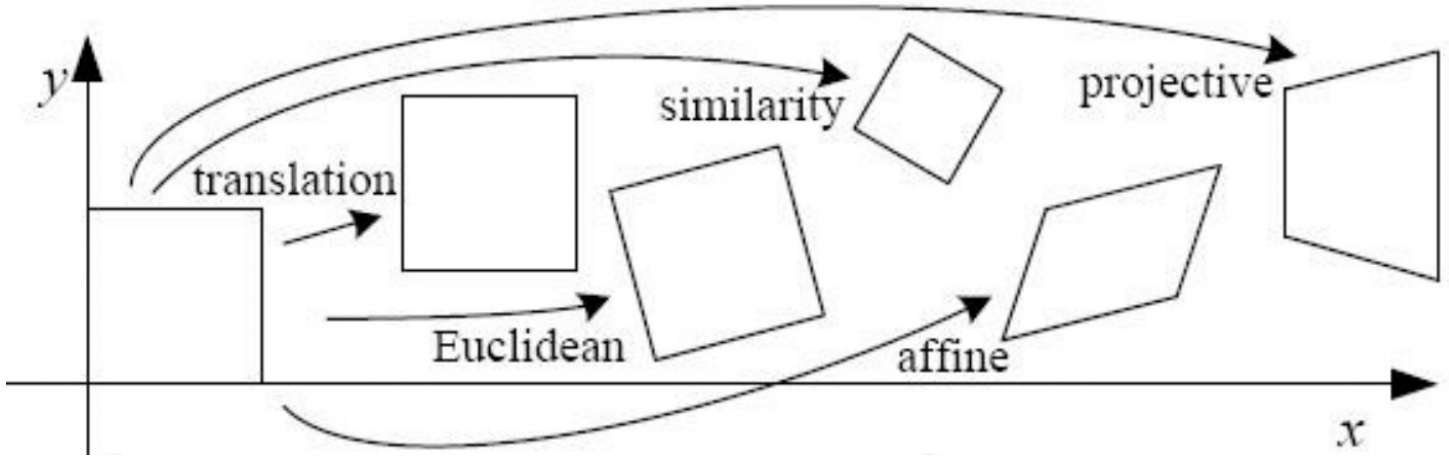
3. Image Transformations and Building Panaromas

Instructor: Shaifali Parashar

Many Slides from Fei-Fei Li's lectures at Standford

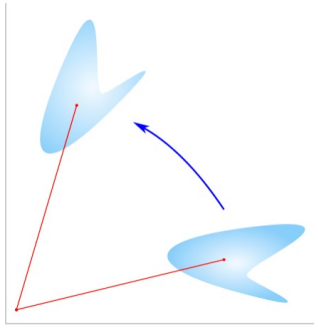
Transformations



Matrix Transformations: Rotation

In 2D: degrees of freedom = 1

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



In 3D: degrees of freedom = 3

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

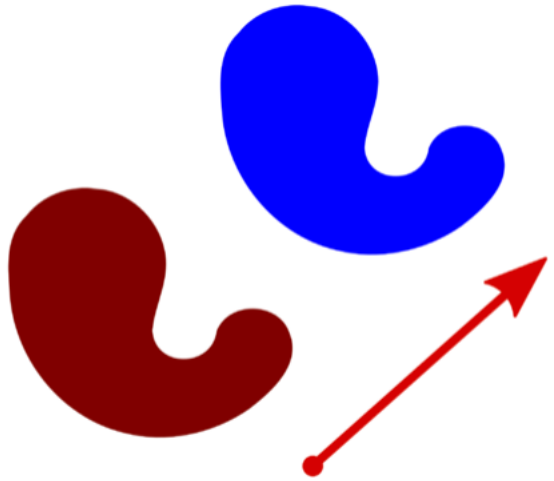
$$\mathbf{R} = \mathbf{R_z}\mathbf{R_y}\mathbf{R_x} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

Rotation is an orthogonal transformation, det(**R**) = 1
Reflection is like rotation, but det(**R**) = -1
Axis-angle representations: Euler angles, Quaternions, Rodrigue's formula

Matrix Transformations: Translation

In 2D: degrees of freedom = 2

In 3D: degrees of freedom = 3



Matrix Transformations: Scaling

In 2D: $\mathbf{S} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$

Isotropic: $s_x = s_y$

Anisotropic: $s_x \neq s_y$

degrees of freedom (isotropic): 1

degrees of freedom (anisotropic): 2

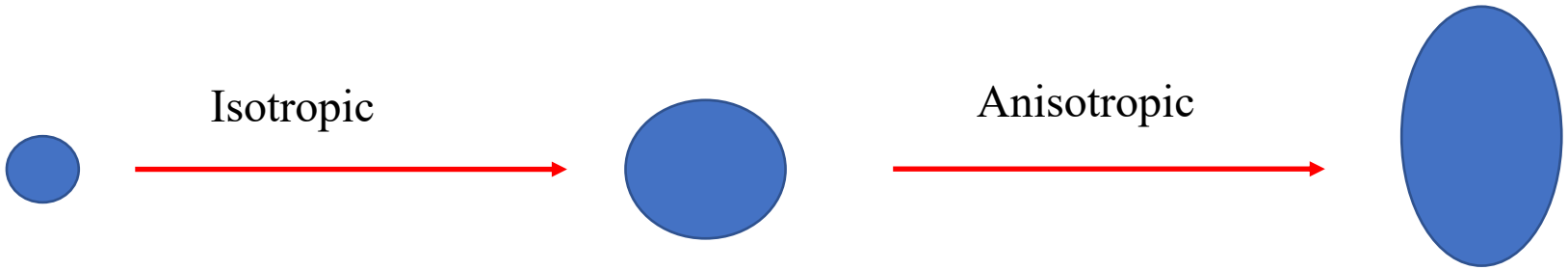
In 3D: $\mathbf{S} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$

$s_x = s_y = s_z$

$s_x \neq s_y \neq s_z$

degrees of freedom (isotropic): 1

degrees of freedom (anisotropic): 3



Matrix Transformations: Shear

In 2D:

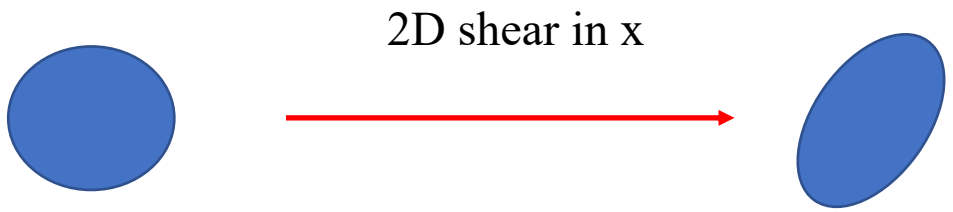
$$\mathbf{S} = \begin{pmatrix} 1 & s_x \\ s_y & 1 \end{pmatrix}$$

degrees of freedom: 2

In 3D:

$$\mathbf{S} = \begin{pmatrix} 1 & s_{x1} & s_{x2} \\ s_{y1} & 1 & s_{y2} \\ s_{z1} & s_{z2} & 1 \end{pmatrix}$$

degrees of freedom: 6



Euclidean Transformations

Also known as rigid transformations or isometries

Preserve lengths, angles and areas

A combination of rotations, reflections and translations

Euclidean Transformations

Also known as rigid transformations or isometries

Preserve lengths, angles and areas

A combination of rotations, reflections and translations

In 2D: degrees of freedom: 1(rotation) + 2(translation) = 3 $[\mathbf{R}|\mathbf{t}]_{2\times3}$

In 3D: degrees of freedom: 3(rotation) + 3(translation) = 6 $[\mathbf{R}|\mathbf{t}]_{3\times4}$

