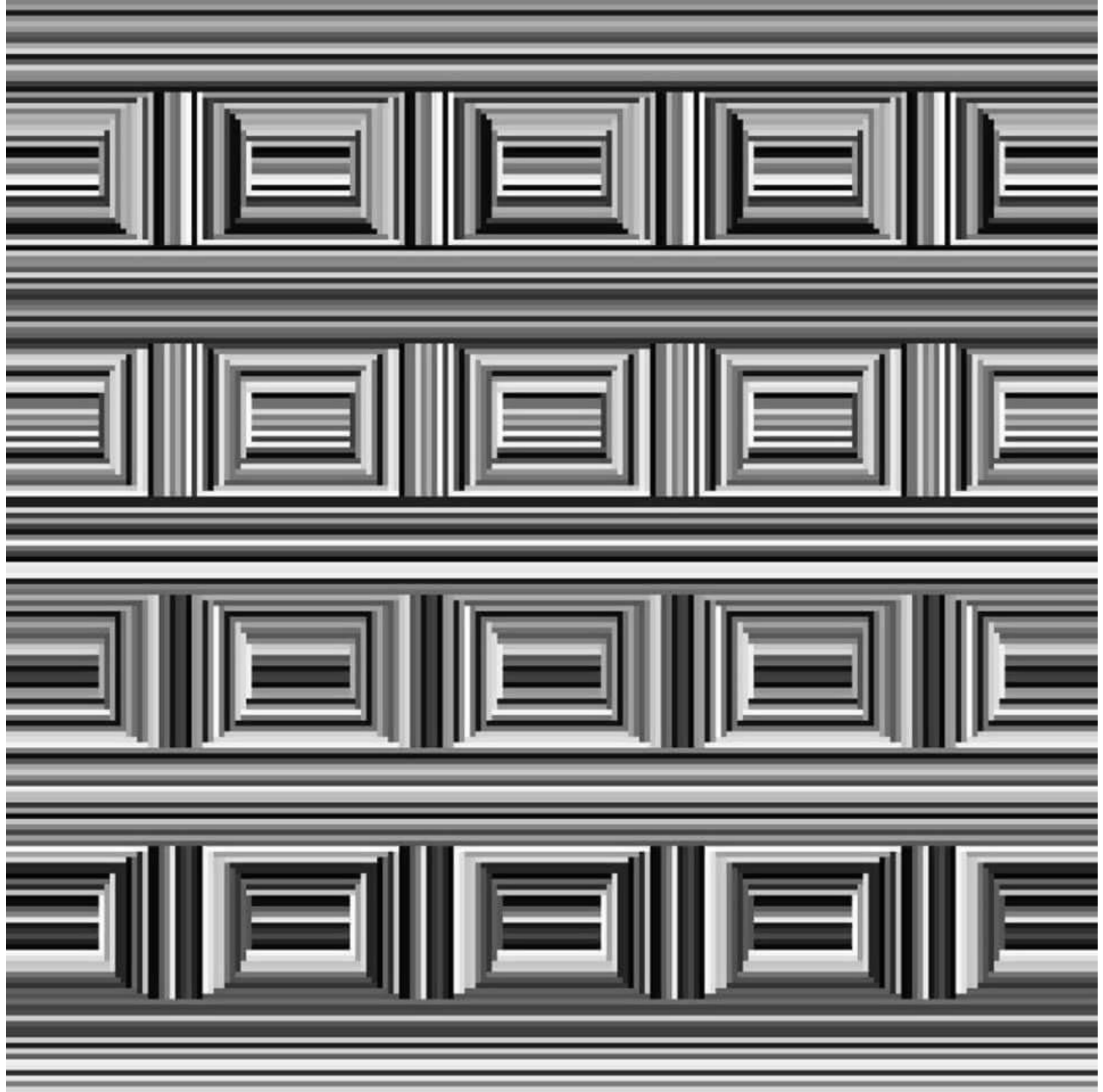
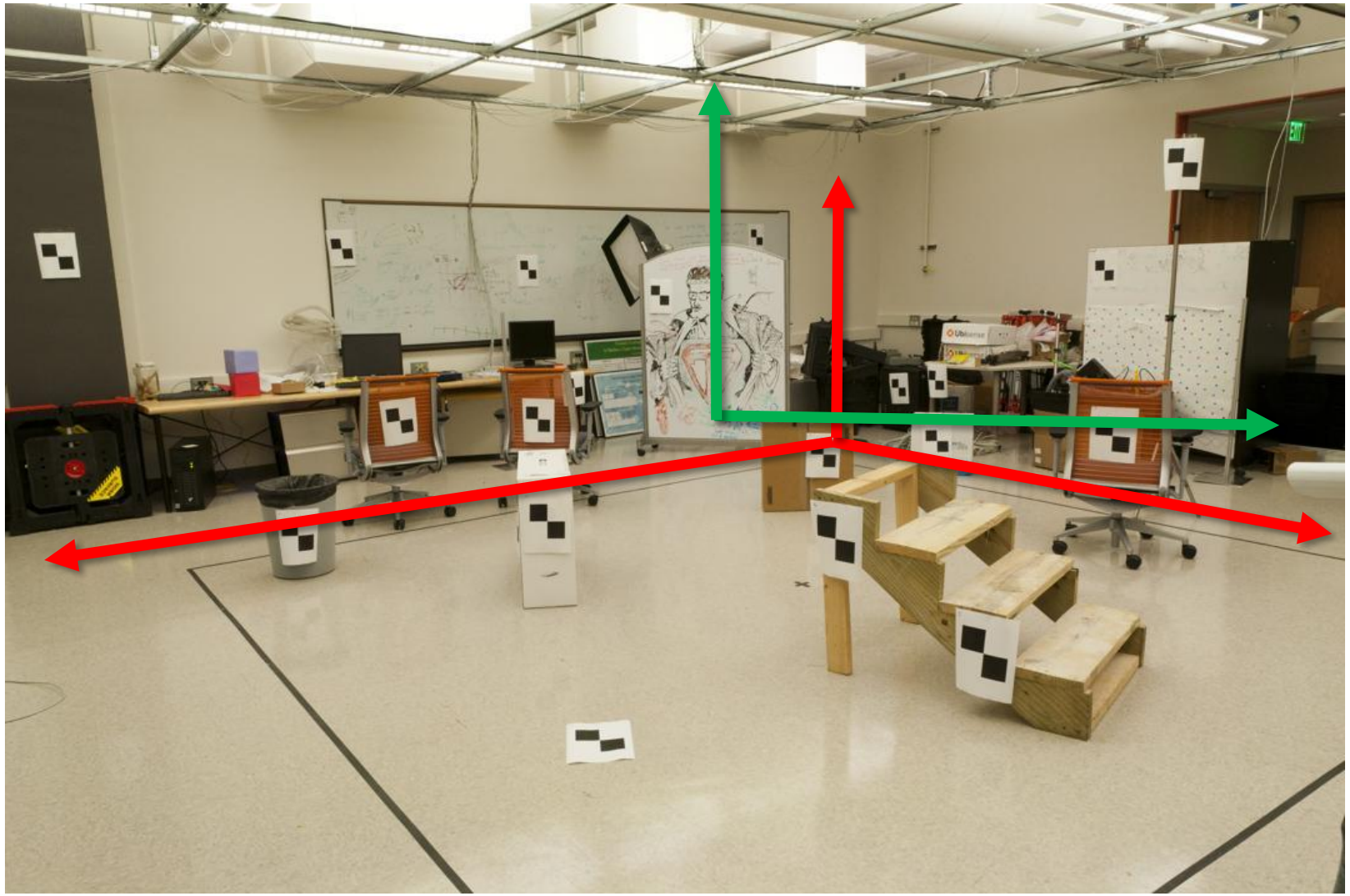


- Stereo and lidar can fall victim to reflections?
- Yes, there’s no easy way around that
- https://youtu.be/pBzU8TD1iks

Previous lecture: World vs Camera coordinates



Previous lecture:

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

4 Part 4: Camera projection matrix

Introduction

The goal is to compute the projection matrix that goes from world 3D coordinates to 2D image coordinates. Recall that using homogeneous coordinates the equation for moving from 3D world to 2D camera coordinates is:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} u*s \\ v*s \\ s \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3}$$

Another way of writing this equation is:

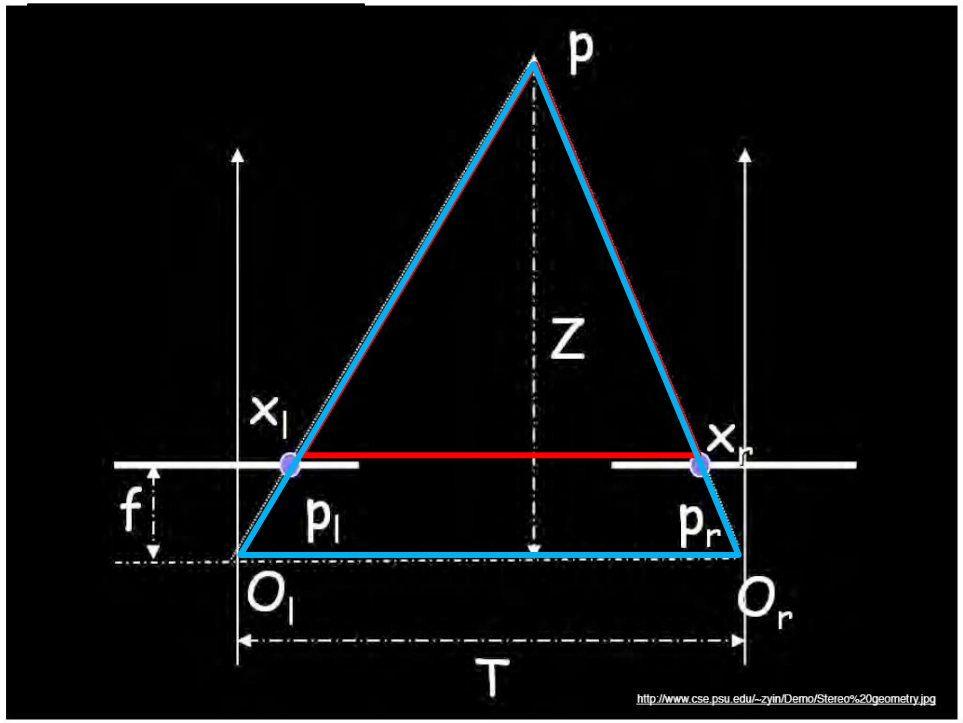
$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \tag{4}$$

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

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

Previous Lecture: 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)

