CIS 5800

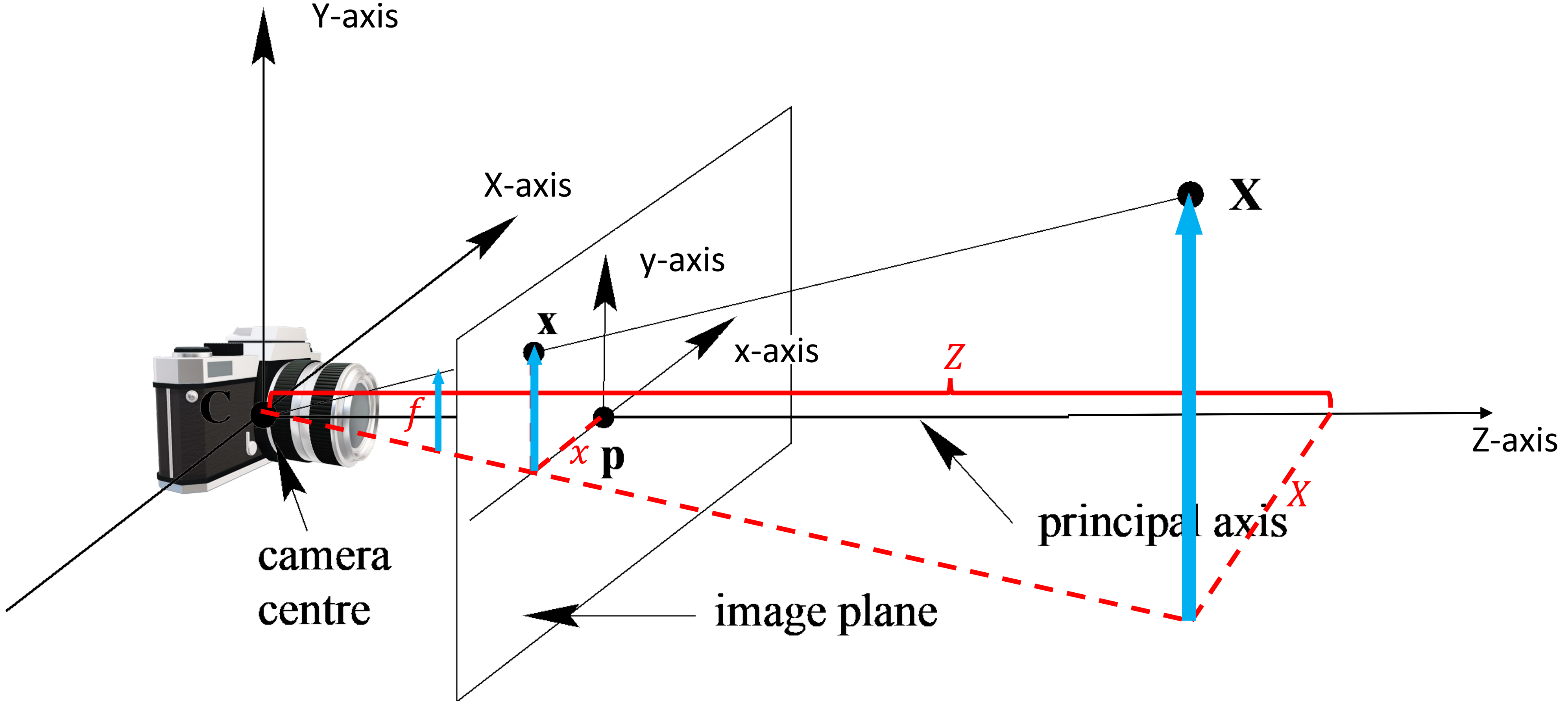
Machine Perception

- Midterm Exam Review

Instructor: Lingjie Liu

Lec 15: March 24, 2025

Basic Perspective Projection Equations



$$\boxed{x = f\frac{X}{Z}, y = f\frac{Y}{Z}}$$

Place in the Hierarchy of Transformations

Projective

Affine

Minus Orthonormality Constraint

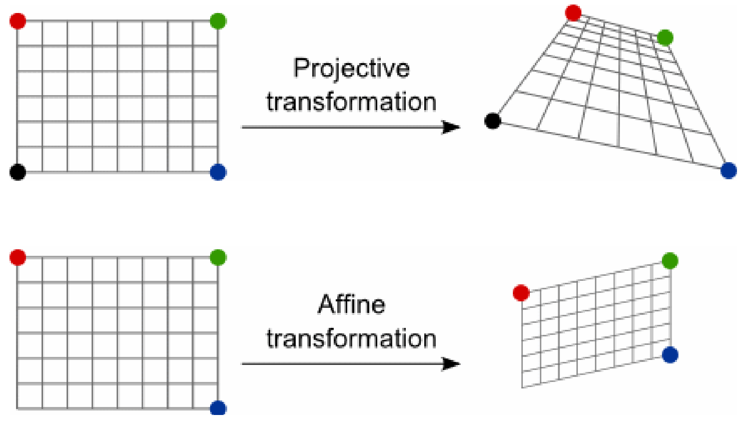
Similarity

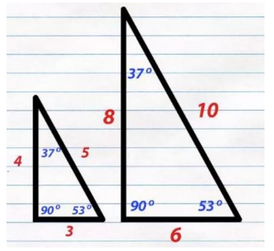