Similarity Transformations

Preserve ratio of lengths, angles

A combination of scale, rotations, reflections and translations

In 2D: degrees of freedom: 1(scale) + 1(rotation) + 2(translation) = 4 $s[\mathbf{R}|\mathbf{t}]_{2\times 3}$

In 3D: degrees of freedom: 1(scale) + 3(rotation) + 3(translation) = 6 $s[\mathbf{R}|\mathbf{t}]_{3\times 4}$

Affine Transformations

Preserve ratio of lengths, ratio of areas, parallelism

A combination of shear, scale, rotations, reflections and translations

In 2D: degrees of freedom: 2(shear) + 2(scale) + 1(rotation) + 2(translation) = 6

In 3D: degrees of freedom: 6(shear) + 3(scale) + 3(rotation) + 3(translation) = 12

Affine Transformations

Preserve ratio of lengths, ratio of areas, parallelism

A combination of shear, scale, rotations, reflections and translations

In 2D: degrees of freedom: 2(shear) + 2(scale) + 1(rotation) + 2(translation) = 6

In 3D: degrees of freedom: 6(shear) + 3(scale) + 3(rotation) + 3(translation) = 12

Degrees of freedom (dof) can’t be more than matrix size.

In 2D, its 2X3 =6 and in 3D, its 3X4=12.

Affine > Similarity > Euclidean

Transformations

To transform **x**, we write (using homogeneous coordinates): $\overline{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{t}$

$$\begin{pmatrix} \overline{\mathbf{x}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & t \\ 0_{1\times d}^{\top} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

d is dimension

Euclidean: **A** = **R**

Similarity: **A** = s**R**

Affine: **Ax + t**

Projective Transformations

Preserves collinearity

A combination of distortion, shear, scale, rotations, reflections and translations

In 2D: degrees of freedom: 8

In 3D: degrees of freedom: 15

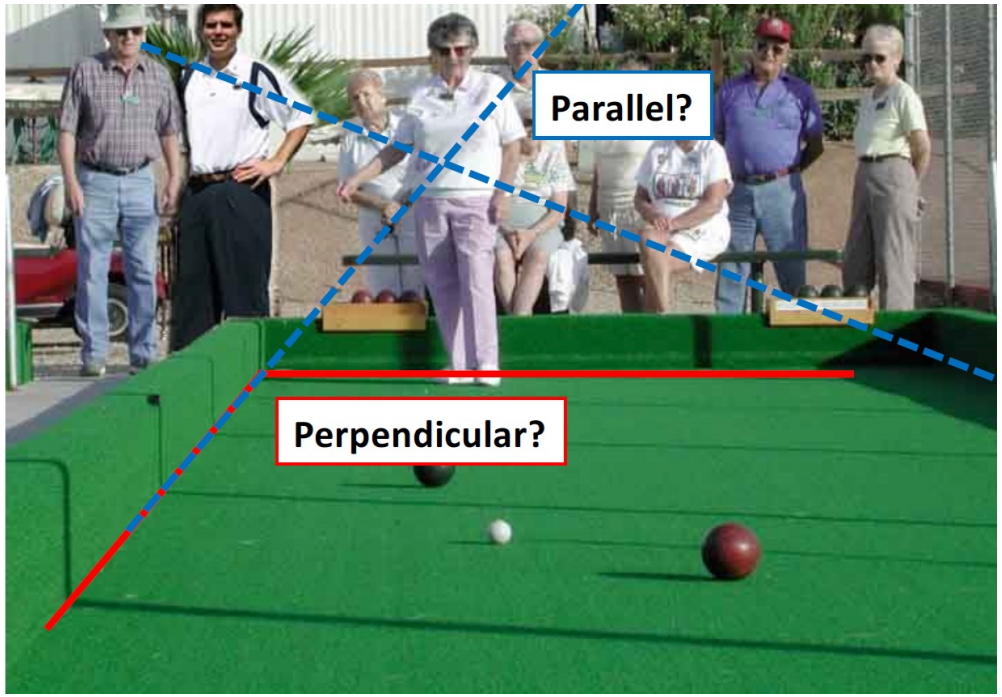
$$\begin{pmatrix} \overline{\mathbf{x}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & t \\ \mathbf{v}^{\top}_{1\times d} & v \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

Projective > Affine > Similarity > Euclidean

Always look at the rank of matrix to see how many dimensions does it impact

Images formation

Lengths, angles are lost
Parallelism is lost
Collinearity is preserved

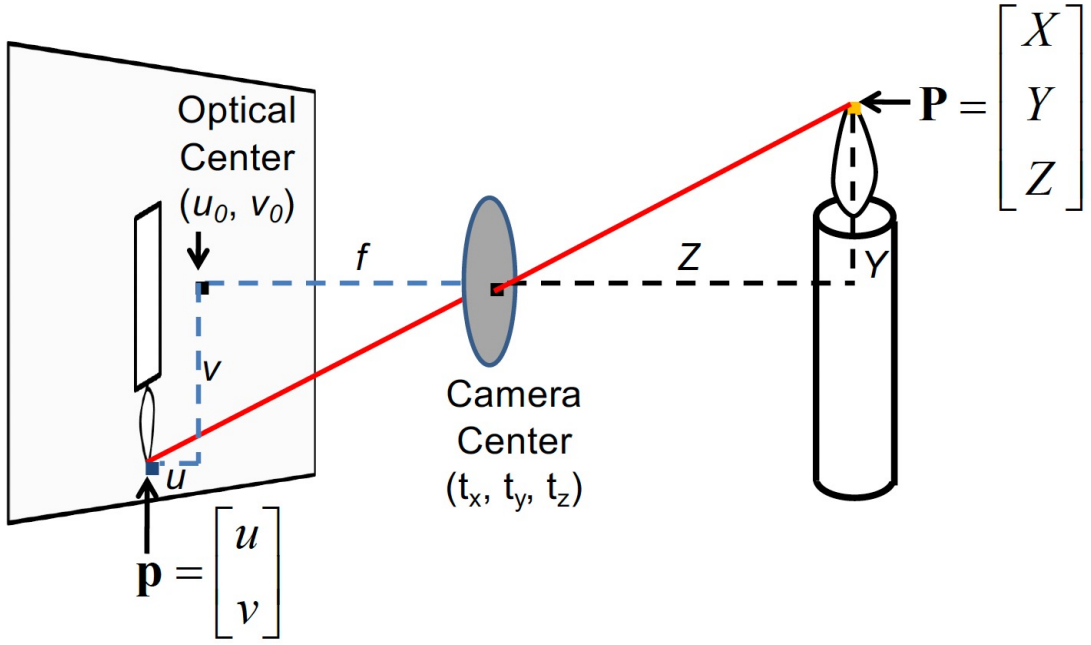


Images formation-Pin hole camera

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f\dfrac{X}{Z} \\ -f\dfrac{Y}{Z} \end{bmatrix}$$

In fact, we invert and shift the image.