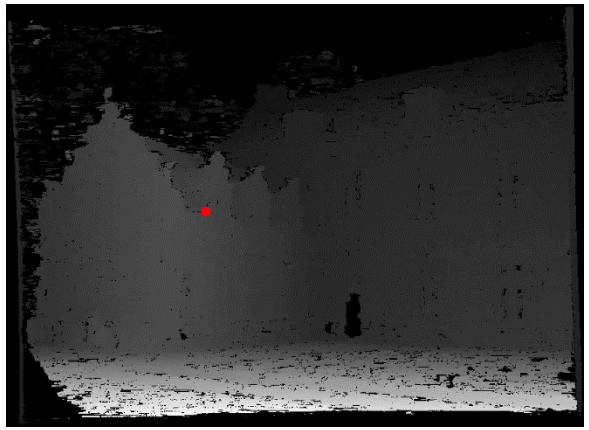
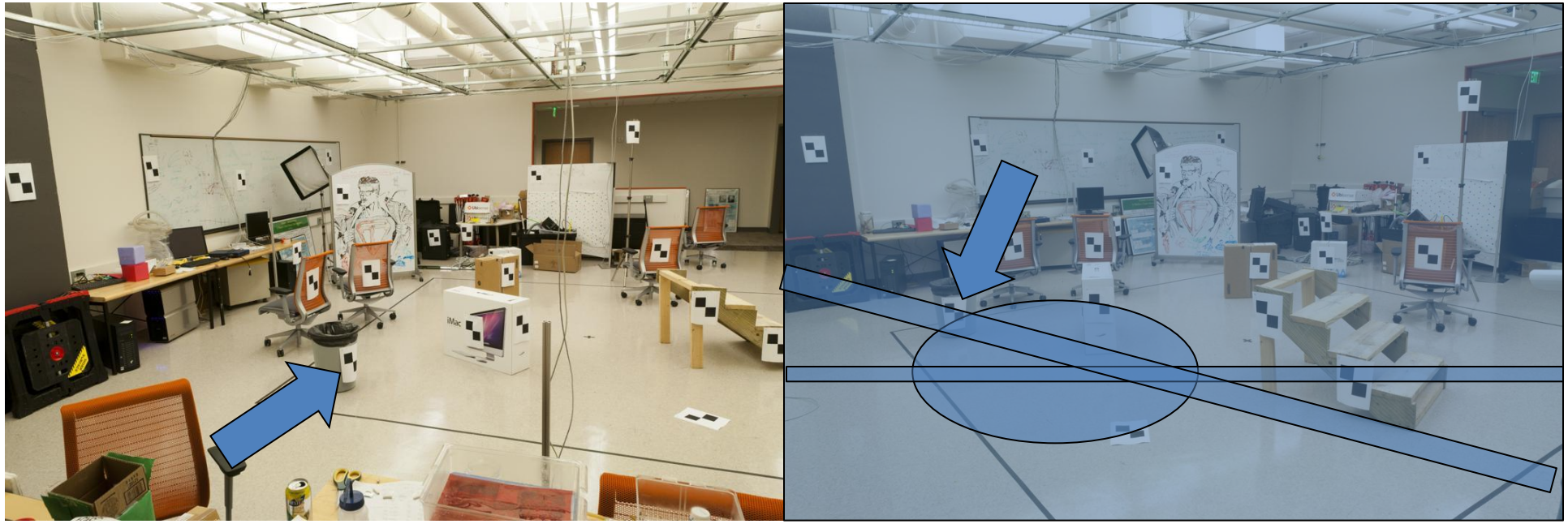


image I´(x´,y´)

$$(x´,y´)=(x+D(x,y),\ y)$$

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

If we have a 2D point of interest, where do we need to search for its corresponding point in another view?



Today’s Outline

- Epipolar Geometry
  - Finding epipolar relationship between two images
  - Using epipolar geometry to rule out outliers
  - Finding dense correspondence along epipolar lines

Epipolar Geometry and Stereo Vision

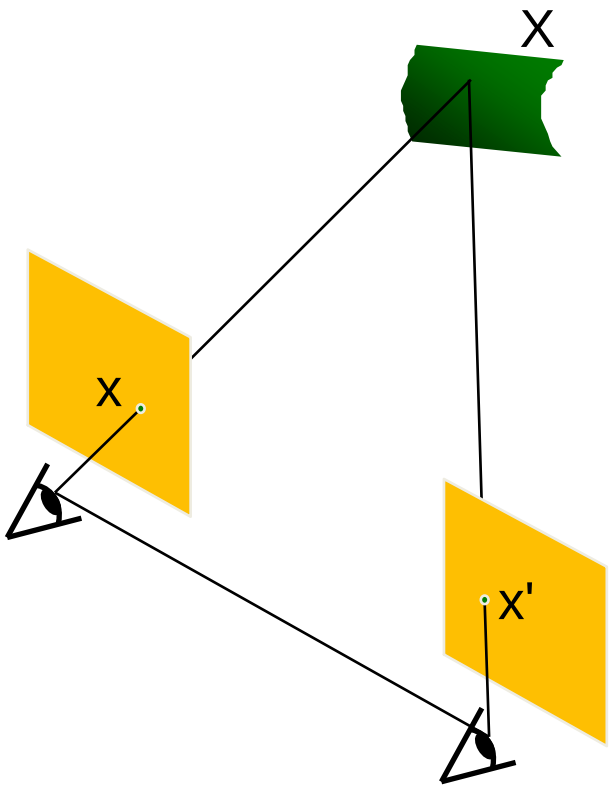
Chapter 11.3 in Szeliski

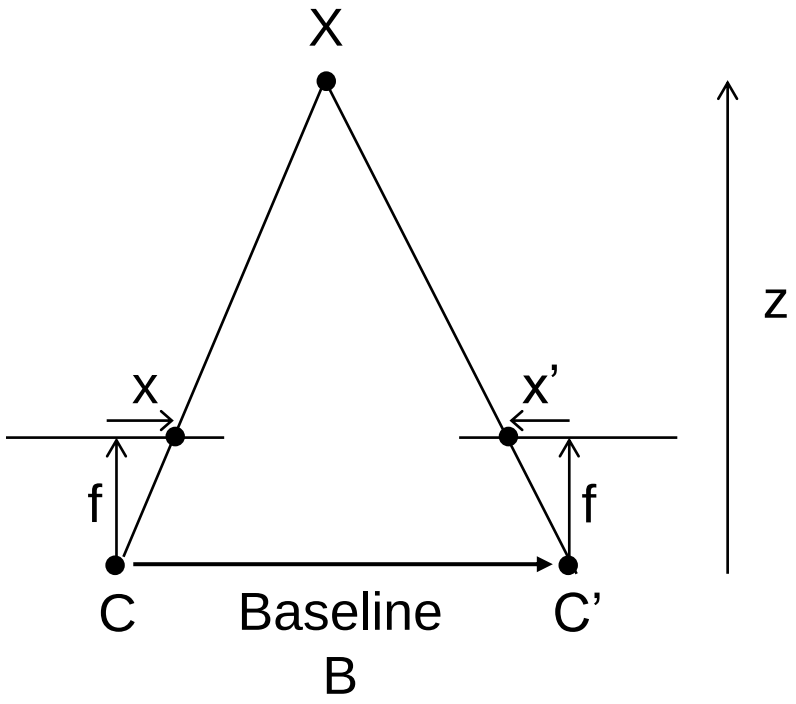
Many slides adapted from Derek Hoiem, Lana Lazebnik, Silvio Saverese, Steve Seitz, many figures from Hartley & Zisserman

- Epipolar geometry
  - Relates cameras from two positions

Depth from Stereo

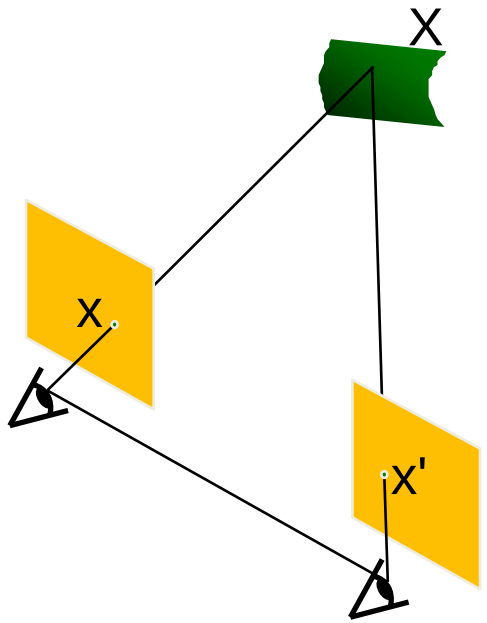
- Goal: recover depth by finding image coordinate x’ that corresponds to x





Depth from Stereo

- Goal: recover depth by finding image coordinate x’ that corresponds to x
- Sub-Problems
  1. Calibration: How do we recover the relation of the cameras (if not already known)?
  2. Correspondence: How do we search for the matching point x’?



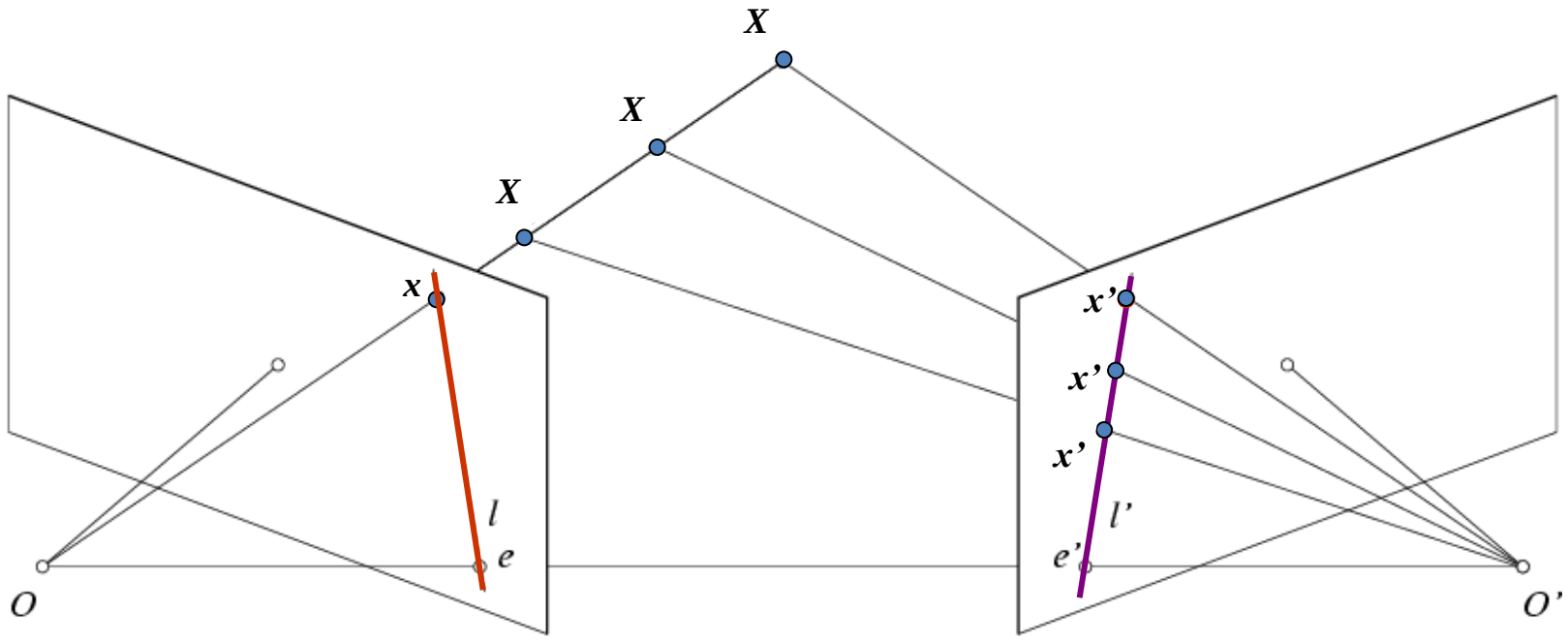
Correspondence Problem

- We have two images taken from cameras with different intrinsic and extrinsic parameters

- How do we match a point in the first image to a point in the second? How can we constrain our search?