Plus Scale

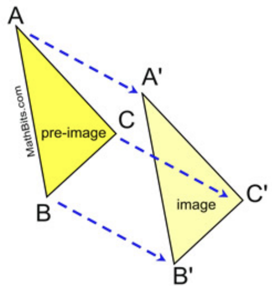
Euclidean

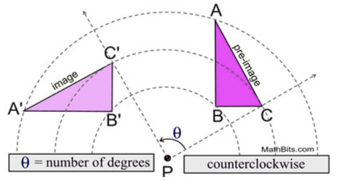
Shifts

Rotations









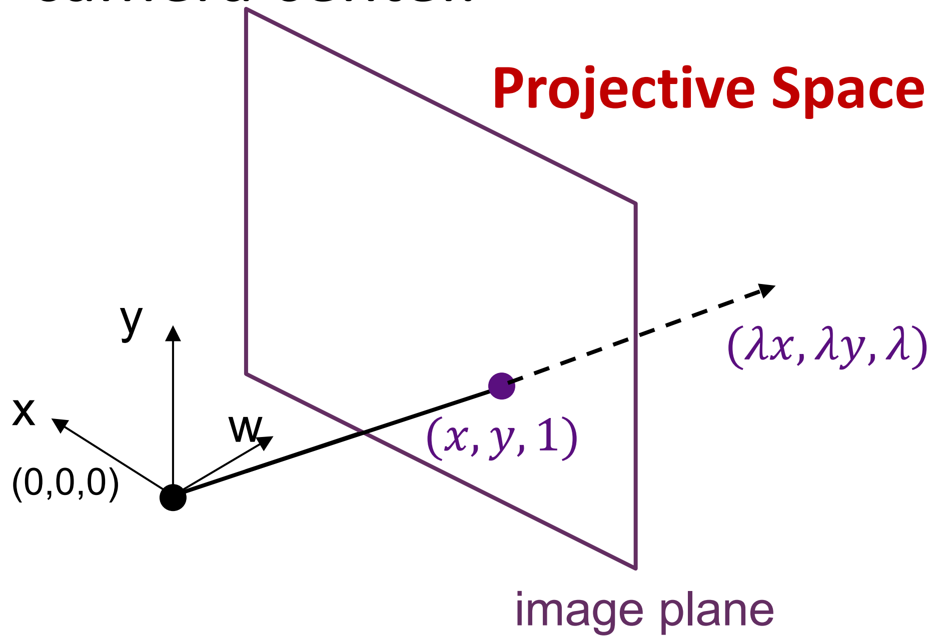
Projective Geometry

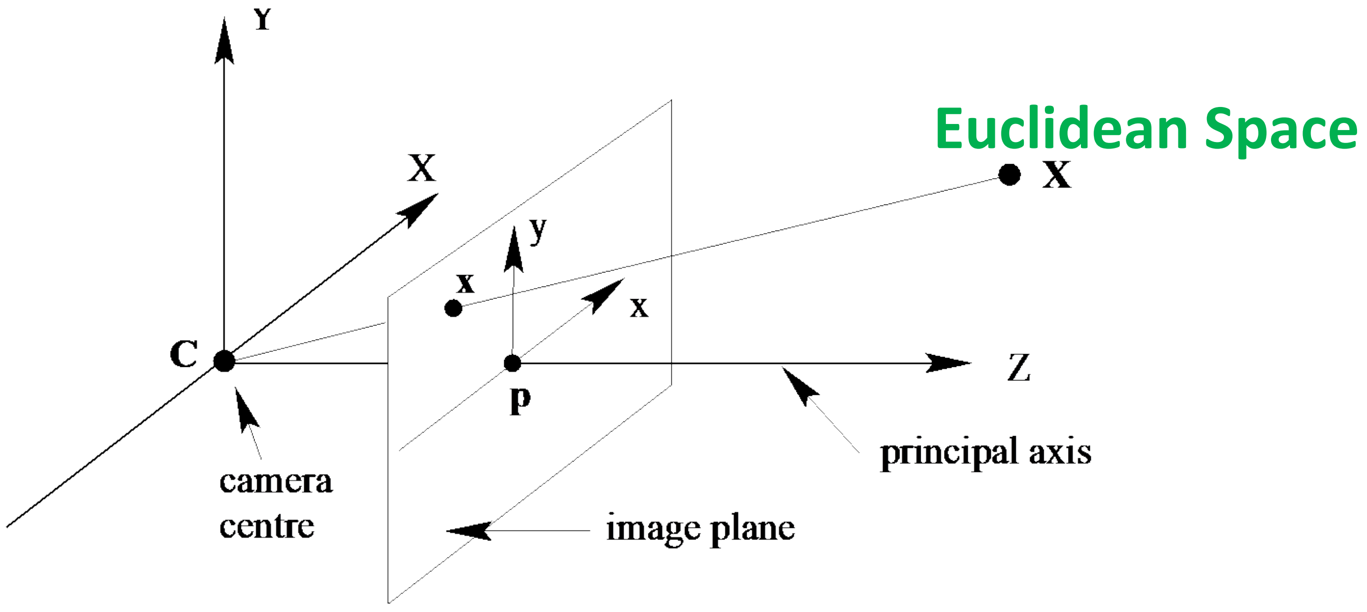
Based on slides by Jianbo Shi, Hyun Soo Park, Kostas Daniilidis

Projective geometry $\leftrightarrow$ Euclidean interpretation

In the Euclidean interpretation, we treat w as the third spatial coordinate.

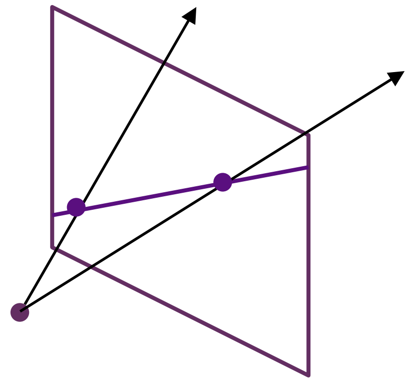
- The w axis is a scaled version of the principal axis Z (in camera-centric coordinates).
- The image plane is $w = 1$, same as $Z = f$
- $w = 0$ is the same as $Z = 0$. Parallel to image plane, passing through camera center.





Projective lines

- What does a line in the image correspond to in projective space?