p ----> $[u_0, v_0]$ - **p**

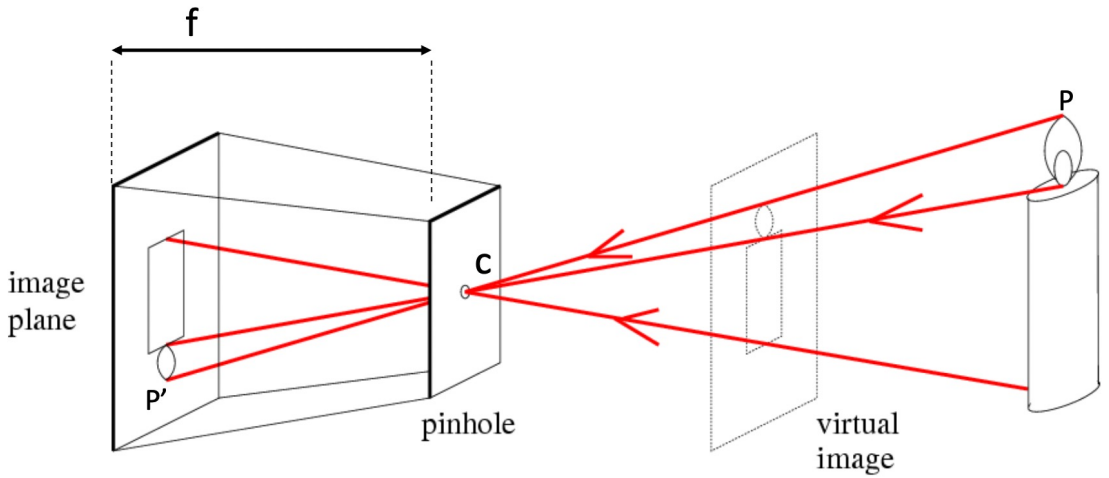


Homogeneous coordinates

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

homogeneous image coordinates

homogeneous scene coordinates



Converting from homogeneous coordinates

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

Also invariant to scale, multiplication with a scalar results movement along a ray

Another problem solved by homogeneous coordinates

Intersection of parallel lines

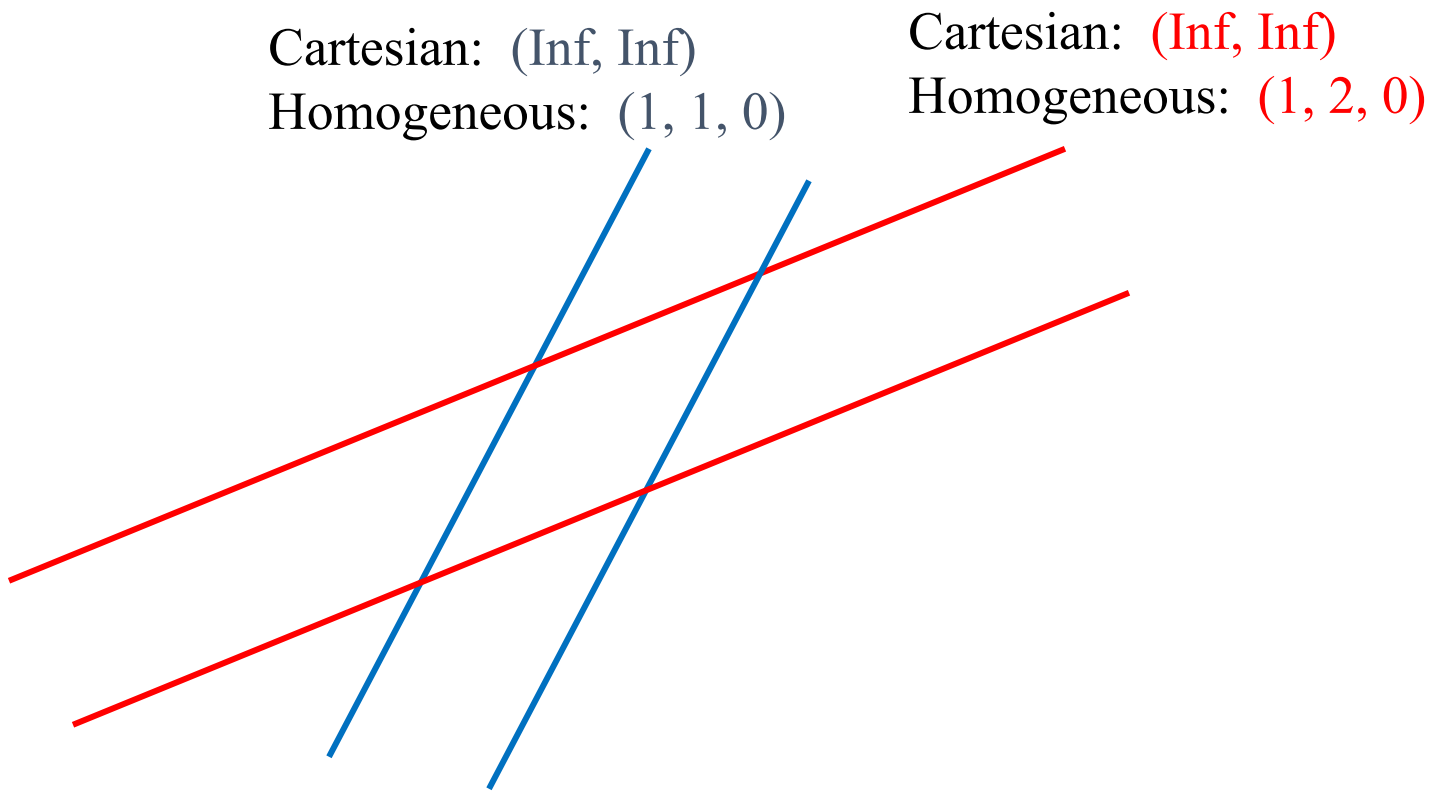
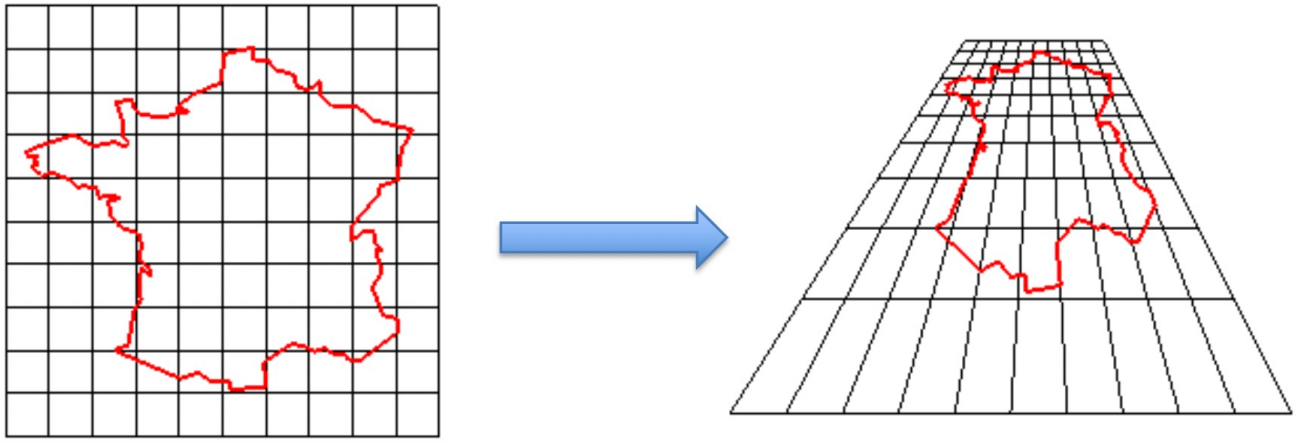


Image Transformations (Homography)



$$c\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Image transformations → Projective transformation → Homography

