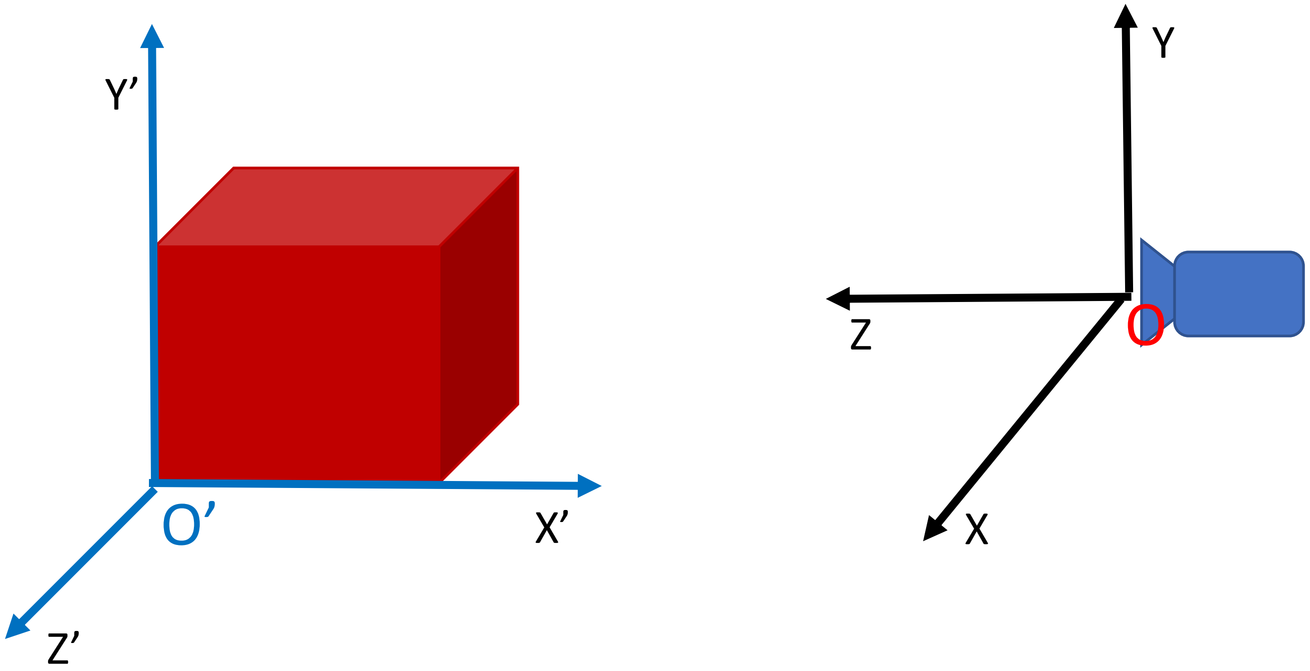
Reconstruction

Perspective projection



Final perspective projection

Camera extrinsics: where your camera is relative to the world. Changes if you move the camera

$$\vec{\mathbf{x}}_{img} \equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

Camera intrinsics: how your camera handles pixel. Changes if you change your camera

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

Final perspective projection

$$\vec{\mathbf{x}}_{img} \equiv \underbrace{K \begin{bmatrix} R & \mathbf{t} \end{bmatrix}}_{\text{Camera parameters}} \vec{\mathbf{x}}_w$$

$$\vec{\mathbf{x}}_{img} \equiv P \vec{\mathbf{x}}_w$$

Camera calibration

- Goal: find the parameters of the camera

$$\vec{\mathbf{x}}_{img} \equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

- Why?
  - Tells you where the camera is relative to the world/particular objects
  - Equivalently, tells you where objects are relative to the camera
  - Can allow you to ”render” new objects into the scene

Camera calibration

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

- Need to estimate P
- How many parameters does P have?
  - Size of P : 3 x 4
  - But: $\lambda P\vec{\mathbf{x}}_w \equiv P\vec{\mathbf{x}}_w$
  - P can only be known *upto a scale*
  - *3*4 - 1 = 11 parameters*

Camera calibration

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

- Suppose we know that (X,Y,Z) in the world projects to (x,y) in the image.
- How many equations does this provide?

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Need to convert equivalence into equality.

Camera calibration

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

- Suppose we know that (X,Y,Z) in the world projects to (x,y) in the image.
- How many equations does this provide?

Note: λ is unknown

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera calibration

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

- Suppose we know that (X,Y,Z) in the world projects to (x,y) in the image.
- How many equations does this provide?

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera calibration

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

- Suppose we know that (X,Y,Z) in the world projects to (x,y) in the image.
- How many equations does this provide?

$$\lambda x = P_{11}X + P_{12}Y + P_{13}Z + P_{14}$$

$$\lambda y = P_{21}X + P_{22}Y + P_{23}Z + P_{24}$$

$$\lambda = P_{31}X + P_{32}Y + P_{33}Z + P_{34}$$

Camera calibration

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

- Suppose we know that (X,Y,Z) in the world projects to (x,y) in the image.
- How many equations does this provide?

$$(P_{31}X + P_{32}Y + P_{33}Z + P_{34})x = P_{11}X + P_{12}Y + P_{13}Z + P_{14}$$
$$(P_{31}X + P_{32}Y + P_{33}Z + P_{34})y = P_{21}X + P_{22}Y + P_{23}Z + P_{24}$$

- 2 equations!
- Are the equations linear in the parameters?
- How many equations do we need?

Camera calibration

$$(P_{31}X + P_{32}Y + P_{33}Z + P_{34})x = P_{11}X + P_{12}Y + P_{13}Z + P_{14}$$

$$XxP_{31} + YxP_{32} + ZxP_{33} + xP_{34} - XP_{11} - YP_{12} - ZP_{13} - P_{14} = 0$$

- In matrix vector form: Ap = 0
- 6 points give 12 equations, 12 variables to solve for
- But can only solve upto scale

Camera calibration

- In matrix vector form: Ap = 0
- We want non-trivial solutions
- If p is a solution, αp is a solution too
- Let's just search for a solution with unit norm

$$A\mathbf{p} = 0$$

s.t

$$\|\mathbf{p}\| = 1$$

- How do you solve this?

Camera calibration

- In matrix vector form: Ap = 0
- We want non-trivial solutions
- If p is a solution, αp is a solution too
- Let's just search for a solution with unit norm

$$A\mathbf{p} = 0$$

s.t

$$\|\mathbf{p}\| = 1$$

- How do you solve this? *Eigenvector with 0 eigenvalue!*

Camera calibration

- We need 6 world points for which we know image locations
- Would any 6 points work?
  - What if all 6 points are the same?
- Need at least 6 non-coplanar points!

Camera calibration

$$\vec{\mathbf{x}}_{img} \equiv P\vec{\mathbf{x}}_w$$

$$\vec{\mathbf{x}}_{img} \equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

- How do we get K, R and t from P?
- Need to make some assumptions about K
- What if K is upper triangular?

$$K = \begin{bmatrix} s_x & \alpha & t_u \\ 0 & s_y & t_v \\ 0 & 0 & 1 \end{bmatrix}$$

Added skew if image x and y axes are not perpendicular

Camera calibration

- How do we get K, R and t from P?
- Need to make some assumptions about K
- What if K is upper triangular?

$$K = \begin{bmatrix} s_x & \alpha & t_u \\ 0 & s_y & t_v \\ 0 & 0 & 1 \end{bmatrix}$$

- P = K [R t]
- First 3 x 3 matrix of P is KR
- “RQ” decomposition: decomposes an n x n matrix into product of upper triangular and rotation matrix

Camera calibration

- How do we get K, R and t from P?
- Need to make some assumptions about K
- What if K is upper triangular?
- $P = K\,[\,R\ \ t]$
- First 3 x 3 matrix of P is KR
- "RQ" decomposition: decomposes an n x n matrix into product of upper triangular and rotation matrix
- $t = K^{-1}P[:,2]$ ← last column of P

Camera calibration and pose estimation

Final perspective projection

$$\vec{\mathbf{x}}_{img} \equiv K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

Can we recover this from just a single equation?

$$\vec{\mathbf{x}}_{img} \equiv P \vec{\mathbf{x}}_w$$

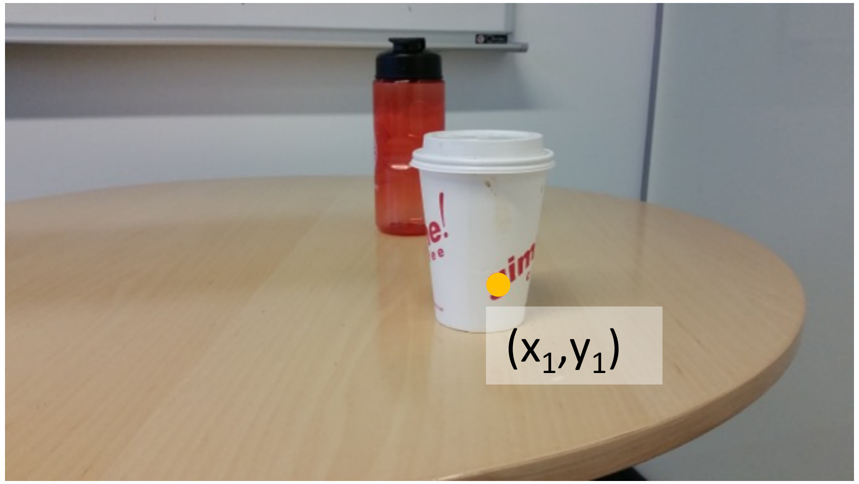
Triangulation

- Suppose we have two cameras
  - Calibrated: parameters known
- And a pair of corresponding pixels
- Find 3D location of point!

Triangulation

- Suppose we have two cameras
  - Calibrated: parameters known
- And a pair of corresponding pixels
- Find 3D location of point!