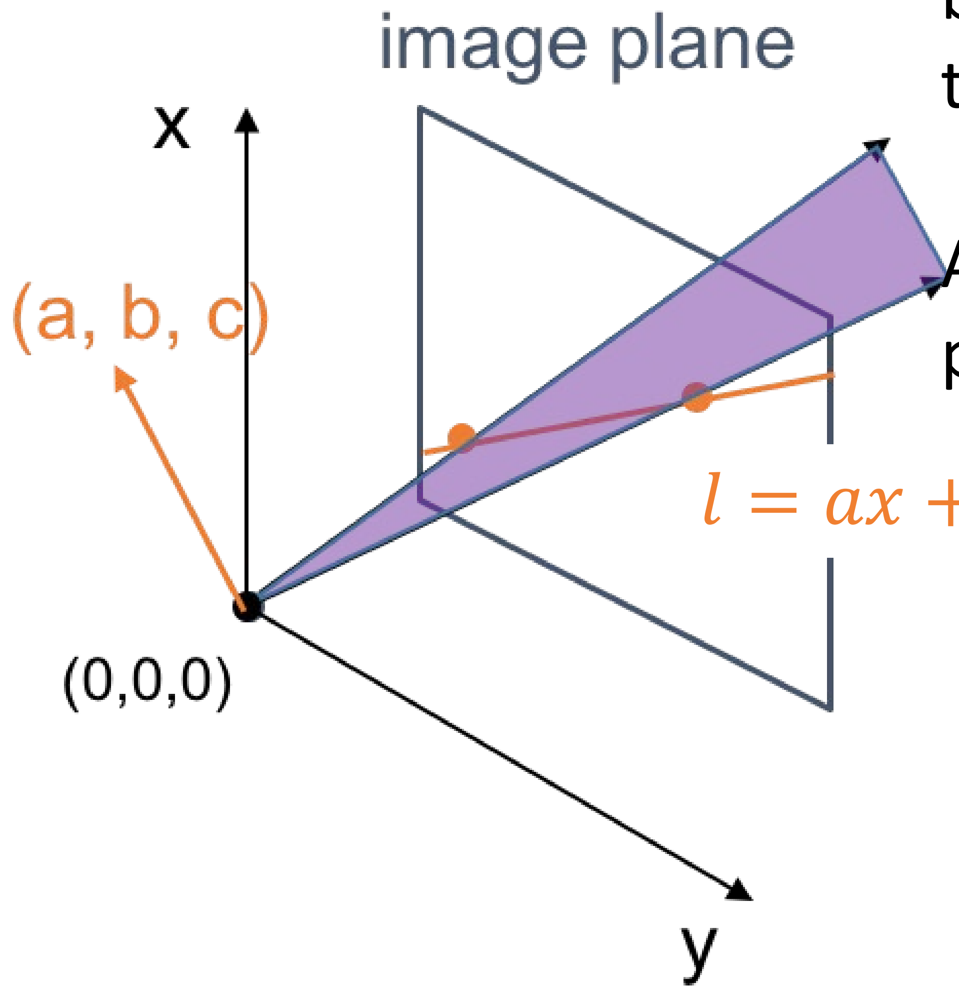


- A line is a *plane* of rays through origin
  - all rays (x, y, w) satisfying: $ax + by + cw = 0$

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0$$

$$\boldsymbol{l}^T \; \boldsymbol{x} = 0$$

Projective Lines



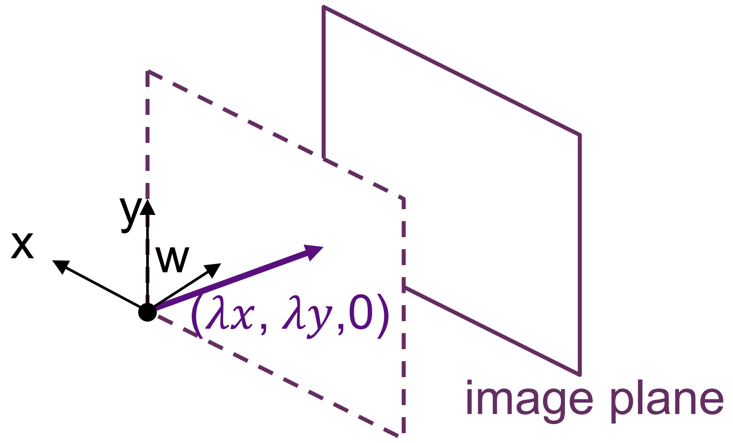
$l = ax + by + cw = 0$

Q: what is the relationship between $l = (a, b, c)^T$ and the plane?

A: It is perpendicular to the plane.

