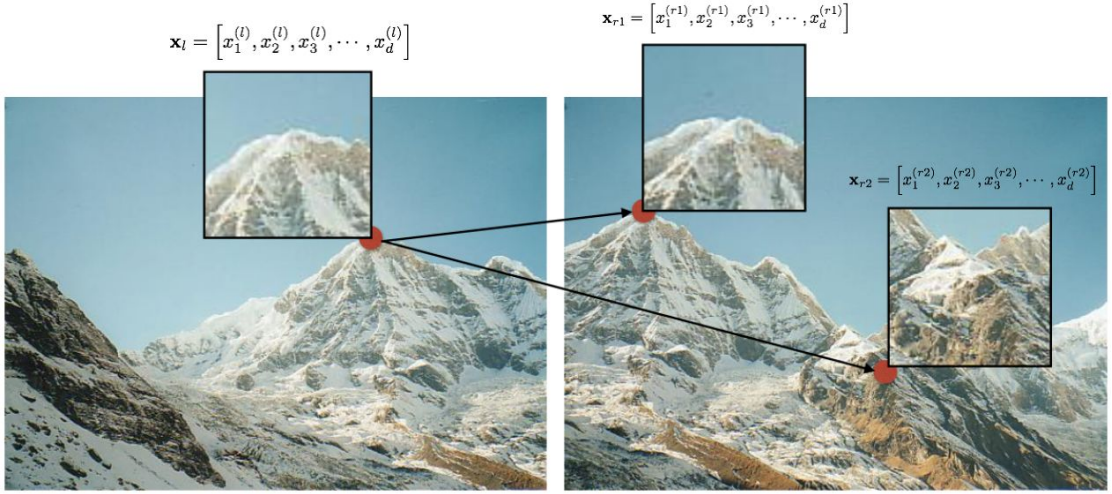
Image transformations and stitching

Midterm exam in 2 weeks (15% of grade)

- October 24, 11am. In-class, pen and paper. No laptops.
- Will test the ability to derive key mathematical concepts on the spot.
- It will also test the ability to “think like AI” in several image analysis scenarios.
- The exam will be designed to be completed in 1:15 mins although you are welcome to stay the full 2.5 hour time slot if needed.

Image transformations

Image alignment / stitching



1) Detect “important” points - e.g. Harris corner detector (Lecture 4)
2) Describe the information round them - e.g. SIFT (Lecture 4)
3) Match points with similar descriptions - L2 distance (Lecture 4)
4) Handle outliers - RANSAC or Hough (Lecture 5)
5) Learn a geometric transformation - **TODAY**
6) Warp image - **TODAY**

Motivation - cameras have limited field of view

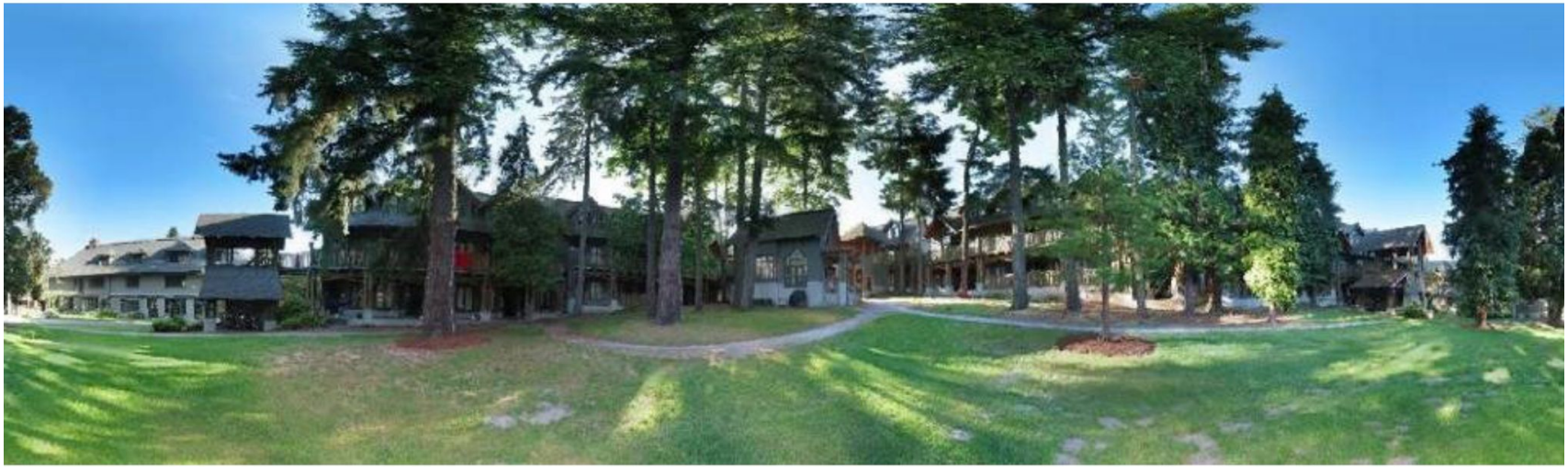
- Compact Camera FOV = 50 x 35°

Motivation - our eyes are “wide angle”

- Compact Camera FOV = 50 x 35°
- Human FOV = 200 x 135°

Motivation - we can go beyond our limits through stitching

- Compact Camera FOV = 50 x 35°
- Human FOV = 200 x 135°
- Panoramic Mosaic = 360 x 180°



“Gigapixel images”

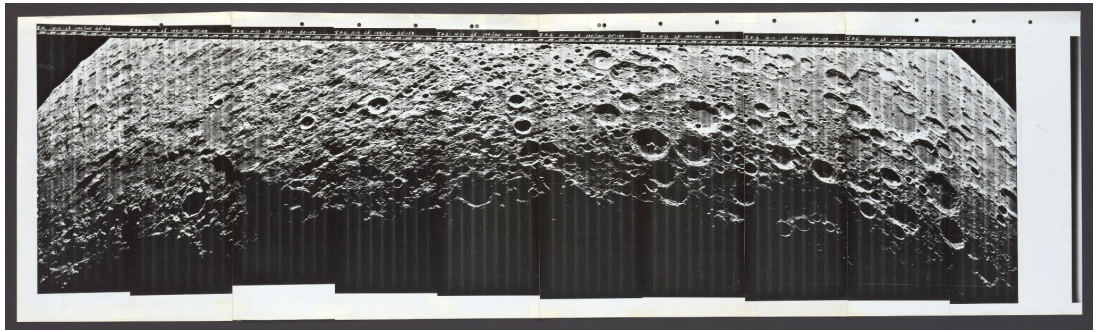
This 120 Gigapixel Photo is the Largest of New York City Ever Taken

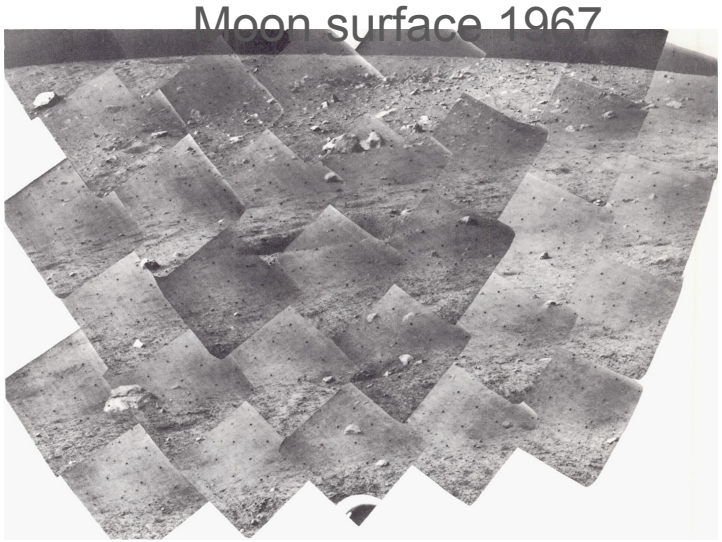
APR 27, 2021 JARON SCHNEIDER

EarthCam recently unveiled the GigapixelCam X80, which is a robotic camera that can produce more than 80,000-megapixel panoramas. To prove it, the company used the X80 to make the highest resolution photo of New York ever captured: 120,000-megapixels.

https://www.earthcam.net/projects/empirestatebuilding/gigapixelpanorama/2021/

Space photography



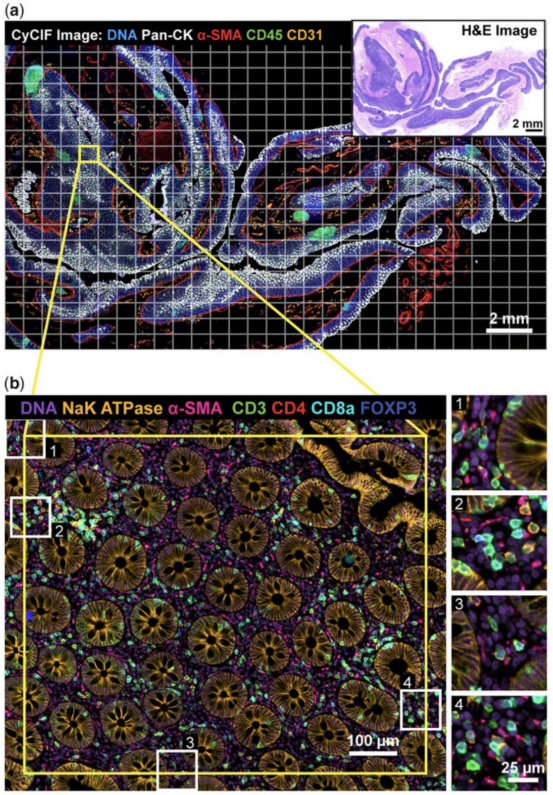


https://en.wikipedia.org/wiki/Surveyor_3

Apollo 17 (1972) https://en.wikipedia.org/wiki/Apollo_17

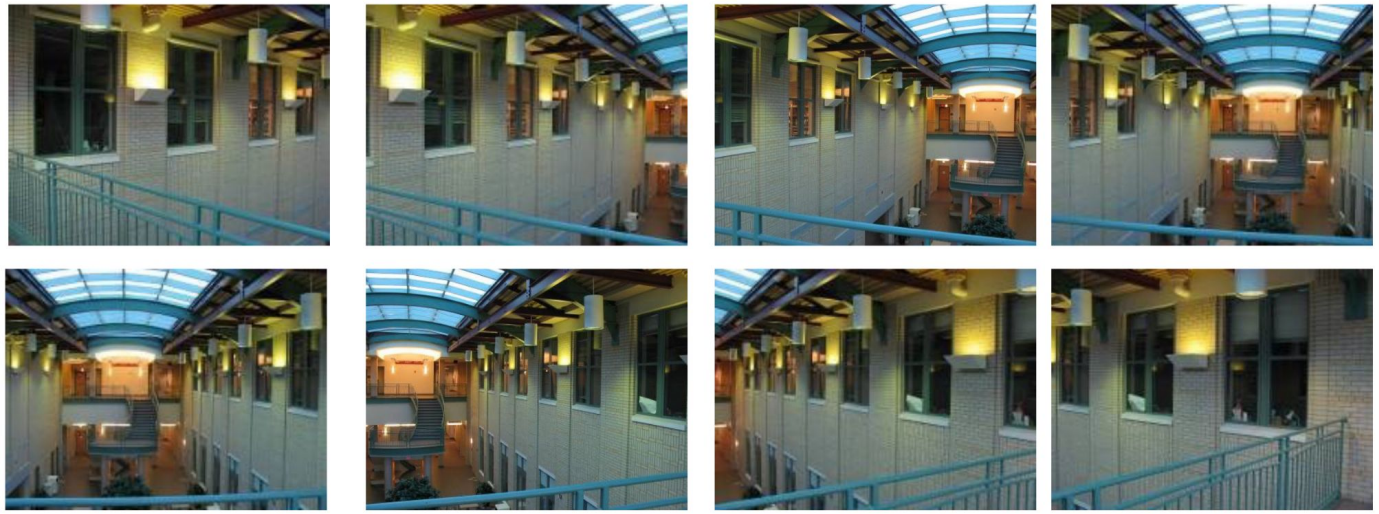
Mosaicing high-res microscopy

https://portal.brain-map.org/atlases-and-data/bkp/abc-atlas



https://academic.oup.com/bioinformatics/article/38/19/4613/6668278

Suppose we do cross-correlation like alignment we did in project 1:



Stitching errors emerge under incorrect transformations

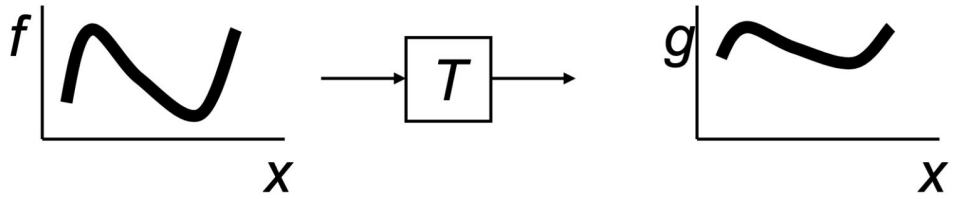
Translation only via alignment

What we actually want

Recall filtering…

Image filtering: change range of image

$$g(x) = T(f(x))$$



Warping / transformation vs. filtering

Image filtering: change range of image

$$g(x) = T(f(x))$$

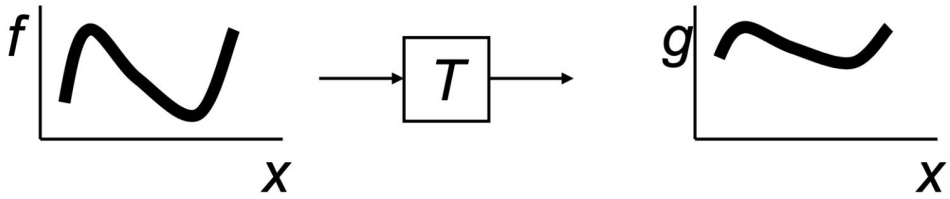
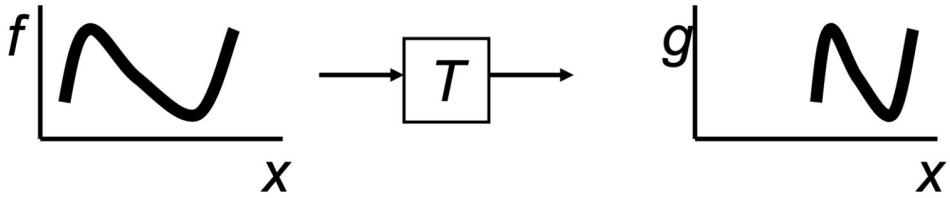
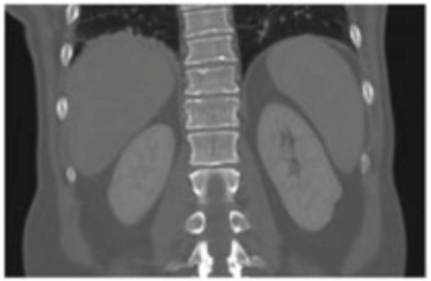


Image warping: change ***domain*** of image

$$g(x) = f(T(x))$$



Recall image representations



Image

83	100	240
22	239	159
143	242	5

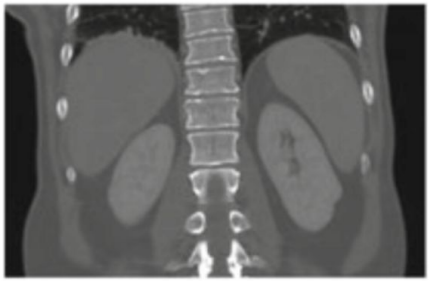
Matrix representation
(Lecture 2)

$(x_i, y_i, I_i = I[y_i, x_i])$

0, 0, 83
1, 0,100
2, 0, 240
0, 1, 222
1, 1, 239
2, 1, 159
0, 2, 143
1, 2, 242
2, 2, 5

Point cloud representation
(Combined with coordinates)

Recall image representations



Image

83	100	240
22	239	159
143	242	5

Matrix representation
(Lecture 2)

$(x_i, y_i, I_i = I[y_i, x_i])$

domain

range

0, 0,	83
1, 0,	100
2, 0,	240
0, 1,	222
1, 1,	239
2, 1,	159
0, 2,	143
1, 2,	242
2, 2,	5

Point cloud representation
(Combined with coordinates)

Example of filtering vs. transformations

Image filtering: change range of image

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

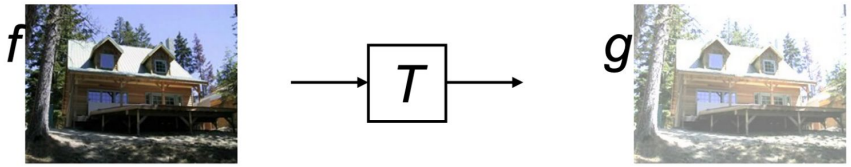
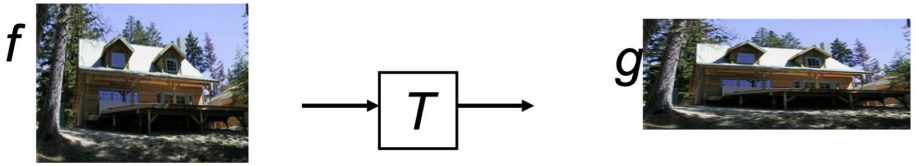


Image warping: change ***domain*** of image

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



Parametric (Global) warping

Examples of parametric warps

translation

rotation

aspect

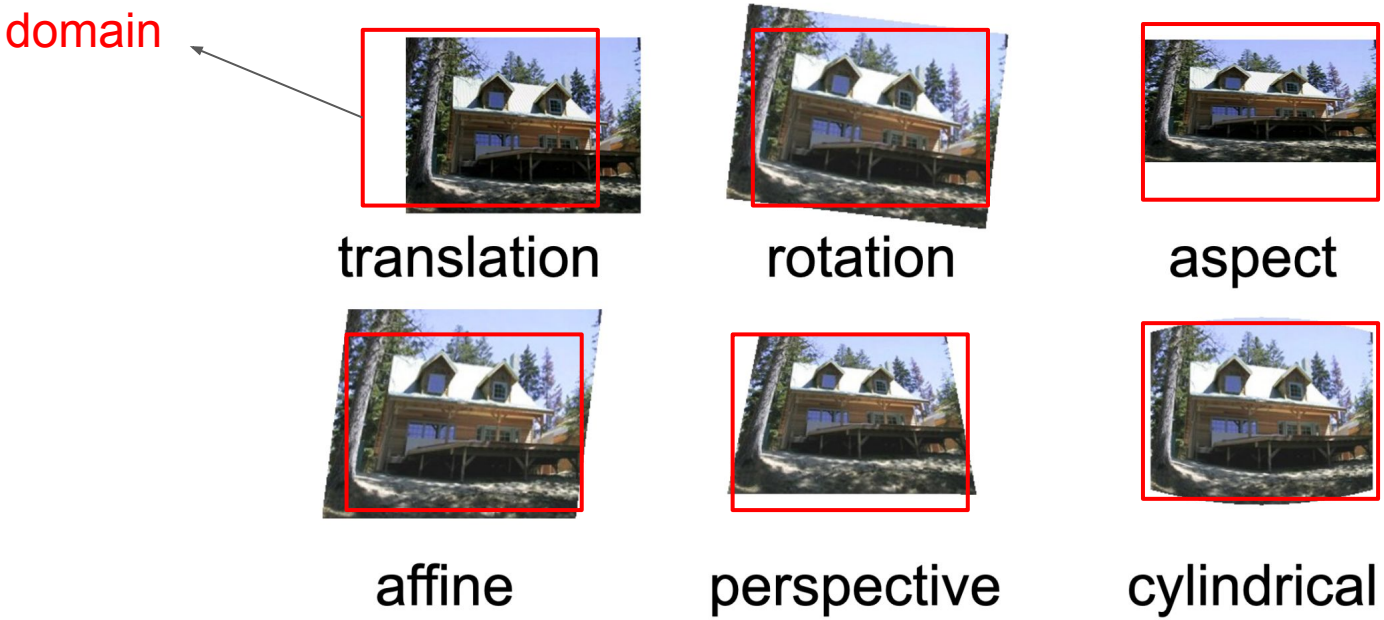
affine

perspective

cylindrical

Parametric (Global) warping

Examples of parametric warps

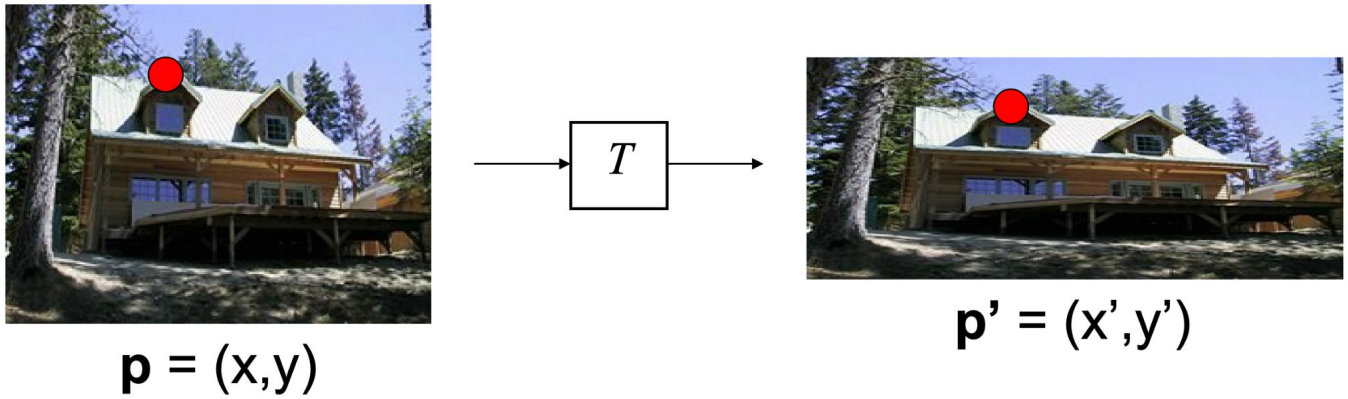


Parametric (Global) Warping

T is a coordinate changing machine

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

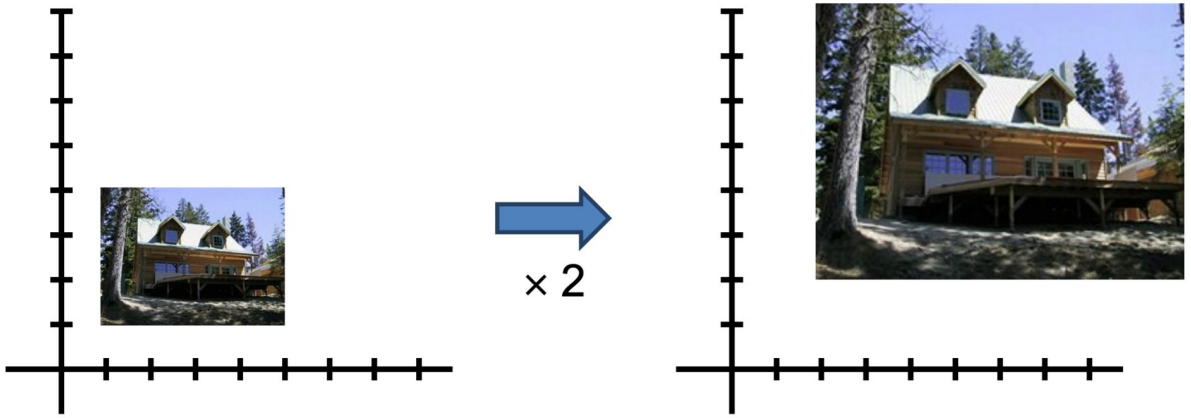
Note: T is the same for all points, has relatively few parameters, and does **not** depend on image content



Let’s start with scaling

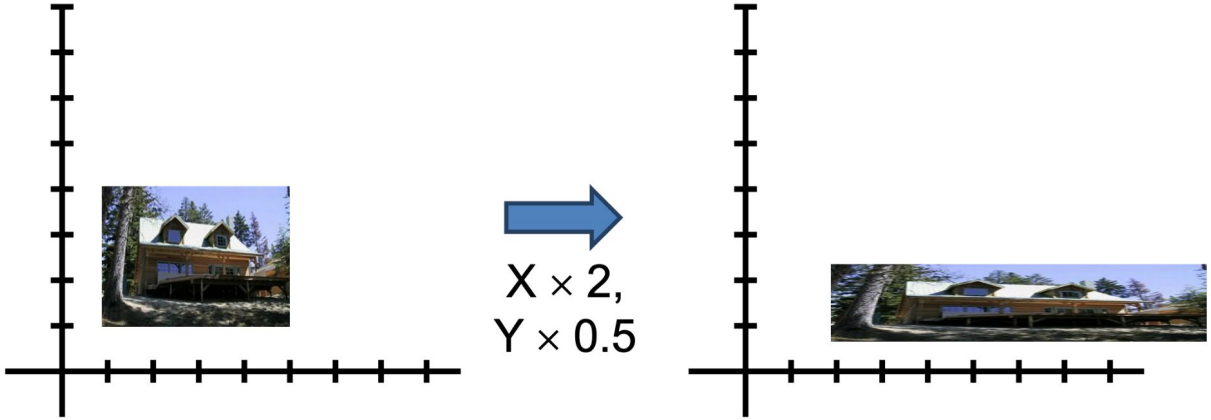
Scaling multiplies each component (x,y) by a scalar.
Uniform scaling is the same for all components.

Note the corner goes from (1,1) to (2,2)



Let’s start with scaling

Non-uniform scaling multiplies each component by a different scalar.



Scaling

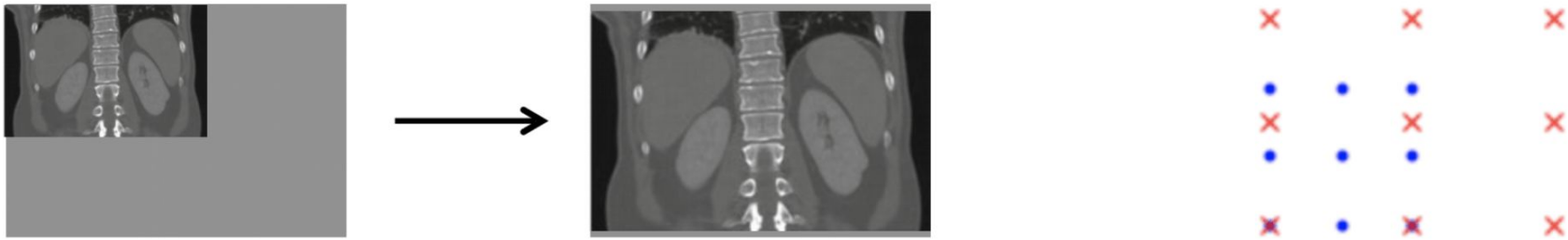
What does T look like?

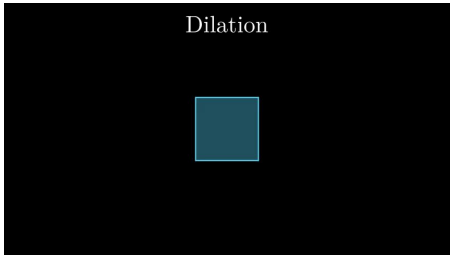
$$x' = ax$$
$$y' = by$$

Let's convert to a matrix:

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

How can we represent <u>scaling</u> as a matrix operation?

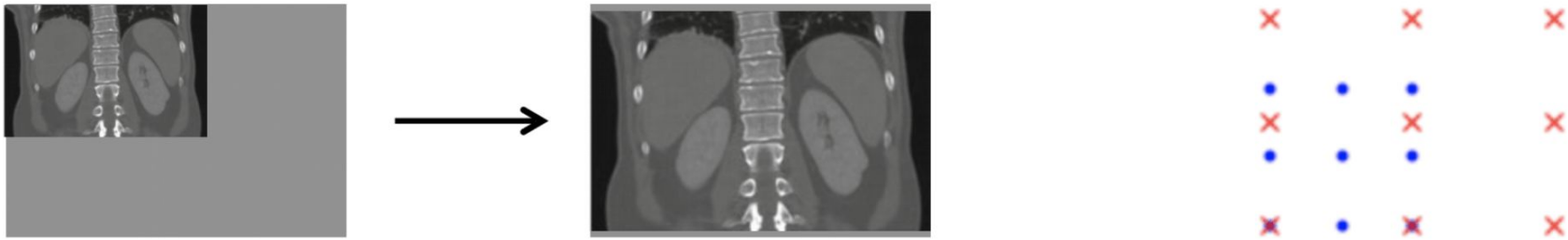




$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

e.g., (1, 2, 83) → (a, 2b, 83)

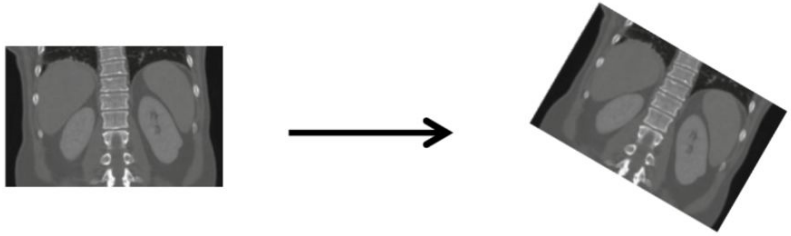
How can we represent <u>scaling</u> as a matrix operation?

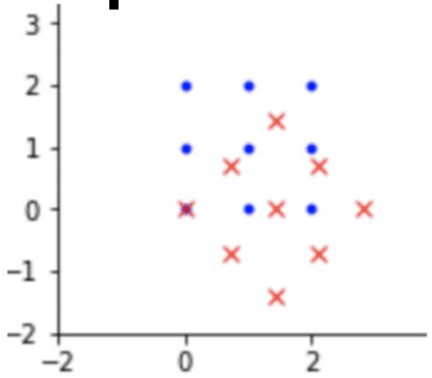


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

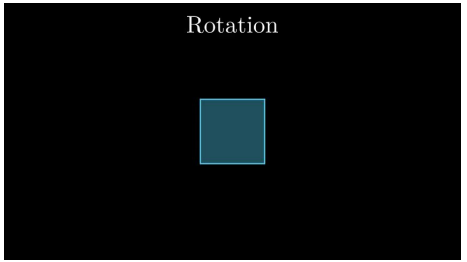
What's the inverse of S?

How can we represent rotation as a matrix operation?