$P^{(1)}$



$P^{(2)}$

Triangulation

$$\begin{tikzcd}
\begin{bmatrix}x_1\\y_1\\1\end{bmatrix} & \vec{\mathbf{x}}^{(1)}_{img} \equiv P^{(1)}\vec{\mathbf{x}}_w \arrow[l] \arrow[dr] & \\
& & \begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \\
\begin{bmatrix}x_2\\y_2\\1\end{bmatrix} & \vec{\mathbf{x}}^{(2)}_{img} \equiv P^{(2)}\vec{\mathbf{x}}_w \arrow[l] \arrow[ur] &
\end{tikzcd}$$

Triangulation

$$\vec{\mathbf{x}}^{(1)}_{img} \equiv P^{(1)} \vec{\mathbf{x}}_w$$

$$\lambda x_1 = P^{(1)}_{11} X + P^{(1)}_{12} Y + P^{(1)}_{13} Z + P^{(1)}_{14}$$

$$\lambda y_1 = P^{(1)}_{21} X + P^{(1)}_{22} Y + P^{(1)}_{23} Z + P^{(1)}_{24}$$

$$\lambda = P^{(1)}_{31} X + P^{(1)}_{32} Y + P^{(1)}_{33} Z + P^{(1)}_{34}$$

$$(P^{(1)}_{31} X + P^{(1)}_{32} Y + P^{(1)}_{33} Z + P^{(1)}_{34}) x_1 = P^{(1)}_{11} X + P^{(1)}_{12} Y + P^{(1)}_{13} Z + P^{(1)}_{14}$$

$$X(P^{(1)}_{31} x_1 - P^{(1)}_{11}) + Y(P^{(1)}_{32} x_1 - P^{(1)}_{12}) + Z(P^{(1)}_{33} x_1 - P^{(1)}_{13}) + (P^{(1)}_{34} x_1 - P^{(1)}_{14}) = 0$$

Triangulation

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv P^{(1)}\vec{\mathbf{x}}_w$$

$$X(P_{31}^{(1)}x_1 - P_{11}^{(1)}) + Y(P_{32}^{(1)}x_1 - P_{12}^{(1)}) + Z(P_{33}^{(1)}x_1 - P_{13}^{(1)}) + (P_{34}^{(1)}x_1 - P_{14}^{(1)}) = 0$$

$$X(P_{31}^{(1)}y_1 - P_{21}^{(1)}) + Y(P_{32}^{(1)}y_1 - P_{22}^{(1)}) + Z(P_{33}^{(1)}y_1 - P_{23}^{(1)}) + (P_{34}^{(1)}y_1 - P_{24}^{(1)}) = 0$$

- 1 image gives 2 equations
- Need 2 images!
- Solve linear equations to get 3D point location

Linear vs non-linear optimization

$$\lambda x_1 = P^{(1)}_{11} X + P^{(1)}_{12} Y + P^{(1)}_{13} Z + P^{(1)}_{14}$$
$$\lambda y_1 = P^{(1)}_{21} X + P^{(1)}_{22} Y + P^{(1)}_{23} Z + P^{(1)}_{24}$$
$$\lambda = P^{(1)}_{31} X + P^{(1)}_{32} Y + P^{(1)}_{33} Z + P^{(1)}_{34}$$

$$x_1 = \frac{P^{(1)}_{11} X + P^{(1)}_{12} Y + P^{(1)}_{13} Z + P^{(1)}_{14}}{P^{(1)}_{31} X + P^{(1)}_{32} Y + P^{(1)}_{33} Z + P^{(1)}_{34}}$$

$$y_1 = \frac{P^{(1)}_{21} X + P^{(1)}_{22} Y + P^{(1)}_{23} Z + P^{(1)}_{24}}{P^{(1)}_{31} X + P^{(1)}_{32} Y + P^{(1)}_{33} Z + P^{(1)}_{34}}$$

Linear vs non-linear optimization

$$x_1 = \frac{P^{(1)}_{11}X + P^{(1)}_{12}Y + P^{(1)}_{13}Z + P^{(1)}_{14}}{P^{(1)}_{31}X + P^{(1)}_{32}Y + P^{(1)}_{33}Z + P^{(1)}_{34}}$$

$$y_1 = \frac{P^{(1)}_{21}X + P^{(1)}_{22}Y + P^{(1)}_{23}Z + P^{(1)}_{24}}{P^{(1)}_{31}X + P^{(1)}_{32}Y + P^{(1)}_{33}Z + P^{(1)}_{34}}$$

$$\boxed{\begin{aligned}&(x_1 - \frac{P^{(1)}_{11}X + P^{(1)}_{12}Y + P^{(1)}_{13}Z + P^{(1)}_{14}}{P^{(1)}_{31}X + P^{(1)}_{32}Y + P^{(1)}_{33}Z + P^{(1)}_{34}})^2\\ +&(y_1 - \frac{P^{(1)}_{21}X + P^{(1)}_{22}Y + P^{(1)}_{23}Z + P^{(1)}_{24}}{P^{(1)}_{31}X + P^{(1)}_{32}Y + P^{(1)}_{33}Z + P^{(1)}_{34}})^2\end{aligned}}$$

Reprojection error

Linear vs non-linear optimization

$$
\boxed{\begin{aligned}
&(x_1 - \frac{P^{(1)}_{11}X + P^{(1)}_{12}Y + P^{(1)}_{13}Z + P^{(1)}_{14}}{P^{(1)}_{31}X + P^{(1)}_{32}Y + P^{(1)}_{33}Z + P^{(1)}_{34}})^2 \\
+&(y_1 - \frac{P^{(1)}_{21}X + P^{(1)}_{22}Y + P^{(1)}_{23}Z + P^{(1)}_{24}}{P^{(1)}_{31}X + P^{(1)}_{32}Y + P^{(1)}_{33}Z + P^{(1)}_{34}})^2
\end{aligned}}
$$

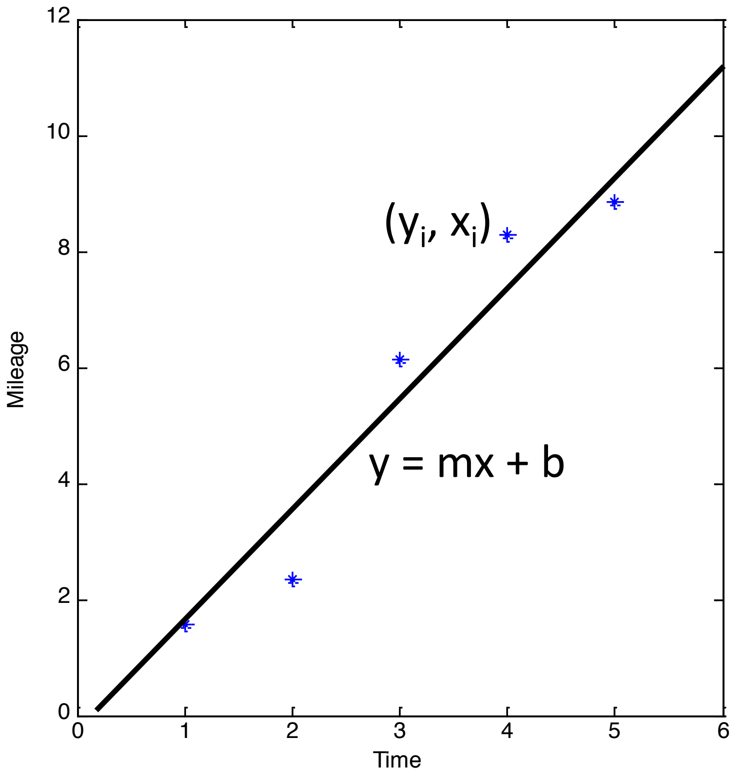
Reprojection error

- Reprojection error is the squared error between the true image coordinates of a point and the projected coordinates of hypothesized 3D point
- Actual error we care about
- Minimize total sum of reprojection error across all images
- *Non-linear optimization*

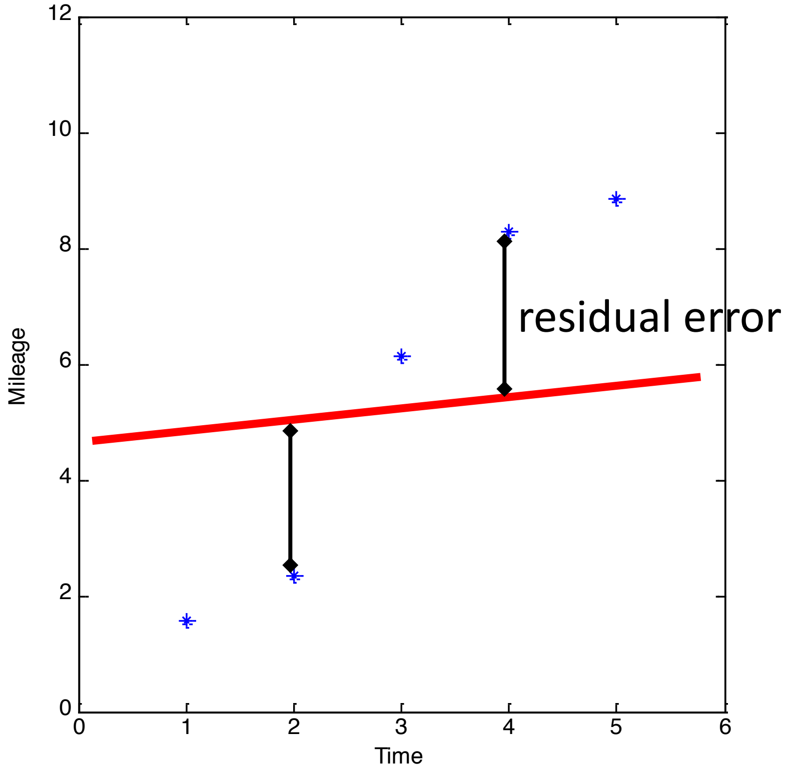
Fitting in general

- Fitting: find the parameters of a model that best fit the data
- Other examples:
  - least squares linear regression