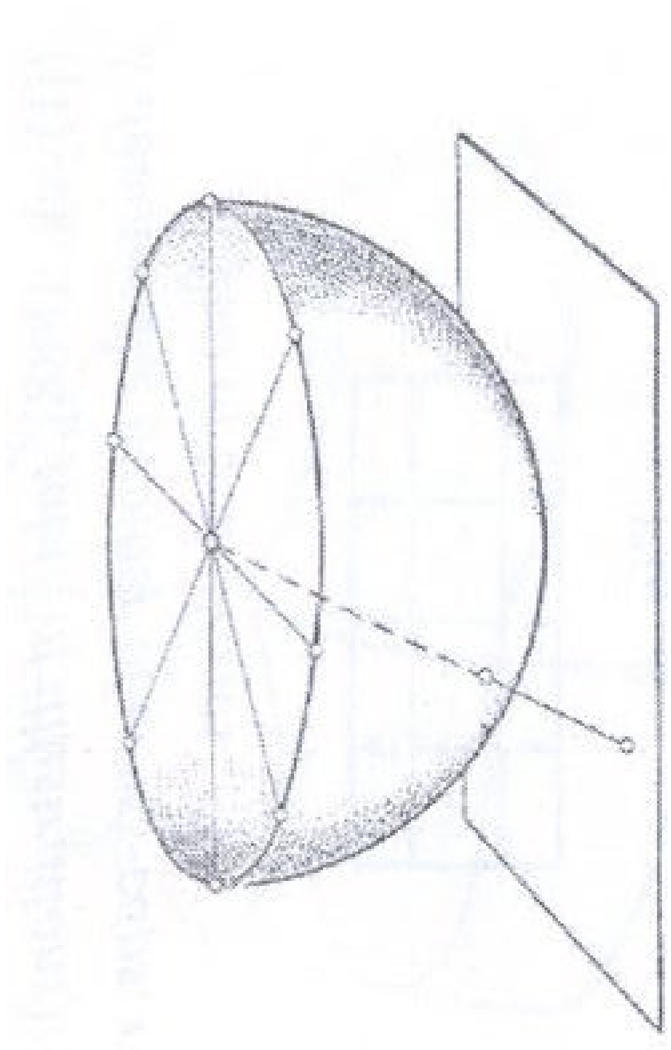
Point at infinity / “ideal” points

- Ideal point (“point at infinity”)
  - p $\cong (x, y, 0)$ – rays through camera center parallel to image plane
  - It has infinite image coordinates



Point at infinity / “ideal” points

$$(\underbrace{x_1,x_2}_{\text{Looking-at direction}},\underbrace{0}_{\text{“Ideal” points}})$$



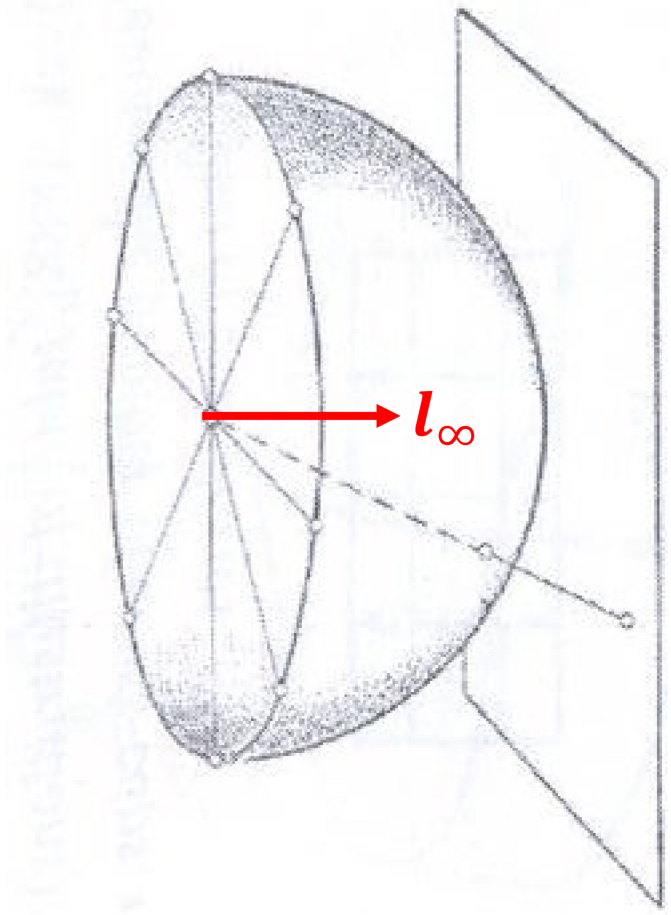
“Line at infinity”