Key idea: Epipolar constraint

Key idea: Epipolar constraint

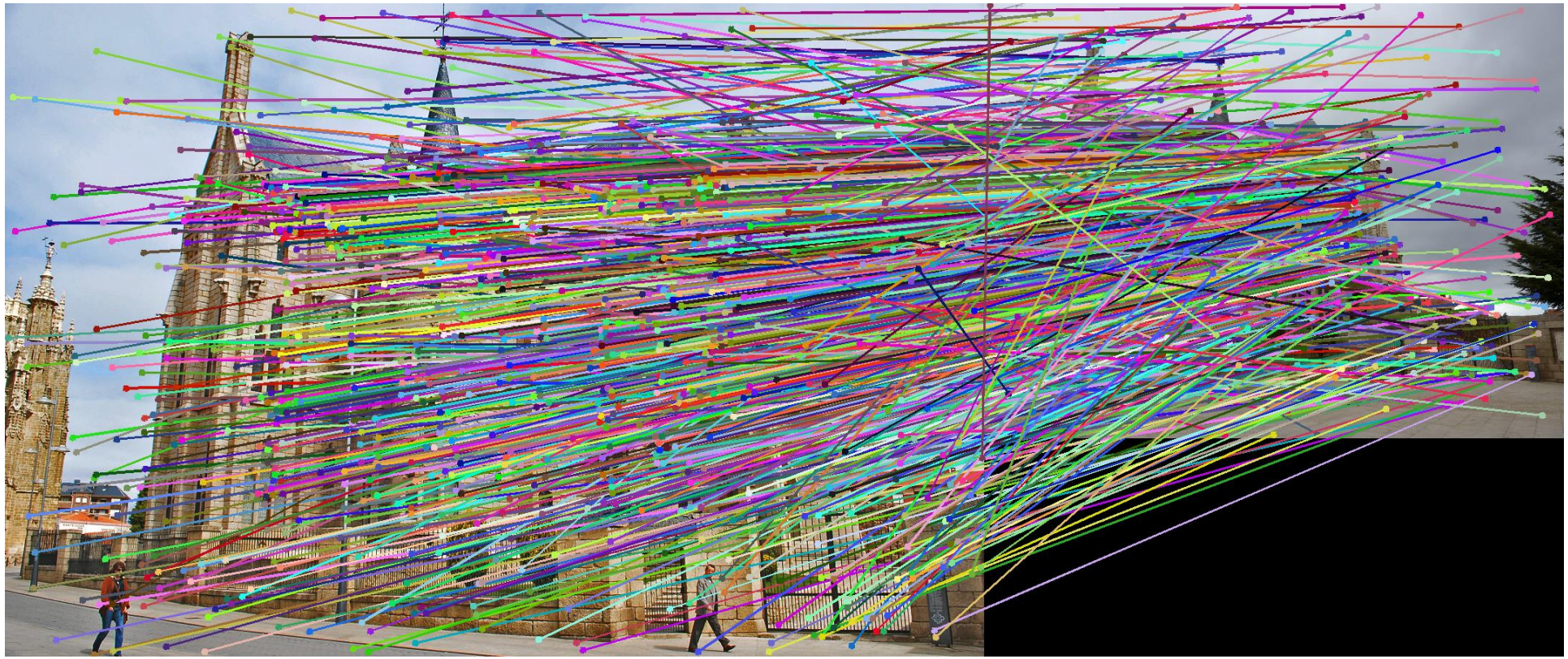


Potential matches for *x* have to lie on the corresponding line *l'*.

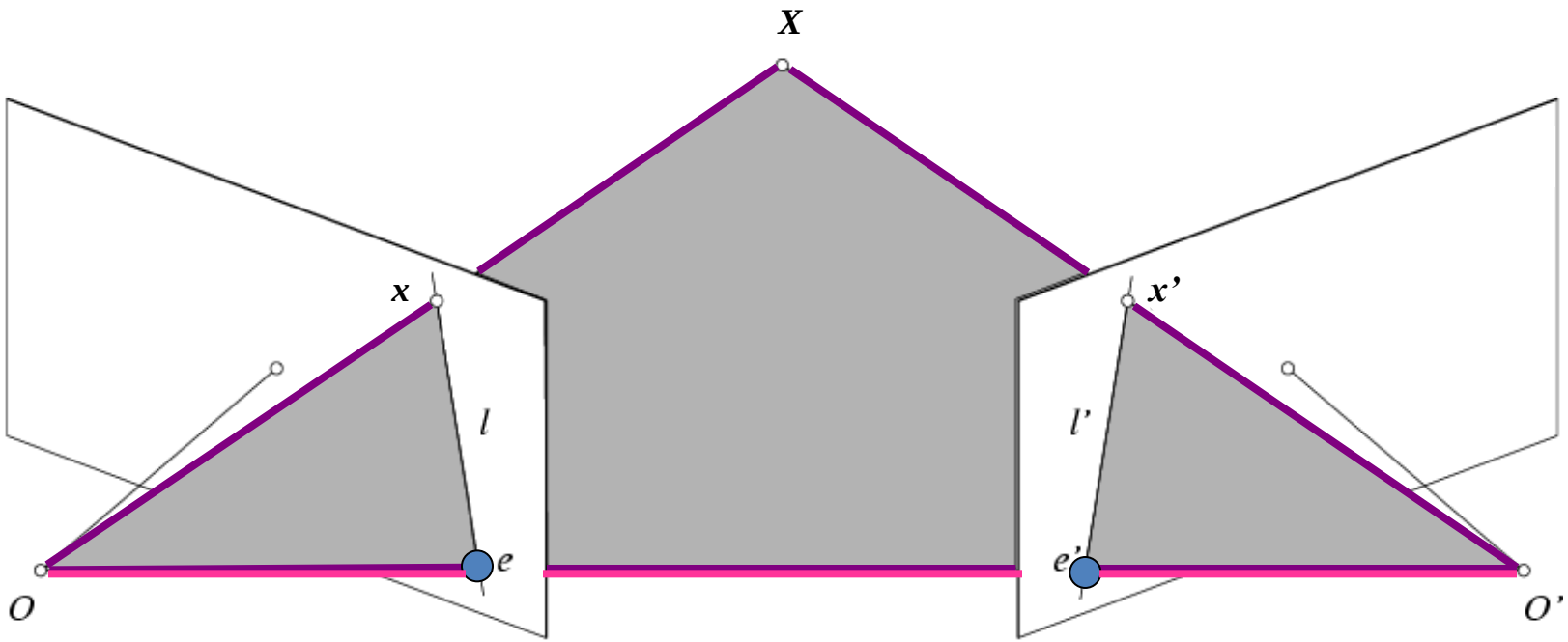
Potential matches for *x'* have to lie on the corresponding line *l*.

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



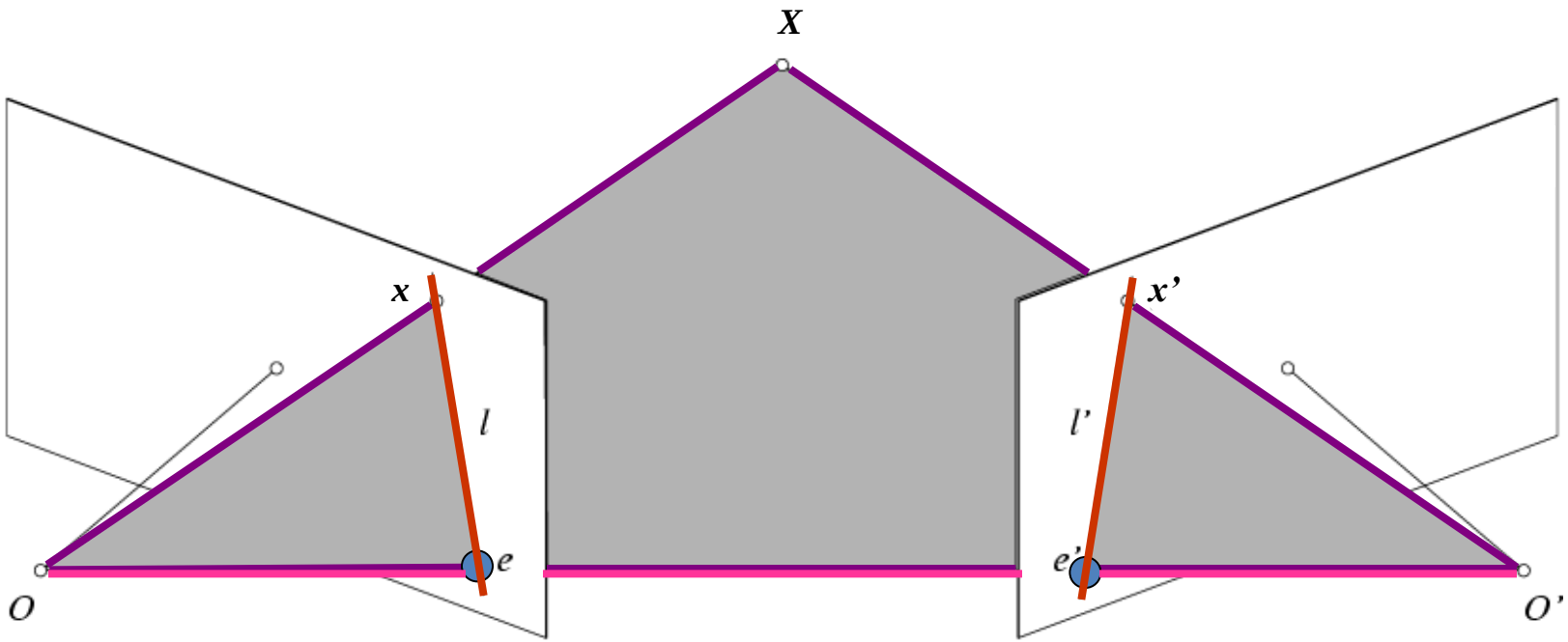
- **Baseline** – line connecting the two camera centers
- **Epipoles**

= intersections of baseline with image planes

= projections of the other camera center

- **Epipolar Plane** – plane containing baseline (1D family)

