3D Vision: Coordinate Spaces

A lot of slides from Noah Snavely +
Shree Nayar's YT series: First principals of Computer Vision

CS180: Intro to Computer Vision and Comp. Photo
Angjoo Kanazawa & Alexei Efros, UC Berkeley, Fall 2023

House keeping

Project 4 Part 1 due tomorrow

Final Project

Easy path: Pre-canned

- Group of 1 : 2 projects
- Group of 2 : 3 projects

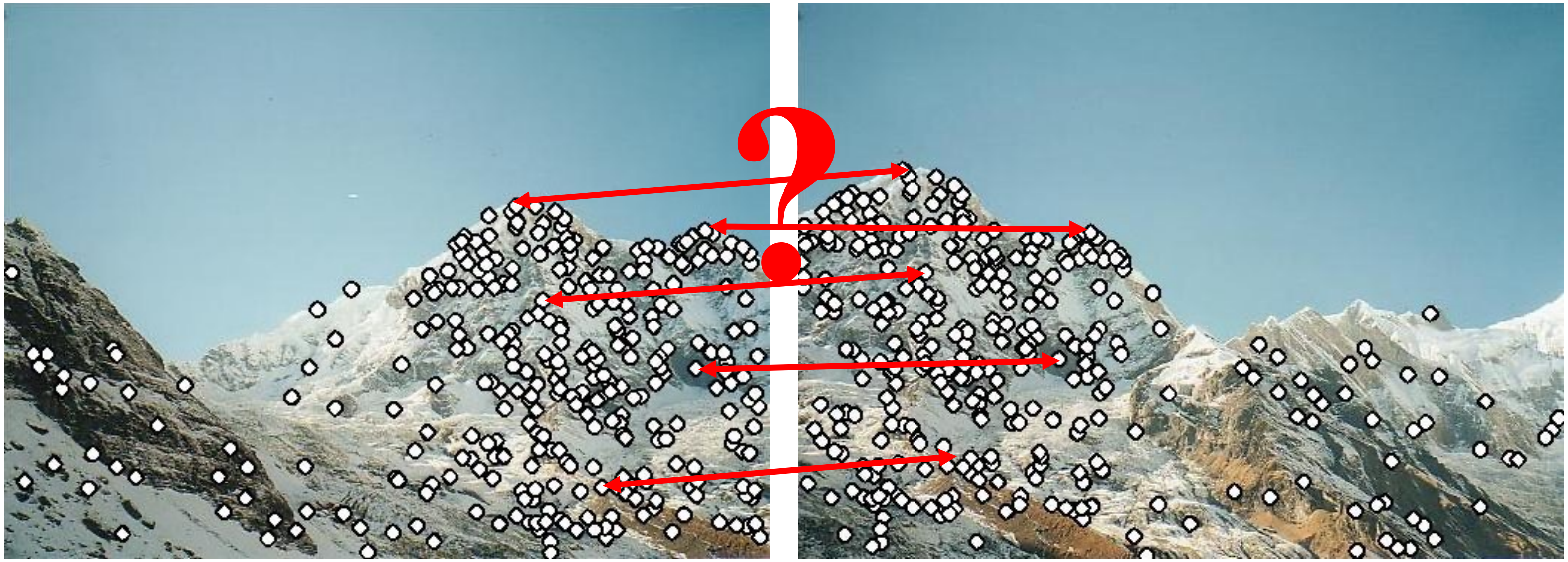
Grad students: Your own project

- 1 page Proposal with pictures **due 11/9**

Recap: Feature descriptors

We know how to detect points

Next question: **How to match them?**



Point descriptor should be:

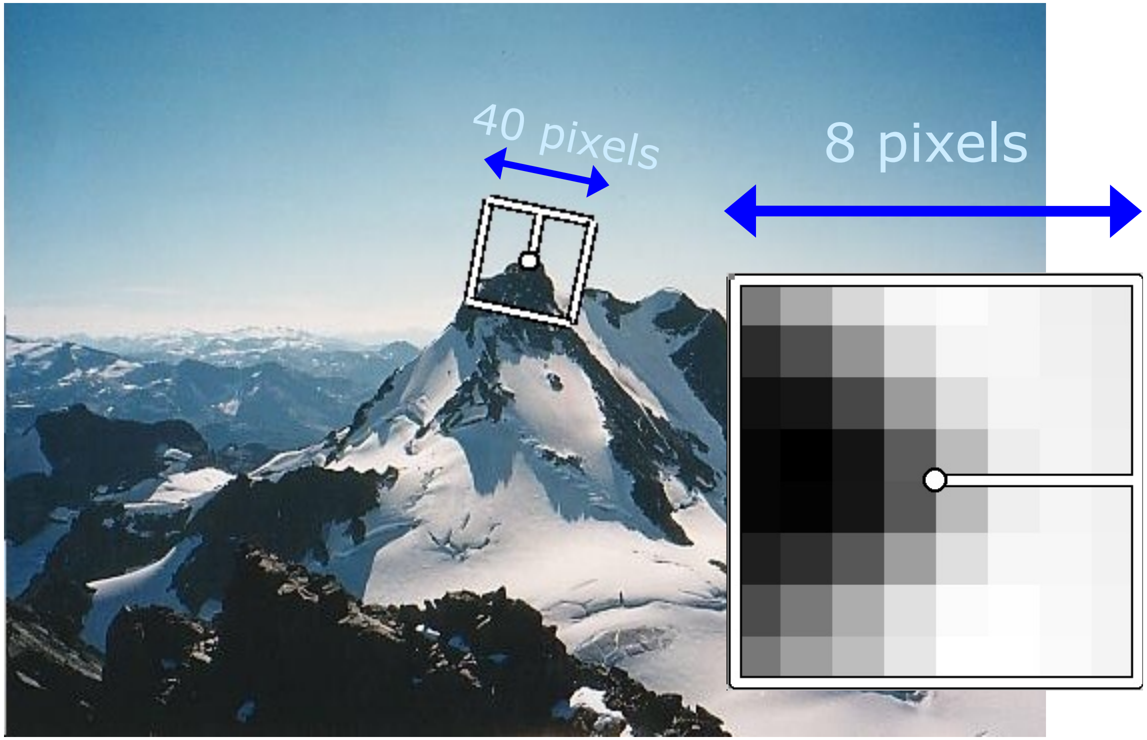
1. Invariant
2. Distinctive

MOPS descriptor vector

8x8 oriented patch

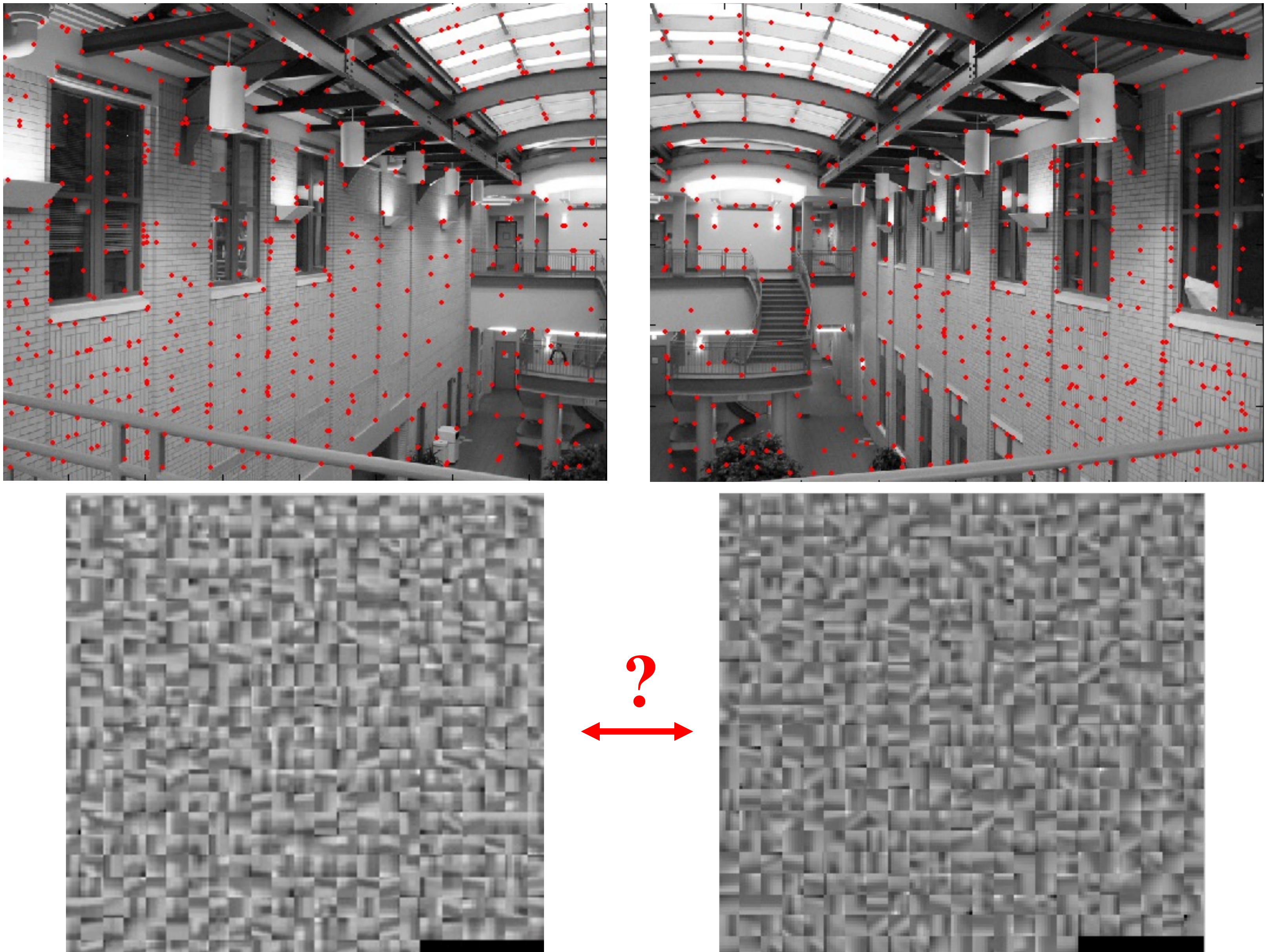
- Sampled at 5 x scale

Bias/gain normalisation: $I' = (I - \mu)/\sigma$



Automatic Feature Matching

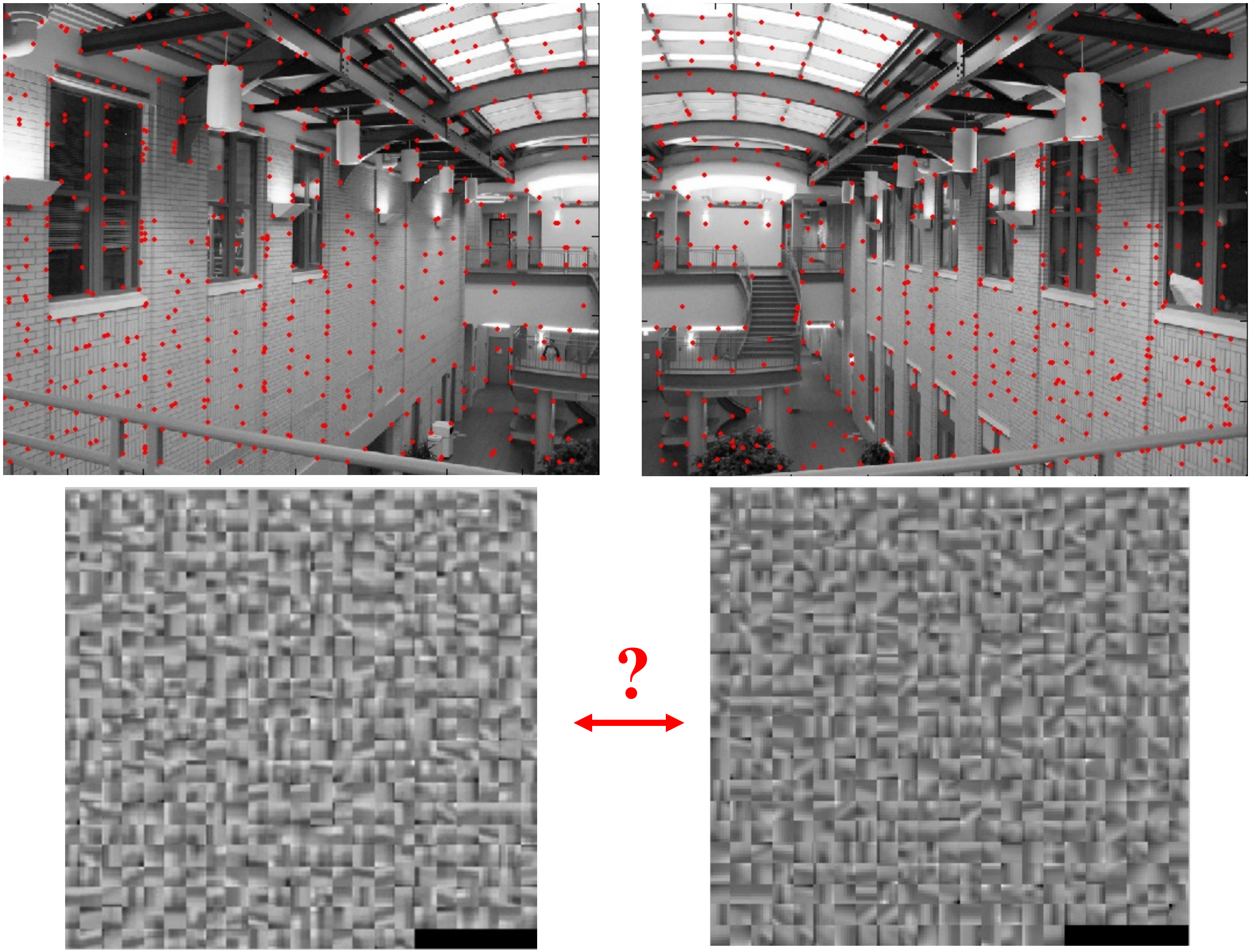
Feature matching



Feature matching

- Pick best match!
  - For every patch in image 1, find the most similar patch (e.g. by SSD).
  - Called “nearest neighbor” in machine learning
- Can do various speed ups:
  - Hashing
    - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
  - Fast Nearest neighbor techniques
    - *kd*-trees and their variants
  - Clustering / Vector quantization
    - So called “visual words”

What about outliers?

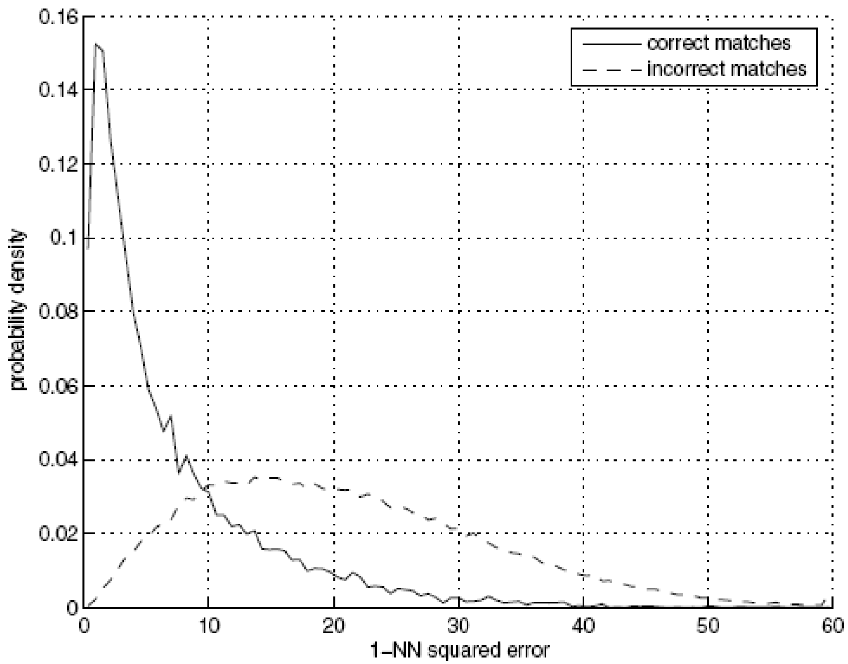


Feature-space outlier rejection

Let’s not match all features, but only these that have “similar enough” matches?

How can we do it?

- SSD(patch1,patch2) < threshold
- How to set threshold?



Feature-space outlier rejection: symmetry

Let's not match all features, but only these that have "similar enough" matches?

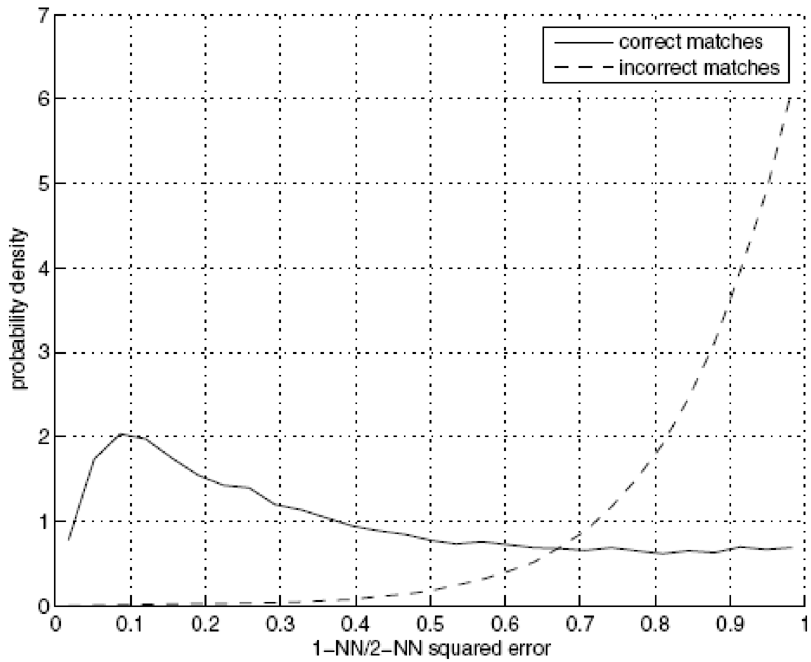
How can we do it?

- Symmetry: x's NN is y, and y's NN is x

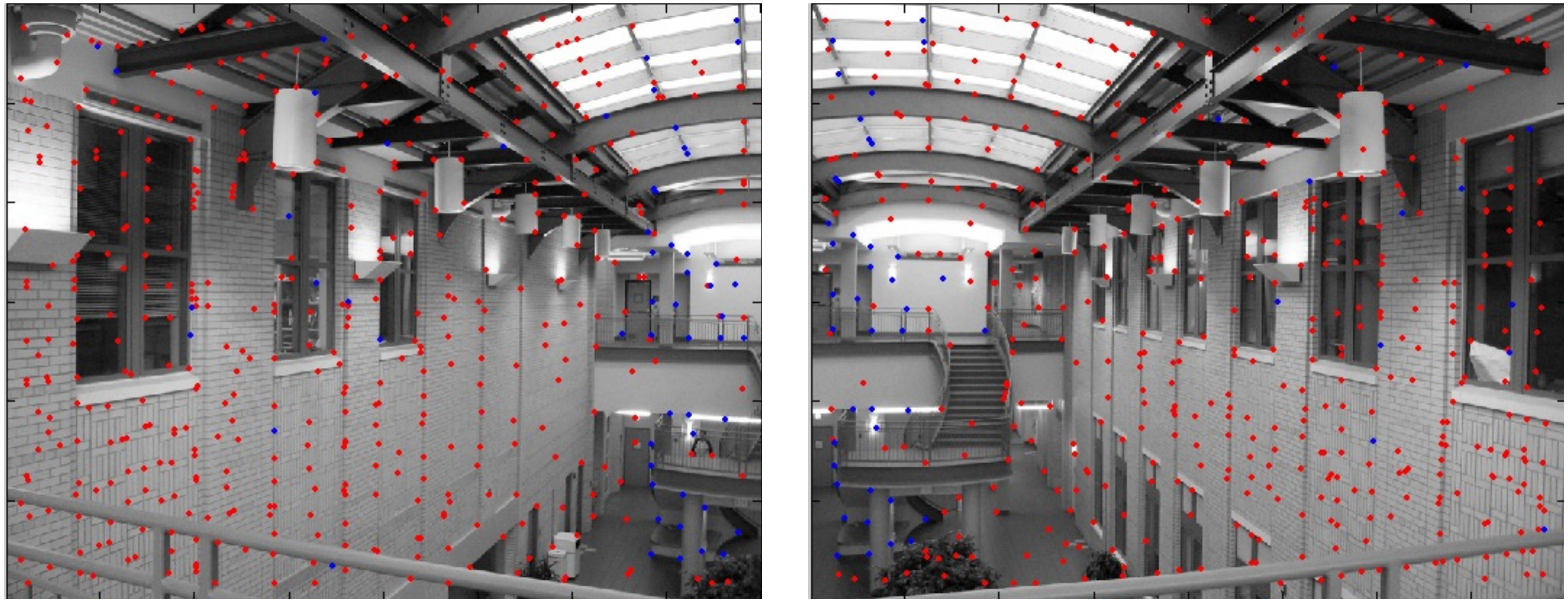
Feature-space outlier rejection: Lowe's trick

A better way [Lowe, 1999]:

- 1-NN: SSD of the closest match
- 2-NN: SSD of the <u>second-closest</u> match
- Look at how much better 1-NN is than 2-NN, e.g. 1-NN/2-NN
- That is, is our best match so much better than the rest?



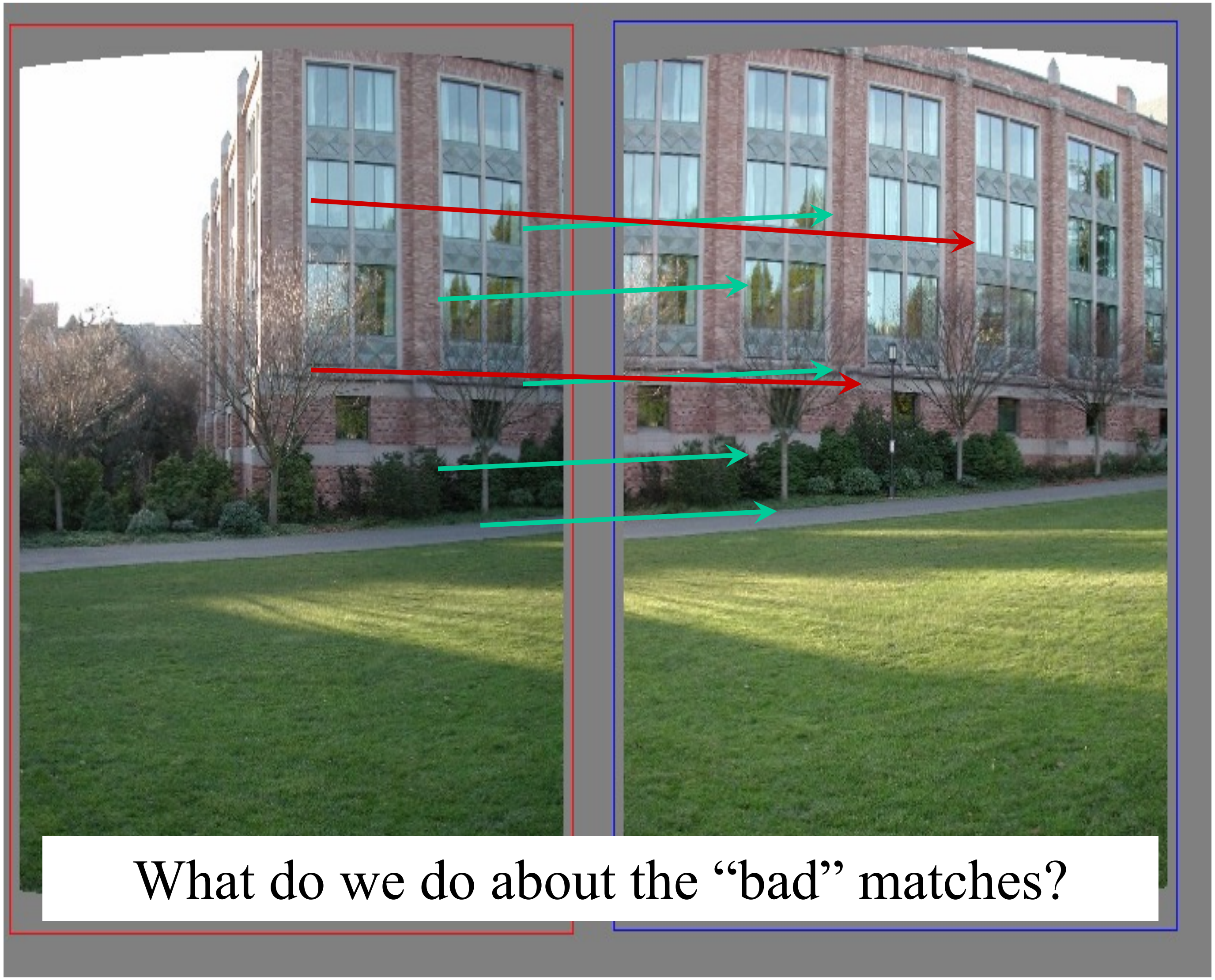
Feature-space outliner rejection



Can we now compute H from the blue points?