Homography Estimation

- If point correspondences (x,y)<-->(x',y') are known

We can find a 3X3 transformation, **A** such that
x' = **Ax**

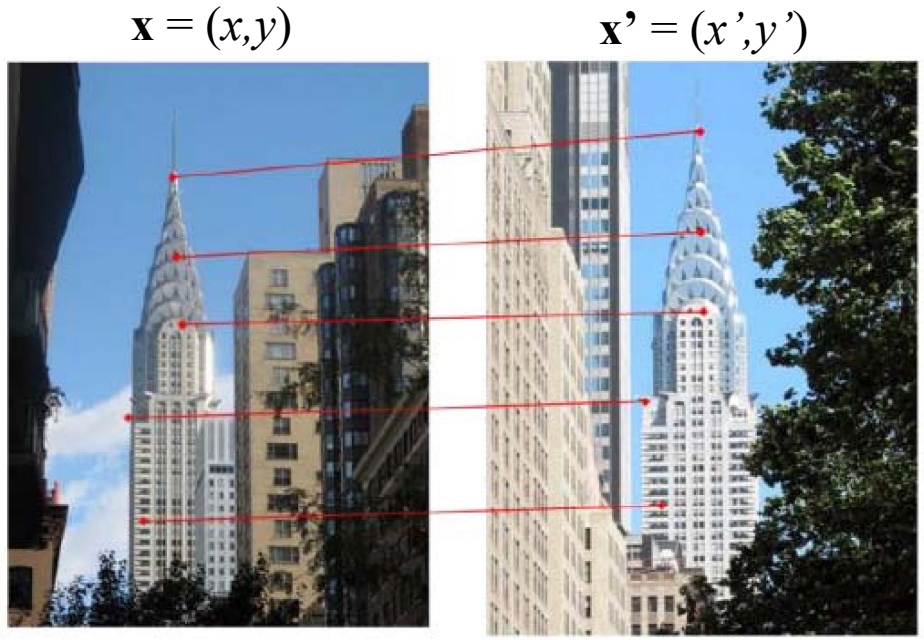
A = $[a_1, a_2, a_3; a_4, a_5, a_6; a_7, a_8, a_9]$

Remember from first lecture on transformations, a 2D projective transformation (most generic transformation) has 8 dof

Assume $a_9 = 1$, therefore

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$



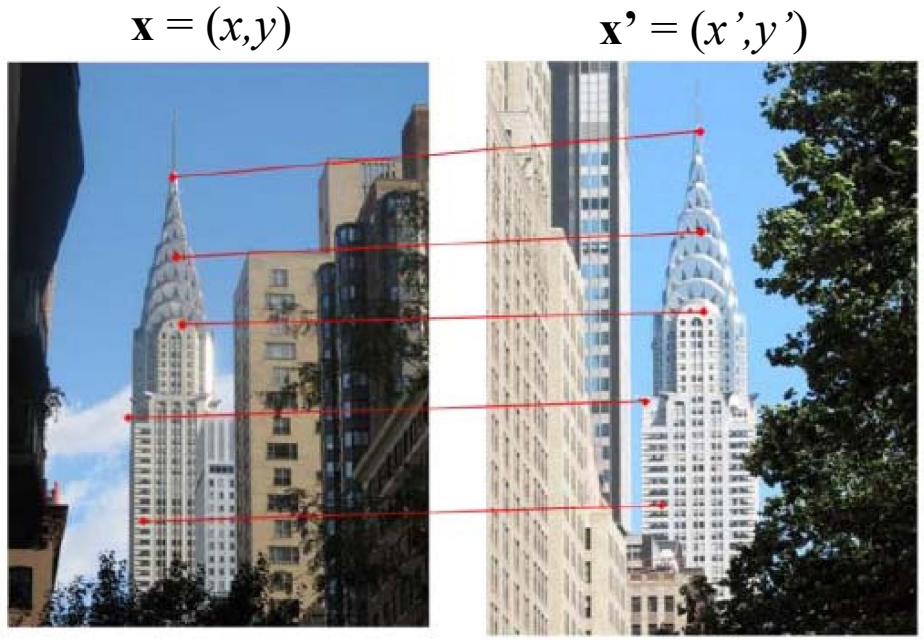
Homography Estimation

Assume $a_9 = 1$, therefore

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

$$a_7 x' x + a_8 x' y + x' = a_1 x + a_2 y + a_3$$
$$a_7 y' x + a_8 y' y + y' = a_7 x + a_8 y + a_6$$
$$x' = a_1 x + a_2 y + a_3 - a_7 x' x - a_8 x' y$$
$$y' = a_7 x + a_8 y + a_6 - a_7 y' x - a_8 y' y$$



Segregate the equations into matrices/vectors of knowns (**x**, **x'**) and unknowns (a_1-a_8)

Homography Estimation

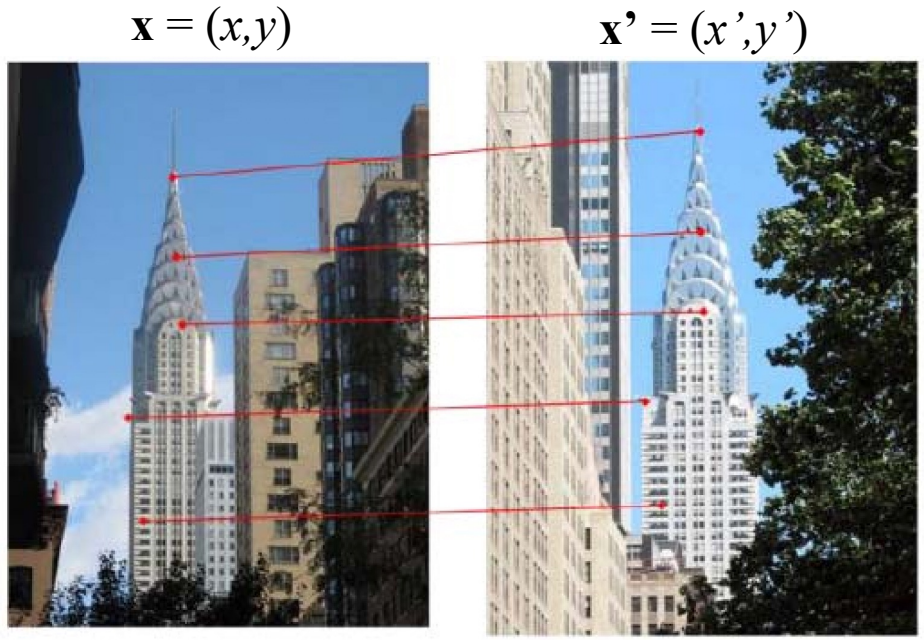
$$a_7 x'x + a_8 x'y + x' = a_1 x + a_2 y + a_3$$
$$a_7 y'x + a_8 y'y + y' = a_7 x + a_8 y + a_6$$
$$x' = a_1 x + a_2 y + a_3 - a_7 x'x - a_8 x'y$$
$$y' = a_7 x + a_8 y + a_6 - a_7 y'x - a_8 y'y$$

Two rows for each point *i*

$$\begin{bmatrix} \vdots & & & & & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i' & -y_i y_i' \\ \vdots & & & & & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

$$\mathbf{T}_{8X8}\mathbf{a}_{8X1} = \mathbf{B}_{8X1}$$

Linear Least squares: $a = (\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T\mathbf{B}$



Homography using Direct Linear Transformation (DLT)

Don't assume $a_9 = 1$, add it to the system

In order to obtain a non-trivial solution,

we obtain **a :**

min || **T'a** || such that ||**a**|| =1

Two rows for each point *i*

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i' & -y_i y_i' \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

a_9

$\mathbf{T'}_{8X9}\mathbf{a}_{9X1} = \mathbf{0}_{9X1}$

$\mathbf{T'} = [\mathbf{T},\mathbf{-B}]$

Homography using Direct Linear Transformation (DLT)

Don't assume $a_9 = 1$, add it to the system

In order to obtain a non-trivial solution,

we obtain **a :**

min || **T'a** || such that ||**a**|| =1

Solution: SVD of **T'**. The eigenvector corresponding to smallest eigenvalue of $\mathbf{T'}^{T}\mathbf{T'}$ is the solution.

In order to get a stable solution, Normalise the data before DLT.