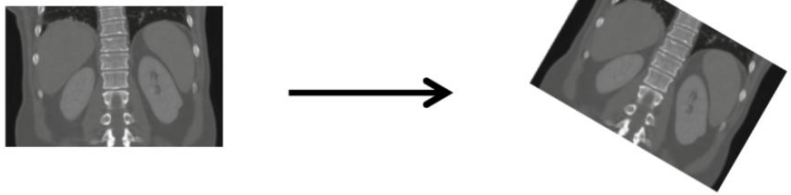


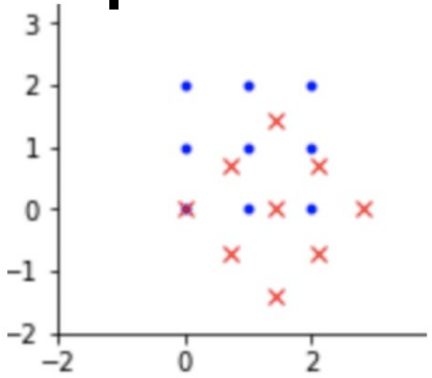
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



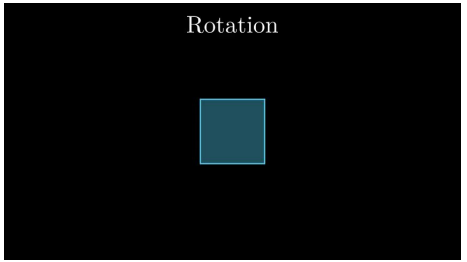
e.g., (1, 0, 83) → (cos(α), -sin(α), 83)

How can we represent rotation as a matrix operation?



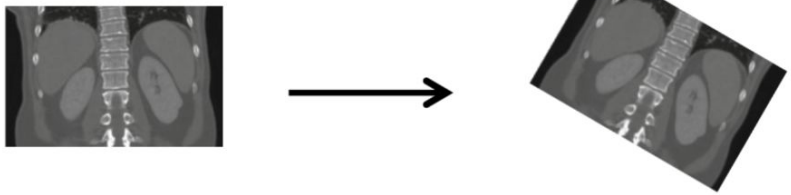


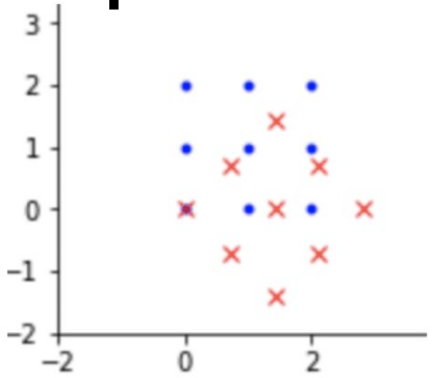
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



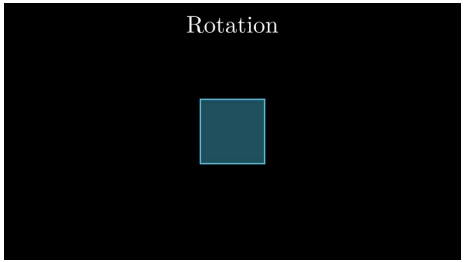
What's the inverse of R_θ?

How can we represent rotation as a matrix operation?





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



What's the inverse of R_θ? $\boldsymbol{I = R_\theta^T R_\theta}$

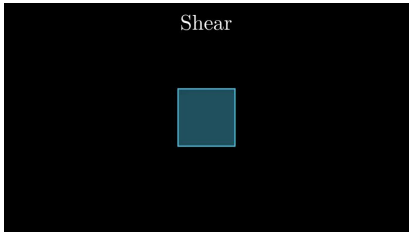
Other 2x2 matrix operations

Identity / No Transformation

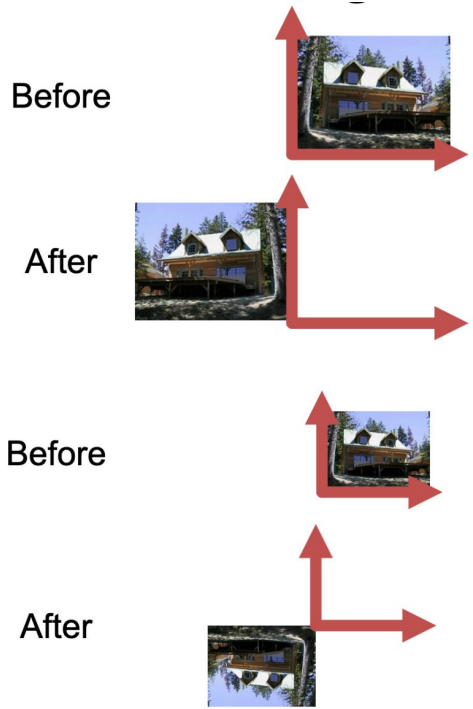
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

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



Other 2x2 matrix operations



2D Mirror About Y-Axis

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

2D Mirror About X,Y

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

What is invariant to 2x2 matrix operations?

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$

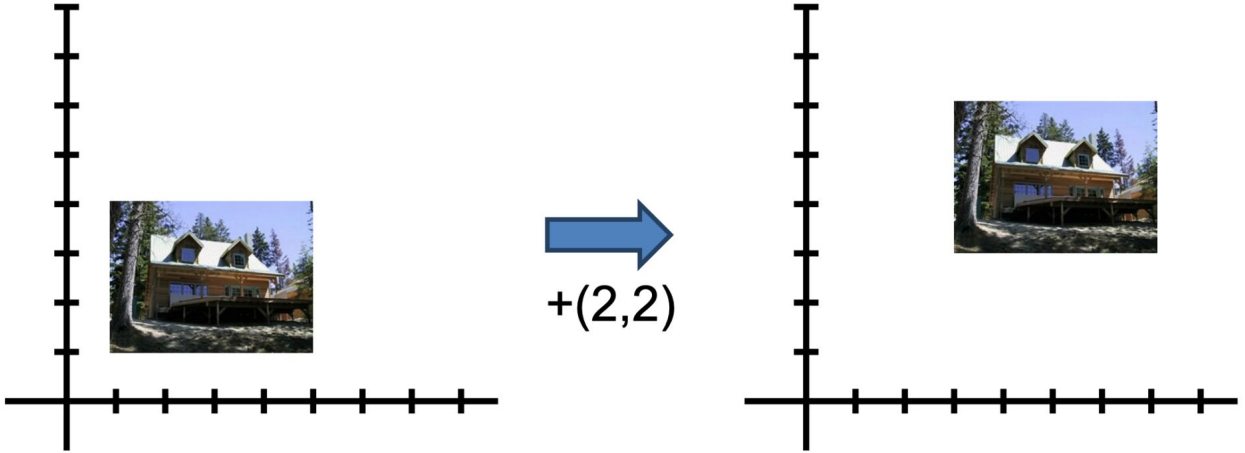
After multiplication by T (irrespective of T)

- Origin is origin: **0 = T0**
- Lines are lines
- Parallel lines are parallel

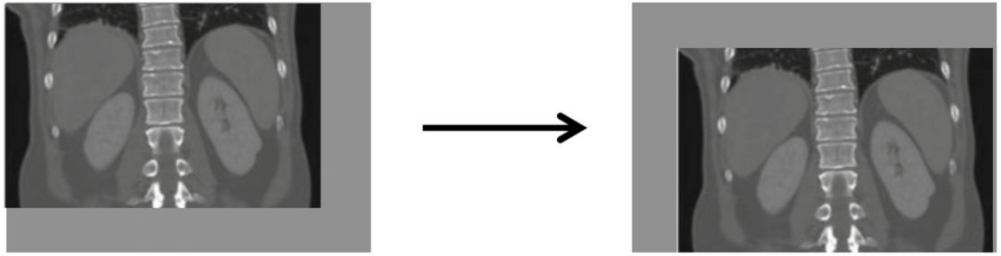
What about translation?

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

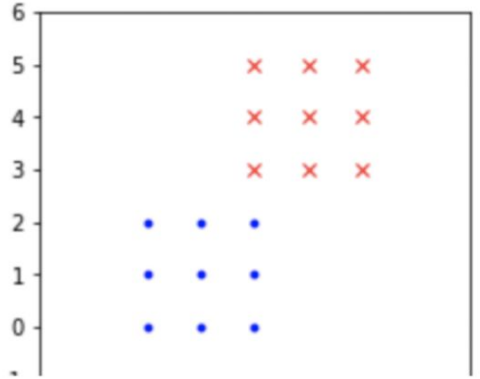
How do we make it linear?



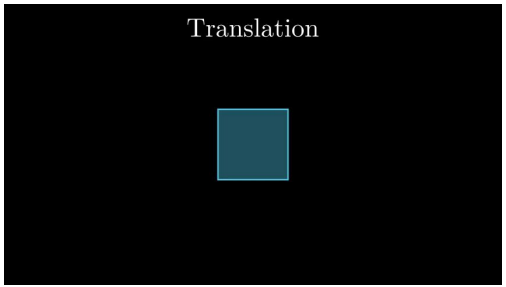
How can we represent translation a matrix operation?



Move to bottom-right by (2,3) pixel

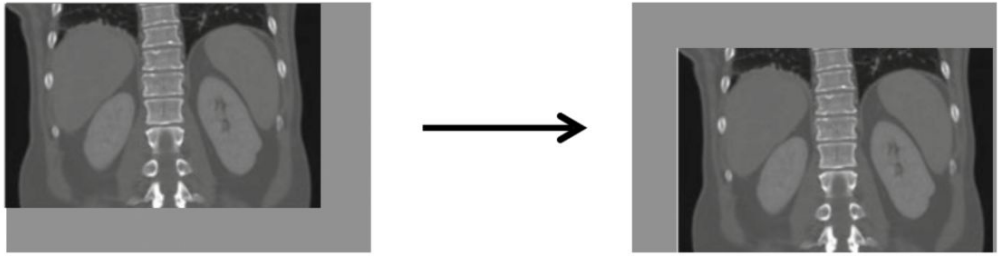


$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

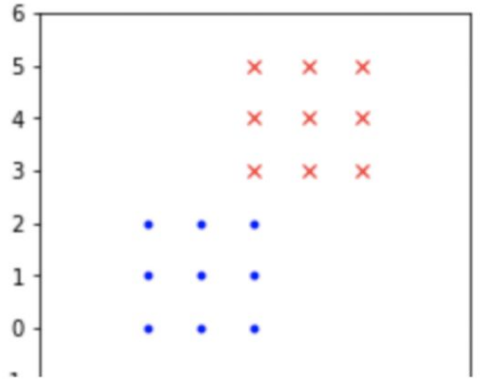


e.g., (0, 0, 83) → (a, b, 83)

How can we represent translation a matrix operation?



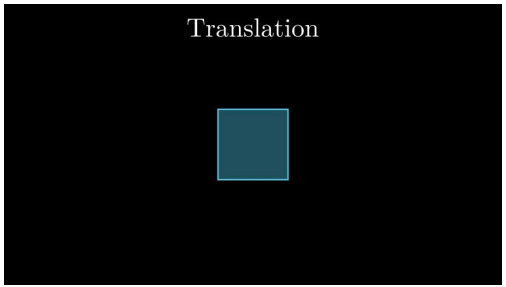
Move to bottom-right by (2,3) pixel



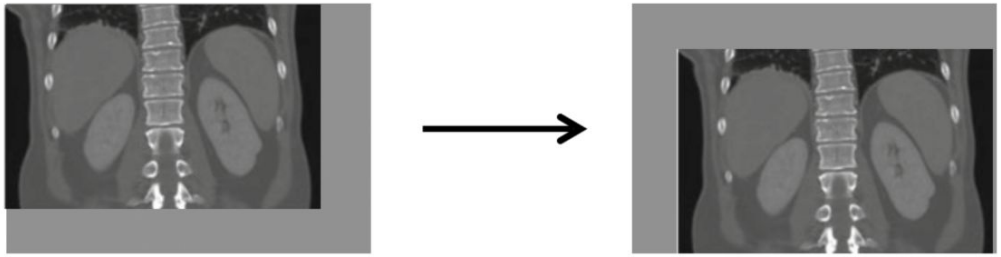
Note that this is not in the p' = Tp form

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

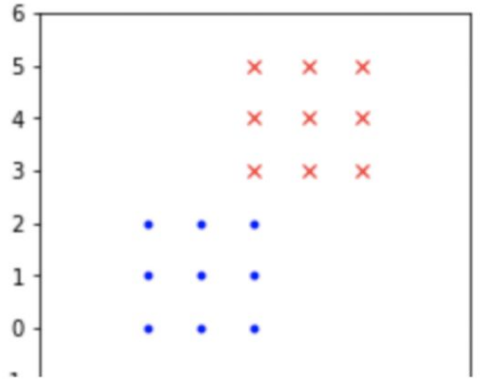
e.g., (0, 0, 83) → (a, b, 83)



How can we represent translation a matrix operation?



Move to bottom-right by (2,3) pixel



$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \rightarrow \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + a \\ y_1 + b \end{bmatrix} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

e.g., (0, 0, 83) → (a, b, 83)

Now T is not 2x2! How can we unify representations?

Use homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \longrightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

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

Translation:

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

Scaling:

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

Rotation:

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

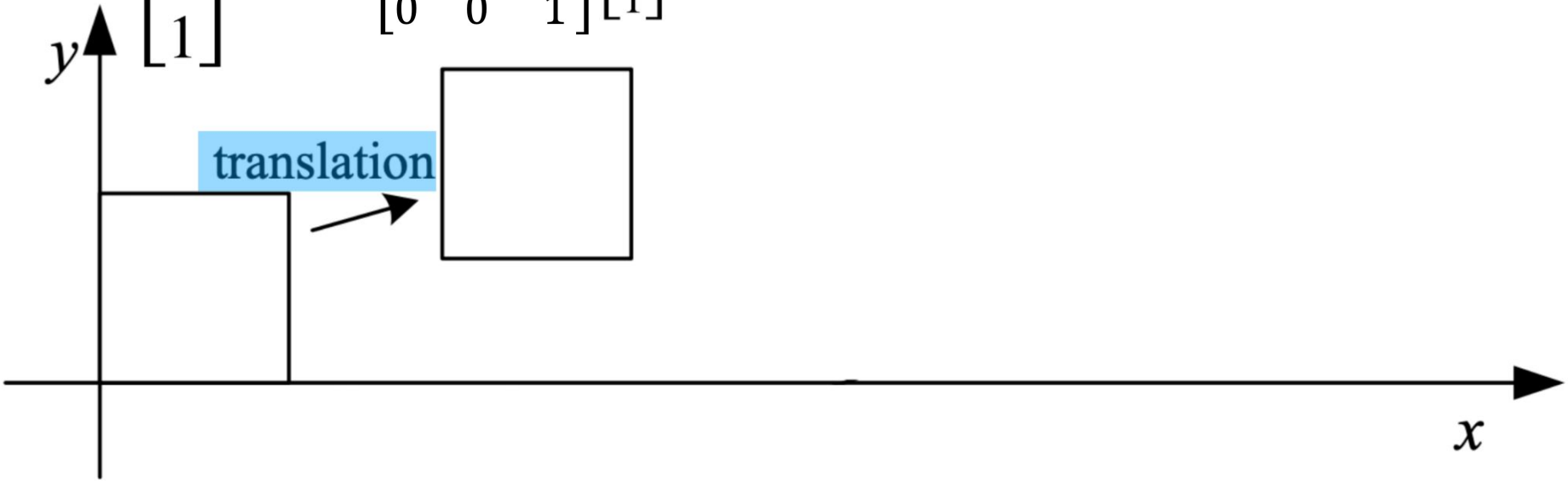
In general (without homogeneous coordinates)

$$\boldsymbol{x' = Ax + b}$$

Starting from translation

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

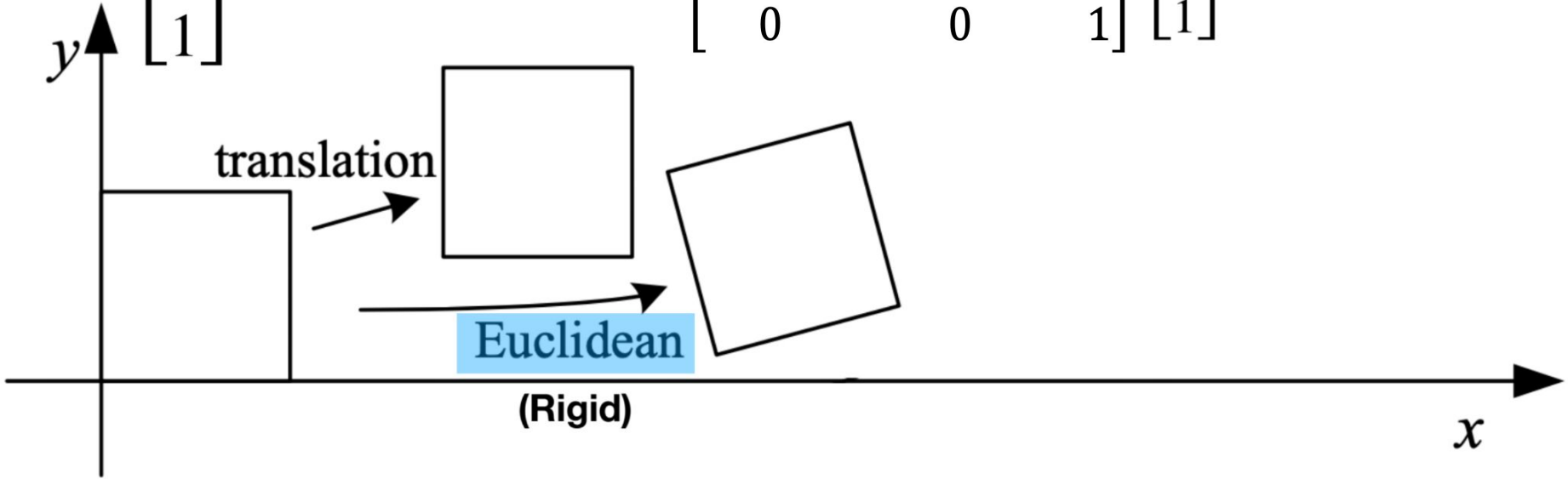
$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Rigid = (translation, rotation)