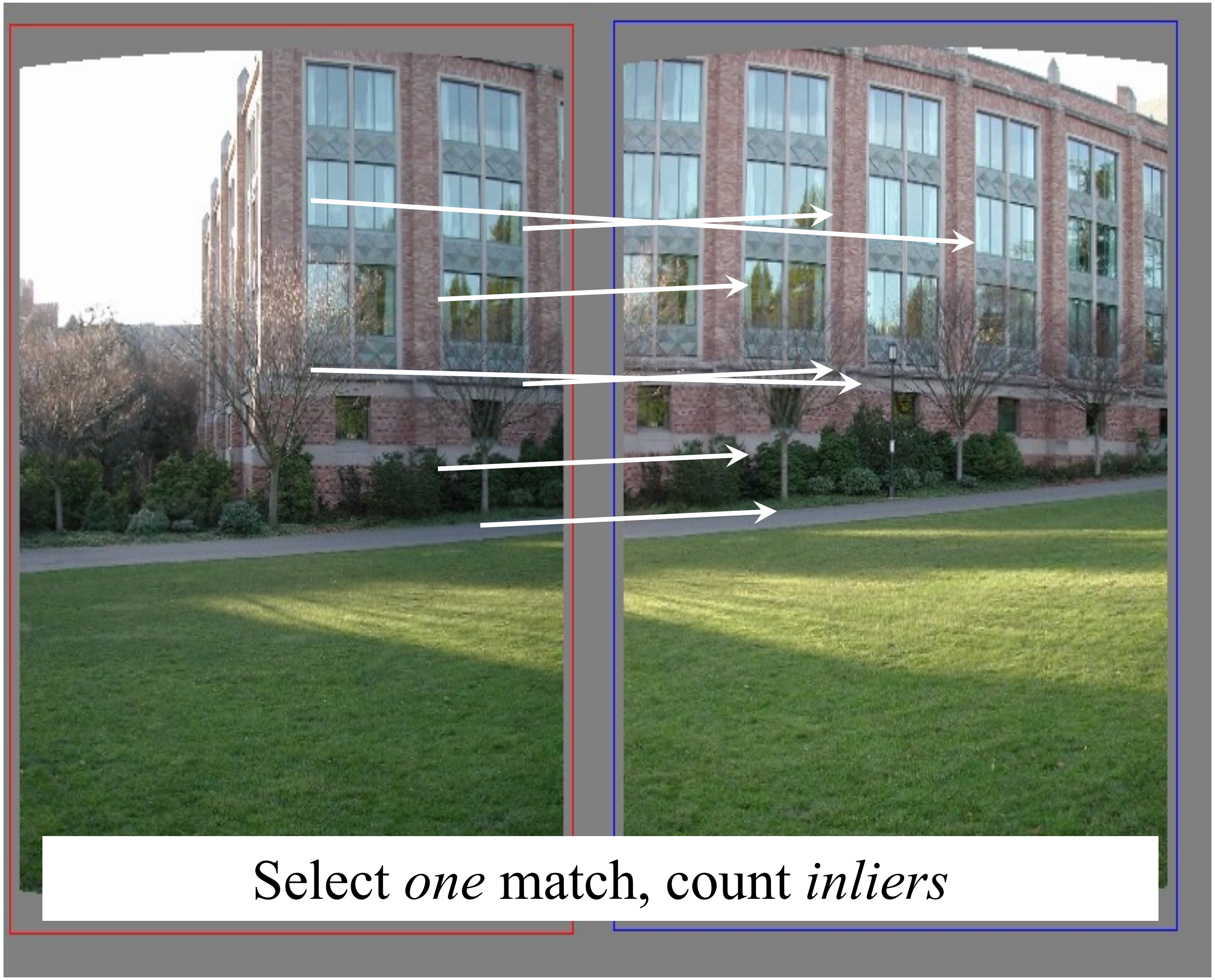
- No! Still too many outliers…
- What can we do?

Matching features



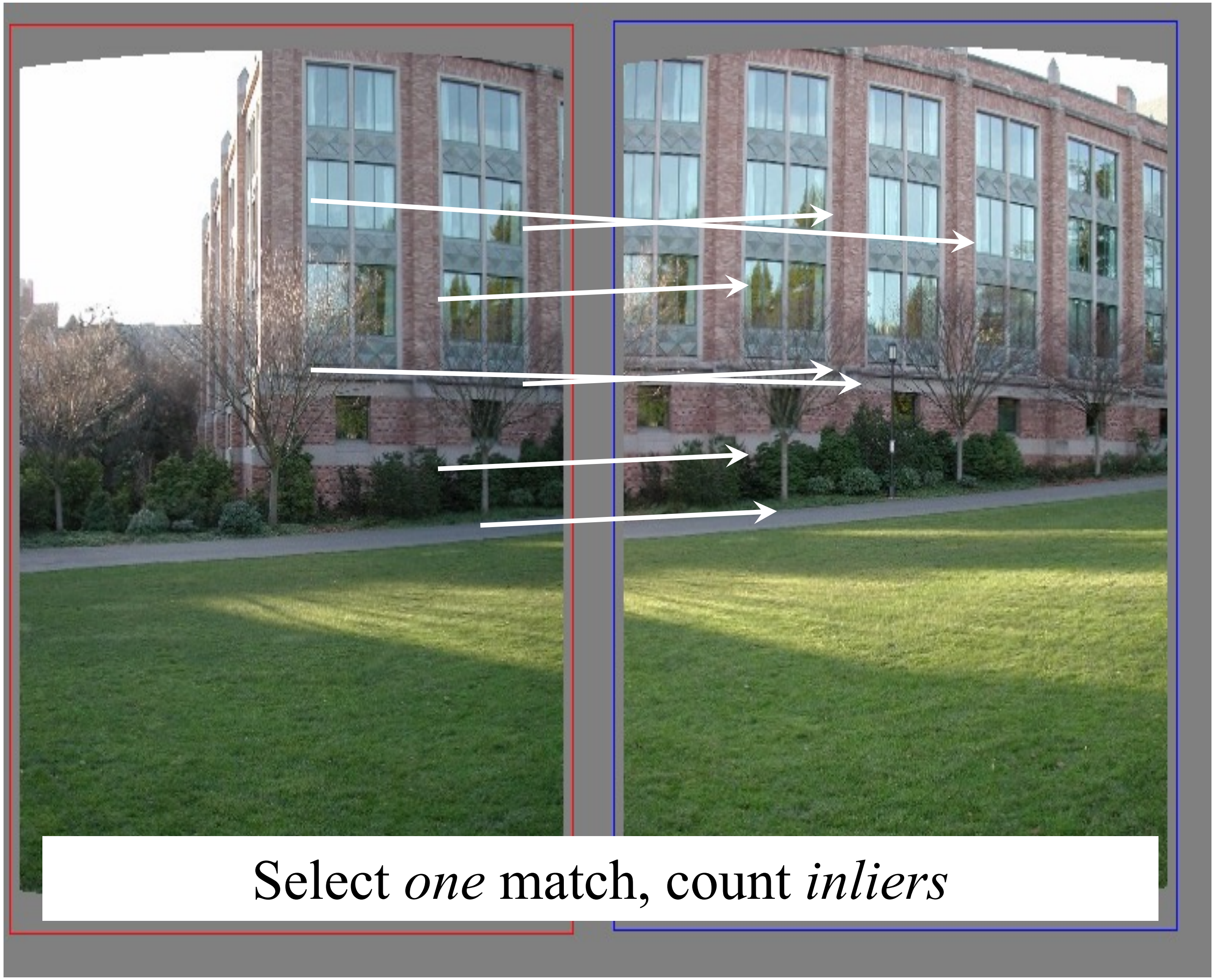
What do we do about the “bad” matches?

RAndom SAmple Consensus



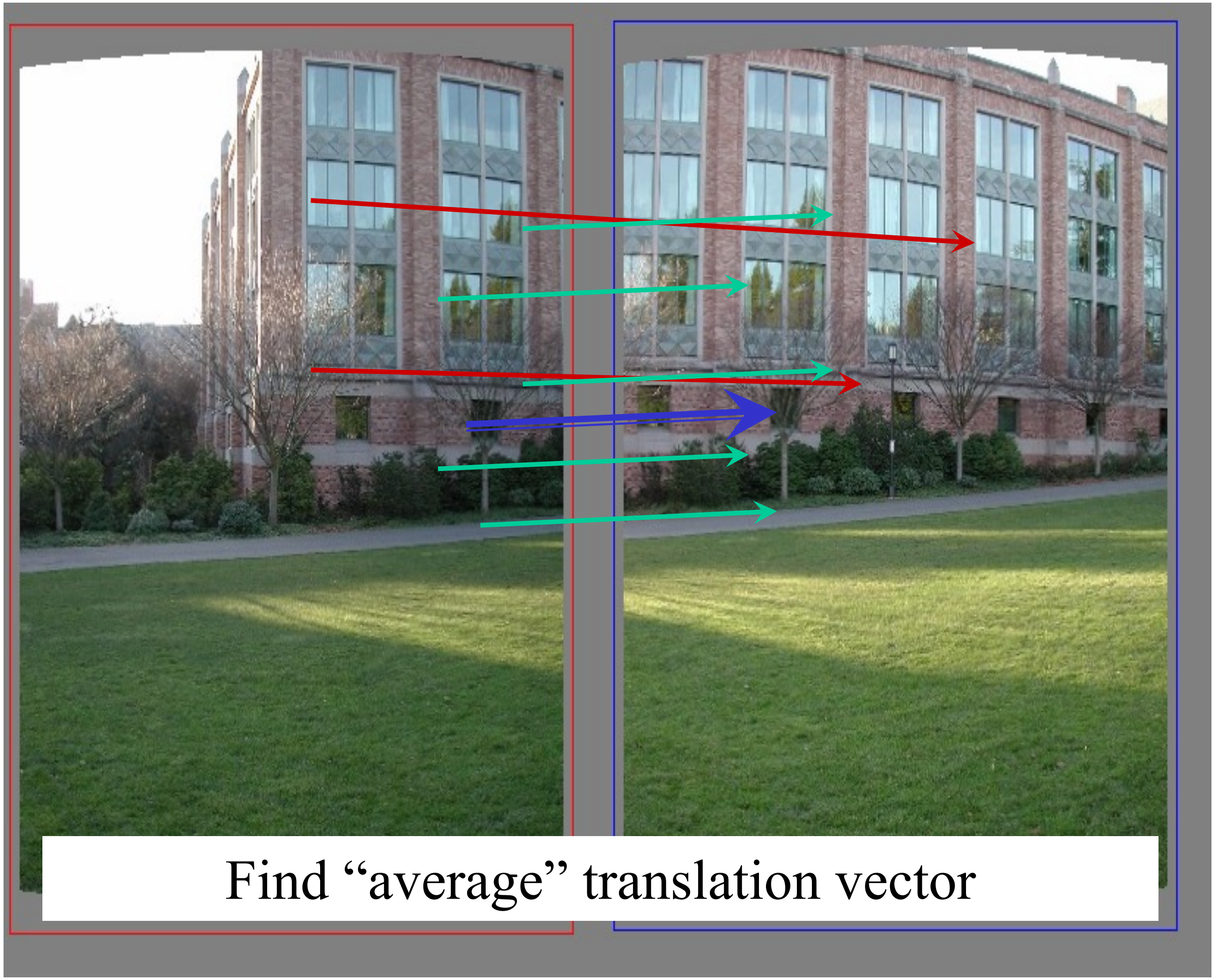
Select *one* match, count *inliers*

RAndom SAmple Consensus



Select *one* match, count *inliers*

Least squares fit

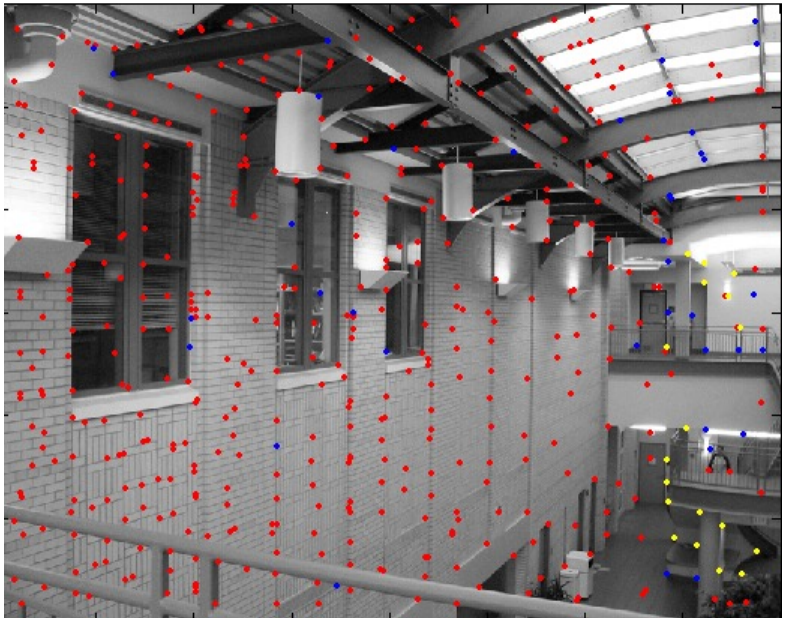


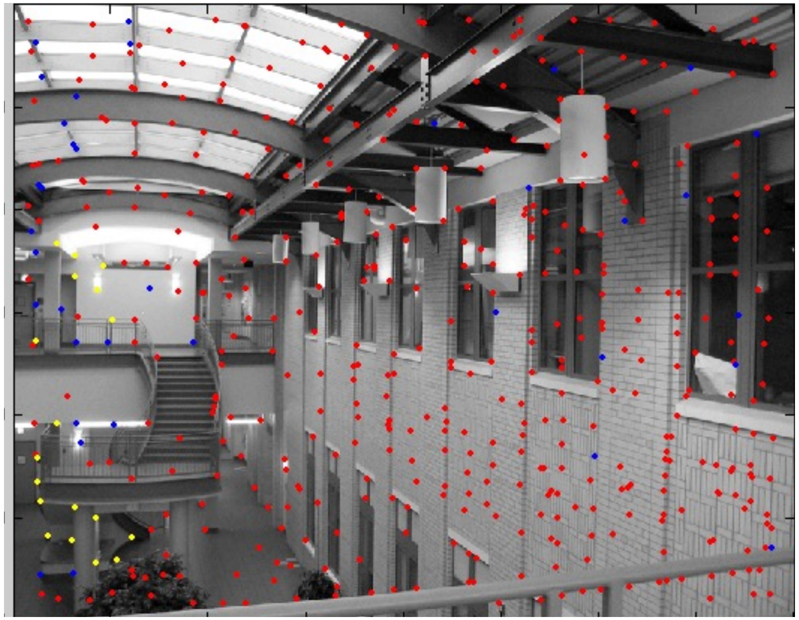
RANSAC for estimating homography

RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers* where $dist(p_i', \boldsymbol{H}\, p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

RANSAC



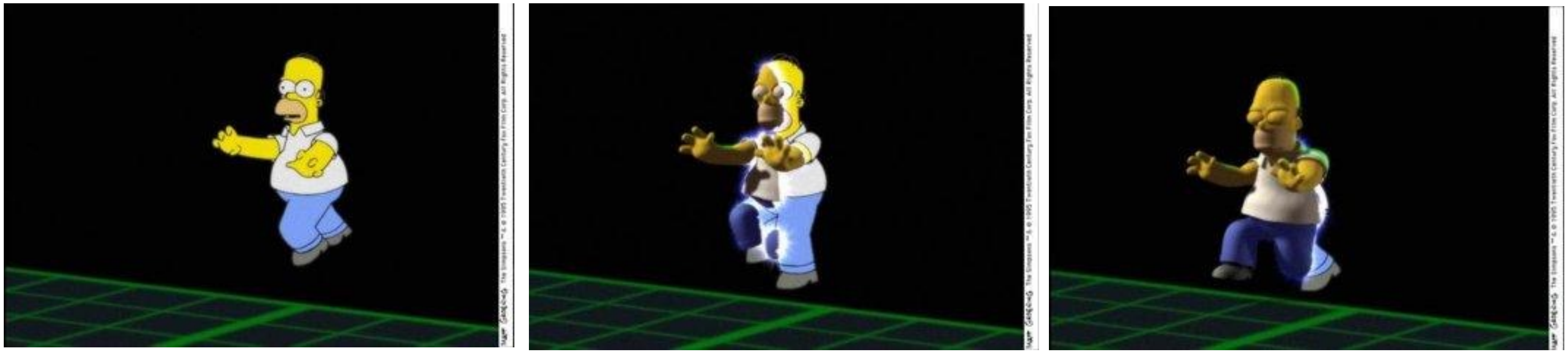


Limitations of Alignment

We need to know the global transform (e.g. affine, homography, etc)

Breaking out of 2D

…now we are ready to break out of 2D



And enter the real world!

on to 3D…

Enough of images!

We want more of the plenoptic function

We want real 3D scene walk-throughs:

- Camera rotation
- Camera translation

3D is super cool!

https://rd.nytimes.com/projects/reconstructing-journalistic-scenes-in-3d

3D is super cool!

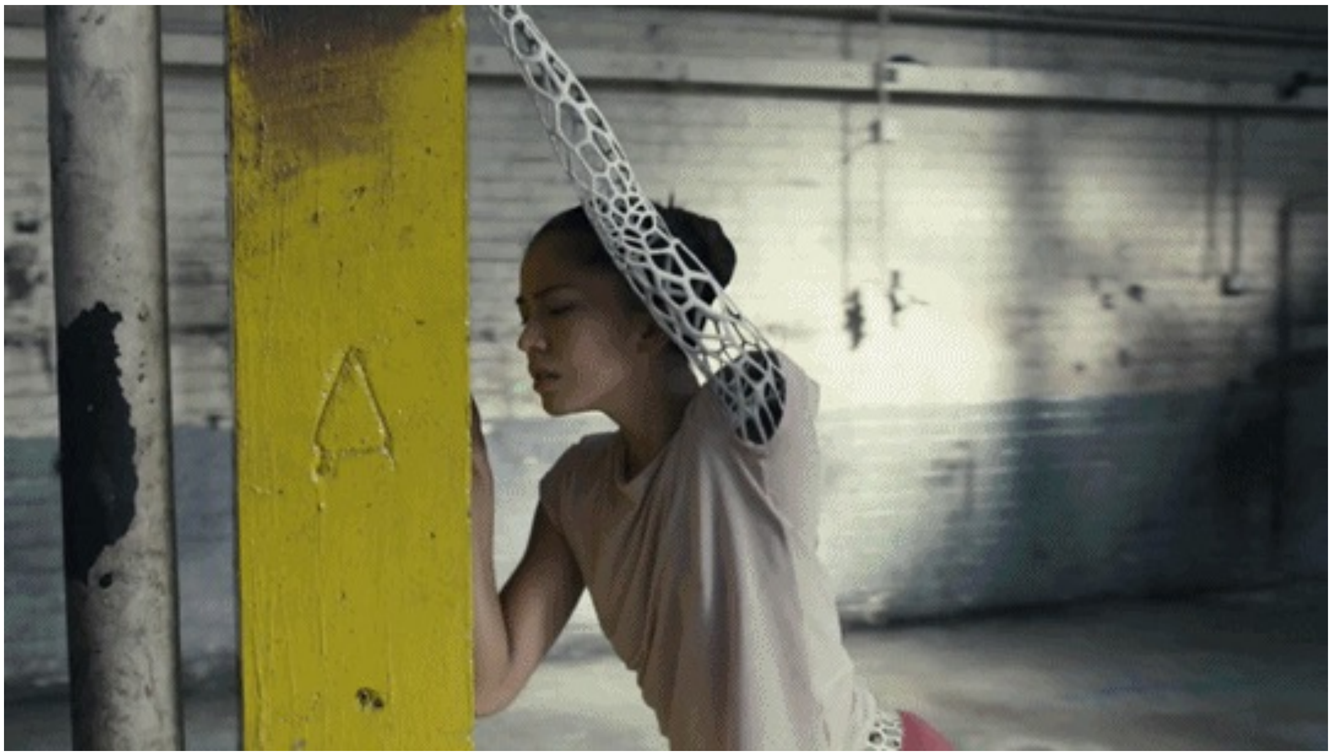
@capturingreality

[@organiccomputer]

NeRF in the wild (will get to in few more lectures)

NeRF in the Wild, Martin-Brualla, Radwan et al. CVPR 2021

Not just about 3D reconstruction

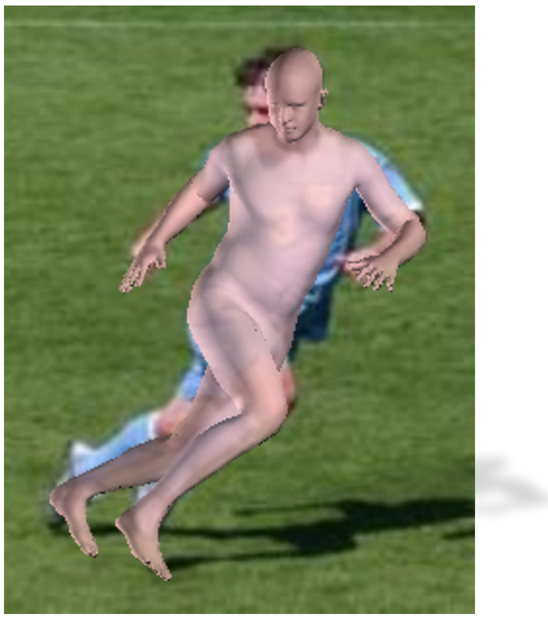


[The Chemical Brothers - Wide Open ft. Beck, MV]

3D for video editing

My Research

Single-View 3D Human Mesh Recovery

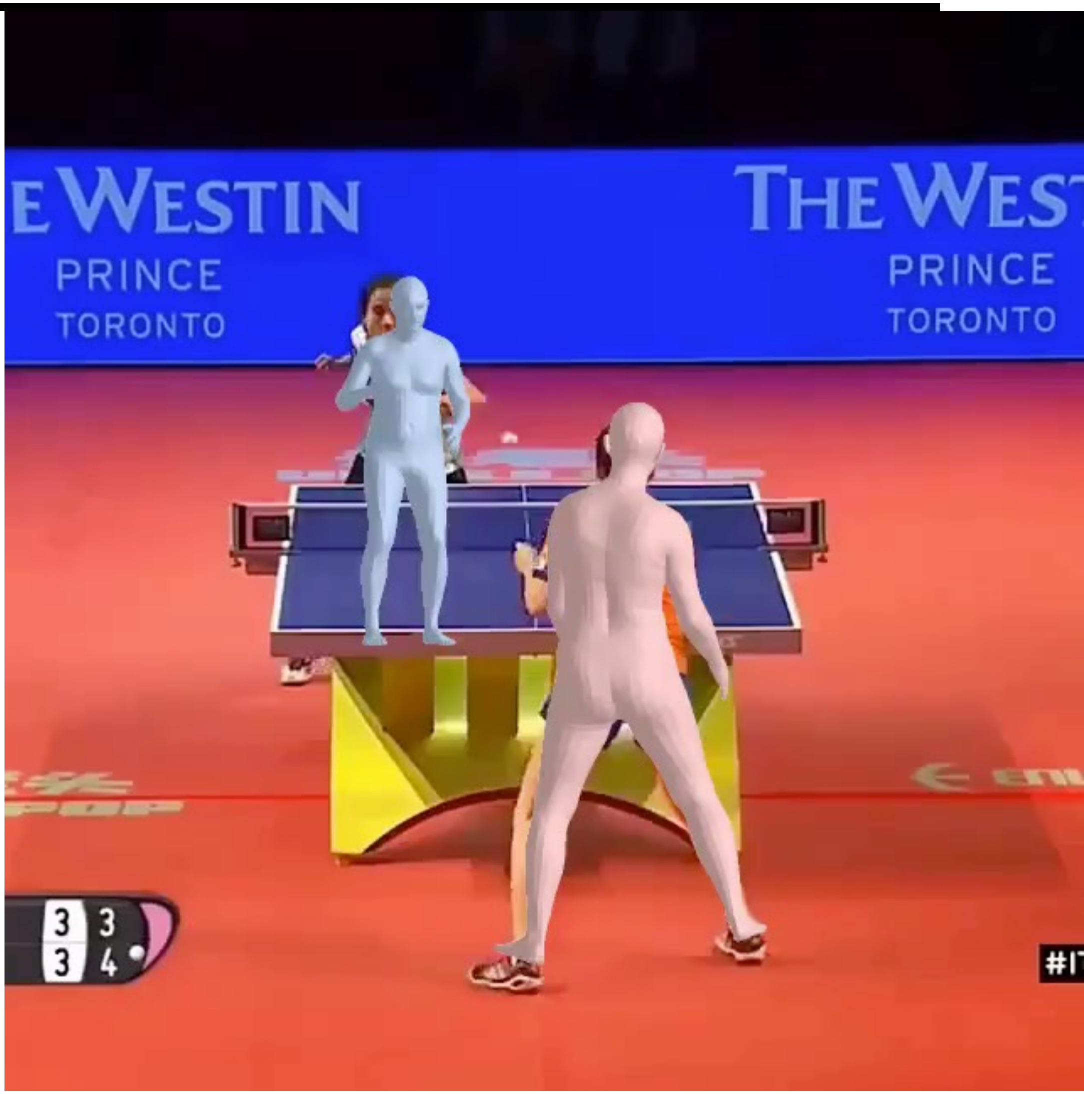


[Bogo∗, **Kanazawa**∗, Lassner, Gehler, Romero, Black ECCV ’16]

In everyday photos

Kanazawa, Black, Jacobs, Malik. CVPR 2018

Or from Video



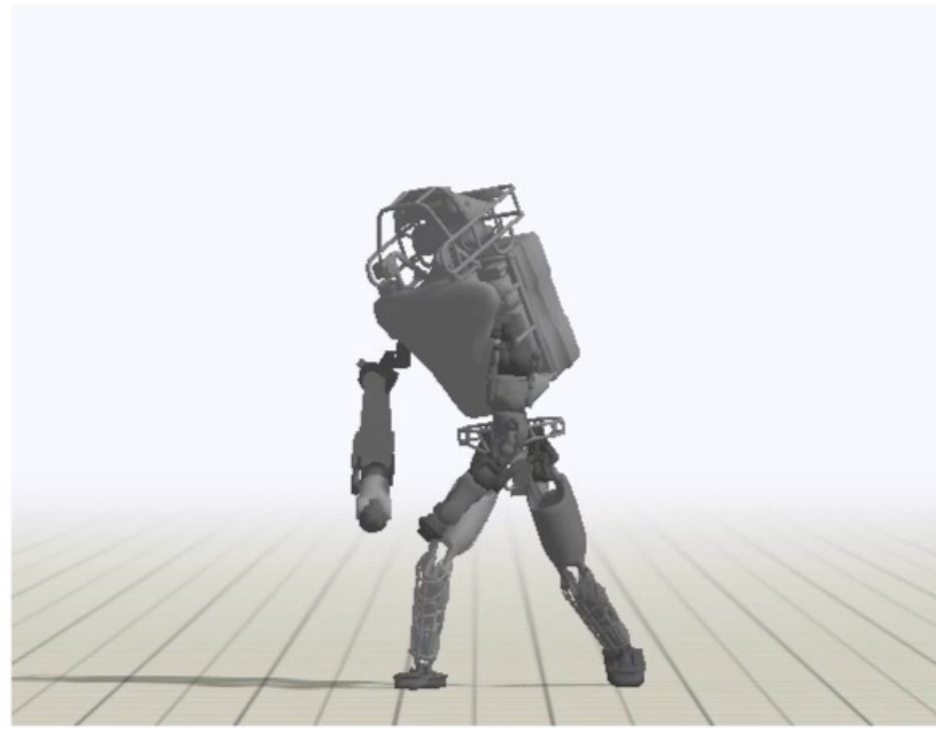
Kanazawa, Zhang, and Felsen et al. CVPR 2019

In more detail

Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization,
Saito, Huang, Natsume, Morishima, **Kanazawa**, Li, ICCV 2019

Teaching robots how to dance from watching YouTube

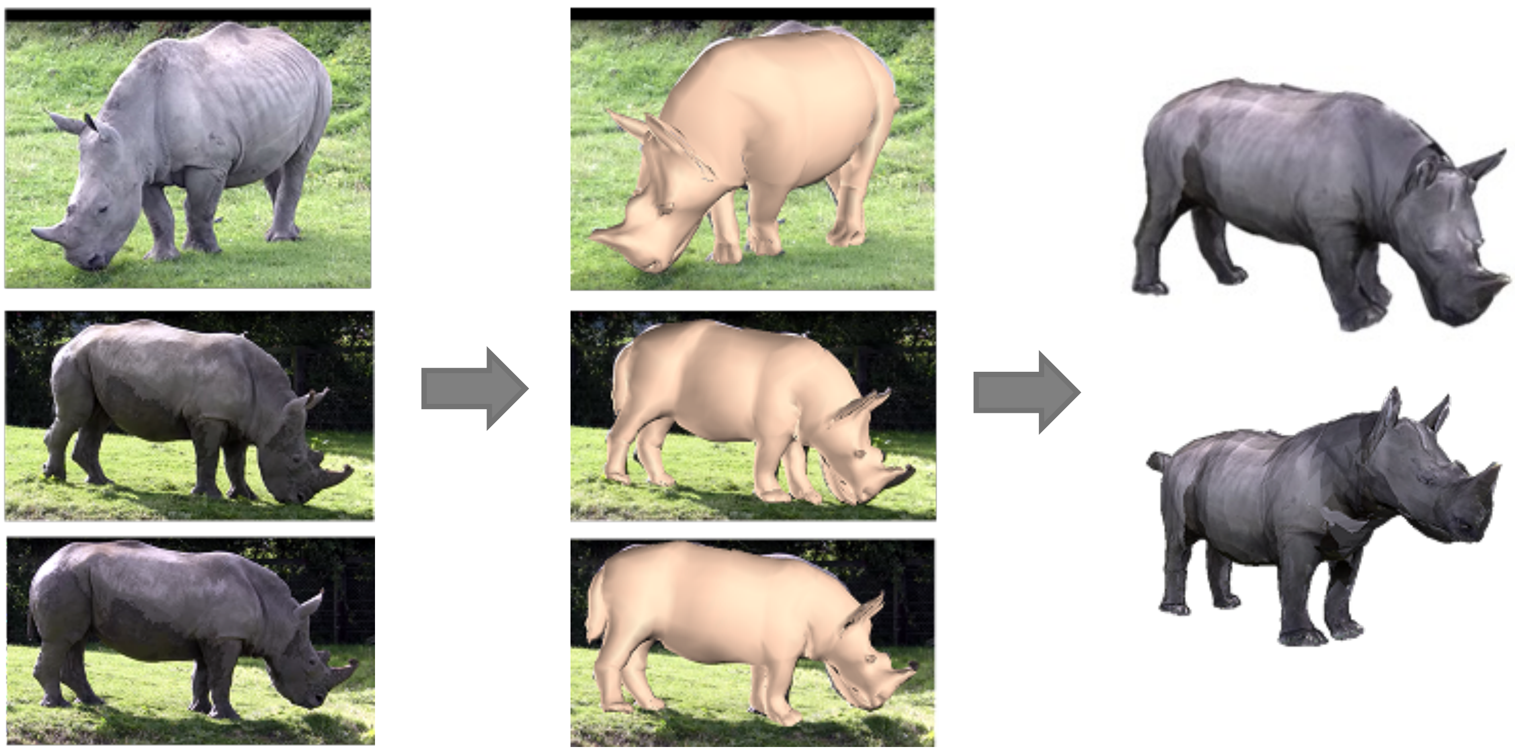
Video



Policy

Peng, Kanazawa, Malik, Abbeel, Levine
"SFV: Reinforcement Learning of Physical Skills from Videos", SIGGRAPH Asia 2018

Reconstructing Animals with Human Input



Zuffi, Kanazawa, Black, *“Lions and Tigers and Bears: Capturing Non-Rigid, 3D, Articulated Shape from Images”,* CVPR 2018

Print it!!

