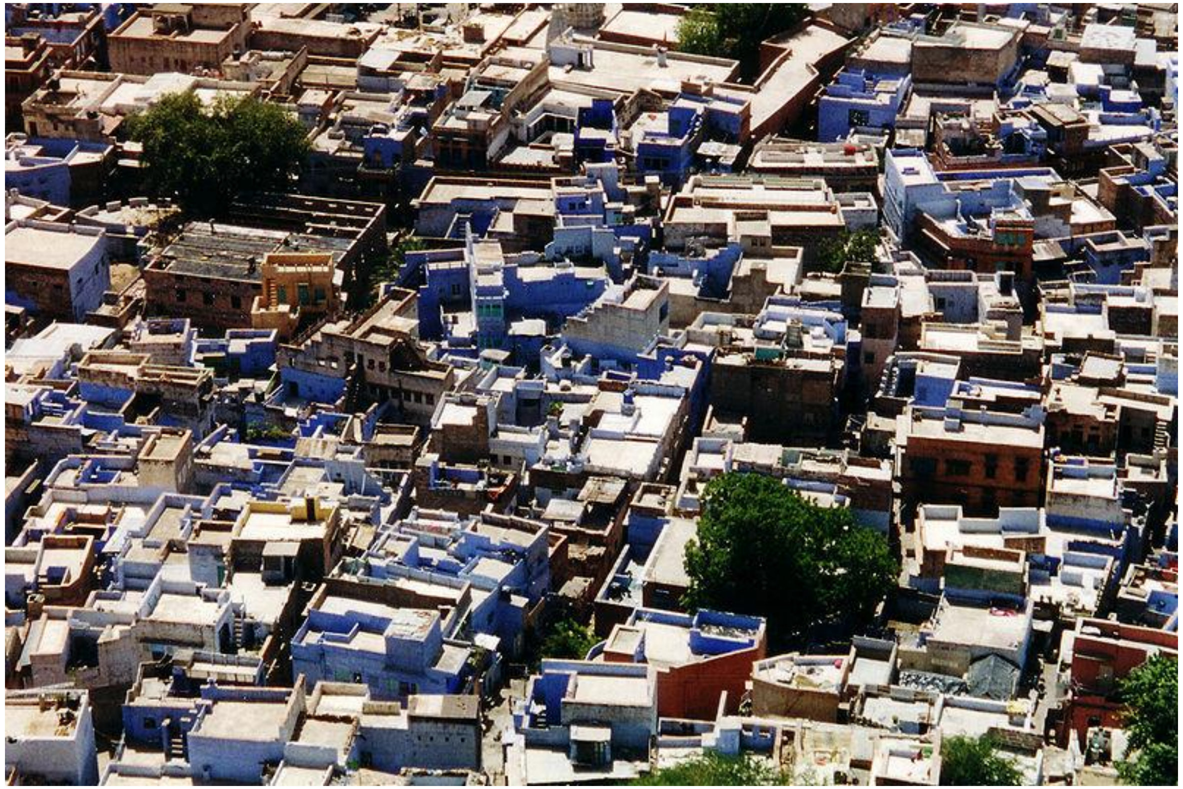
# Miniature faking

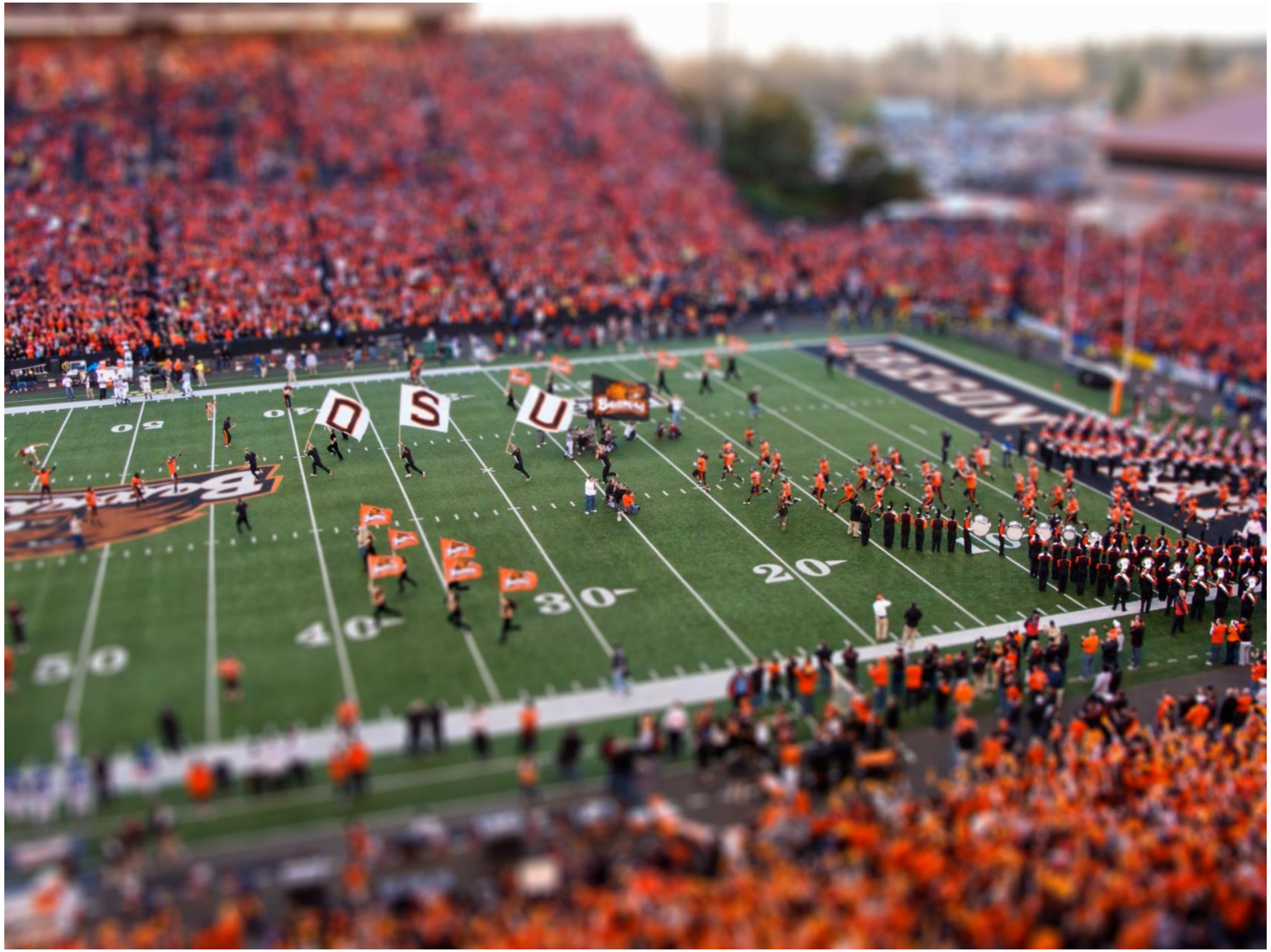
In close-up photo, the depth of field is limited.

http://en.wikipedia.org/wiki/File:Jodhpur_tilt_shift.jpg

# Miniature faking



# Miniature faking

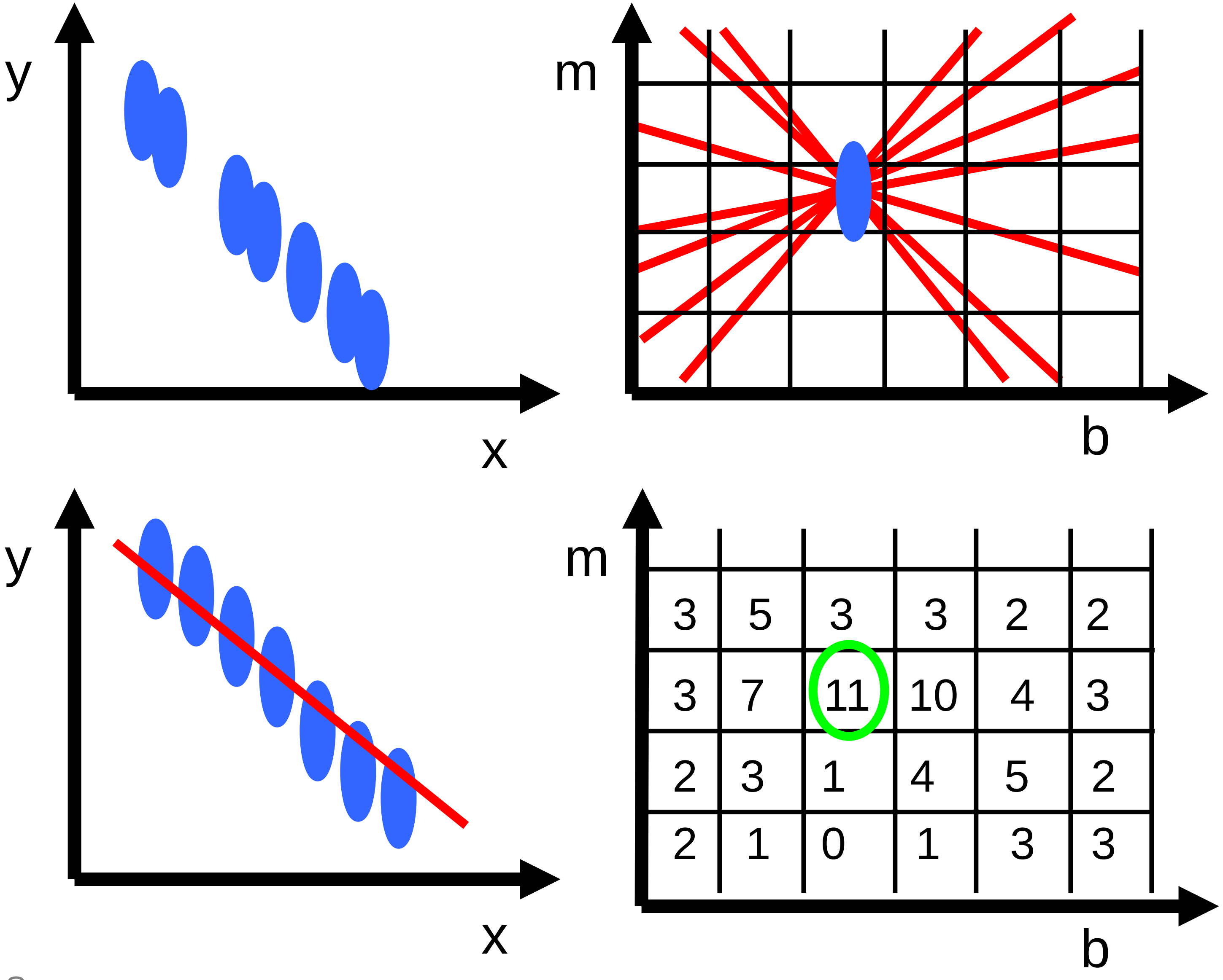



http://en.wikipedia.org/wiki/File:Oregon_State_Beavers_Tilt-Shift_Miniature_Greg_Keene.jpg

# Review

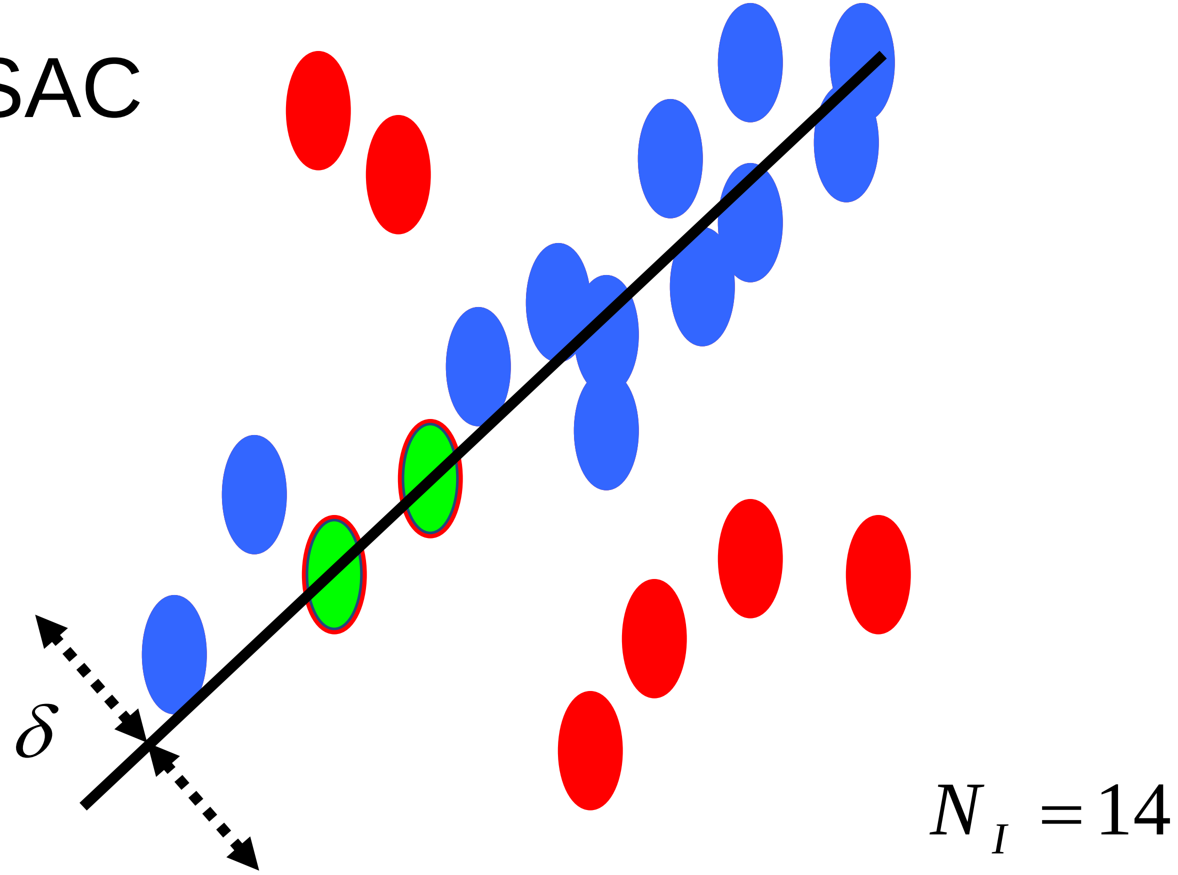
- Previous section:
  - Model fitting and outlier rejection

# Review: Hough transform




Slide from S. Savarese

# Review: RANSAC

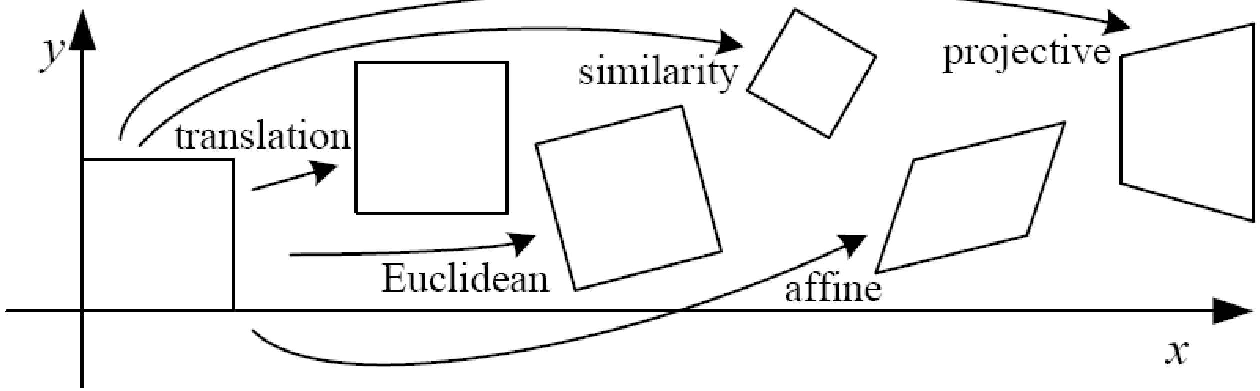



Algorithm:

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

# Review: 2D image transformations



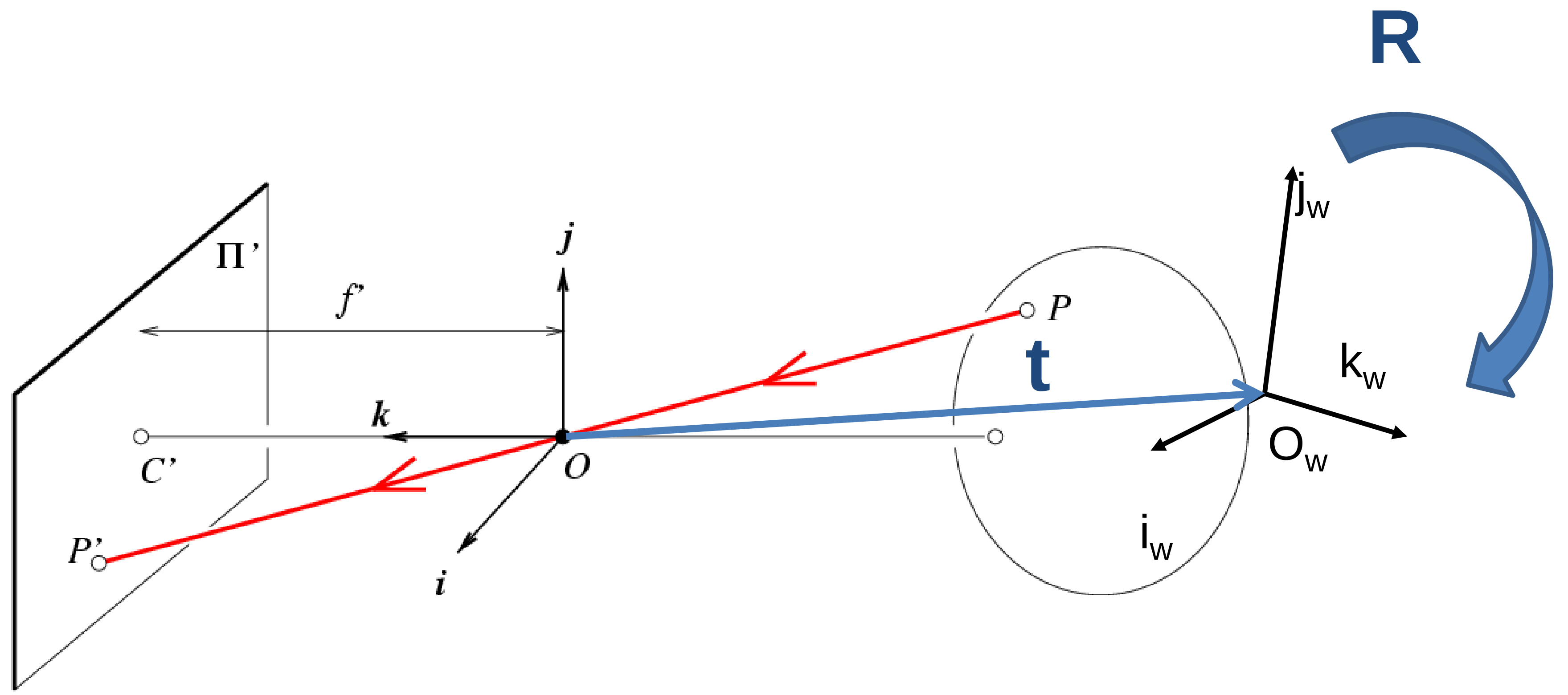

| Name | Matrix | # D.O.F. | Preserves: | Icon |
|---|---|---|---|---|
| translation | $\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$ | 2 | orientation $+ \cdots$ | |
| rigid (Euclidean) | $\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$ |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

Szeliski 2.1

# This section – multiple views

- Today – Camera Calibration. Intro to multiple views and Stereo.
- Next Lecture – Epipolar Geometry and Fundamental Matrix. Stereo Matching (if there is time).
- Both lectures are relevant for project 3.

# Recap: Oriented and Translated Camera



# Recap: Degrees of freedom

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

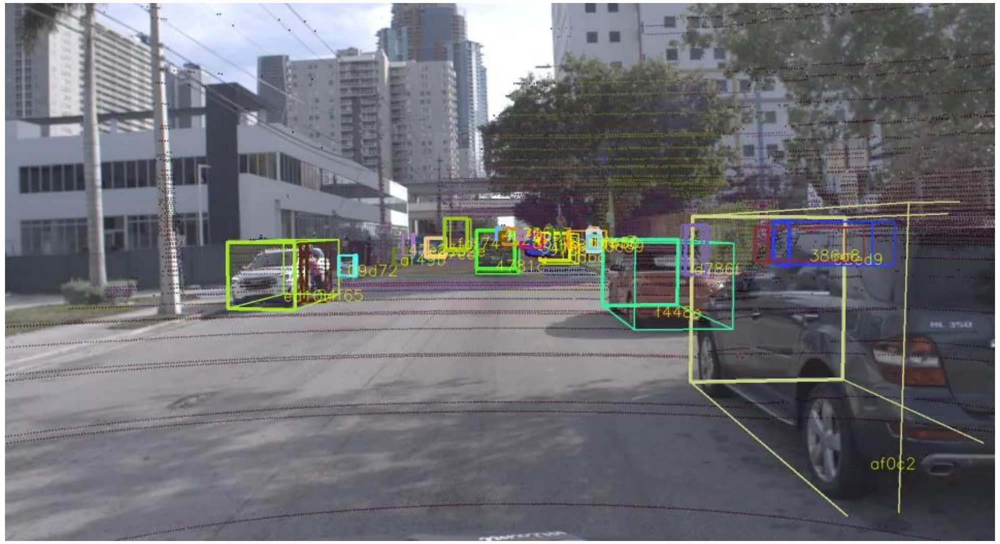
$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overset{5}{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{6}{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

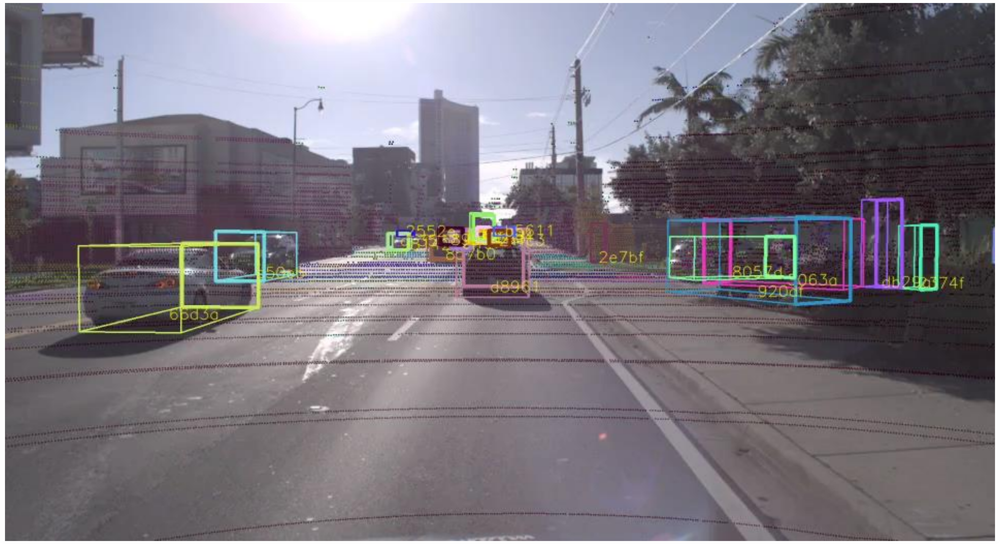
# This Lecture: How to calibrate the camera?