Epipolar geometry: notation



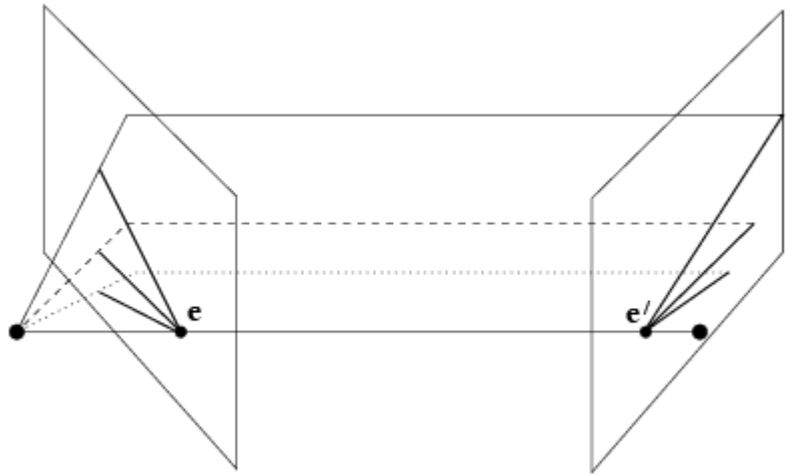
- **Baseline** – line connecting the two camera centers
- **Epipoles**

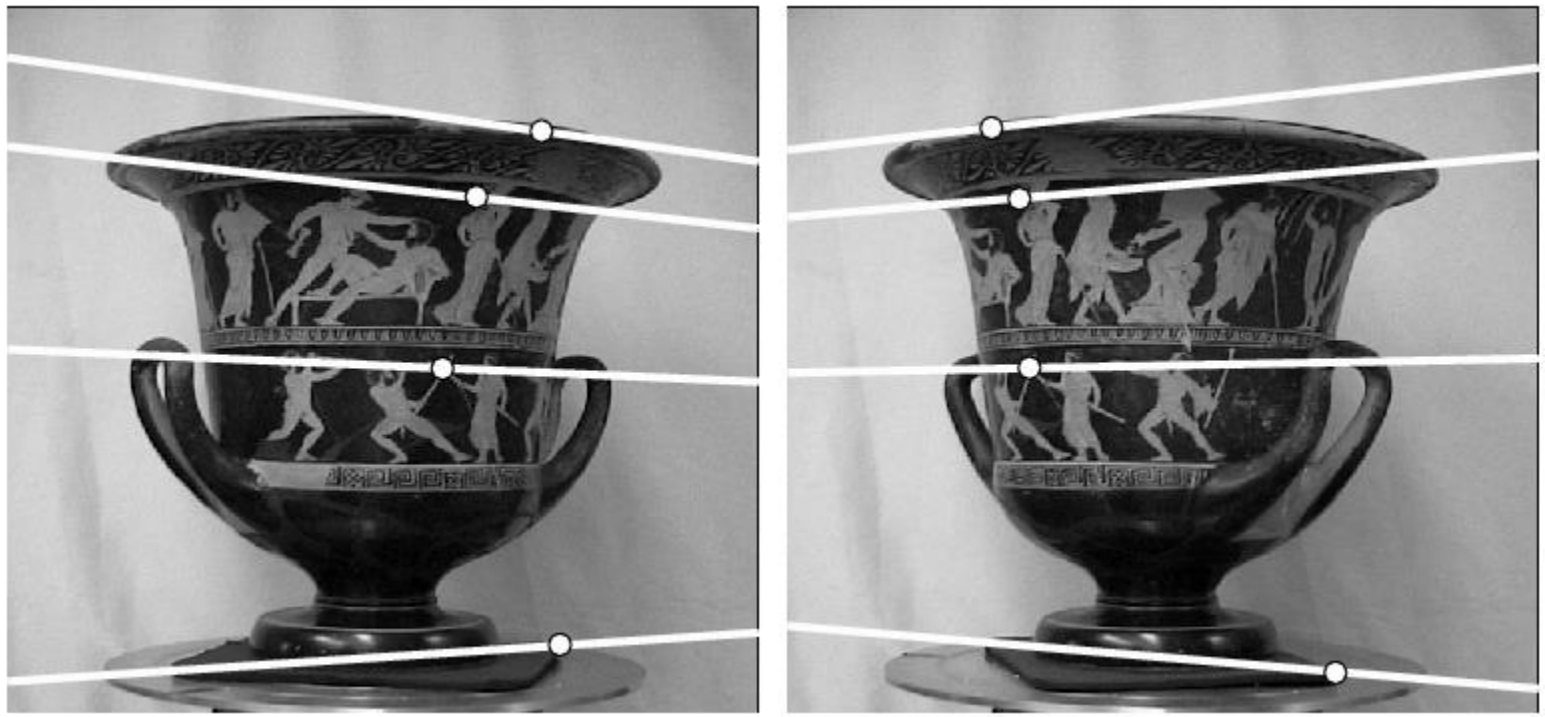
= intersections of baseline with image planes

= projections of the other camera center

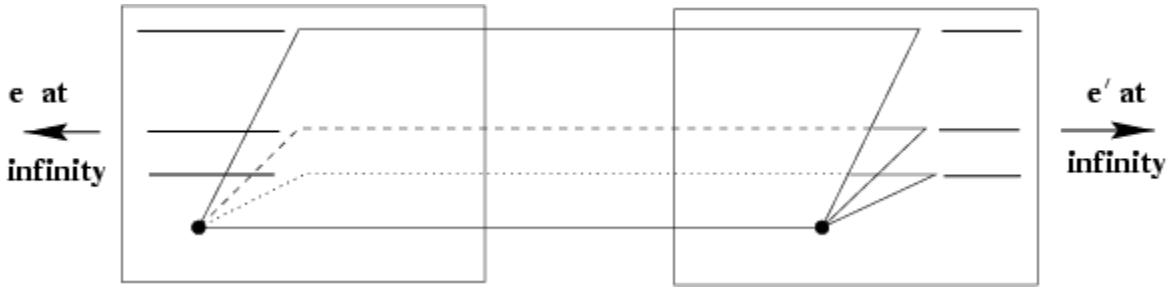
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

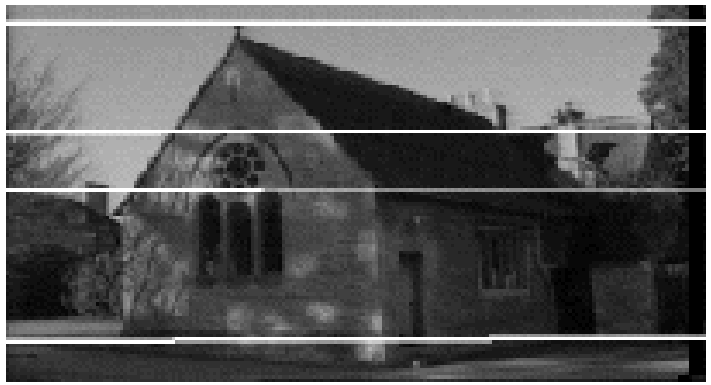
Example: Converging cameras





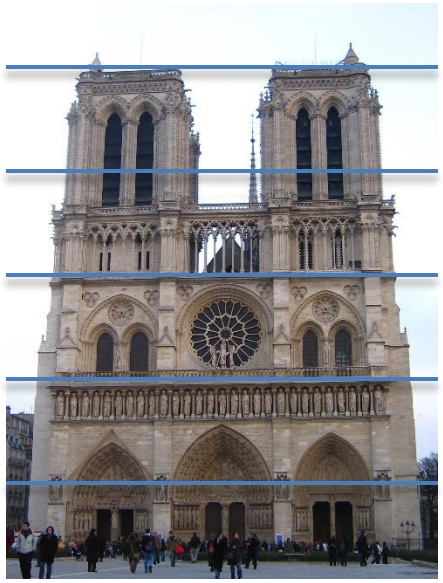
Example: Motion or displacement parallel to image plane