[Kanazawa*, Tulsiani*, Efros, Malik, ECCV 2018]

Zuffi, Kanazawa, Black, *“Lions and Tigers and Bears: Capturing Non-Rigid, 3D, Articulated Shape from Images”*, CVPR 2018

Flying into an image

Infinite Nature: Perpetual View Generation of Natural Scenes from a Single Image, ICCV 2021

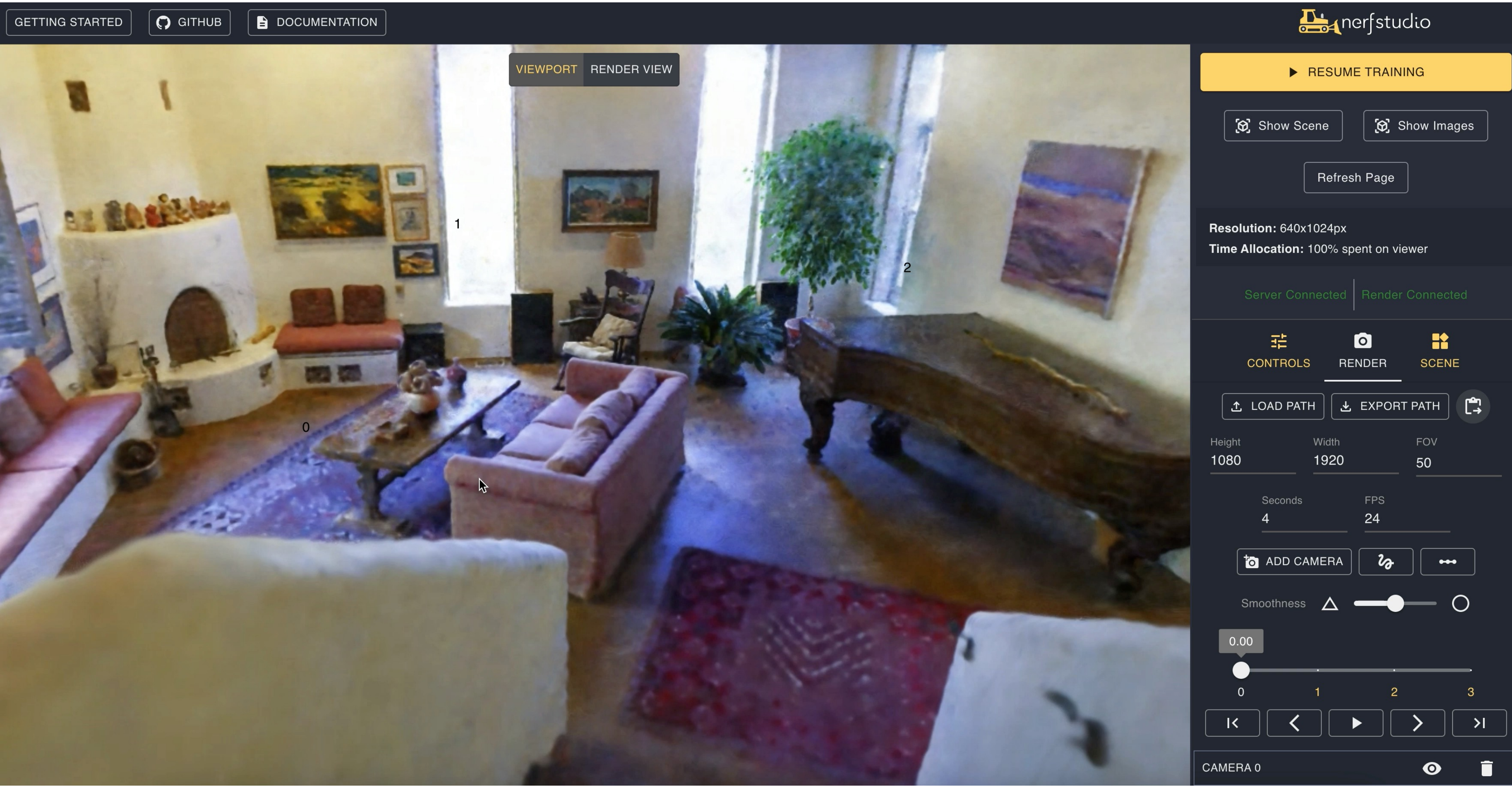
nerfstudio

Matthew Tancik*, Ethan Weber*, Evonne Ng*, Ruilong Li, Brent Yi,
Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi,
Abhik Ahuja, David McAllister, Angjoo Kanazawa

100+ additional Github contributors

Matt Ethan Evonne

3D Capture


GETTING STARTED
GITHUB
DOCUMENTATION
nerfstudio
VIEWPORT
RENDER VIEW
RESUME TRAINING
Show Scene
Show Images
Refresh Page
Resolution: 640x1024px
Time Allocation: 100% spent on viewer
Server Connected
Render Connected
CONTROLS
RENDER
SCENE
LOAD PATH
EXPORT PATH
Height
1080
Width
1920
FOV
50
Seconds
4
FPS
24
ADD CAMERA
Smoothness
0.00
0
1
2
3
CAMERA 0

so on to 3D…

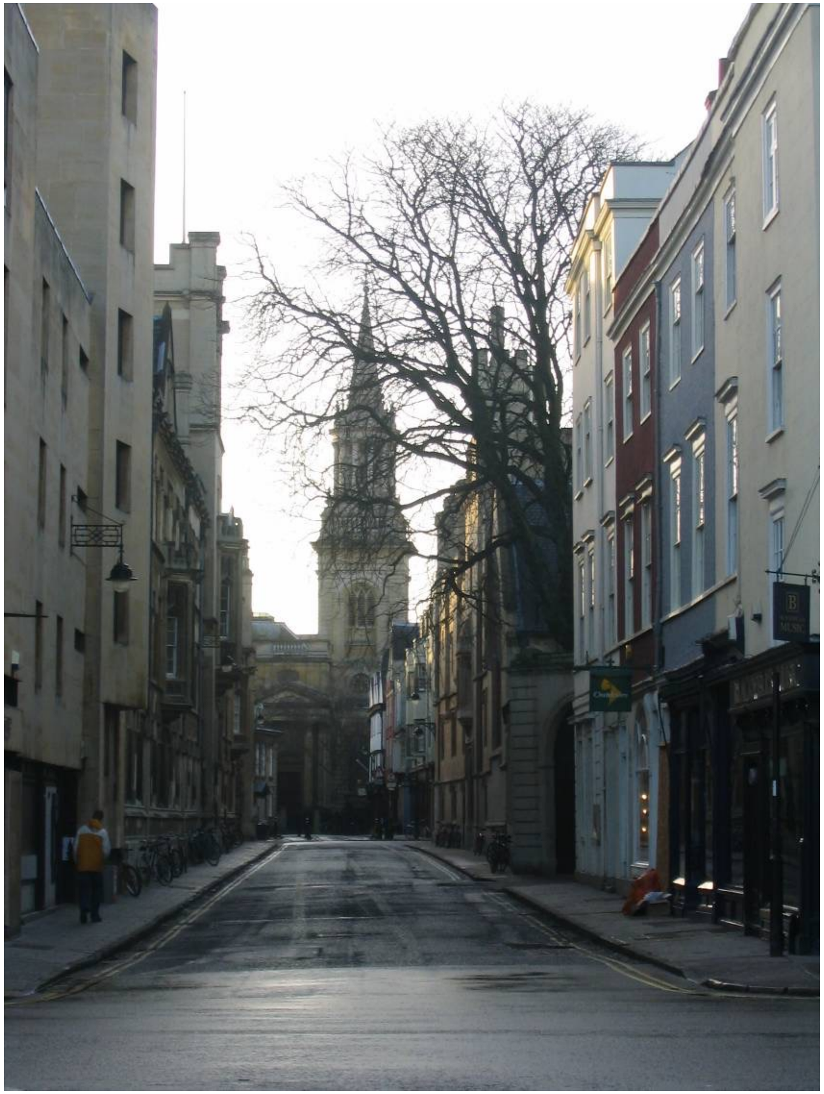
Enough of images!

We want more of the plenoptic function

We want real 3D scene walk-throughs:

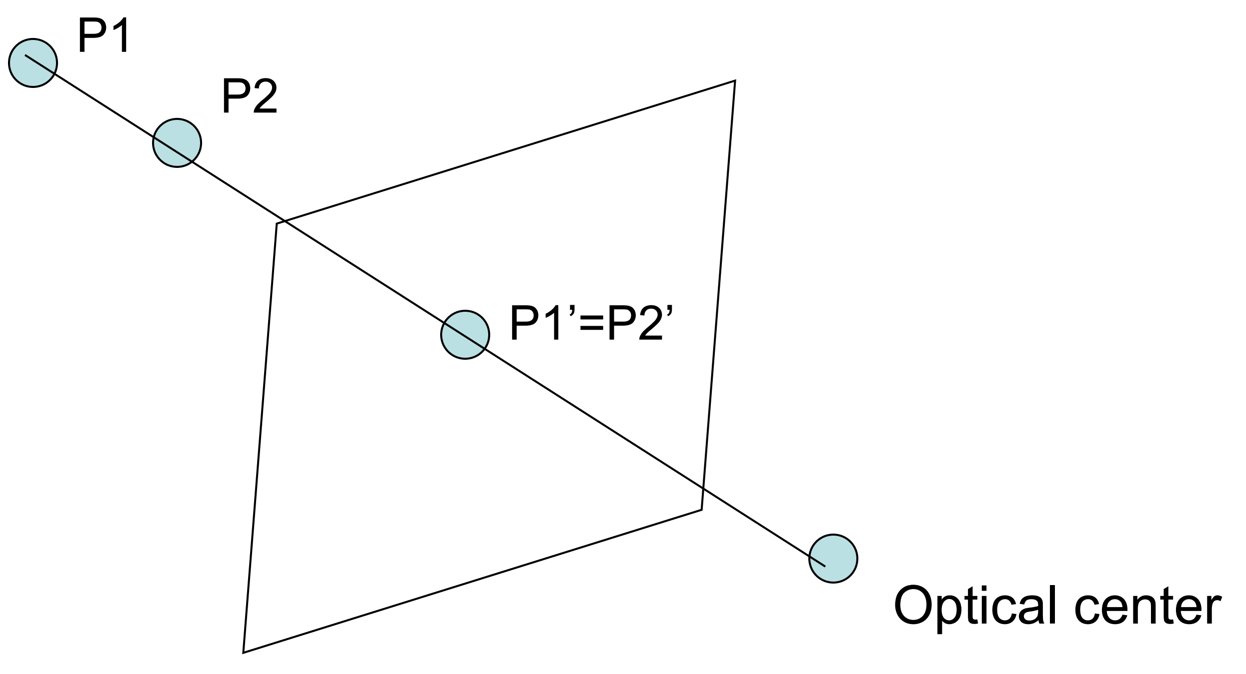
- Camera rotation
- Camera translation

Can we do it from a single photograph?



Why multiple views?

- Structure and depth are inherently ambiguous from single views.

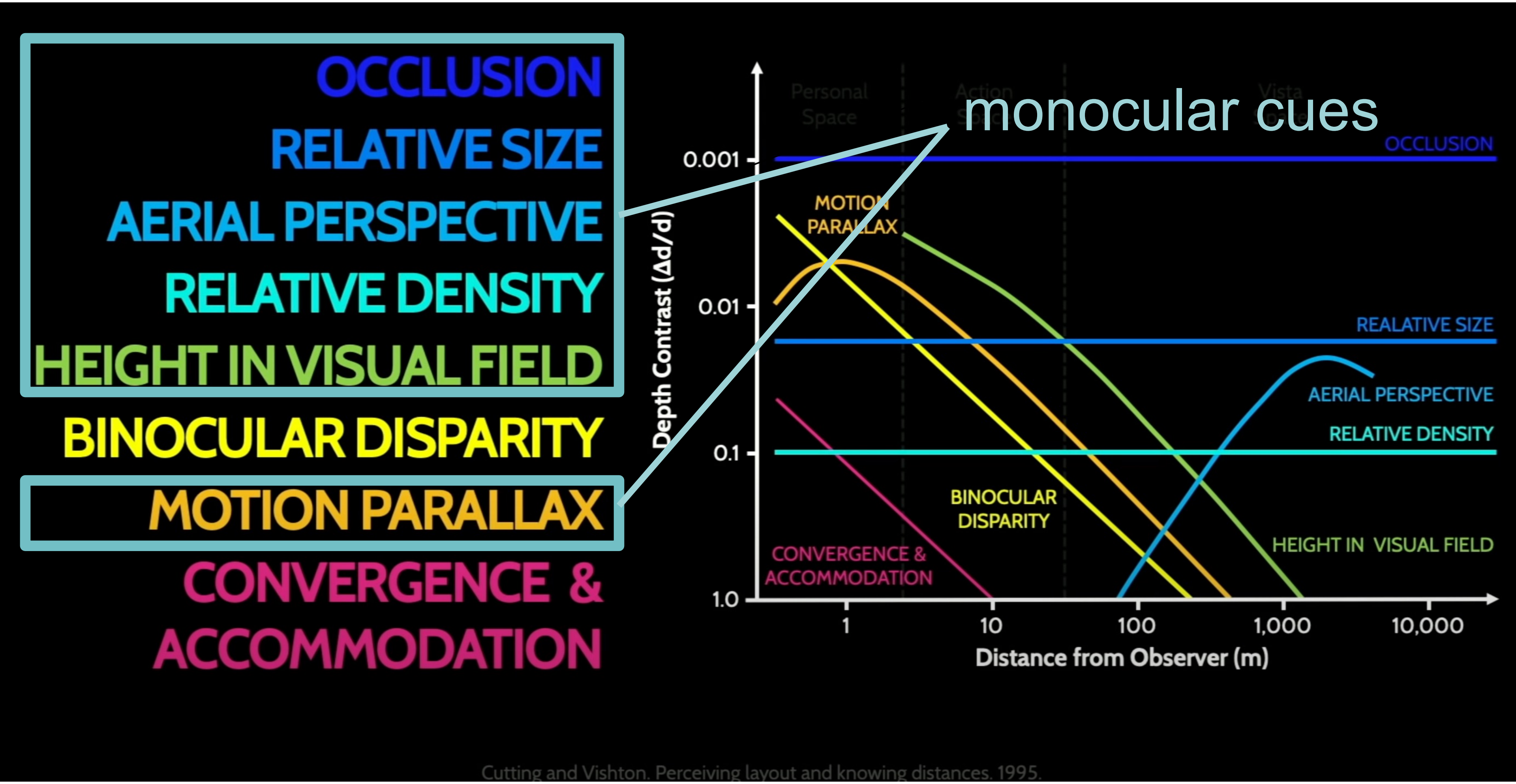


Why multiple views?

- Structure and depth are inherently ambiguous from single views.

Images from Lana Lazebnik

Human Depth Cues



https://youtu.be/LQwMAI9bGNY?si=6rf4F9NmB7o8vfqY

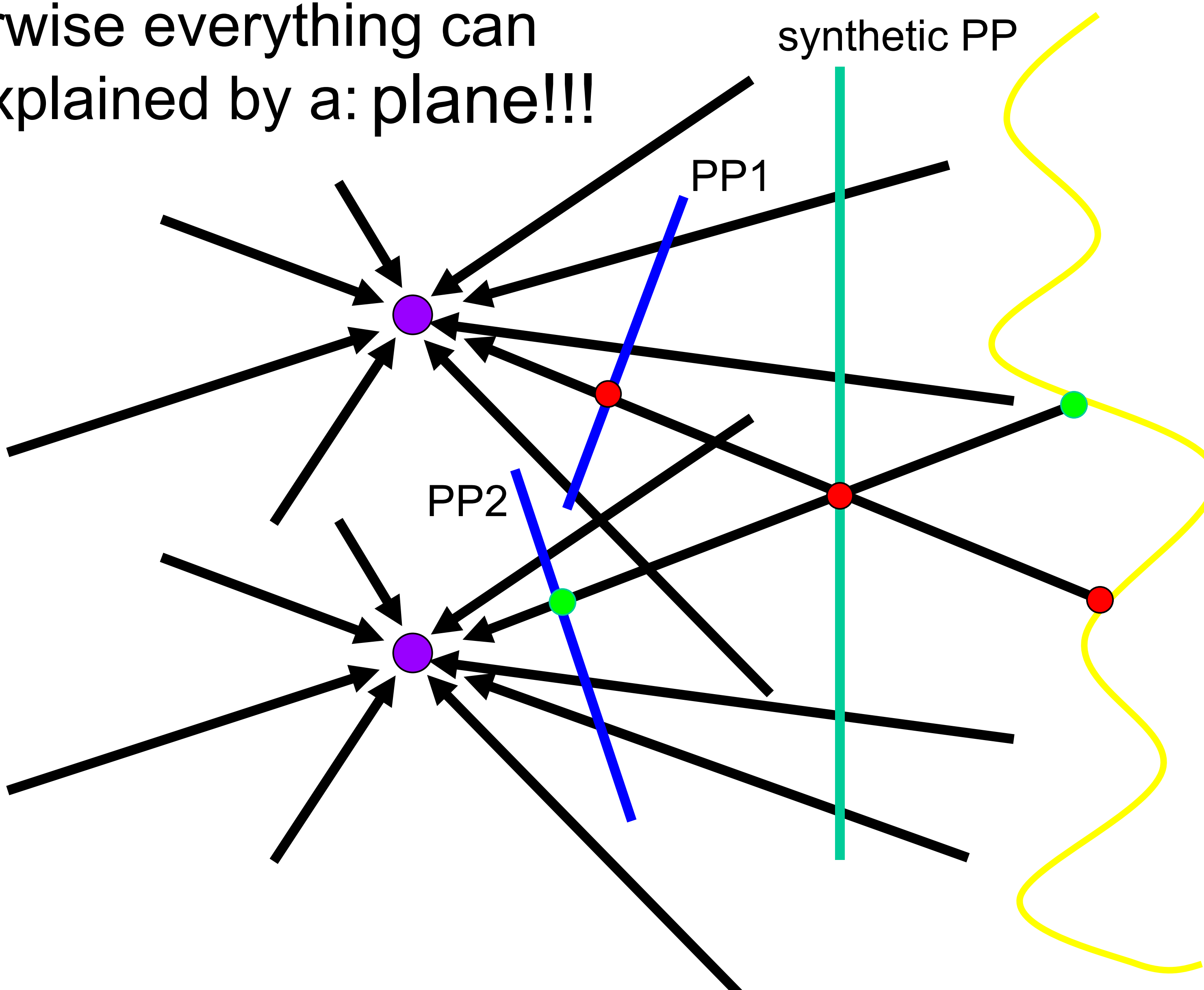
Geometric Depth Understanding

- Ambiguous from a single image

- Why?

Need two different camera center

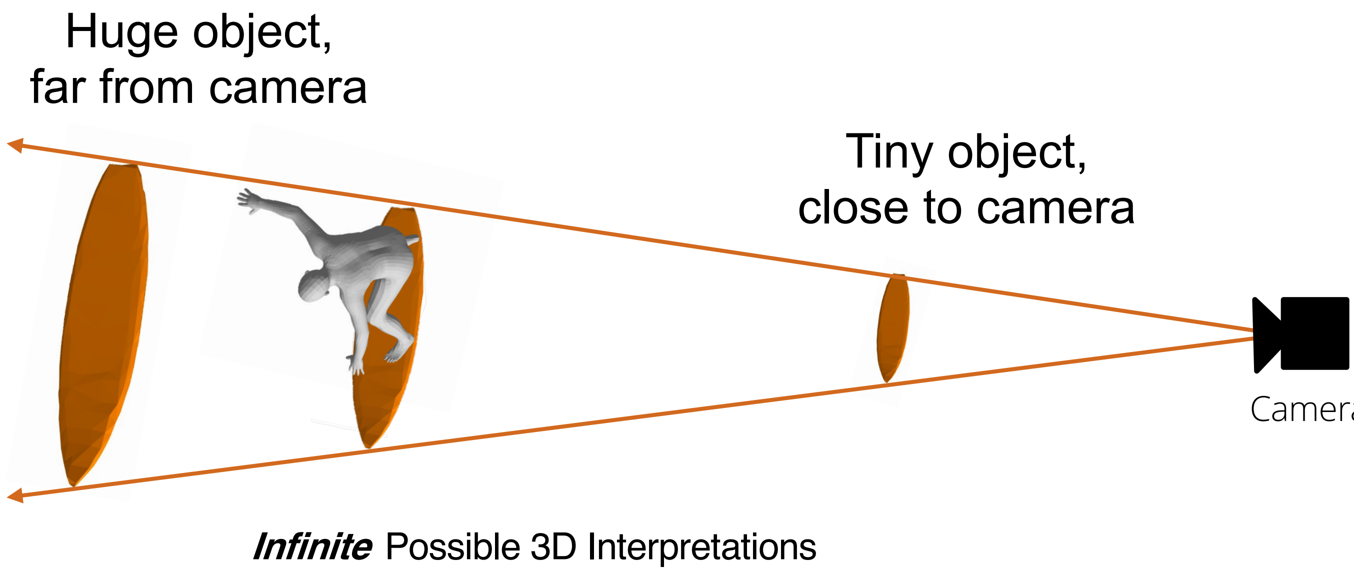
otherwise everything can be explained by a: plane!!!