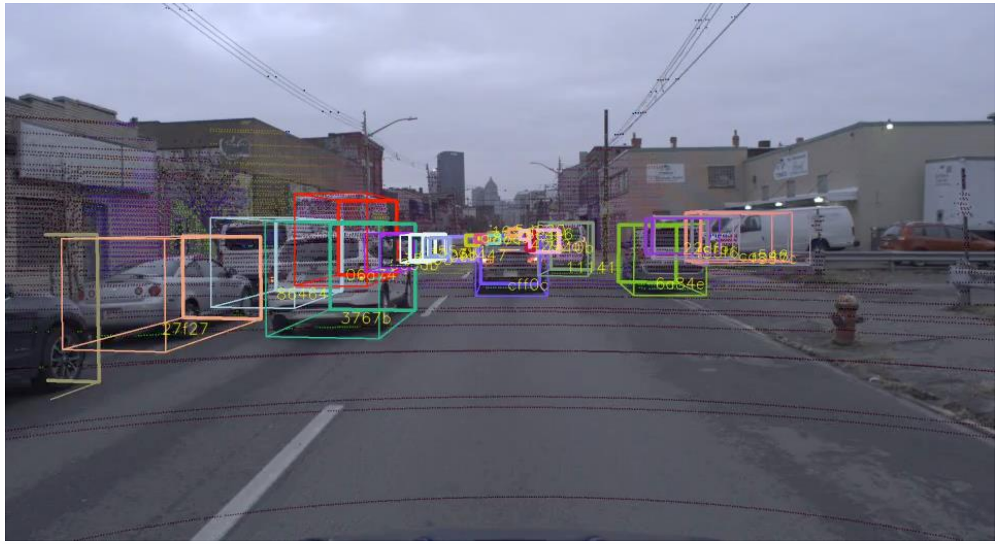
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

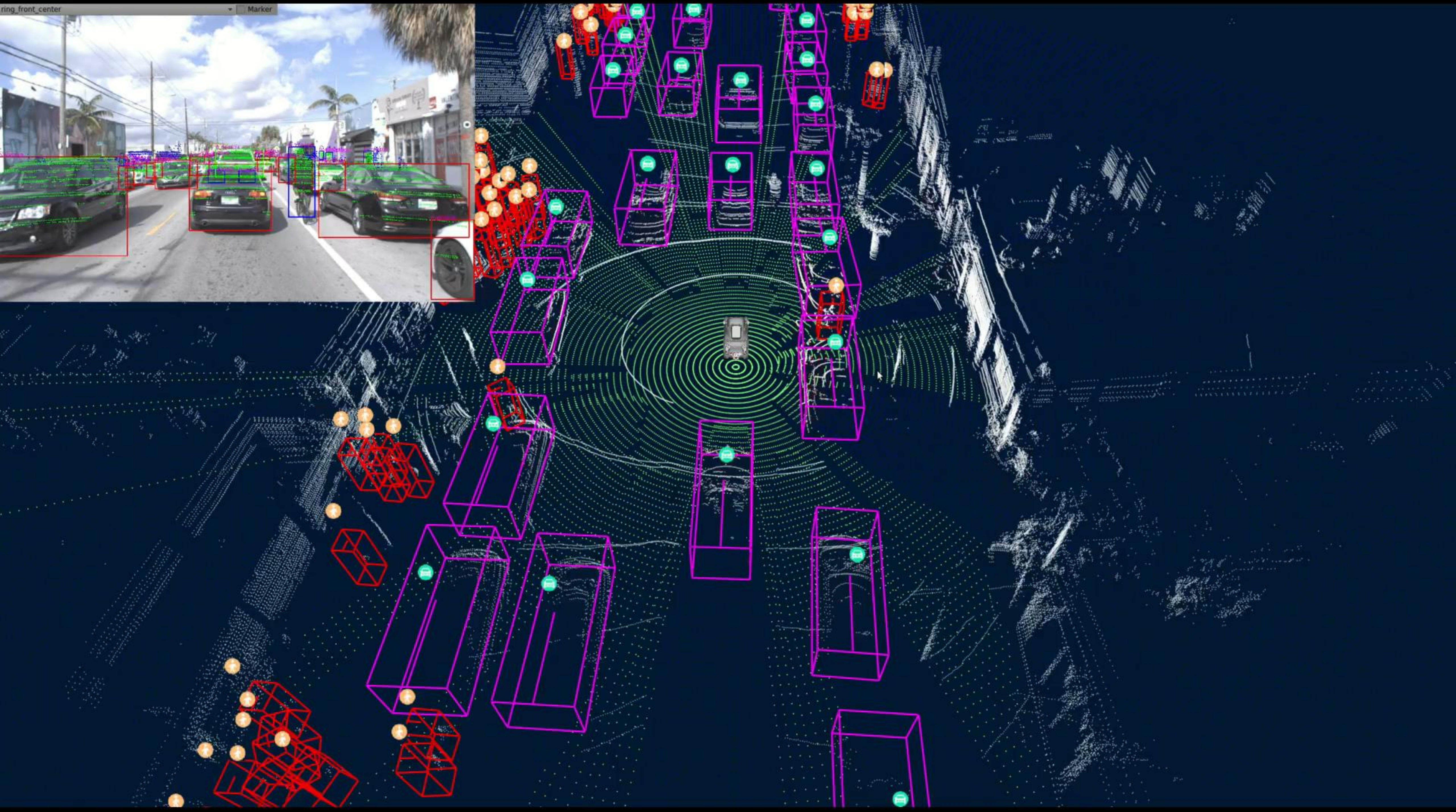
$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

# What can we do with camera calibration?



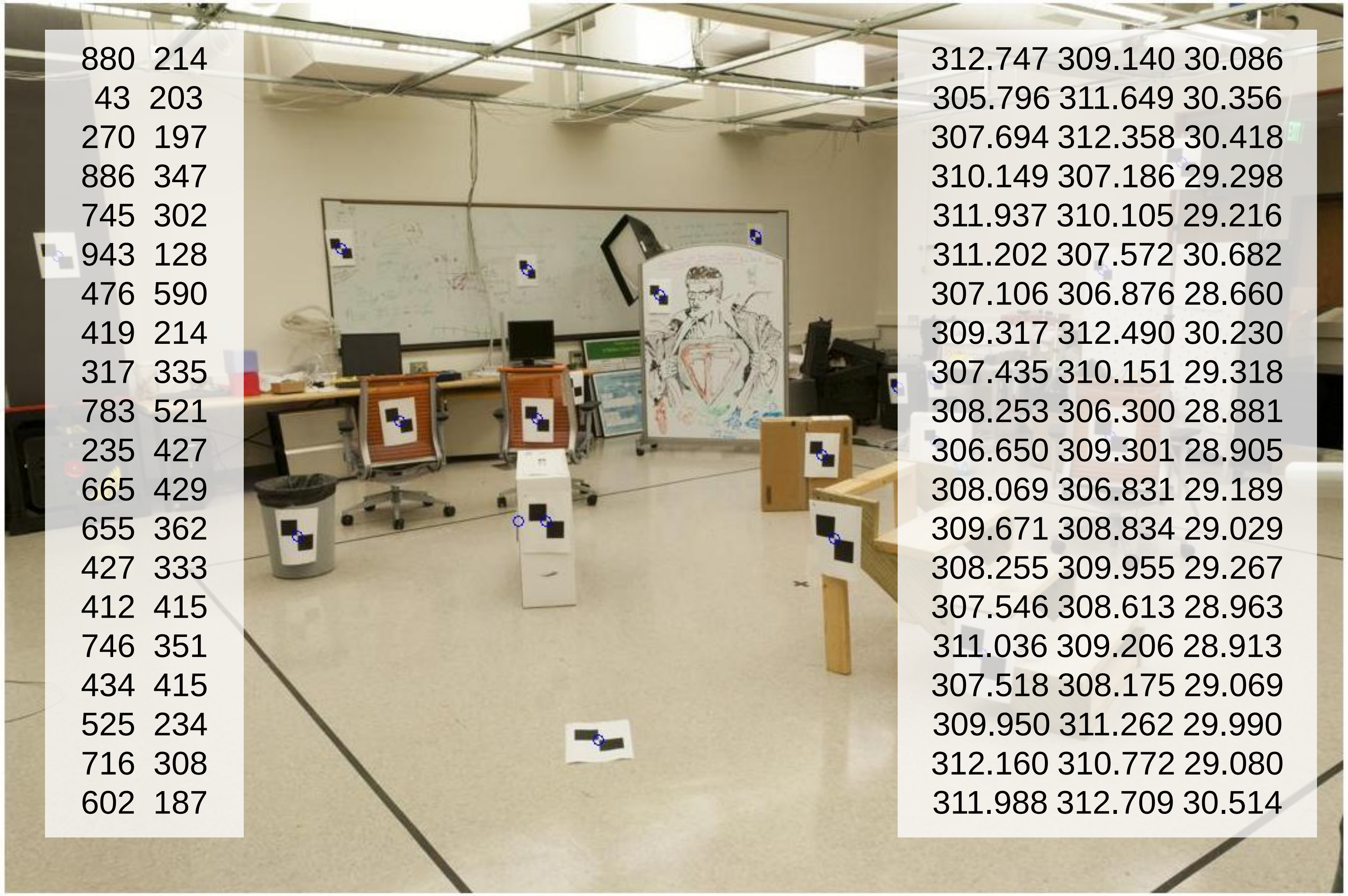




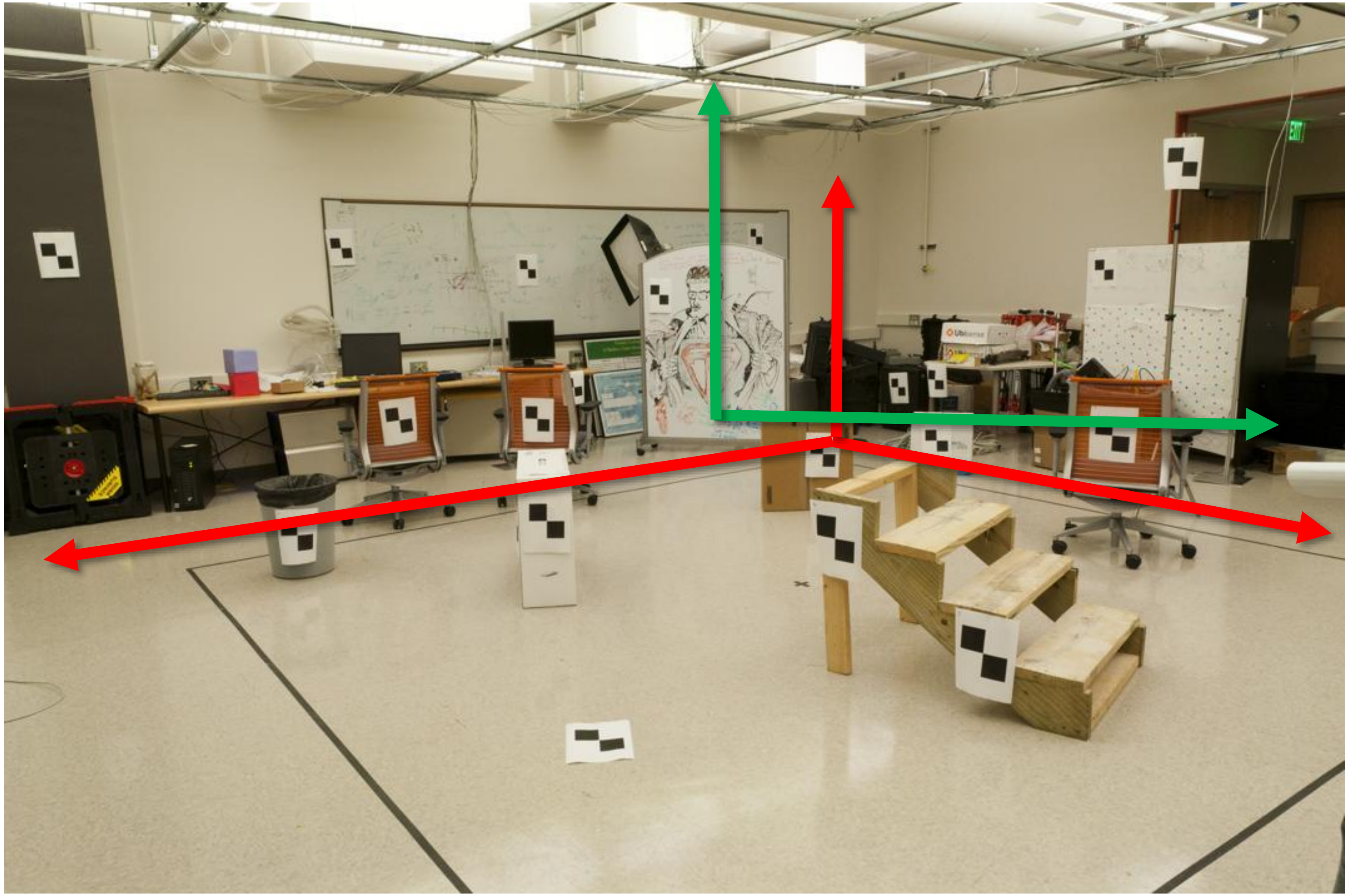


ring_front_center
Marker

# How do we calibrate a camera?

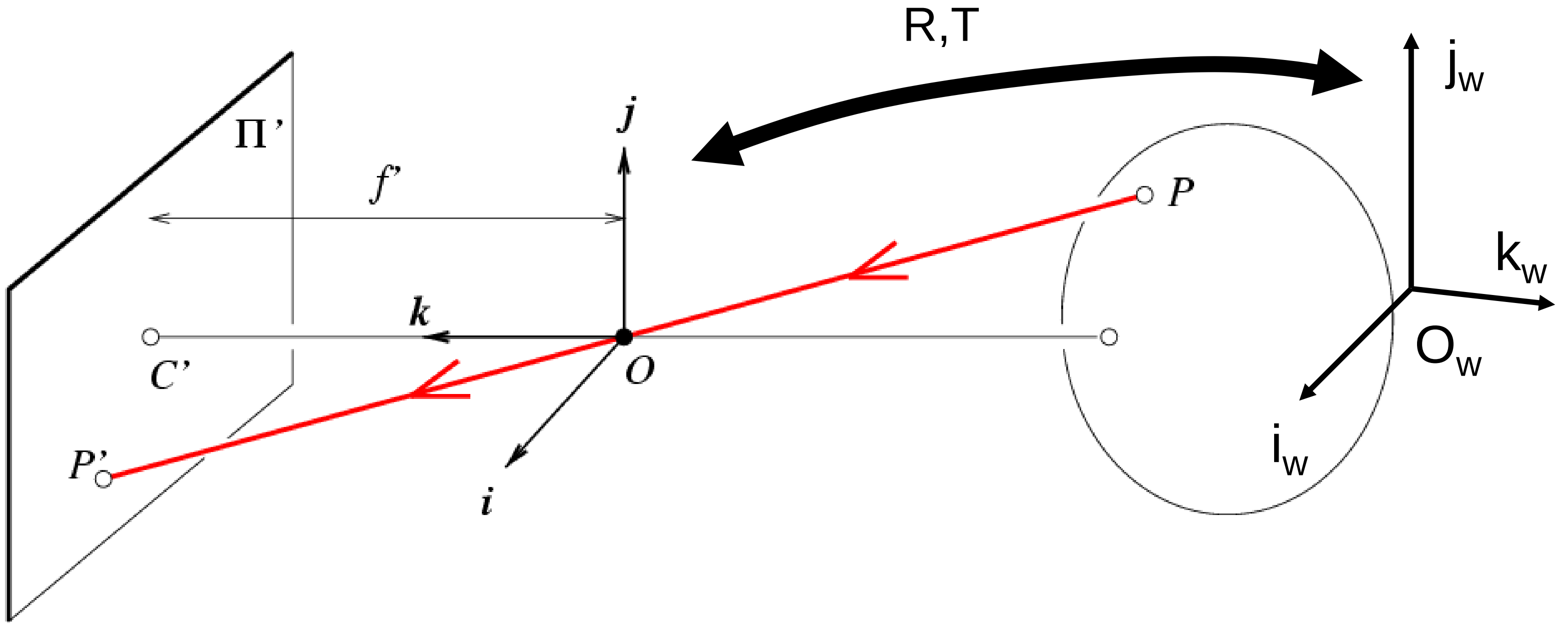


# World vs Camera coordinates



Slide Credit: Saverese

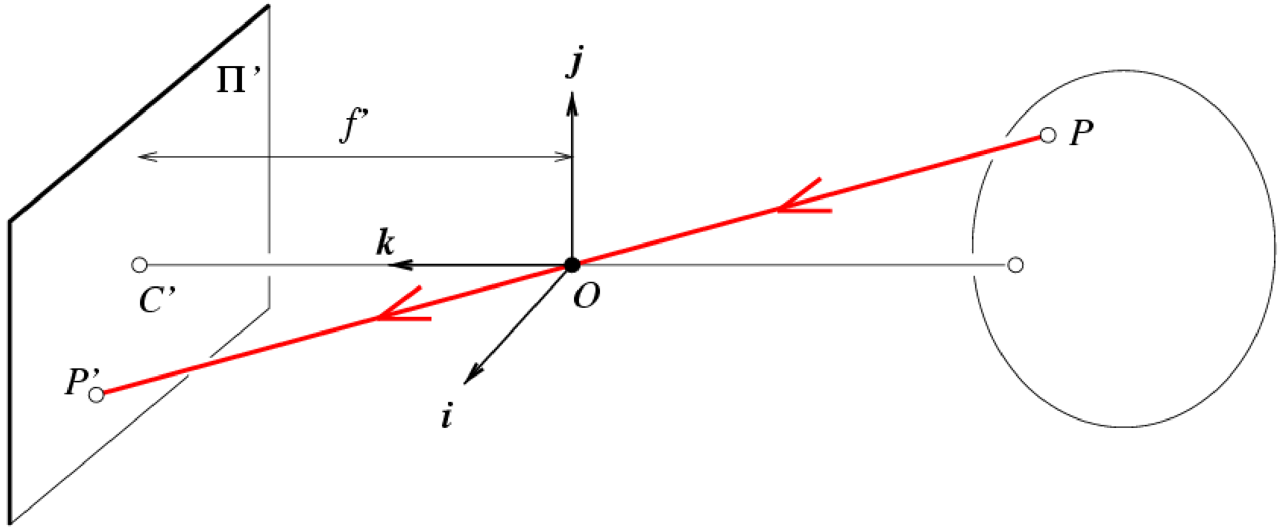
# Projection matrix




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

**x**: Image Coordinates: (u,v,1)
**K**: Intrinsic Matrix (3x3)
**R**: Rotation (3x3)
**t**: Translation (3x1)
**X**: World Coordinates: (X,Y,Z,1)

# Projection matrix




**Intrinsic Assumptions**

- Unit aspect ratio
- Optical center at (0,0)
- No skew

**Extrinsic Assumptions**

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\boxed{\begin{matrix}f & 0 & 0\\0 & f & 0\\0 & 0 & 1\end{matrix}}_{\mathbf{K}} & \begin{matrix}0\\0\\0\end{matrix}\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Slide Credit: Saverese

# Remove assumption: known optical center

Intrinsic Assumptions

- Unit aspect ratio
- No skew

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u \\ v \\ 1\end{bmatrix} = \begin{bmatrix}f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x \\ y \\ z \\ 1\end{bmatrix}$$

# Remove assumption: square pixels

Intrinsic Assumptions
- No skew

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \quad \Rightarrow \quad w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Remove assumption: non-skewed pixels

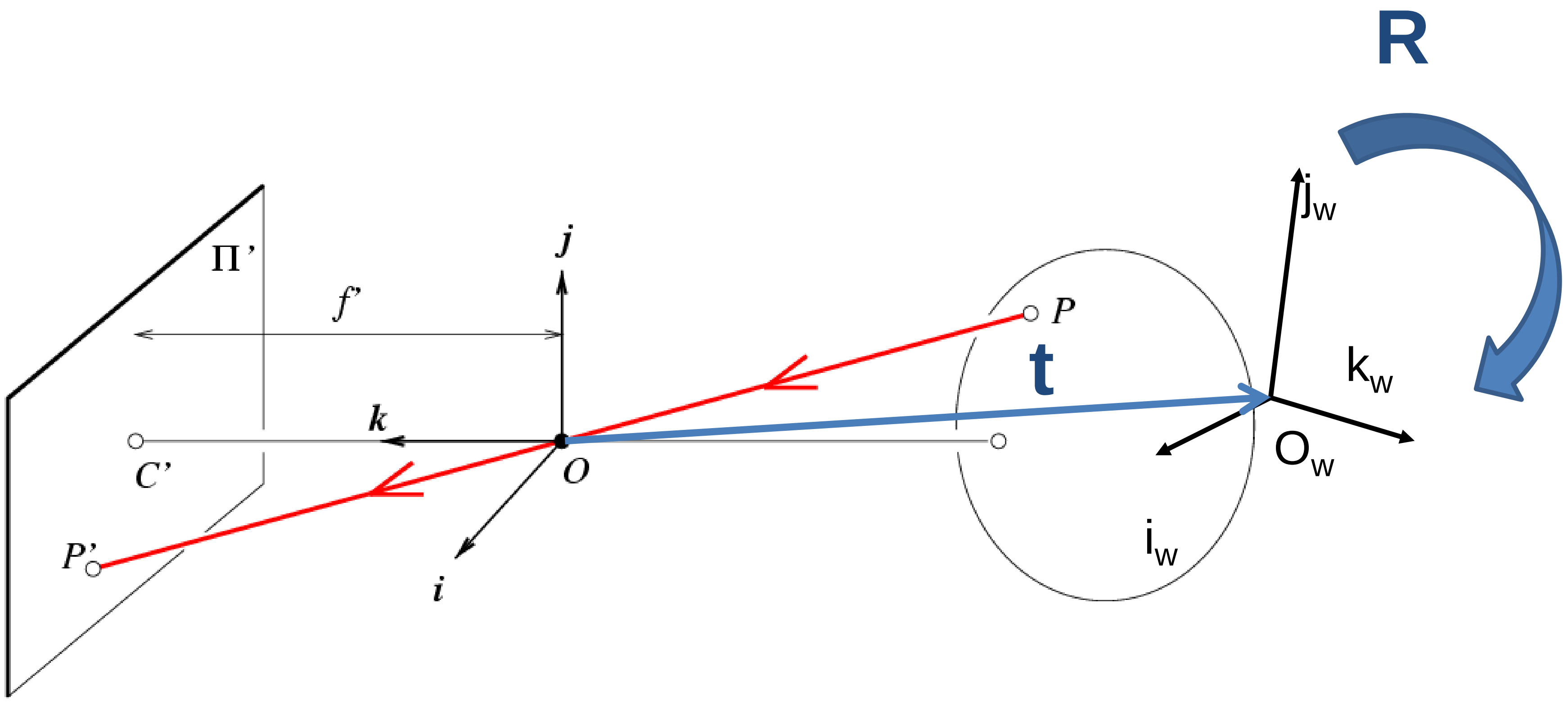
Intrinsic Assumptions

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & s & u_0 & 0\\0 & \beta & v_0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Note: different books use different notation for parameters

# Oriented and Translated Camera



# Allow camera translation

Intrinsic Assumptions

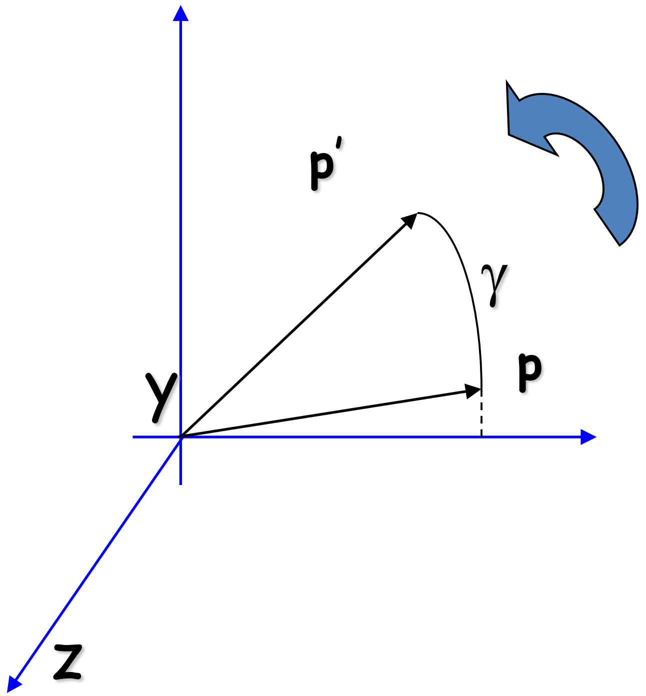
Extrinsic Assumptions
- No rotation

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{t}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & s & u_0\\0 & \beta & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & t_x\\0 & 1 & 0 & t_y\\0 & 0 & 1 & t_z\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Slide Credit: Saverese

# 3D Rotation of Points

Rotation around the coordinate axes, counter-clockwise:




$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Allow camera rotation

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

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

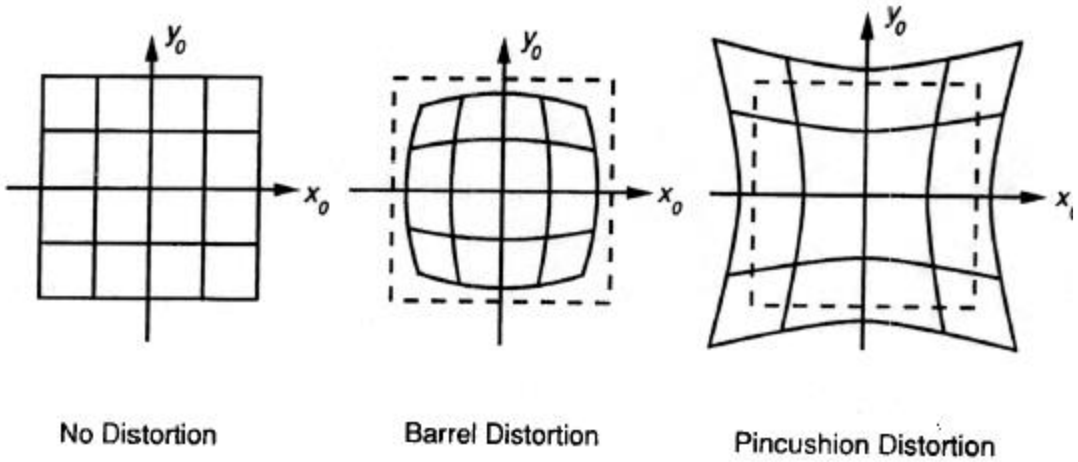
# Degrees of freedom

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

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overset{5}{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{6}{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Beyond Pinholes: Radial Distortion

- Common in wide-angle lenses or for special applications (e.g., security)
- Creates non-linear terms in projection
- Usually handled by through solving for non-linear terms and then correcting image




Corrected Barrel Distortion

Image from Martin Habbecke

# How to calibrate the camera?

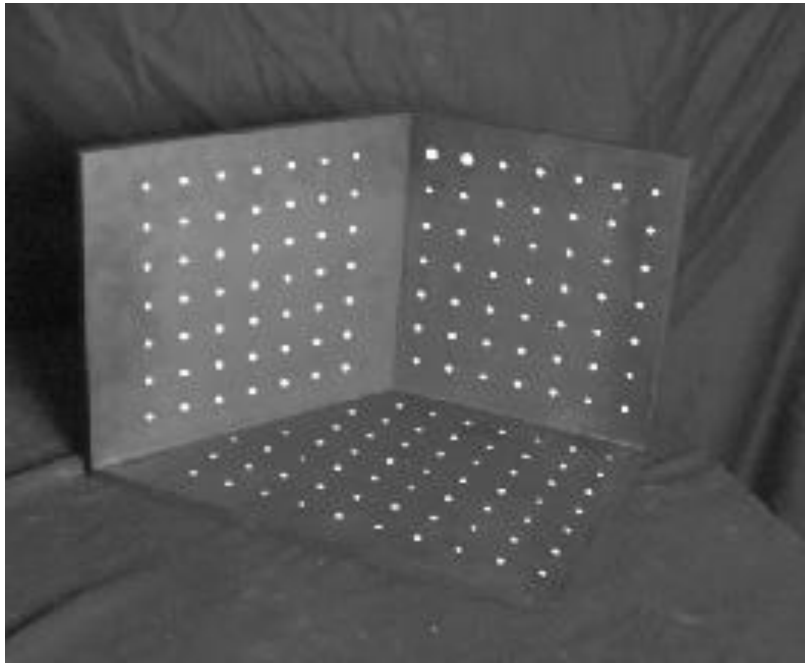
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

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

# Calibrating the Camera

Use a scene with known geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)



Known 2d image coords

Known 3d locations

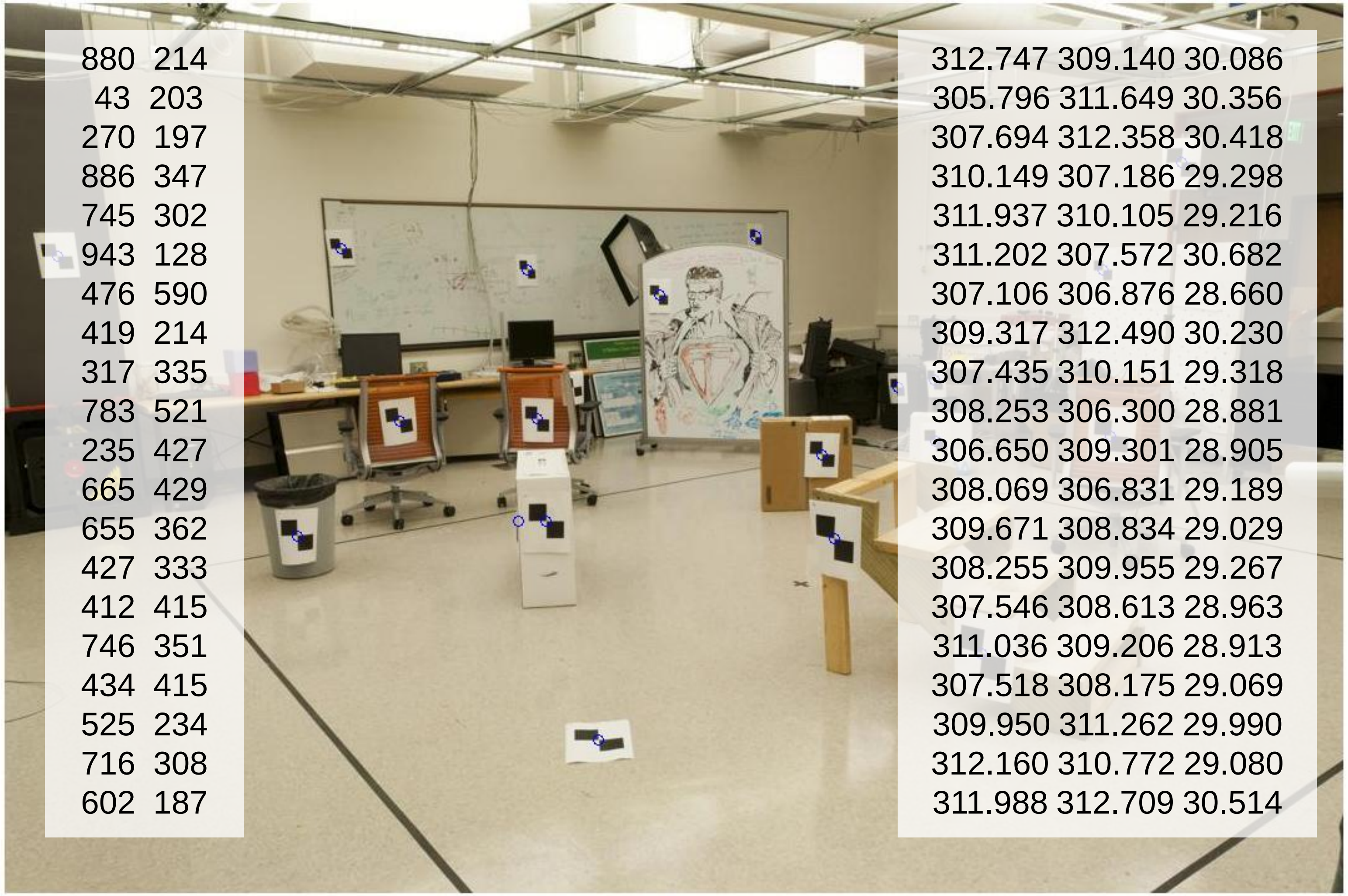
$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Unknown Camera Parameters

# How do we calibrate a camera?

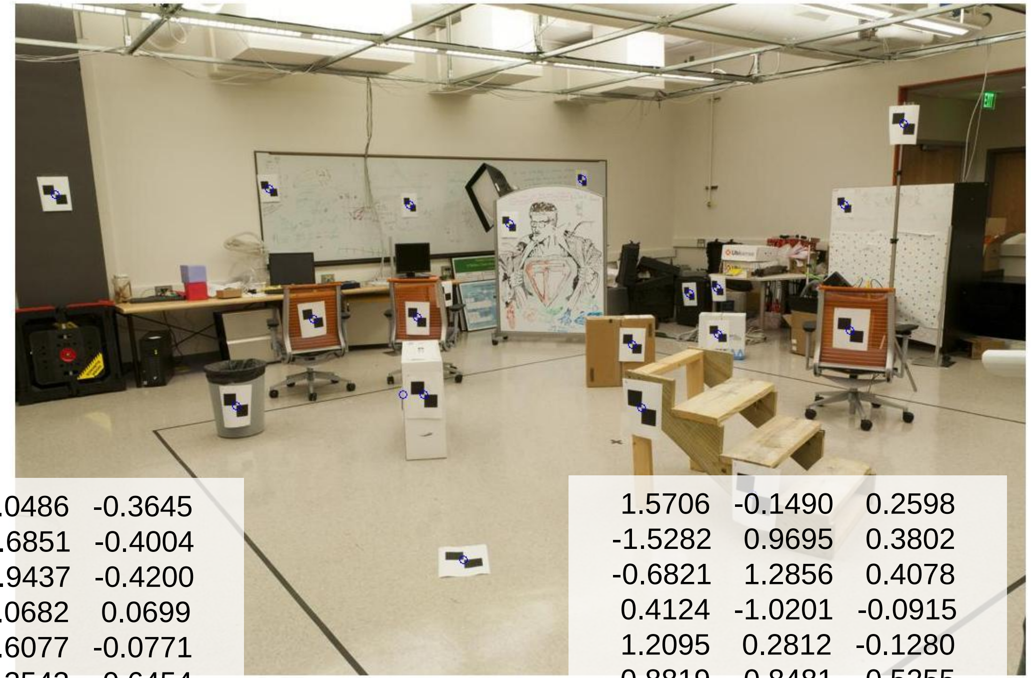
Known 2d image coords

Known 3d locations



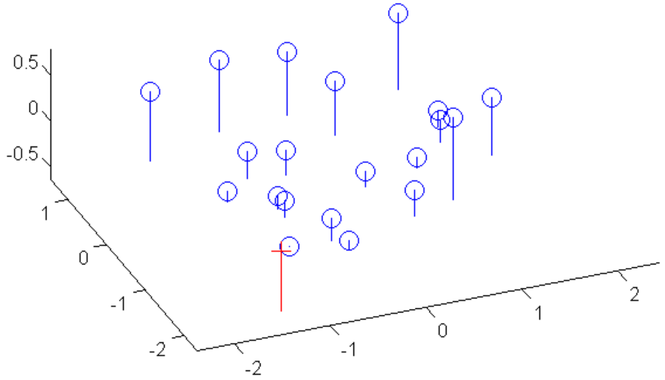
| Known 2d image coords | | Known 3d locations | | |
|---|---|---|---|---|
| 880 | 214 | 312.747 | 309.140 | 30.086 |
| 43 | 203 | 305.796 | 311.649 | 30.356 |
| 270 | 197 | 307.694 | 312.358 | 30.418 |
| 886 | 347 | 310.149 | 307.186 | 29.298 |
| 745 | 302 | 311.937 | 310.105 | 29.216 |
| 943 | 128 | 311.202 | 307.572 | 30.682 |
| 476 | 590 | 307.106 | 306.876 | 28.660 |
| 419 | 214 | 309.317 | 312.490 | 30.230 |
| 317 | 335 | 307.435 | 310.151 | 29.318 |
| 783 | 521 | 308.253 | 306.300 | 28.881 |
| 235 | 427 | 306.650 | 309.301 | 28.905 |
| 665 | 429 | 308.069 | 306.831 | 29.189 |
| 655 | 362 | 309.671 | 308.834 | 29.029 |
| 427 | 333 | 308.255 | 309.955 | 29.267 |
| 412 | 415 | 307.546 | 308.613 | 28.963 |
| 746 | 351 | 311.036 | 309.206 | 28.913 |
| 434 | 415 | 307.518 | 308.175 | 29.069 |
| 525 | 234 | 309.950 | 311.262 | 29.990 |
| 716 | 308 | 312.160 | 310.772 | 29.080 |
| 602 | 187 | 311.988 | 312.709 | 30.514 |

# Estimate of camera center



| | |
|---|---|
| 1.0486 | -0.3645 |
| -1.6851 | -0.4004 |
| -0.9437 | -0.4200 |
| 1.0682 | 0.0699 |
| 0.6077 | -0.0771 |
| 1.2543 | -0.6454 |
| -0.2709 | 0.8635 |
| -0.4571 | -0.3645 |
| -0.7902 | 0.0307 |
| 0.7318 | 0.6382 |
| -1.0580 | 0.3312 |
| 0.3464 | 0.3377 |
| 0.3137 | 0.1189 |
| -0.4310 | 0.0242 |
| -0.4799 | 0.2920 |
| 0.6109 | 0.0830 |
| -0.4081 | 0.2920 |
| -0.1109 | -0.2992 |
| 0.5129 | -0.0575 |
| 0.1406 | -0.4527 |

| | | |
|---|---|---|
| 1.5706 | -0.1490 | 0.2598 |
| -1.5282 | 0.9695 | 0.3802 |
| -0.6821 | 1.2856 | 0.4078 |
| 0.4124 | -1.0201 | -0.0915 |
| 1.2095 | 0.2812 | -0.1280 |
| 0.8819 | -0.8481 | 0.5255 |
| -0.9442 | -1.1583 | -0.3759 |
| 0.0415 | 1.3445 | 0.3240 |
| -0.7975 | 0.3017 | -0.0826 |
| -0.4329 | -1.4151 | -0.2774 |
| -1.1475 | -0.0772 | -0.2667 |
| -0.5149 | -1.1784 | -0.1401 |
| 0.1993 | -0.2854 | -0.2114 |
| -0.4320 | 0.2143 | -0.1053 |
| -0.7481 | -0.3840 | -0.2408 |
| 0.8078 | -0.1196 | -0.2631 |
| -0.7605 | -0.5792 | -0.1936 |
| 0.3237 | 0.7970 | 0.2170 |
| 1.3089 | 0.5786 | -0.1887 |
| 1.2323 | 1.4421 | 0.4506 |



Unknown Camera Parameters

Known 2d image coords

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

Known 3d locations

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

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

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

# Unknown Camera Parameters

Known 2d image coords

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

Known 3d locations

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$
$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u$$
$$0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v$$

Unknown Camera Parameters

Known 2d image coords

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

Known 3d locations

$$0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u$$

$$0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v$$

- Method 1 – homogeneous linear system. Solve for m's entries using linear least squares

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

```
[U, S, V] = svd(A);
M = V(:,end);
M = reshape(M,[],3)';
```

**For python, see numpy.linalg.svd**

# Unknown Camera Parameters

Known 2d image coords

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

Known 3d locations

- Method 2 – nonhomogeneous linear system. Solve for m's entries using linear least squares

**Ax=b** form

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

```
M = A\Y;
M = [M;1];
M = reshape(M,[],3)';
```

**For python, see numpy.linalg.lstsq**

# Calibration with linear method

- Advantages
  - Easy to formulate and solve
  - Provides initialization for non-linear methods

- Disadvantages
  - Doesn’t directly give you human-interpretable camera parameters
  - Doesn’t model radial distortion
  - Can’t impose constraints, such as known focal length

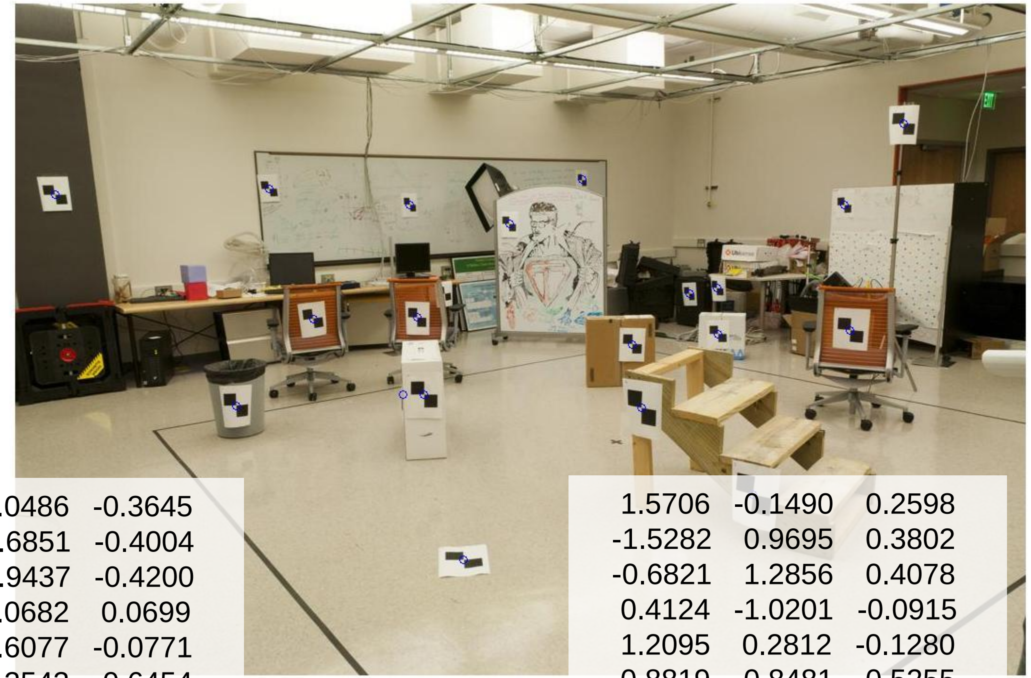
- Non-linear methods are preferred
  - Define error as difference between projected points and measured points
  - Minimize error using Newton’s method or other non-linear optimization

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

- Yes!
- You can use *RQ* factorization (note – not the more familiar *QR* factorization). *R* (right diagonal) is K, and *Q* (orthogonal basis) is R. T, the last column of [R | T], is inv(K) * last column of M.
  - But you need to do a bit of post-processing to make sure that the matrices are valid. See http://ksimek.github.io/2012/08/14/decompose/

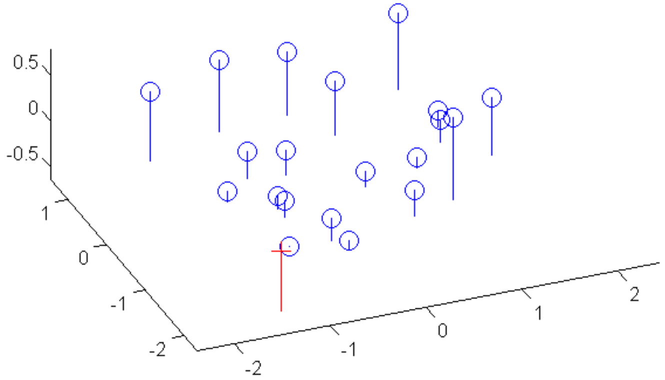
# For project 3, we want the camera center

# Estimate of camera center

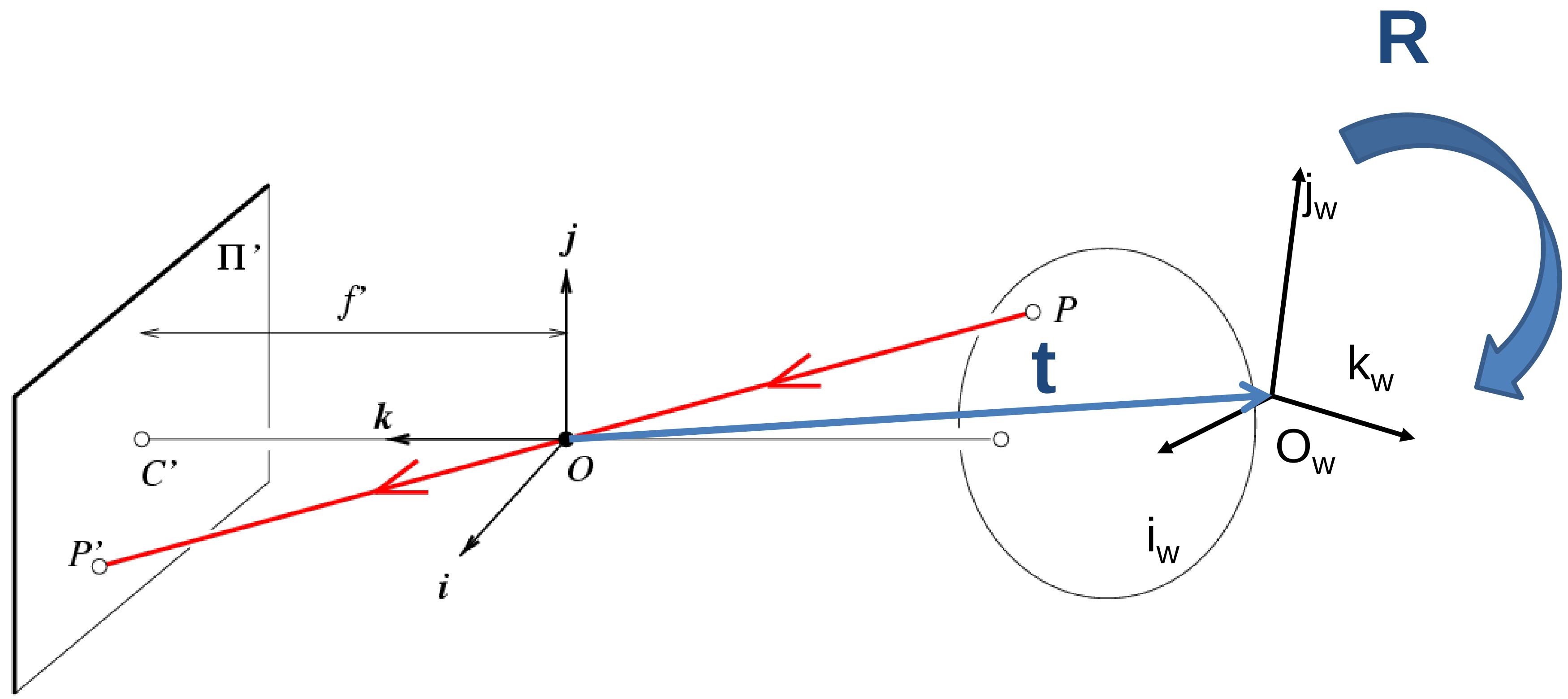


| | |
|---|---|
| 1.0486 | -0.3645 |
| -1.6851 | -0.4004 |
| -0.9437 | -0.4200 |
| 1.0682 | 0.0699 |
| 0.6077 | -0.0771 |
| 1.2543 | -0.6454 |
| -0.2709 | 0.8635 |
| -0.4571 | -0.3645 |
| -0.7902 | 0.0307 |
| 0.7318 | 0.6382 |
| -1.0580 | 0.3312 |
| 0.3464 | 0.3377 |
| 0.3137 | 0.1189 |
| -0.4310 | 0.0242 |
| -0.4799 | 0.2920 |
| 0.6109 | 0.0830 |
| -0.4081 | 0.2920 |
| -0.1109 | -0.2992 |
| 0.5129 | -0.0575 |
| 0.1406 | -0.4527 |

| | | |
|---|---|---|
| 1.5706 | -0.1490 | 0.2598 |
| -1.5282 | 0.9695 | 0.3802 |
| -0.6821 | 1.2856 | 0.4078 |
| 0.4124 | -1.0201 | -0.0915 |
| 1.2095 | 0.2812 | -0.1280 |
| 0.8819 | -0.8481 | 0.5255 |
| -0.9442 | -1.1583 | -0.3759 |
| 0.0415 | 1.3445 | 0.3240 |
| -0.7975 | 0.3017 | -0.0826 |
| -0.4329 | -1.4151 | -0.2774 |
| -1.1475 | -0.0772 | -0.2667 |
| -0.5149 | -1.1784 | -0.1401 |
| 0.1993 | -0.2854 | -0.2114 |
| -0.4320 | 0.2143 | -0.1053 |
| -0.7481 | -0.3840 | -0.2408 |
| 0.8078 | -0.1196 | -0.2631 |
| -0.7605 | -0.5792 | -0.1936 |
| 0.3237 | 0.7970 | 0.2170 |
| 1.3089 | 0.5786 | -0.1887 |
| 1.2323 | 1.4421 | 0.4506 |



# Oriented and Translated Camera



# Recovering the camera center

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

$$w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & s & u_0\\0 & \beta & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & \boxed{t_x}\\r_{21} & r_{22} & r_{23} & \boxed{t_y}\\r_{31} & r_{32} & r_{33} & \boxed{t_z}\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

This is not the camera center -C. It is –RC (because a point will be rotated before $t_x$, $t_y$, and $t_z$ are added)

$$\begin{bmatrix}su\\sv\\s\end{bmatrix} = \begin{bmatrix}\underbrace{\boxed{\begin{matrix}* & * & *\\ * & * & *\\ * & * & *\end{matrix}}}_{Q} & \boxed{\begin{matrix}*\\ *\\ *\end{matrix}}\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

This, $m_4$, is K * **t**

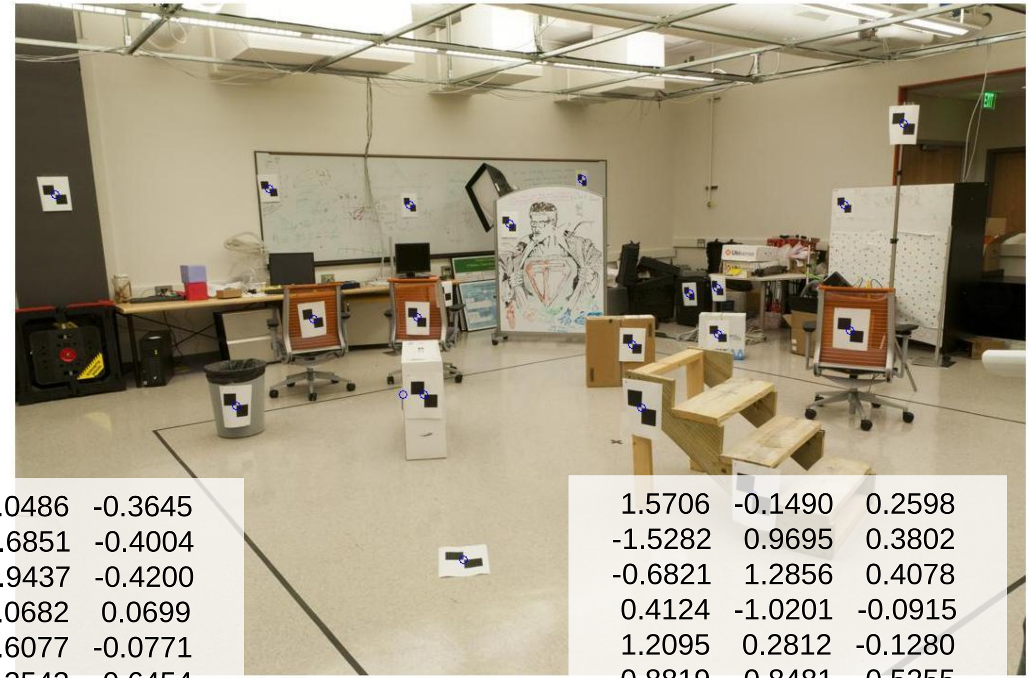
So $K^{-1}$ $m_4$ is **t**

So we need

$-R^{-1} K^{-1} m_4$ to get C

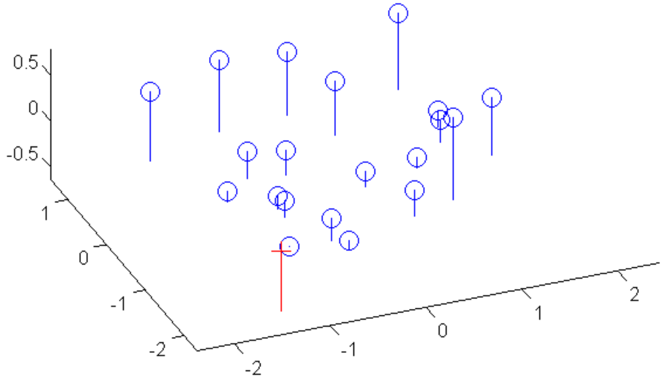
Q is K * R. So we just need $-Q^{-1} m_4$

# Estimate of camera center



| | |
|---|---|
| 1.0486 | -0.3645 |
| -1.6851 | -0.4004 |
| -0.9437 | -0.4200 |
| 1.0682 | 0.0699 |
| 0.6077 | -0.0771 |
| 1.2543 | -0.6454 |
| -0.2709 | 0.8635 |
| -0.4571 | -0.3645 |
| -0.7902 | 0.0307 |
| 0.7318 | 0.6382 |
| -1.0580 | 0.3312 |
| 0.3464 | 0.3377 |
| 0.3137 | 0.1189 |
| -0.4310 | 0.0242 |
| -0.4799 | 0.2920 |
| 0.6109 | 0.0830 |
| -0.4081 | 0.2920 |
| -0.1109 | -0.2992 |
| 0.5129 | -0.0575 |
| 0.1406 | -0.4527 |

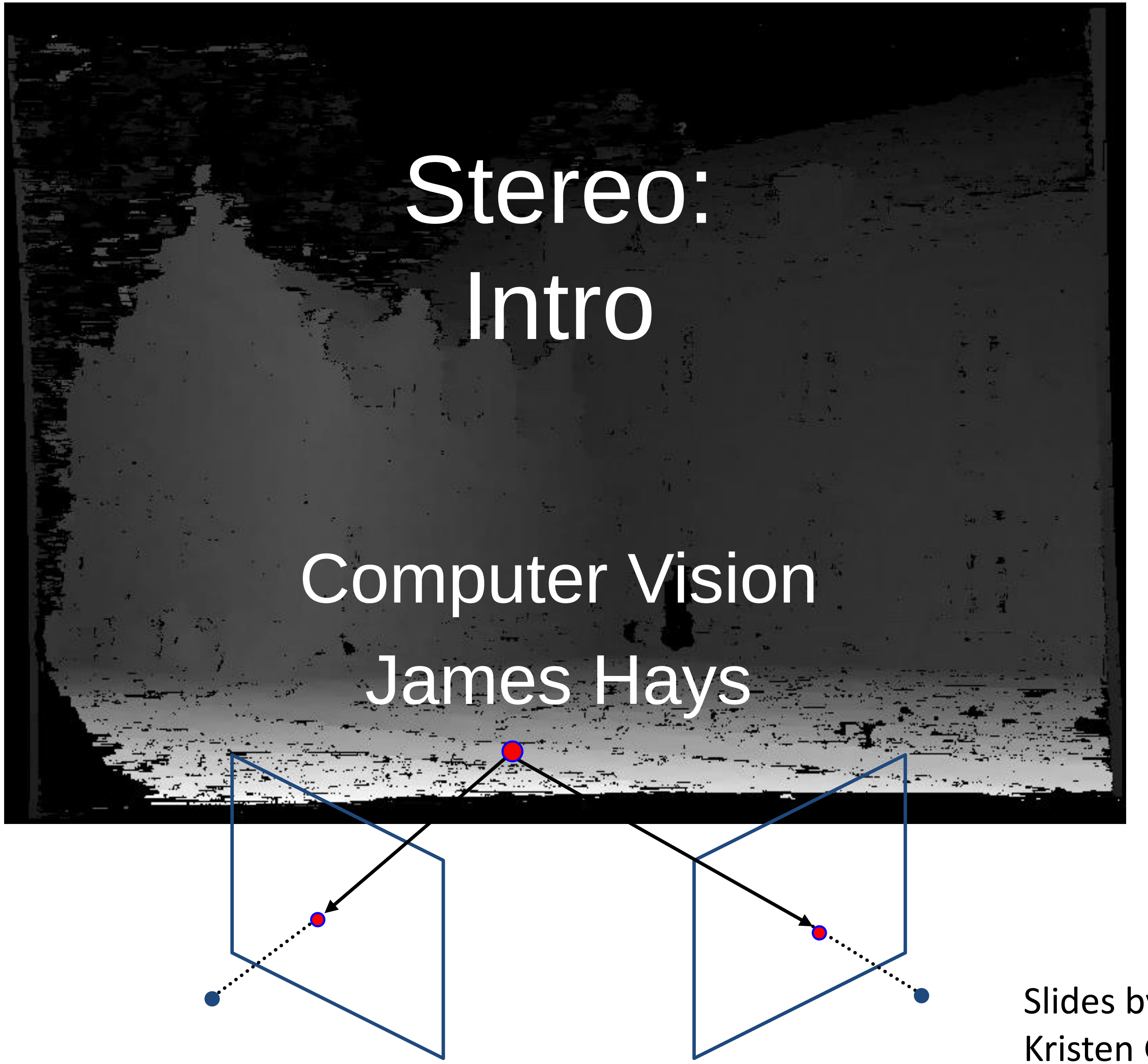
| | | |
|---|---|---|
| 1.5706 | -0.1490 | 0.2598 |
| -1.5282 | 0.9695 | 0.3802 |
| -0.6821 | 1.2856 | 0.4078 |
| 0.4124 | -1.0201 | -0.0915 |
| 1.2095 | 0.2812 | -0.1280 |
| 0.8819 | -0.8481 | 0.5255 |
| -0.9442 | -1.1583 | -0.3759 |
| 0.0415 | 1.3445 | 0.3240 |
| -0.7975 | 0.3017 | -0.0826 |
| -0.4329 | -1.4151 | -0.2774 |
| -1.1475 | -0.0772 | -0.2667 |
| -0.5149 | -1.1784 | -0.1401 |
| 0.1993 | -0.2854 | -0.2114 |
| -0.4320 | 0.2143 | -0.1053 |
| -0.7481 | -0.3840 | -0.2408 |
| 0.8078 | -0.1196 | -0.2631 |
| -0.7605 | -0.5792 | -0.1936 |
| 0.3237 | 0.7970 | 0.2170 |
| 1.3089 | 0.5786 | -0.1887 |
| 1.2323 | 1.4421 | 0.4506 |



# Stereo:
# Intro

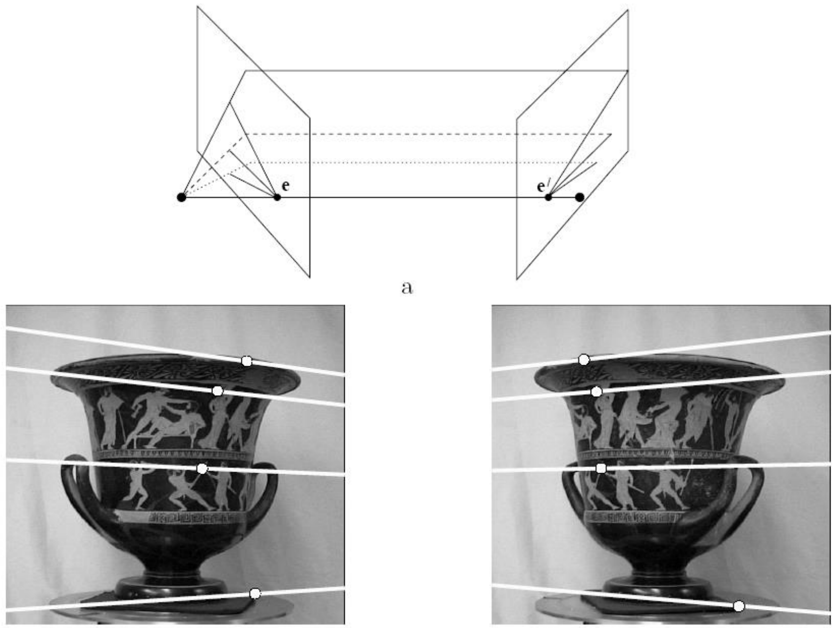
Computer Vision

James Hays



Slides by
Kristen Grauman

# Multiple views




Hartley and Zisserman

stereo vision
structure from motion
optical flow

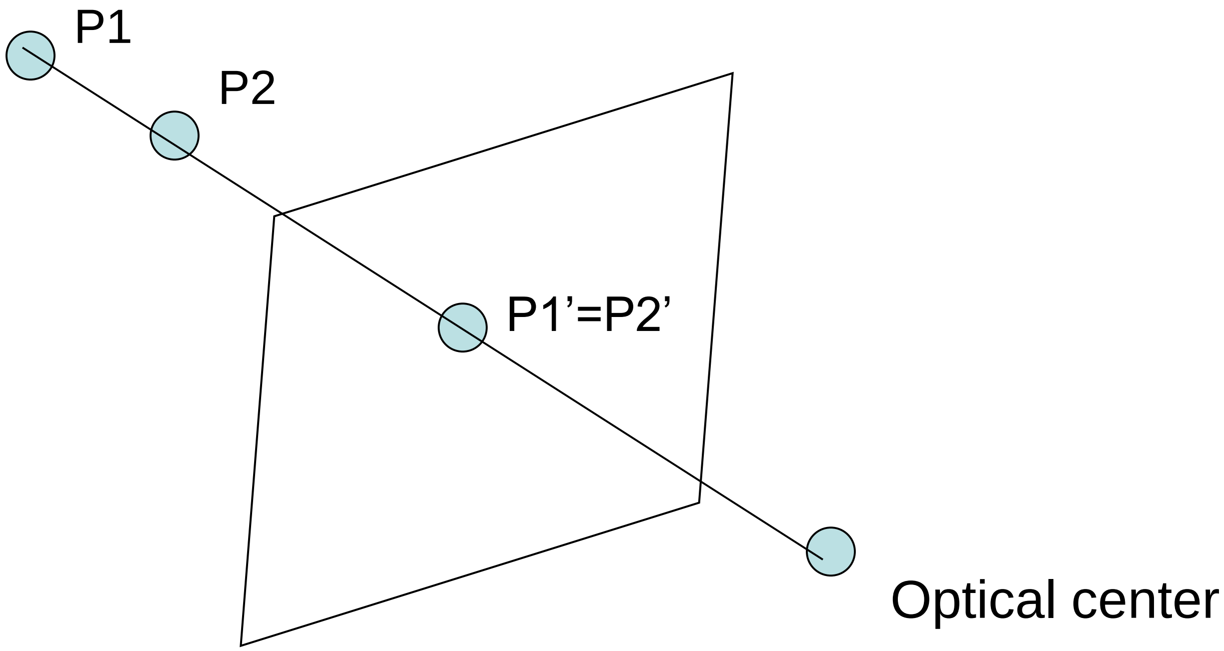
# Why multiple views?

- Structure and depth are inherently ambiguous from single views.

Images from Lana Lazebnik

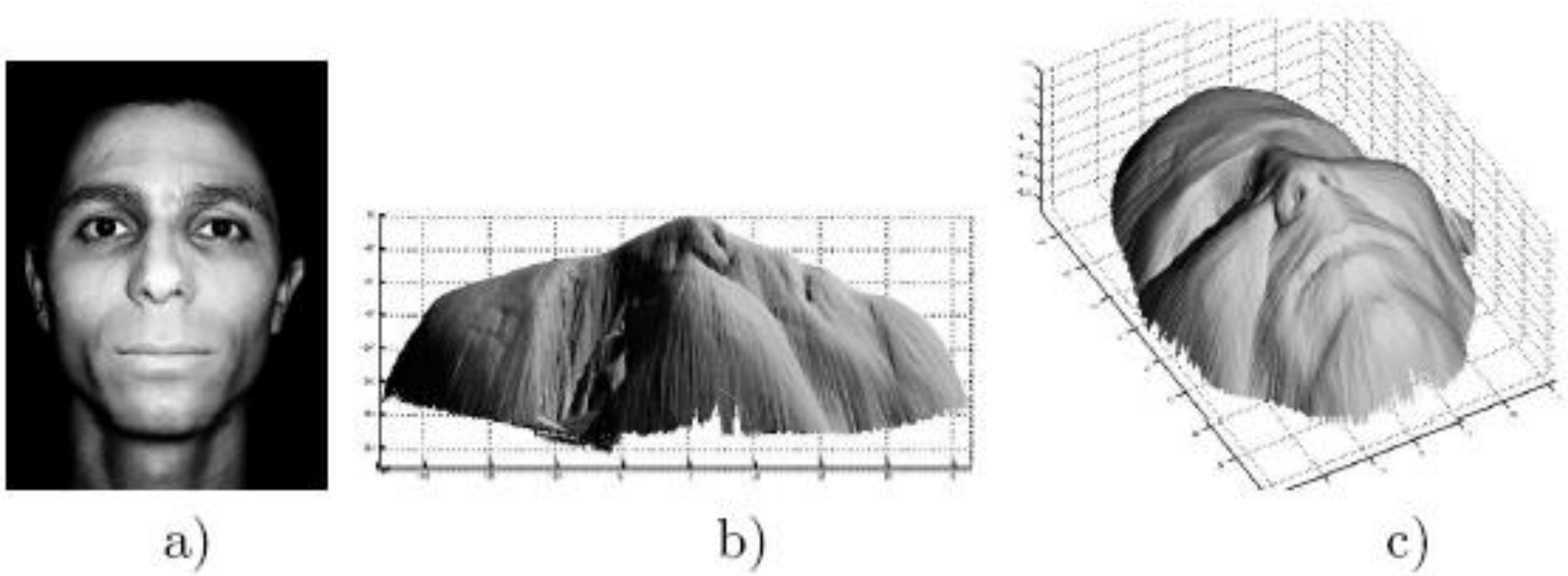
# Why multiple views?

- Structure and depth are inherently ambiguous from single views.



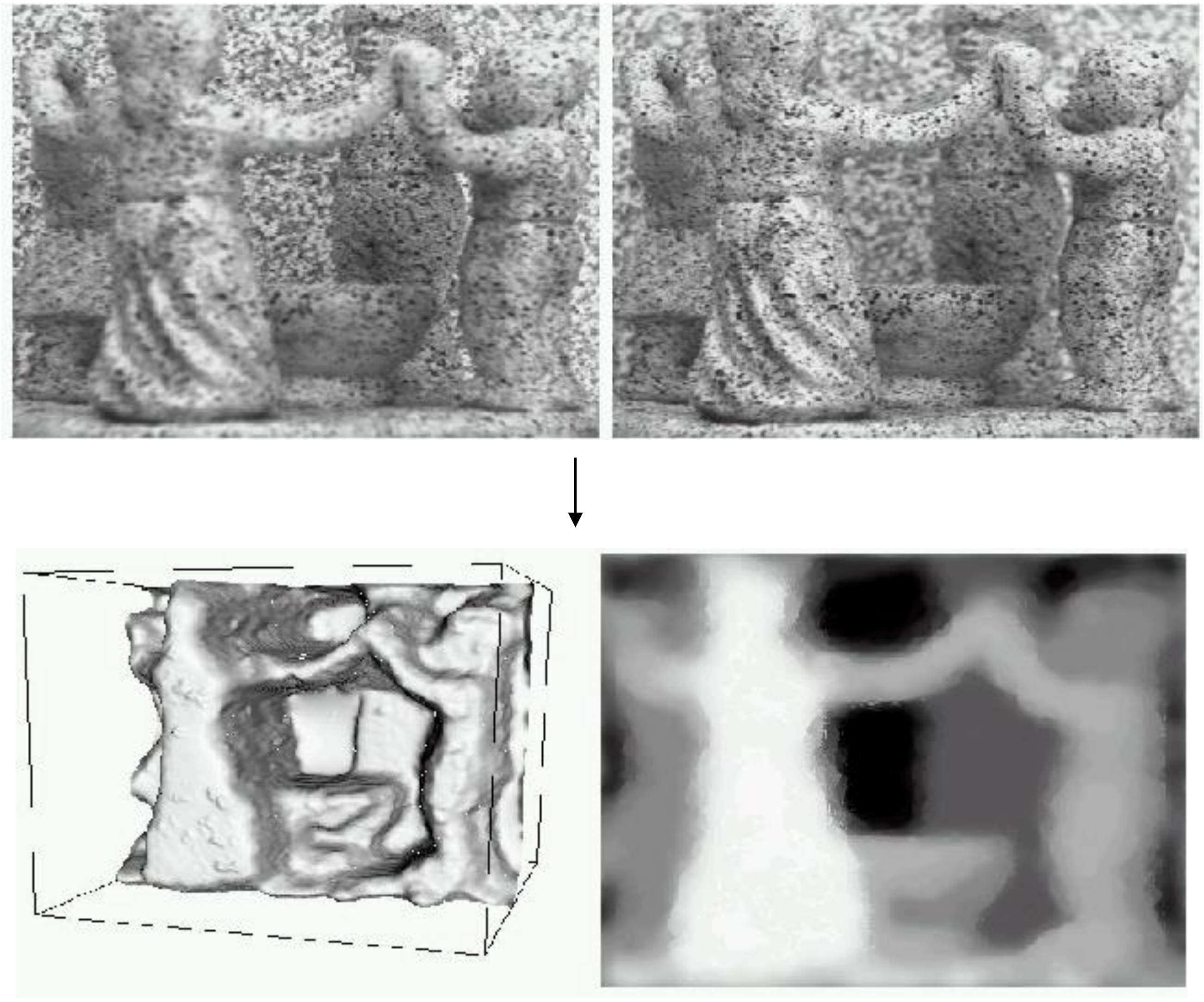
- What cues help us to perceive 3d shape and depth?

# Shading




[Figure from Prados & Faugeras 2006]

# Focus/defocus

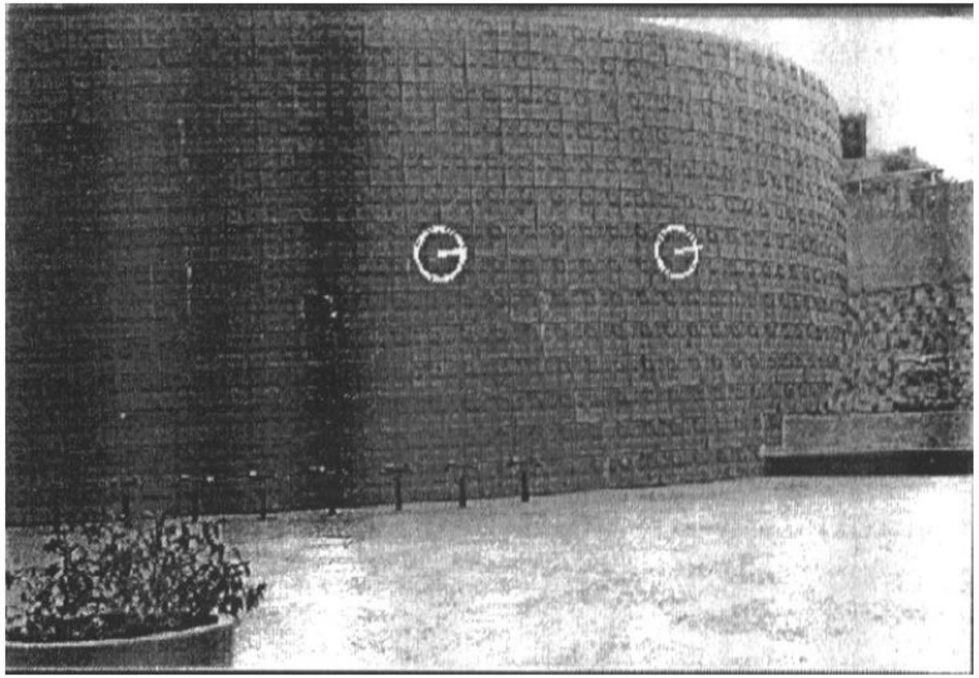


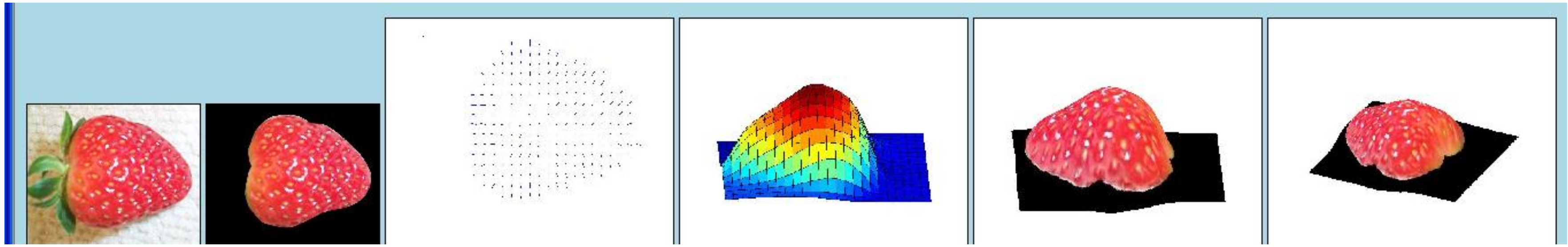
Images from same point of view, different camera parameters

3d shape / depth estimates

[figs from H. Jin and P. Favaro, 2002]

# Texture



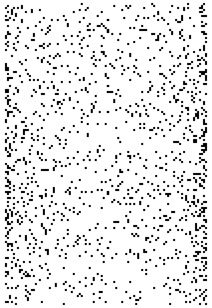


[From A.M. Loh. The recovery of 3-D structure using visual texture patterns. PhD thesis]

# Perspective effects


Image credit: S. Seitz

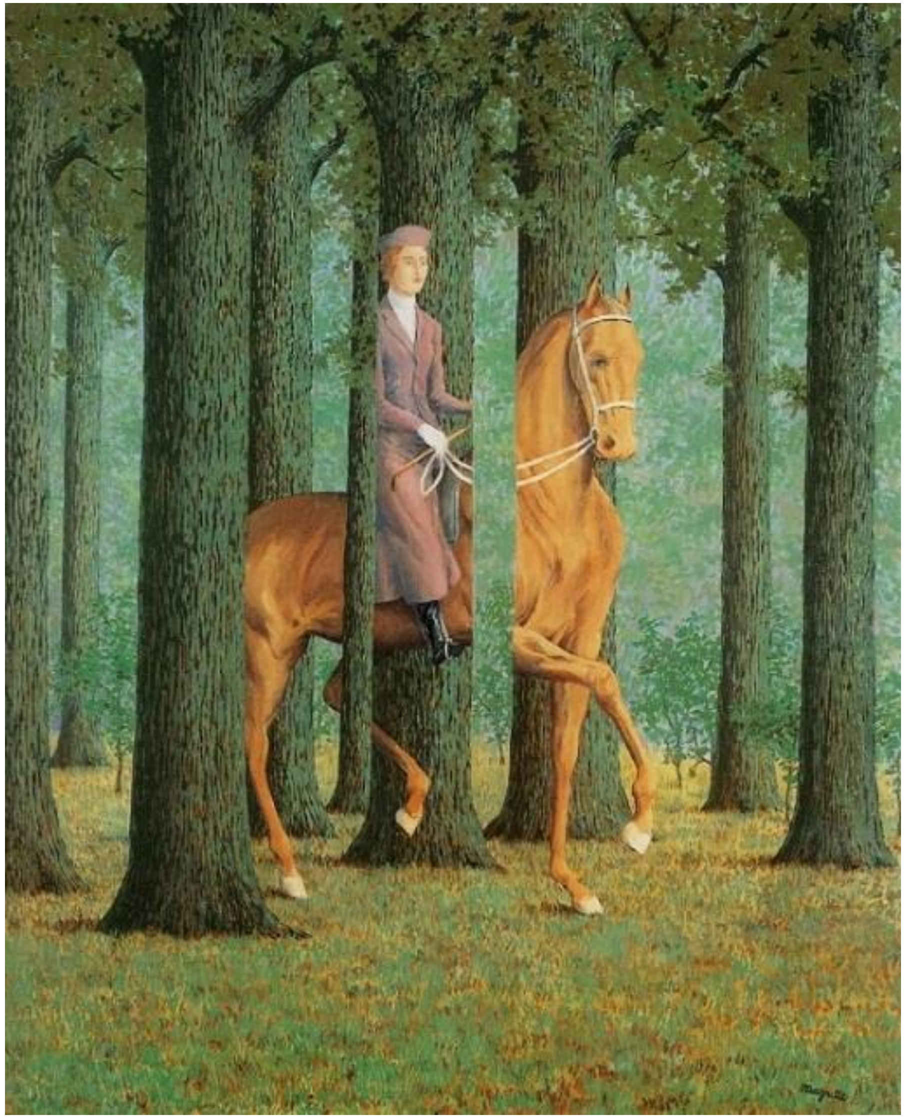
# Motion



Figures from L. Zhang

http://www.brainconnection.com/teasers/?main=illusion/motion-shape

# Occlusion

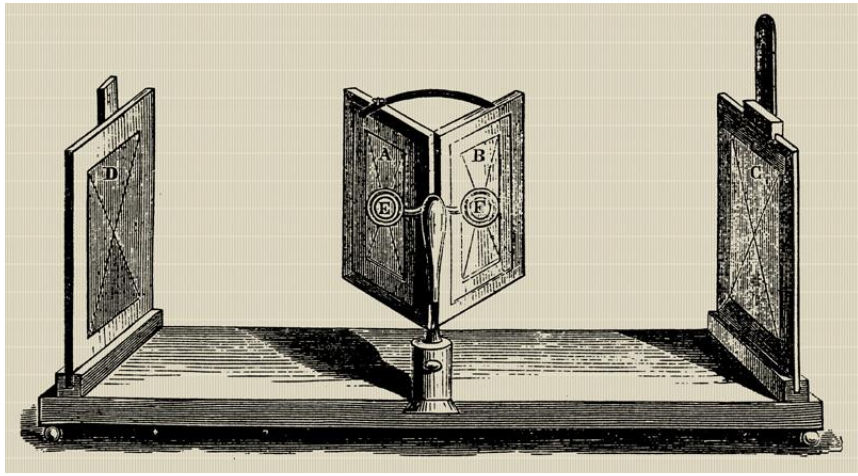


Rene Magritt'e famous painting *Le Blanc-Seing* (literal translation: "The Blank Signature") roughly translates as "free hand“. 1965


If stereo were critical for depth perception, navigation, recognition, etc., then this would be a problem

# Stereo photography and stereo viewers

Take two pictures of the same subject from two slightly different viewpoints and display so that each eye sees only one of the images.




Invented by Sir Charles Wheatstone, 1838

Image from fisher-price.com


http://www.johnsonshawmuseum.org

© Copyright 2001 Johnson-Shaw Stereoscopic Museum

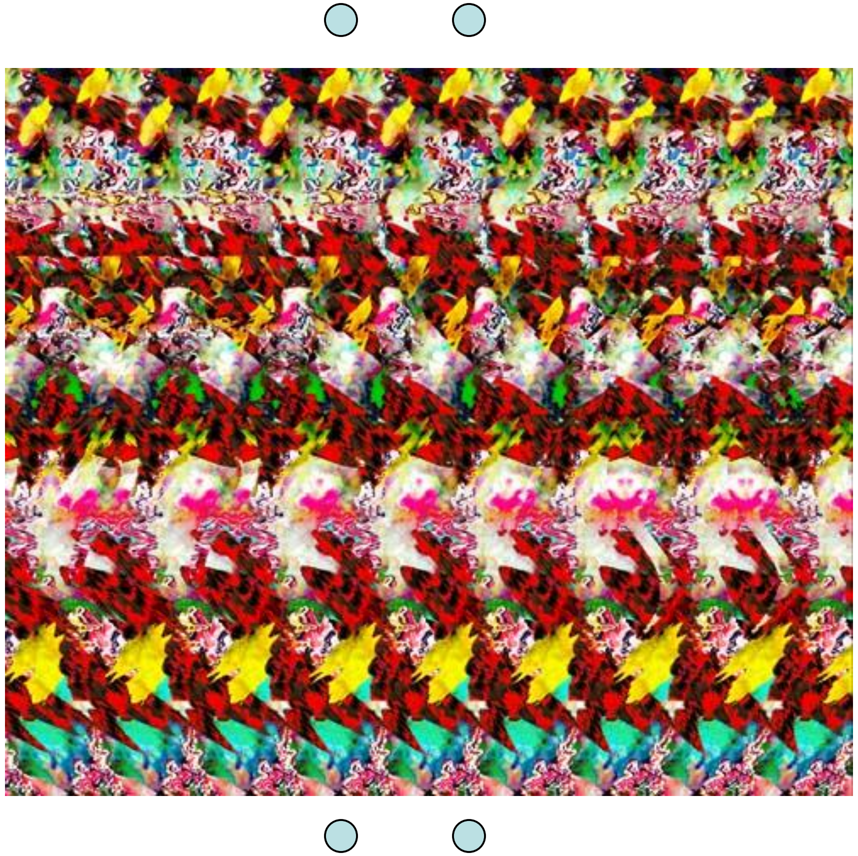

http://www.johnsonshawmuseum.org


Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923

http://www.well.com/~jimg/stereo/stereo_list.html

http://www.well.com/~jimg/stereo/stereo_list.html

# Autostereograms



Exploit disparity as depth cue using single image.

(Single image random dot stereogram, Single image stereogram)

Images from magiceye.com

# Autostereograms

Images from magiceye.com

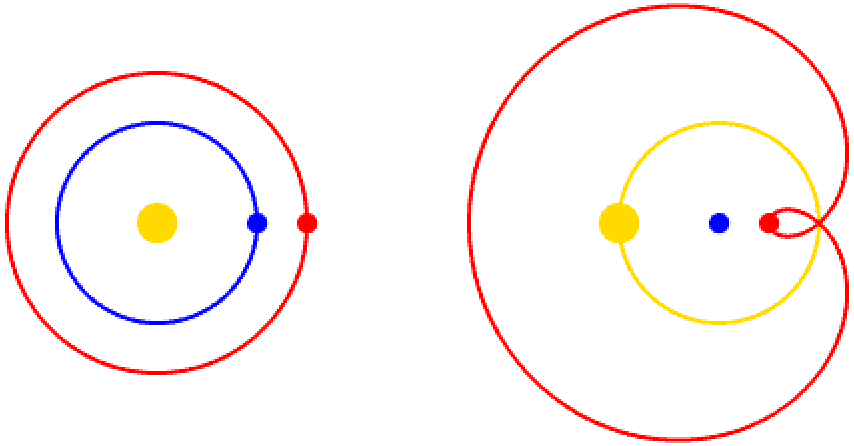
# Parallax and our universe

Look again at that dot. That's here. That's home. That's us. On it everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every "superstar," every "supreme leader," every saint and sinner in the history of our species lived there--on a mote of dust suspended in a sunbeam.

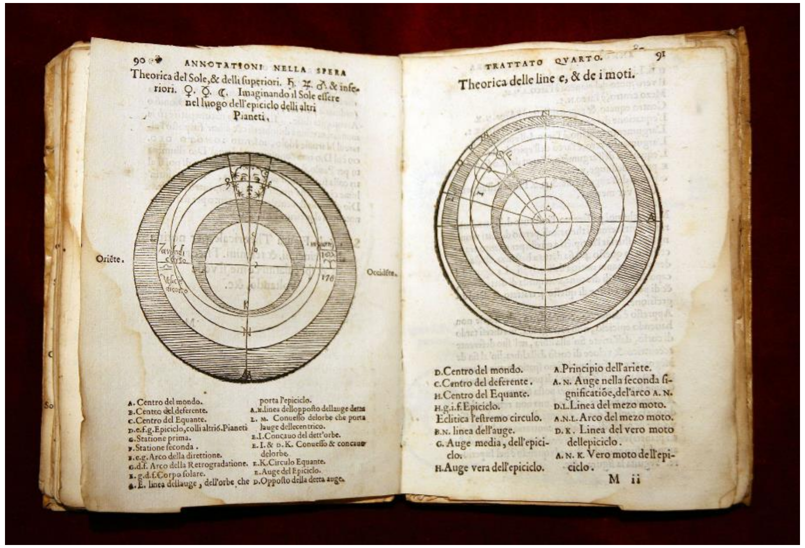
— Carl Sagan

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

**Nicolaus Copernicus**



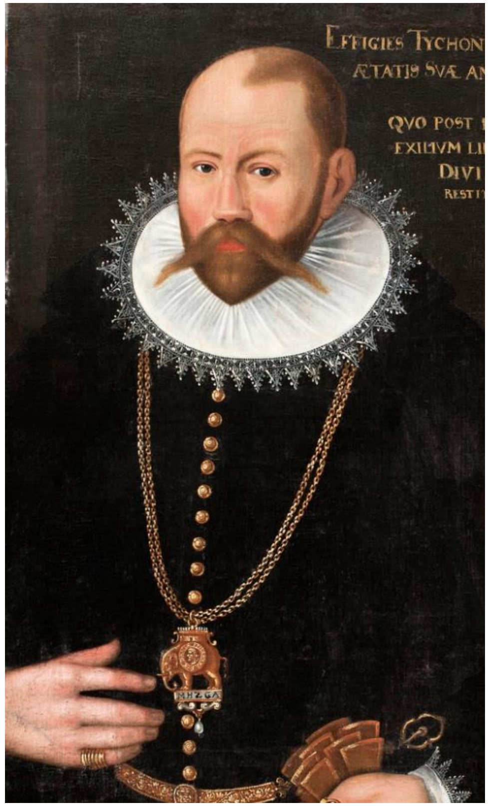
Motion of Sun (yellow), Earth (blue), and Mars (red). At left, Copernicus' heliocentric motion. At right, traditional geocentric motion, including the retrograde motion of Mars.



**geocentric model** (often exemplified specifically by the **Ptolemaic system**)

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

If the apparent motion of the planets is caused by parallax, why aren't we seeing parallax for stars?

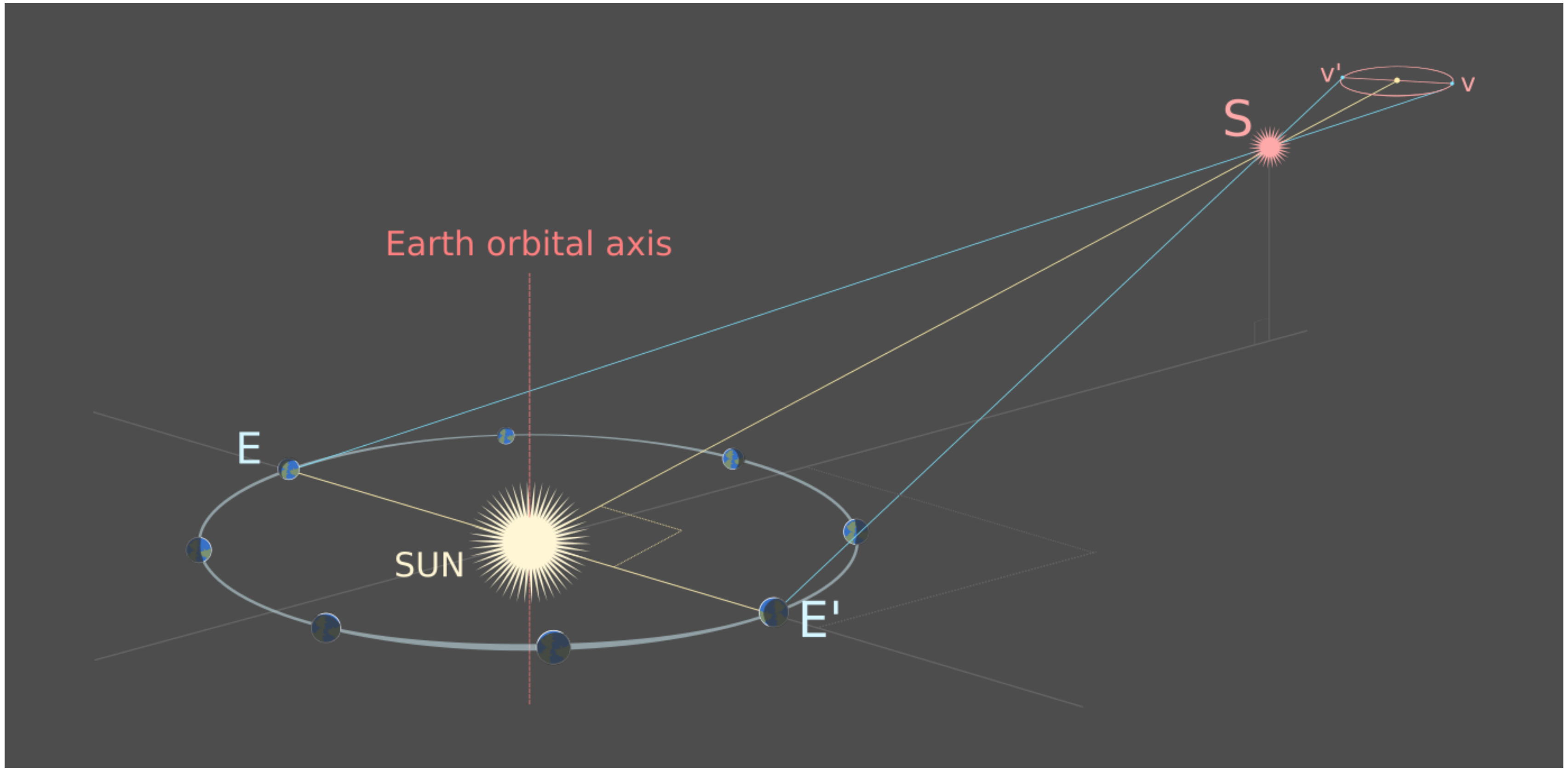


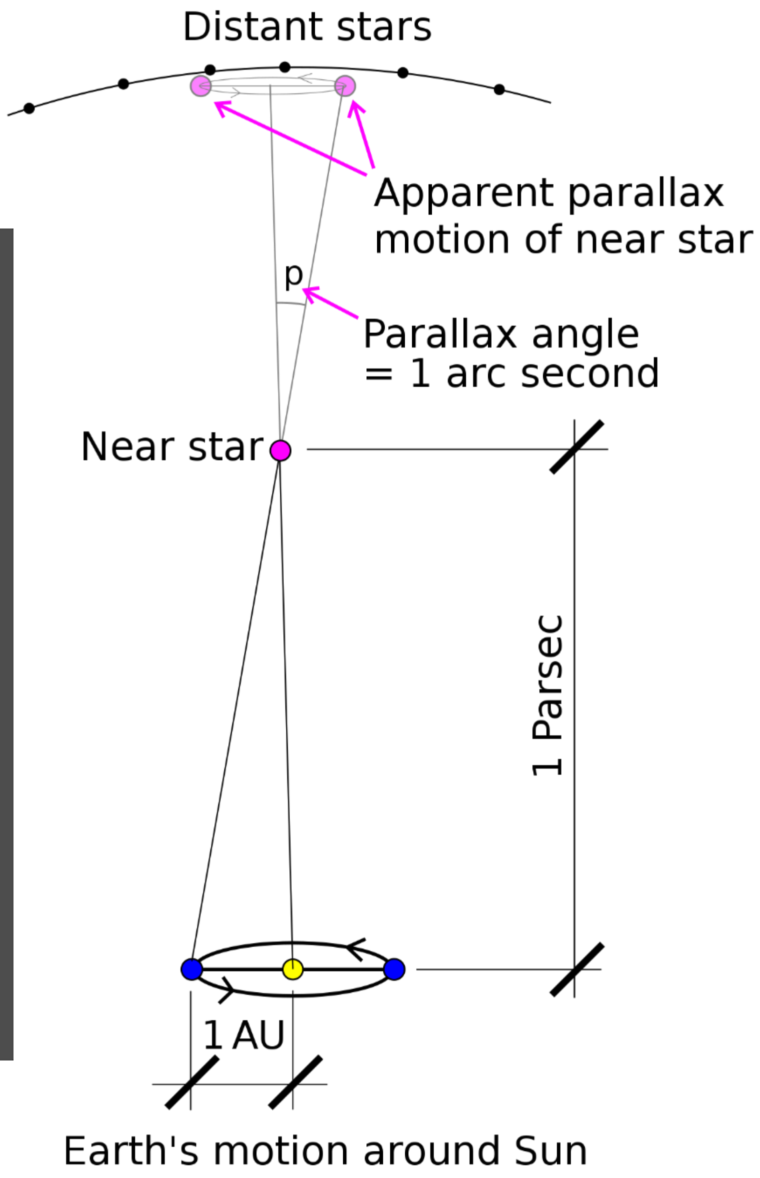
Tycho Brahe

It was one of Tycho Brahe's principal objections to Copernican heliocentrism that for it to be compatible with the lack of observable stellar parallax, there would have to be an enormous and unlikely void between the orbit of Saturn and the eighth sphere (the fixed stars).

The angles involved in these calculations are very small and thus difficult to measure. The nearest star to the Sun (and also the star with the largest parallax), Proxima Centauri, has a parallax of 0.7685 ± 0.0002 arcsec.[1] This angle is approximately that subtended by an object 2 centimeters in diameter located 5.3 kilometers away. First reliable measurements of parallax were not made until 1838, by Friedrich Bessel

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



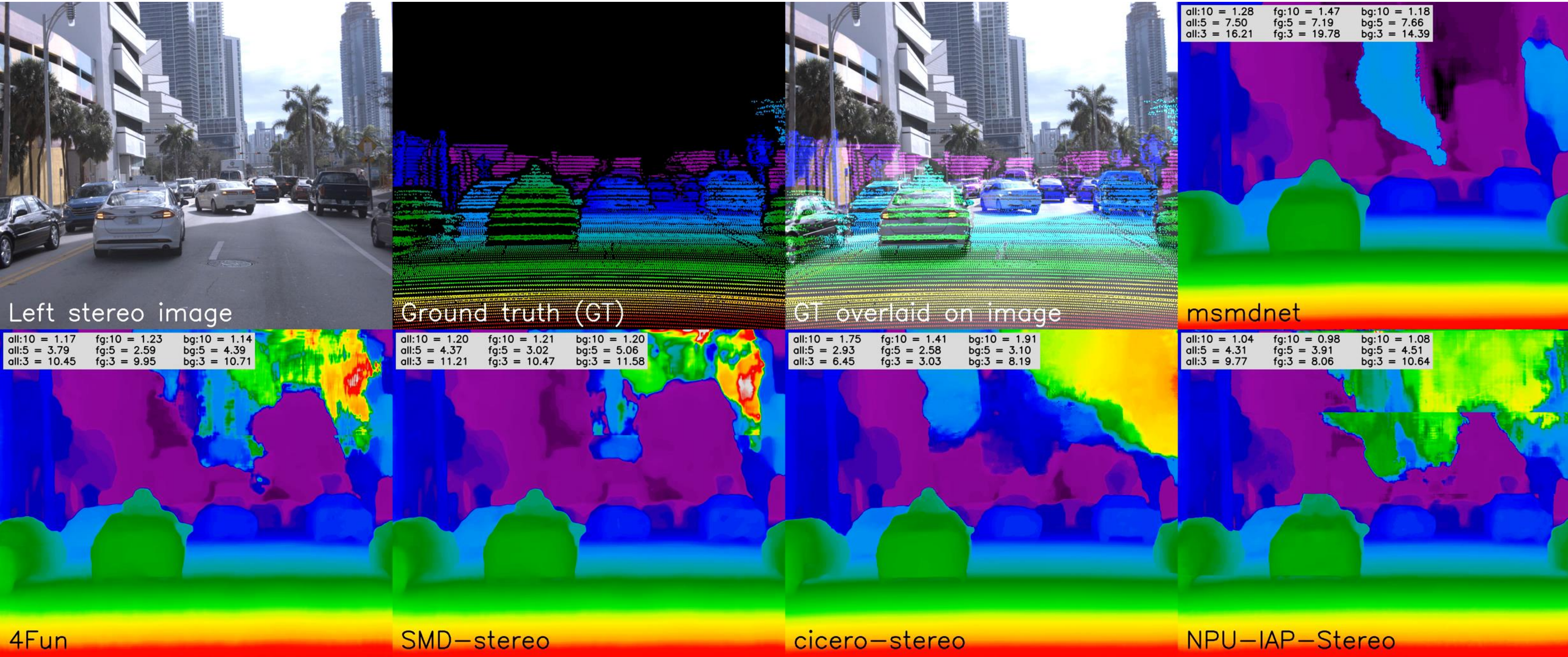

https://en.wikipedia.org/wiki/Stellar_parallax

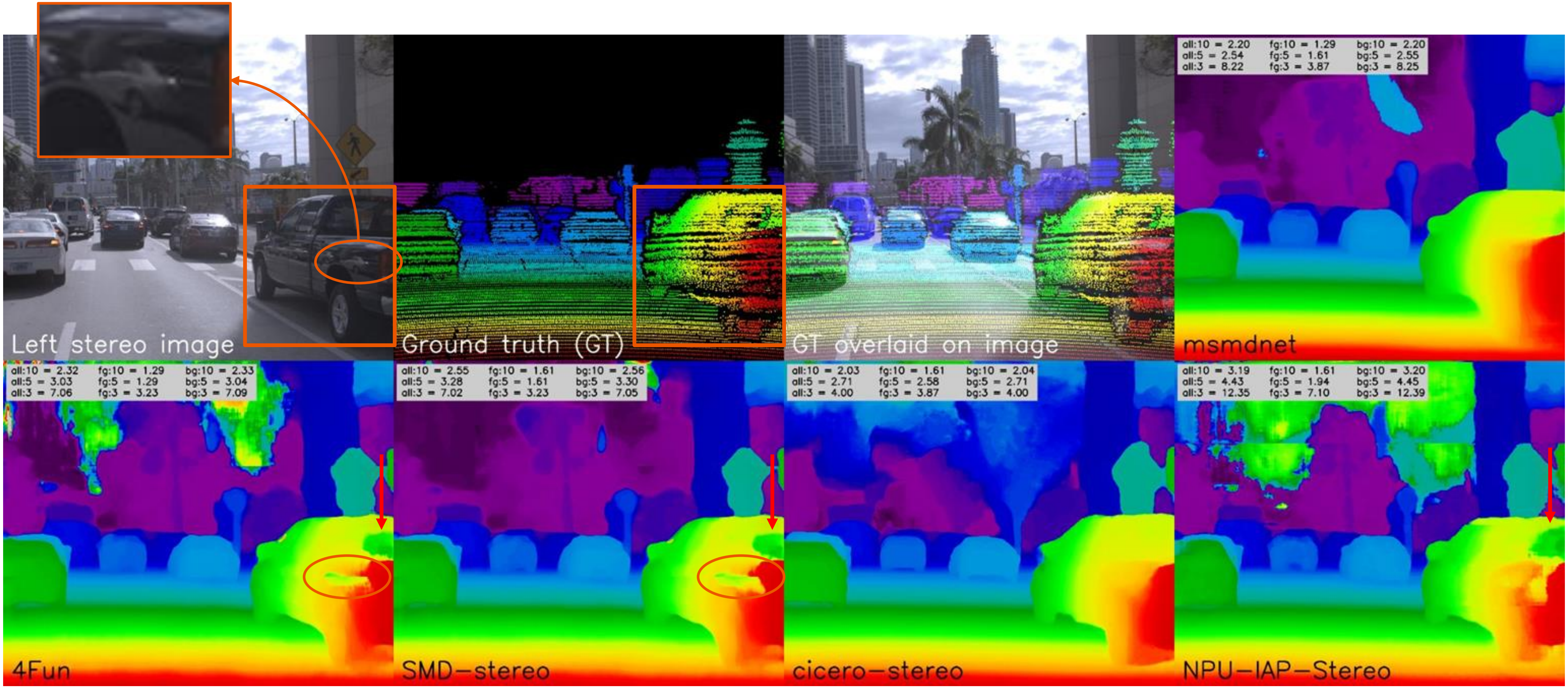
# Stereo vision

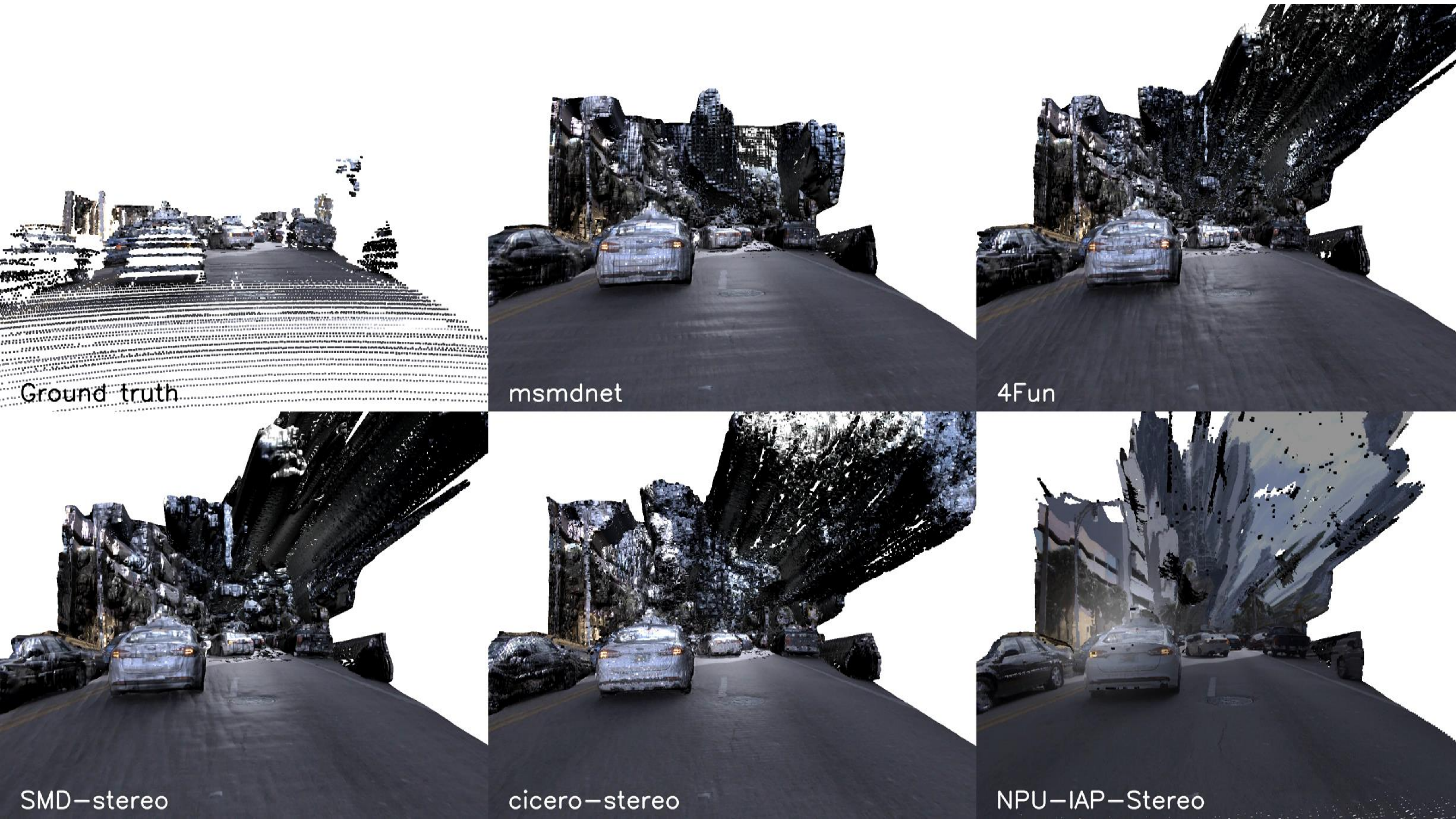

Two cameras, simultaneous views

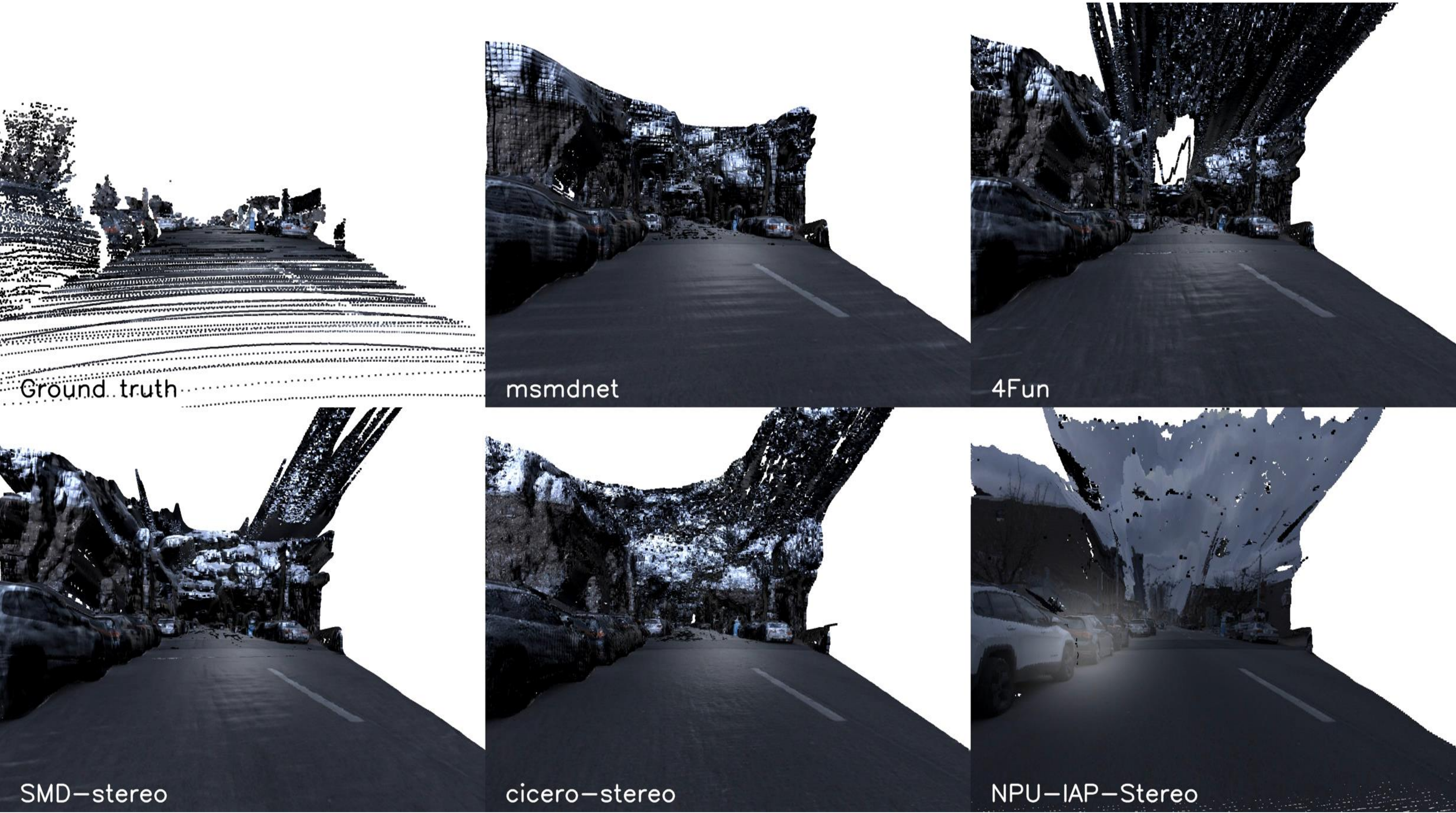
Single moving camera and static scene

# Modern stereo depth estimation example




Left stereo image
Ground truth (GT)
GT overlaid on image
msmdnet
all:10 = 2.20 fg:10 = 1.29 bg:10 = 2.20
all:5 = 2.54 fg:5 = 1.61 bg:5 = 2.55
all:3 = 8.22 fg:3 = 3.87 bg:3 = 8.25
4Fun
all:10 = 2.32 fg:10 = 1.29 bg:10 = 2.33
all:5 = 3.03 fg:5 = 1.29 bg:5 = 3.04
all:3 = 7.06 fg:3 = 3.23 bg:3 = 7.09
SMD−stereo
all:10 = 2.55 fg:10 = 1.61 bg:10 = 2.56
all:5 = 3.28 fg:5 = 1.61 bg:5 = 3.30
all:3 = 7.02 fg:3 = 3.23 bg:3 = 7.05
cicero−stereo
all:10 = 2.03 fg:10 = 1.61 bg:10 = 2.04
all:5 = 2.71 fg:5 = 2.58 bg:5 = 2.71
all:3 = 4.00 fg:3 = 3.87 bg:3 = 4.00
NPU−IAP−Stereo
all:10 = 3.19 fg:10 = 1.61 bg:10 = 3.20
all:5 = 4.43 fg:5 = 1.94 bg:5 = 4.45
all:3 = 12.35 fg:3 = 7.10 bg:3 = 12.39

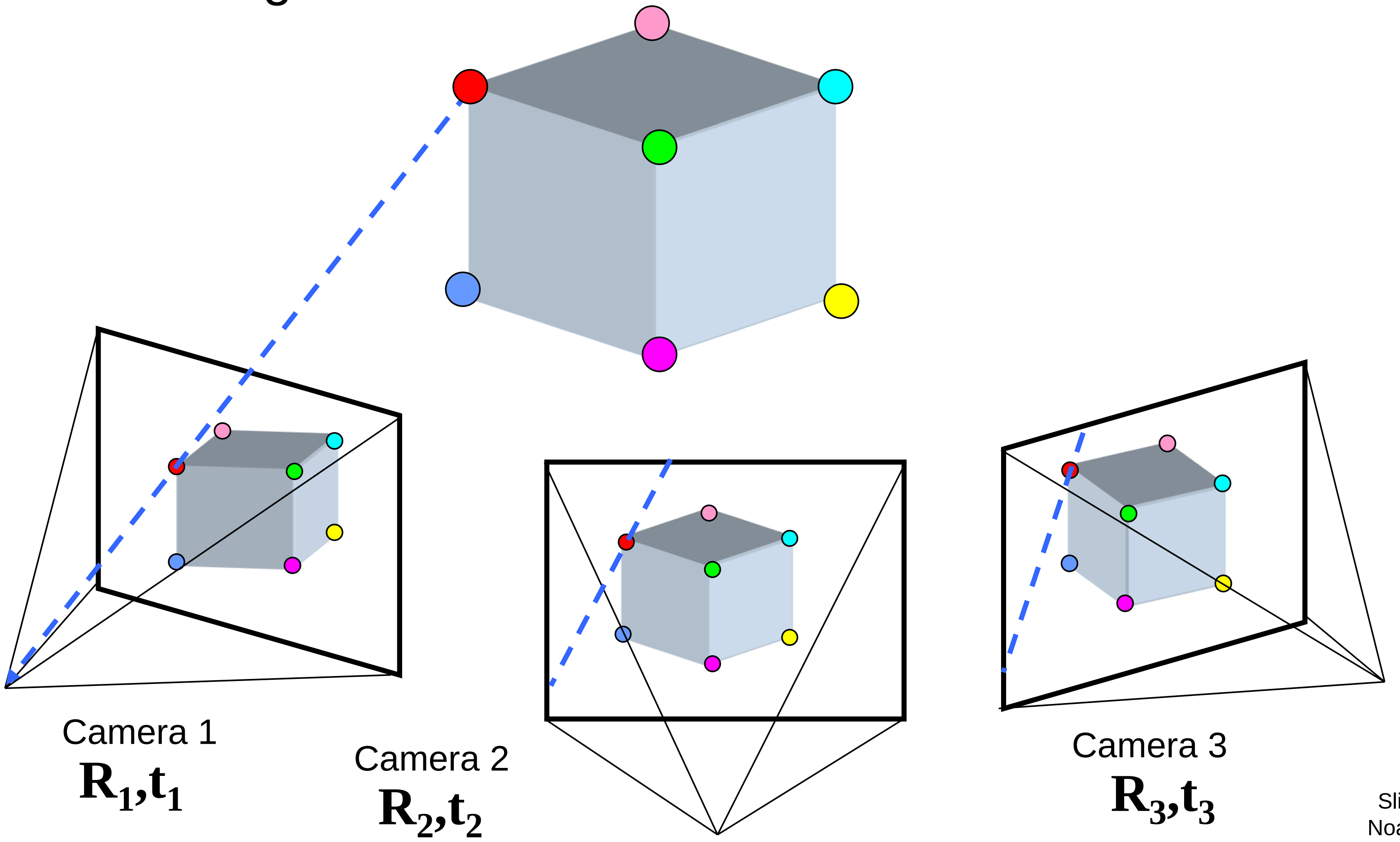

Ground truth
msmdnet
4Fun
SMD-stereo
cicero-stereo
NPU-IAP-Stereo



Ground truth
msmdnet
4Fun
SMD-stereo
cicero-stereo
NPU-IAP-Stereo

# Multi-view geometry problems

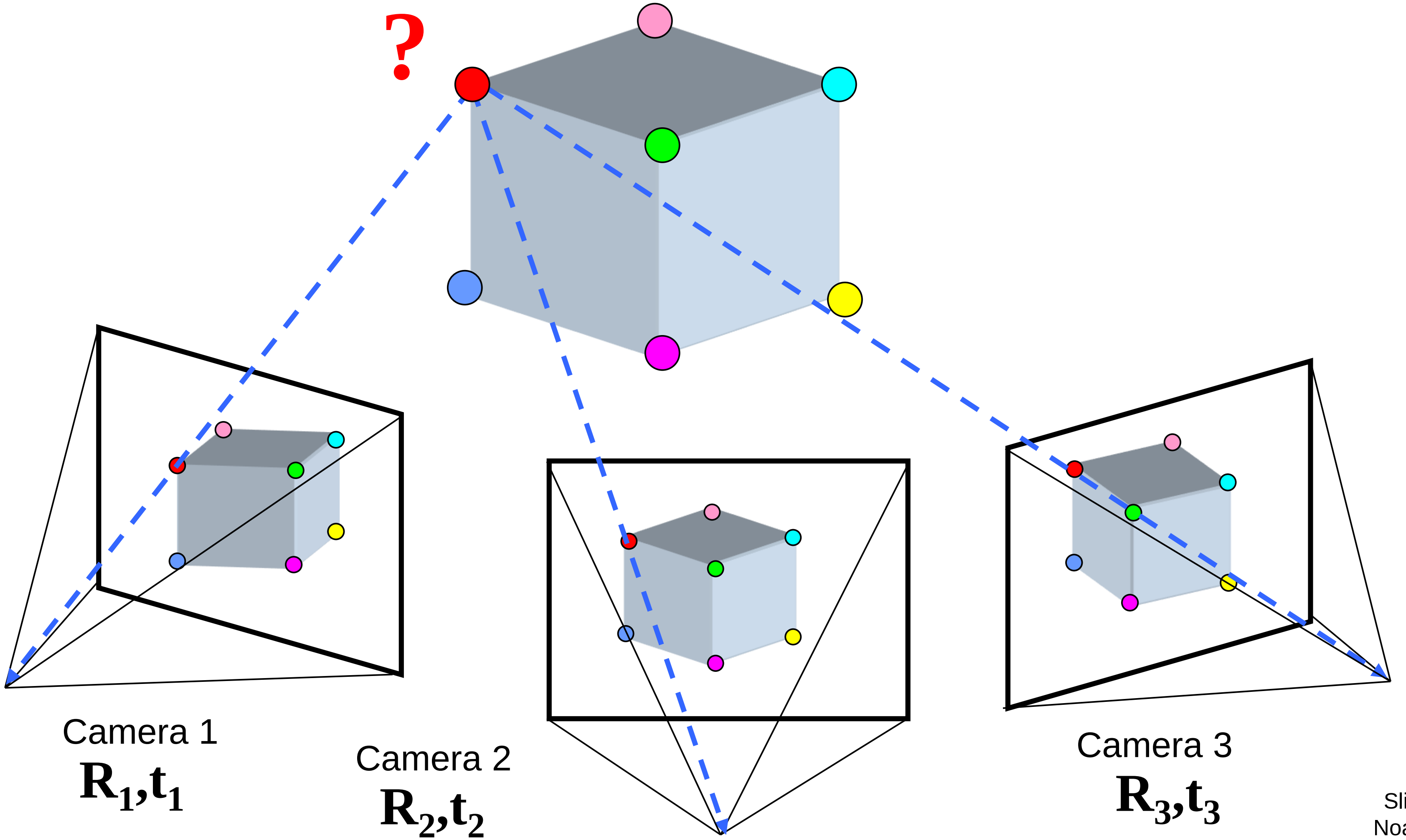
- **Stereo correspondence:** Given a point in one of the images, where could its corresponding points be in the other images?