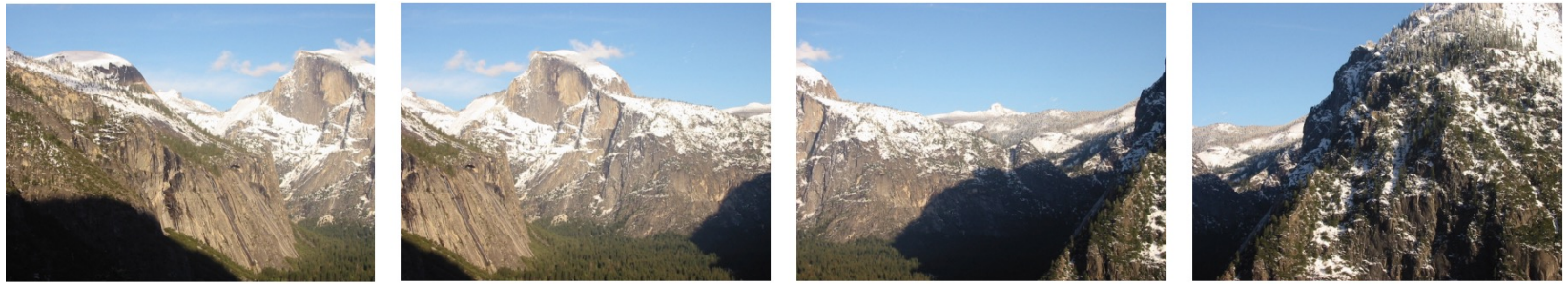
a) Translate for zero mean

b) b) Scale so that average distance to origin is ~sqrt(2)

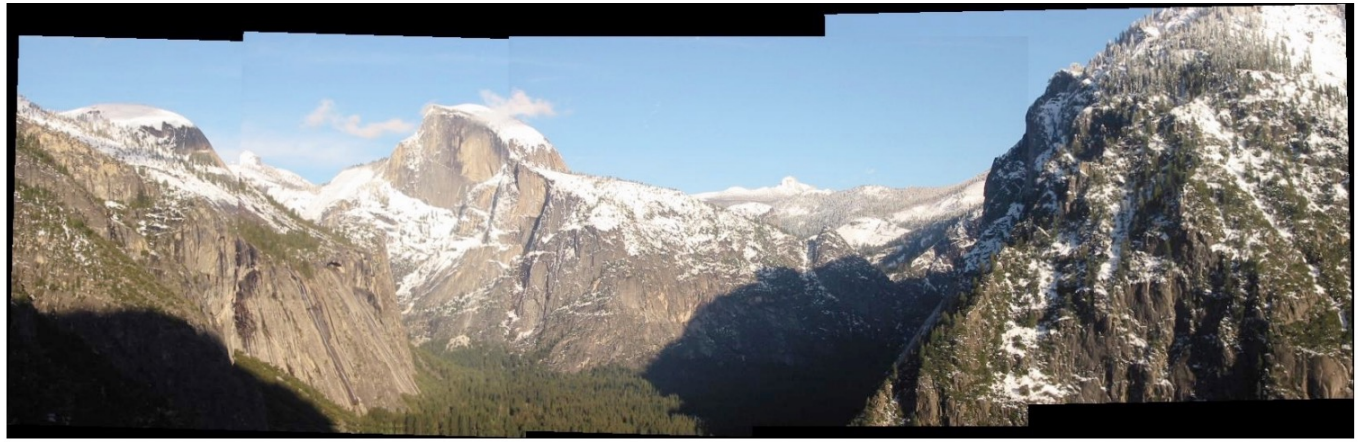
https://www.cs.cmu.edu/~16385/s17/Slides/10.2_2D_Alignment__DLT.pdf

Panoramic Image Stitching

Given a set of images



Obtain:



Problem with Linear Least Squares (LLS) fitting

1. Noisy data and outliers

Solution:

a) Voting. Check all possible combinations. Cycle through features, cast votes for model parameters. Bad idea due to high computation cost.

b) RANdom SAmple Consensus (RANSAC)

RANSAC

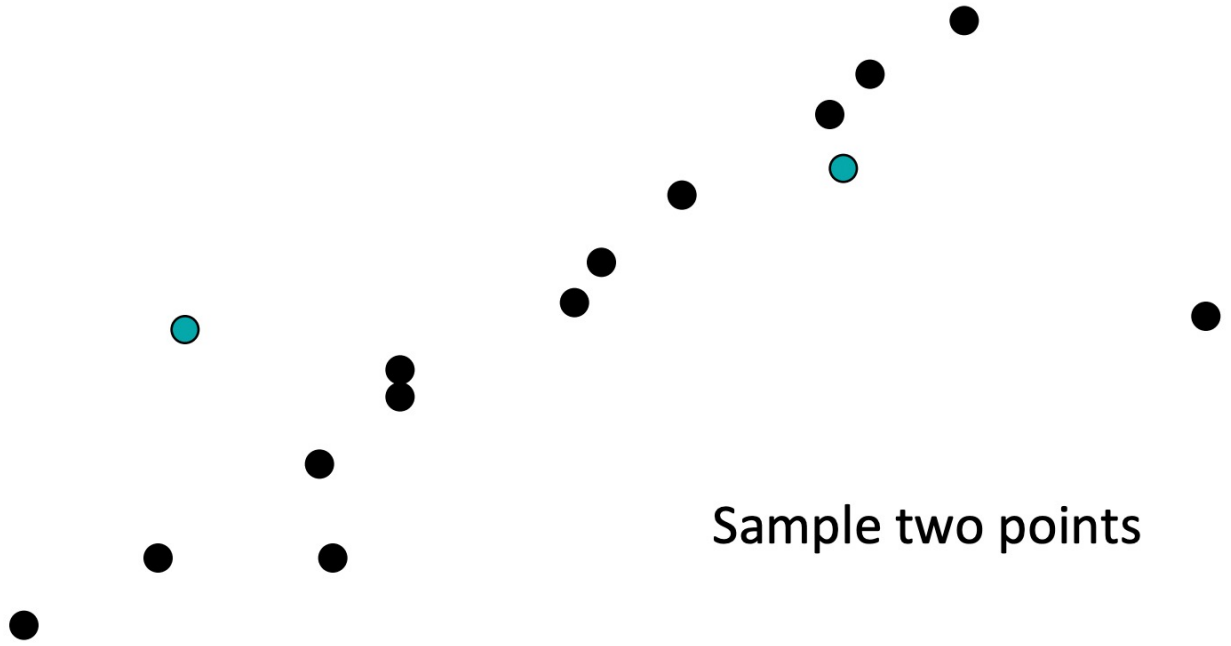
Look for ‘only’ inliers.

1. Randomly select a seed group to estimate transformation
2. Compute transformation
3. Compute inliers
4. If inliers are large, recompute transformation