Least squares: linear regression



Linear regression

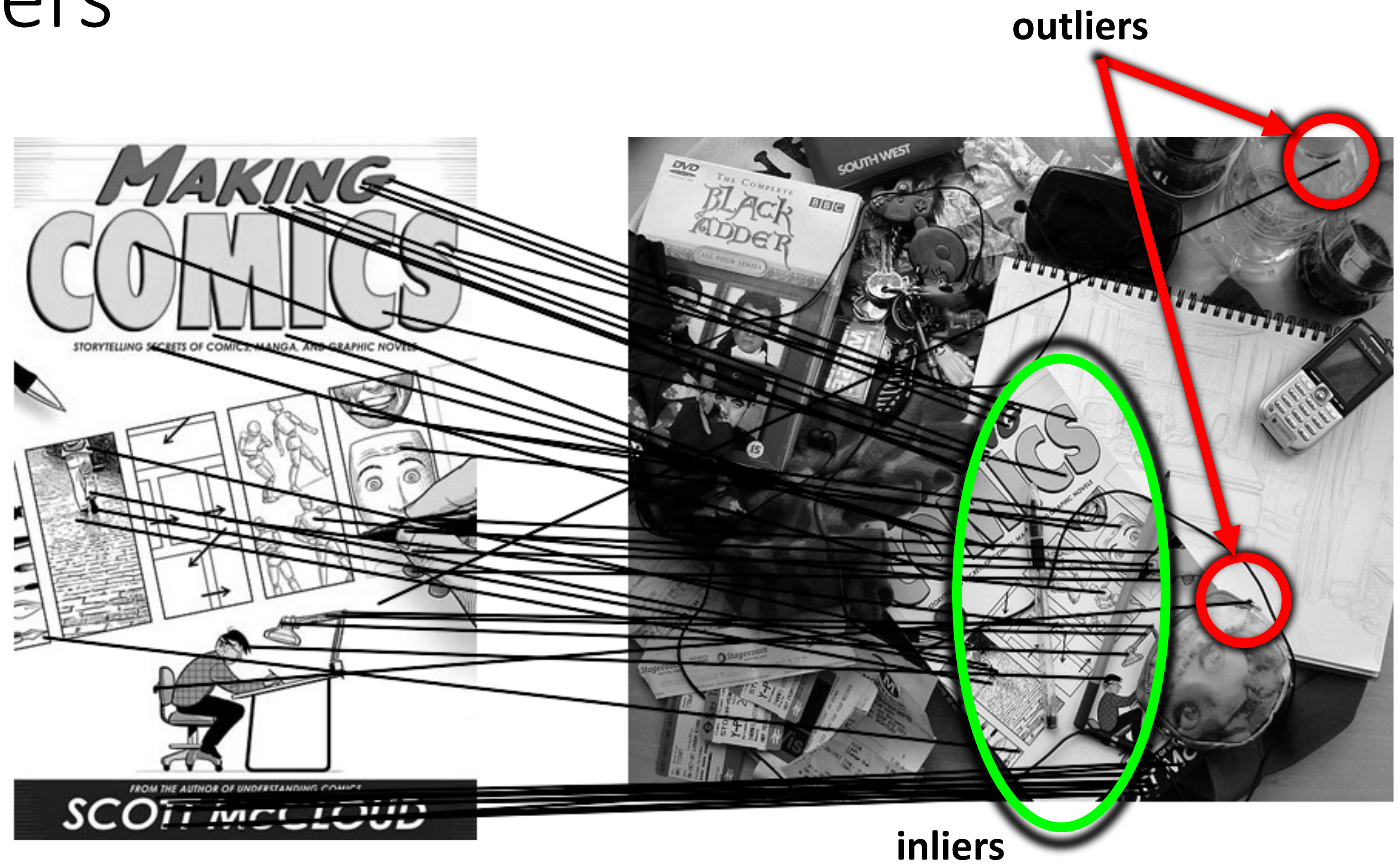


$$\text{Cost}(m, b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2$$

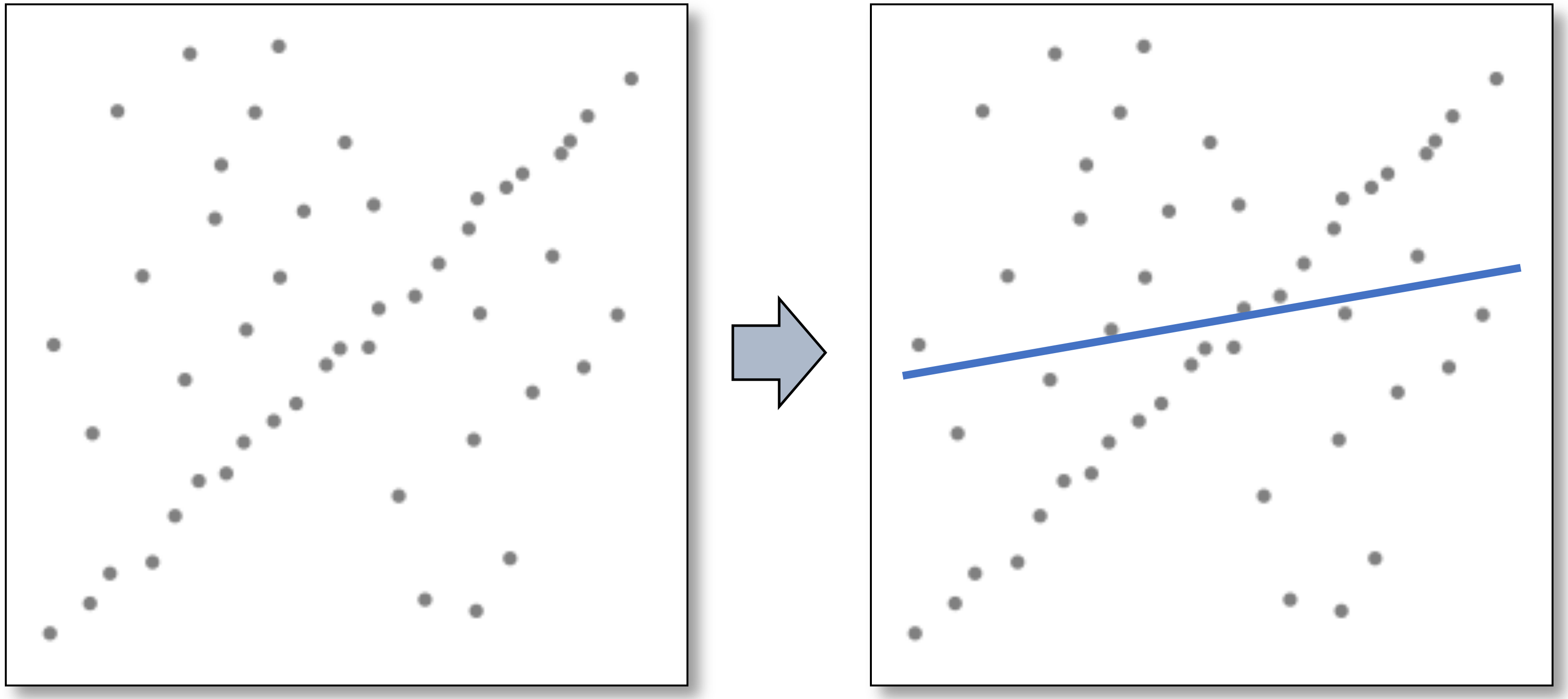
Linear regression

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Outliers


outliers
inliers
MAKING
COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
SCOTT McCLOUD
SOUTH WEST
DVD
THE COMPLETE
BLACK ADDER
BBC

Robustness



Problem: Fit a line to these datapoints

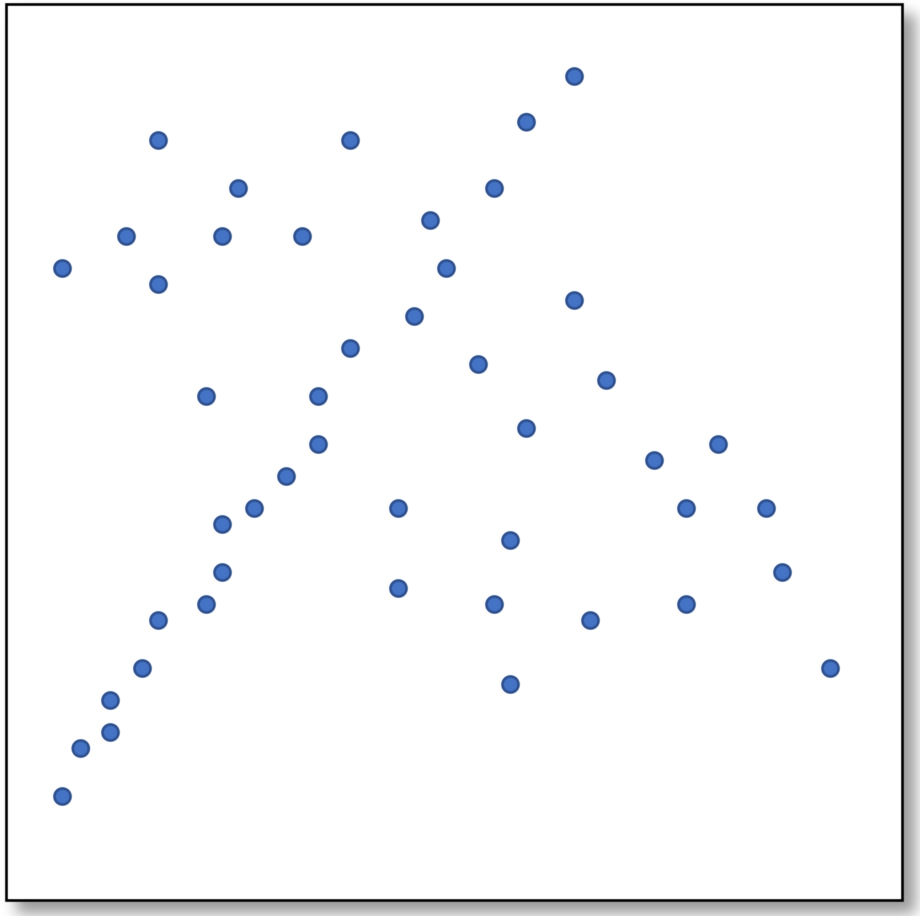
Least squares fit

Idea

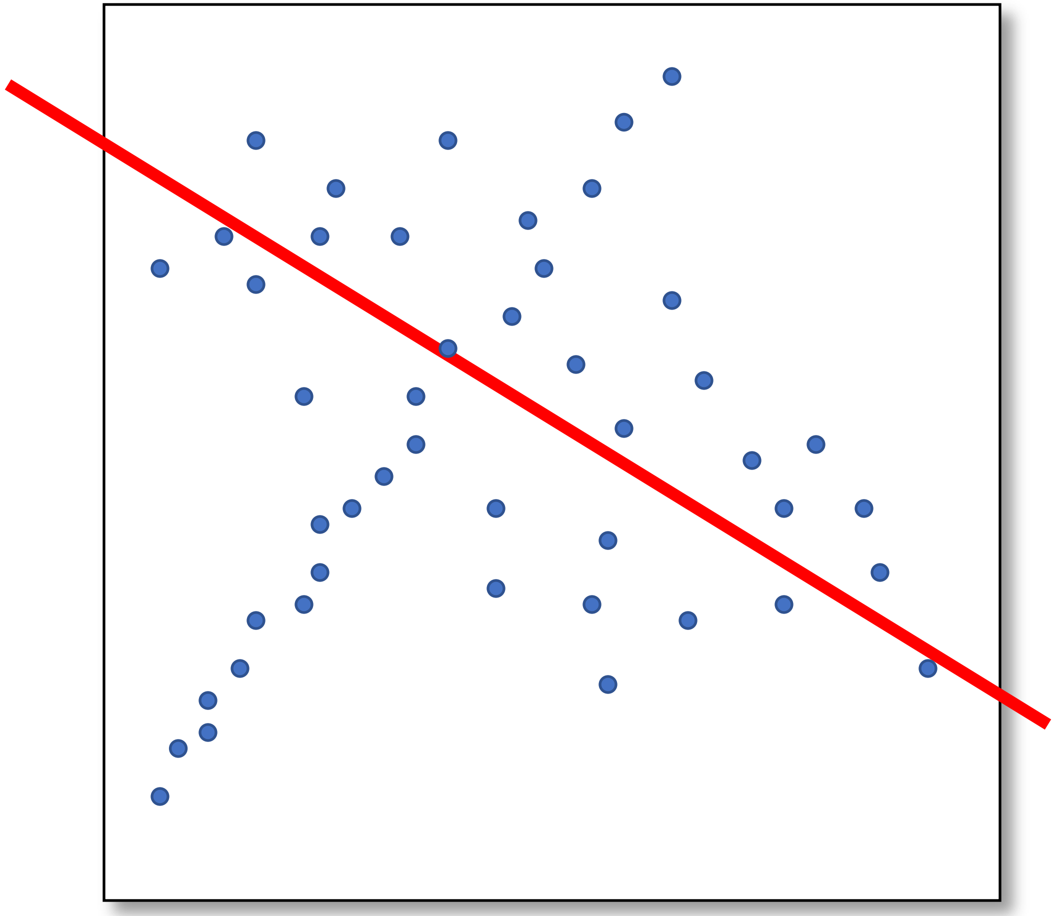
- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