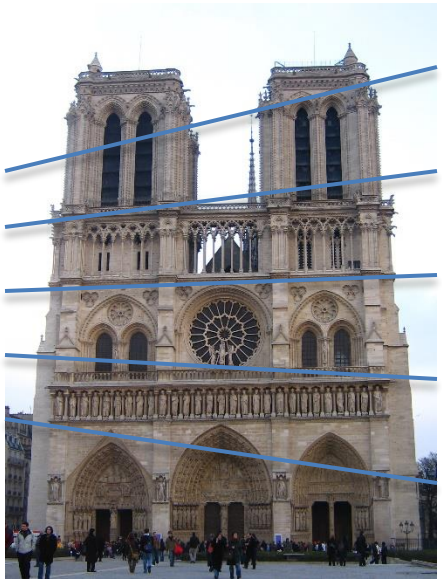


Example: Forward motion

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

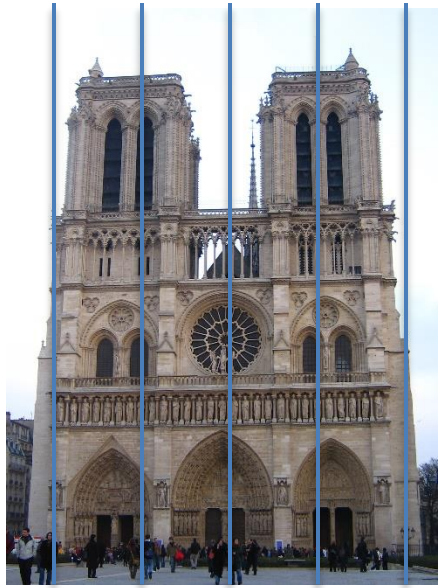


a



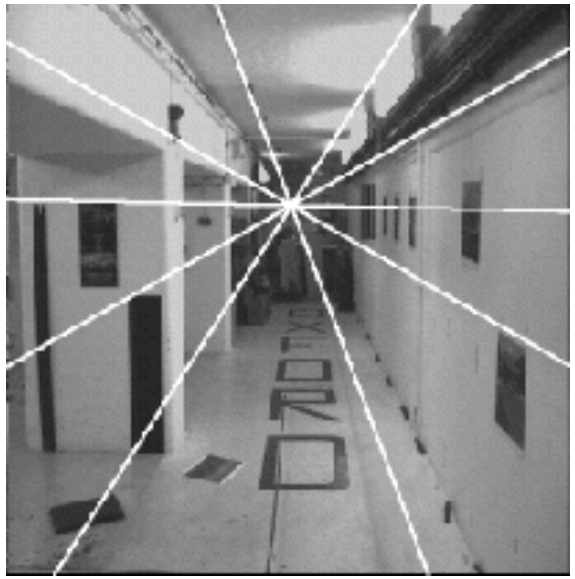
b

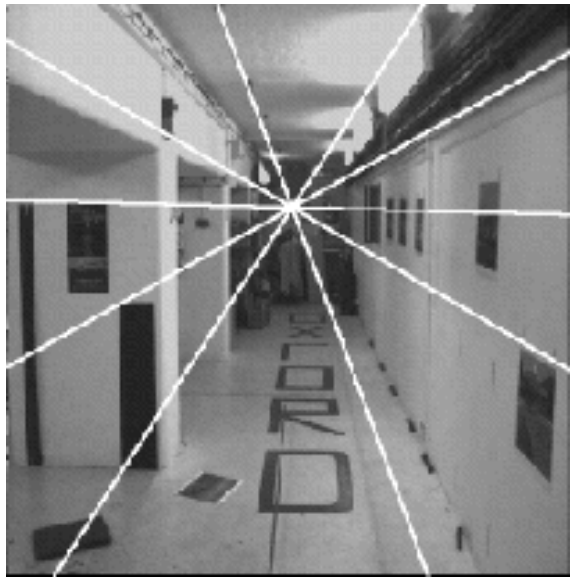
c

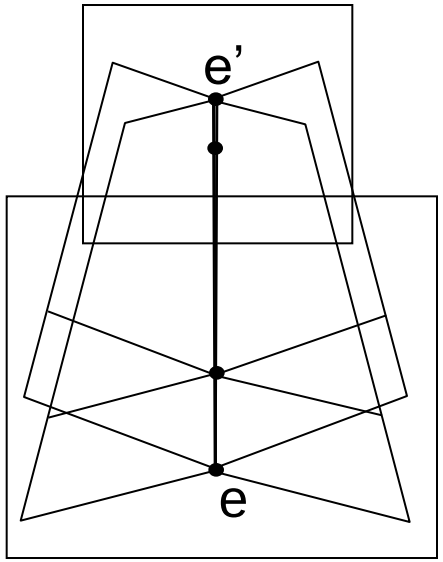


d

Example: Forward motion

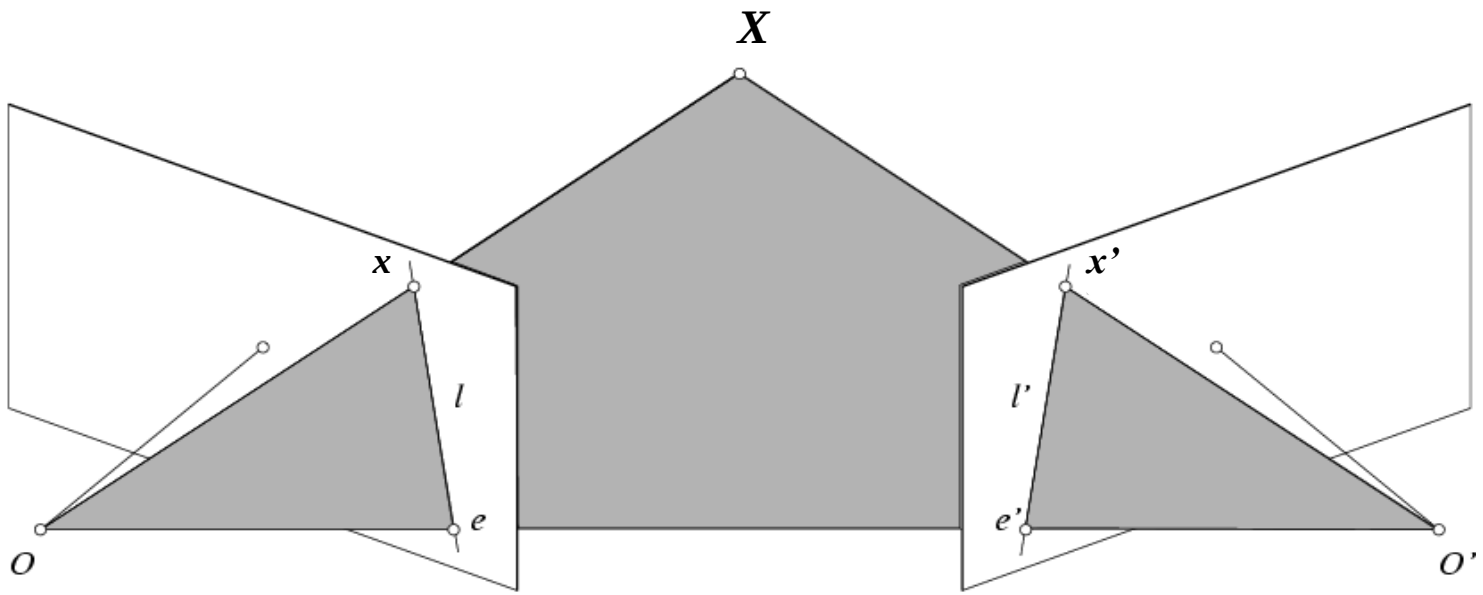






Epipole has same coordinates in both images.
Points move along lines radiating from e: “Focus of expansion”

Epipolar constraint: Calibrated case



Given the intrinsic parameters of the cameras:

1. Convert to normalized coordinates by pre-multiplying all points with the inverse of the calibration matrix; set first camera's coordinate system to world coordinates

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

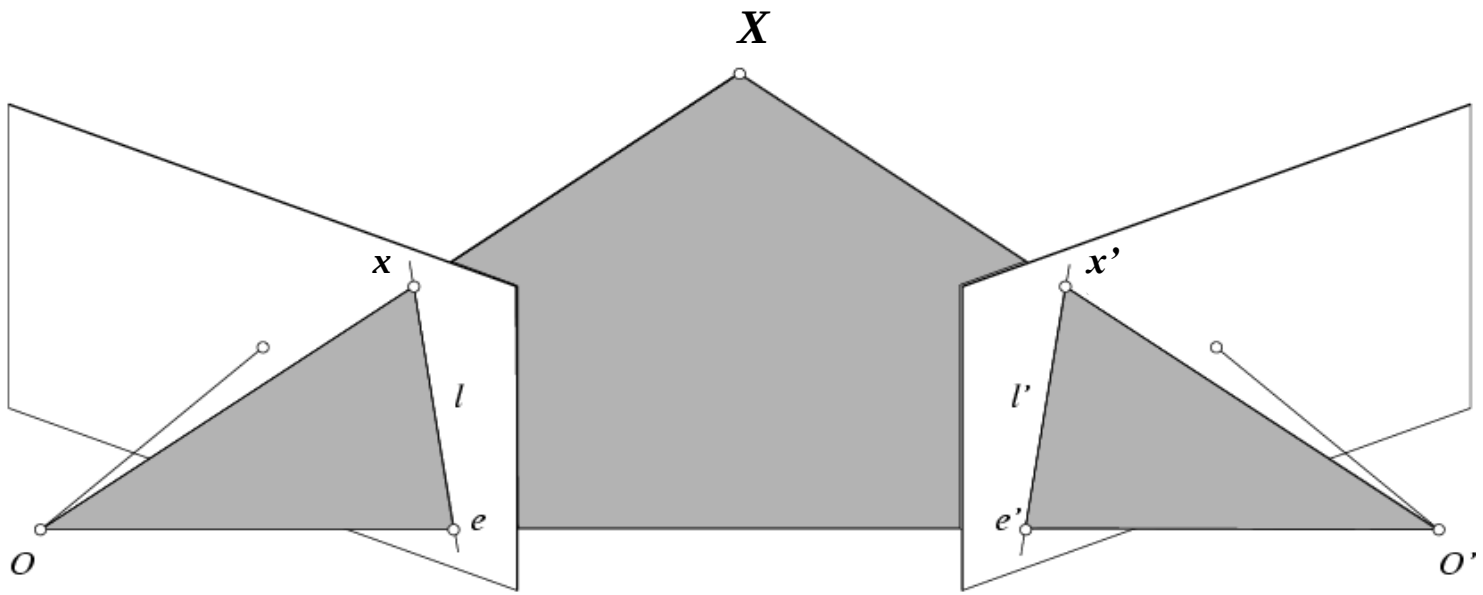
Homogeneous 2d point (3D ray towards X)

2D pixel coordinate (homogeneous)

3D scene point

3D scene point in 2nd camera's 3D coordinates

Epipolar constraint: Calibrated case



Given the intrinsic parameters of the cameras:

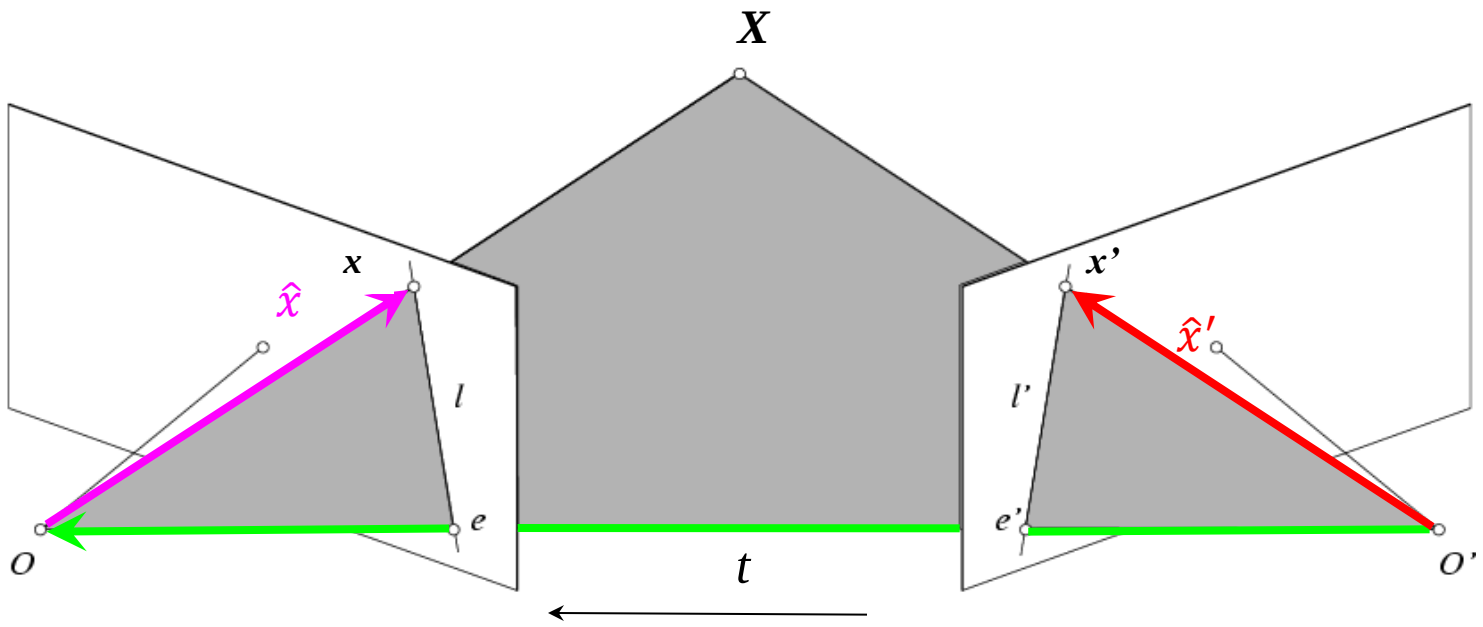
1. Convert to normalized coordinates by pre-multiplying all points with the inverse of the calibration matrix; set first camera's coordinate system to world coordinates
2. Define some *R* and *t* that relate X to X' as below