Keep transformation with most inliers

RANSAC: line fitting example

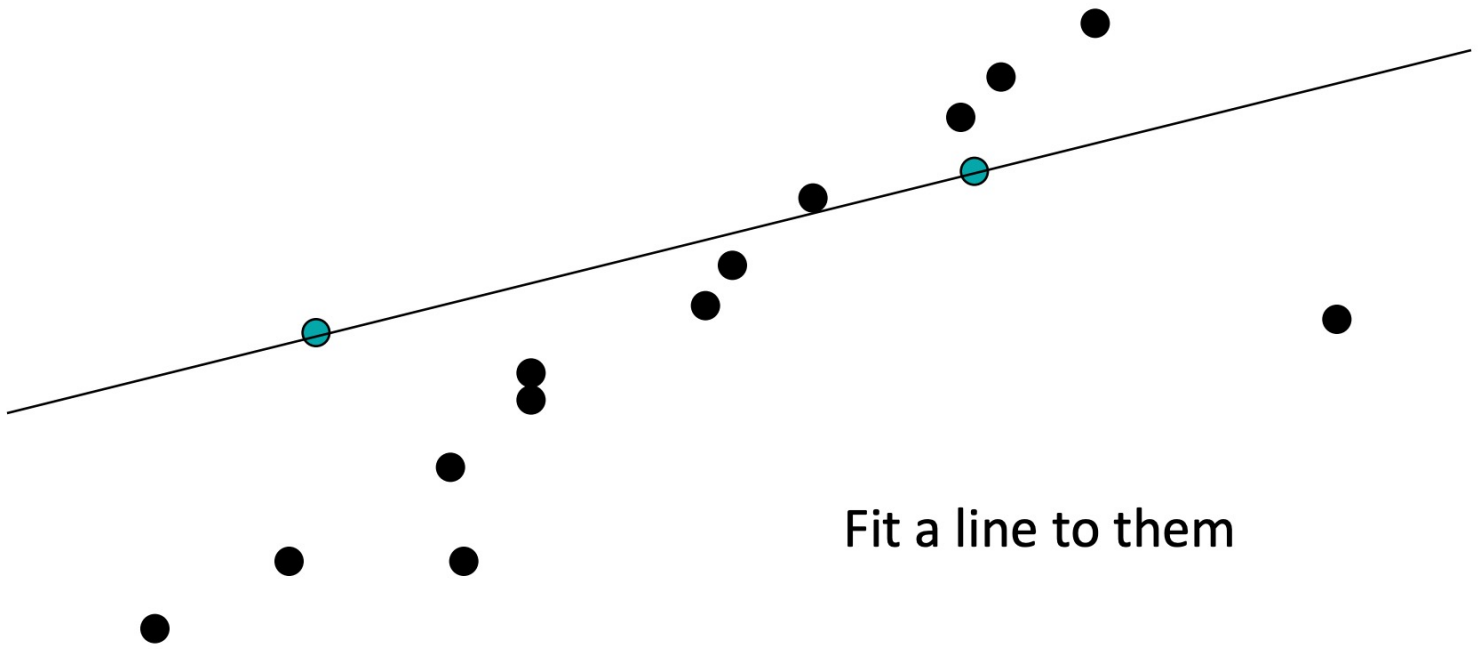
Estimate the best line



Slide credit: Jinxiang Chai

RANSAC: line fitting example

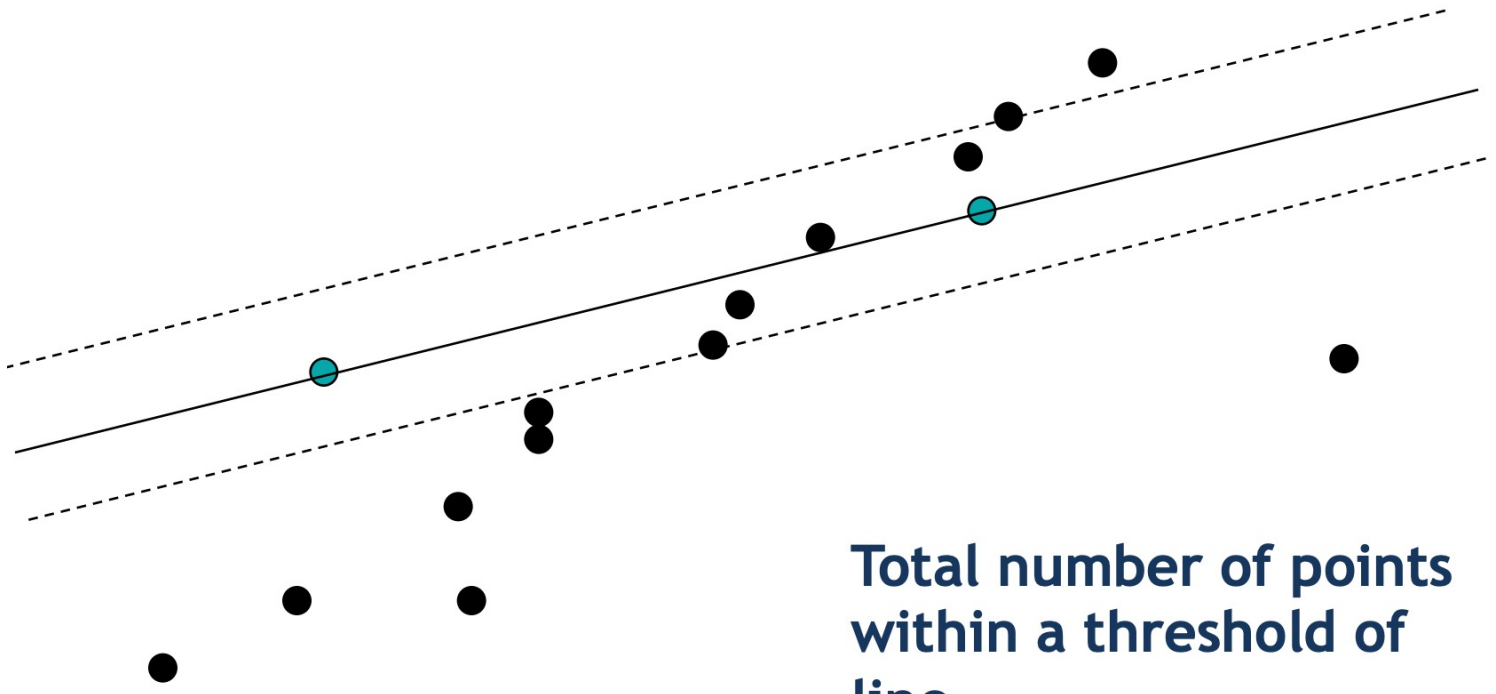
Estimate the best line



Slide credit: Jinxiang Chai

RANSAC: line fitting example

Estimate the best line

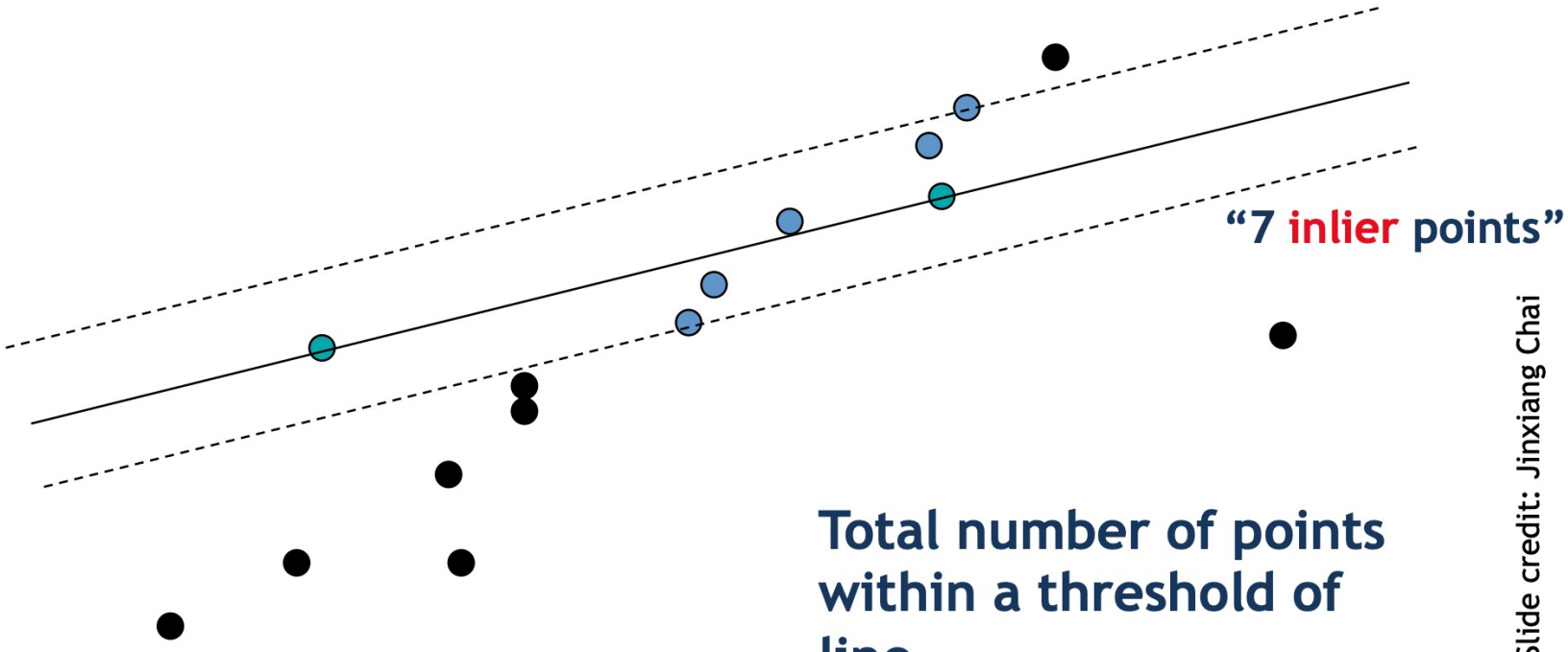


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

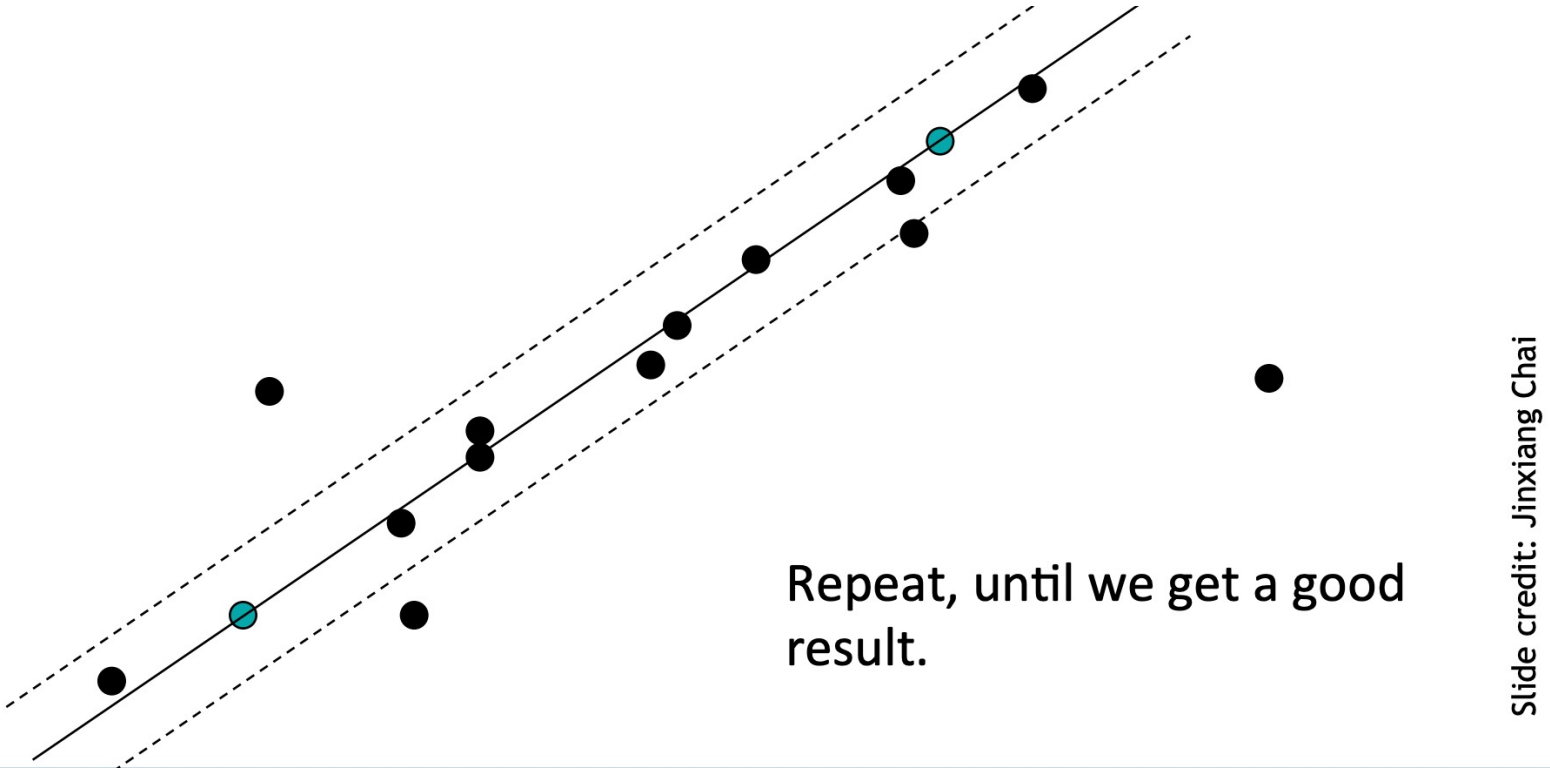
RANSAC: line fitting example

Estimate the best line



RANSAC: line fitting example

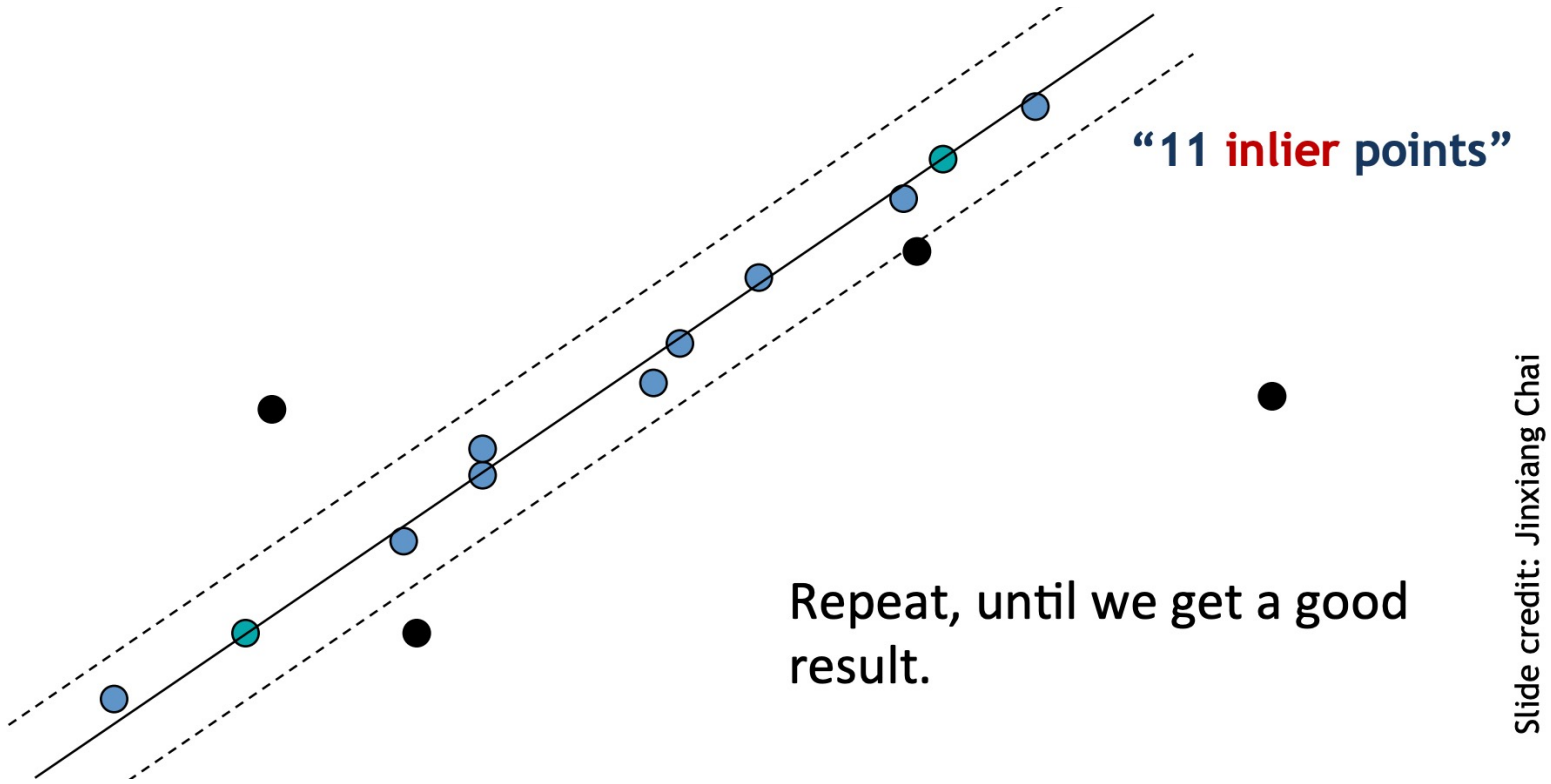
Estimate the best line



Slide credit: Jinxiang Chai

RANSAC: line fitting example