Fundamental Depth Ambiguity in 2D → 3D

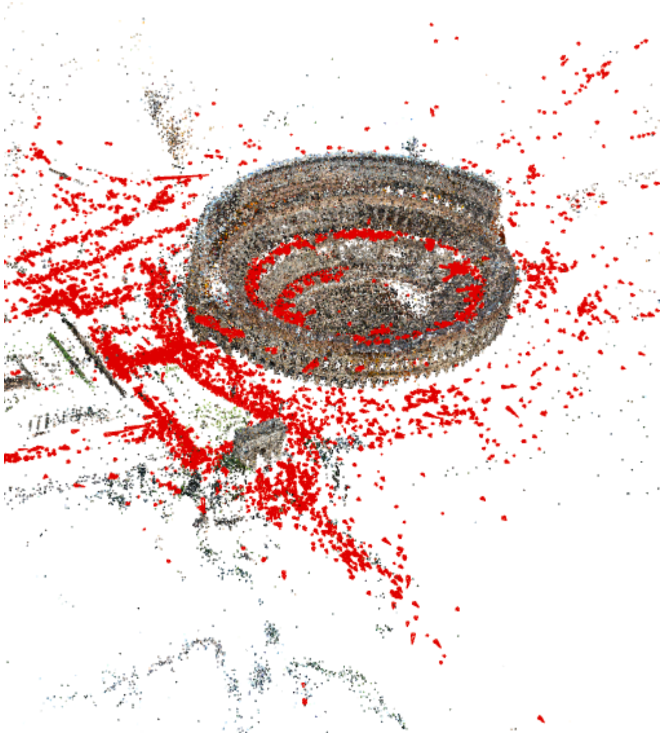
Original Image

Same 2D Projection



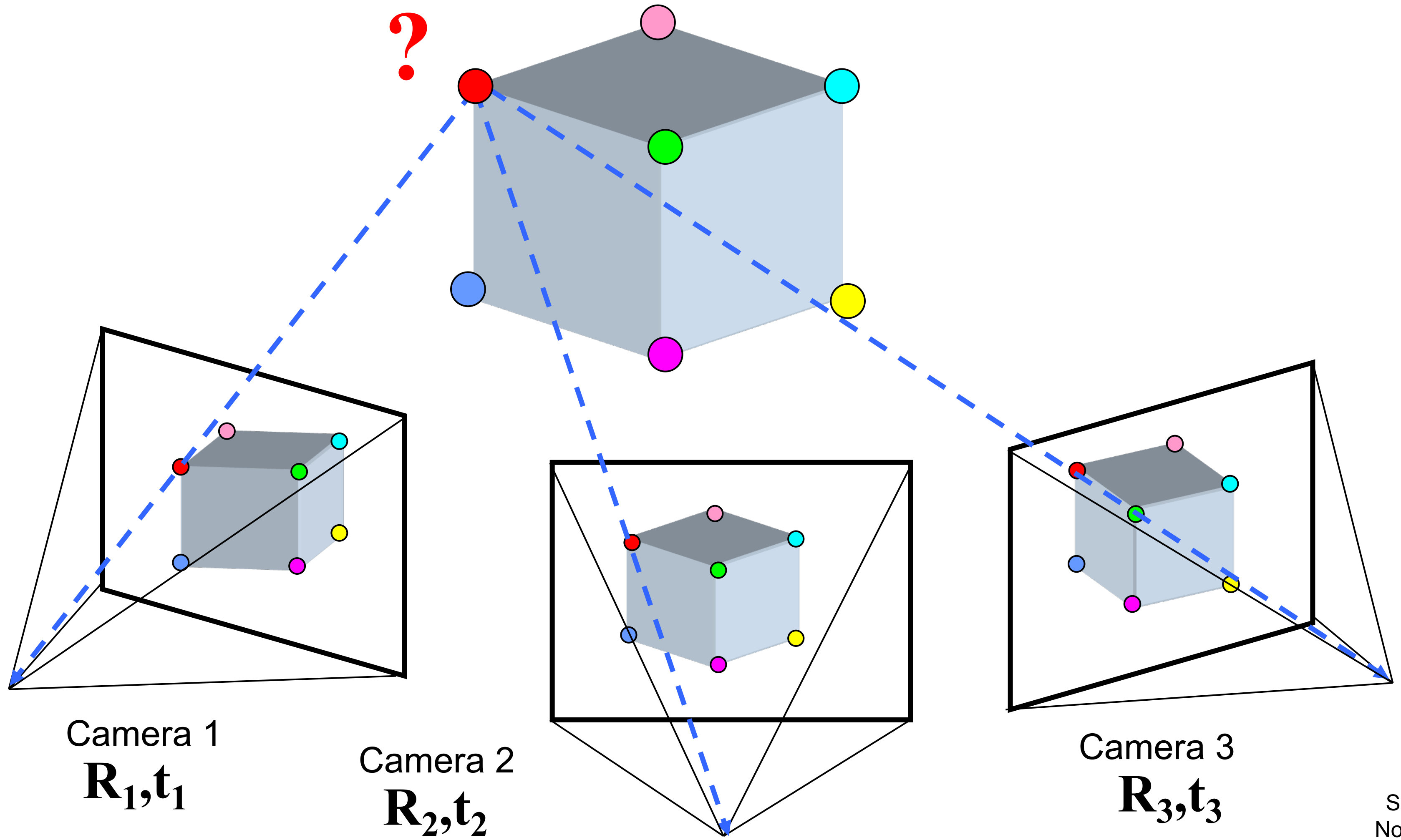
2.5D vs 3D

- is 3D = depth from a single image?



Multi-view geometry problems

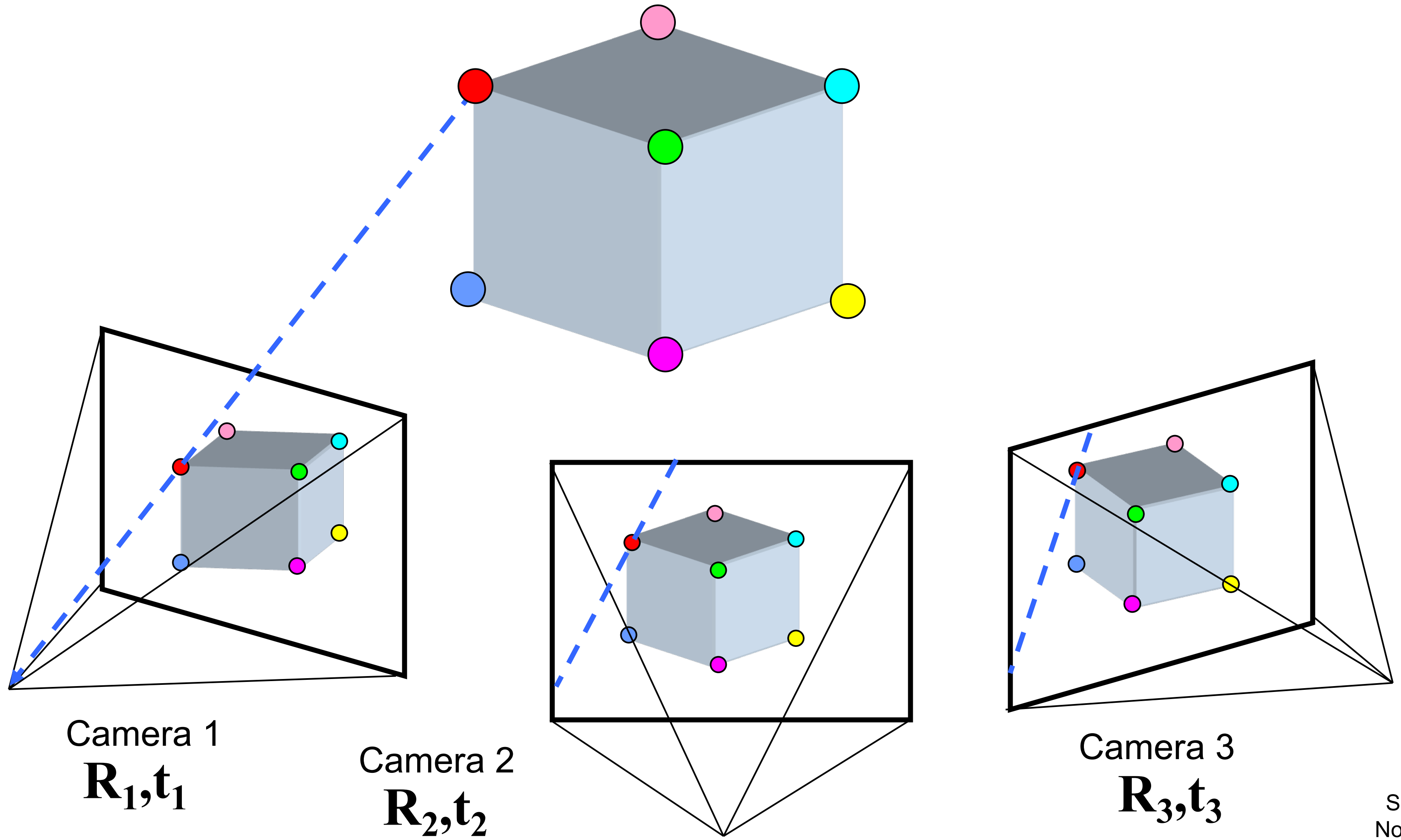
- **Structure:** What is the 3D coordinate of a point that can be seen in multiple images?



Slide credit: Noah Snavely

Multi-view geometry problems

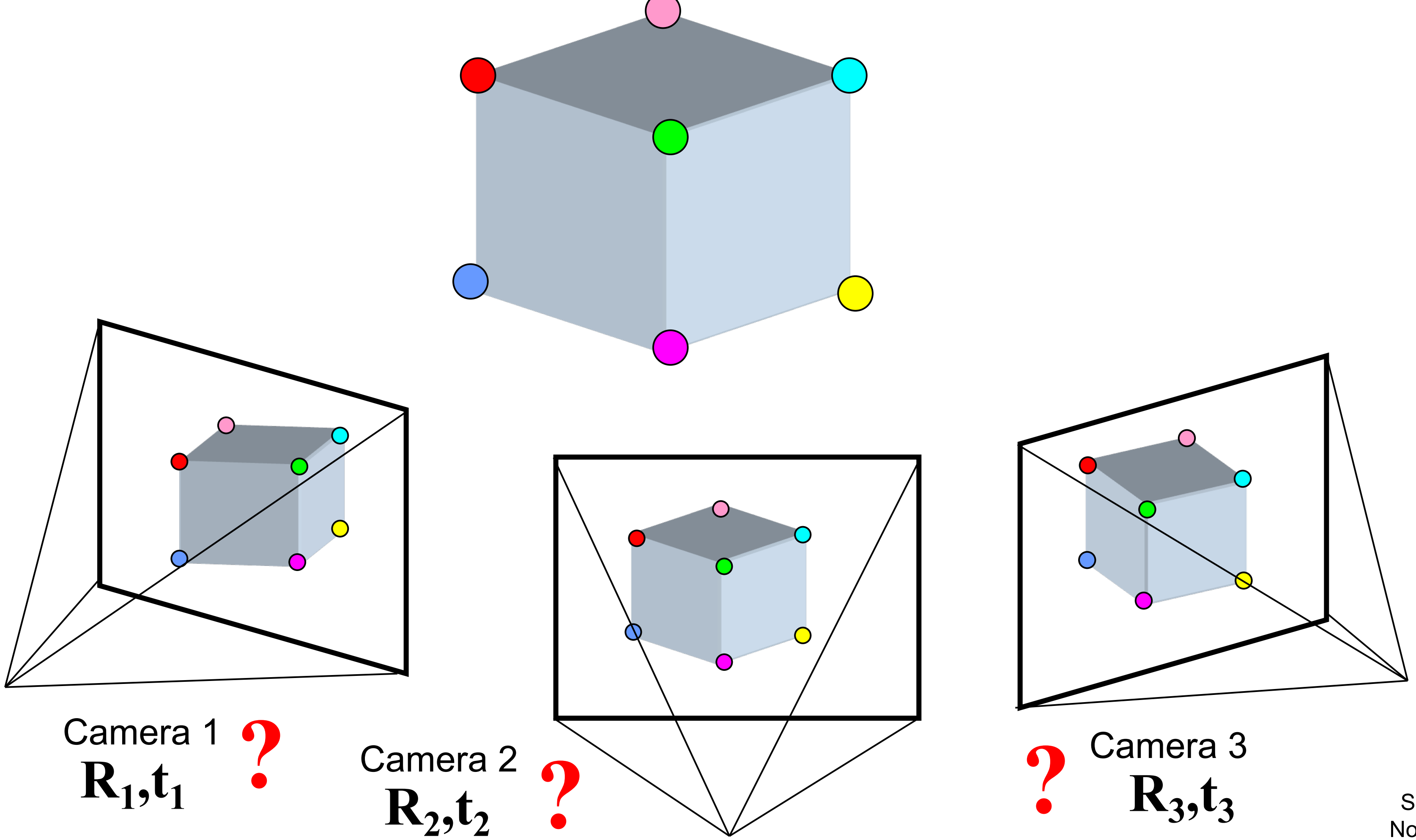
- **Correspondence:** Given a point in one of the images, where are the corresponding points in the other images?



Slide credit: Noah Snavely

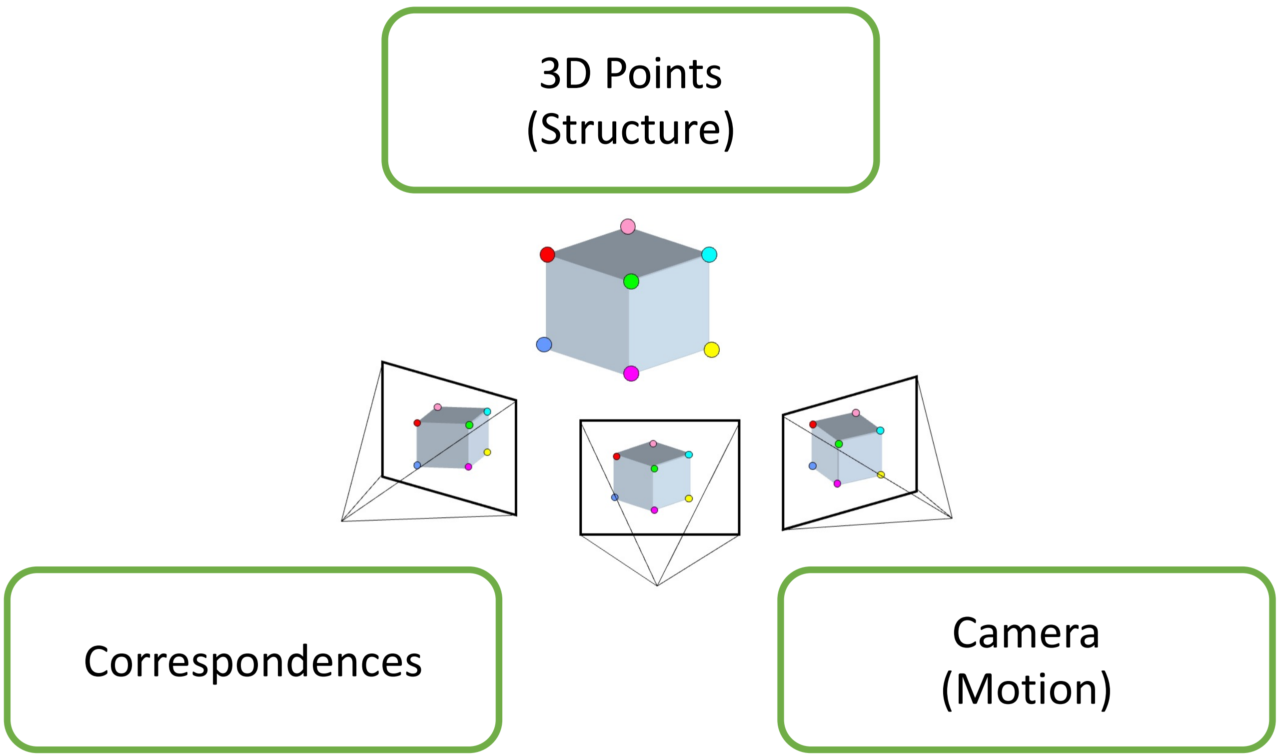
Multi-view geometry problems

- **Motion:** Given a set of corresponding points in two or more images, what is the relative camera parameters between the images?

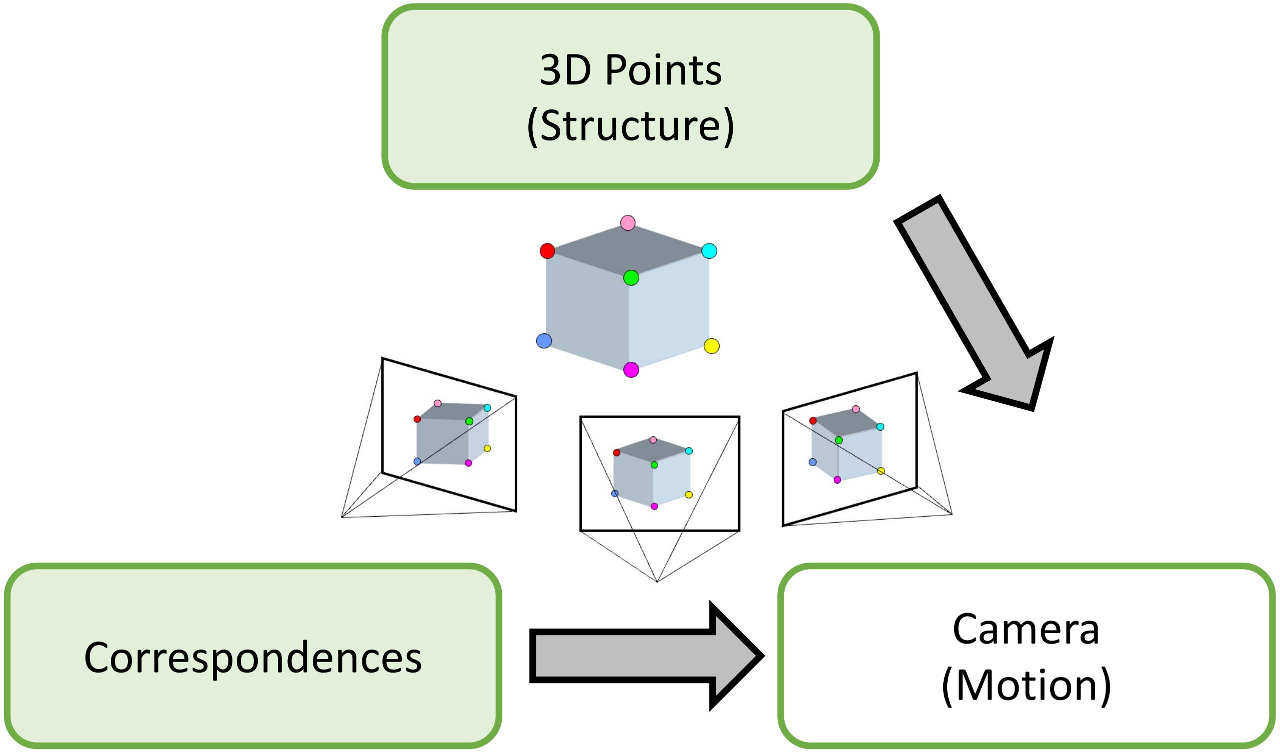


Slide credit: Noah Snavely

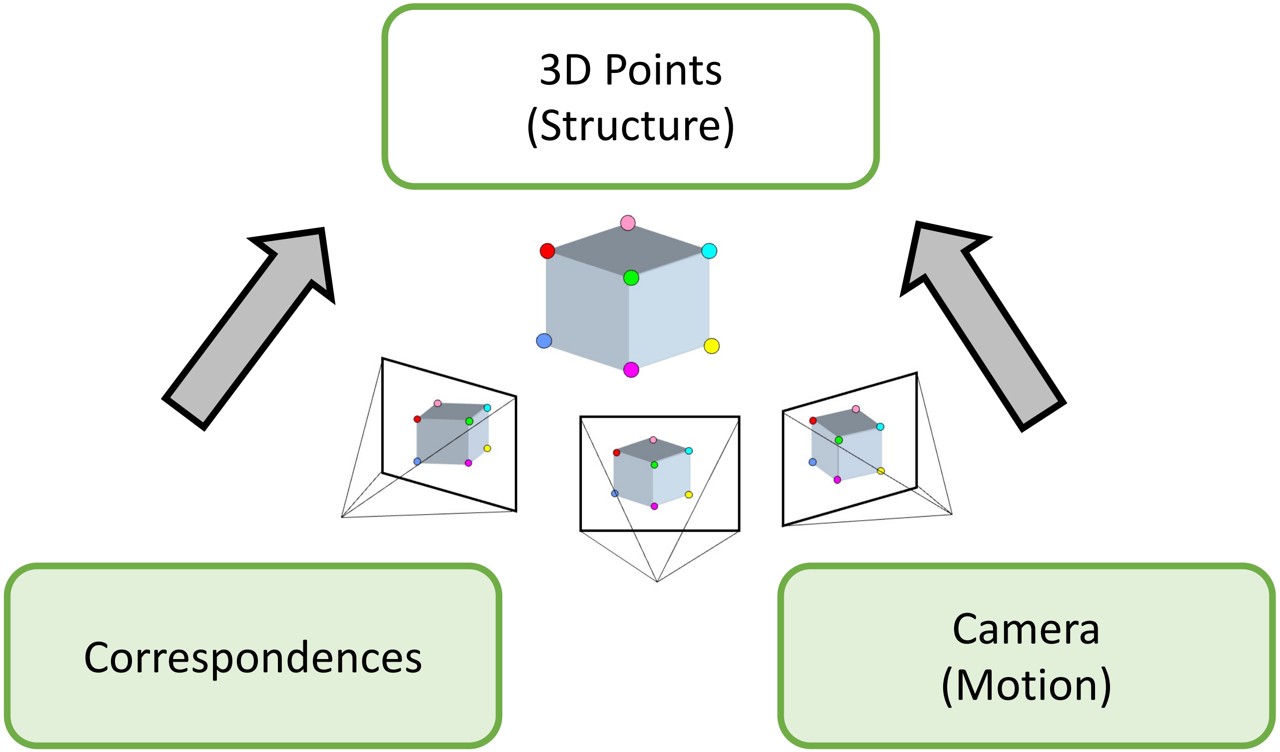
Big picture: 3 key components in 3D

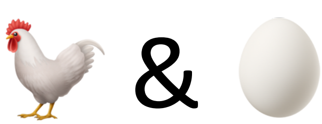


Big picture: 3 key components in 3D

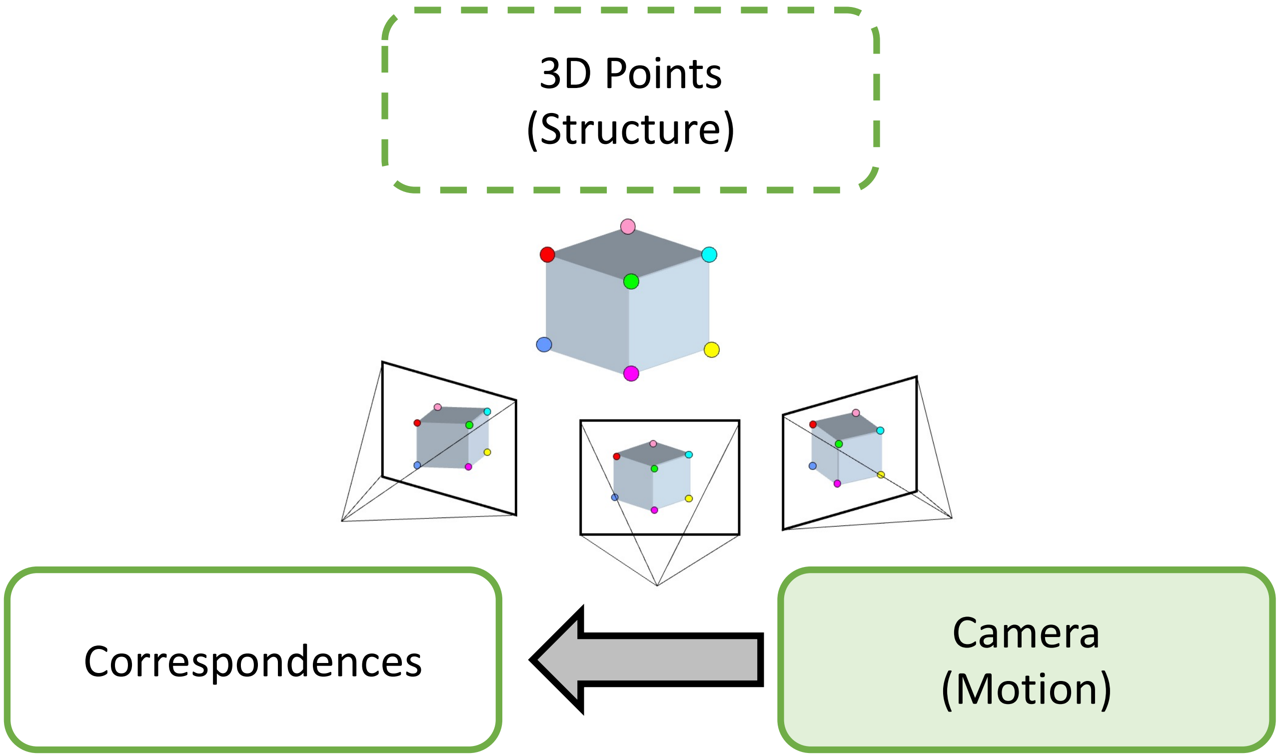


Big picture: 3 key components in 3D





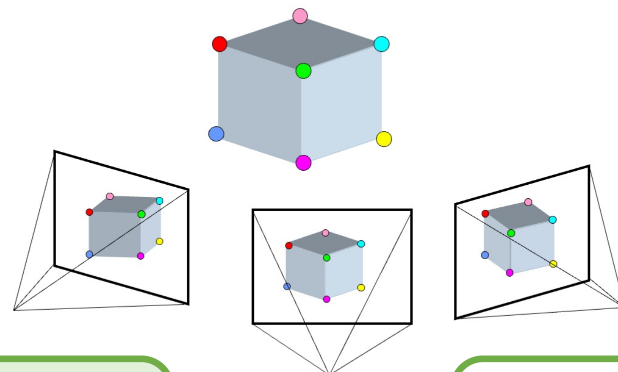
Big picture: 3 key components in 3D



🐓 & 🥚

Big picture: 3 key components in 3D

3D Points
(Structure)



Correspondences → Camera
(Motion)

 & 

From pixels to the 3D world

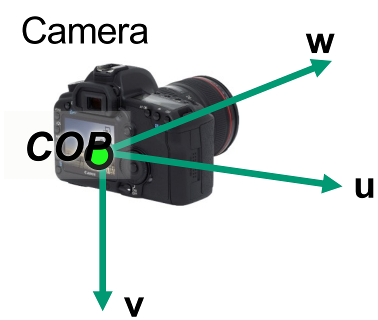
To go from pixels to 3D location in the **world coordinates**, we need to know two things about the camera:

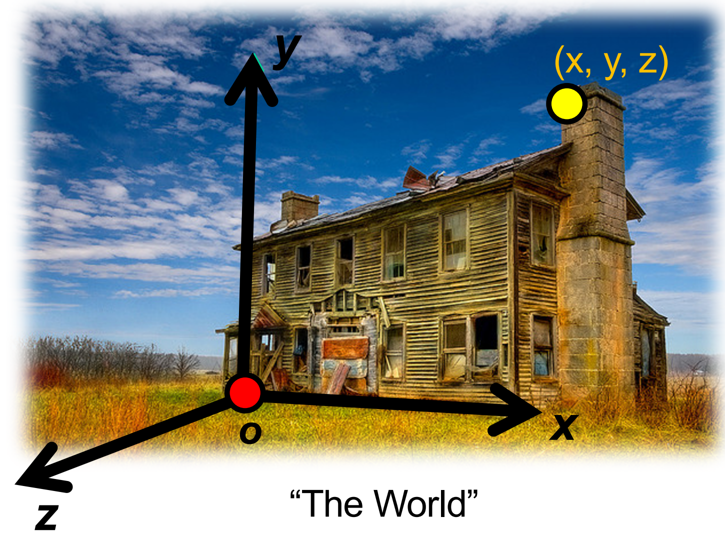
1. Position of the camera with respect to the world (extrinsics)
2. How the camera maps a point in the world to image (intrinsics)

Problem setup

There is a world coordinate frame and camera looking at the world

How can we model the geometry of a camera?





Three important coordinate systems:

1. *World* coordinates
2. *Camera* coordinates
3. *Image* coordinates

Slide credit: Noah Snavely

Coordinate frames + Transforms

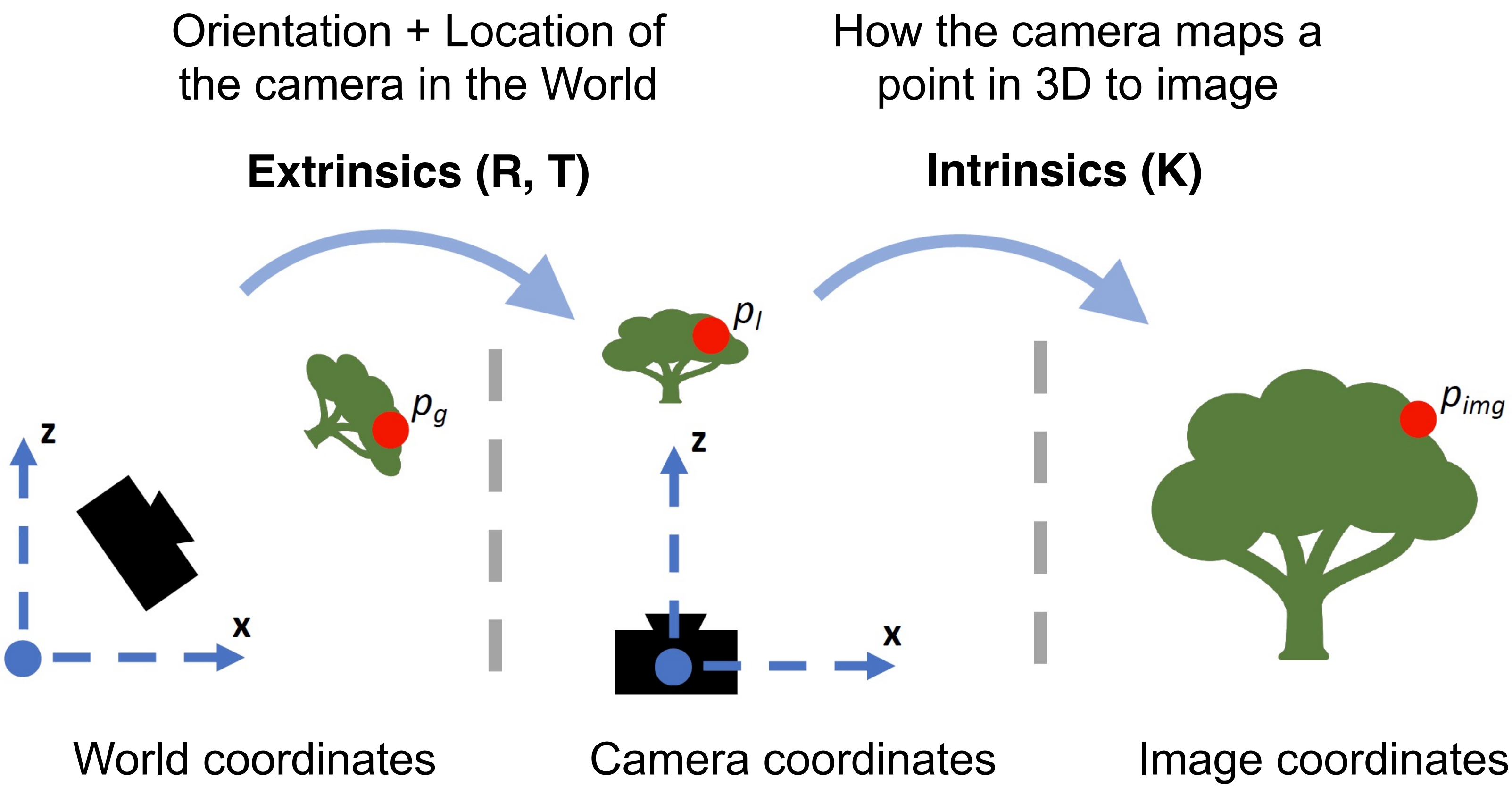


Figure credit: Peter Hedman

Camera: Specifics

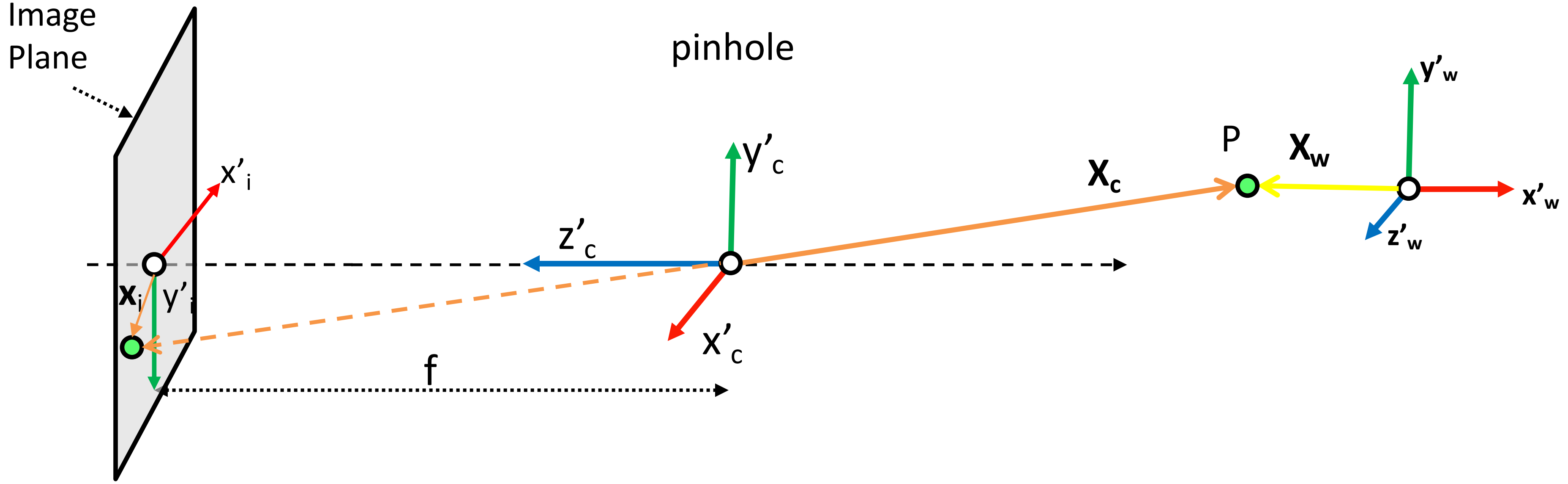


Image Coordinates

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Perspective Projection (3D to 2D)

Camera Coordinates

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

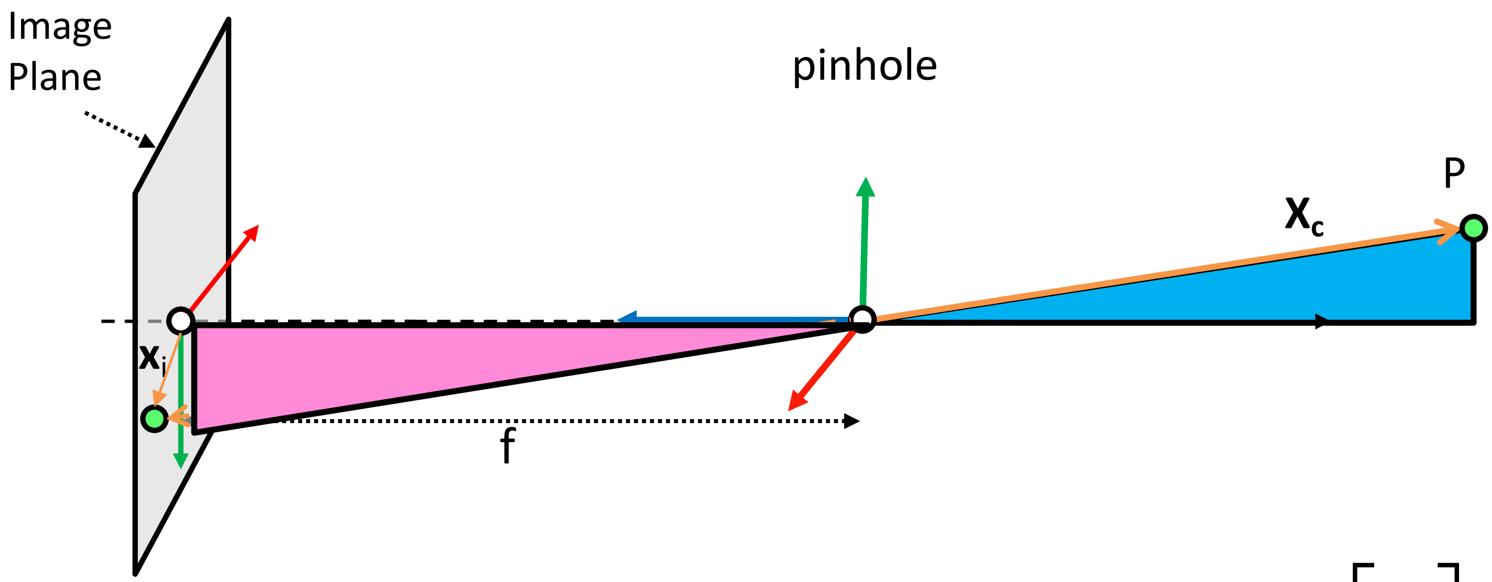
Coordinate Transformation (3D to 3D)

World Coordinates

$$\mathbf{X}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

Slide inspired by Shree Nayar

Perspective Projection



$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Image Coordinates

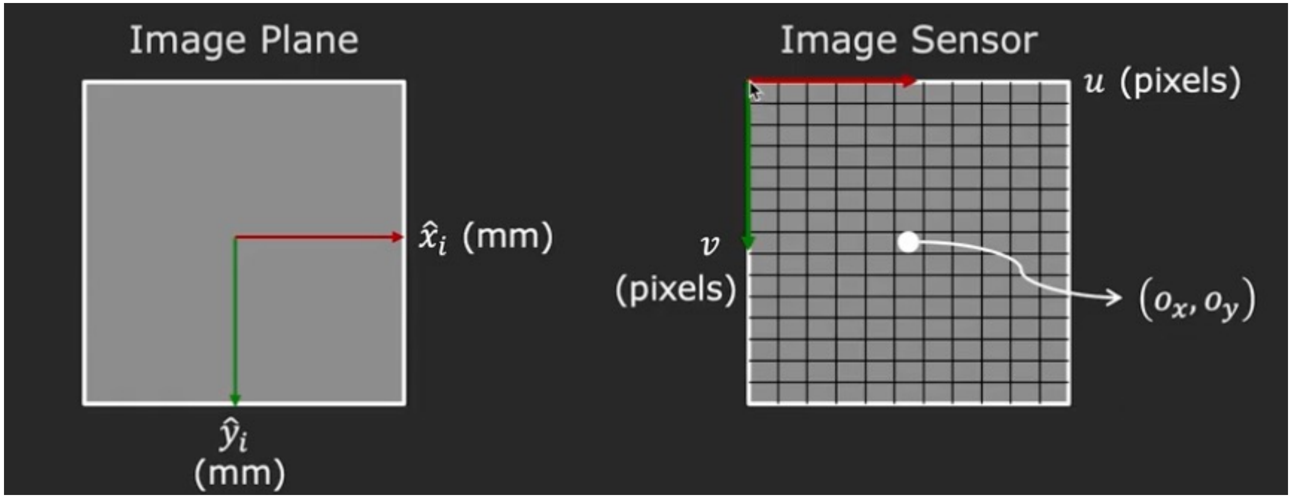
$$\frac{x_i}{f} = \frac{x_c}{z_c}$$

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Camera Coordinates

$$x_i = f\frac{x_c}{z_c}$$

Image Plane to Image Sensor Mapping



1. Account for pixel density (pixel/mm) & aspect ratio by scalars: $[m_x, m_y]$

$$m_x x_i, m_y y_i$$

2. Usually the top left corner is the origin. But in the image plane, the origin is where the optical axis pierces the plane! Need to shift by:

$$(o_x, o_y)$$

$$u_i = \alpha_x x_i + o_x = \alpha_x f \frac{x_c}{z_c} + o_x$$

where $[f_x, f_y] = [m_x f, m_y f]$

Pixel Coordinates:

$$u_i = f_x \frac{x_c}{z_c} + o_x \quad v_i = f_y \frac{y_c}{z_c} + o_y$$

With homogeneous coordinates

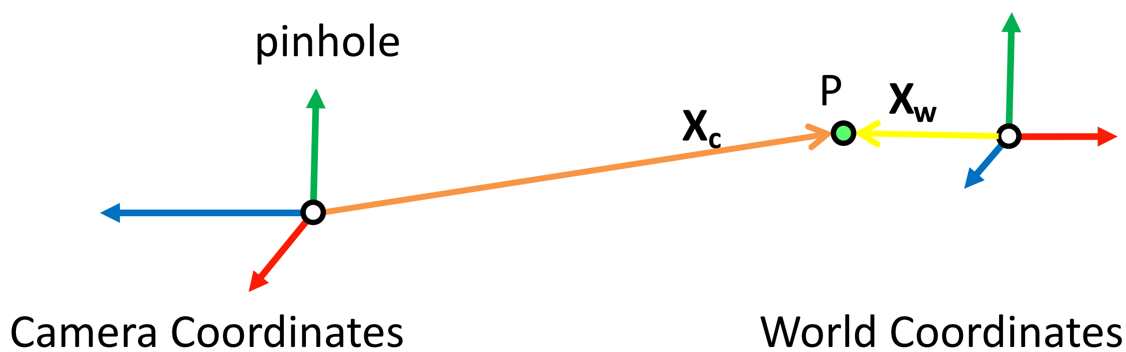
Perspective projection + Transformation to Pixel Coordinates:

$$u_i = f_x \frac{x_c}{z_c} + o_x \quad v_i = f_y \frac{y_c}{z_c} + o_y$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Intrinsic Matrix

Camera Transformation (3D-to-3D)



$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \quad \Leftarrow \quad \mathbf{X}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

Coordinate Transformation

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times 3} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Extrinsic

Matrix

Putting it all together

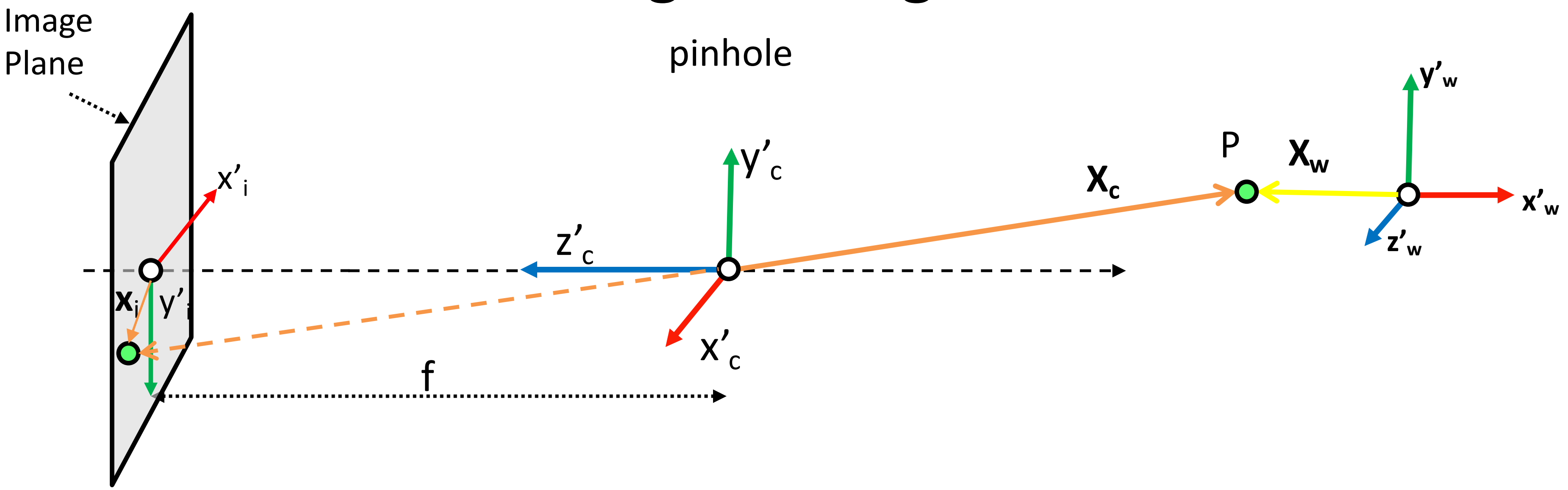


Image Coordinates

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Camera Coordinates

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

World Coordinates

$$\mathbf{X}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

Perspective Projection

$$\begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Coordinate Transformation

$$\begin{bmatrix} R_{3\times3} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$$

Slide inspired by Shree Nayar

Projection Matrix

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3\times 3} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

3 x 4 Projection matrix

What's the Degrees of Freedom?

For completeness, we need to add **skew** (this is 0 unless pixels are shaped like rhombi/parallelograms)

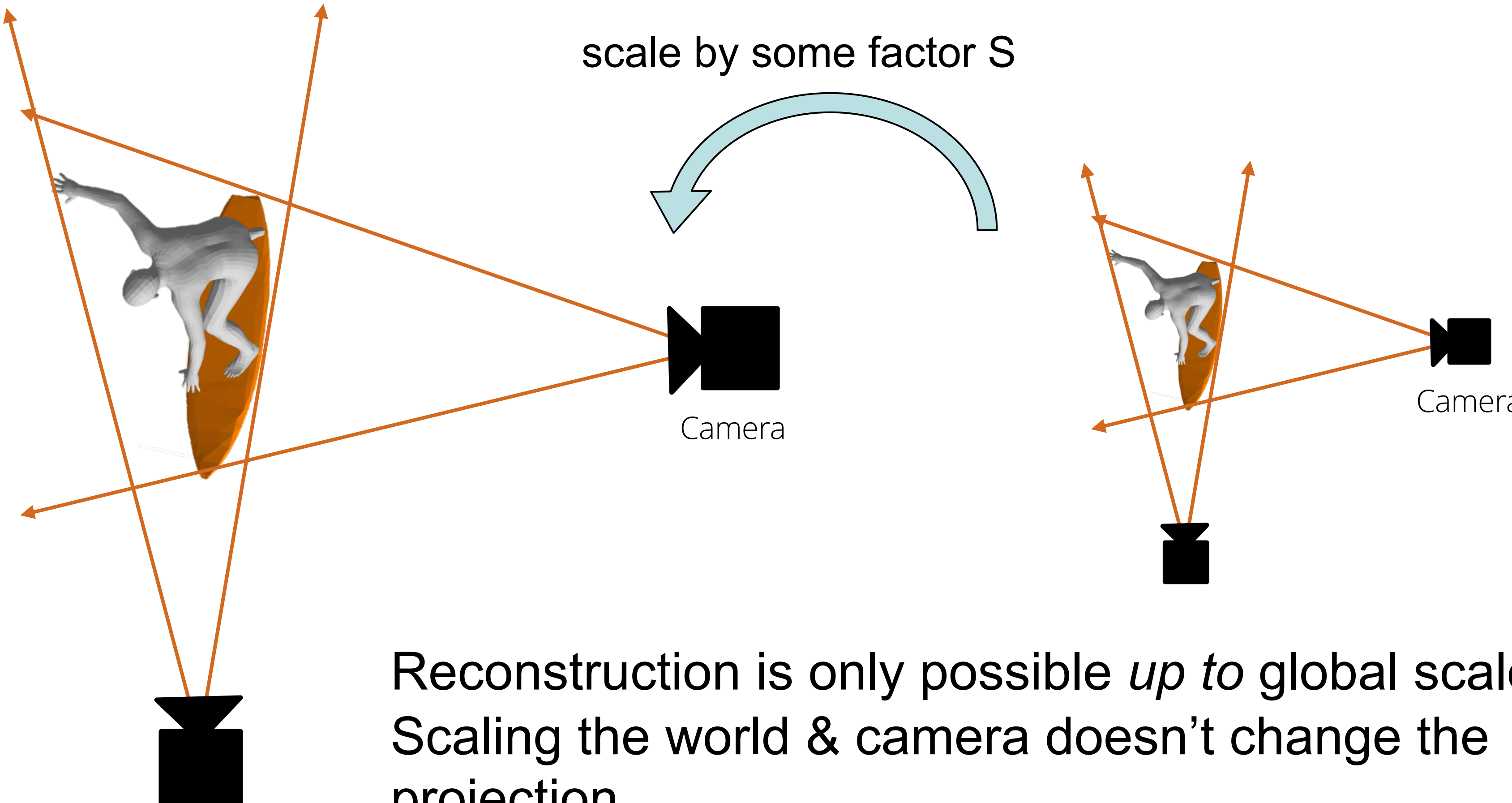
$$K = \begin{bmatrix} f_x & s & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Intrinsics: 4 + 1 (skew)

Extrinsic: 3 + 3 = 6

11 unknowns (up to scale)

Fundamental Scale Ambiguity



Reconstruction is only possible *up to* global scale
Scaling the world & camera doesn't change the projection
Unless you know something metric about the scene
e.g. surfboard is 2.1m

Going from World to Camera

Camera Coordinates

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

World Coordinates

$$\mathbf{X}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Extrinsic Matrix:

$$T_{w2c} = \begin{bmatrix} R_{3\times 3} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

$$\mathbf{X}_c = T_{w2c}\mathbf{X}_w$$

Going from Camera to World

Camera Coordinates

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

World Coordinates

$$\mathbf{X}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Extrinsic Matrix:

$$T_{w2c} = \begin{bmatrix} R_{3\times 3} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

$$T_{w2c}^{-1}\mathbf{X}_c = \mathbf{X}_w$$

Camera to Image

Image Coordinates

$$\mathbf{x}_i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}$$

Camera Coordinates

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Intrinsics Matrix:

$$K = \begin{bmatrix} f_x & s & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{x}_i = K\mathbf{X}_c$$

Image to Camera?

Image Coordinates

$$\mathbf{x}_i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}$$

Camera Coordinates

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

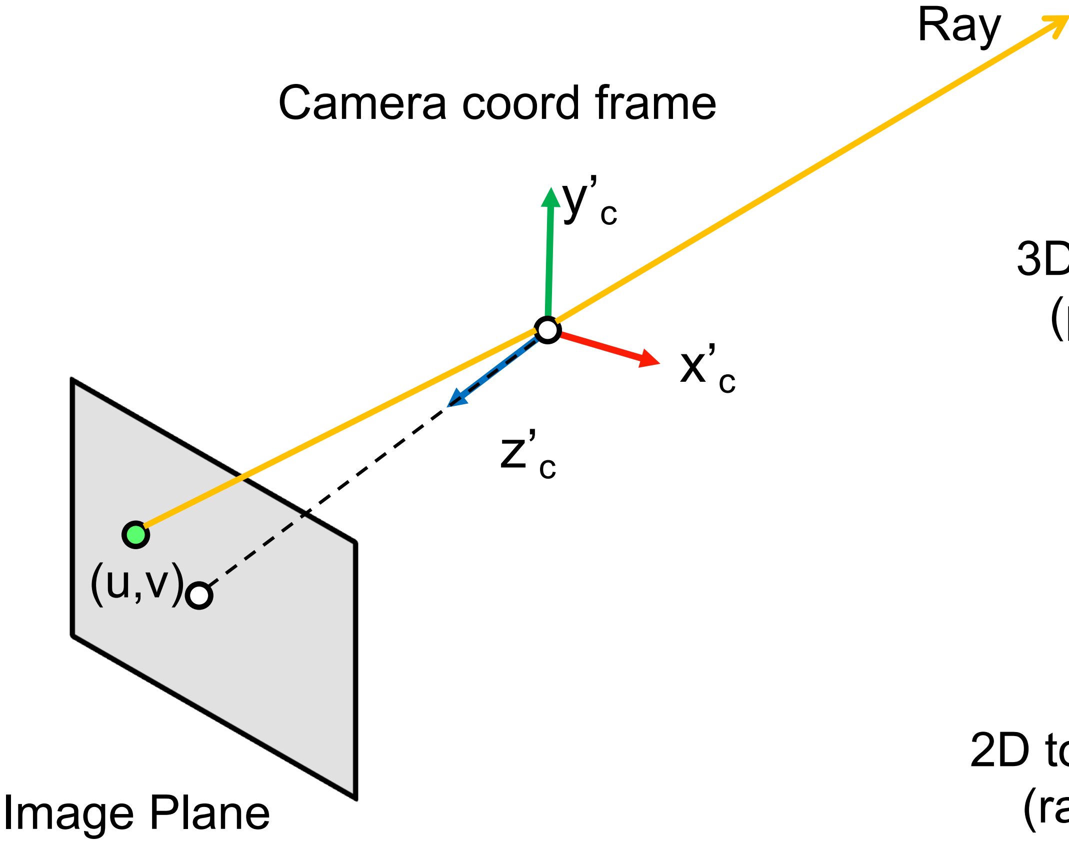
Intrinsics Matrix:

$$K = \begin{bmatrix} f_x & s & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$u = f_x \frac{x_c}{z_c} + o_x \longrightarrow x = \frac{z}{f_x}(u - o_x)$$

What's the problem?

We don’t know the depth!
but at the least it will be:
on the ray!



3D to 2D:
(point)

$$u = f_x \frac{x_c}{z_c} + o_x$$

$$v = f_y \frac{y_c}{z_c} + o_y$$

2D to 3D:
(ray)
Back projection

$$x = \frac{z}{f_x}(u - o_x)$$

$$y = \frac{z}{f_y}(v - o_y)$$

$$z > 0$$

Slide credit: Shree Nayar

What is your coordinate space?

- In Project 5 (and in life) always make sure you're in the right coordinate space.
- eg. Which space is the ray defined in?

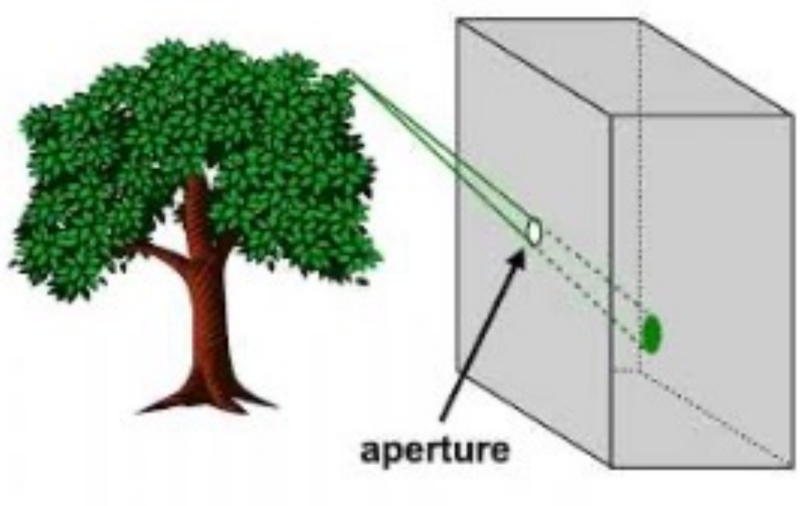
2D to 3D:
(ray)
Back projection

$$x = \frac{z}{f_x}(u - o_x)$$

$$y = \frac{z}{f_y}(v - o_y)$$

$$z > 0$$

Watch these 5 min videos



https://www.youtube.com/watch?v=F5WA26W4JaM
https://www.youtube.com/watch?v=g7Pb8mrwcJ0

Where are my cameras?

How to calibrate the camera?