- A line passing through all ideal points i.e. point

$$l_\infty = (0, 0, 1)^T$$

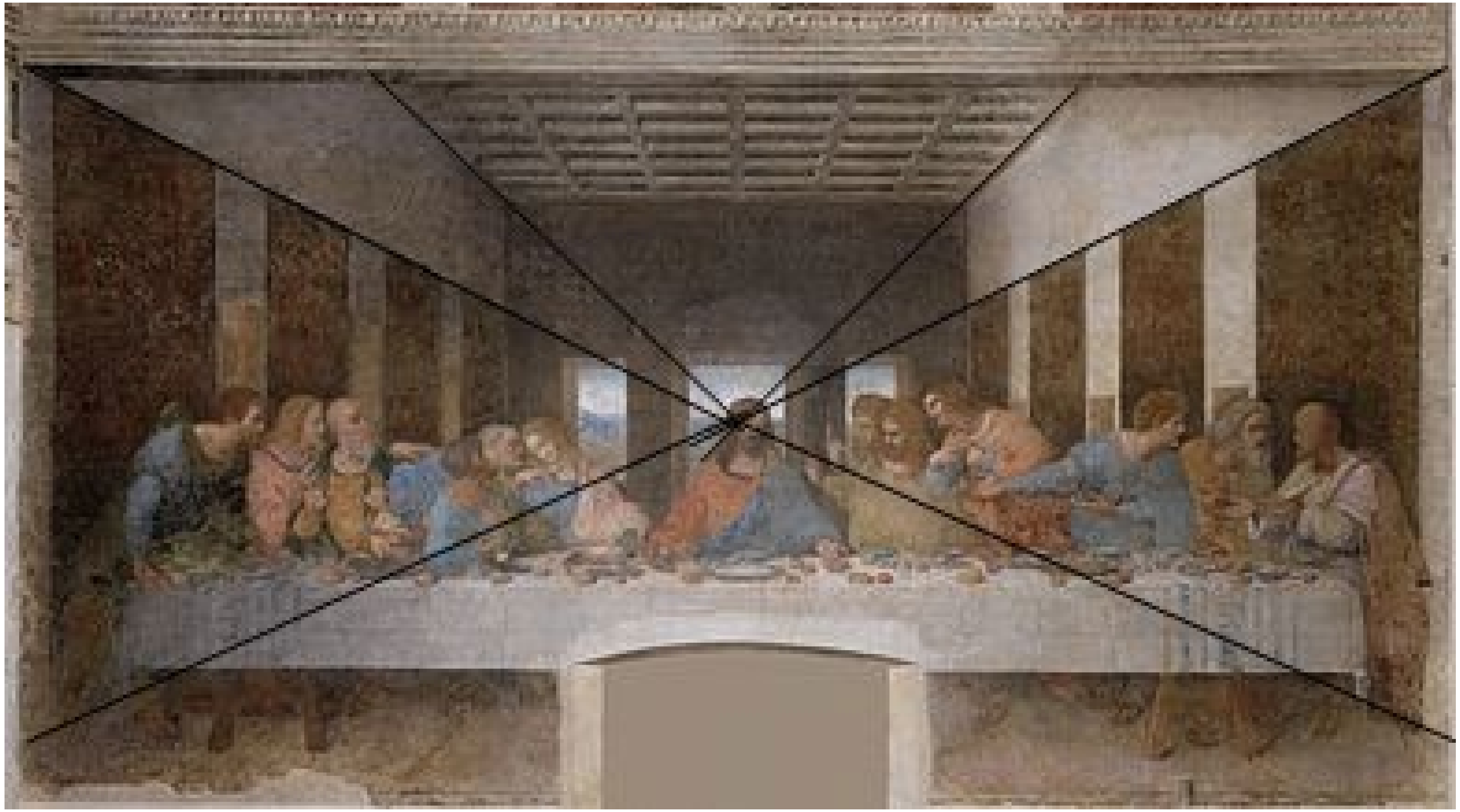
- Because :