Estimate the best line



Slide credit: Jinxiang Chai

RANSAC: line-fitting example

After RANSAC, use all inliers to improve transformation. This may change the transformation, so perform another reclassification of inliers-outliers.

RANSAC is fast but it can accommodate relatively small number of outliers.

Voting is efficient but computationally expensive.

RANSAC: line-fitting example

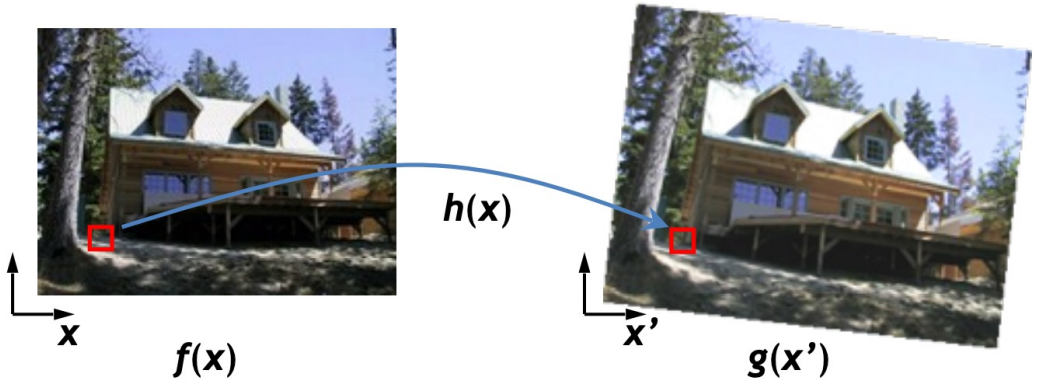
After RANSAC, use all inliers to improve transformation. This may change the transformation, so perform another reclassification of inliers-outliers.

RANSAC is fast but it can accommodate relatively small number of outliers.

Voting is efficient but computationally expensive.