for some scale factor

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

$$\hat{x} = R\hat{x}' + t$$

Epipolar constraint: Calibrated case

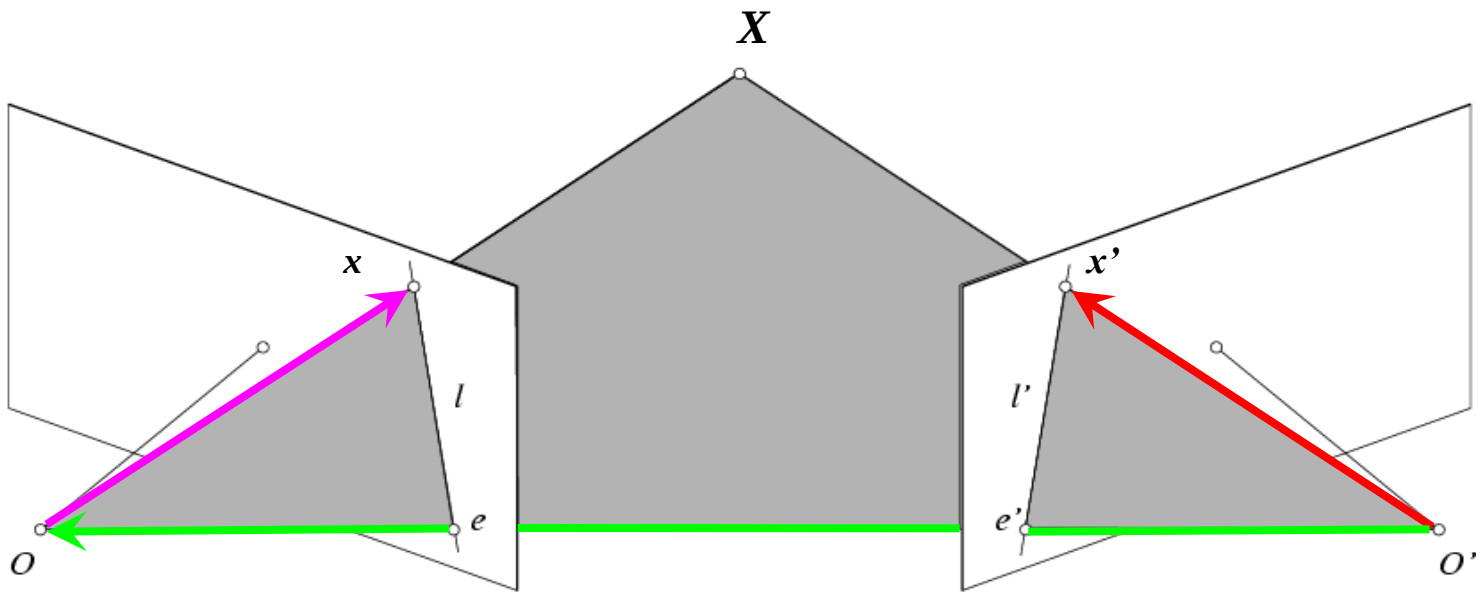


$$\hat{x} = K^{-1}x = X \qquad \hat{x}' = K'^{-1}x' = X'$$

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

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

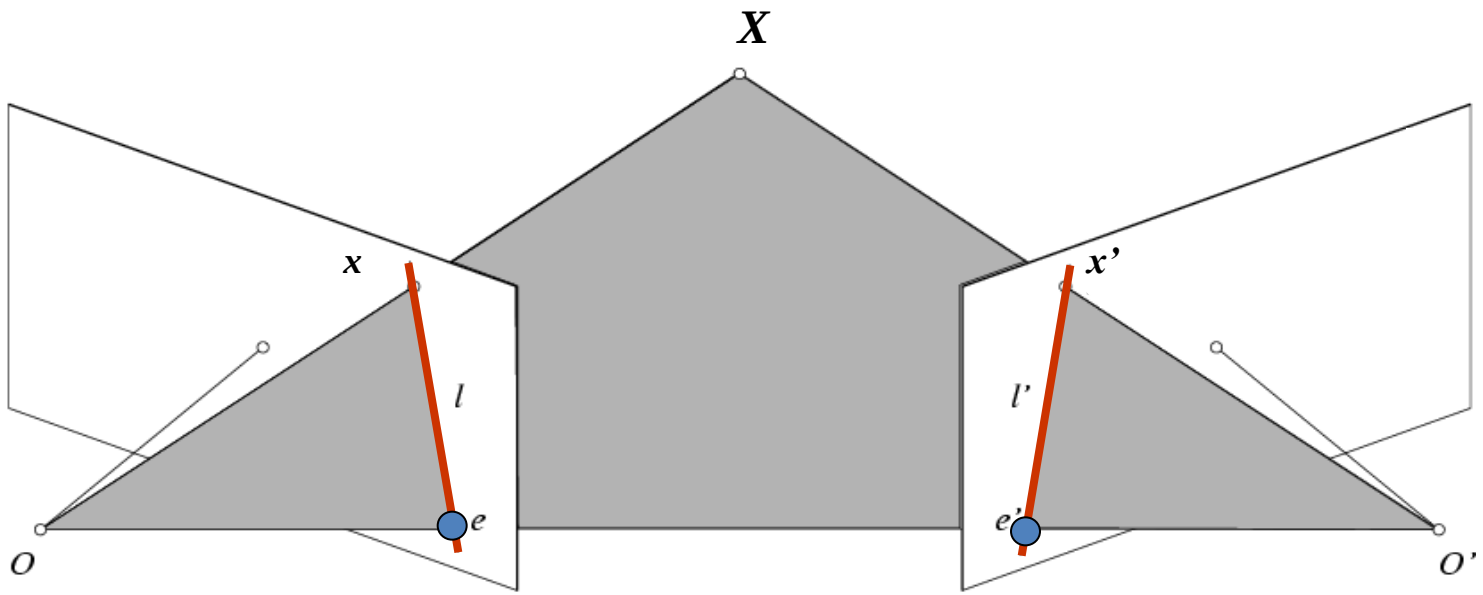
Essential matrix



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

Essential Matrix
(Longuet-Higgins, 1981)

Properties of the Essential matrix



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

Drop ^ below to simplify notation

Skew-symmetric matrix

- $E\,x'$ is the epipolar line associated with x' ($l = E\,x'$)
- $E^T x$ is the epipolar line associated with x ($l' = E^T x$)
- $E\,e' = 0$ and $E^T e = 0$
- E is singular (rank two)
- E has five degrees of freedom
  - (3 for R, 2 for t because it's up to a scale)

The Fundamental Matrix

Without knowing K and K', we can define a similar relation using *unknown* normalized coordinates

$$\hat{x}^T E \hat{x}' = 0$$

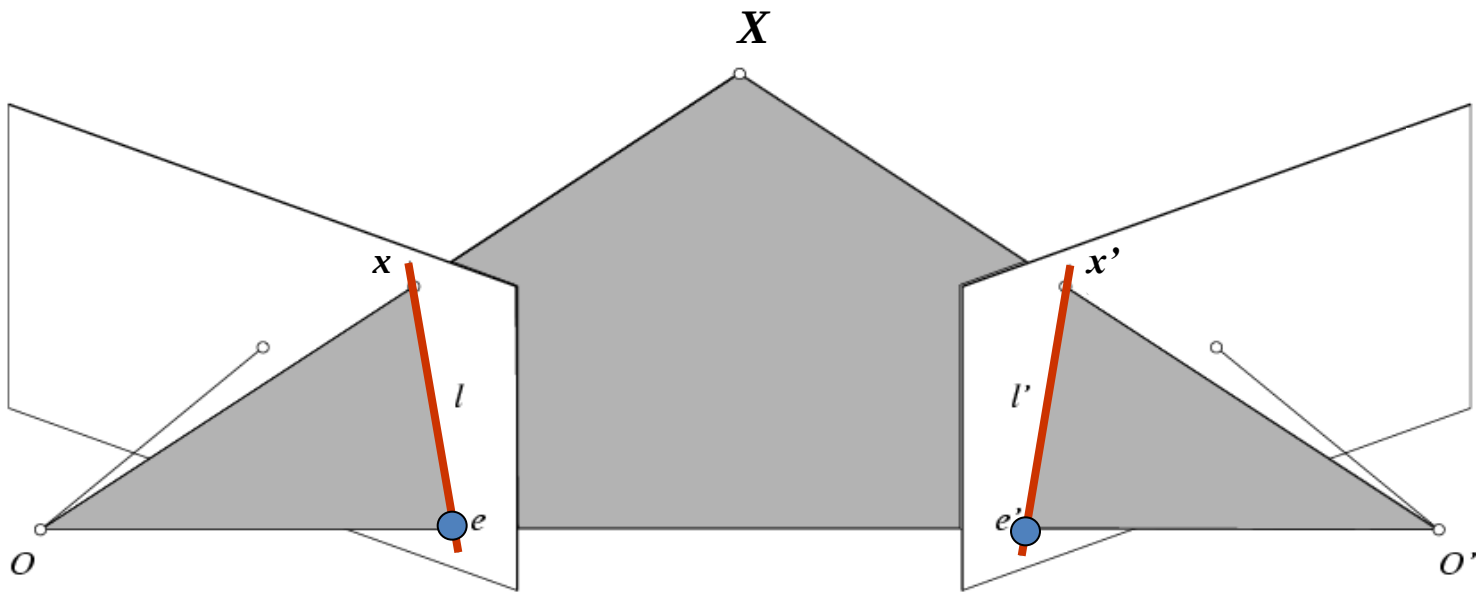
$$\hat{x} = K^{-1} x$$

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

$$\Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Properties of the Fundamental matrix



$$x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

- $F\,x' = 0$ is the epipolar line associated with x'
- $F^T x = 0$ is the epipolar line associated with x
- $F\,e' = 0$ and $F^T e = 0$
- F is singular (rank two): det(F)=0
- F has seven degrees of freedom: 9 entries but defined up to scale, det(F)=0

Estimating the Fundamental Matrix

- 8-point algorithm
  - Least squares solution using SVD on equations from 8 pairs of correspondences
  - Enforce det(F)=0 constraint using SVD on F

- 7-point algorithm
  - Use least squares to solve for null space (two vectors) using SVD and 7 pairs of correspondences
  - Solve for linear combination of null space vectors that satisfies det(F)=0

- Minimize reprojection error
  - Non-linear least squares

Note: estimation of F (or E) is degenerate for a planar scene.