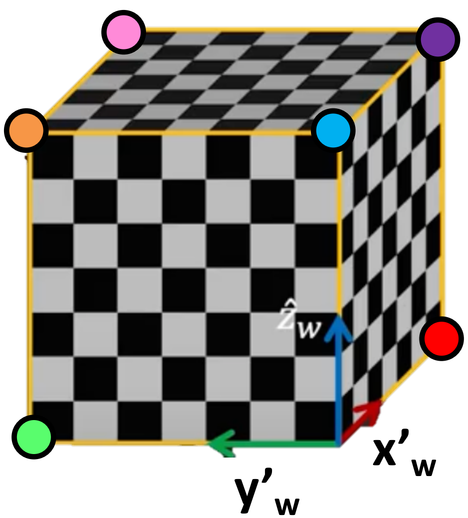
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

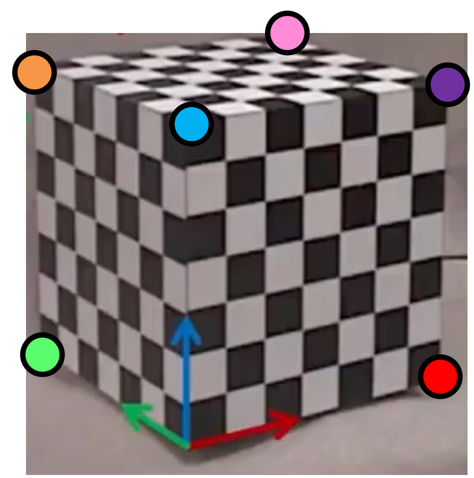
$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

If we know the points in 3D we can estimate the camera!!

Step 1: With a known 3D object

1. Take a picture of an object with known 3D geometry

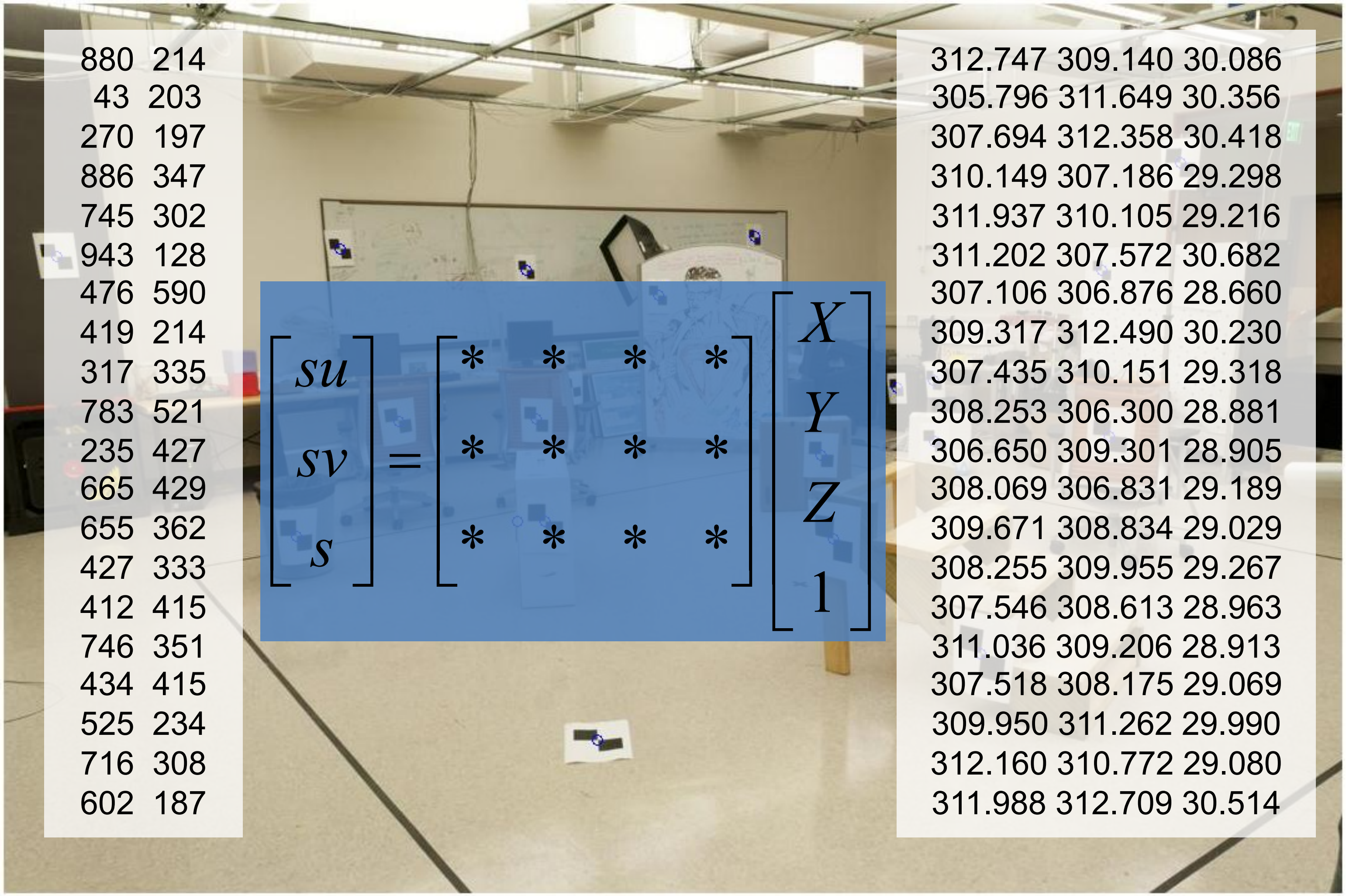




2. Identify correspondences

Slide from Shree Nayar

How do we calibrate a camera?



880	214
43	203
270	197
886	347
745	302
943	128
476	590
419	214
317	335
783	521
235	427
665	429
655	362
427	333
412	415
746	351
434	415
525	234
716	308
602	187

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

312.747	309.140	30.086
305.796	311.649	30.356
307.694	312.358	30.418
310.149	307.186	29.298
311.937	310.105	29.216
311.202	307.572	30.682
307.106	306.876	28.660
309.317	312.490	30.230
307.435	310.151	29.318
308.253	306.300	28.881
306.650	309.301	28.905
308.069	306.831	29.189
309.671	308.834	29.029
308.255	309.955	29.267
307.546	308.613	28.963
311.036	309.206	28.913
307.518	308.175	29.069
309.950	311.262	29.990
312.160	310.772	29.080
311.988	312.709	30.514

Method: Set up a linear system

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Solve for m's entries using linear least squares

Ax=0 form

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Similar to how you solved for homography!

Can we factorize M back to K [R | T]?

- Yes.
- Why? because K and R have a very special form:

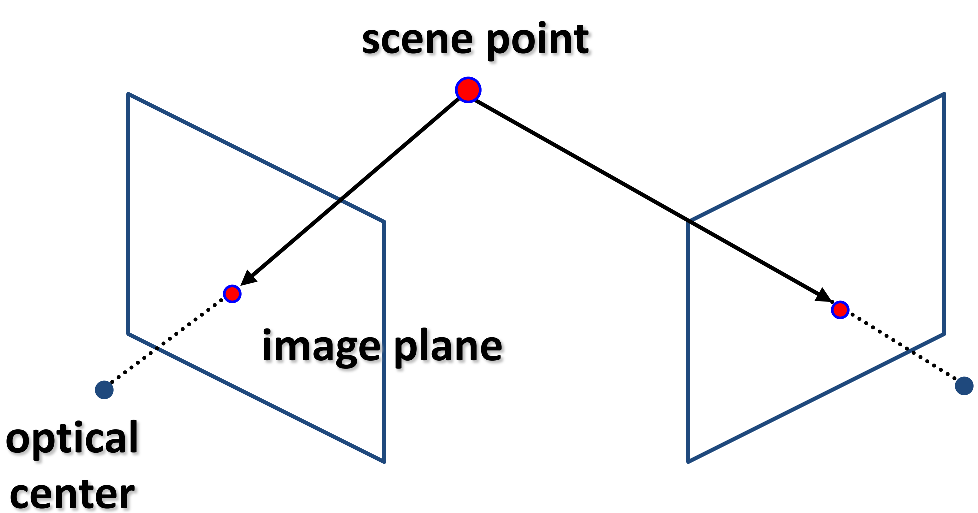
$$\begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

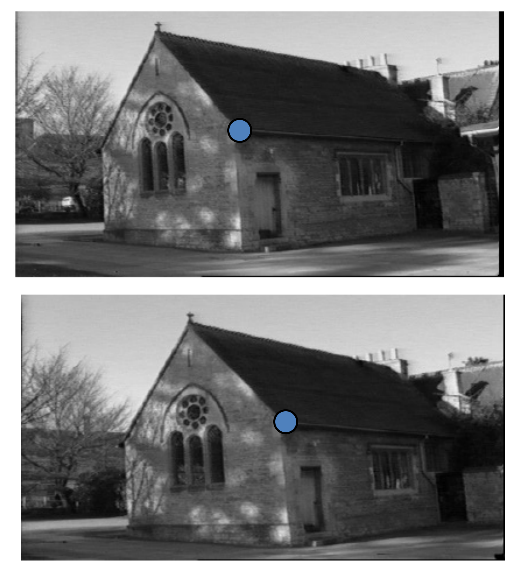
- QR decomposition
- Practically, use camera calibration packages (there is a good one in OpenCV)

Now that our cameras are calibrated, can we find the 3D scene point of a pixel?

Estimating depth with stereo

- **Stereo**: shape from "motion" between **two** views
- We'll need to consider:
  - 1. Camera pose ("calibration")
  - 2. Image point correspondences





Stereo vision

Two cameras, simultaneous views

Single moving camera and static scene

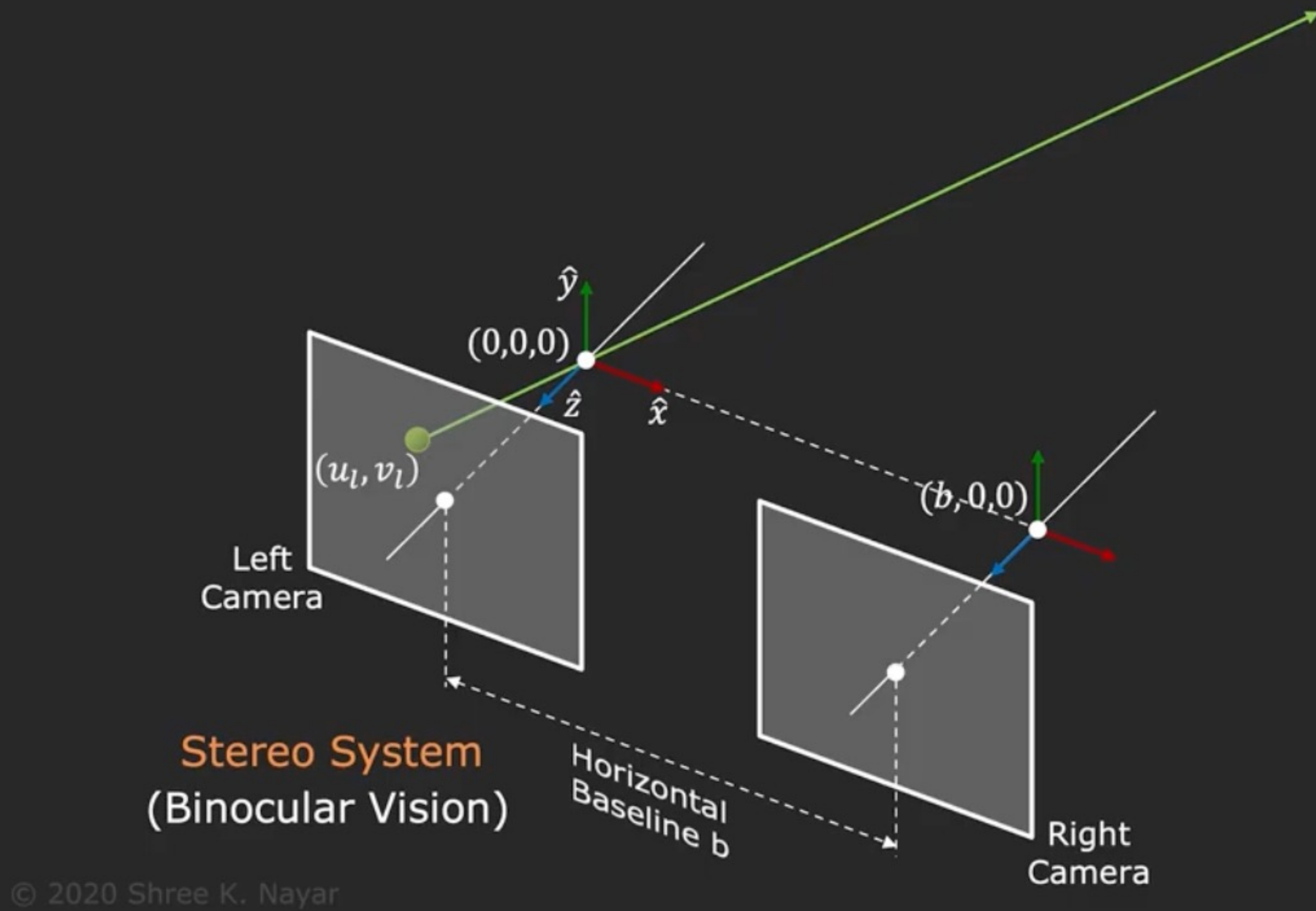
Simple Stereo Setup

- Assume **parallel** optical axes
- Two cameras are calibrated
- Find relative depth

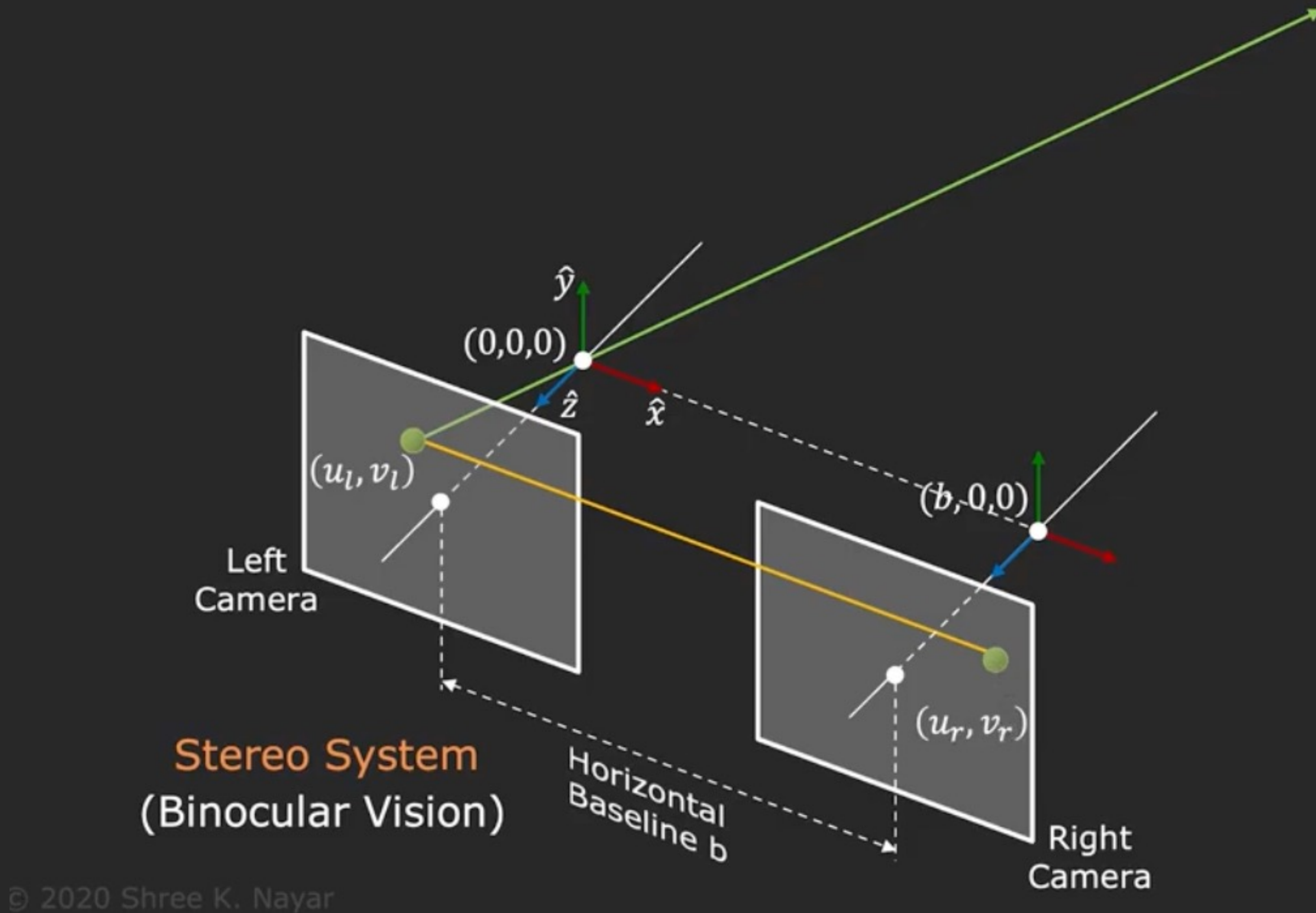
Key Idea: difference in corresponding points to understand shape

Slide credit: Noah Snavely

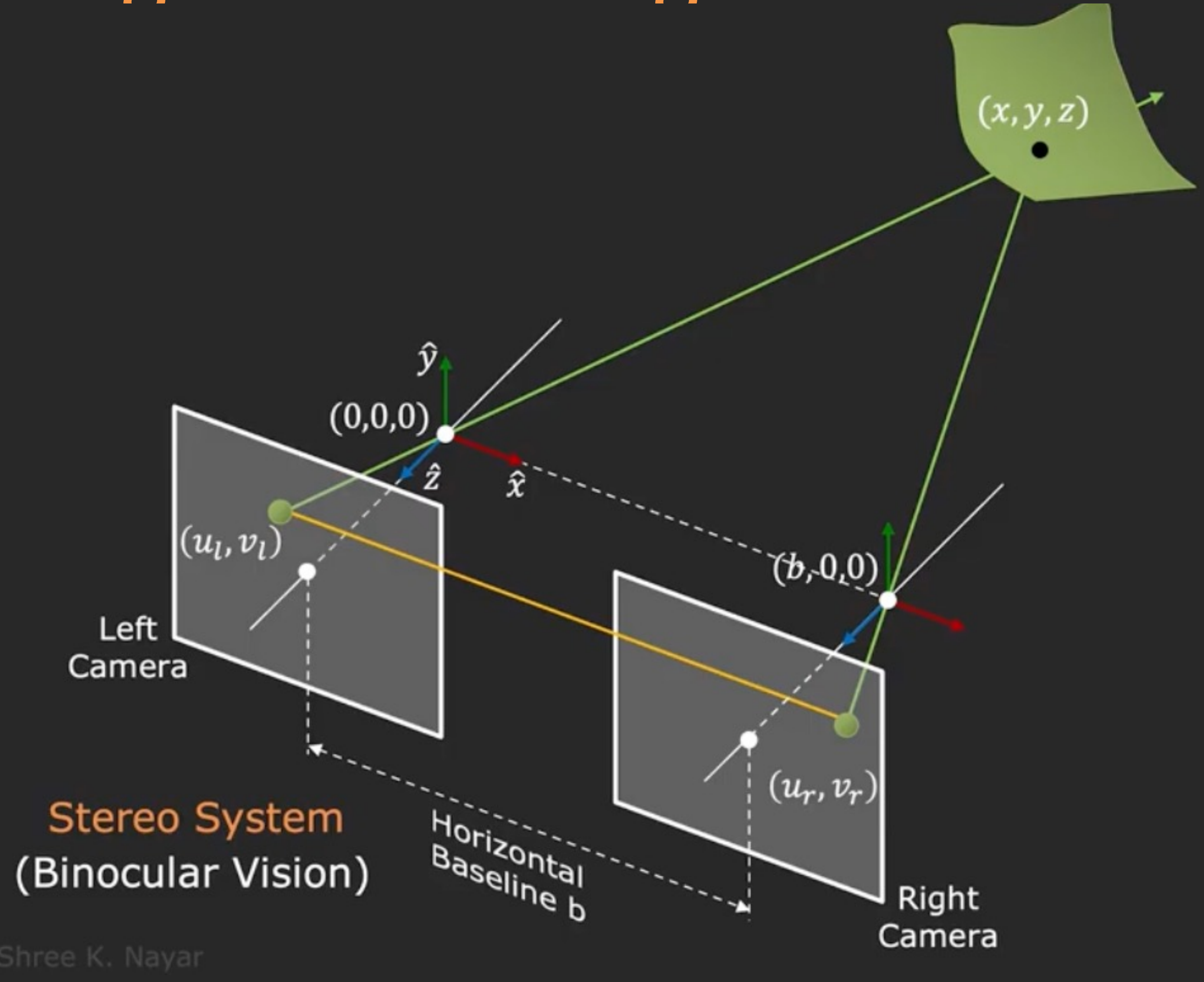
Triangulation using two cameras



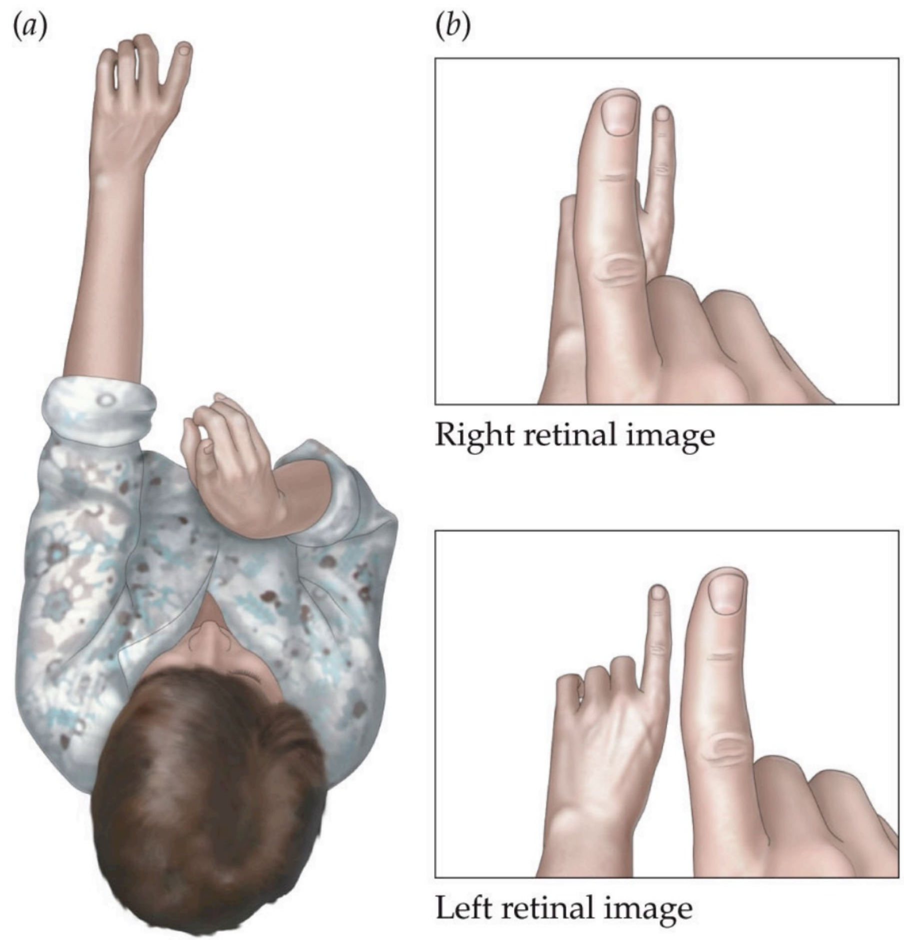
Triangulation using two cameras



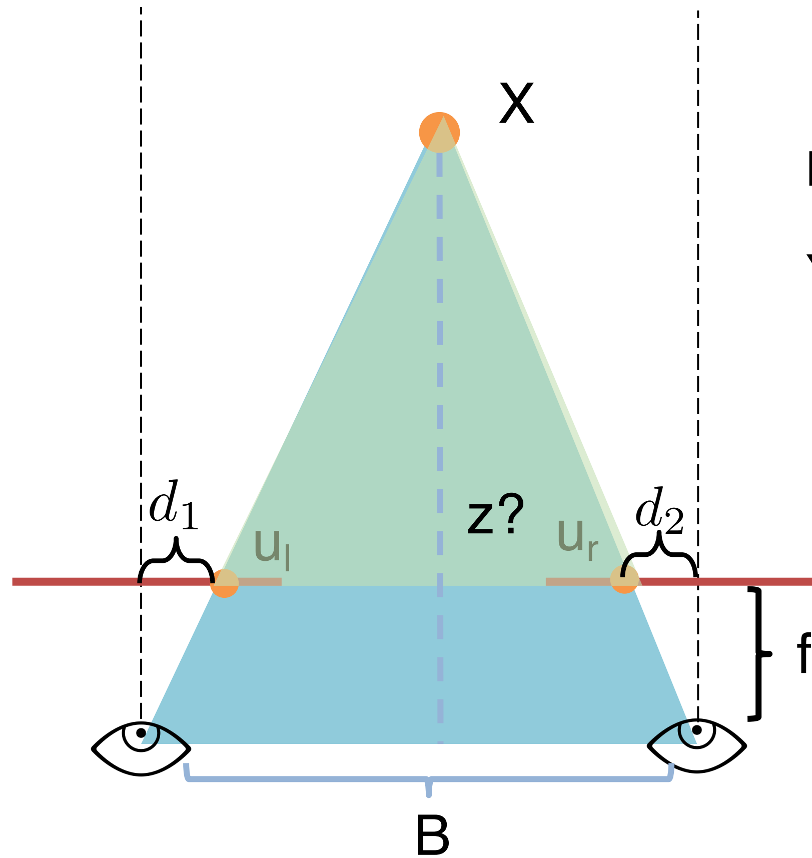
Triangulation using two cameras



We are equipped with binocular vision. Let's try!



Solving for Depth in Simple Stereo



Do we have enough to know what is Z?

Yes, similar triangles!