Counting inliers

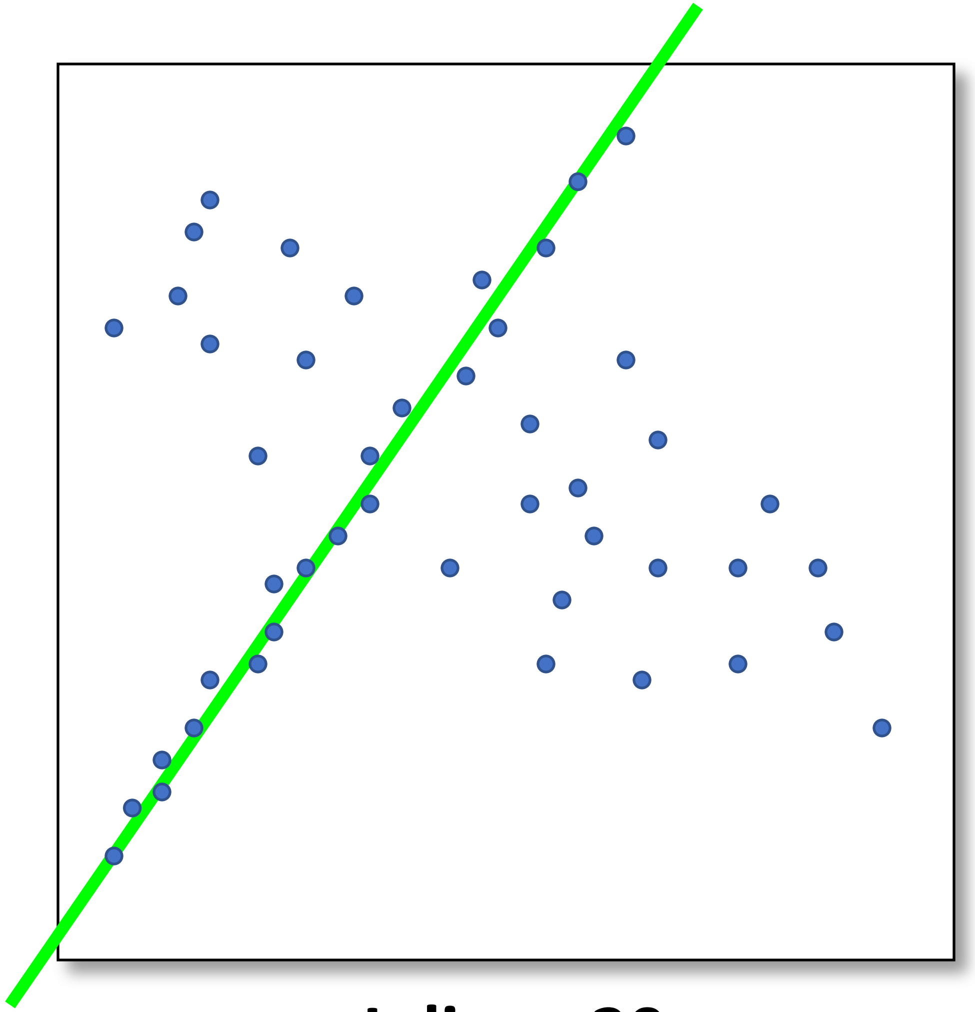


Counting inliers



Inliers: 3

Counting inliers



Inliers: 20

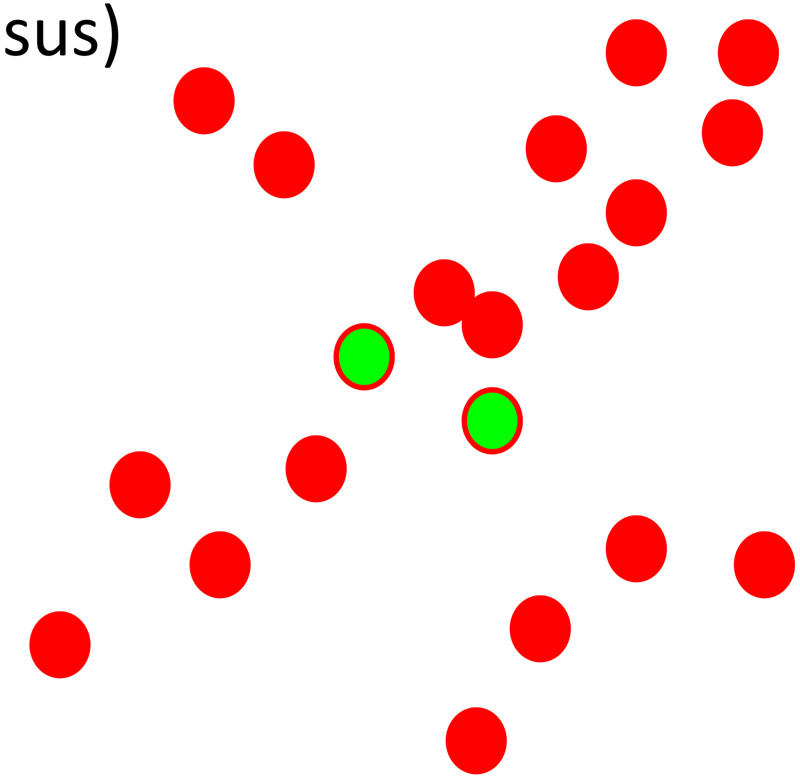
How do we find the best line?

- Unlike least-squares, no simple closed-form solution

- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

RANSAC (Random Sample Consensus)

Line fitting example



Algorithm:

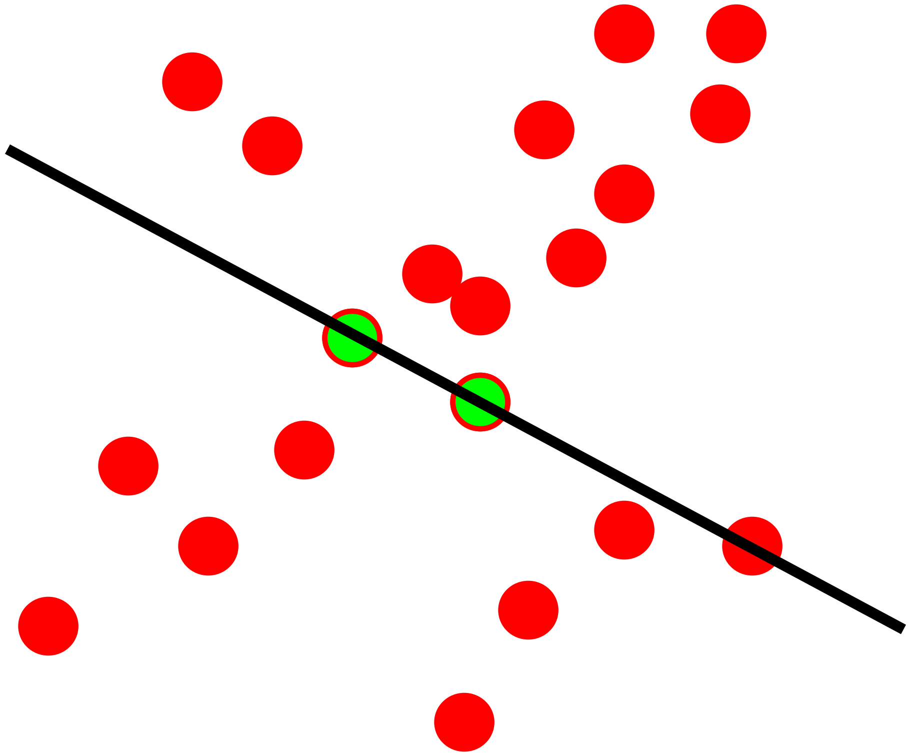
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example



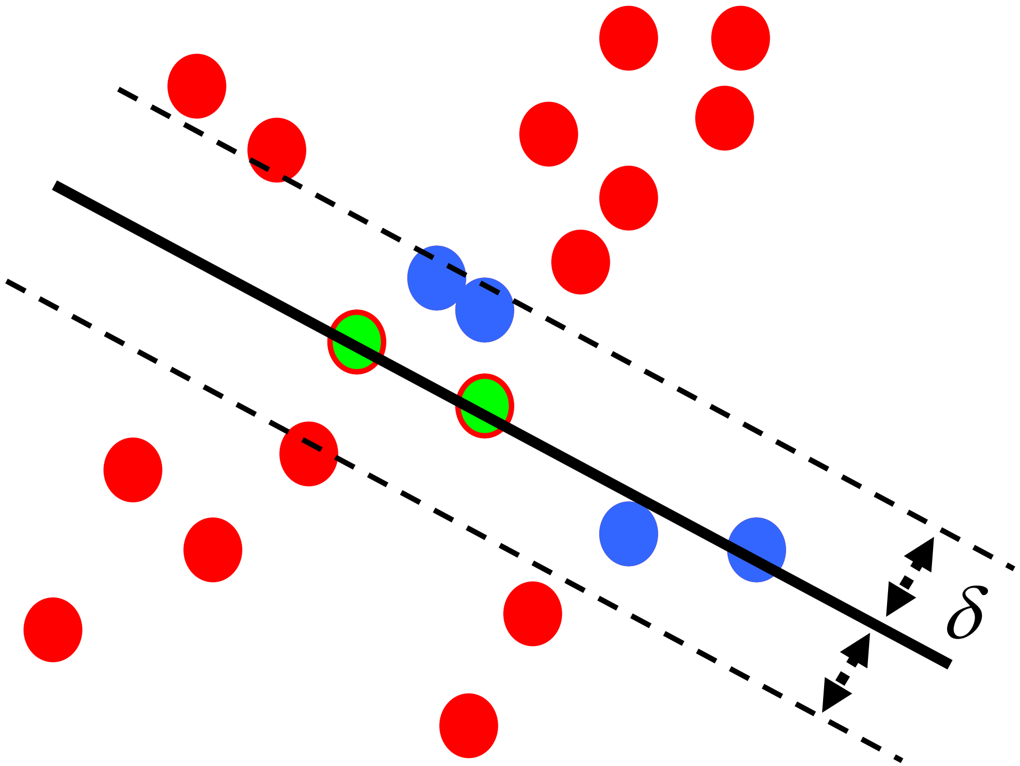
Algorithm:

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

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

RANSAC

Line fitting example



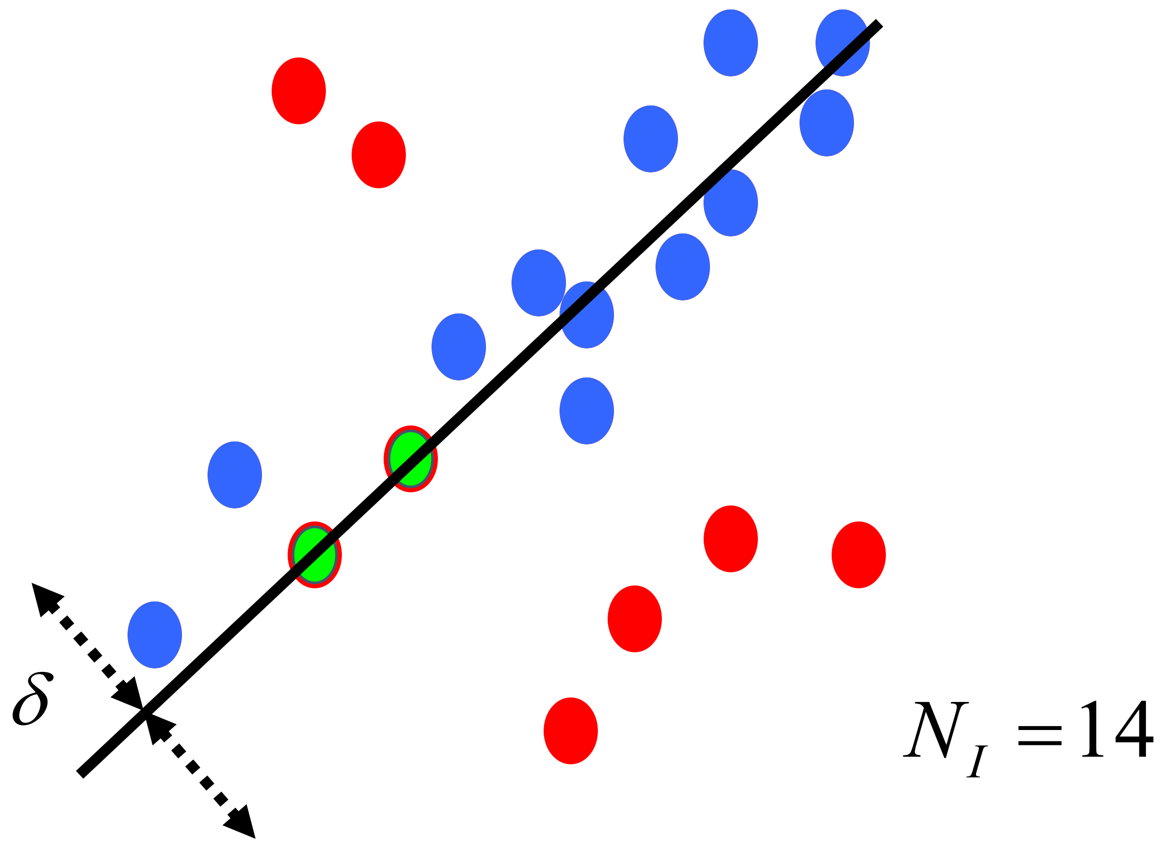
$$N_I = 6$$

Algorithm:

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

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

RANSAC



Algorithm:

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

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

RANSAC

- Idea:
  - All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
    - RANSAC only has guarantees if there are < 50% outliers

  - “All good matches are alike; every bad match is bad in its own way.”

– Tolstoy via Alyosha Efros

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
  - Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
  - How many rounds do we need?

How many rounds?

- If we have to choose *k* samples each time
  - with an inlier ratio p
  - and we want the right answer with probability *P*

	proportion of inliers *p*						
k	95%	90%	80%	75%	70%	60%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

$P = 0.99$

Source: M. Pollefeys

To ensure that the random sampling has a good chance of finding a true set of inliers, a sufficient number of trials S must be tried. Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials. The likelihood in one trial that all k random samples are inliers is p^k. Therefore, the likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S \tag{6.29}$$

and the required minimum number of trials is

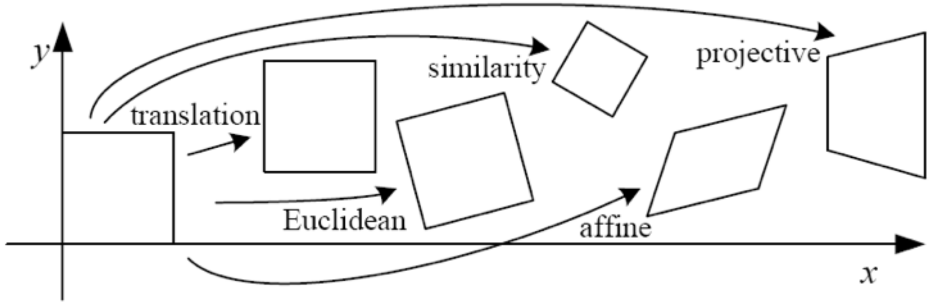
$$S = \frac{\log(1 - P)}{\log(1 - p^k)}. \tag{6.30}$$

	proportion of inliers *p*						
k	95%	90%	80%	75%	70%	60%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

$P = 0.99$

How big is *k*?

- For alignment, depends on the motion model
  - Here, each sample is a correspondence (pair of matching points)



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

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Parameters to tune
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios

RANSAC

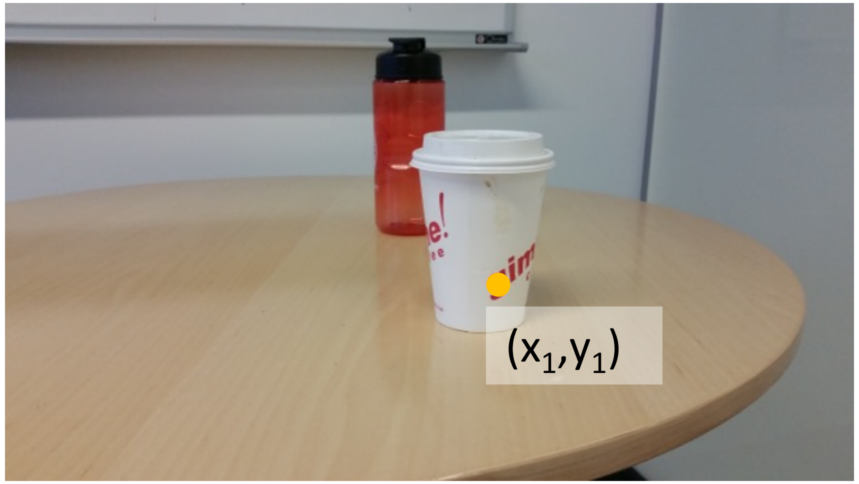
- An example of a “voting”-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins

- There are many other types of voting schemes
  - E.g., Hough transforms…

Triangulation

- Suppose we have two cameras
  - Calibrated: parameters known
- And a pair of corresponding pixels
- Find 3D location of point!

$P^{(1)}$



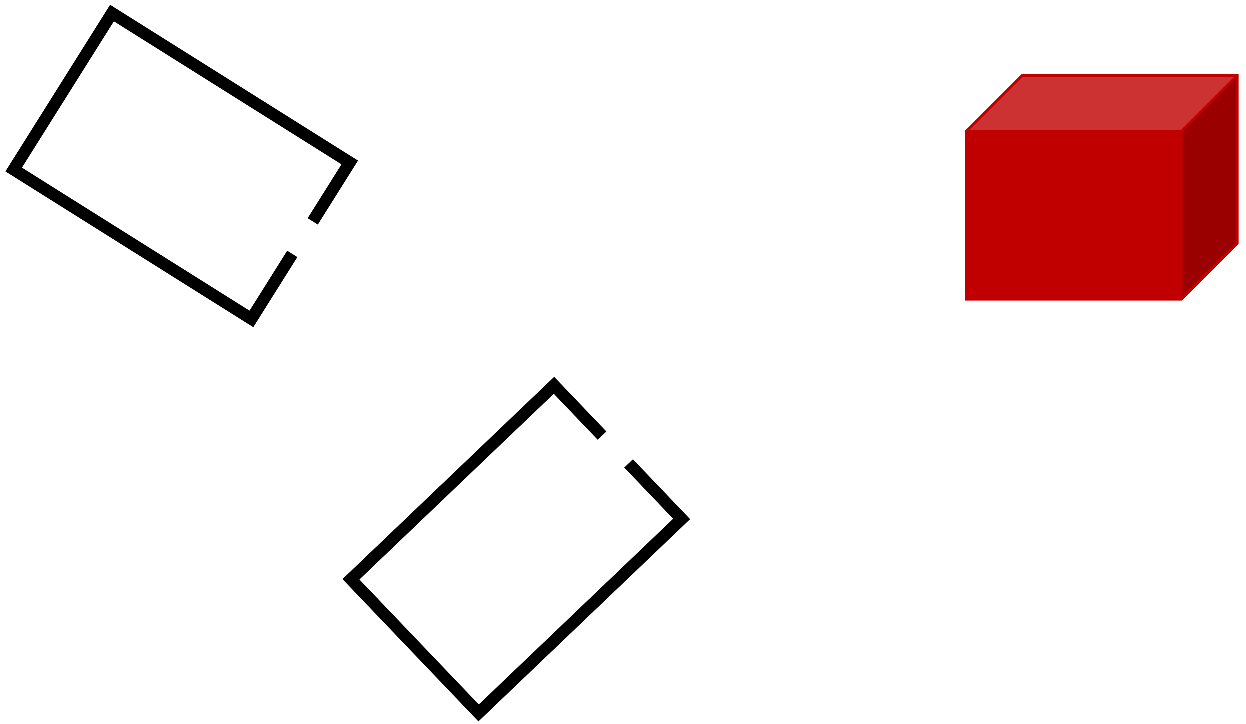
$P^{(2)}$

Binocular stereo

- Given two *calibrated* cameras
    - Find pairs of corresponding pixels
    - Use corresponding image locations to set up equations on world coordinates
    - Solve!