Slide credit: Noah Snavely

# Multi-view geometry problems

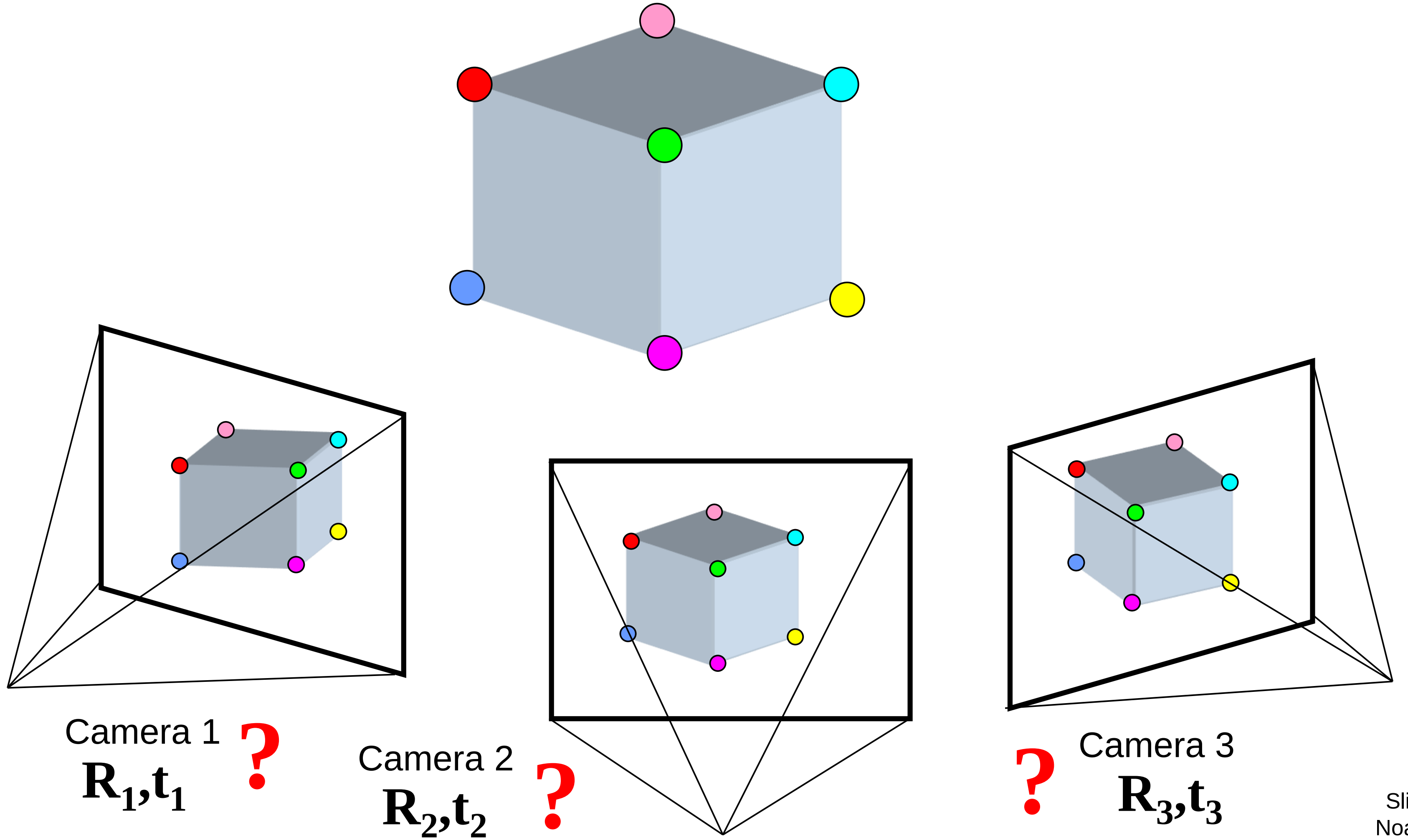
- **Structure:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point




Slide credit: Noah Snavely

# Multi-view geometry problems

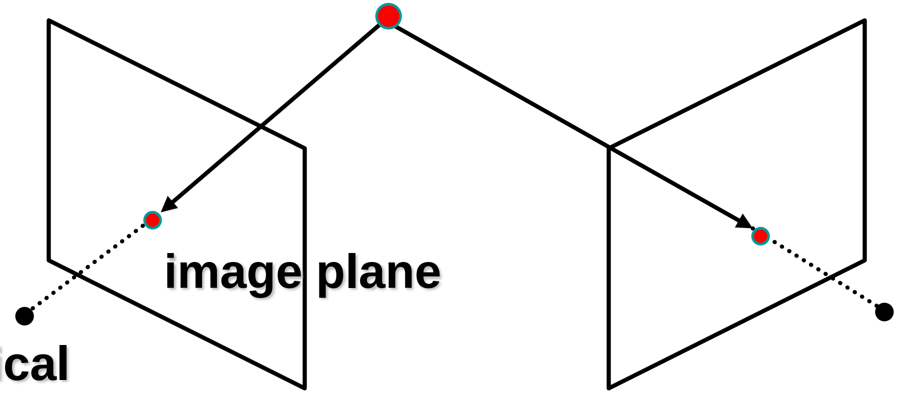
- **Motion:** Given a set of corresponding points in two or more images, compute the camera parameters

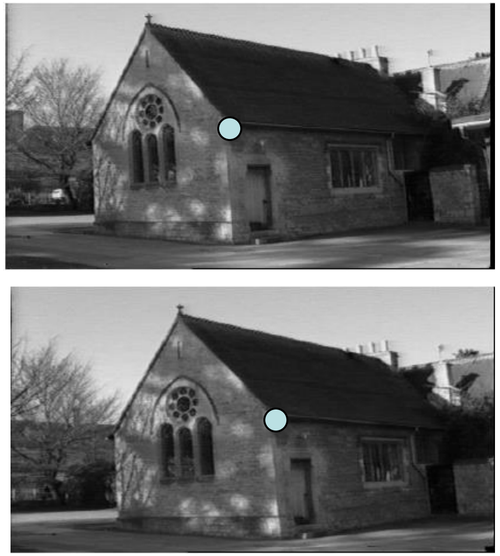



Slide credit: Noah Snavely

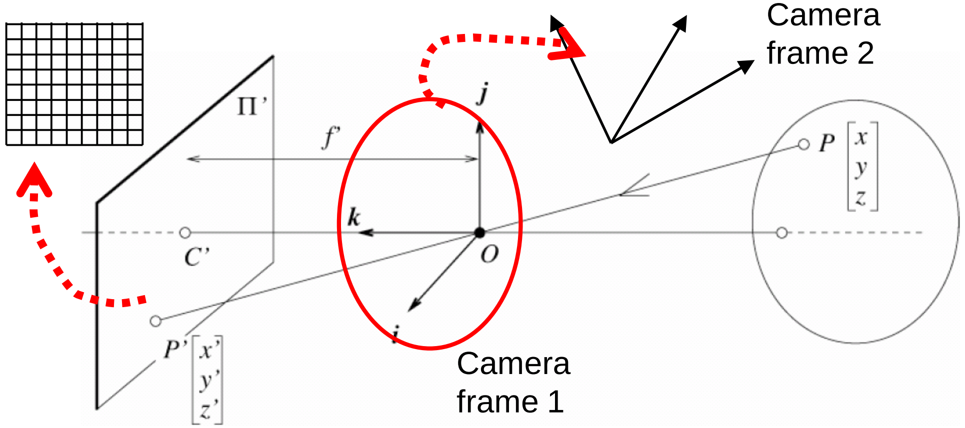
# Estimating depth with stereo

- **Stereo**: shape from “motion” between two views
- We’ll need to consider:
  - Info on camera pose (“calibration”)
  - Image point correspondences



# Camera parameters




**Extrinsic** parameters:
Camera frame 1 ←→ Camera frame 2

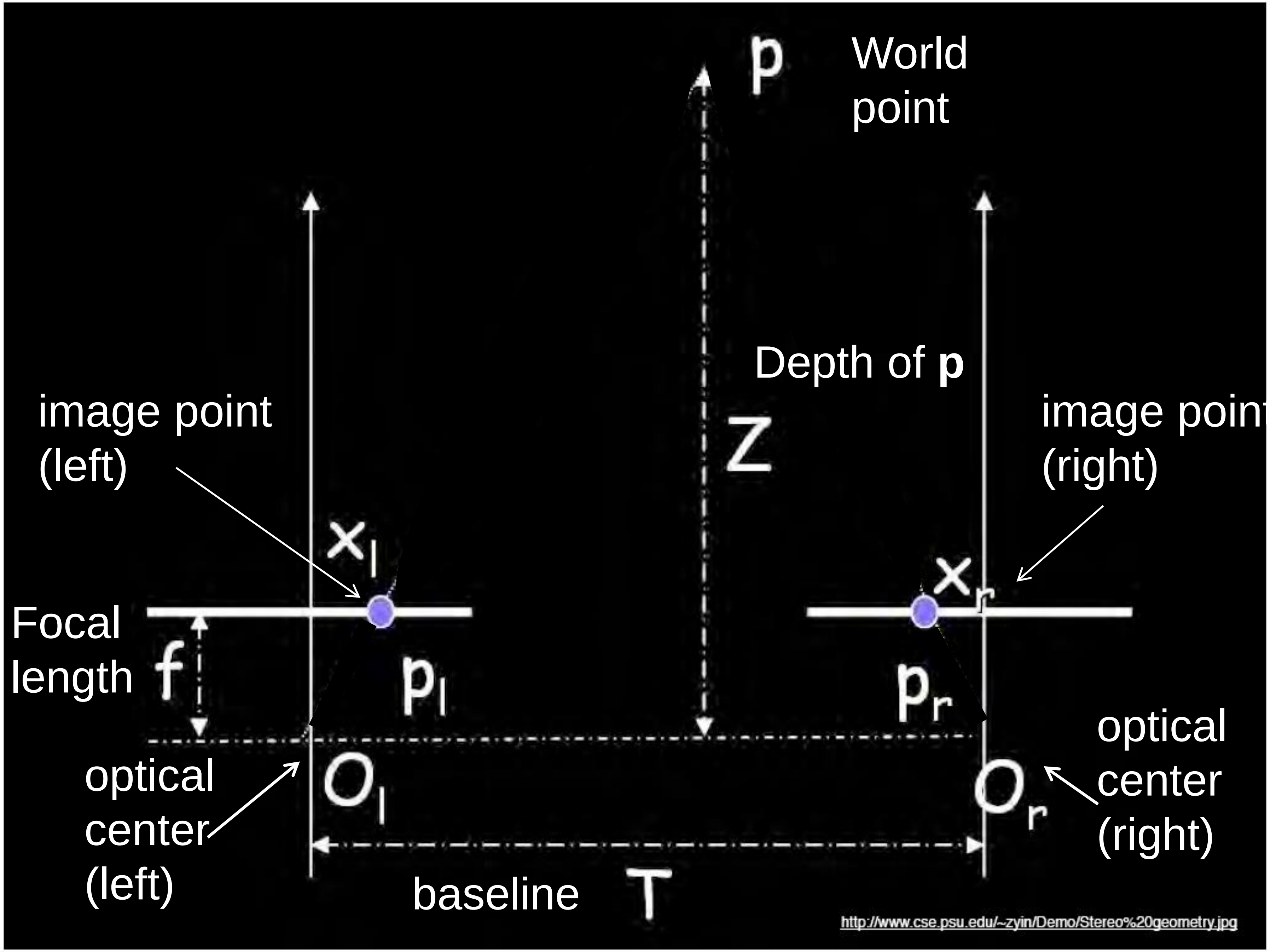
**Intrinsic** parameters:
Image coordinates relative to camera ←→ Pixel coordinates

- *Extrinsic* params: rotation matrix and translation vector
- *Intrinsic* params: focal length, pixel sizes (mm), image center point, radial distortion parameters

*We'll assume for now that these parameters are given and fixed.*

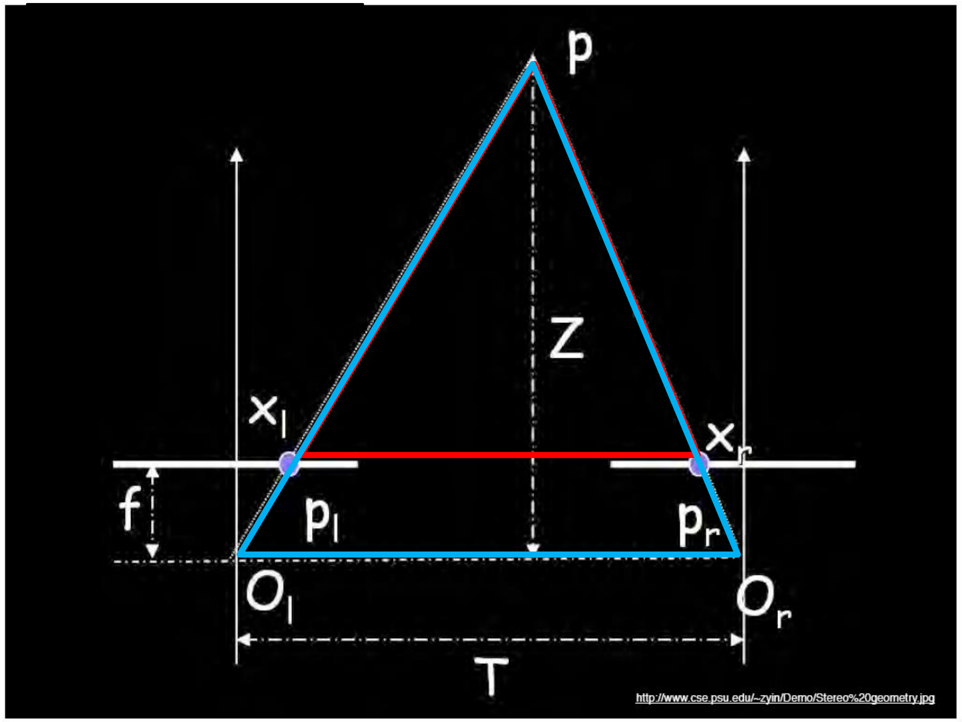
# Geometry for a simple stereo system

- First, assuming parallel optical axes, known camera parameters (i.e., calibrated cameras):


p
World
point
Depth of **p**
Z
image point
(left)
image point
(right)
$x_l$
$x_r$
Focal
length
f
$p_l$
$p_r$
optical
center
(left)
$O_l$
$O_r$
optical
center
(right)
baseline
T
http://www.cse.psu.edu/~zyin/Demo/Stereo%20geometry.jpg

# Geometry for a simple stereo system

- Assume parallel optical axes, known camera parameters (i.e., calibrated cameras). **What is expression for Z?**




Similar triangles ($p_l$, P, $p_r$) and ($O_l$, P, $O_r$):

$$\frac{T - x_l + x_r}{Z - f} = \frac{T}{Z}$$

$$Z = f \frac{T}{x_l - x_r}$$

**disparity**

# Depth from disparity

image I(x,y)

Disparity map D(x,y)

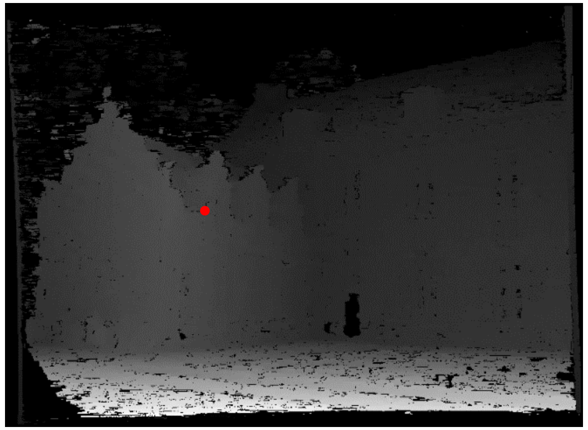
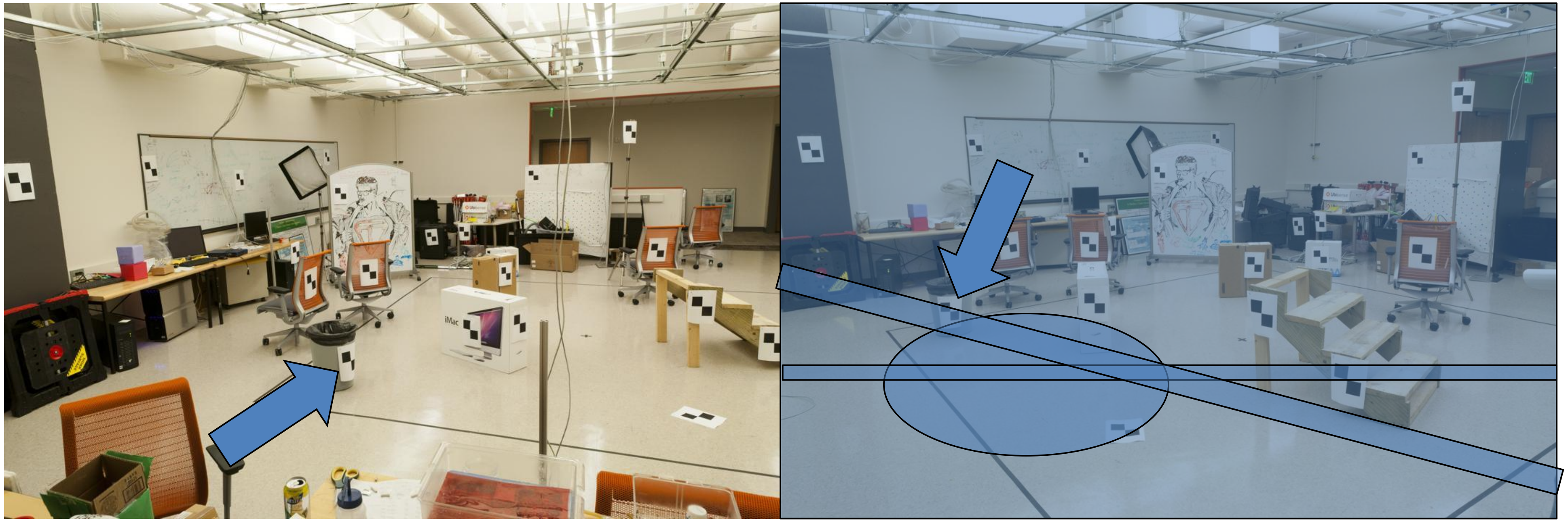


image I´(x´,y´)

(x´,y´)=(x+D(x,y), y)

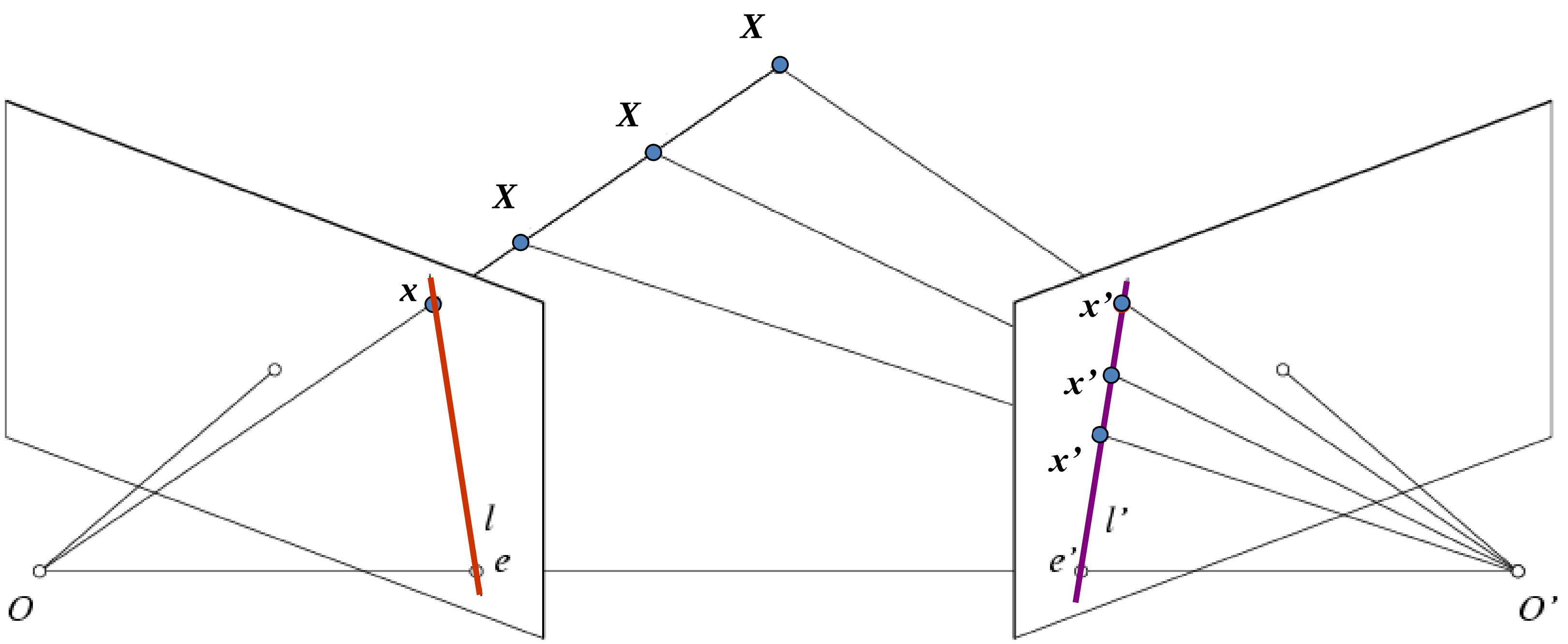
So if we could find the **corresponding points** in two images, we could **estimate relative depth**…

# Where do we need to search?



# Key idea: Epipolar constraint

# Key idea: Epipolar constraint

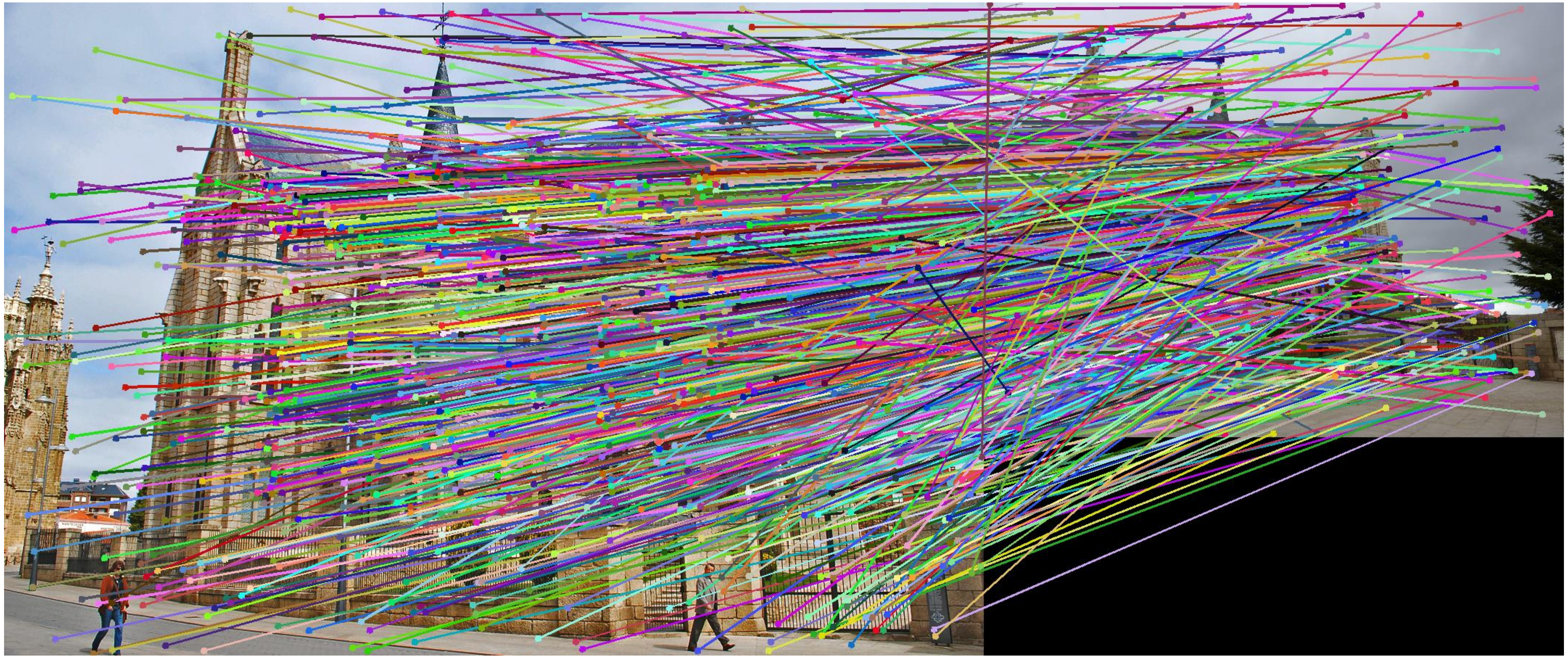



Potential matches for *x* have to lie on the corresponding line *l'*.