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

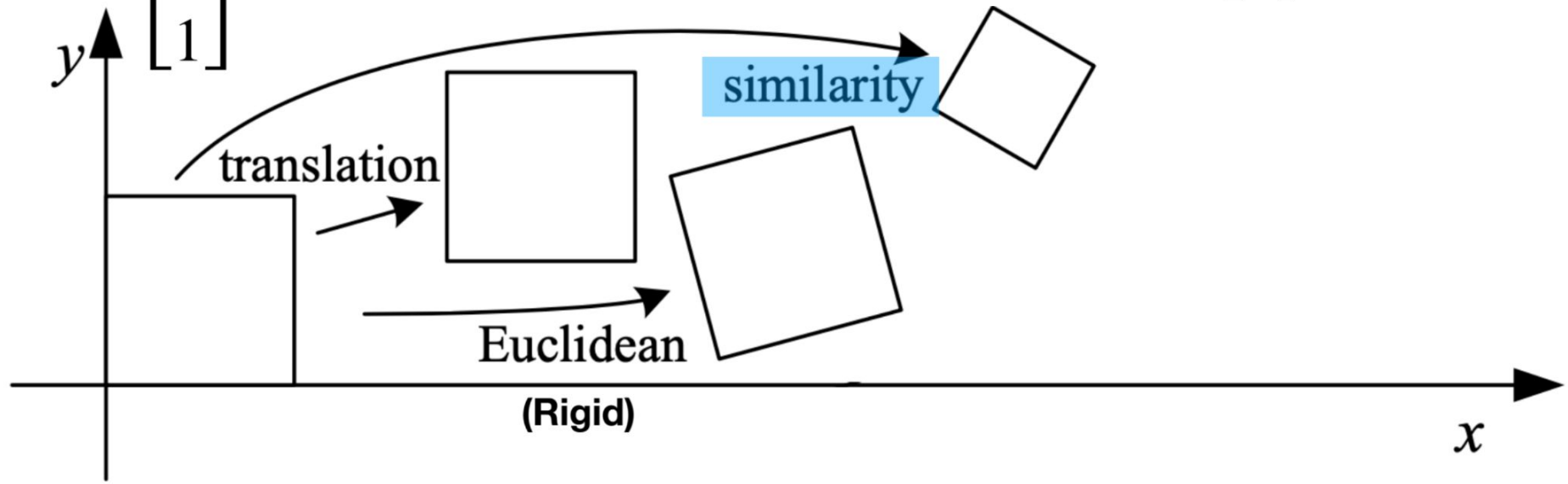
$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Similarity = (translation, rotation, **uniform** scaling)

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

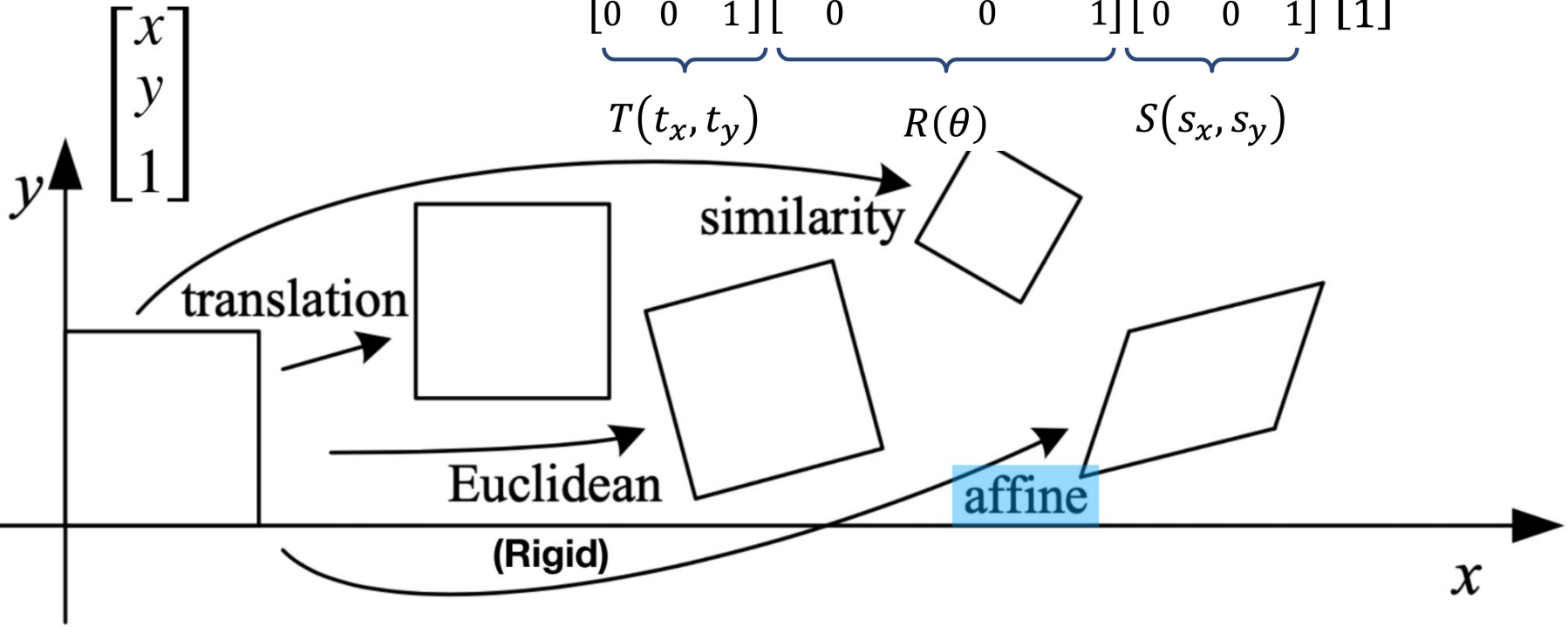
$$s \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Affine = non-uniform scaling + rotation + translation

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

$T(t_x, t_y)$ $R(\theta)$ $S(s_x, s_y)$



Matrix Composition

We can combine transformations via matrix multiplication.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{T(t_x, t_y)} \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R(\theta)} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S(s_x, s_y)} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Does order matter?

What's Preserved With Affine

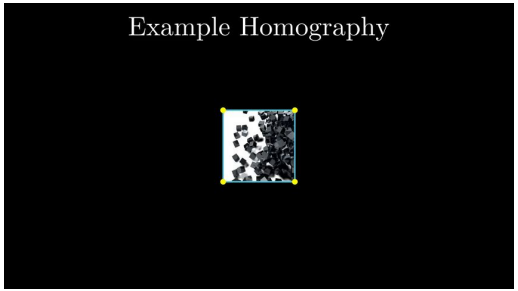
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \boldsymbol{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

After multiplication by T (irrespective of T)

- ~~**Origin is origin: 0 = T0**~~
- Lines are lines
- Parallel lines are parallel

Perspective Transformations

Set bottom row to not [0,0,1]
Called a perspective/projective transformation or a ***homography***



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

What's Preserved With Perspective

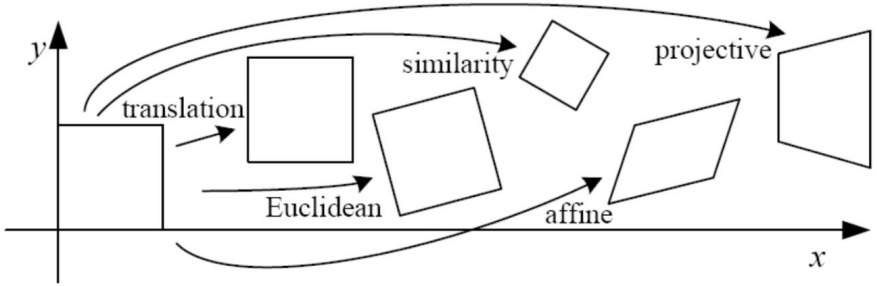
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \boldsymbol{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

After multiplication by T (irrespective of T)

- ~~**Origin is origin: 0 = T0**~~
- Lines are lines
- ~~Parallel lines are parallel~~
- ~~Ratios between distances~~

Transformation Families

In general: transformations are a nested set of groups

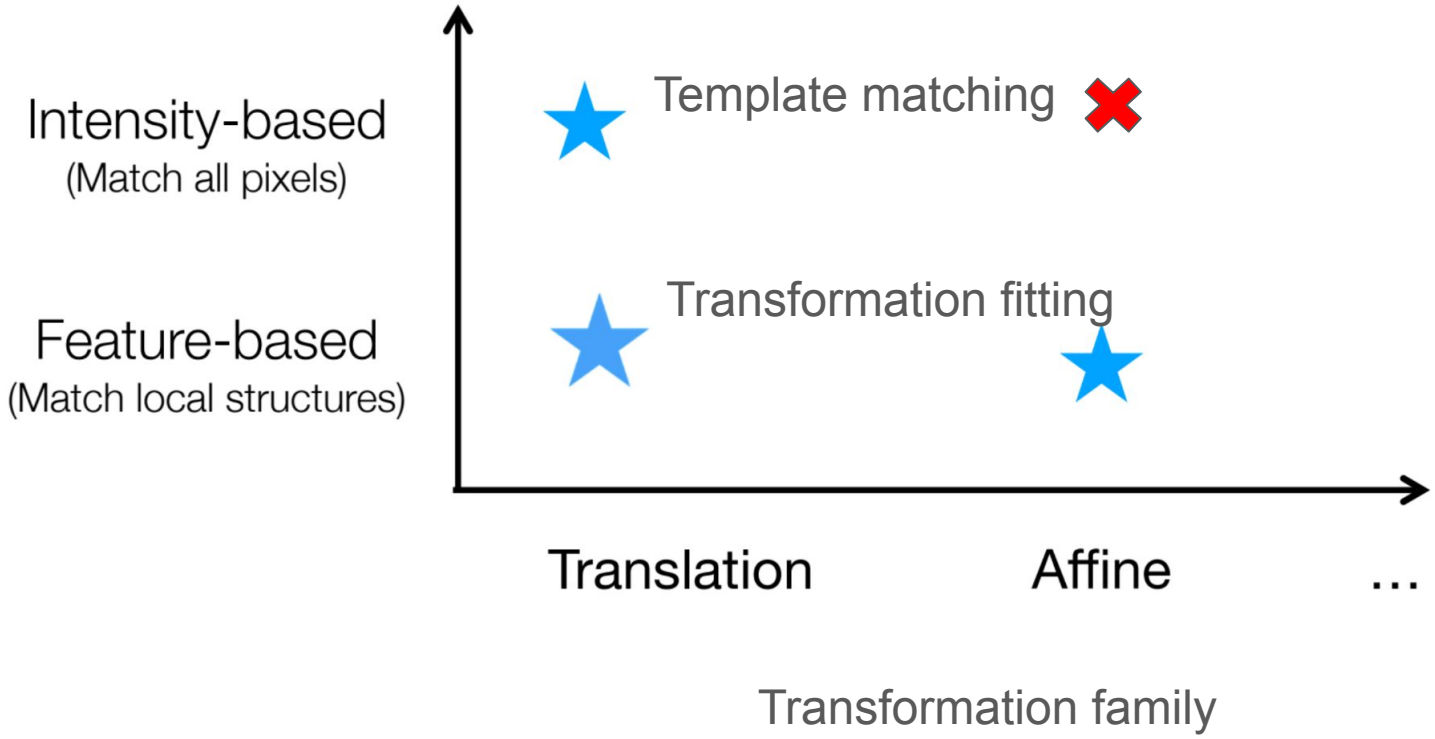


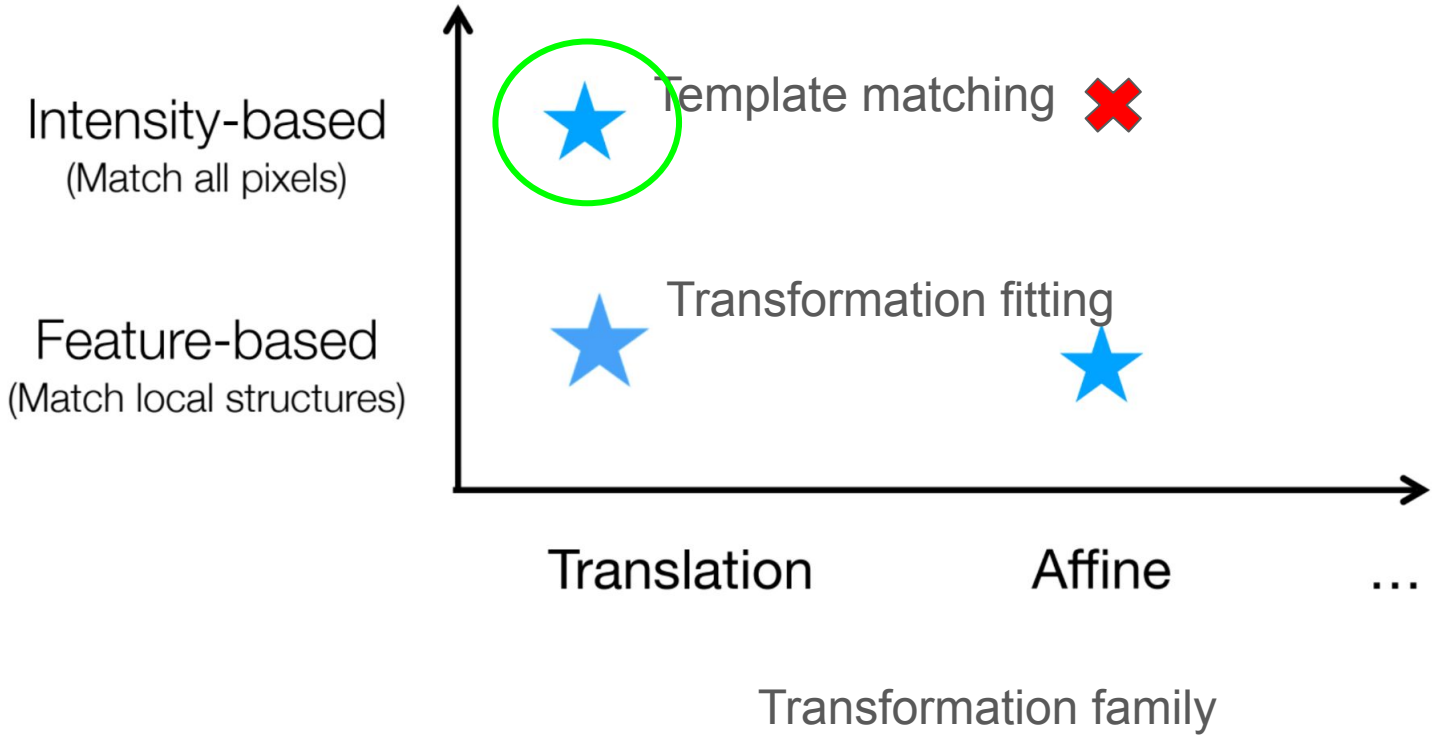
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	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	

What Can Homographies Do?

Homography example 1: any two views of a *planar* surface

Fitting transformations


Intensity-based
(Match all pixels)
Feature-based
(Match local structures)
Template matching
Transformation fitting
Translation
Affine
…
Transformation family


Intensity-based
(Match all pixels)
Feature-based
(Match local structures)
Template matching
Transformation fitting
Translation
Affine
…
Transformation family

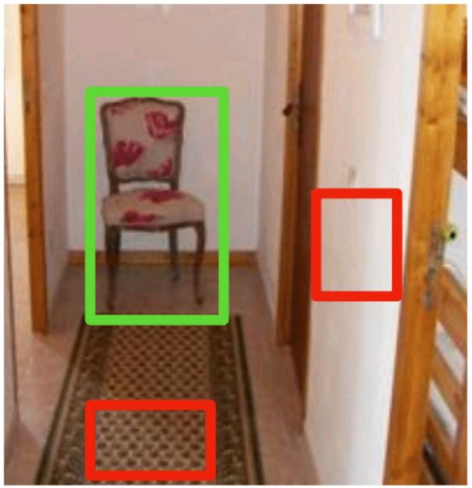
Method 1: Intensity-based for translation

Im1

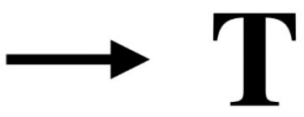
Im2

→ **T**

Idea I: Find a “distinctive” patch (template)



Bad: little texture to match



Bad: many matches

Im1

Im2 patch

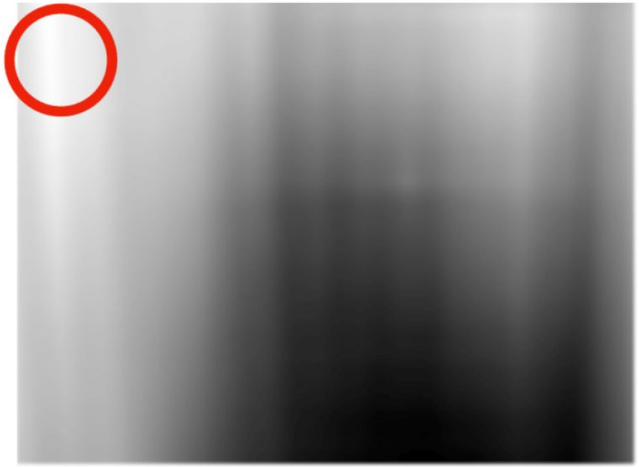
Idea II: Filtering -> Cross-correlation

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y'))$$

For each im1 patch, dot-product

Im1

Im2 patch



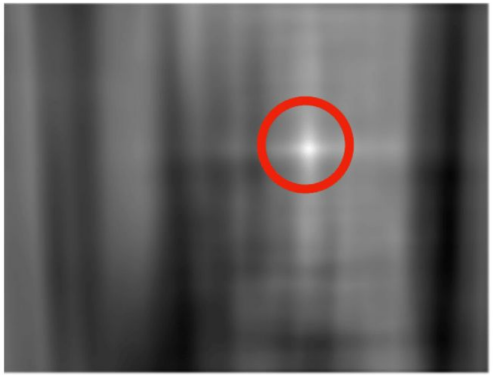
Idea III: Filtering -> Normalized cross-correlation

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

For each im1 patch, **normalized** dot-product

Im1

Im2 patch



Template matching -> Image alignment

Im1

Im2 patch

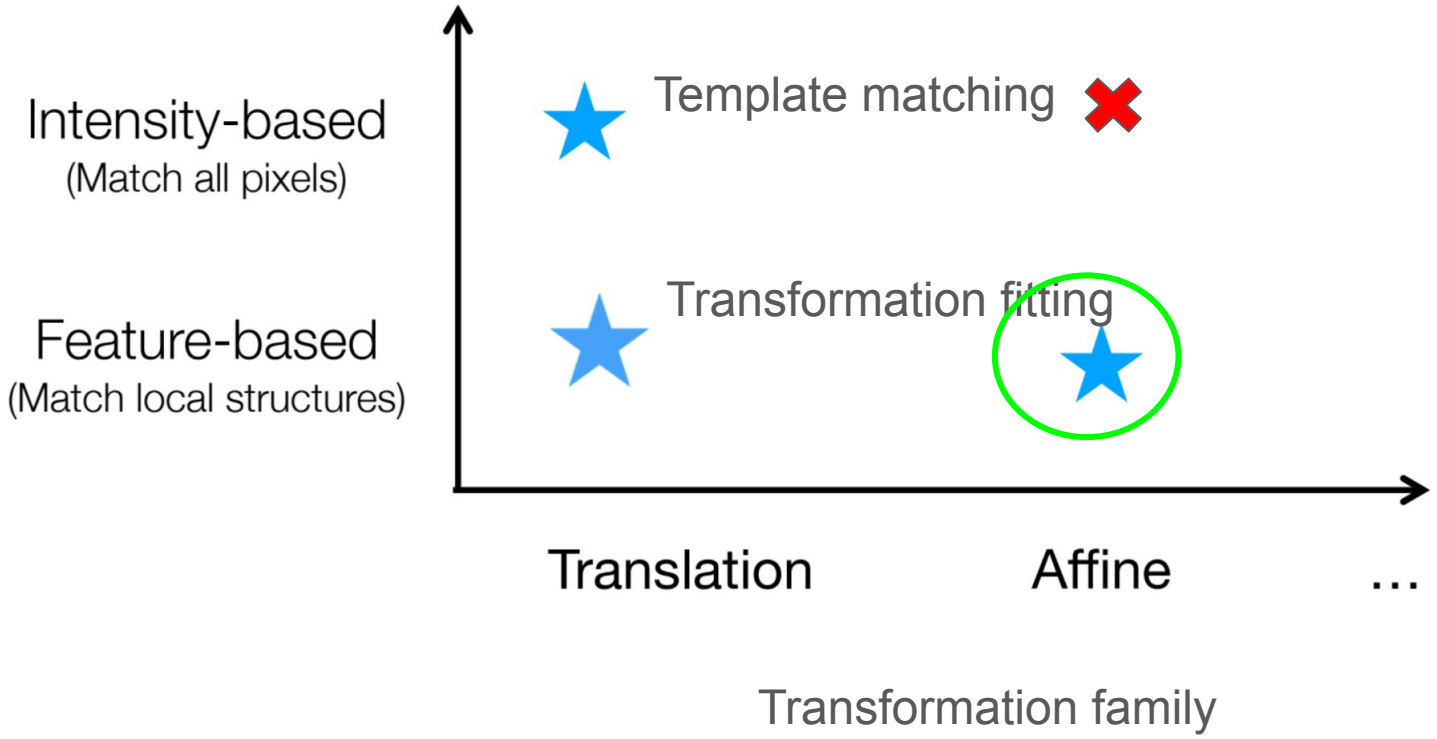
Im1: matched position (x_1, y_1)

Im2: cropped position (x_2, y_2)

First move to (0,0), then (x_1, y_1)

Solved translation

$$\mathbf{T}: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x - x_2 + x_1 \\ y - y_2 + y_1 \end{bmatrix}$$


Intensity-based
(Match all pixels)
Feature-based
(Match local structures)
Template matching
Transformation fitting
Translation
Affine
…
Transformation family

Method 2: feature-based for affine transformation

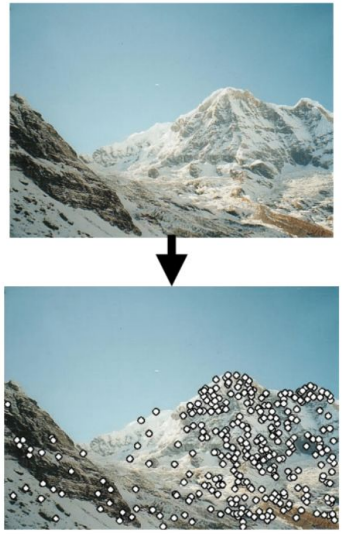
Im1

Im2

Image stitching (Panaroma)

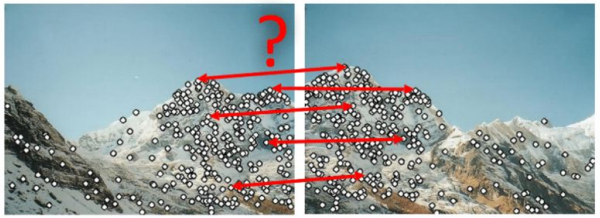
Pipeline

1. Find Key points



(discriminative regions)

2. Find matches (can be noisy)

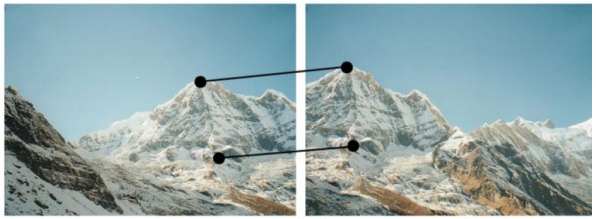


3. Estimate transformation

RANSAC

(find the best among many random trials)

3a. Pick inlier matches



3b. Estimate transformation

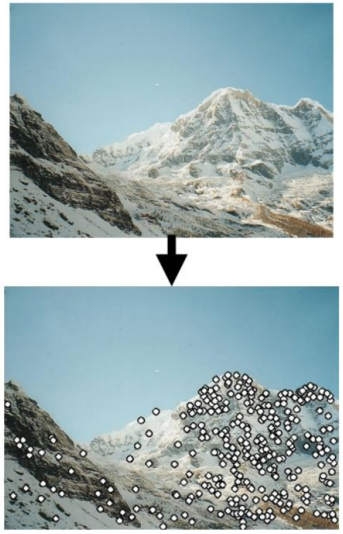
$$\{(x, y), (x', y')\}_K \rightarrow T$$

4. Warp!

Pipeline

Lecture 4

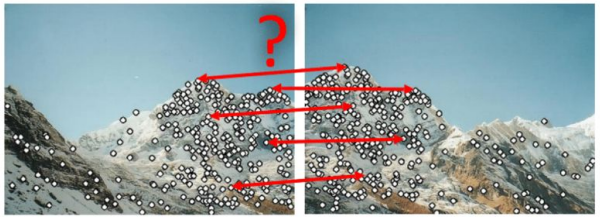
1. Find Key points



(discriminative regions)

Lecture 4

2. Find matches (can be noisy)



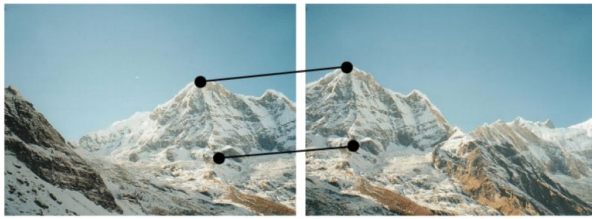
We are finally ready for this…

3. Estimate transformation

Lecture 5 **RANSAC**

(find the best among many random trials)

3a. Pick inlier matches



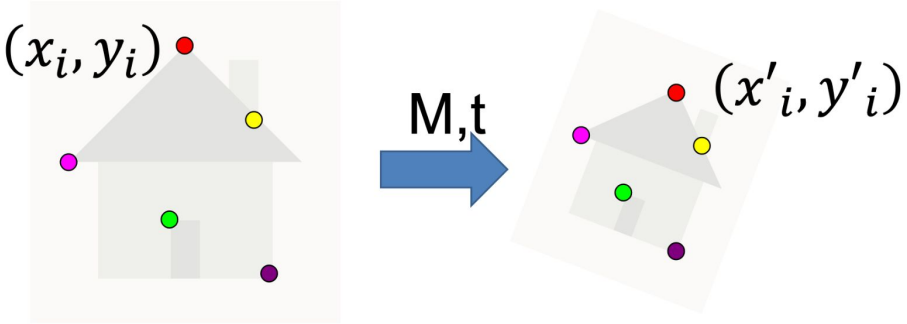
3b. Estimate transformation

$$\{(x, y), (x', y')\}_K \rightarrow T$$

4. Warp!

Fitting Transformations

Setup: Have a set of **inlier** corresponding keypoints after RANSAC



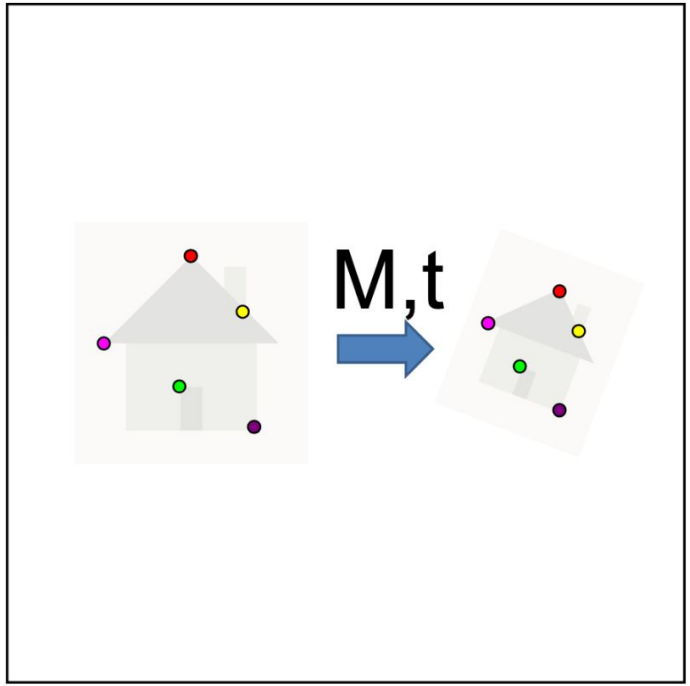
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \boldsymbol{t}$$

Affine Transformation: M,t

Data: (x_i, y_i, x'_i, y'_i) for $i=1,\ldots,k$

Model:
$[x'_i, y'_i] = \mathbf{M}[x_i, y_i] + \mathbf{t}$

Objective function:
$||[x'_i, y'_i] - (\mathbf{M}[x_i, y_i] + \mathbf{t})||^2$



Fitting affine transformations

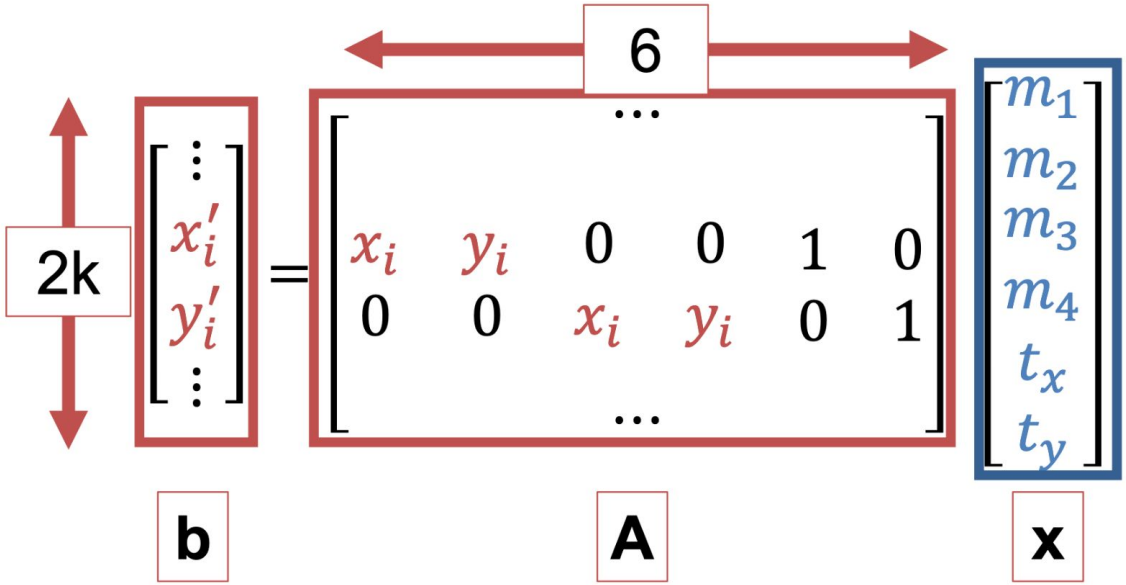
Given correspondences: $[x'_i, y'_i] \leftrightarrow [x_i, y_i]$

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Set up two equations per point

$$\begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix}$$

Fitting affine transformations



Want: **b = Ax** (**x** contains all parameters)

Overconstrained, so solve $\arg\min||\boldsymbol{Ax} - \boldsymbol{b}||$

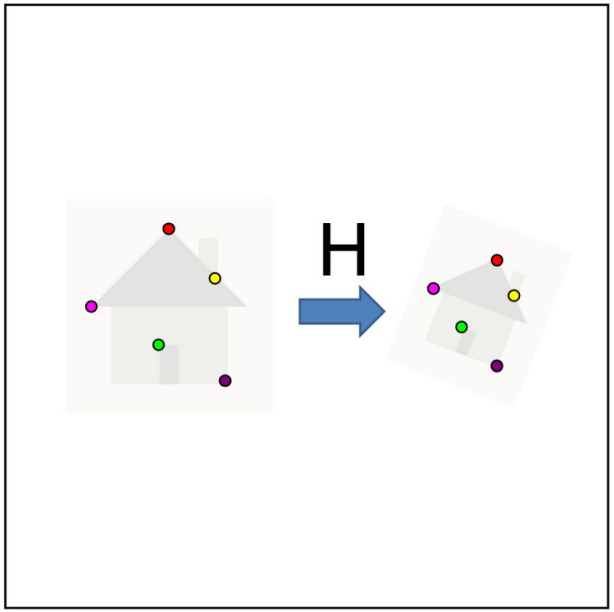
How?

Fitting a homography

Data: (x_i, y_i, x'_i, y'_i) for $i=1,\ldots,k$

Model: $[x'_i, y'_i, 1] \equiv \mathbf{H}[x_i, y_i, 1]$

Objective function: It's complicated



Fitting a homography - gets messy but its all linear algebra

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

Fitting a homography - gets messy but its all linear algebra

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

$$\begin{aligned} x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Fitting a homography - gets messy but its all linear algebra

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{A} \qquad \underset{9}{h} \qquad \underset{2n}{0}$$

Defines a least squares problem: $\text{minimize } \|\mathbf{Ah} - 0\|^2$

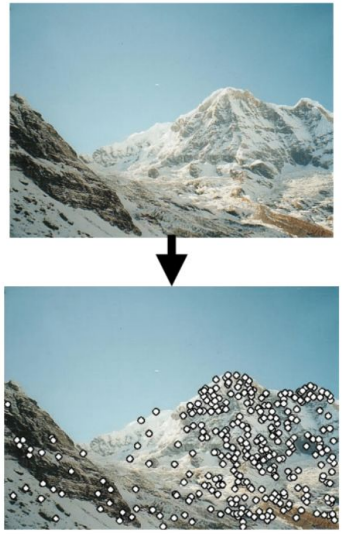
- Since h is only defined up to scale, solve for unit vector $\hat{h}$
- Solution: $\hat{h}$ = eigenvector of A^TA with smallest eigenvalue
- Works with 4 or more points

Last step: Warping

Pipeline

Lecture 4

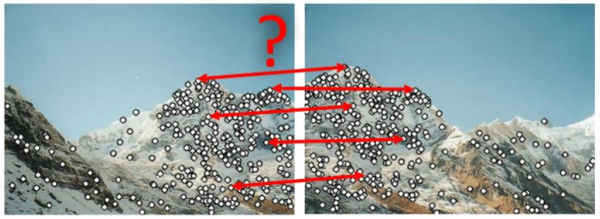
1. Find Key points



(discriminative regions)

Lecture 4

2. Find matches (can be noisy)



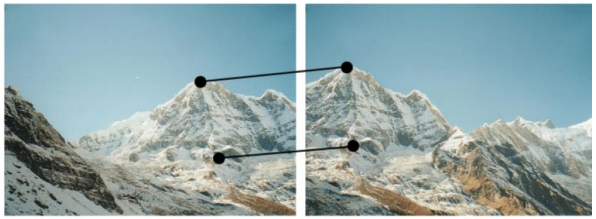
DONE

3. Estimate transformation

Lecture 5 **RANSAC**

(find the best among many random trials)

3a. Pick inlier matches

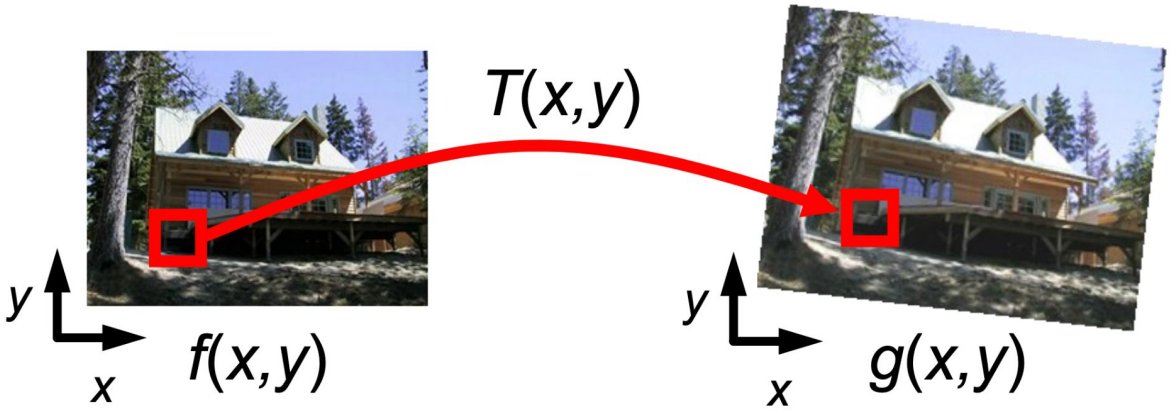


3b. Estimate transformation

$$\{(x, y), (x', y')\}_K \rightarrow T$$

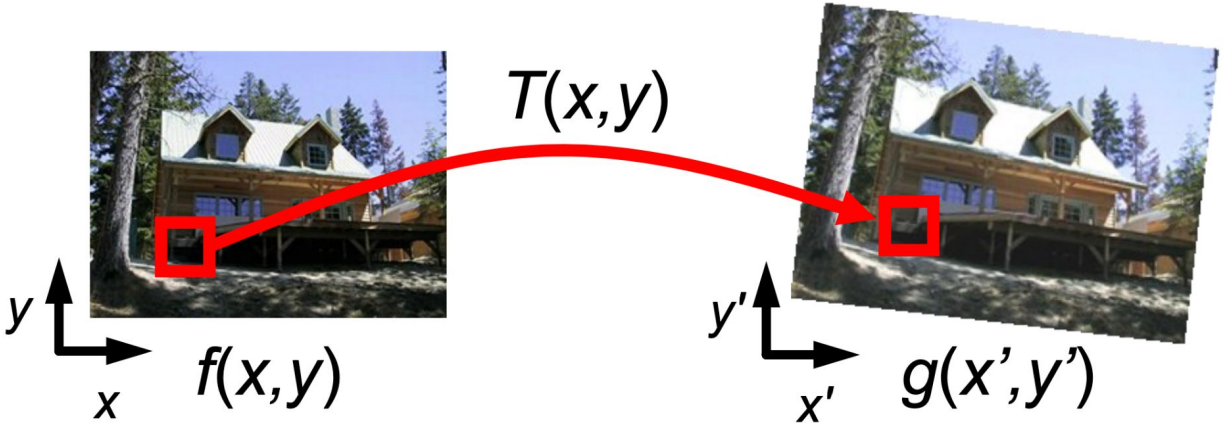
4. Warp!

Image Warping



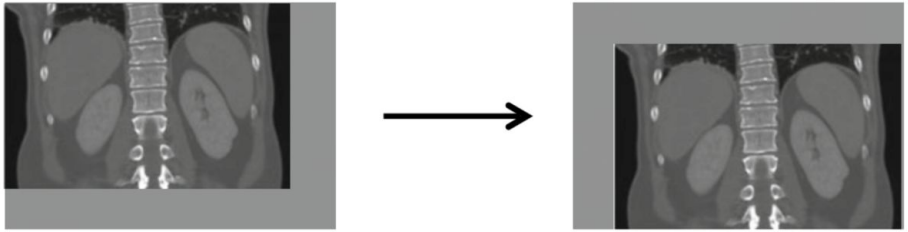
Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?

Forward Warping



Send the value at each pixel (x,y) to
the new pixel (x',y') = T([x,y])

Problem 1: What about non-integer translation



Move to bottom-right by **(0.8,0.2)** pixel

83	100	240	128
22	239	159	128
143	242	5	128
128	128	128	128

→ **???**

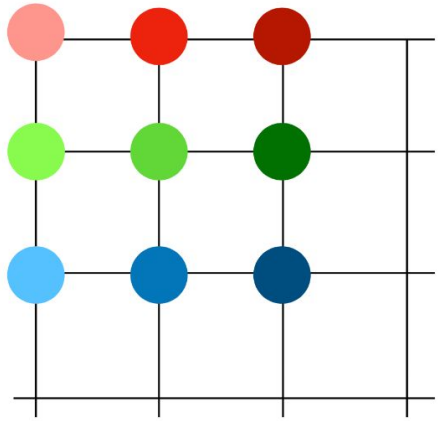
Matrix representation

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \rightarrow \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + 0.8 \\ y_1 + 0.2 \end{bmatrix}$$

e.g., (0, 0, 83) → (0.8, 0.2, 83)

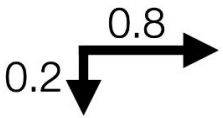
Point cloud representation

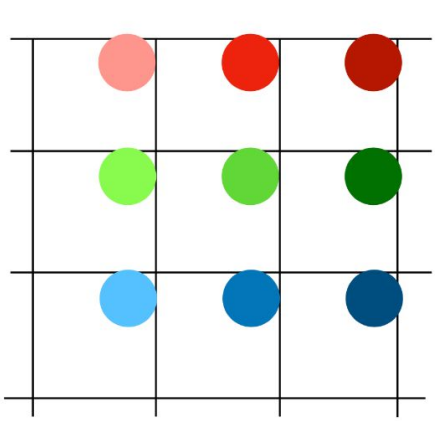
Solution 1: Forward Mapping



Input image

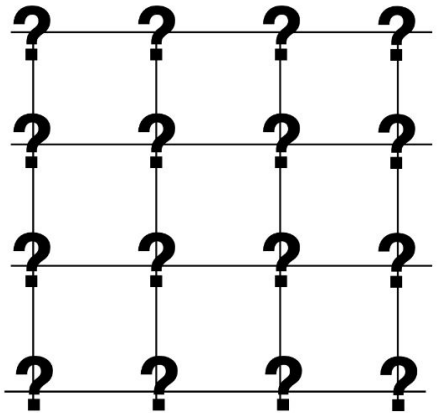
(x_i, y_i)





Translated image

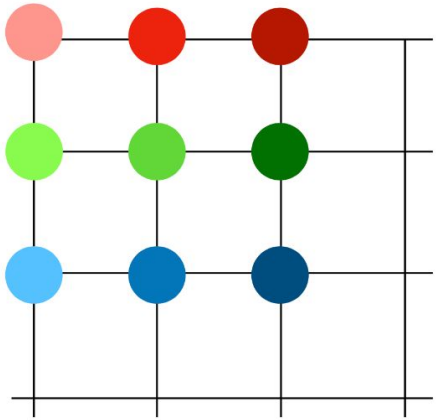
Interpolation on integer positions



Matrix rendering

Option 1: Nearest neighbor

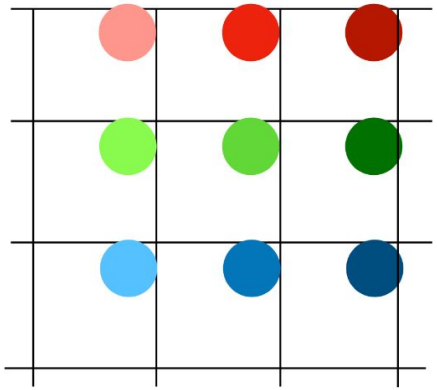
(0,0) (1,0) (2,0)



Input image

(x_i, y_i)

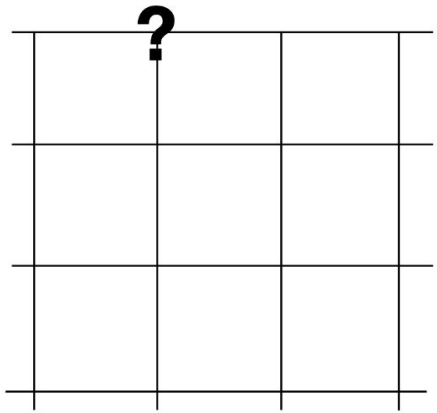
(0.8,0.2) (1.8,0.2) (2.8,0.2)



Translated image

0.8

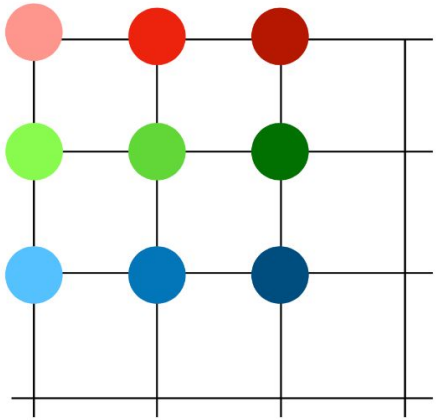
0.2



Matrix rendering
(Ignore the border for now)

Option 1: Nearest neighbor

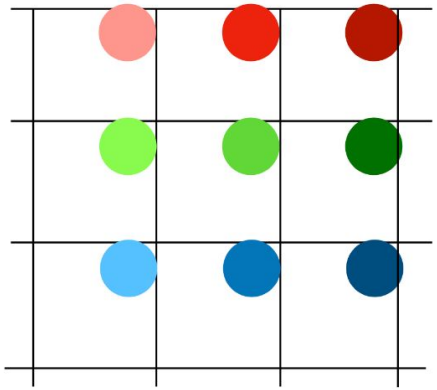
(0,0) (1,0) (2,0)



Input image

(x_i, y_i)

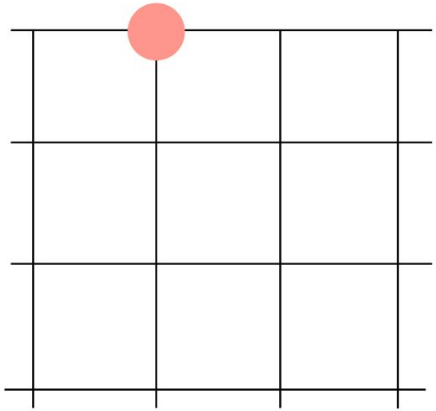
(0.8,0.2) (1.8,0.2) (2.8,0.2)



Translated image

0.8

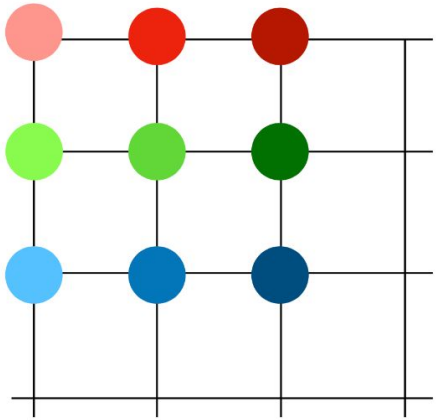
0.2



Matrix rendering
(Ignore the border for now)

Option 1: Nearest neighbor

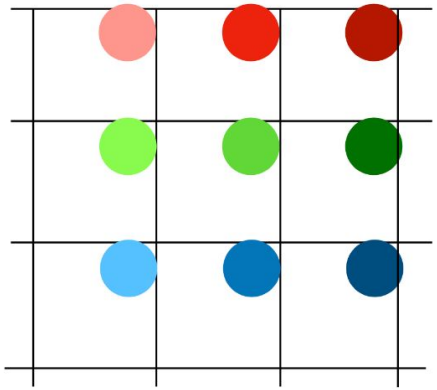
(0,0) (1,0) (2,0)



Input image

(x_i, y_i)

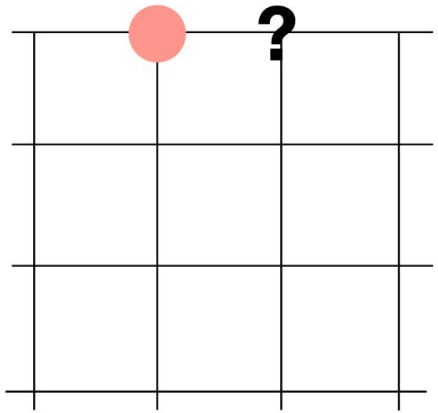
(0.8,0.2) (1.8,0.2) (2.8,0.2)



Translated image

0.8

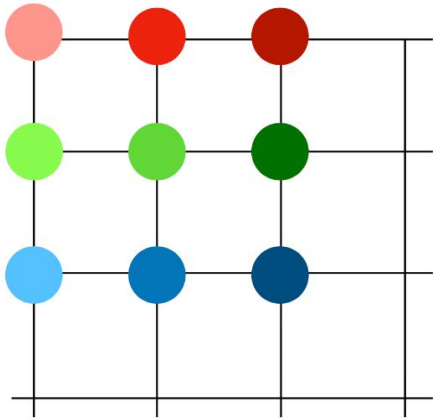
0.2



Matrix rendering
(Ignore the border for now)

Option 1: Nearest neighbor

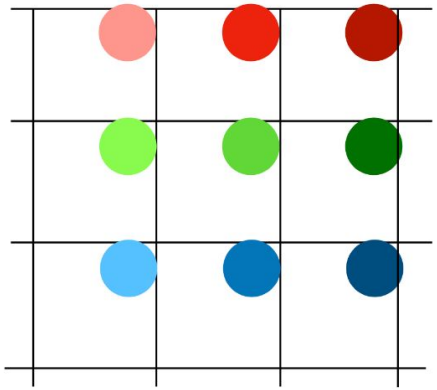
(0,0) (1,0) (2,0)



Input image

(x_i, y_i)

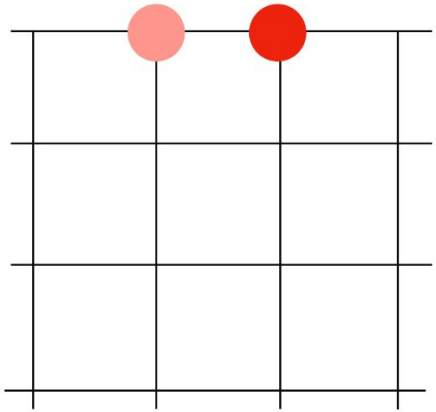
(0.8,0.2) (1.8,0.2) (2.8,0.2)



Translated image

0.8

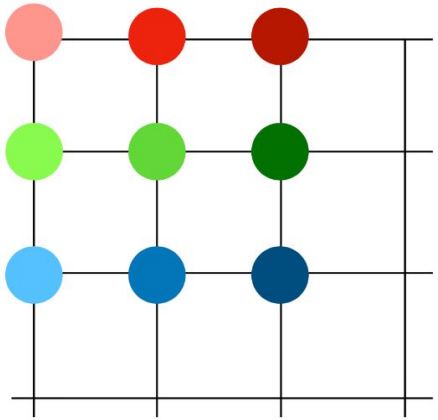
0.2



Matrix rendering
(Ignore the border for now)

Option 1: Nearest neighbor

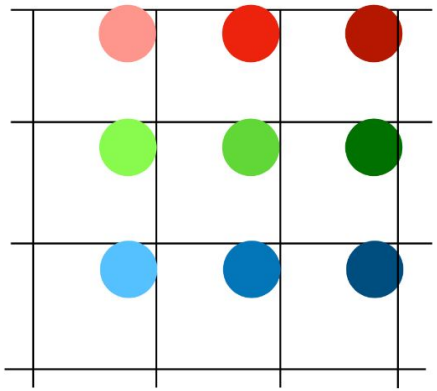
(0,0) (1,0) (2,0)



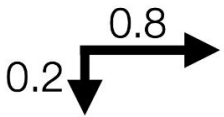
Input image

(x_i, y_i)

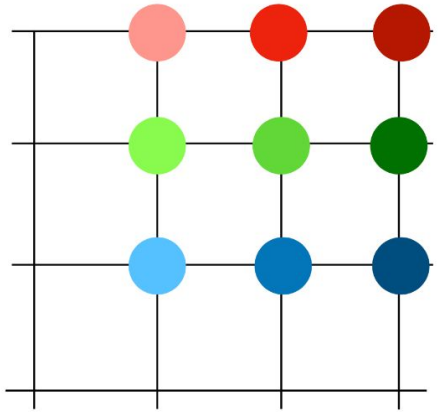
(0.8,0.2) (1.8,0.2) (2.8,0.2)



Translated image

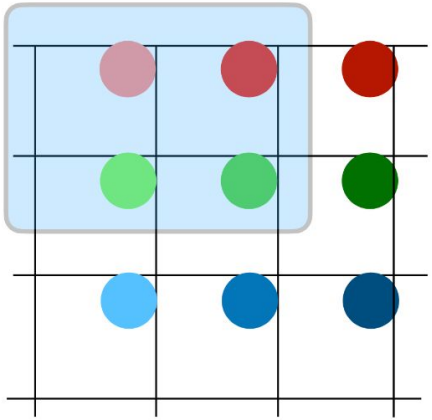


Problem: same as translating (1,0)

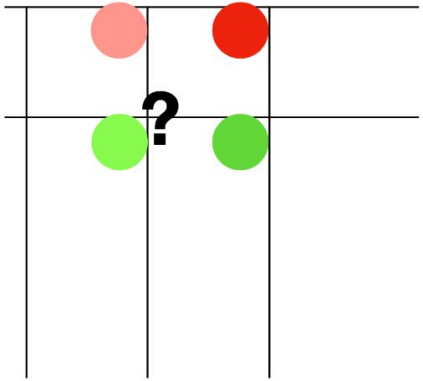


Matrix rendering
(Ignore the border for now)

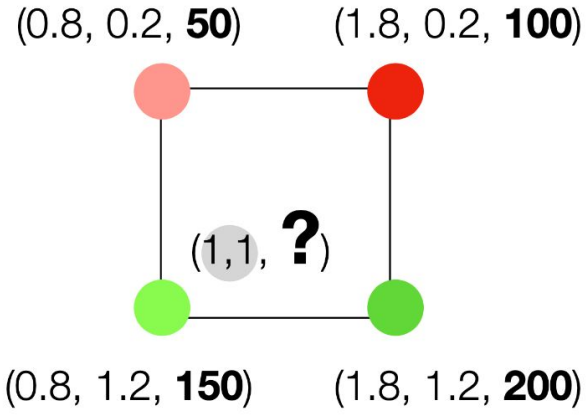
Option 2: Bilinear



Translated image

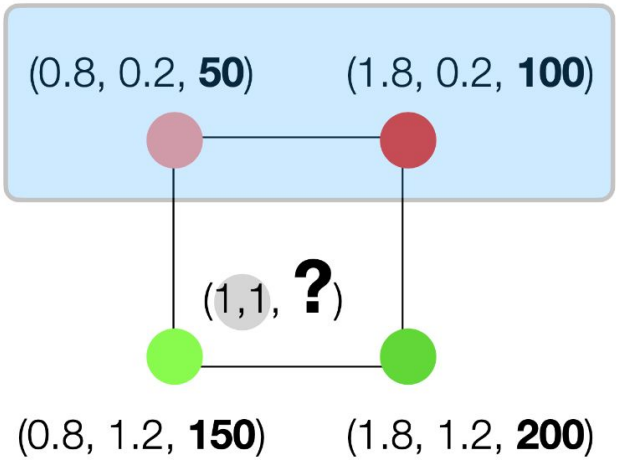


Zoom to one pixel

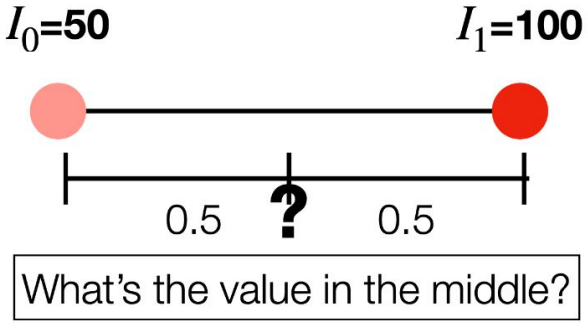


Goal: interpolate pixel value at (1,1)

Look at 1D case: Middle point

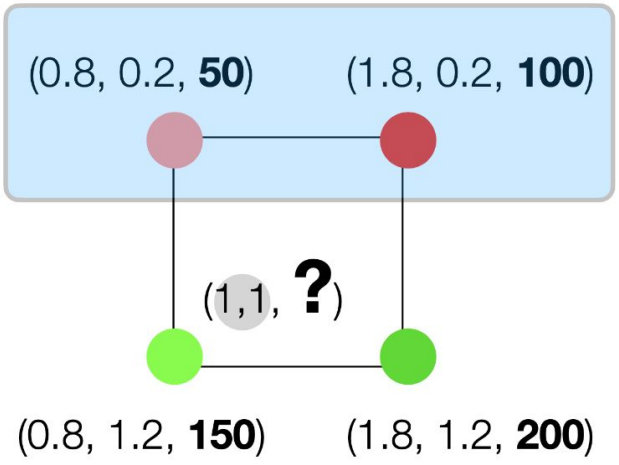


Goal: interpolate pixel value at (1,1)

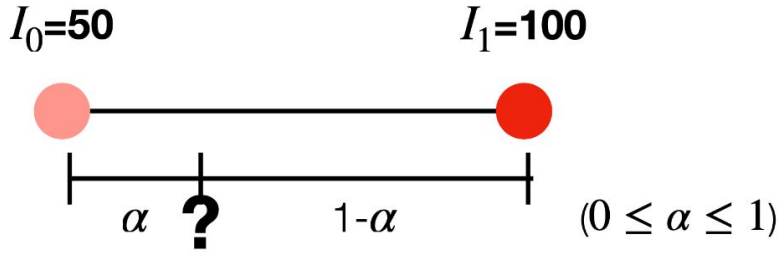


(1.3, 0.2, **75**): take the average

Look at 1D case: General point



Goal: interpolate pixel value at (1,1)



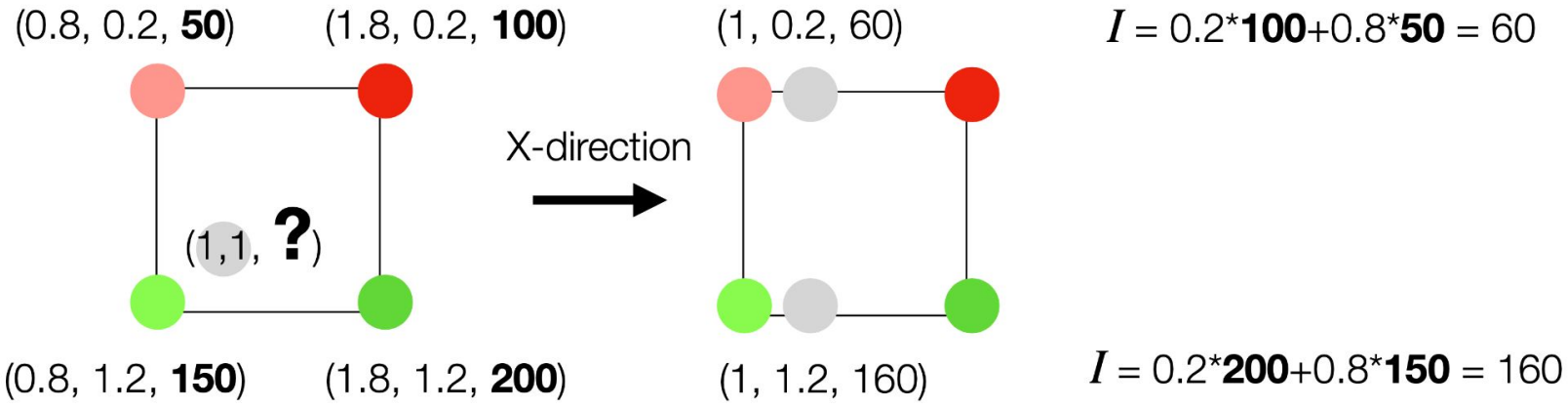
What's the value in any position?

$$I = I_1 * \alpha + I_0 * (1-\alpha)$$

- α=0: $I = I_0 = 50$
- α=1: $I = I_1 = 150$

Convert 2D to 1D: **Bi**linear interpolation

$$I = I_1 * \alpha + I_0 * (1-\alpha)$$



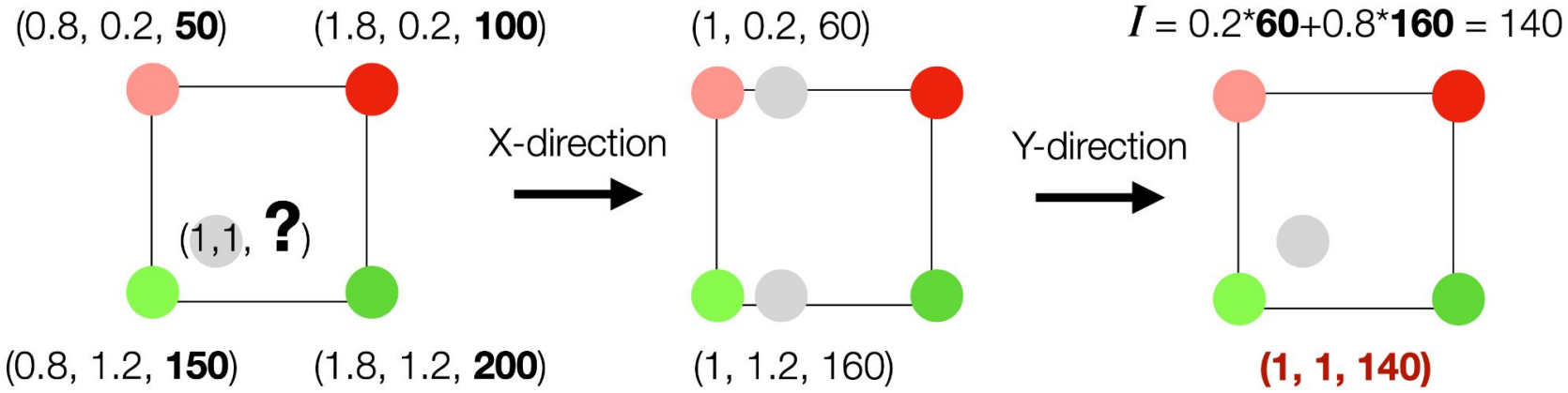
I = 0.2*__100__+0.8*__50__ = 60

I = 0.2*__200__+0.8*__150__ = 160

Goal: interpolate pixel value at (1,1)

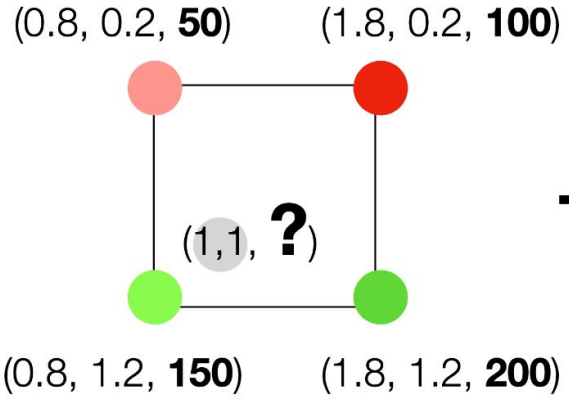
Convert 2D to 1D: **Bi**linear interpolation

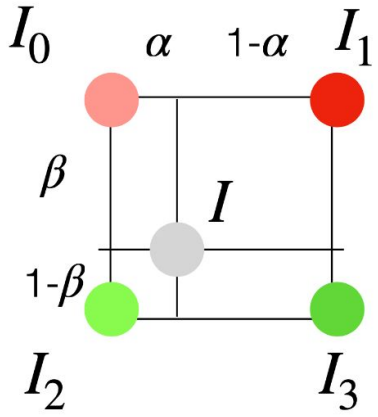
$$I = I_1 * \alpha + I_0 * (1-\alpha)$$



Goal: interpolate pixel value at (1,1)

Convert 2D to 1D: **Bi**linear interpolation





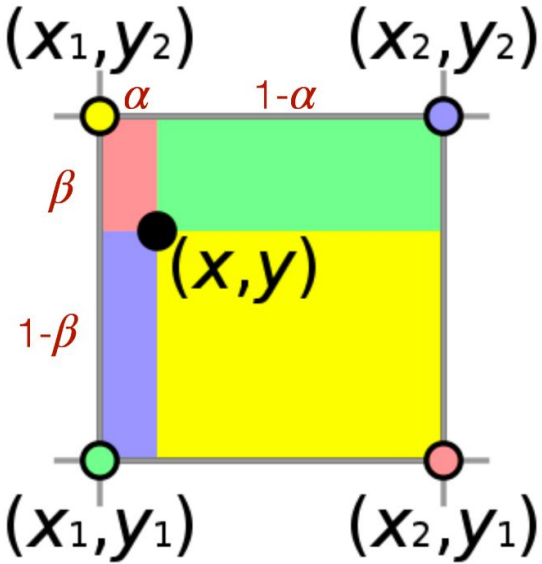
$$I = (1-\alpha)(1-\beta)I_0 + \alpha(1-\beta)I_1 + (1-\alpha)\beta I_2 + \alpha\beta I_3$$

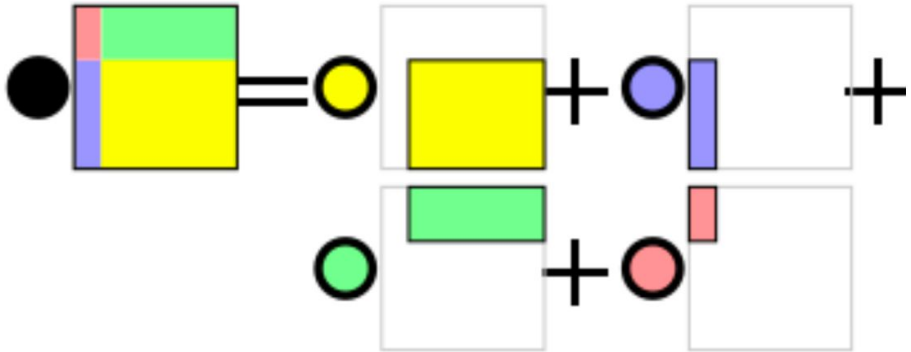
Linear combination of four corner values

Goal: interpolate pixel value at (1,1)

Convert 2D to 1D: **Bi**linear interpolation

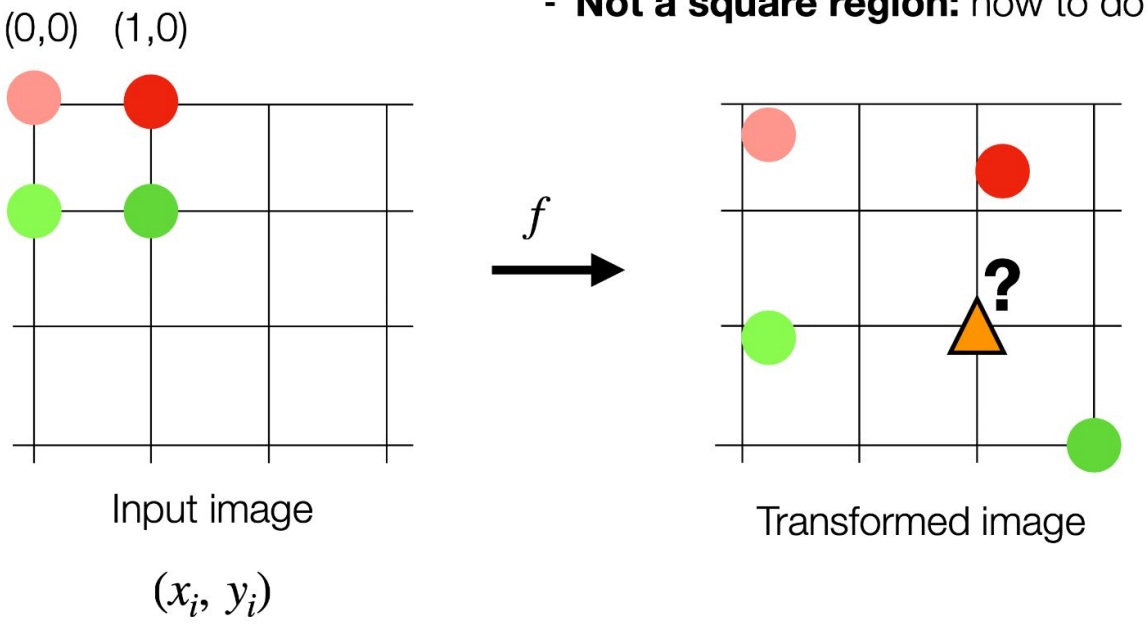
$$I = (1-\alpha)(1-\beta)I_0 + \alpha(1-\beta)I_1 + (1-\alpha)\beta I_2 + \alpha\beta I_3$$



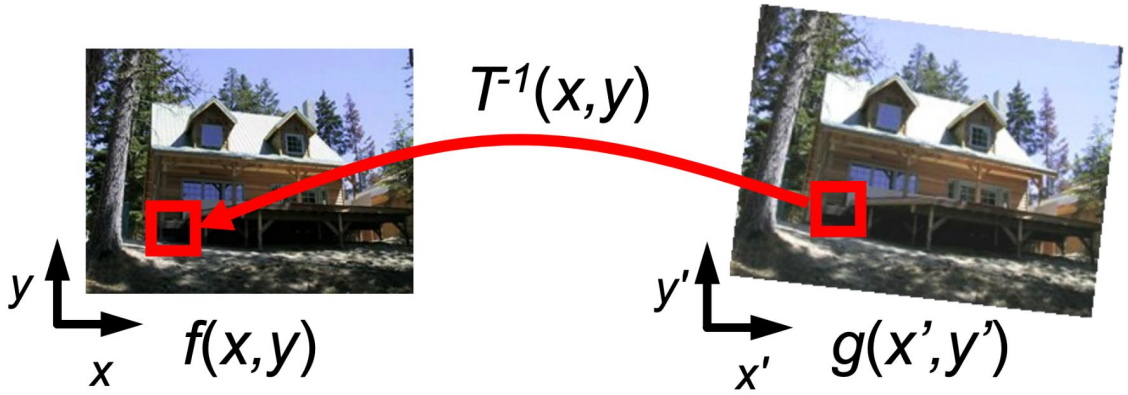


Problem: What about arbitrary transformation

- **Not a square region:** how to do bilinear?



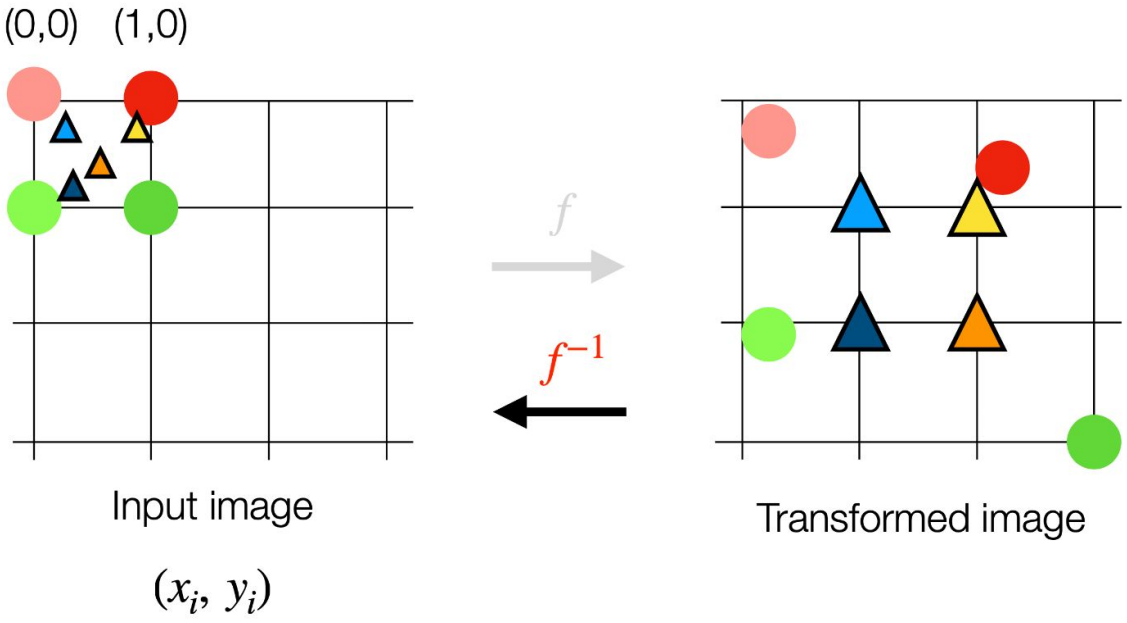
Inverse Warping



Find out where each pixel g(x',y') should get its value from, and steal it.
Note: requires ability to invert T

Inverse Warping

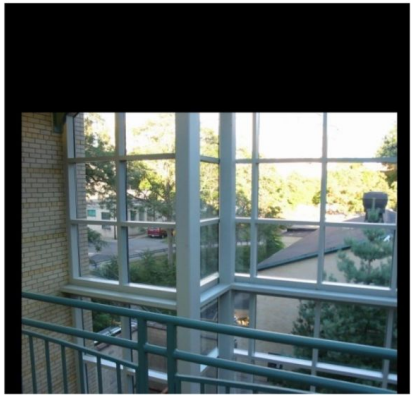
- **Back to a square region:** easy to do bilinear



Mosaicing

Warped Input 1 I_1

Warped Input 2 I_2

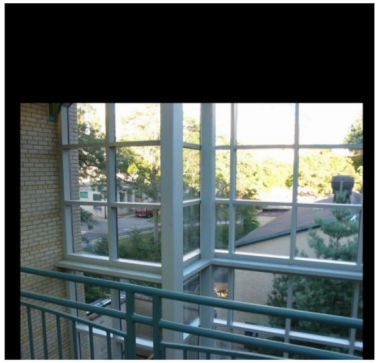


Can warp an image. Pixels that don't have a corresponding pixel in the image are set to a chosen value (often 0)

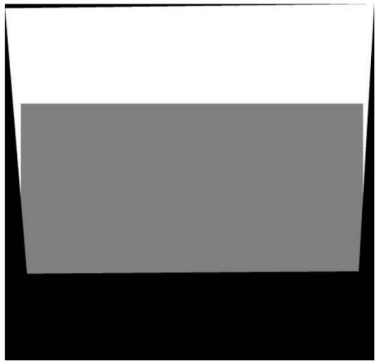
Mosaicing

Warped Input 1 I_1

Warped Input 2 I_2



α

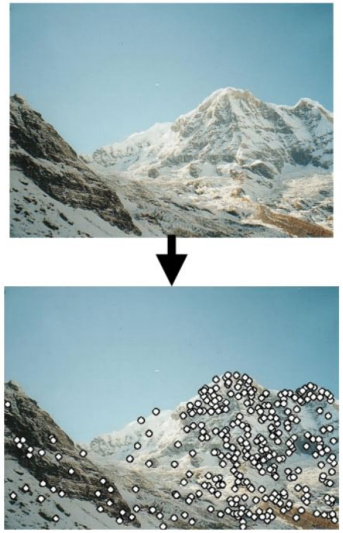


$\alpha I_1 + (1-\alpha) I_2$

Finale:

Lecture 4

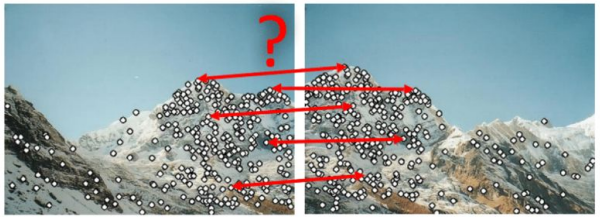
1. Find Key points



(discriminative regions)

Lecture 4

2. Find matches (can be noisy)

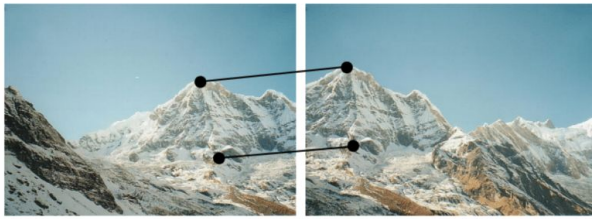


3. Estimate transformation

Lecture 5 **RANSAC**

(find the best among many random trials)

3a. Pick inlier matches



Lecture 6 - DONE

3b. Estimate transformation

$$\{(x, y), (x', y')\}_K \rightarrow T$$

Lecture 6 - DONE 4. Warp!

Coding lab

Next lecture: segmentation