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix} = 0$$

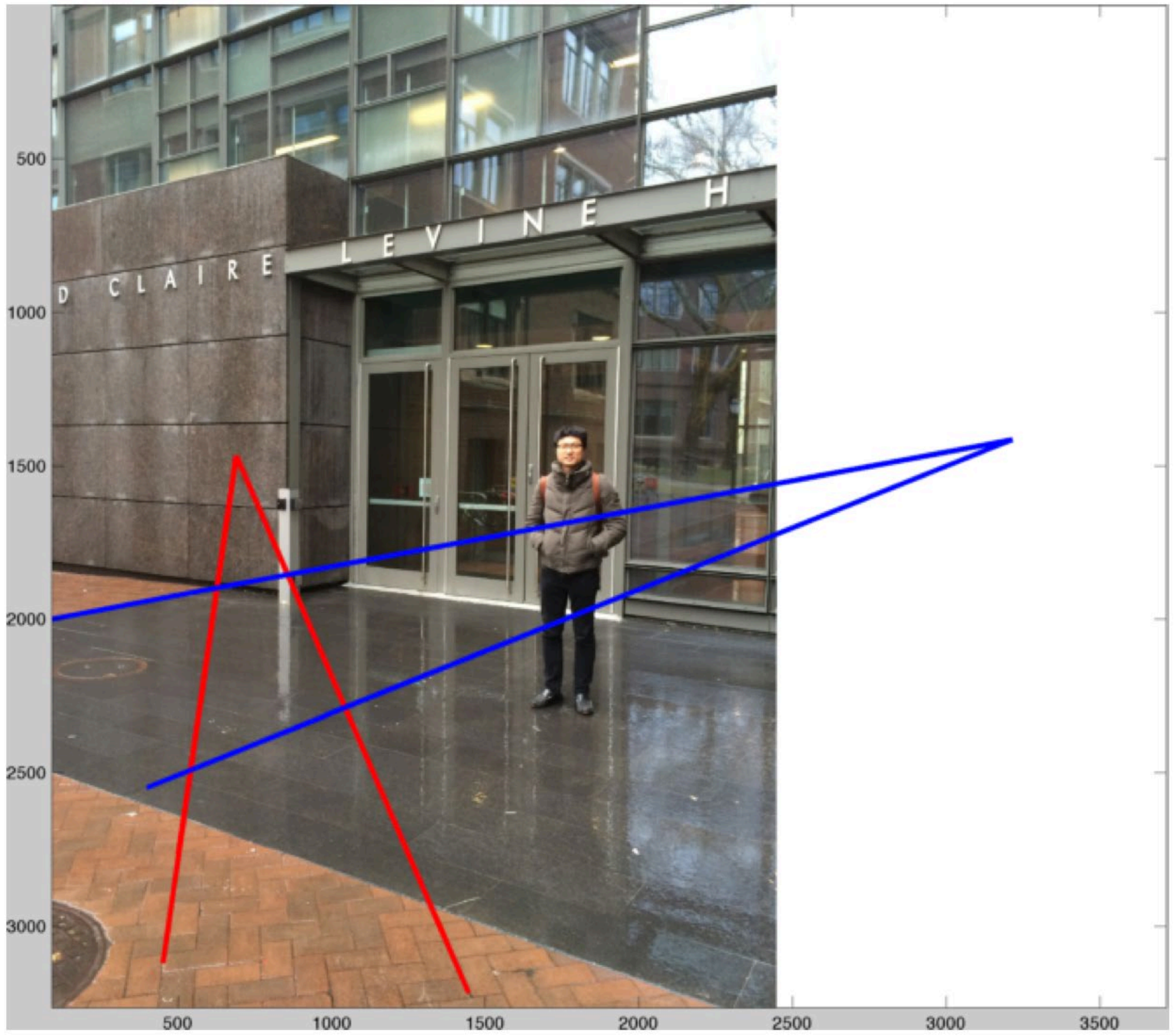


Vanishing points

A vanishing point is a point on the image plane of a perspective rendering where the two-dimensional perspective projections of mutually parallel lines in three-dimensional space appear to converge.