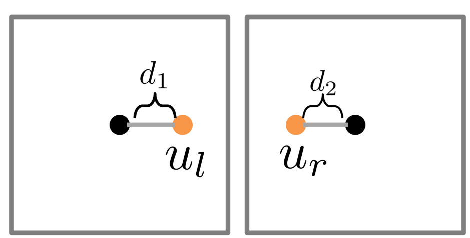
$$\frac{B-(u_l-u_r)}{z-f}=\frac{B}{z}$$

$$z=\frac{fB}{u_l-u_r}$$

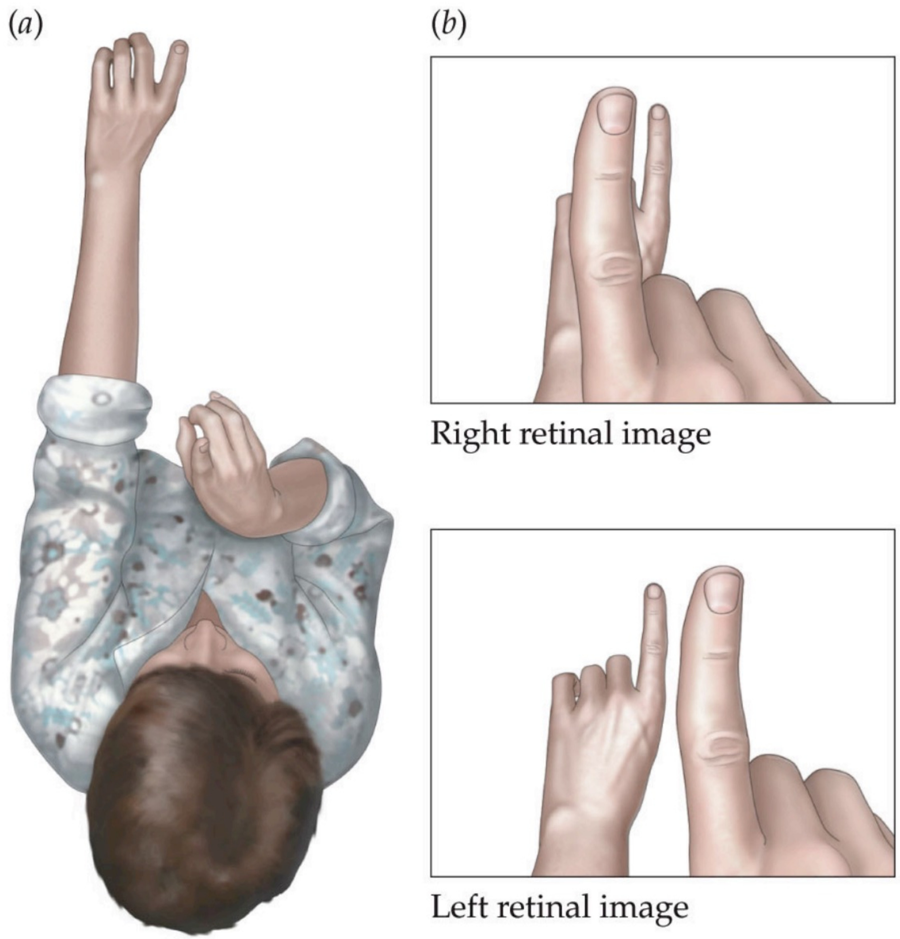
disparity (how much corrsp. pixels move)

Base of : $B-(d1+d_2)$

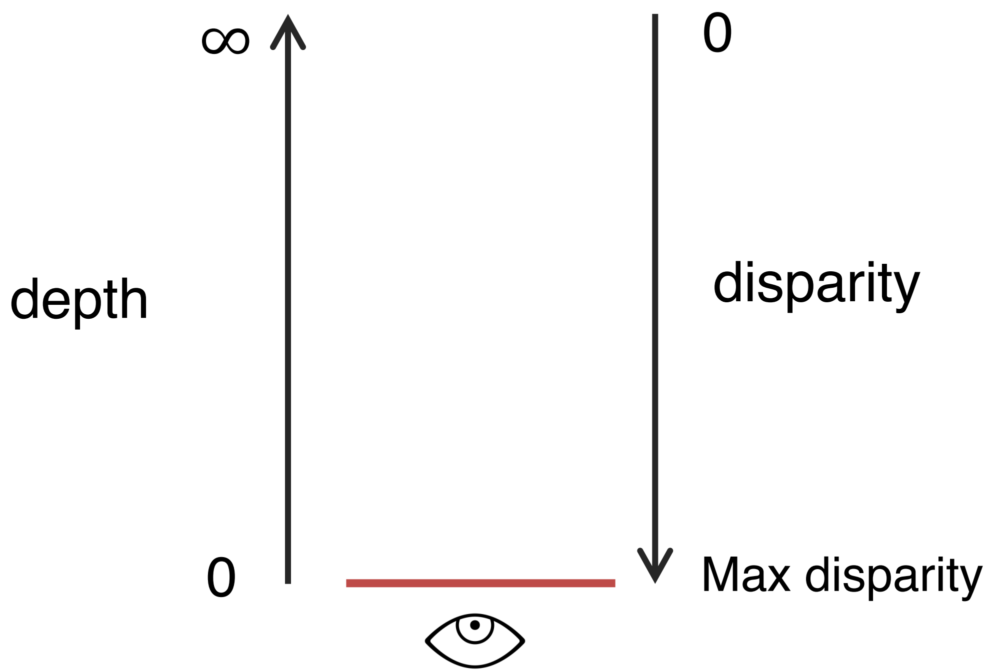
in image coordinates: $=B-(u_l-u_r)$



Try with your hands!



Depth is inversely proportional to disparity



$$z = \frac{fB}{u_l - u_r}$$

disparity

$$z \propto \frac{1}{u_l - u_r} = \frac{1}{d}$$

what is the disparity of the closer point?
what is the disparity of the far away point?

Disparity gives you the depth information!

Try again

1. Setup so your fingers are on the same line of sight from one eye
2. Now look in the other eye

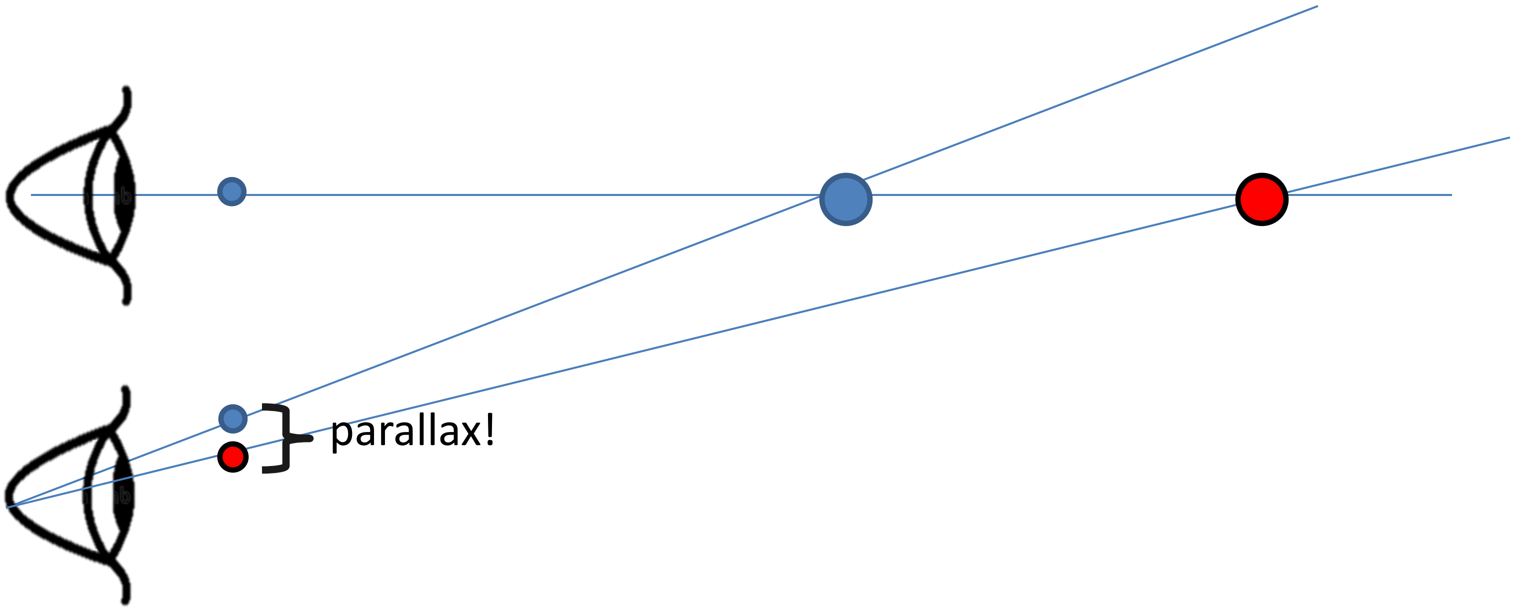
They move!

Relative displacement is higher as the relative distance grows

== Parallax


optical
center
5 px
50 cm
f = 10 mm

Parallax



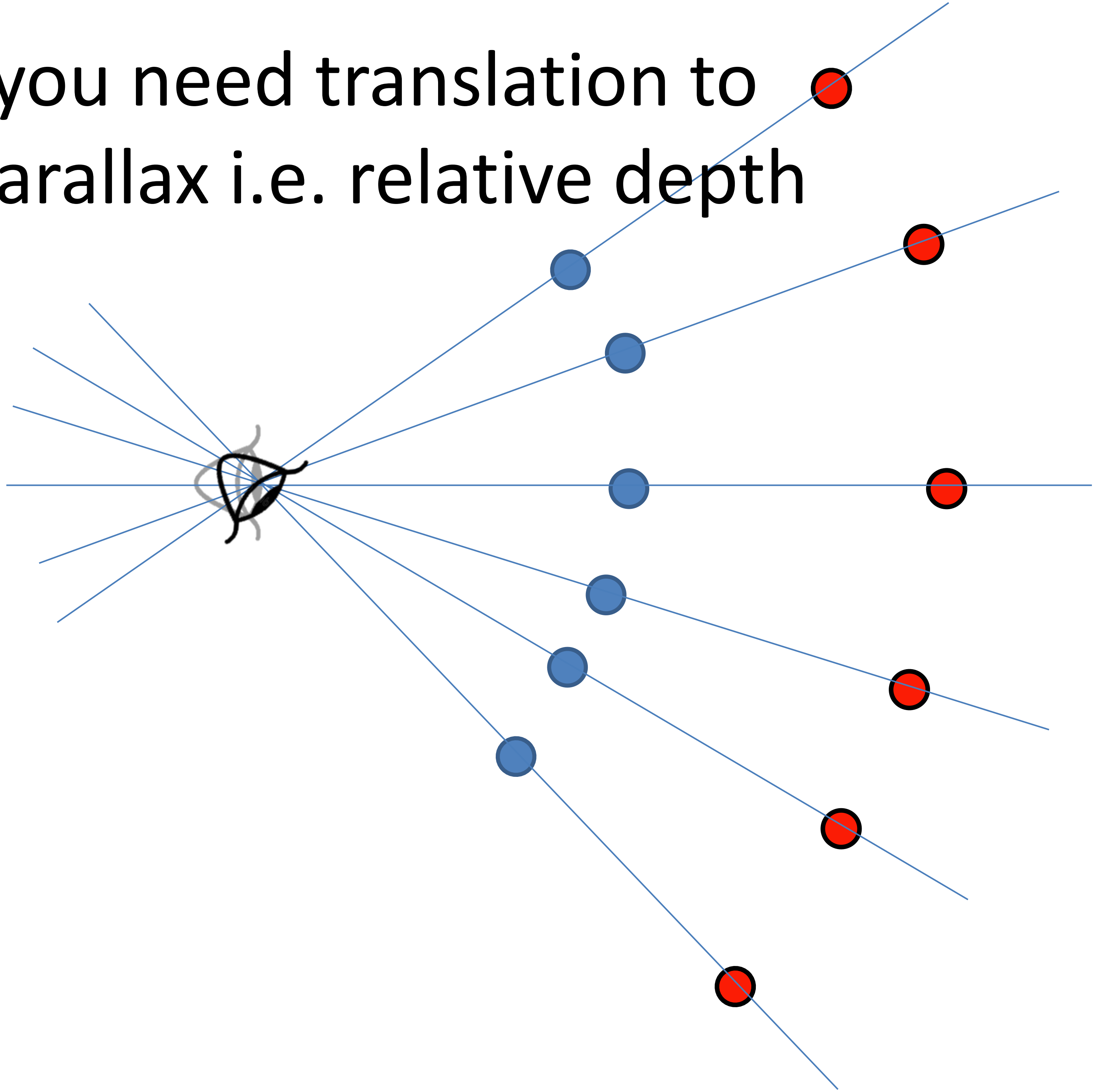
Parallax = *from ancient Greek parállaxis*

= *Para (*side by side) + *allássō*, (to alter)

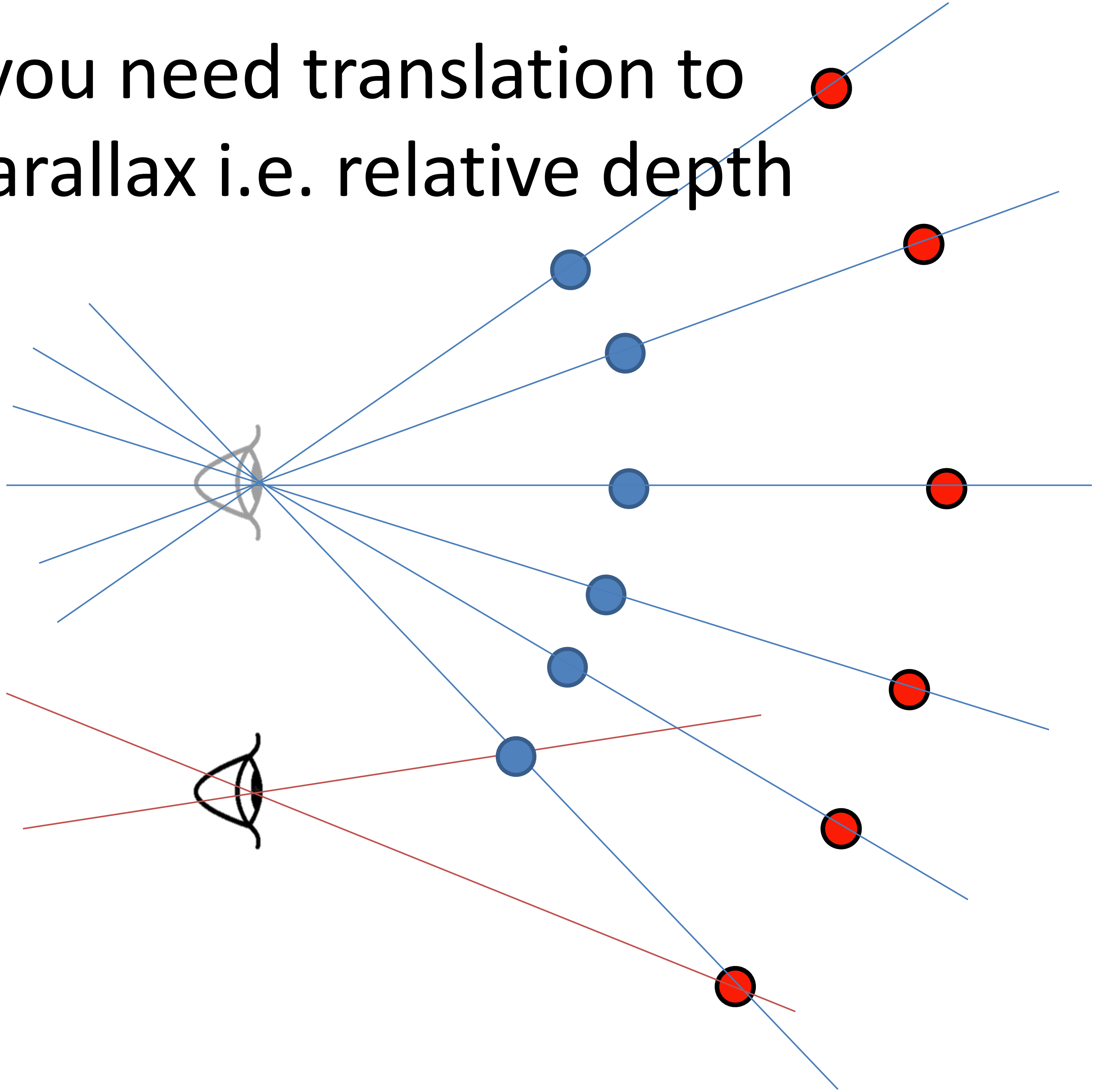
= *Change in position from different view point*

Two eyes give you parallax, you can also move to see more parallax = "Motion Parallax"

Why you need translation to see parallax i.e. relative depth



Why you need translation to see parallax i.e. relative depth



Stereo Matching: Finding Disparities

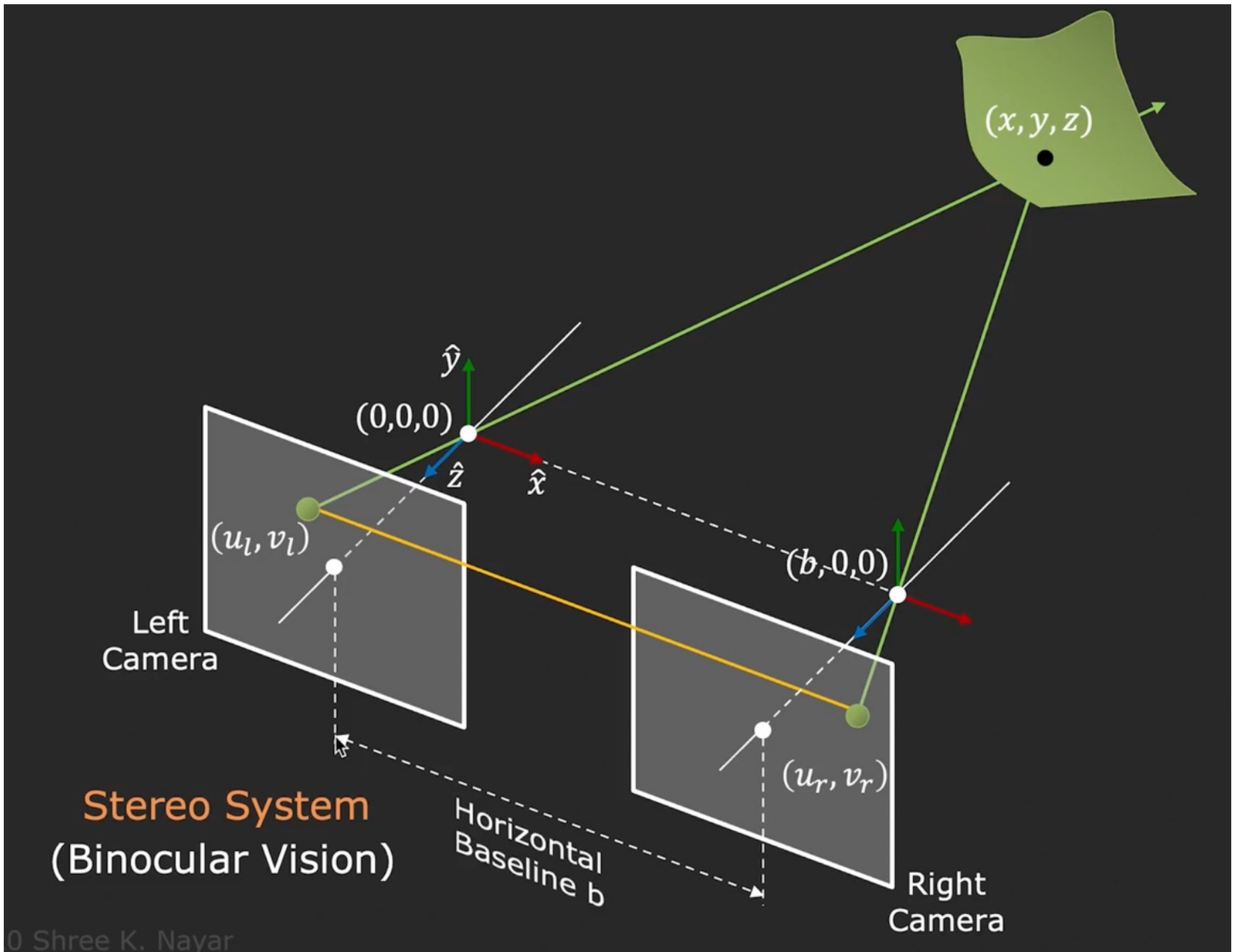
Goal: Find the disparity between left and right stereo pairs.

Left/Right Camera Images

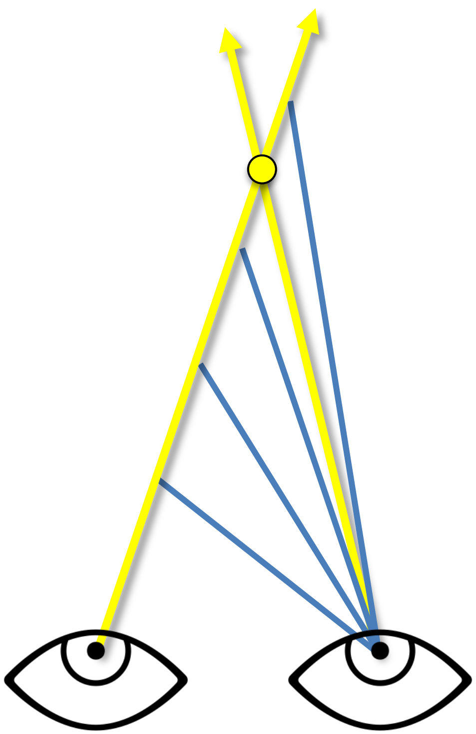
Disparity Map (Ground Truth)

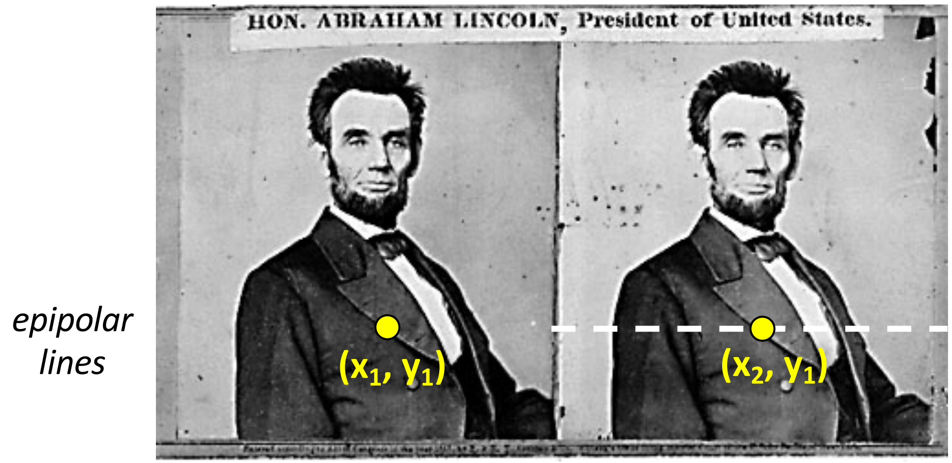
Where is the corresponding point going to be?

Hint



Epipolar Line

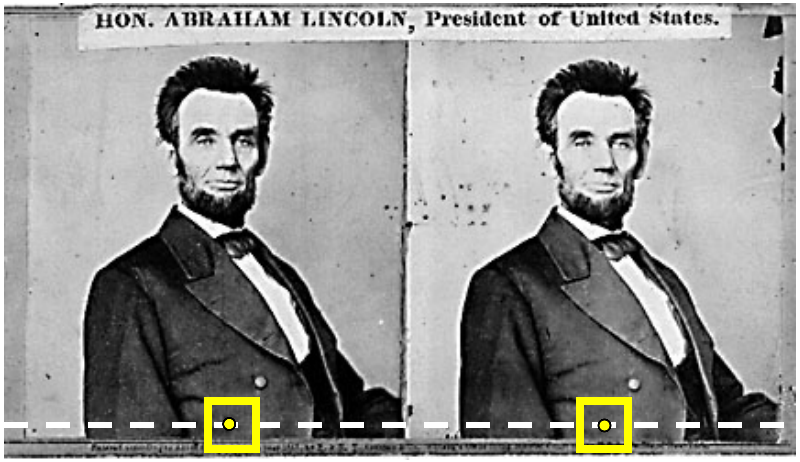




Two images captured by a purely horizontal translating camera (*rectified* stereo pair)

$x_1 - x_2$ = the *disparity* of pixel (x_1, y_1)

Your basic stereo algorithm



For every epipolar line:

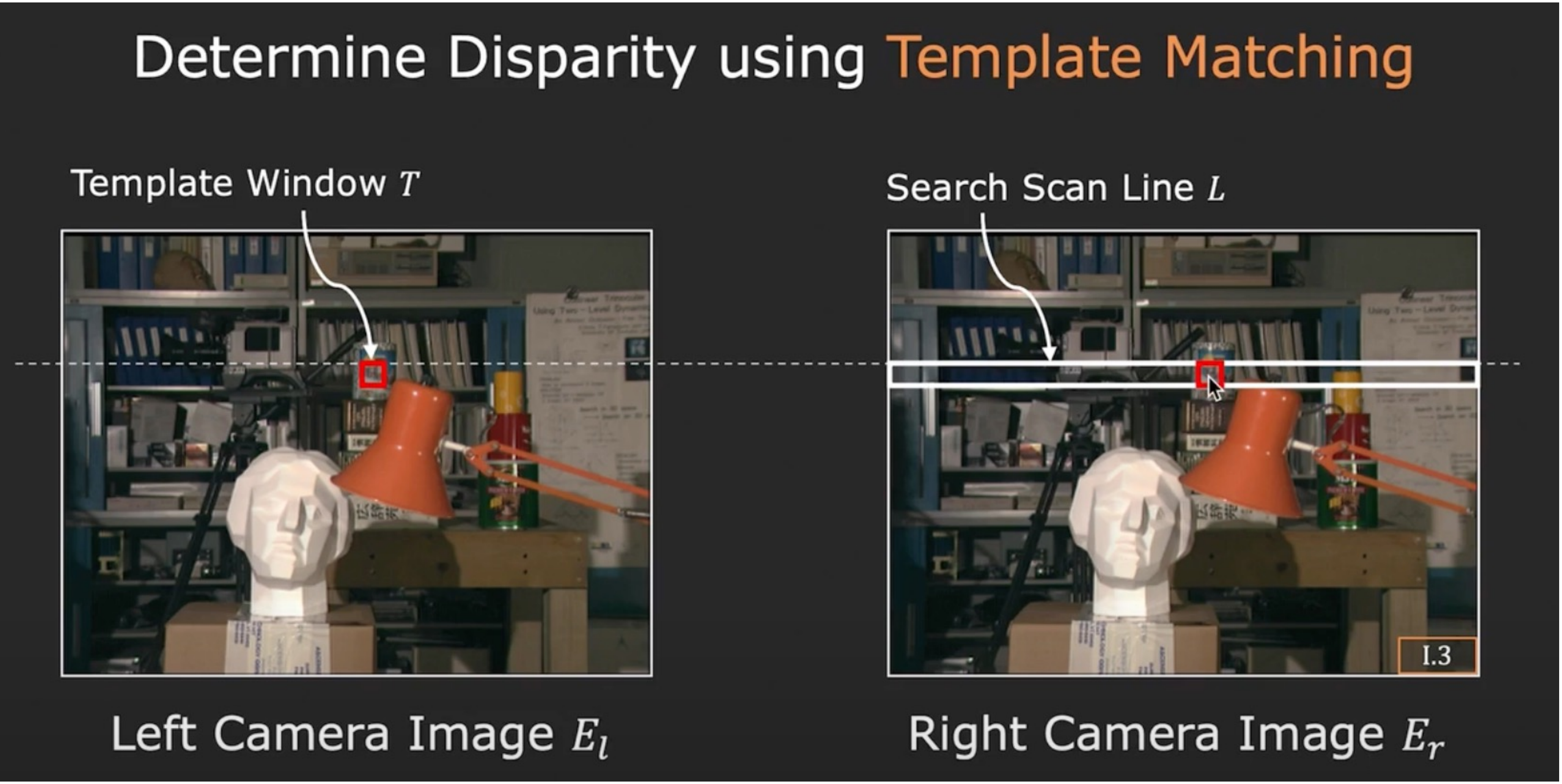
For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows,*** *+ clearly lots of matching strategies*

Slide credit: Noah Snavely

Your basic stereo algorithm

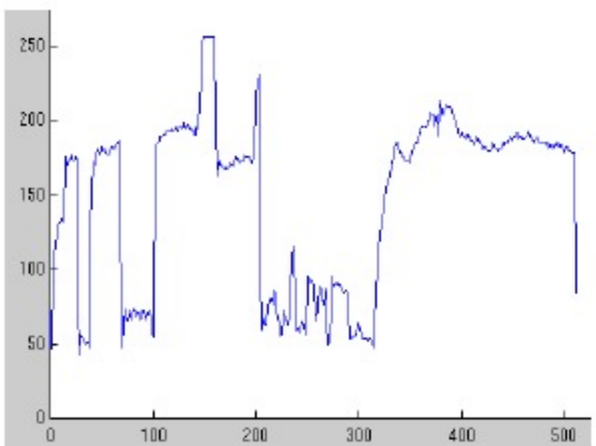


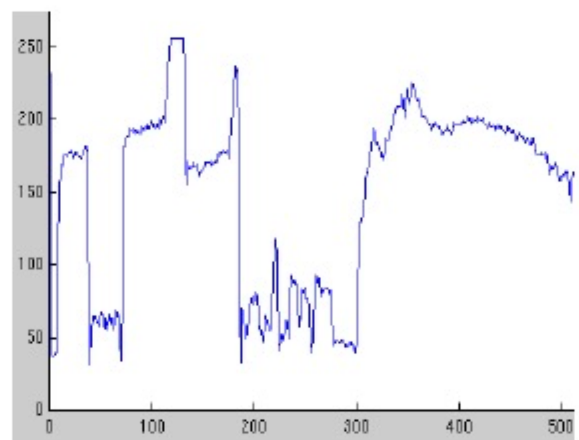
Correspondence problem

Parallel camera example – epipolar lines are corresponding rasters

Source: Andrew Zisserman

Intensity profiles





- Clear correspondence between intensities, but also noise and ambiguity

Source: Andrew Zisserman

Correspondence problem

Neighborhood of corresponding points are similar in intensity patterns.