8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations

$$\mathbf{x}^T F \mathbf{x}' = 0$$

$$uu'f_{11} + uv'f_{12} + uf_{13} + vu'f_{21} + vv'f_{22} + vf_{23} + u'f_{31} + v'f_{32} + f_{33} = 0$$

$$\mathrm{A}\boldsymbol{f} = \begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_v' & u_nv_n' & u_n & v_nu_n' & v_nv_n' & v_n & u_n' & v_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{0}$$

8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations
   b. Solve **f** from A**f**=**0** using SVD

Matlab:

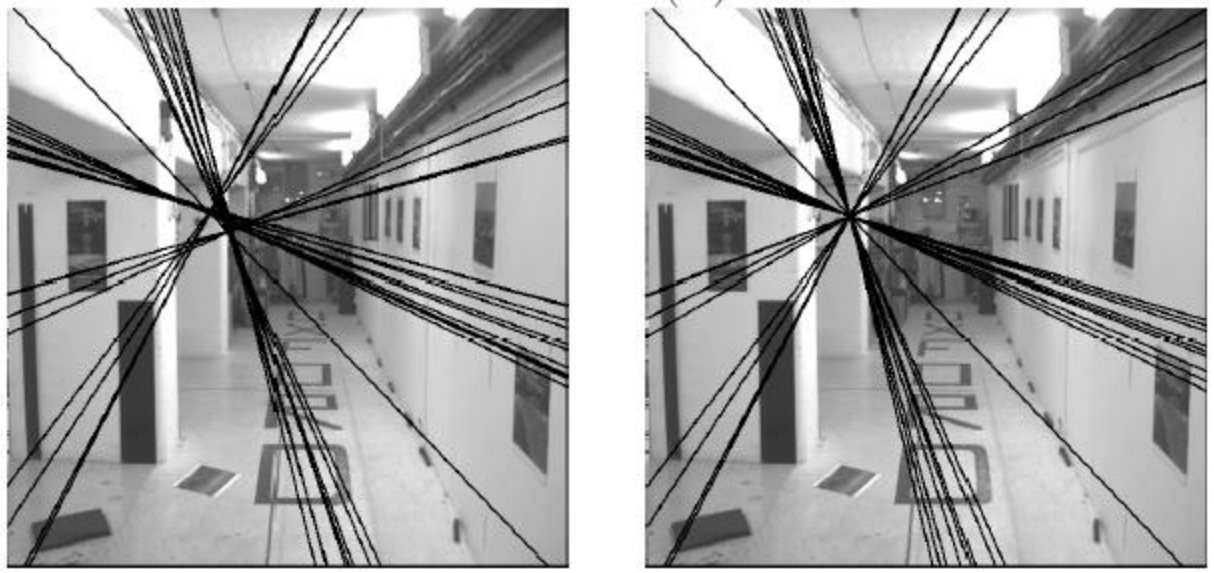
```
[U, S, V] = svd(A);
f = V(:, end);
F = reshape(f, [3 3])';
```

For python, see

numpy.linalg.svd

Need to enforce singularity constraint

Fundamental matrix has rank 2 : $\det(\mathbf{F}) = 0$.



Left : Uncorrected $\mathtt{F}$ – epipolar lines are not coincident.

Right : Epipolar lines from corrected $\mathtt{F}$.

8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations
   b. Solve **f** from A**f**=**0** using SVD

Matlab:

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

2. Resolve det(F) = 0 constraint using SVD

Matlab:

```
[U, S, V] = svd(F);
S(3,3) = 0;
F = U*S*V';
```

For python, see

numpy.linalg.svd

5 Part 5: Fundamental matrix

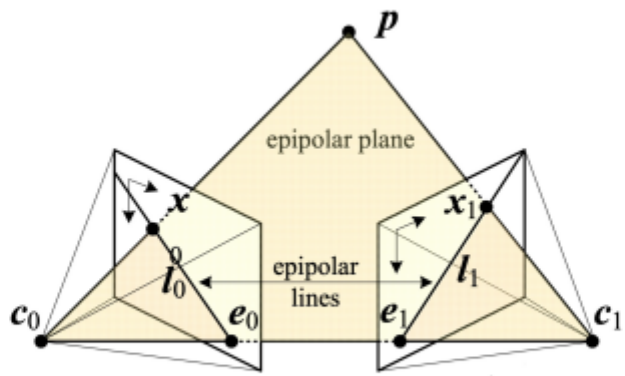


Figure 2: Two-camera setup. Reference: Szeliski, p. 682.

The next part of this project is estimating the mapping of points in one image to lines in another by means of the fundamental matrix. This will require you to use similar methods to those in part 4. We will make use of the corresponding point locations listed in `pts2d-pic_a.txt` and `pts2d-pic_b.txt`. Recall that the definition of the fundamental matrix is:

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0 \tag{9}$$

for a point $(u, v, 1)$ in image A, and a point $(u', v', 1)$ in image B. See Appendix A for the full derivation. Note: the fundamental matrix is sometimes defined as the transpose of the above matrix with the left and right image points swapped. Both are valid fundamental matrices, but the visualization functions in the starter code assume you use the above form.

Another way of writing this matrix equations is:

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} f_{11}u + f_{12}v + f_{13} \\ f_{21}u + f_{22}v + f_{23} \\ f_{31}u + f_{32}v + f_{33} \end{pmatrix} = 0 \tag{10}$$

Which is the same as:

$$\left(f_{11}uu' + f_{12}vu' + f_{13}u' + f_{21}uv' + f_{22}vv' + f_{23}v' + f_{31}u + f_{32}v + f_{33}\right) = 0 \tag{11}$$

Problem with eight-point algorithm

$$\begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \end{bmatrix} = -1$$

Problem with eight-point algorithm

250906.36	183269.57	921.81	200931.10	146766.13	738.21	272.19	198.81
2692.28	131633.03	176.27	6196.73	302975.59	405.71	15.27	746.79
416374.23	871684.30	935.47	408110.89	854384.92	916.90	445.10	931.81
191183.60	171759.40	410.27	416435.62	374125.90	893.65	465.99	418.65
48988.86	30401.76	57.89	298604.57	185309.58	352.87	846.22	525.15
164786.04	546559.67	813.17	1998.37	6628.15	9.86	202.65	672.14
116407.01	2727.75	138.89	169941.27	3982.21	202.77	838.12	19.64
135384.58	75411.13	198.72	411350.03	229127.78	603.79	681.28	379.48

$$\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \end{bmatrix} = -1$$

Poor numerical conditioning

Can be fixed by rescaling the data

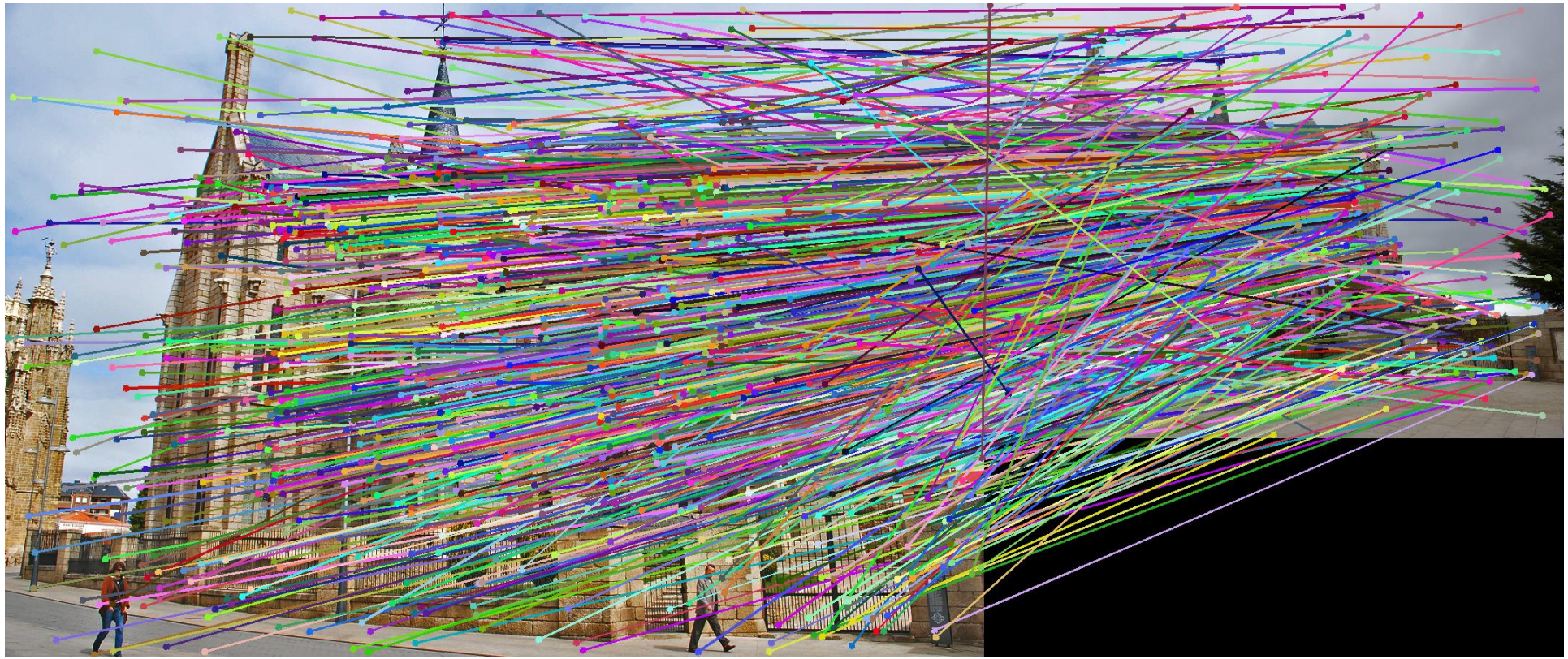
The normalized eight-point algorithm

(Hartley, 1995)

- Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels
- Use the eight-point algorithm to compute ***F*** from the normalized points
- Enforce the rank-2 constraint (for example, take SVD of ***F*** and throw out the smallest singular value)
- Transform fundamental matrix back to original units: if ***T*** and ***T'*** are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is $\boldsymbol{T'^{T} F T}$

But which 8 points do we choose?

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



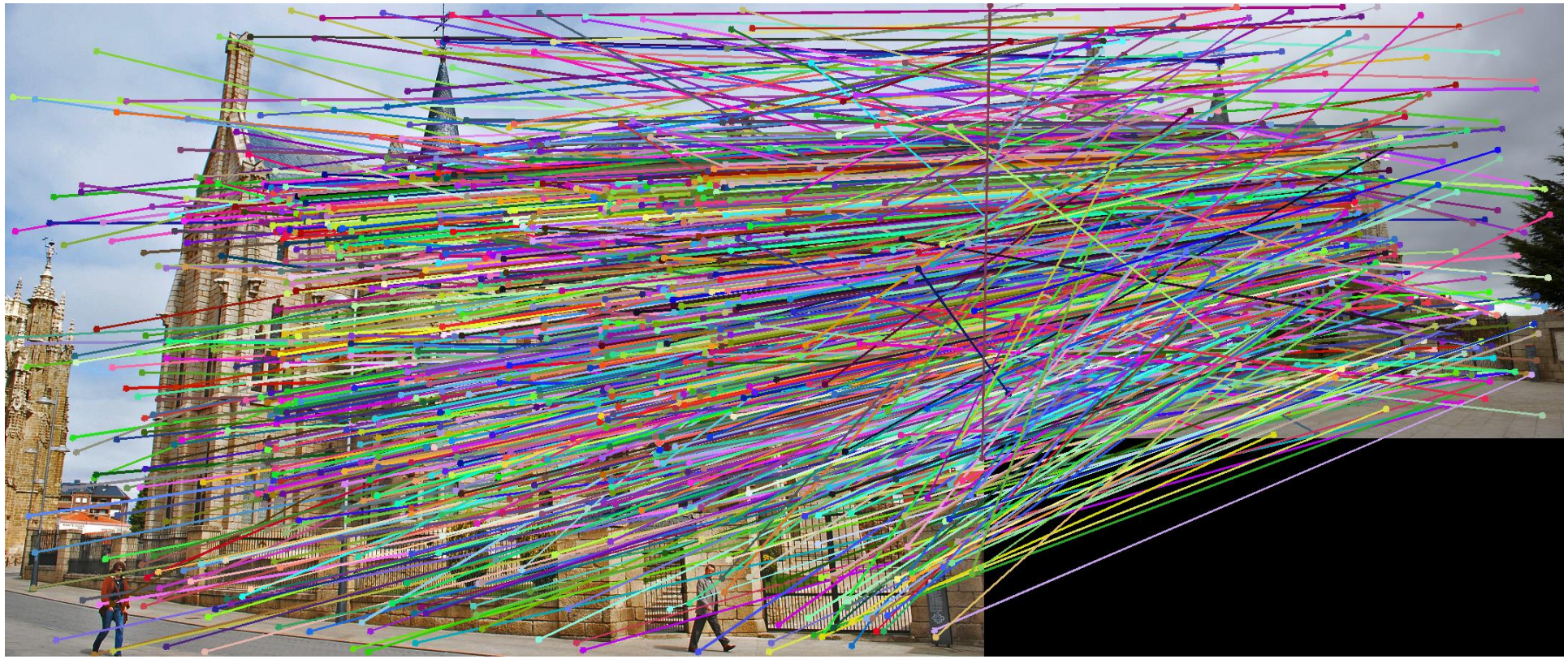
6 Part 6: Fundamental matrix with RANSAC

For two photographs of a scene it's unlikely that you'd have perfect point correspondence with which to do the regression for the fundamental matrix. So, next you are going to compute the fundamental matrix with point correspondences computed using SIFT. As discussed in class, least squares regression alone is not appropriate in this scenario due to the presence of multiple outliers. In order to estimate the fundamental matrix from this noisy data you'll need to use RANSAC in conjunction with your fundamental matrix estimation.

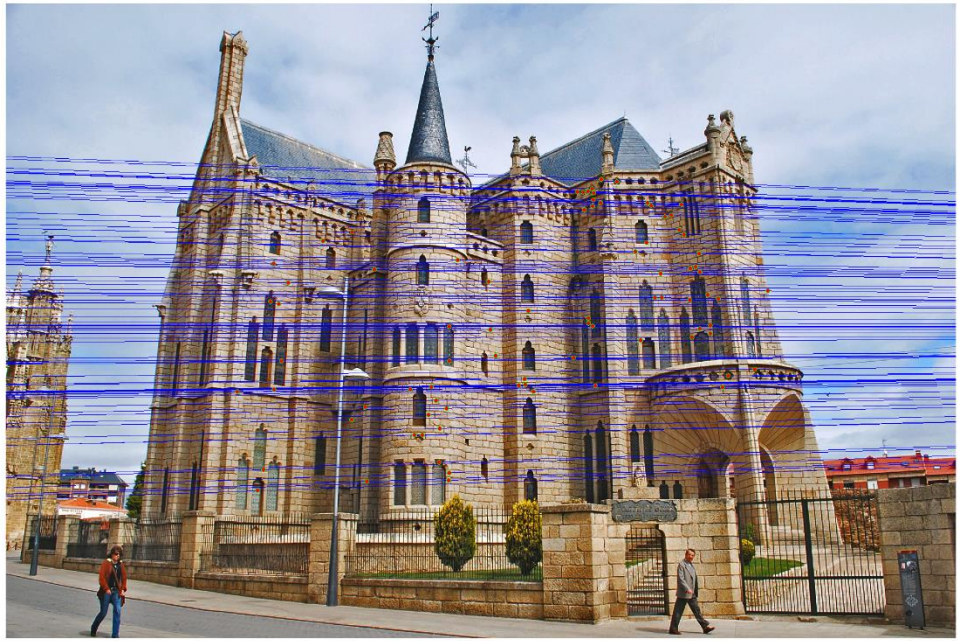
You'll use these putative point correspondences and RANSAC to find the "best" fundamental matrix. You will iteratively choose some number of point correspondences (8, 9, or some small number), solve for the fundamental matrix using the function you wrote for part 5, and then count the number of inliers. Inliers in this context will be point correspondences that "agree" with the estimated fundamental matrix. In order to count how many inliers a fundamental matrix has, you'll need a distance metric based on the fundamental matrix. (Hint: For a point correspondence (x, x') what properties does the fundamental matrix have?). You'll need to pick a threshold between inliers and outliers and your results are very sensitive to this threshold, so explore a range of values. You don't want to be too permissive about what you consider an inlier, nor do you want to be too conservative. Return the fundamental matrix with the most inliers.

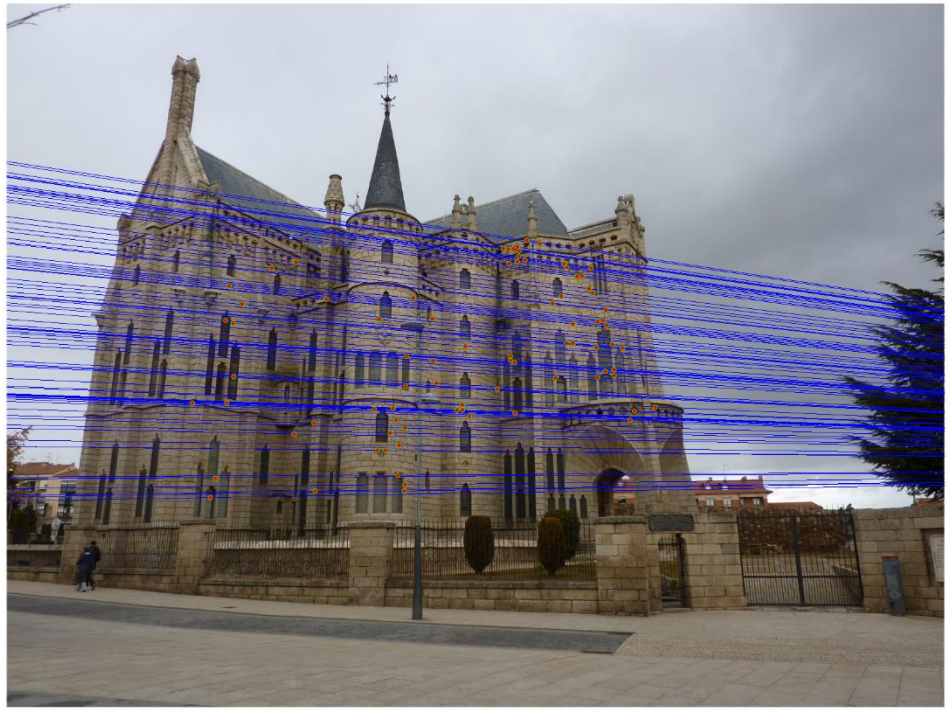
Recall from lecture the expected number of iterations of RANSAC to find the "right" solution in the presence of outliers. For example, if half of your input correspondences are wrong, then you have a $(0.5)^8 = 0.39\%$

How to test for outliers?

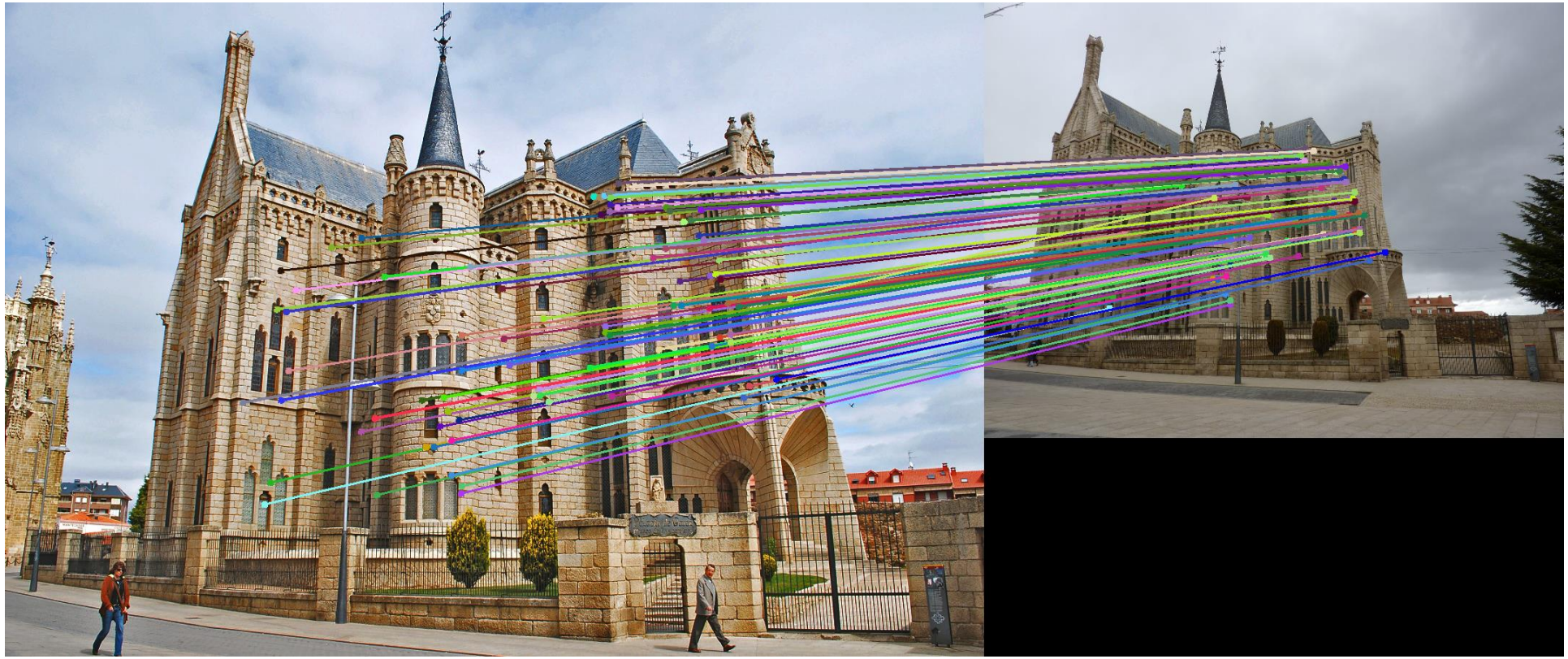


Epipolar lines

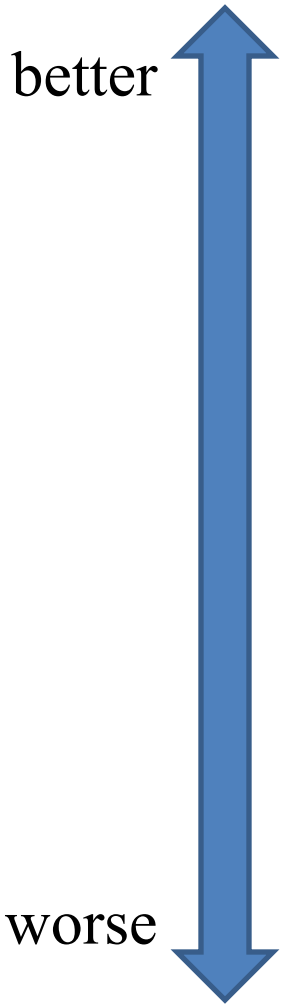




Keep only the matches at are “inliers” with respect to the “best” fundamental matrix



The scale of algorithm name quality



RANSAC

SIFT

Deep Learning

Optical Flow

Hough Transform

Neural Networks

Essential and Fundamental Matrix

Dynamic Programming

In class written Quiz format

- 15 to 20 short answer or multiple-choice questions
- Typically can be done in half an hour
- No calculators needed
- Closed book
- Only covers material discussed in class, not book. But the book is still a useful resource
- Covers all material through the quiz date