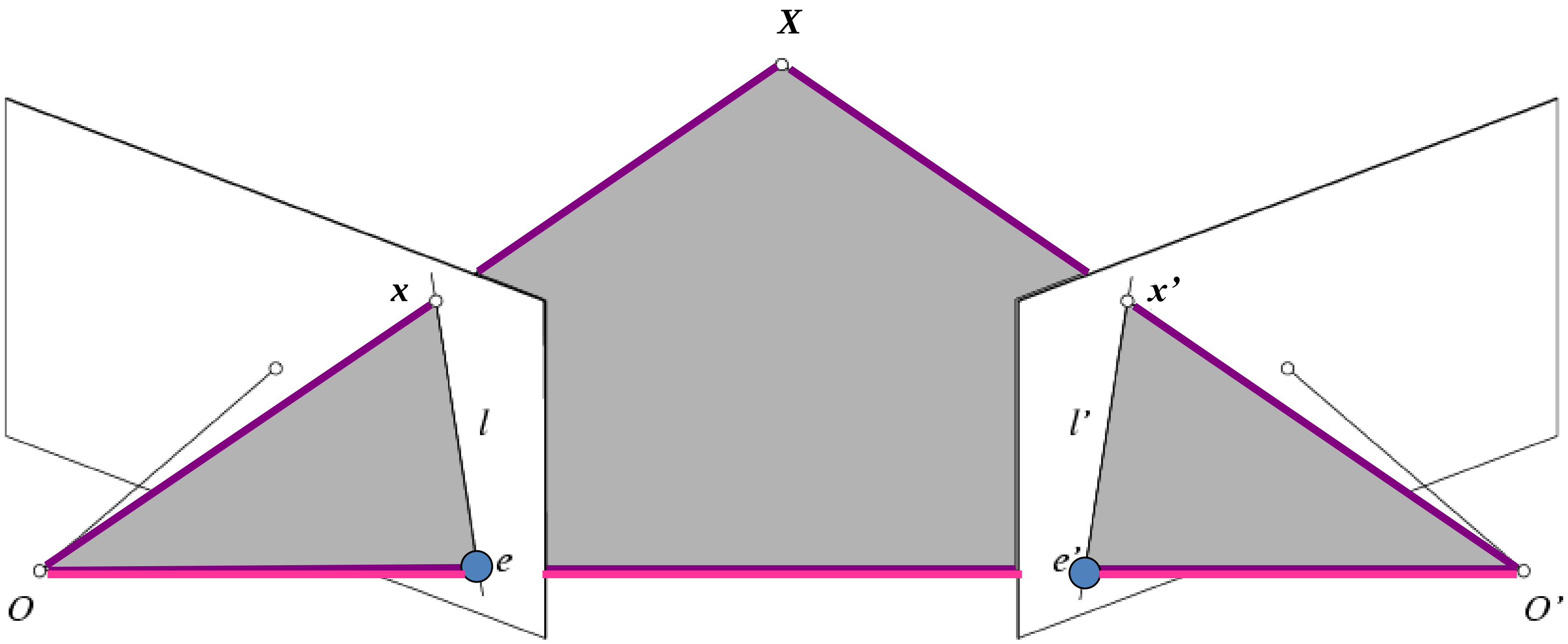
Potential matches for *x'* have to lie on the corresponding line *l*.

Wouldn’t it be nice to know where matches can live? To constrain our 2d search to 1d.

# VLFeat's 800 most confident matches among 10,000+ local features.

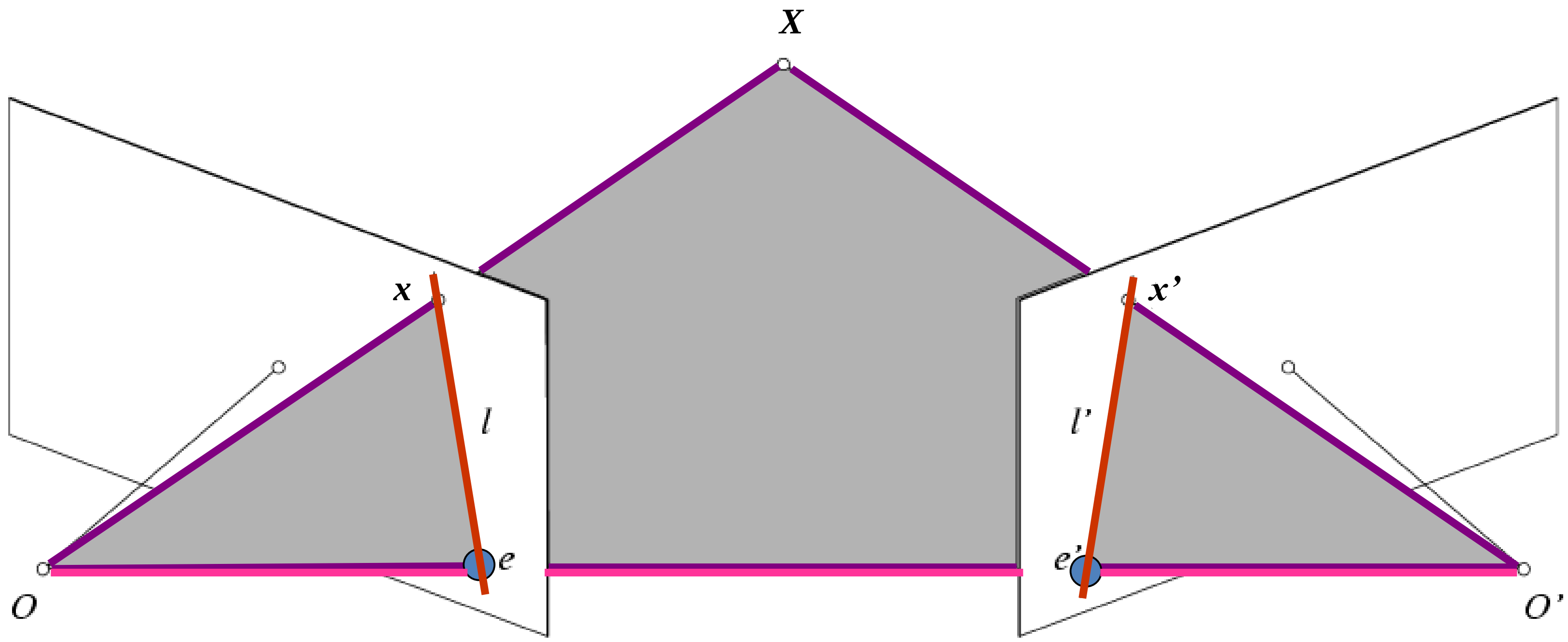


# Epipolar geometry: notation



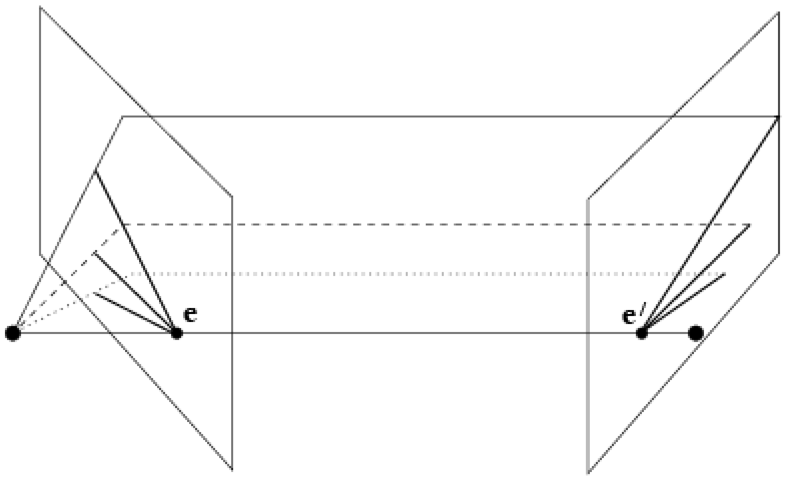

- **Baseline** – line connecting the two camera centers
- **Epipoles**
  = intersections of baseline with image planes
  = projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)

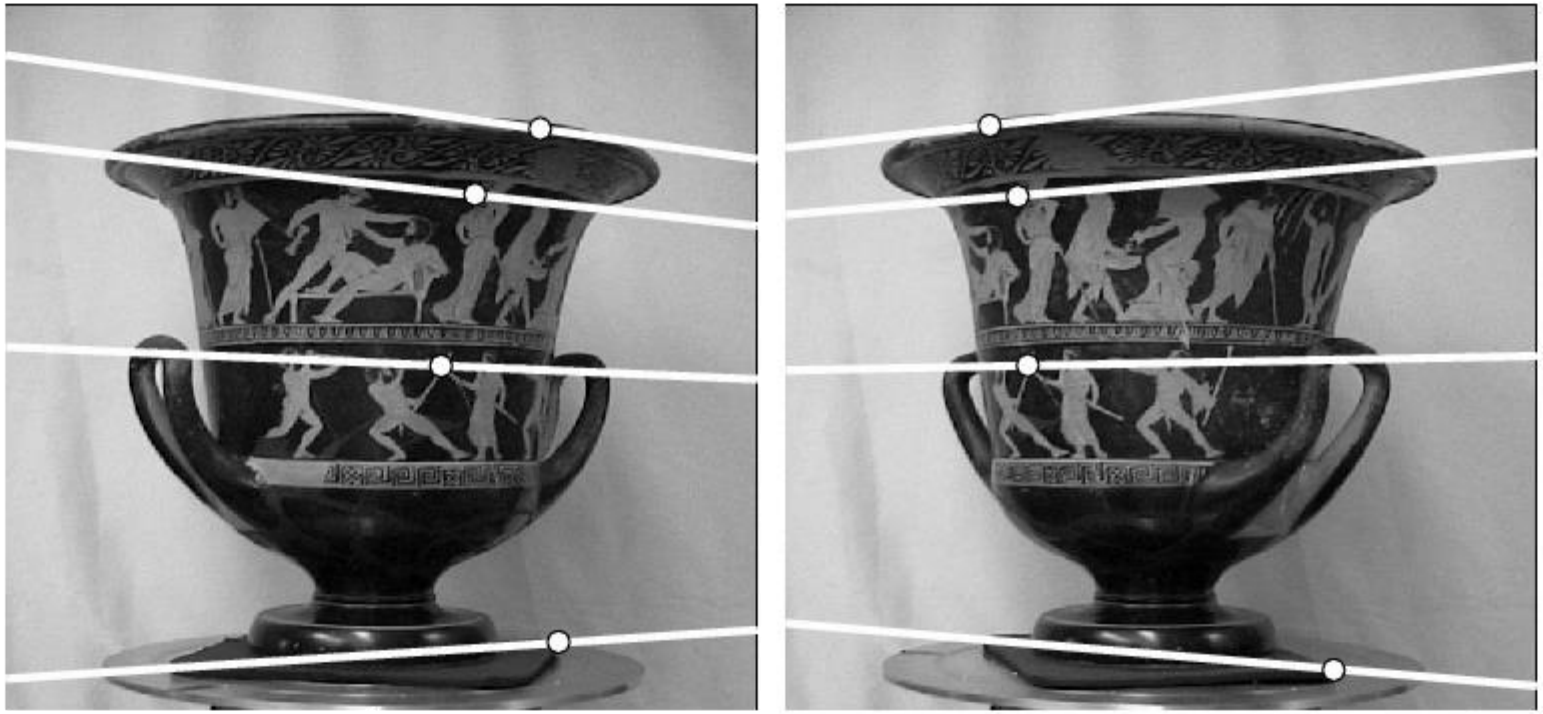
# Epipolar geometry: notation



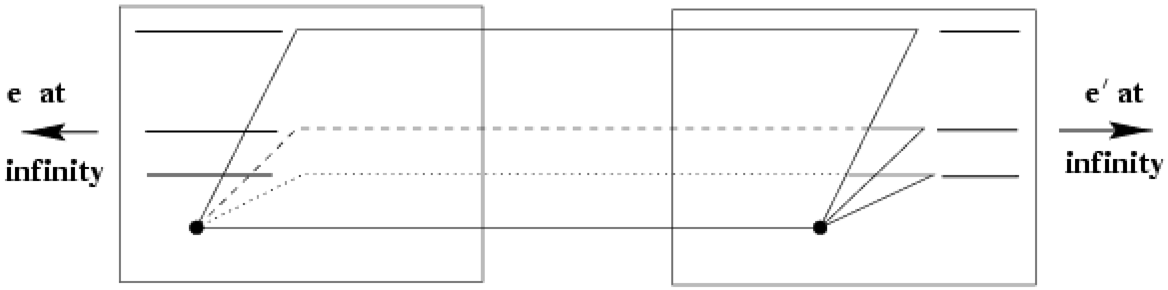

- **Baseline** – line connecting the two camera centers
- **Epipoles**
= intersections of baseline with image planes
= projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

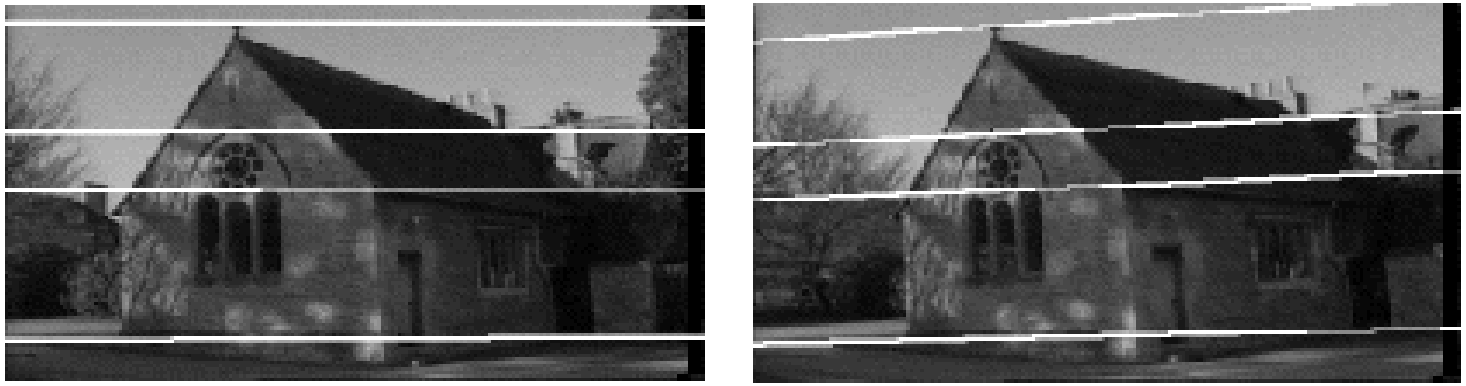
# Example: Converging cameras



# Example: Motion parallel to image plane

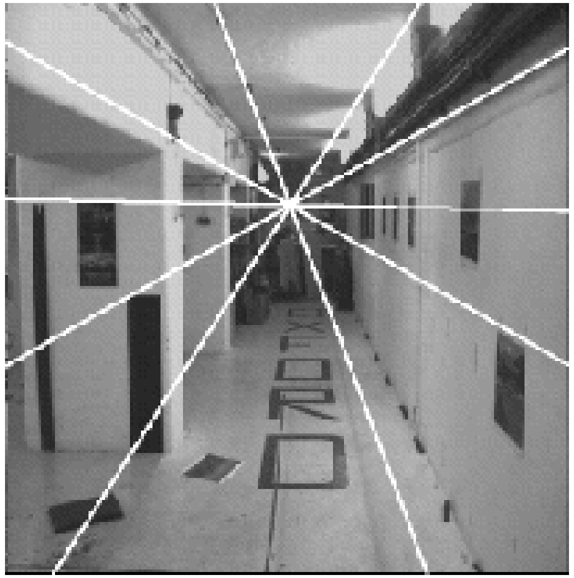


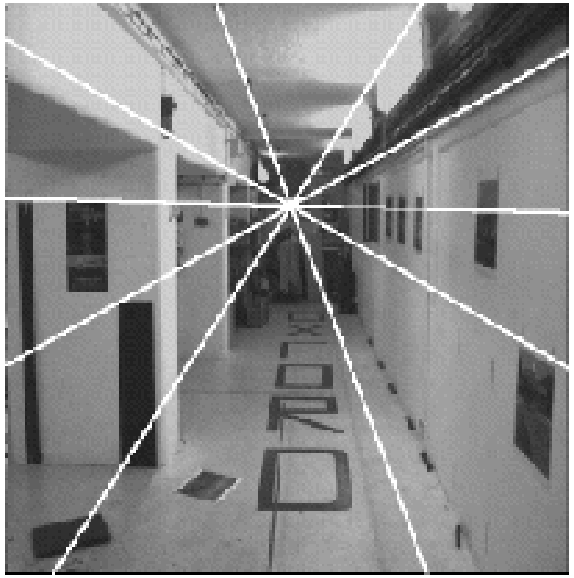



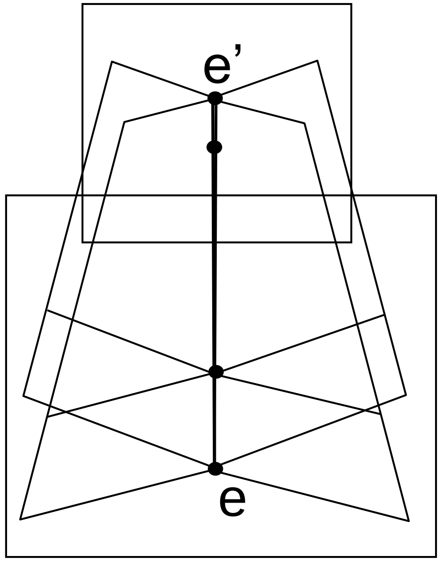
# Example: Forward motion

What would the epipolar lines look like if the camera moves directly forward?

# Example: Forward motion

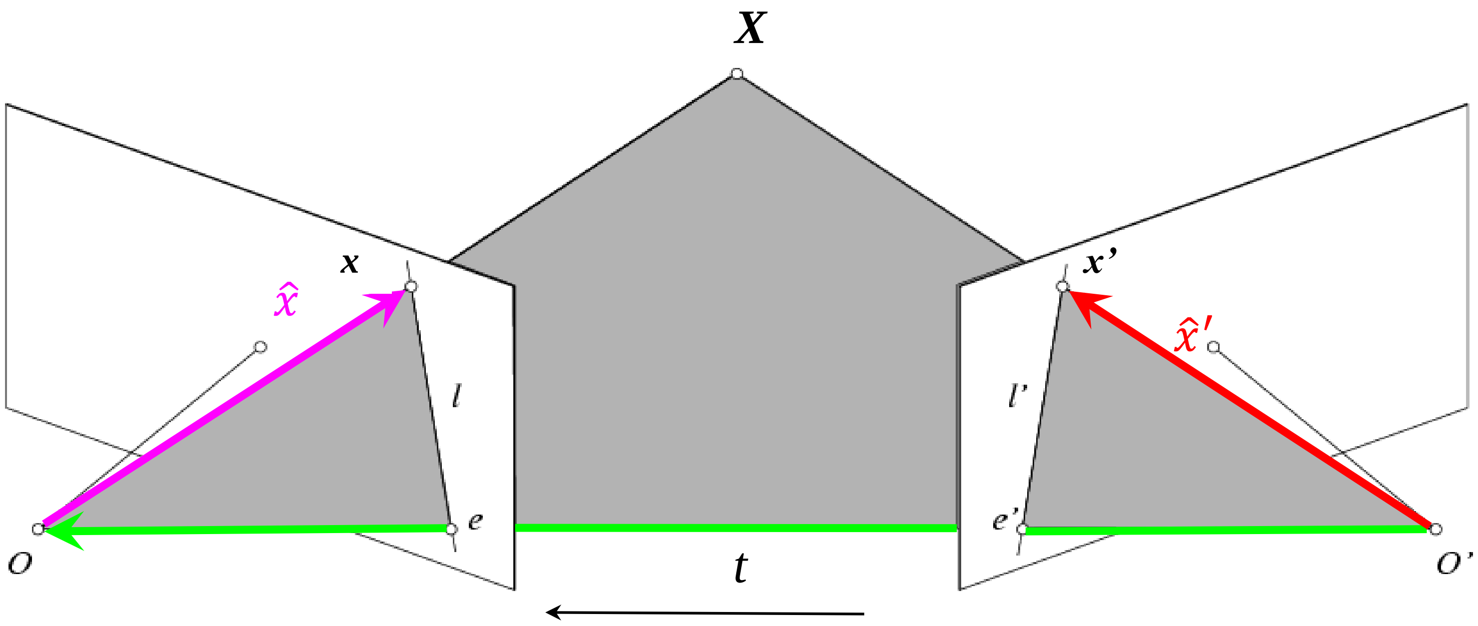







Epipole has same coordinates in both images.
Points move along lines radiating from e: “Focus of expansion”

# Epipolar constraint: Calibrated case




$$\hat{x} = K^{-1}x = X \qquad \hat{x}' = K'^{-1}x' = X'$$

$$\hat{x} \cdot [t \times (R\hat{x}')] = 0$$

(because $\hat{x}, R\hat{x}'$, and $t$ are co-planar)

# To be continued