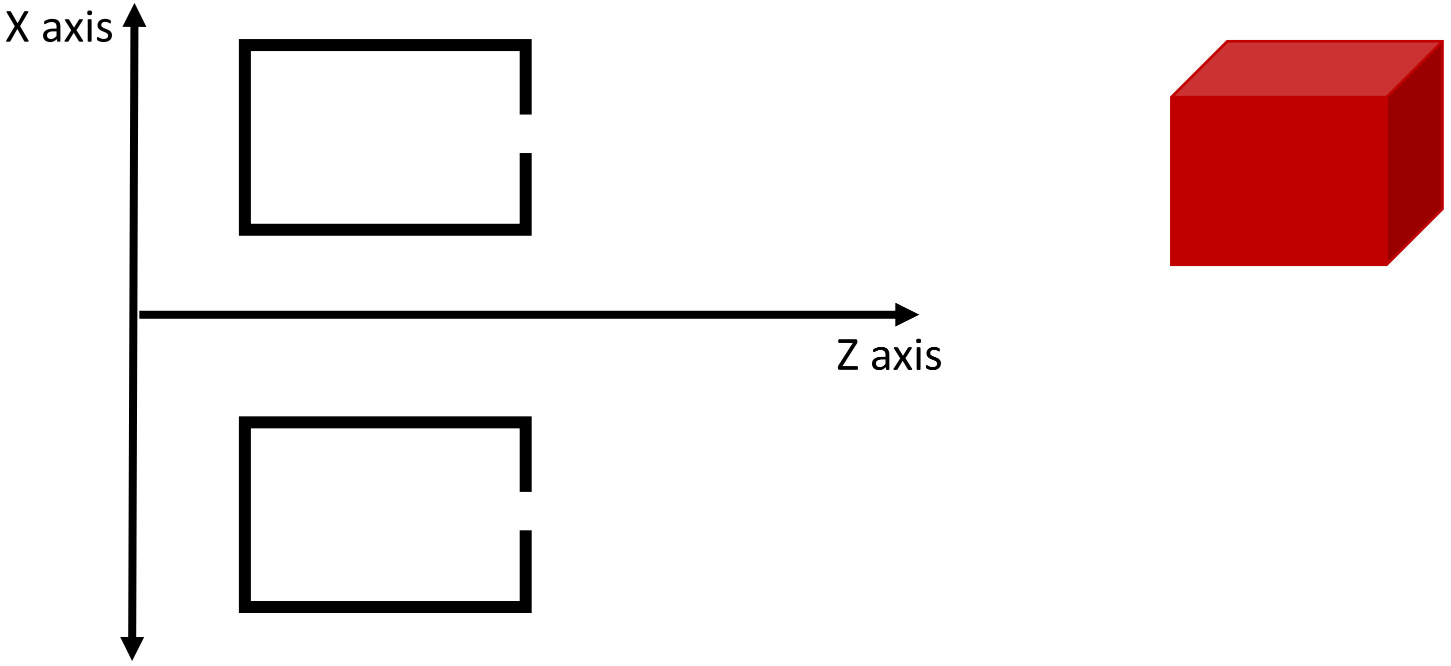
Binocular stereo

- General case: cameras can be arbitrary locations and orientations



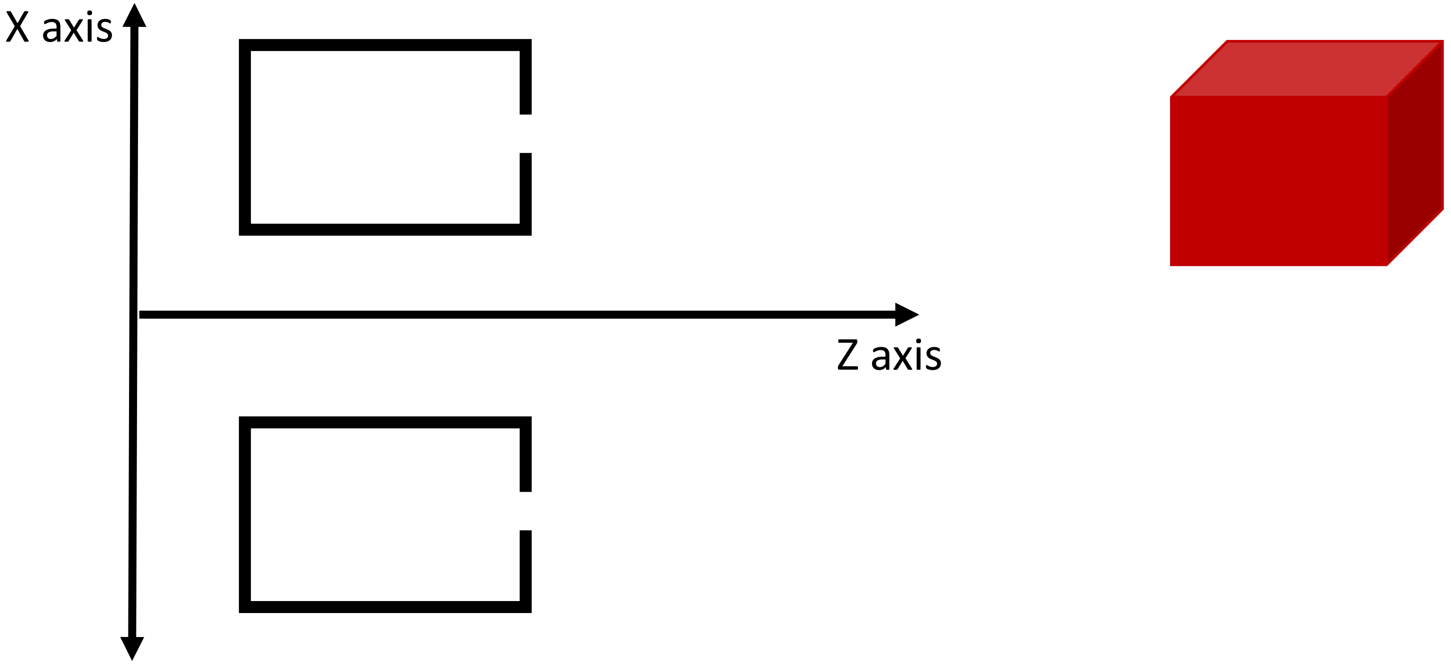
Binocular stereo

- Special case: cameras are parallel to each other and translated along X axis

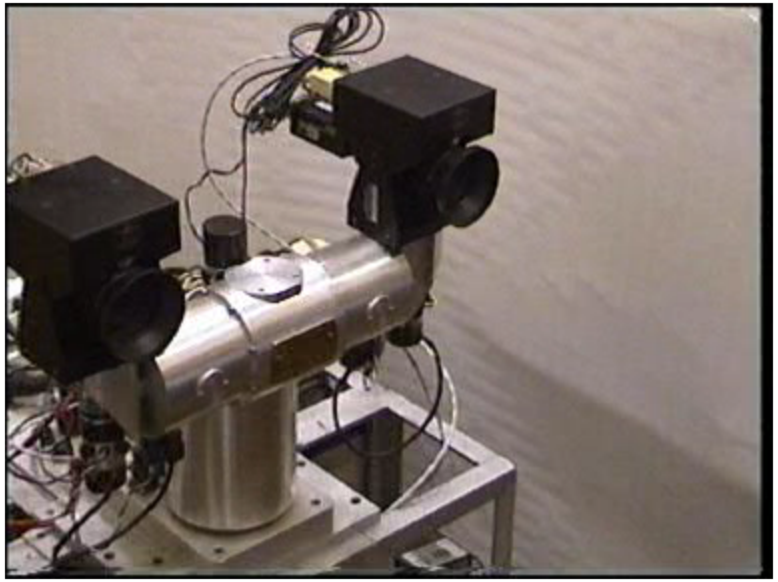


Stereo with *rectified cameras*

- Special case: cameras are parallel to each other and translated along X axis



Stereo head



Kinect / depth cameras

Stereo with “rectified cameras”

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} I & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\mathbf{t} = \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} I & \mathbf{t} \end{bmatrix} \vec{\mathbf{x}}_w$$

$$\mathbf{t} = \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix} \quad \vec{\mathbf{x}}_w = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix} = \mathbf{x}_w = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} I & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ 1 \end{bmatrix} = \mathbf{x}_w + \mathbf{t} = \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\vec{\mathbf{x}}_{img}^{(1)} \equiv \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\vec{\mathbf{x}}_{img}^{(2)} \equiv \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

$$\begin{bmatrix} \lambda x_1 \\ \lambda y_1 \\ \lambda \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} \lambda x_2 \\ \lambda y_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} X + t_x \\ Y \\ Z \end{bmatrix}$$

Perspective projection in rectified cameras

- Without loss of generality, assume origin is at pinhole of 1st camera

X coordinate differs by t_x/Z

$$x_1 = \frac{X}{Z} \qquad x_2 = \frac{X + t_x}{Z}$$

$$y_1 = \frac{Y}{Z} \qquad y_2 = \frac{Y}{Z}$$

Y coordinate is the same!

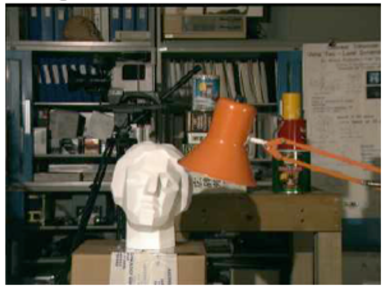
Perspective projection in rectified cameras

- X coordinate differs by t_x/Z
- That is, difference in X coordinate is *inversely proportional to depth*
- Difference in X coordinate is called *disparity*
- Translation between cameras (tx) is called *baseline*

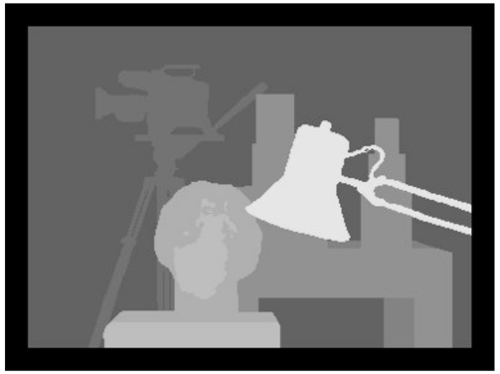
- *disparity = baseline / depth*

The disparity image

- For pixel (x,y) in one image, only need to know disparity to get correspondence
- Create an image with color at (x,y) = disparity



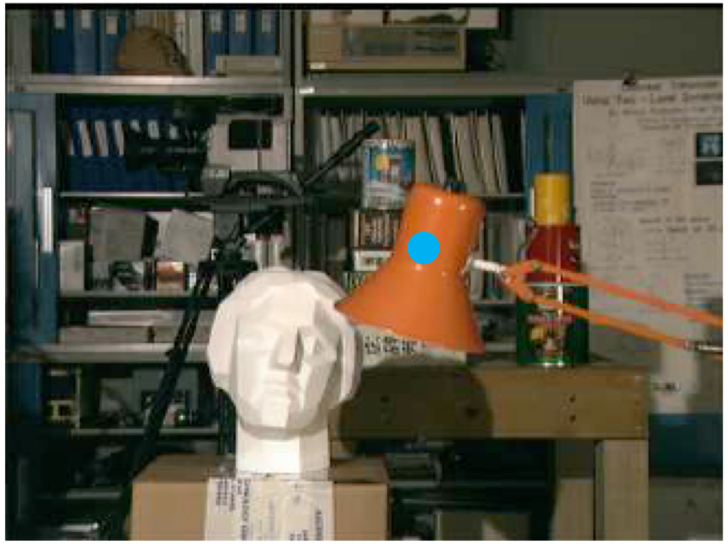
right image

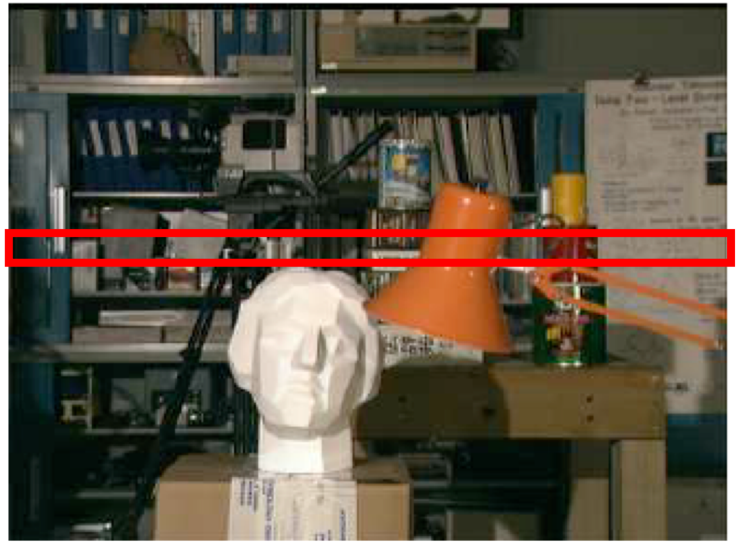


disparity

left image

Perspective projection in rectified cameras



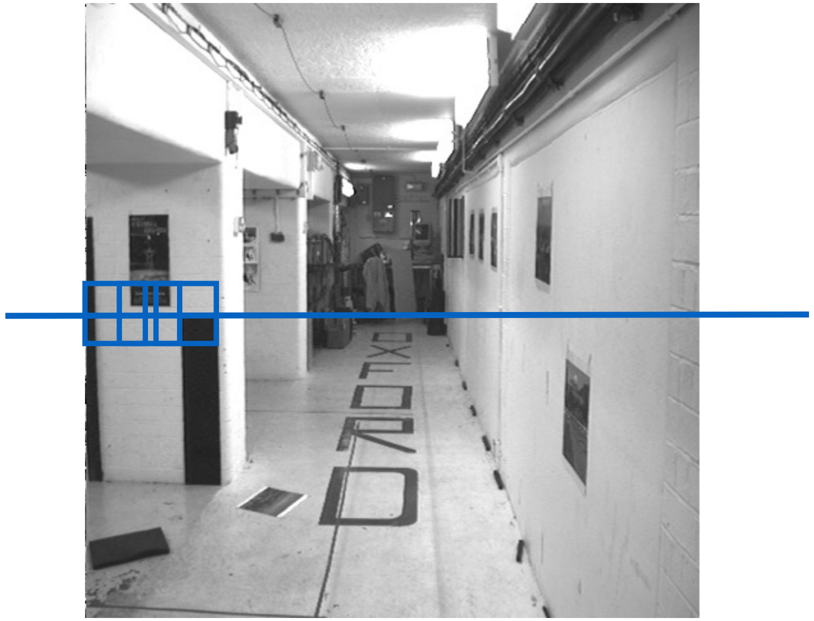


- For rectified cameras, correspondence problem is easier
- Only requires searching along a particular *row.*

NCC - Normalized Cross Correlation

- Lighting and color change pixel intensities
- Example: increase brightness / contrast
- $I' = \alpha I + \beta$
- Subtract patch mean: invariance to β
- Divide by norm of vector: invariance to α
- $x' = x - < x >$
- $x'' = \frac{x'}{||x'||}$
- *similarity* = $x'' \cdot y''$

Cross-correlation of neighborhood



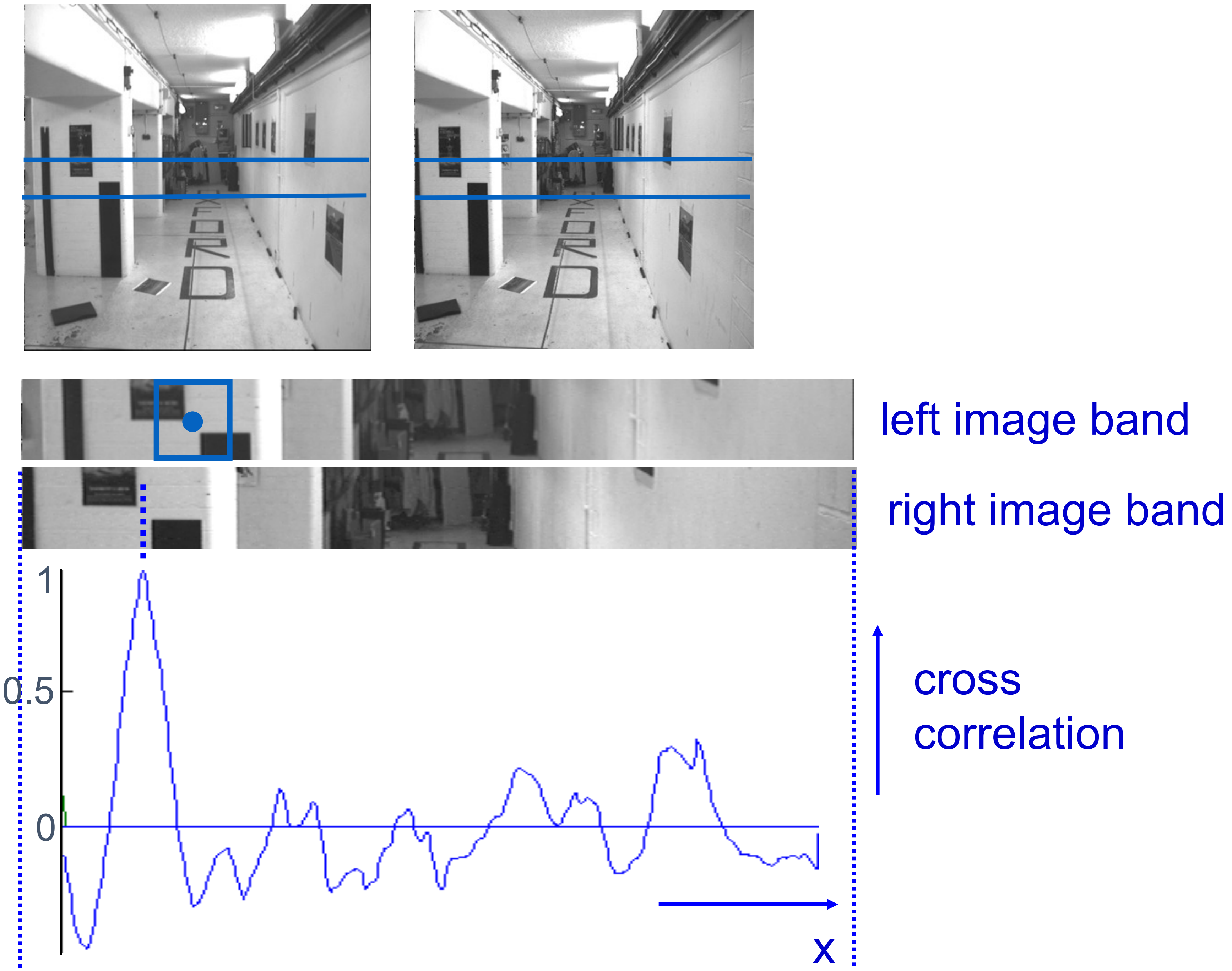
regions $\mathtt{A}, \mathtt{B}$, write as vectors $\mathbf{a}, \mathbf{b}$

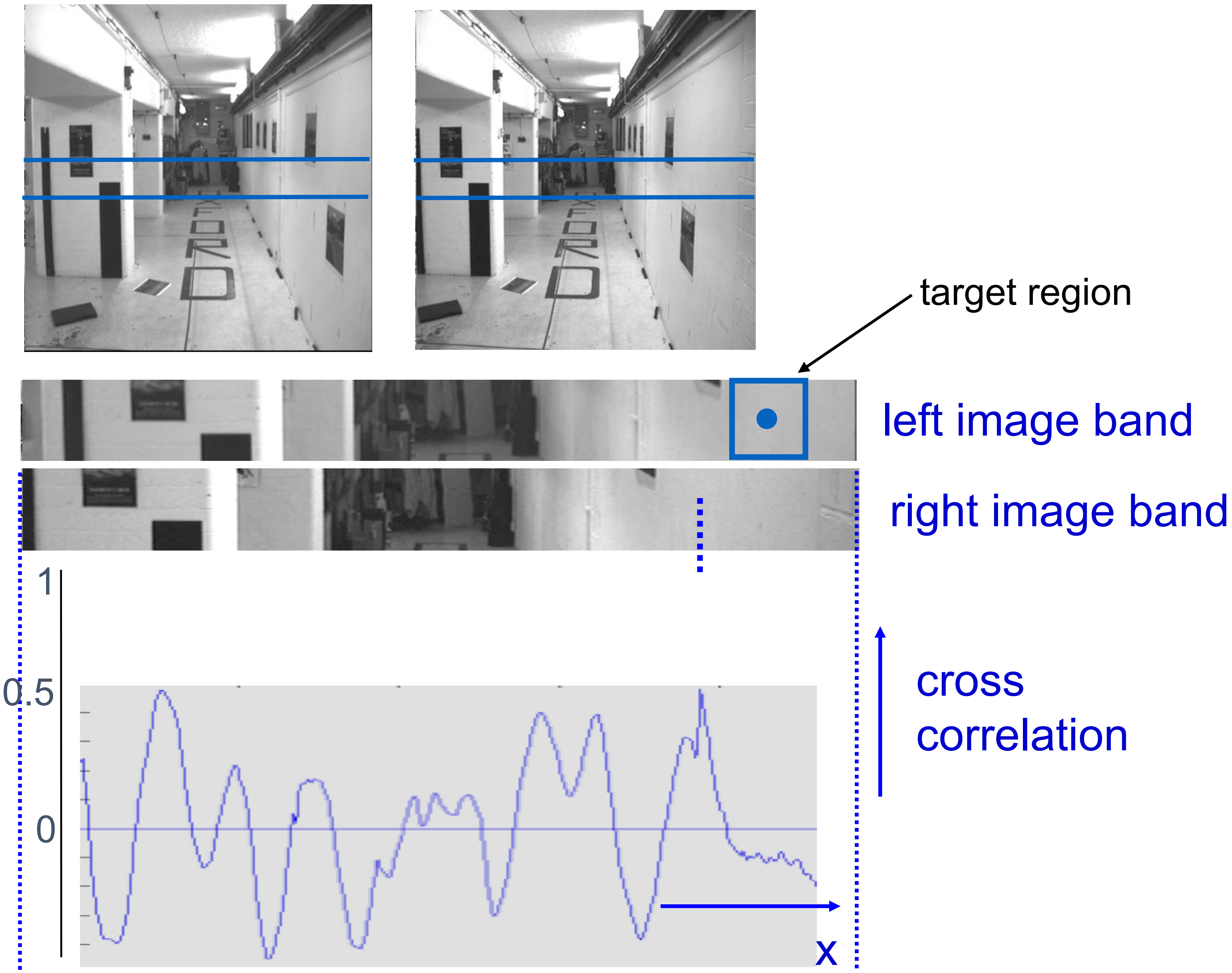
translate so that mean is zero

$$\mathtt{a} \rightarrow \mathtt{a} - \langle \mathbf{a} \rangle, \quad \mathtt{b} \rightarrow \mathtt{b} - \langle \mathbf{b} \rangle$$

$$\text{cross correlation} = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$

Invariant to $I \rightarrow \alpha I + \beta$


left image band
right image band
1
0.5
0
cross
correlation
x



target region
left image band
right image band
1
0.5
0
cross
correlation
x

The NCC cost volume

- Consider M x N image
- Suppose there are D possible disparities.
- For every pixel, D possible scores
- Can be written as an M x N x D array
- To get disparity, take max along 3rd axis

Computing the NCC volume

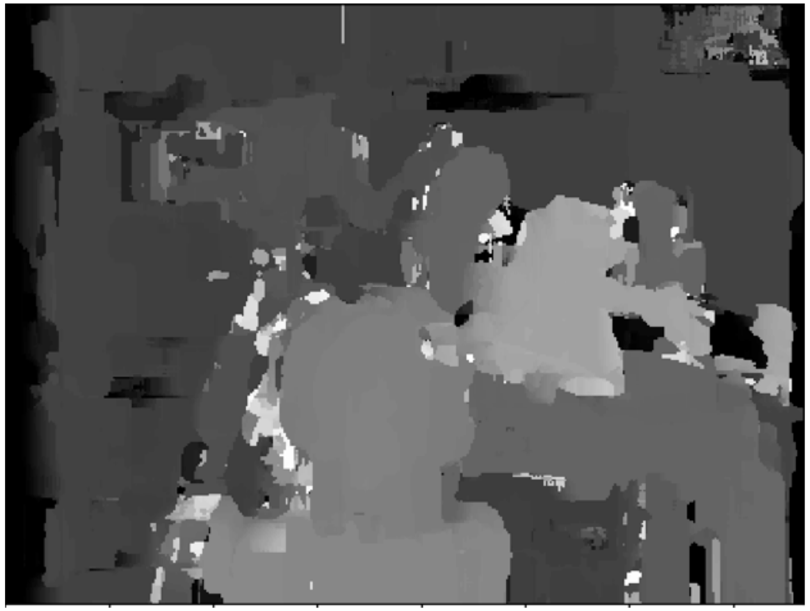
1. For every pixel (x, y)
    1. For every disparity d
        1. Get normalized patch from image 1 at (x, y)
        2. Get normalized patch from image 2 at (x + d, y)
        3. Compute NCC

Computing the NCC volume

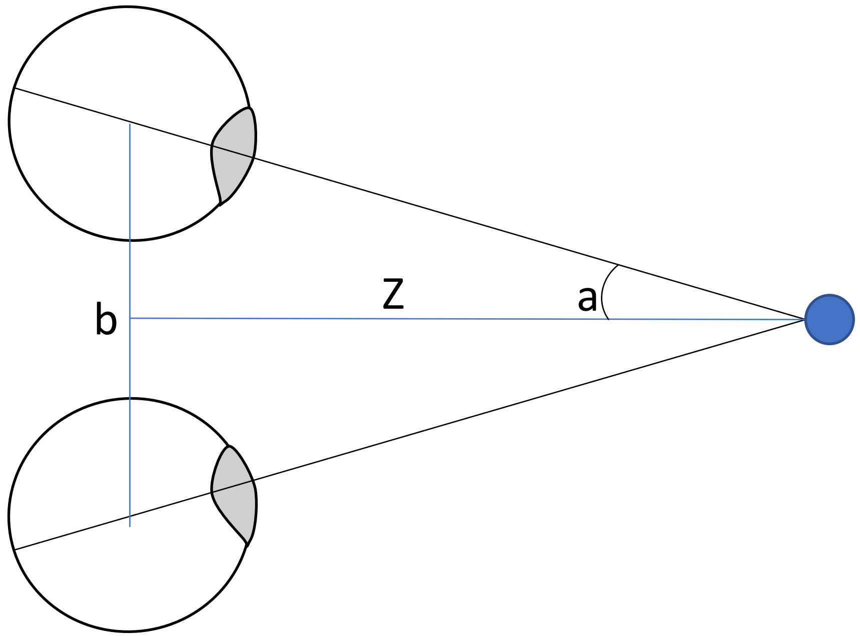
1. For every disparity d
   1. For every pixel (x, y)
      1. Get normalized patch from image 1 at (x, y)
      2. Get normalized patch from image 2 at (x + d, y)
      3. Compute NCC

Assume all pixels lie at same disparity d (i.e., lie on same plane) and compute cost for each

Plane sweep stereo



A similar special case

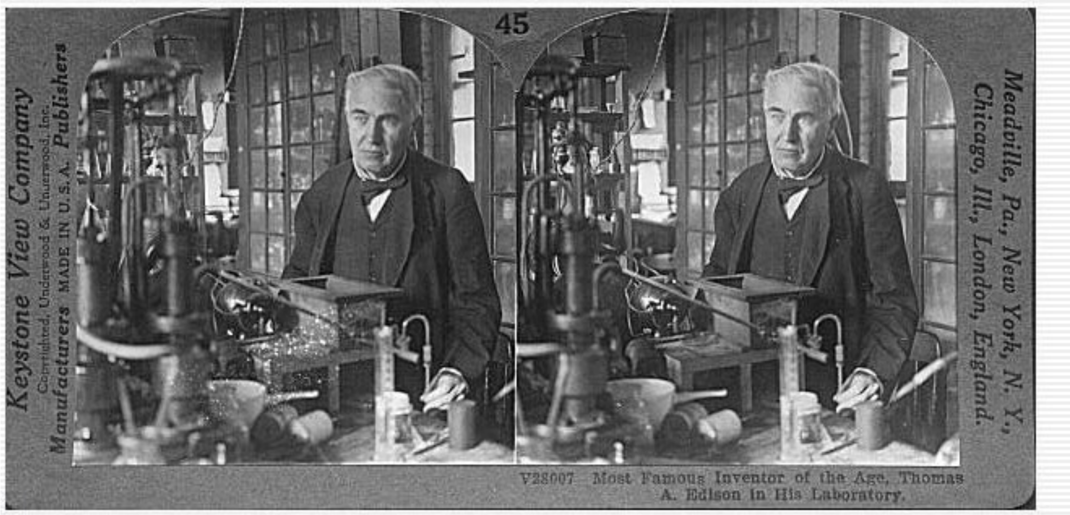


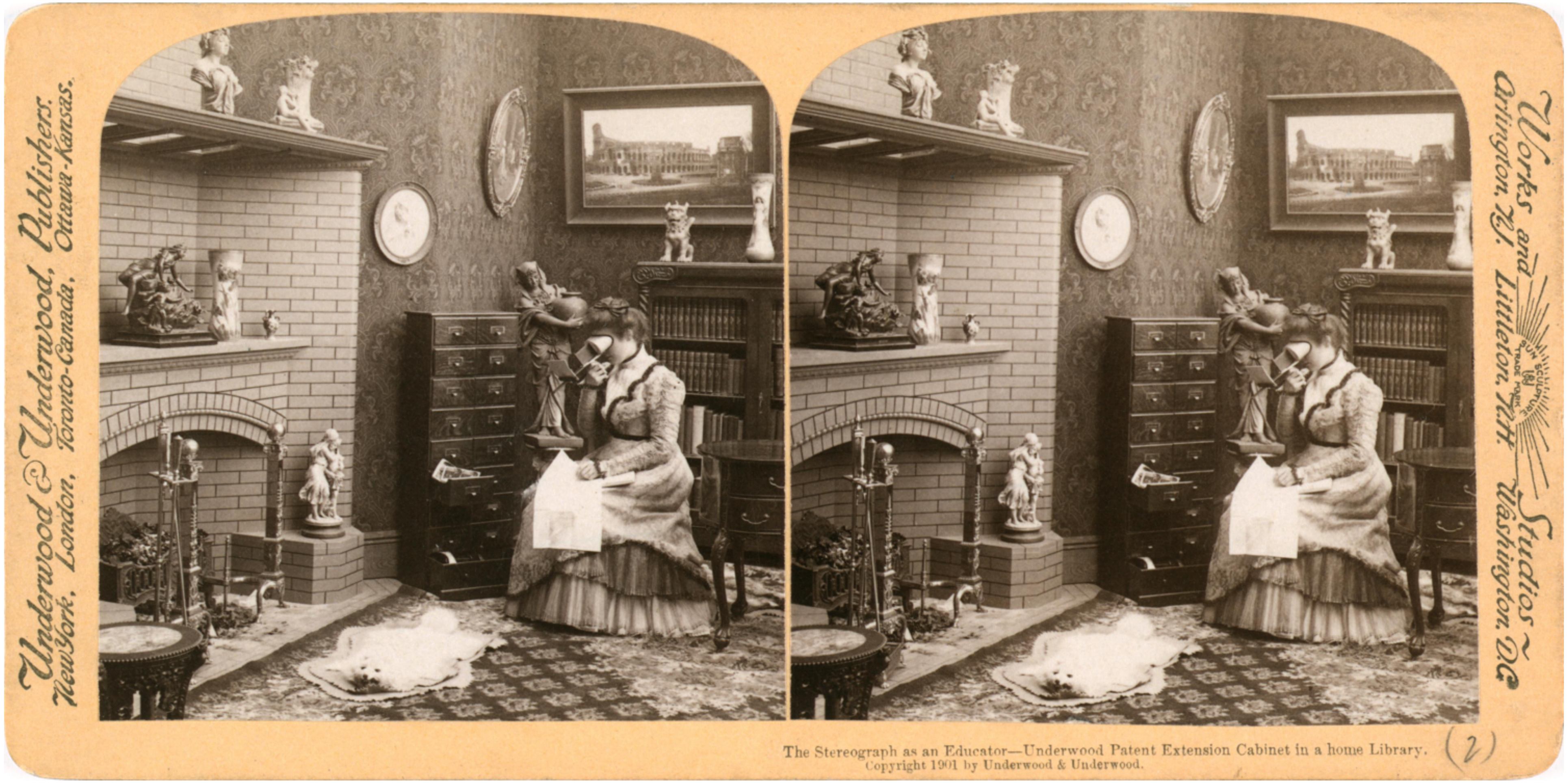
$$\tan a = \frac{b}{2Z}$$

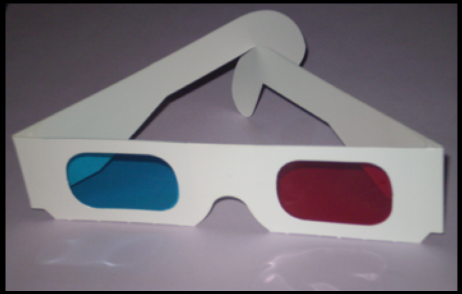
- *Fixating* camera system
- Eyes fixate on object
- Angle at which they merge proportional to inverse depth

Stereograms

- Invented by Sir Charles Wheatstone, 1838




Underwood & Underwood, Publishers.
New York, London, Toronto-Canada, Ottawa-Kansas.
Works and Studios ~
Arlington, N.J. Littleton, N.H. Washington, D.C.
SUN SCULPTURE TRADE MARK
The Stereograph as an Educator—Underwood Patent Extension Cabinet in a home Library.
Copyright 1901 by Underwood & Underwood.
(2)



Mark Twain at Pool Table", no date, UCR Museum of Photography