Source: Andrew Zisserman

Normalized cross correlation

subtract mean: $A \leftarrow A - < A >, B \leftarrow B - < B >$

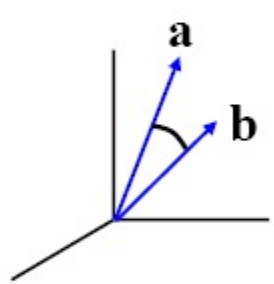
$$\text{NCC} = \frac{\sum_i \sum_j A(i,j)B(i,j)}{\sqrt{\sum_i \sum_j A(i,j)^2}\sqrt{\sum_i \sum_j B(i,j)^2}}$$

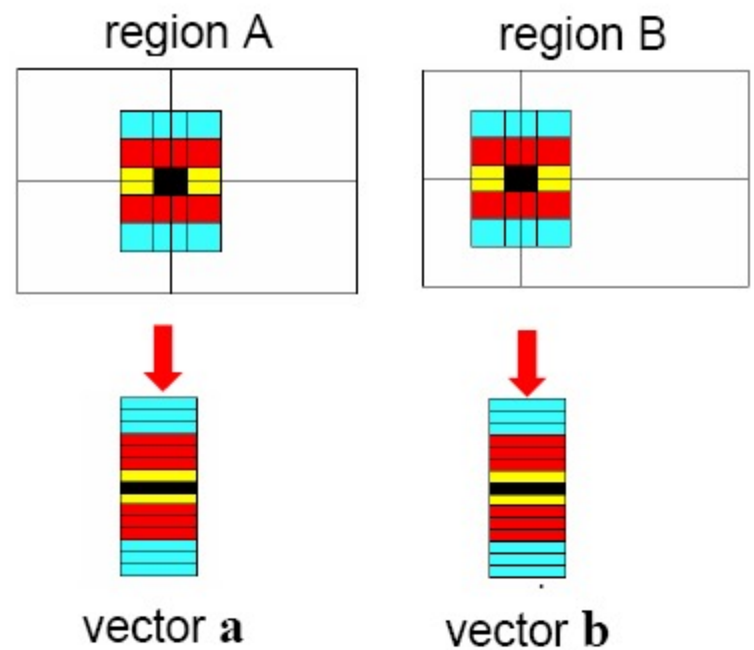
Write regions as vectors

$$\mathtt{A} \rightarrow \mathbf{a},\ \mathtt{B} \rightarrow \mathbf{b}$$

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

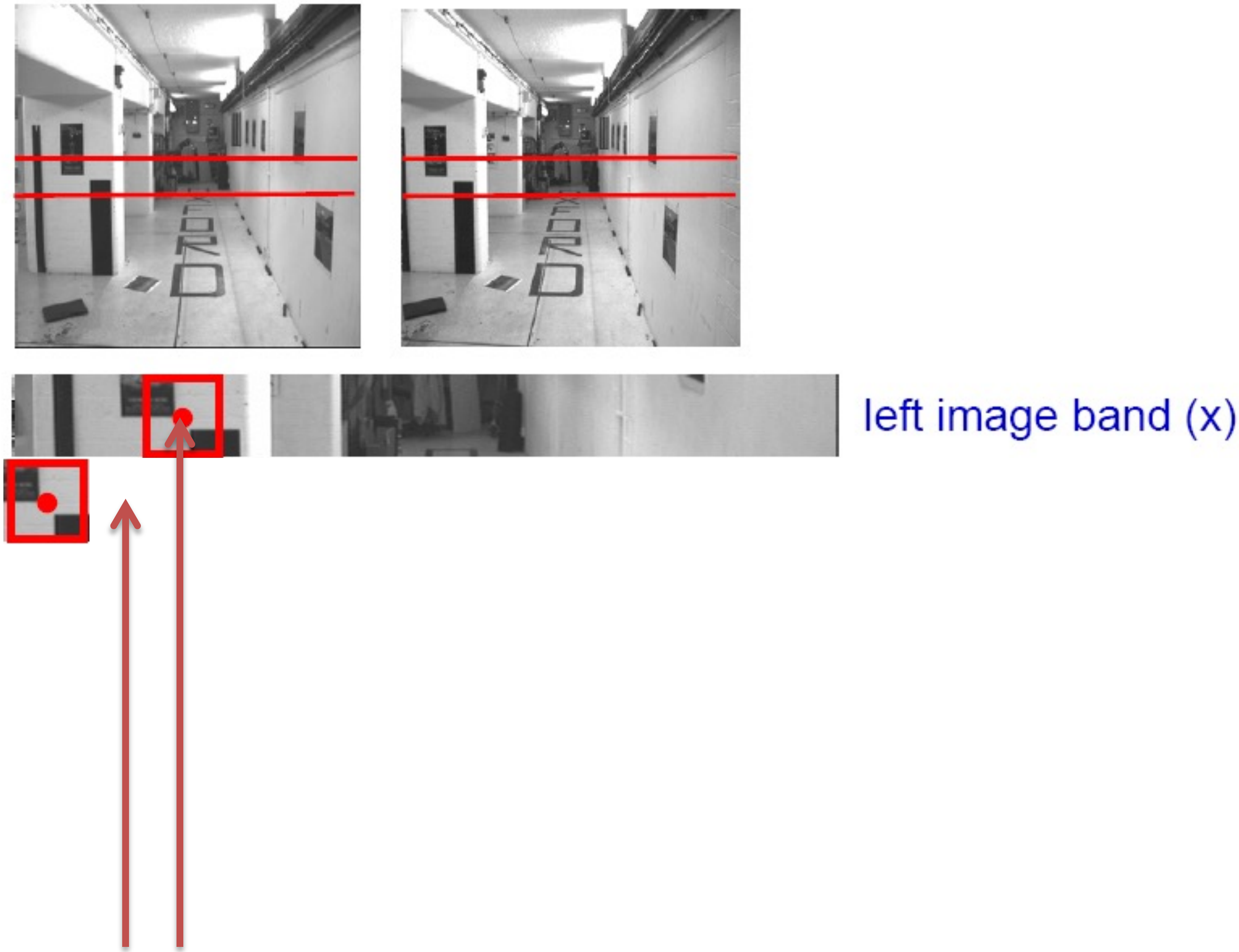
$$-1 \leq \text{NCC} \leq 1$$





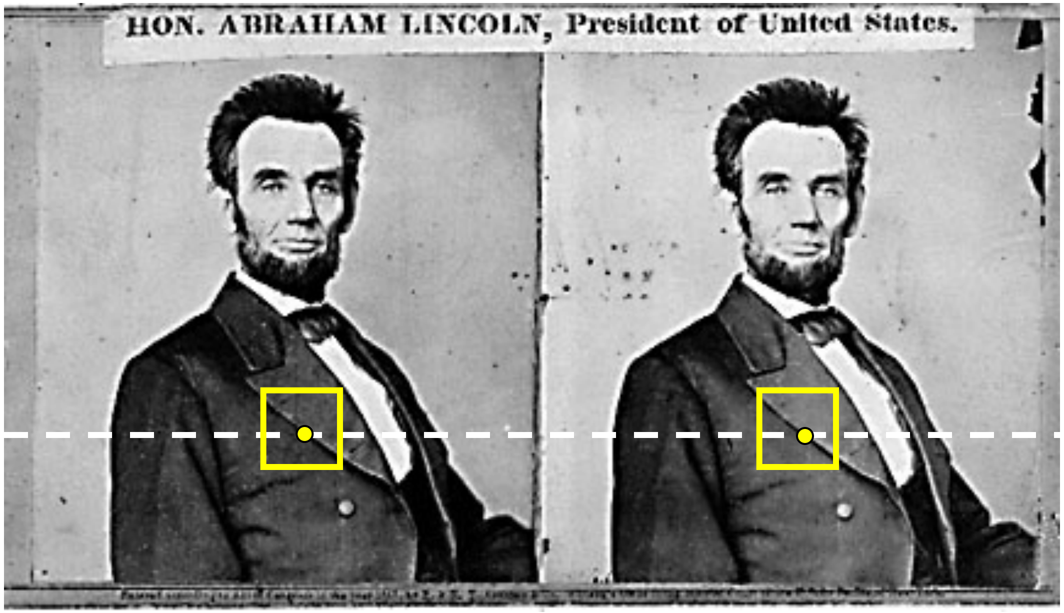
Source: Andrew Zisserman

Correlation-based window matching



Source: Andrew Zisserman

Dense correspondence search



For each epipolar line

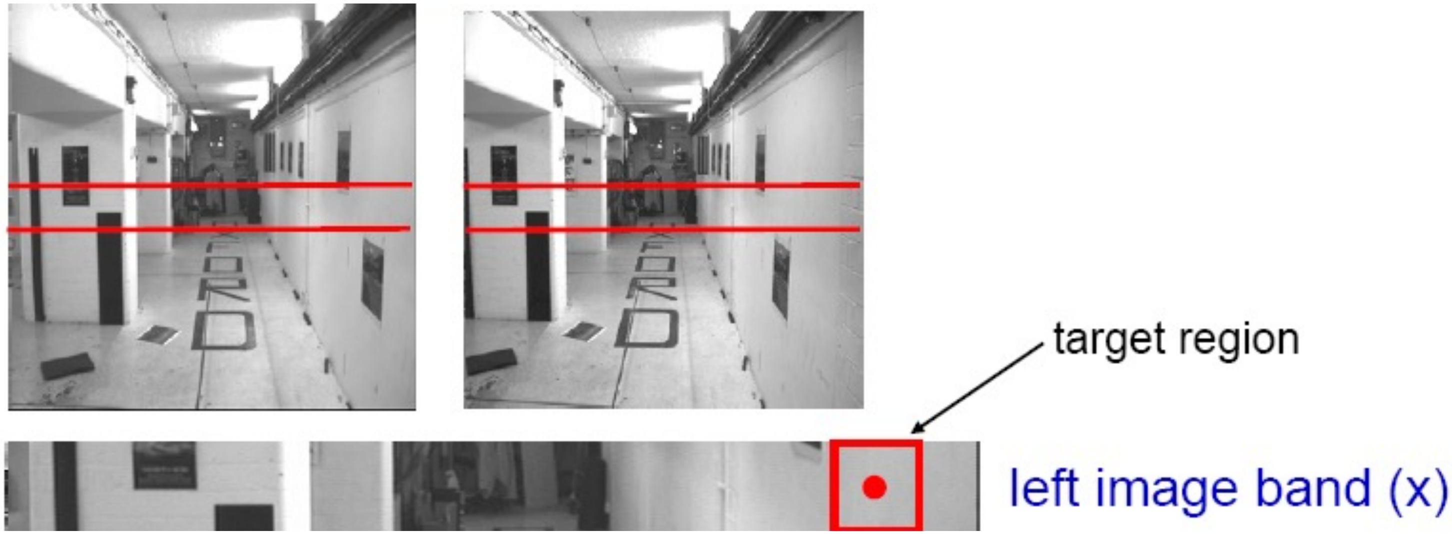
For each pixel / window in the left image

- compare with every pixel / window on same epipolar line in right image
- pick position with minimum match cost (e.g., SSD, correlation)

Adapted from Li Zhang

Grauman

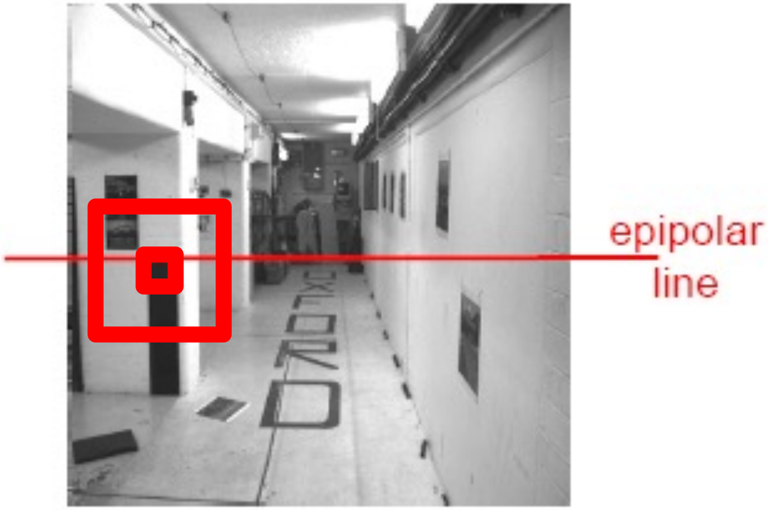
Textureless regions



Source: Andrew Zisserman

Grauman

Effect of window size



Source: Andrew Zisserman

Effect of window size

W = 3 W = 20

Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same disparity.

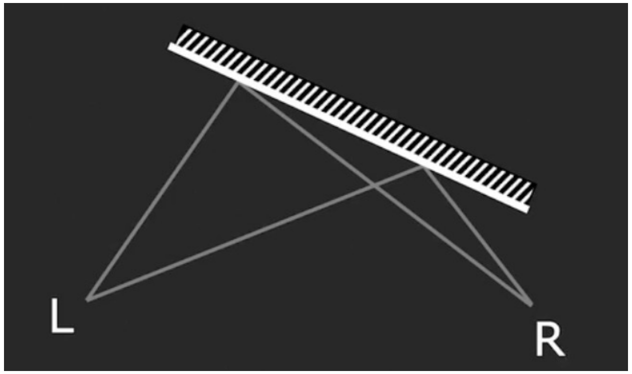
Figures from Li Zhang

Grauman

Issues with Stereo

- Surface must have non-repetitive texture

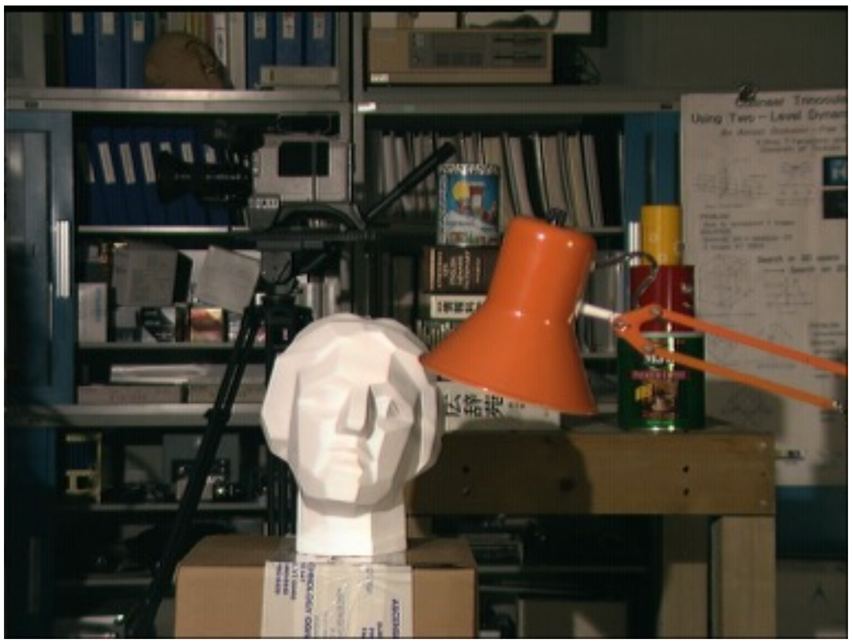
- Foreshortening effect makes matching a challenge



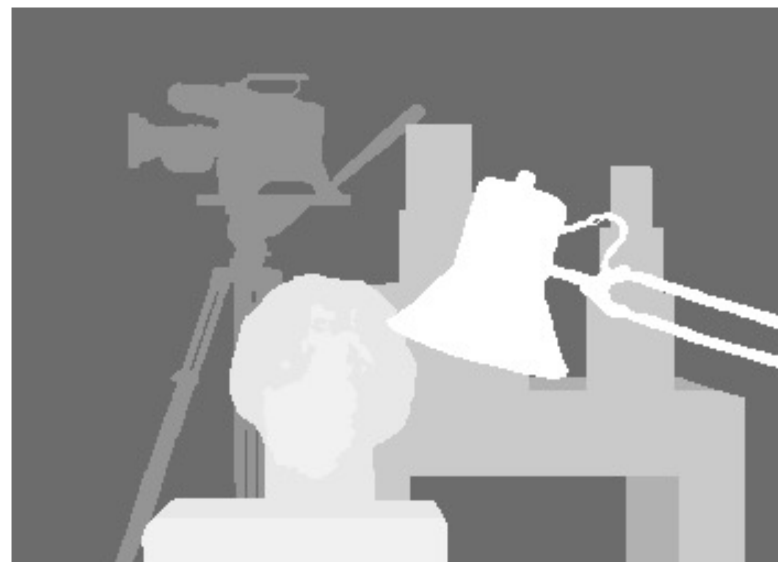
Slide Credit: Shree Nayar

Stereo Results

- Data from University of Tsukuba

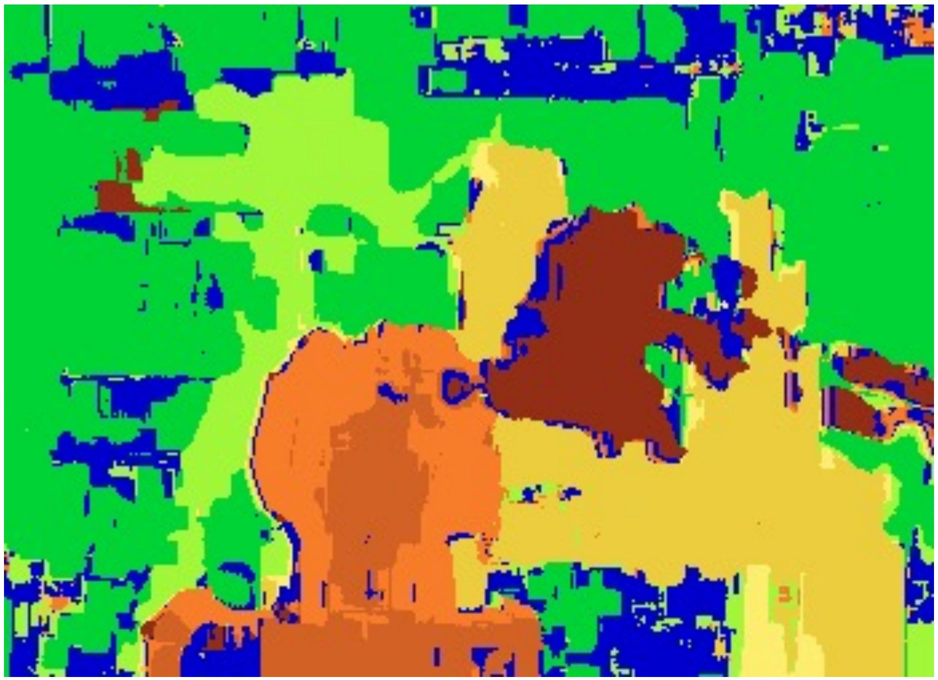


Scene



Ground truth

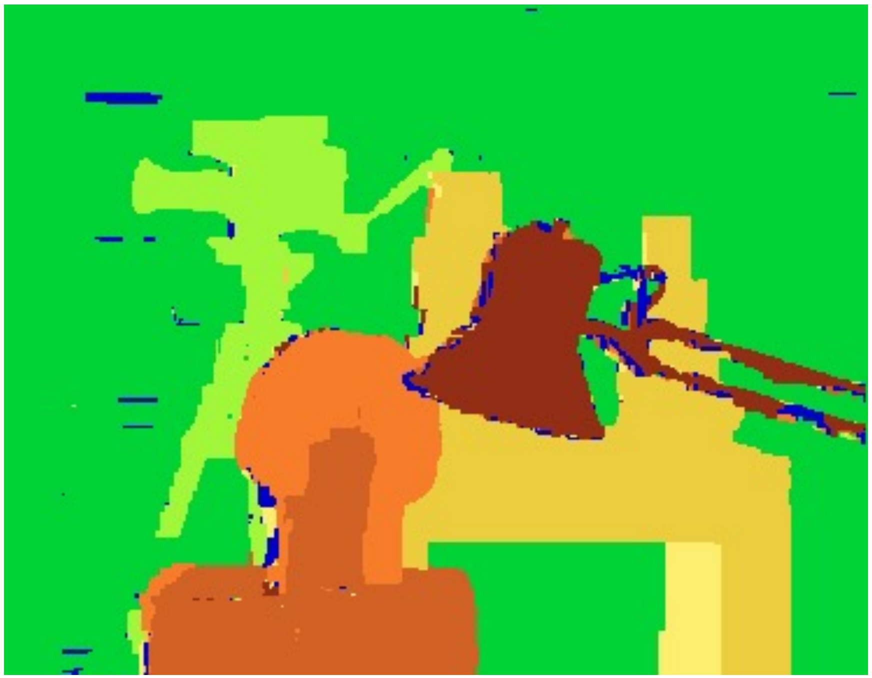
Results with Window Search



Window-based matching
(best window size)

Ground truth

Better methods exist...



Energy Minimization

Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision, September 1999.

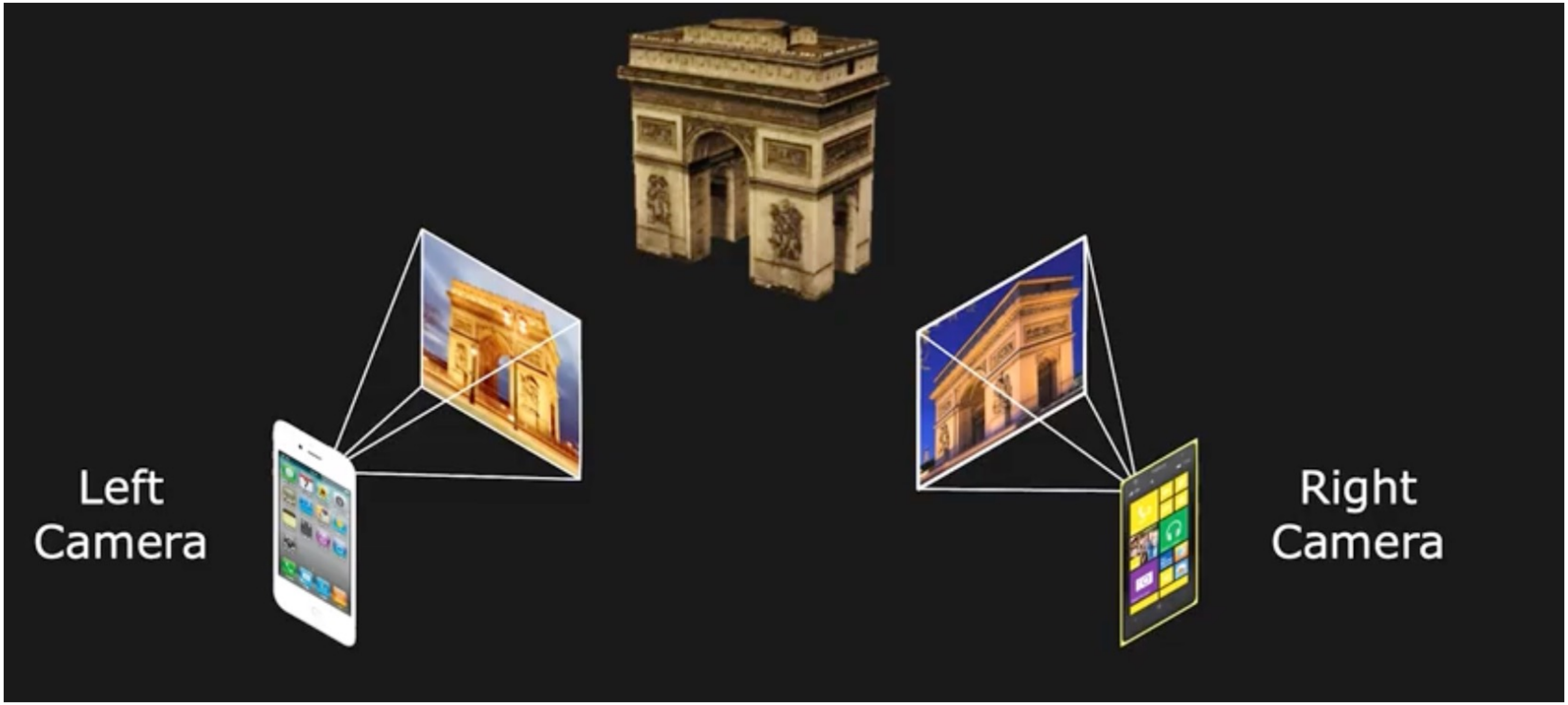
Ground truth

Summary

- With a simple stereo system, **how much pixels move, or "disparity"** give information about the depth
- Correspondences to measure the pixel disparity

Next: Uncalibrated Stereo

- From two arbitrary views



- Assume intrinsics are known (fx, fy, ox, oy)

Slide Credit: Shree Nayar