Image Warping: Forward

Given a transformation x'=h(x) and source image f(x), how do we compute a transformed image g(x')=f(h(x))?



- What if pixel lands "between" two pixels?
- Answer: add "contribution" to several pixels, normalize later (*splatting*)

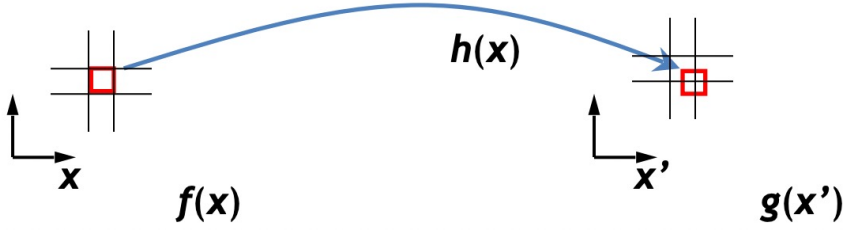
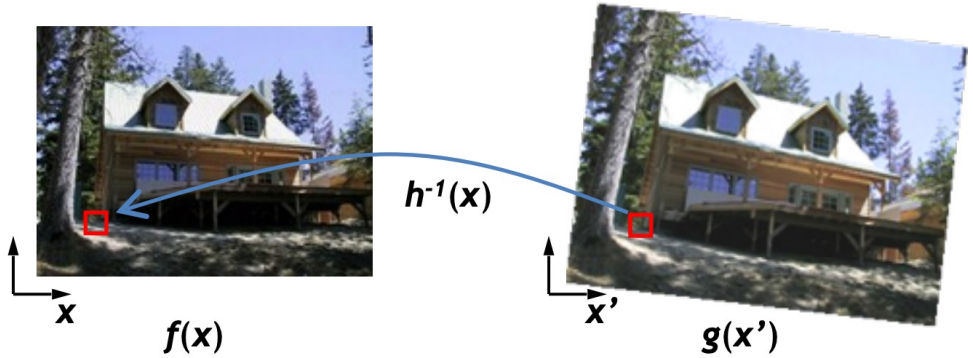
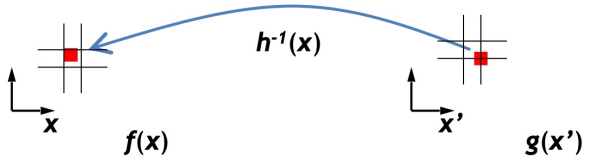


Image Warping: Backward

Get each pixel g(x') from its corresponding location x' = h(x) in f(x) ?

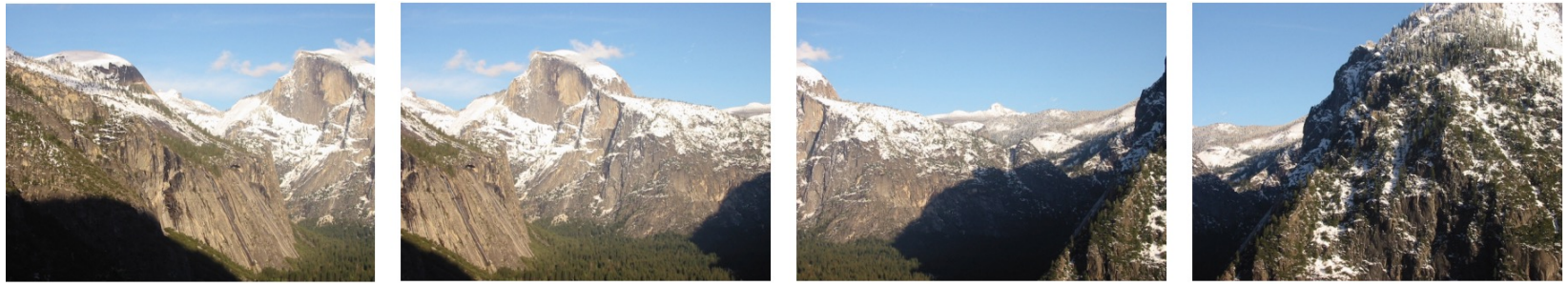


- What if pixel comes from "between" two pixels?
- Answer: *resample* color value from *interpolated* source image



Panoramic Image Stitching

Given a set of images



Obtain: