CSE 152: Computer Vision

Hao Su

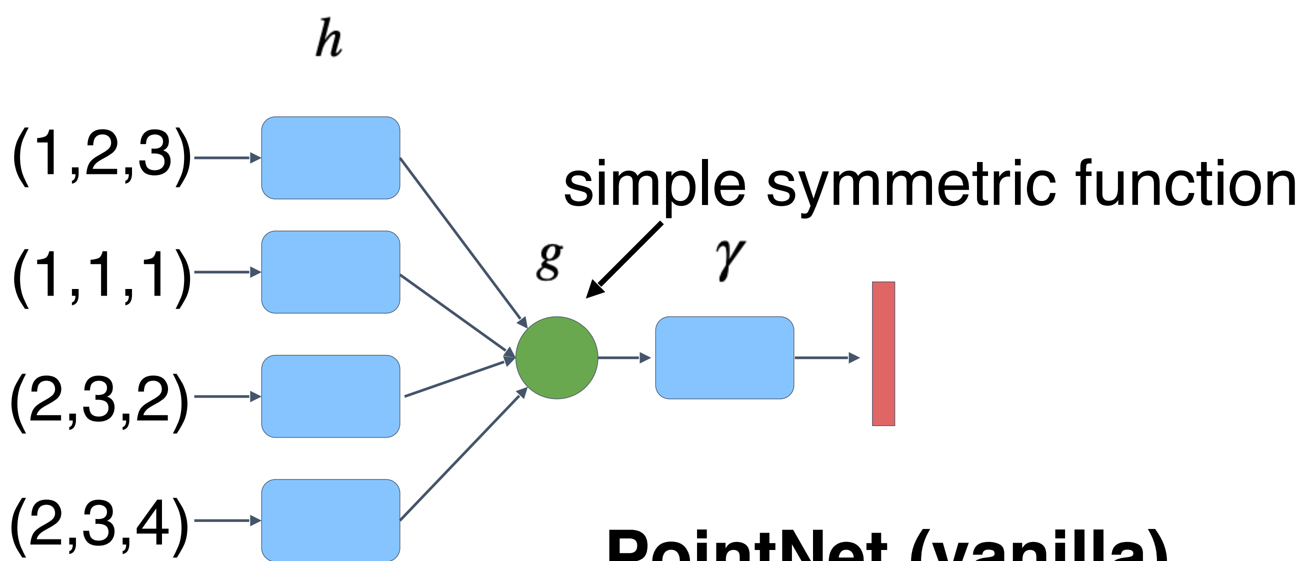
Lecture 14: Multiview Geometry

Some hints to HW3

Construct a Symmetric Function

Observe:

$f(x_1,x_2,\ldots,x_n)=\gamma \circ g(h(x_1),\ldots,h(x_n))$ is symmetric if g is symmetric

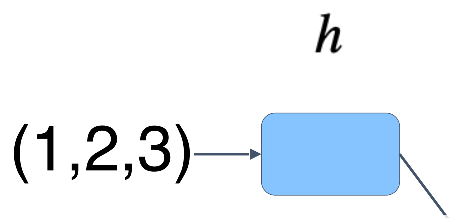


PointNet (vanilla)

Construct a Symmetric Function

Observe:

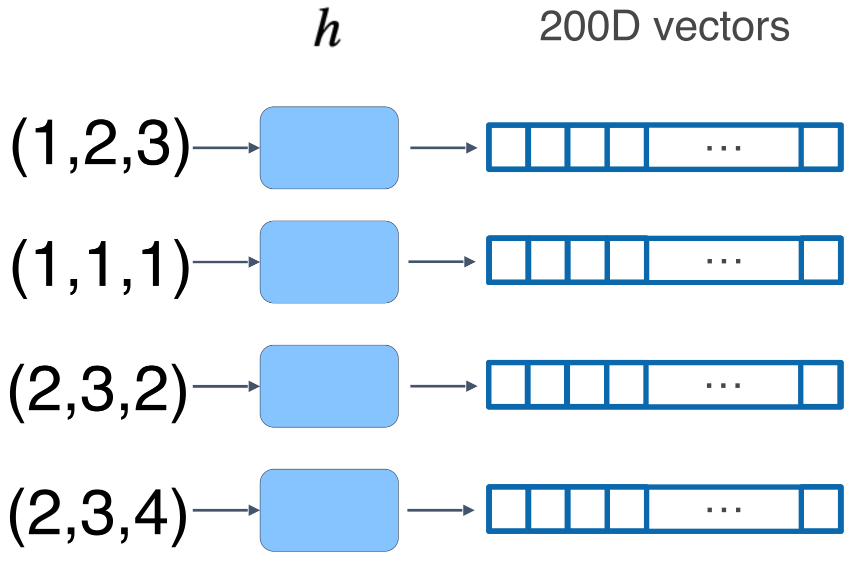
$f(x_1,x_2,\ldots,x_n)=\gamma \circ g(h(x_1),\ldots,h(x_n))$ is symmetric if g is symmetric

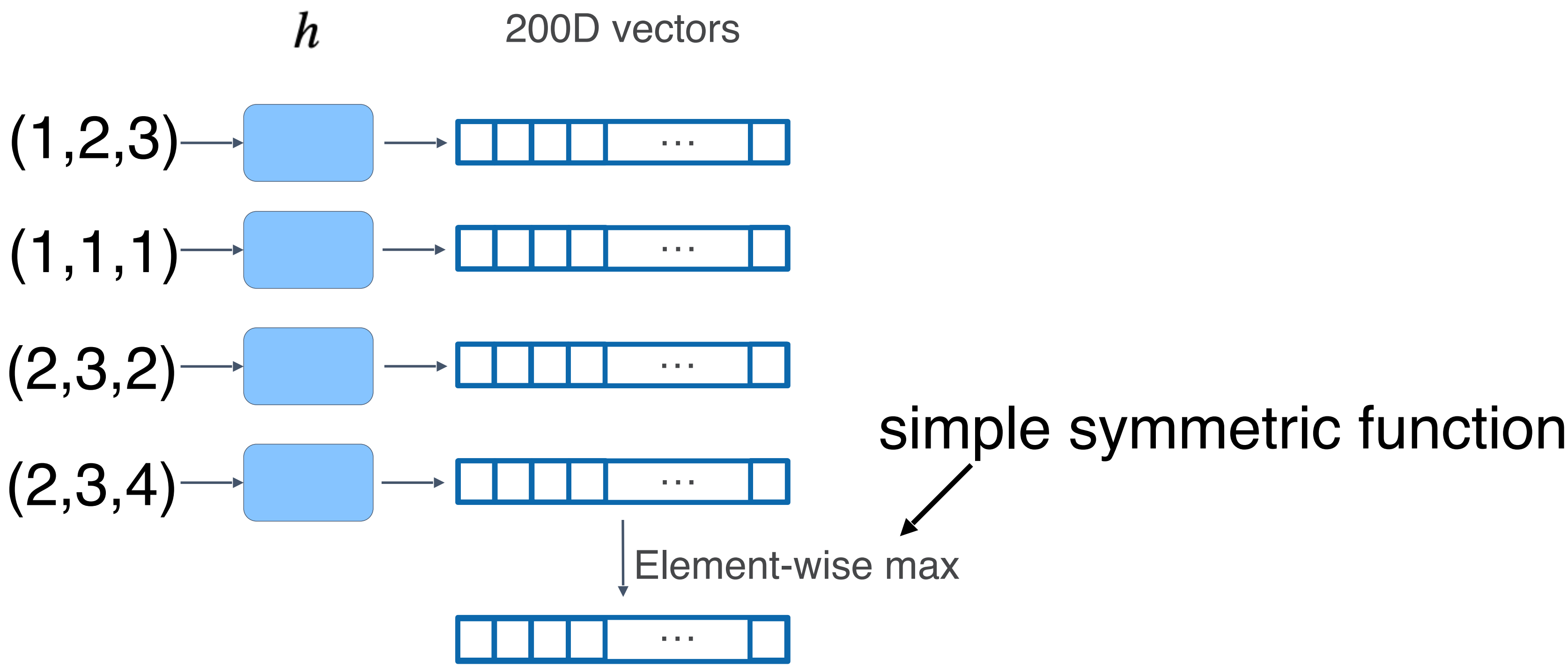


simple symmetric function

h:

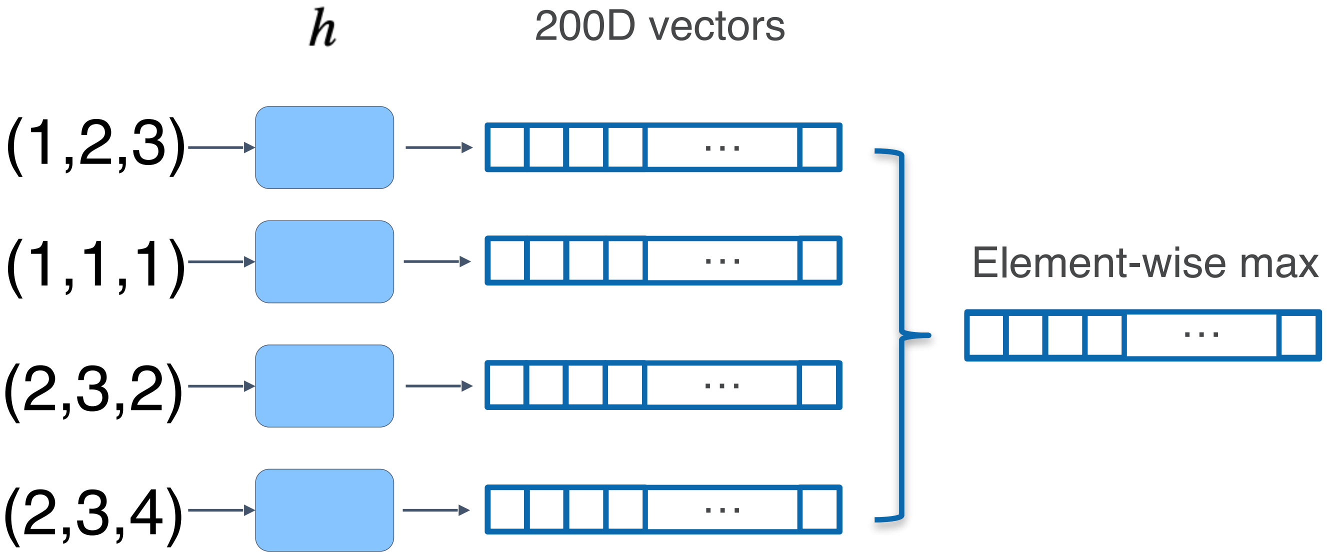
- a fully-connected network (a.k.a. MLP)
- To transform each point from 3D to high dim, e.g. 200 dim
- For example, you use 64, 64, 128, 200 dimensions for each layer of the MLP.


h
200D vectors
(1,2,3)
(1,1,1)
(2,3,2)
(2,3,4)
···
···
···
···

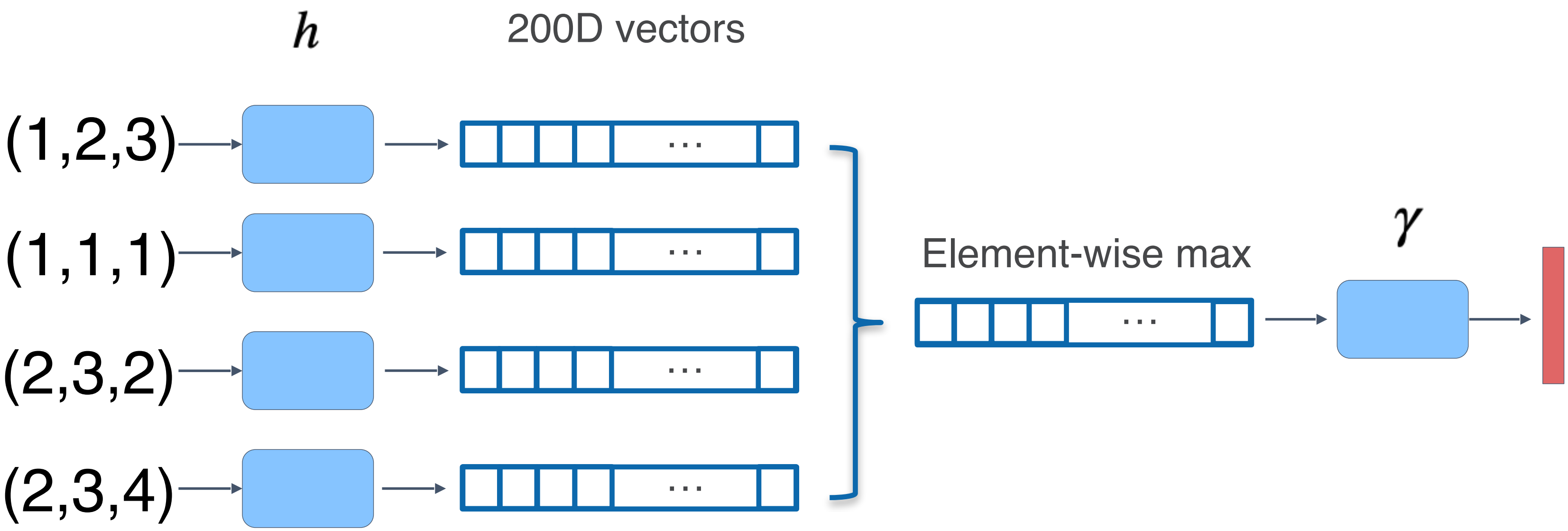


Suppose you have 10,000 points in total

- build a matrix of 10000×200 dims, where each row is the feature vector coming from a point.
- you do column-wise max operation, which will produce a single vector of 200 dims.



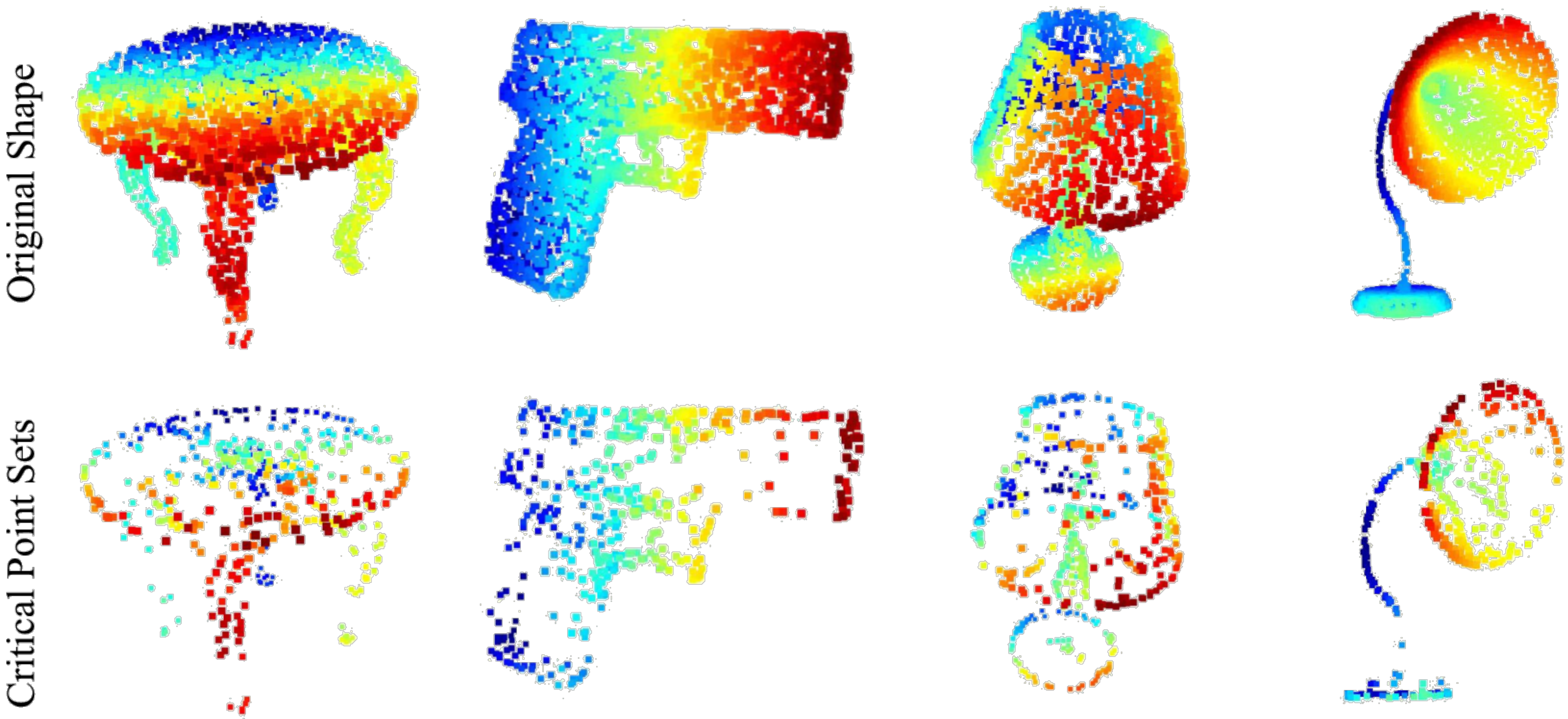
h
200D vectors
(1,2,3)
(1,1,1)
(2,3,2)
(2,3,4)
Element-wise max

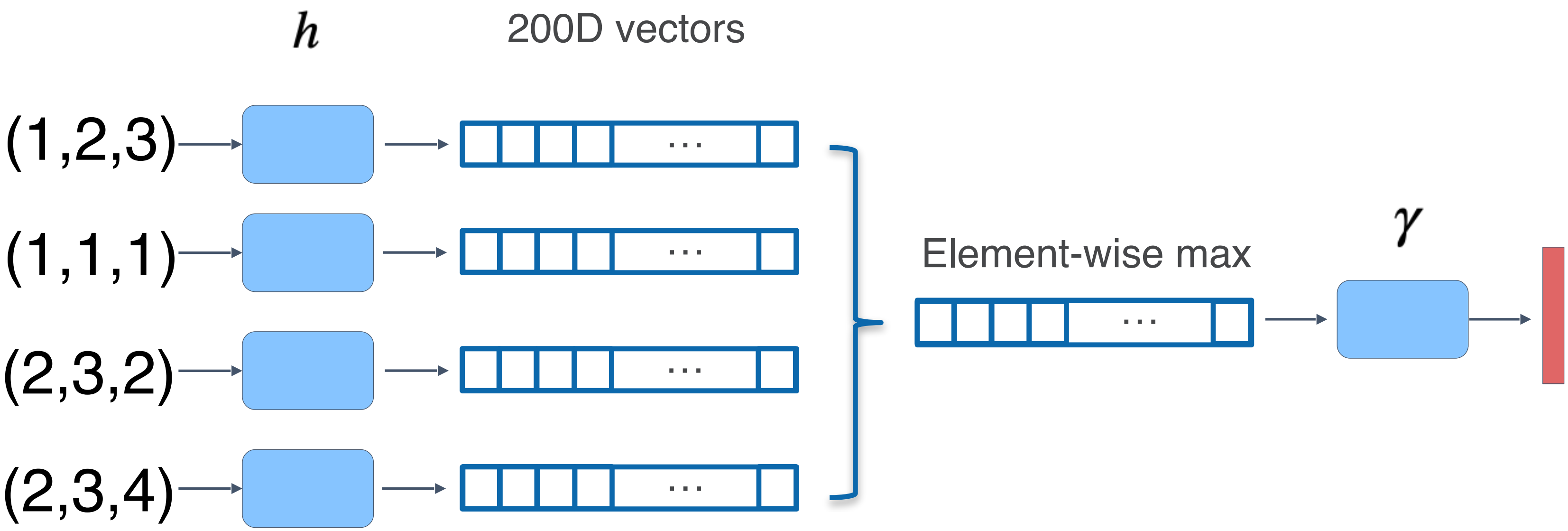


Build another MLP that takes the aggregated vector as input and predict the category label of the object

- the input is 200 dim, and you may use 128 dim as the intermediate layer size (dim),
- finally predict the confidence for the k categories of objects by softmax.

Critical Points: Points with Non-Zero Gradient w.r.t. Positions

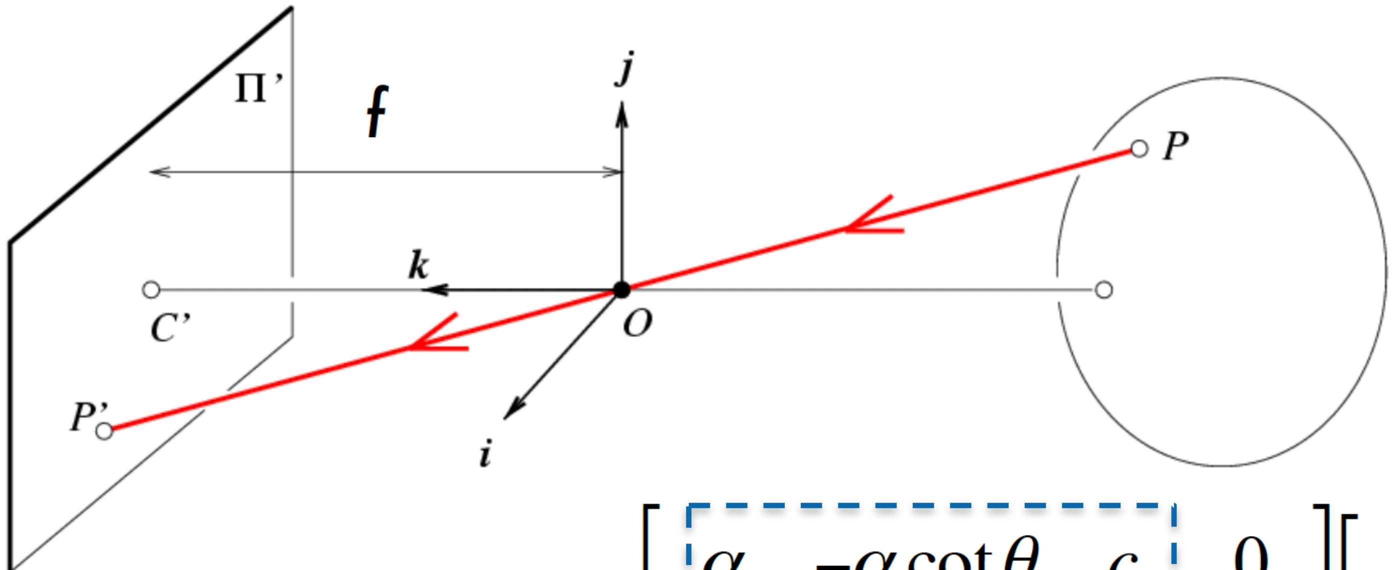


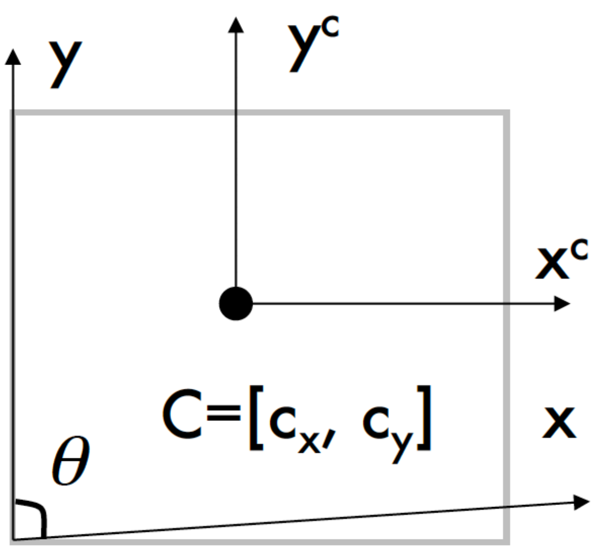


1. Train the PointNet classifier
2. Pick a test instance and run forward pass to make prediction. You wonder what points affect the prediction.
3. Compute gradient over each dimension of each input point using back-prop (check P2 for adversarial attack on how to compute gradient)
4. Filter the points with high gradients. Their movement affects the prediction most!

Review of Last Lecture

Intrinsic Camera Matrix

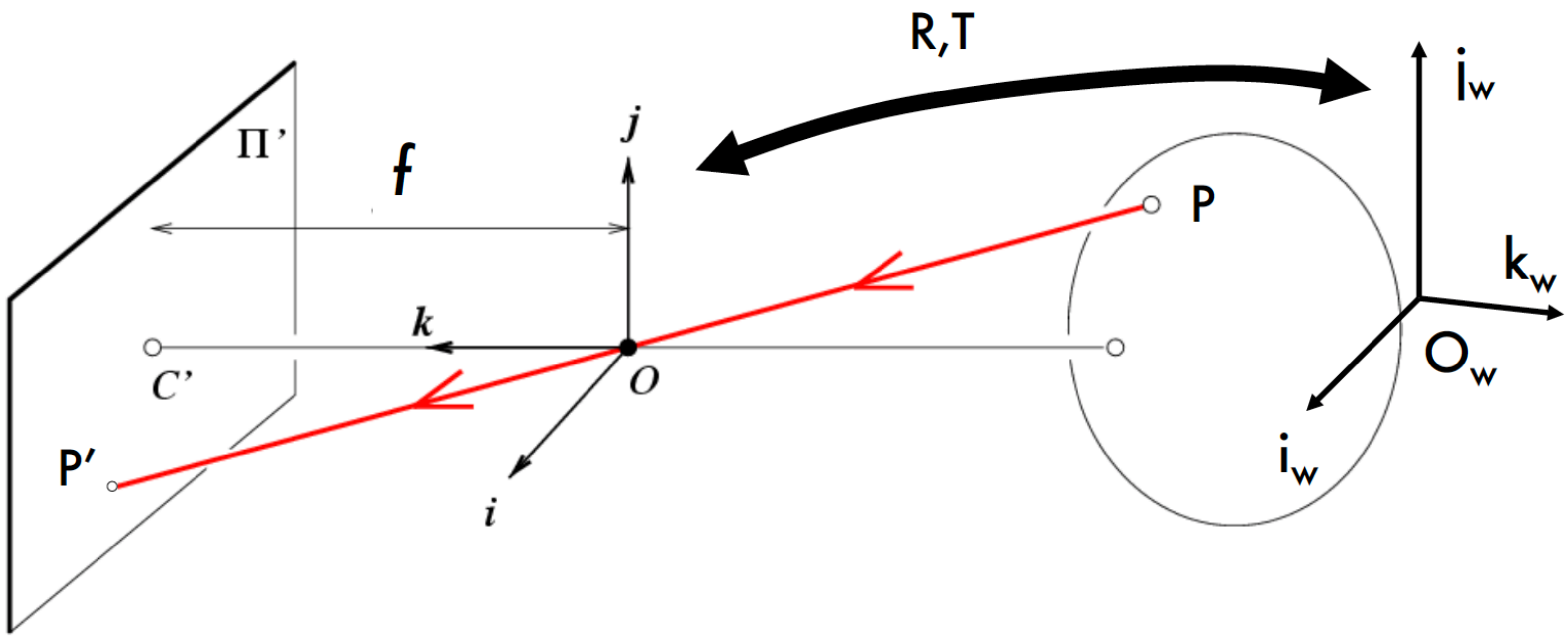




$$P' = \begin{bmatrix} \alpha & -\alpha\cot\theta & c_x & 0 \\ 0 & \dfrac{\beta}{\sin\theta} & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

How many degree does K have?
5 degrees of freedom!

The Projective Transformation



$$P'_{3\times1} = M_{3x4}\,P_w = K_{3\times3}\begin{bmatrix} R & T \end{bmatrix}_{3\times4} P_{w\,4\times1}$$

How many degrees of freedom?

5 + 3 + 3 =11!

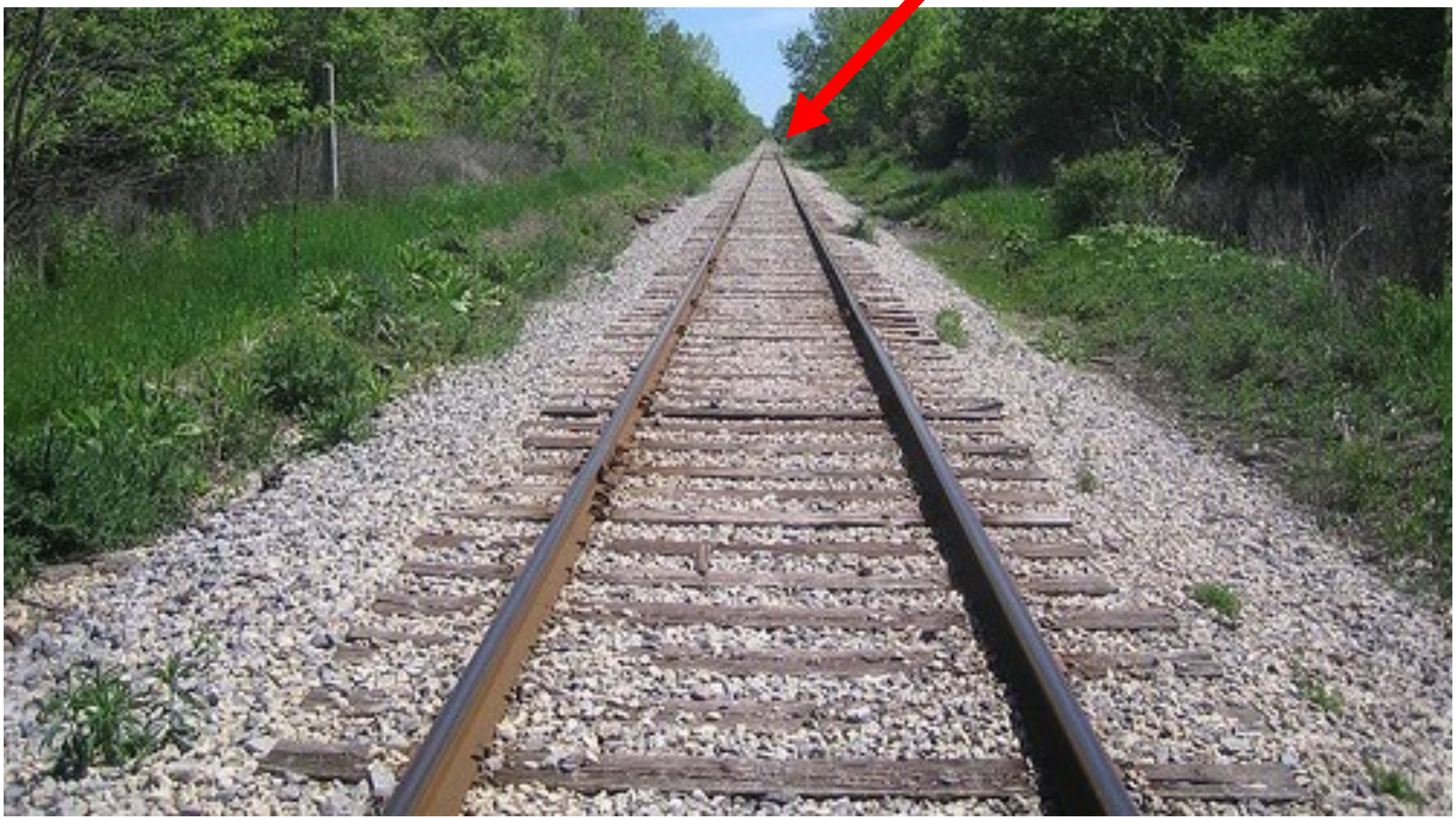
Properties of Projective Transformations

- Points project to points
- line project to lines, rays or degenerate into points
- Distant objects look smaller

Properties of Projection

- Angles are not preserved
- Parallel lines meet (except for horizontal lines)

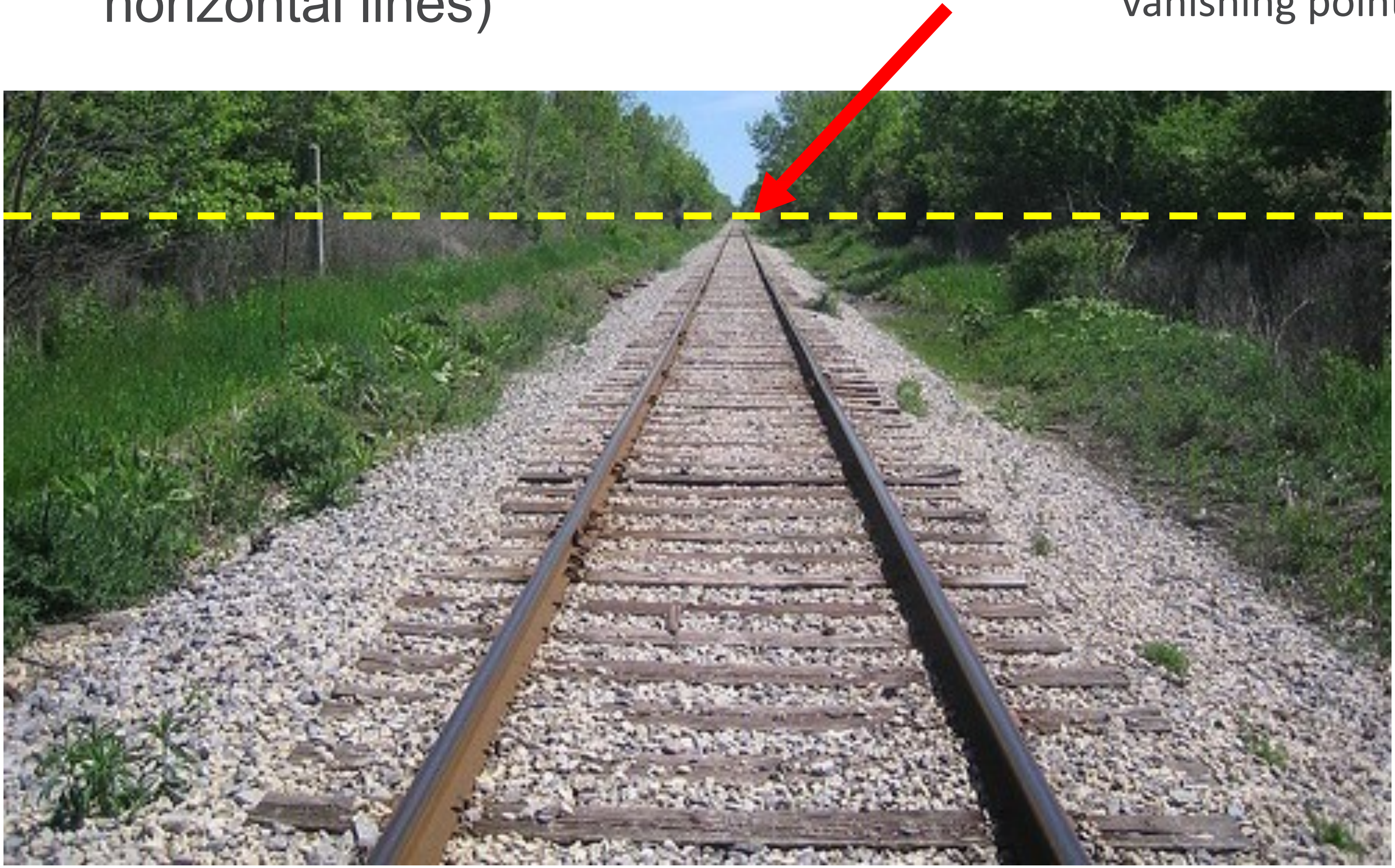
Parallel lines in the world intersect in the image at a “vanishing point”



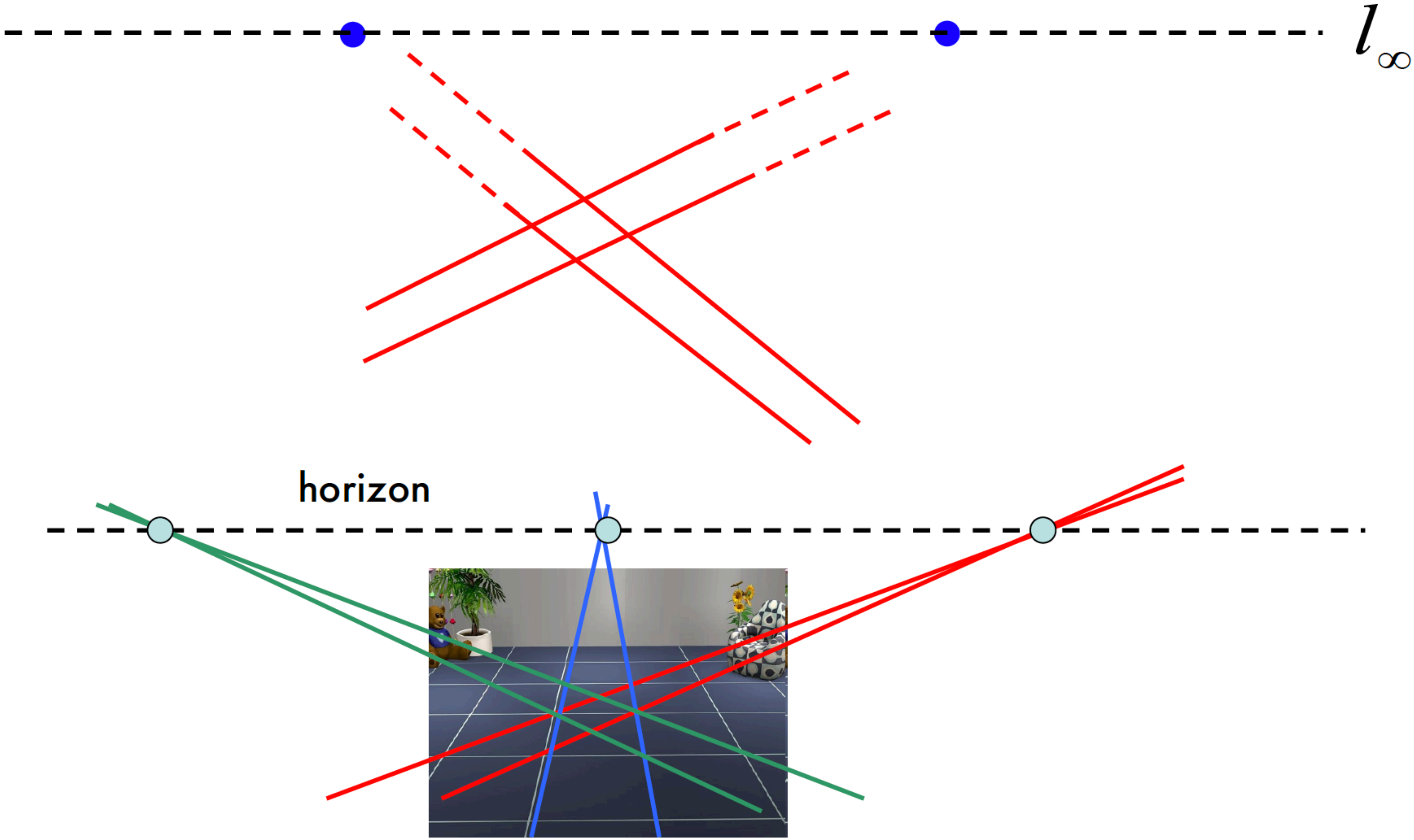
Horizon Line (Vanishing Line)

- Angles are not preserved
- Parallel lines meet (except for horizontal lines)

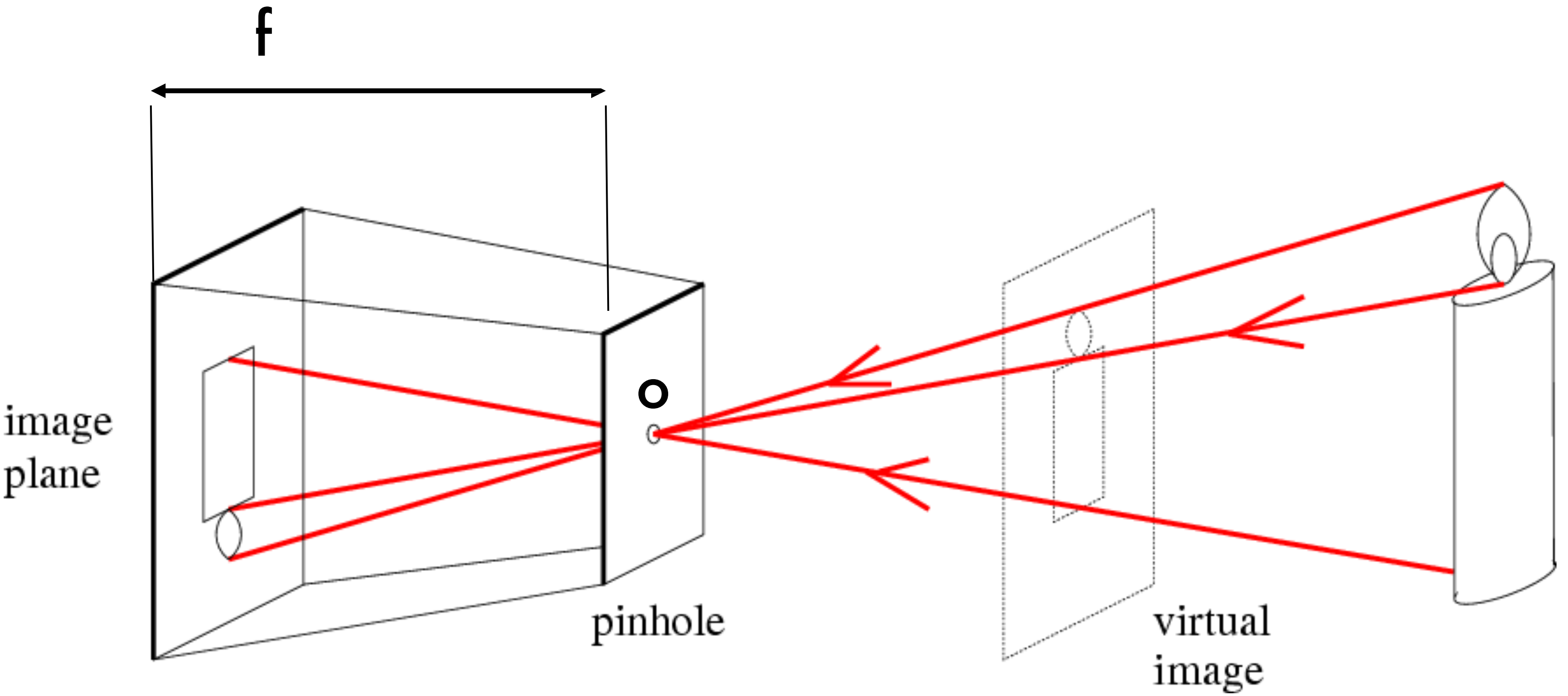
Parallel lines in the world intersect in the image at a "vanishing point"



Horizon Line (Vanishing Line)



Will Use Virtual Image In the Future



f = focal length

o = aperture = pinhole = center of the camera

Multi-View Geometry

Agenda

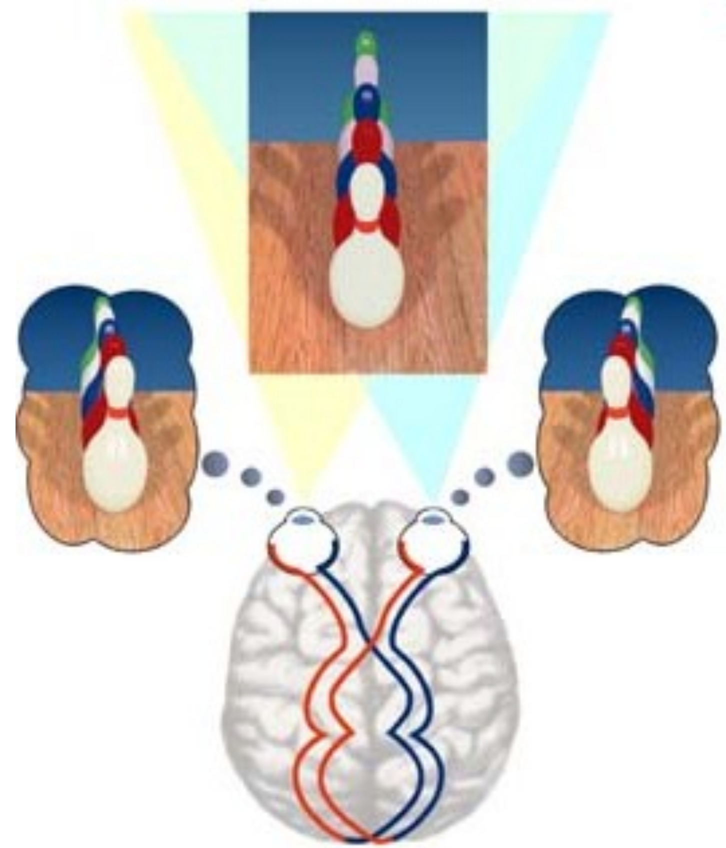
- Why is stereo useful?
- Epipolar constraints
- Fundamental matrix

Recovering Structure From a Single View

Intrinsic ambiguity of the mapping from 3D to image (2D)

Courtesy slide S. Lazebnik

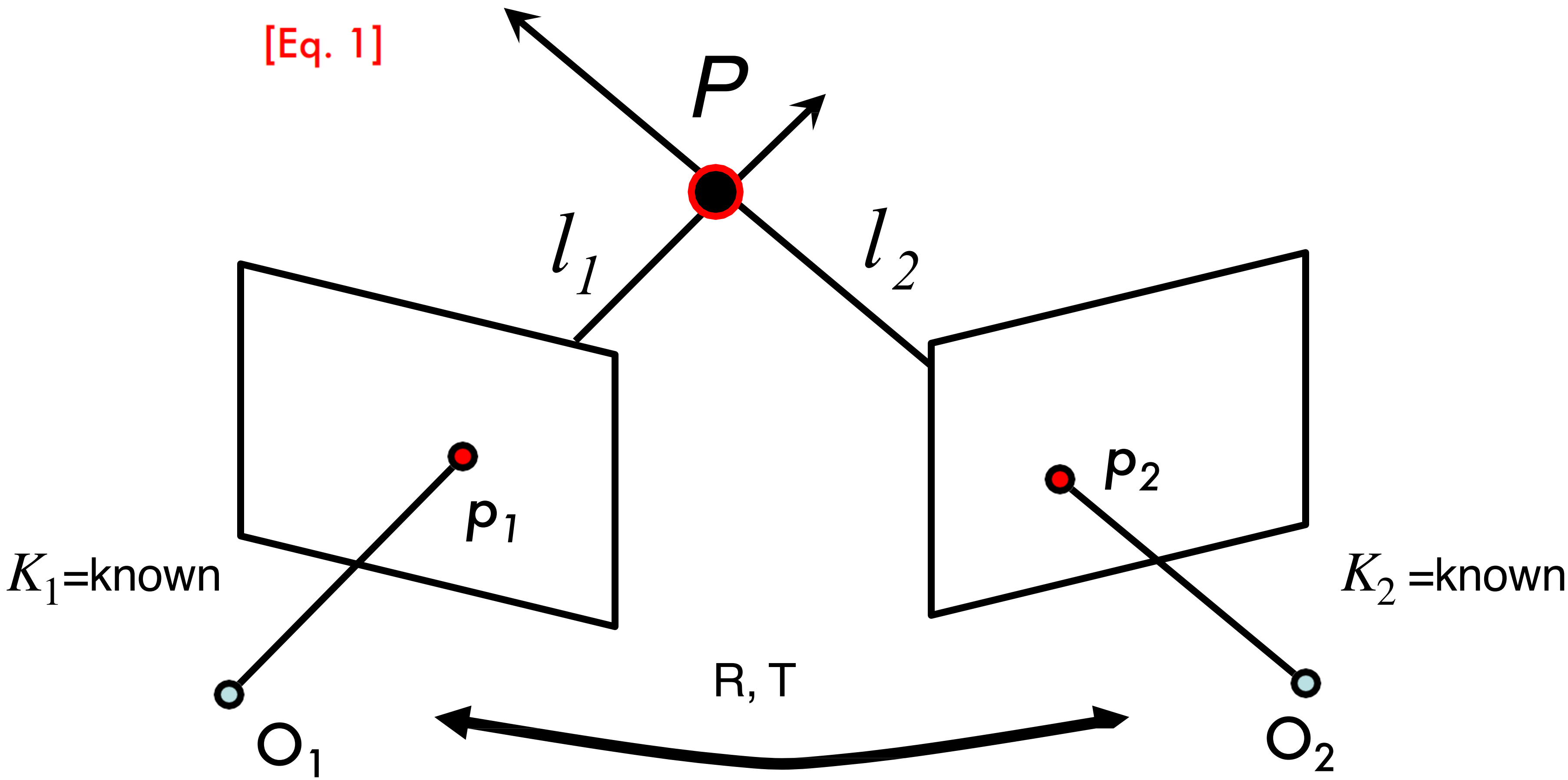
Two Eyes Help!



Two Eyes Help!

$$P = l_1 \times l_2$$

[Eq. 1]

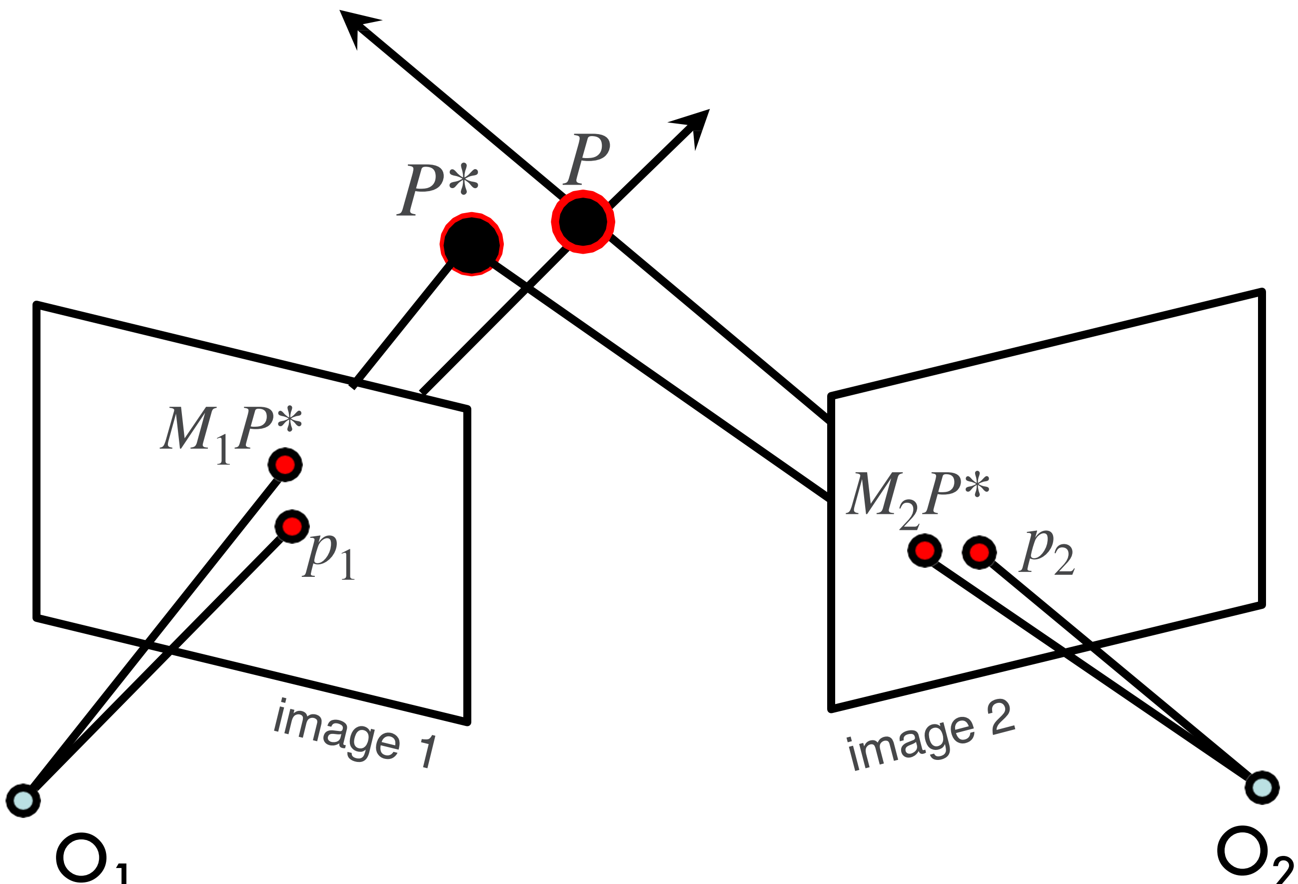


This is called triangulation

Triangulation

- Find P* that minimizes

$$d(p_1, M_1P^*) + d(p_2, M_2P^*)$$ [Eq. 2]



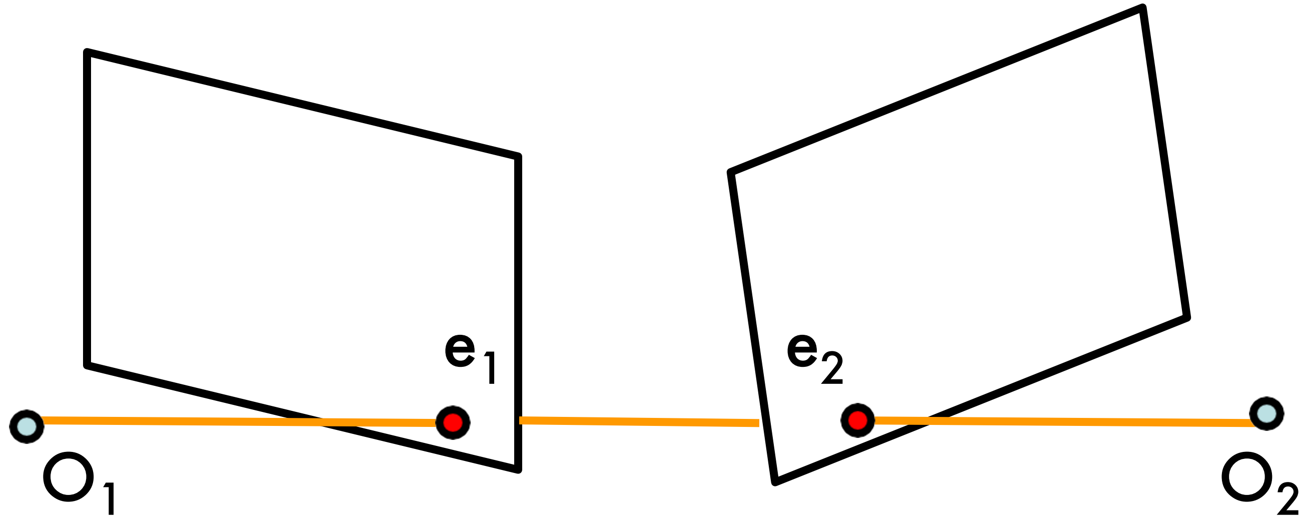
Multi (stereo)-View Geometry

- **Camera geometry:** Given corresponding points in two images, find camera matrices, position, and pose.
- **Scene geometry:** Find coordinates of 3D point from its projection into 2 or multiple images.
- **Correspondence:** Given a point p in one image, how can I find the corresponding point p' in another one?

Agenda

- Why is stereo useful?
- **Epipolar constraints**
- Fundamental matrix

Epipolar Geometry

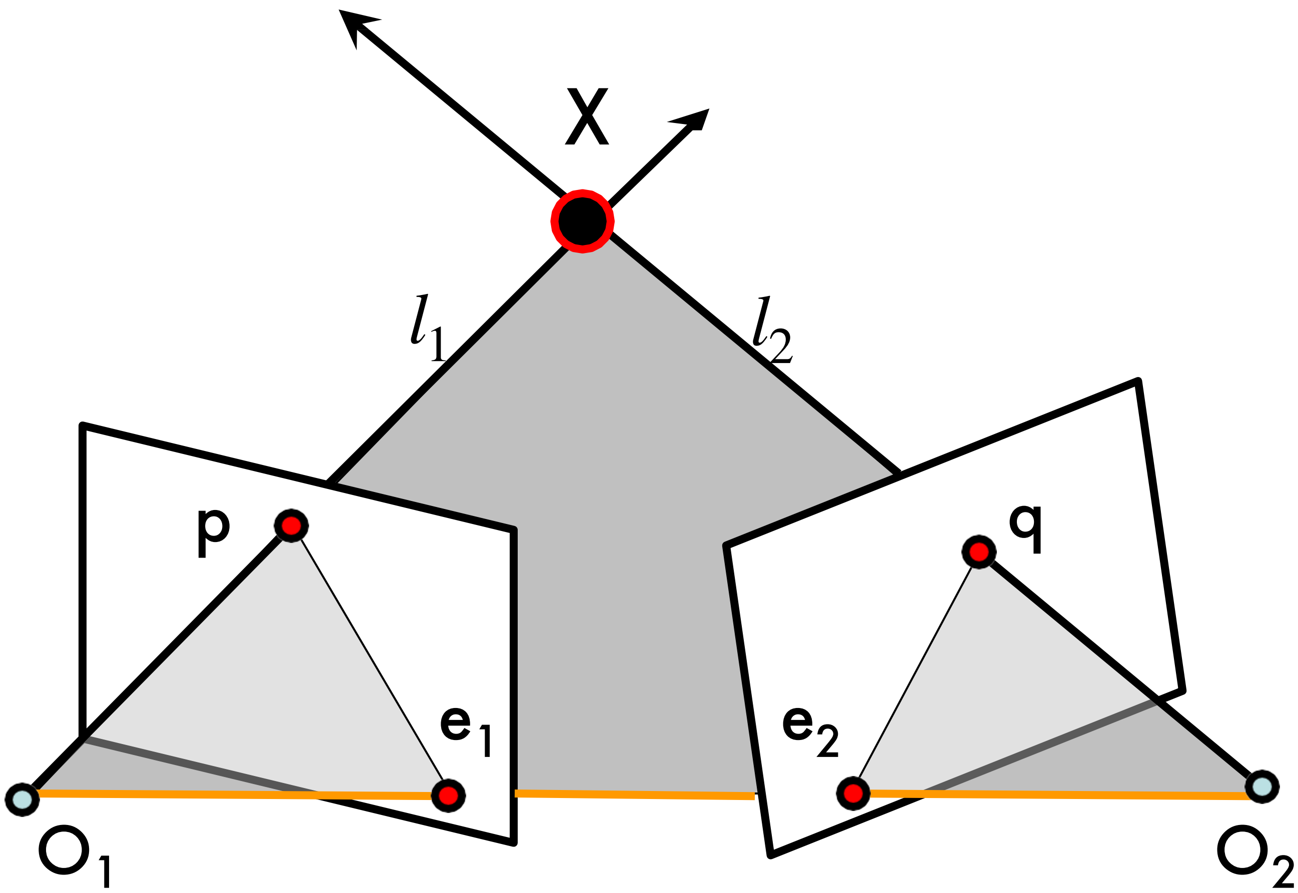


- Baselines
- Epipoles: e_1 , e_2

  = intersections of baseline with image planes

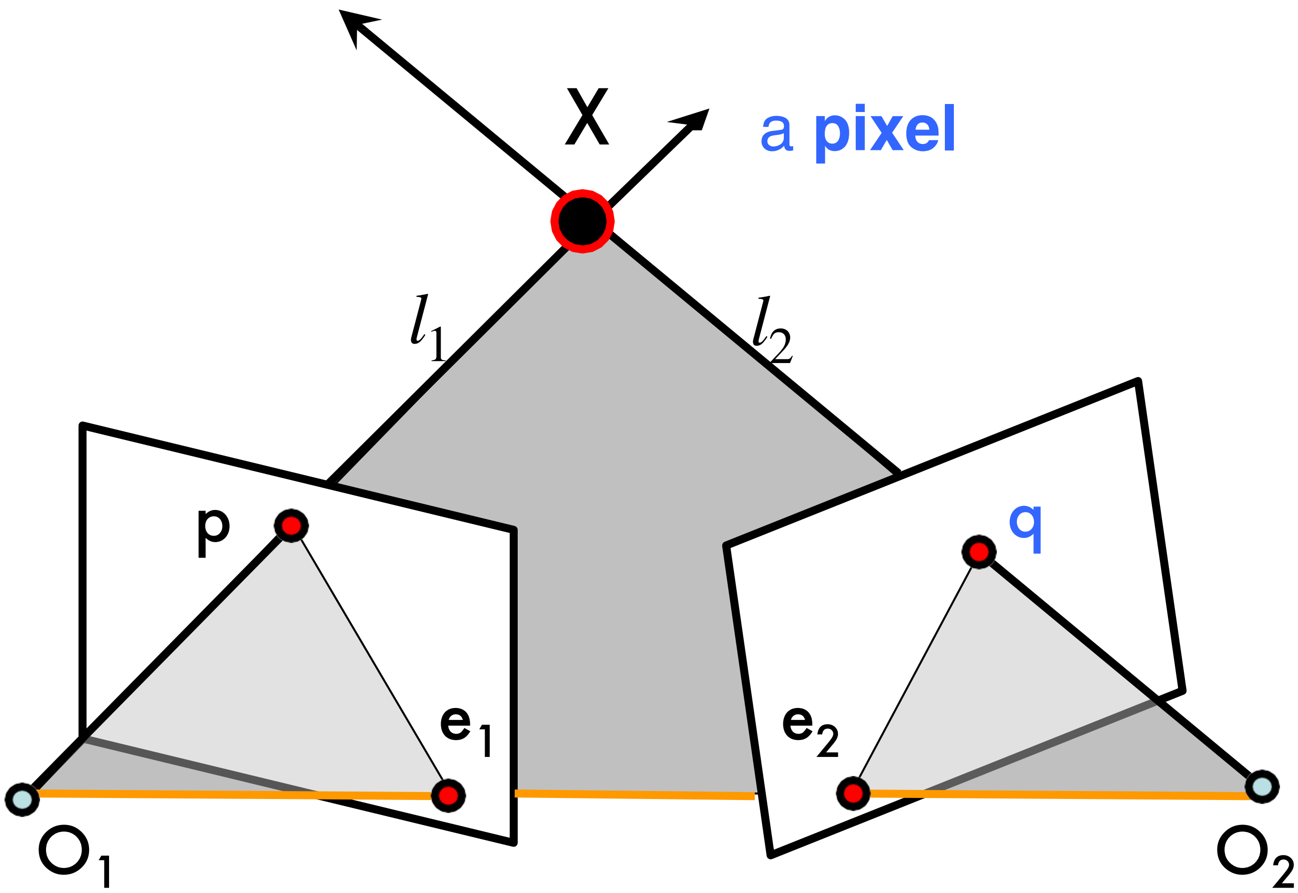
  = projections of the other camera center

Epipolar Geometry



- Baselines
- Epipolar plane
- Epipoles: e_1 , e_2
  = intersections of baseline with image planes
  = projections of the other camera center

Epipolar Geometry

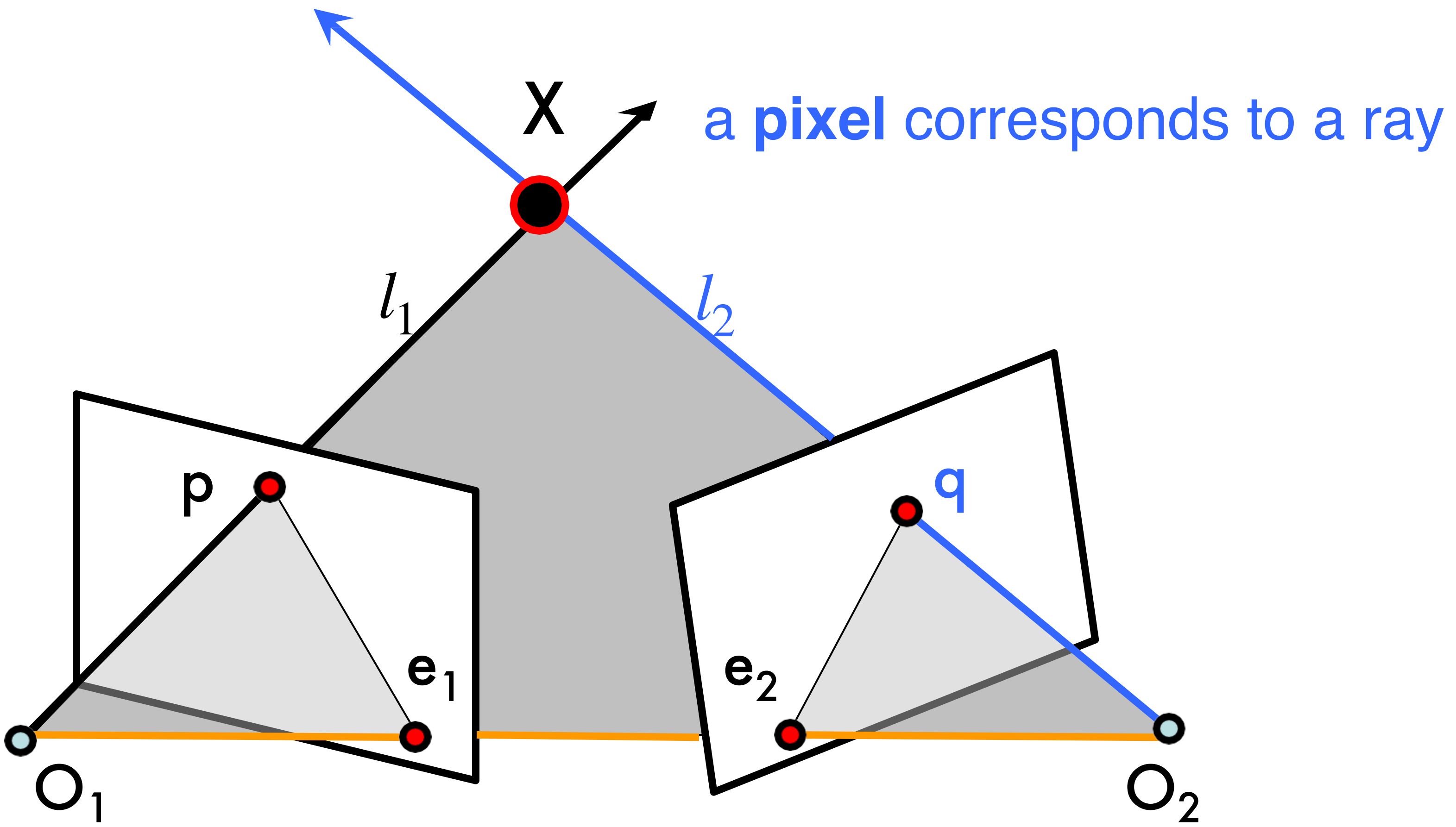


- Baselines
- Epipolar plane
- Epipoles: e_1 , e_2

  = intersections of baseline with image planes

  = projections of the other camera center

Epipolar Geometry

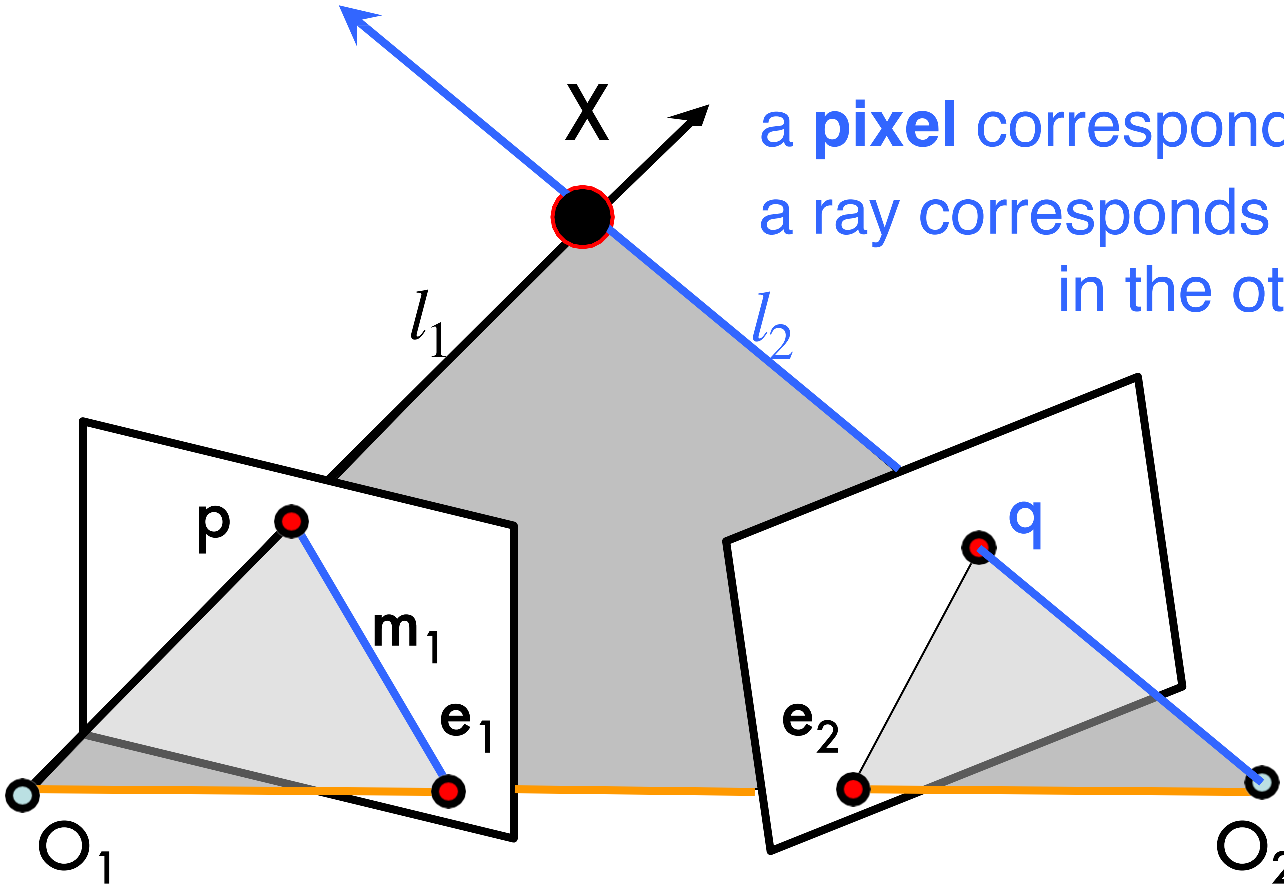


- Baselines
- Epipolar plane
- Epipoles: e_1 , e_2

  = intersections of baseline with image planes

  = projections of the other camera center

Epipolar Geometry

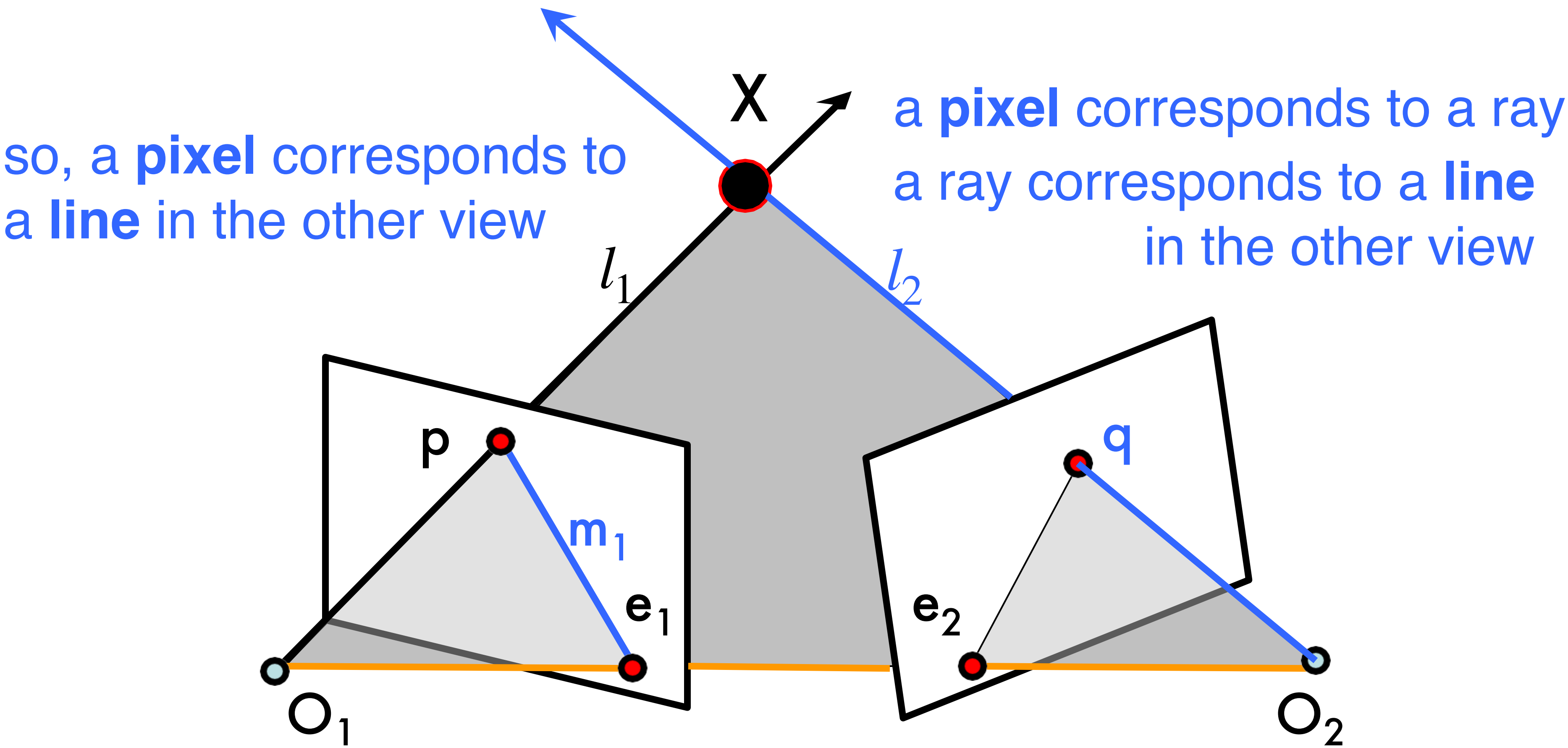


a **pixel** corresponds to a ray
a ray corresponds to a **line**
in the other view

- Baselines
- Epipolar plane

- Epipoles: e_1 , e_2
  = intersections of baseline with image planes
  = projections of the other camera center

Epipolar Geometry

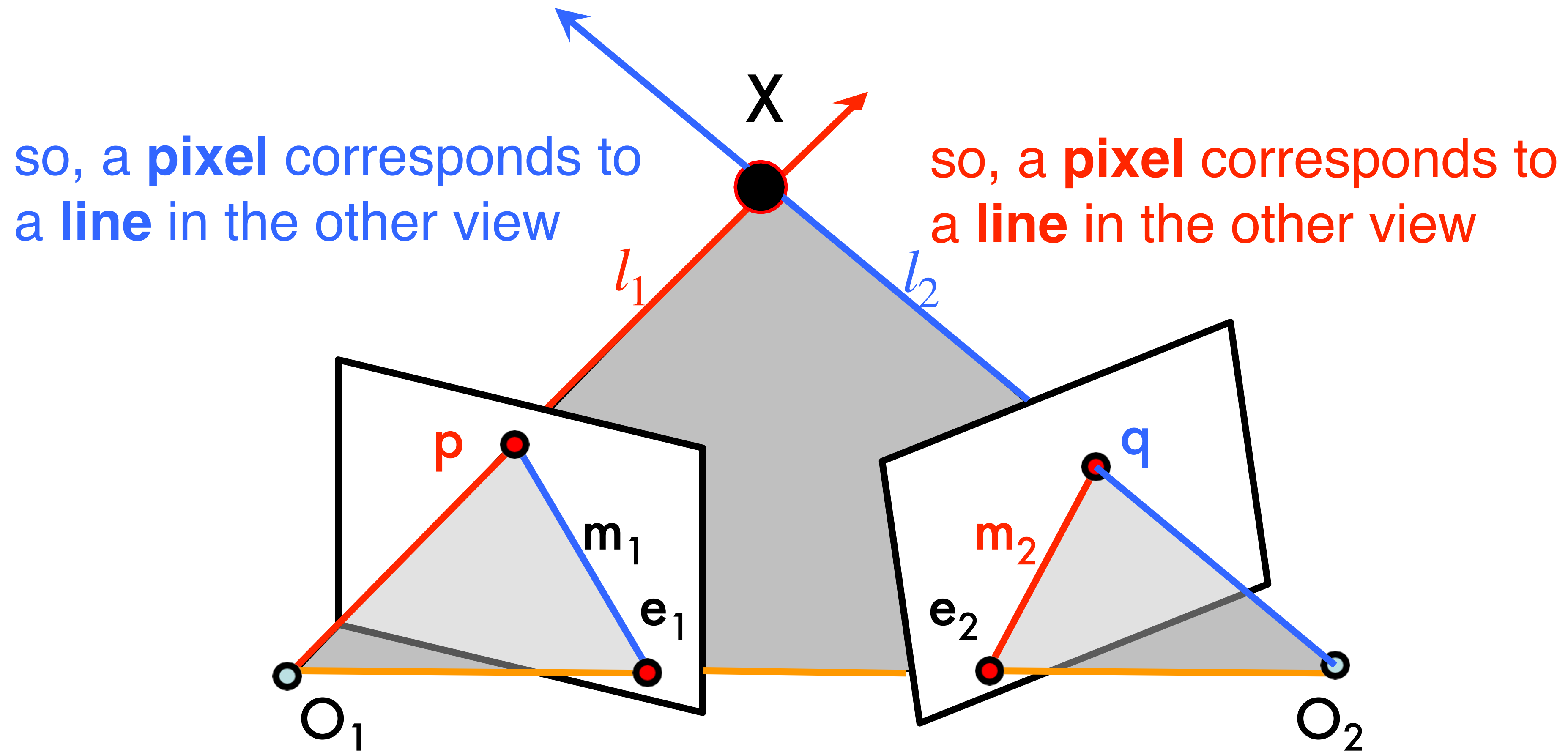


- Baselines
- Epipolar plane
- Epipoles: e_1 , e_2

  = intersections of baseline with image planes

  = projections of the other camera center

Epipolar Geometry



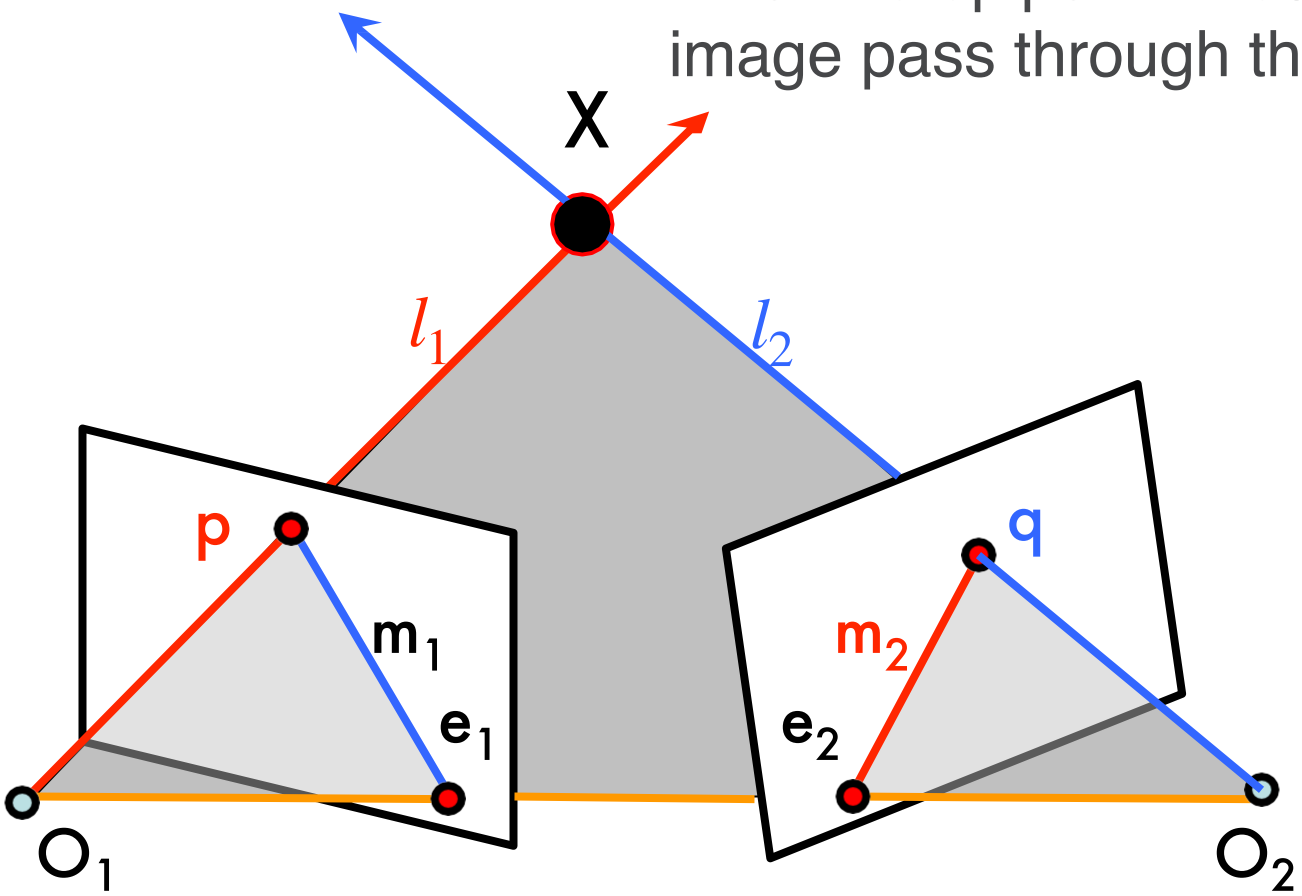
- Baselines
- Epipolar plane
- Epipolar line
- Epipoles: e_1 , e_2

  = intersections of baseline with image planes

  = projections of the other camera center

Epipolar Geometry

All of the epipolar lines in an image pass through the epipole.



- Baselines
- Epipolar plane
- Epipolar line

- Epipoles: e_1 , e_2

  = intersections of baseline with image planes

  = projections of the other camera center

Example of Epipolar Lines