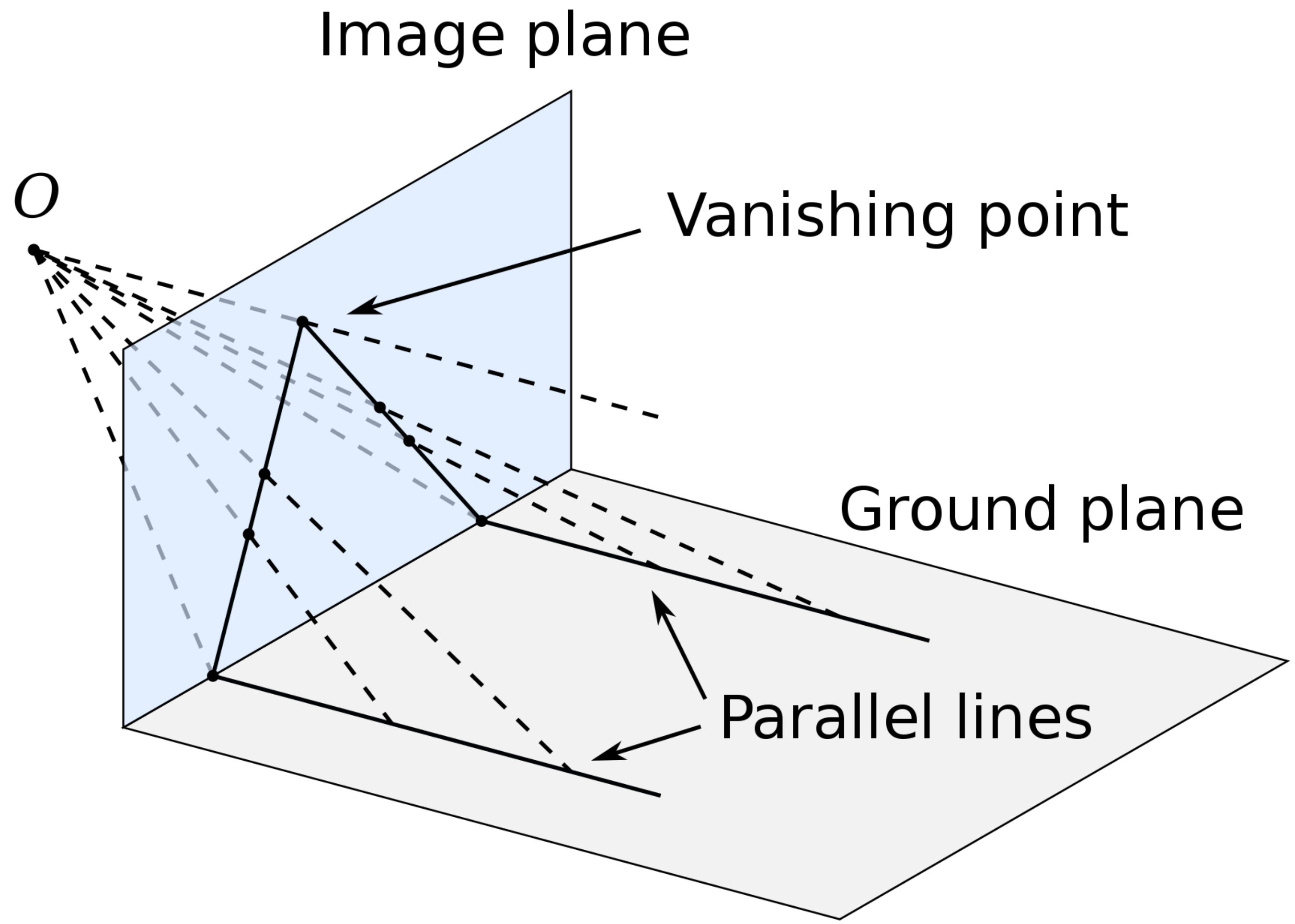


Da Vinci’s ”The Last Supper” c. 1495-98. http://pennpaint.blogspot.com/



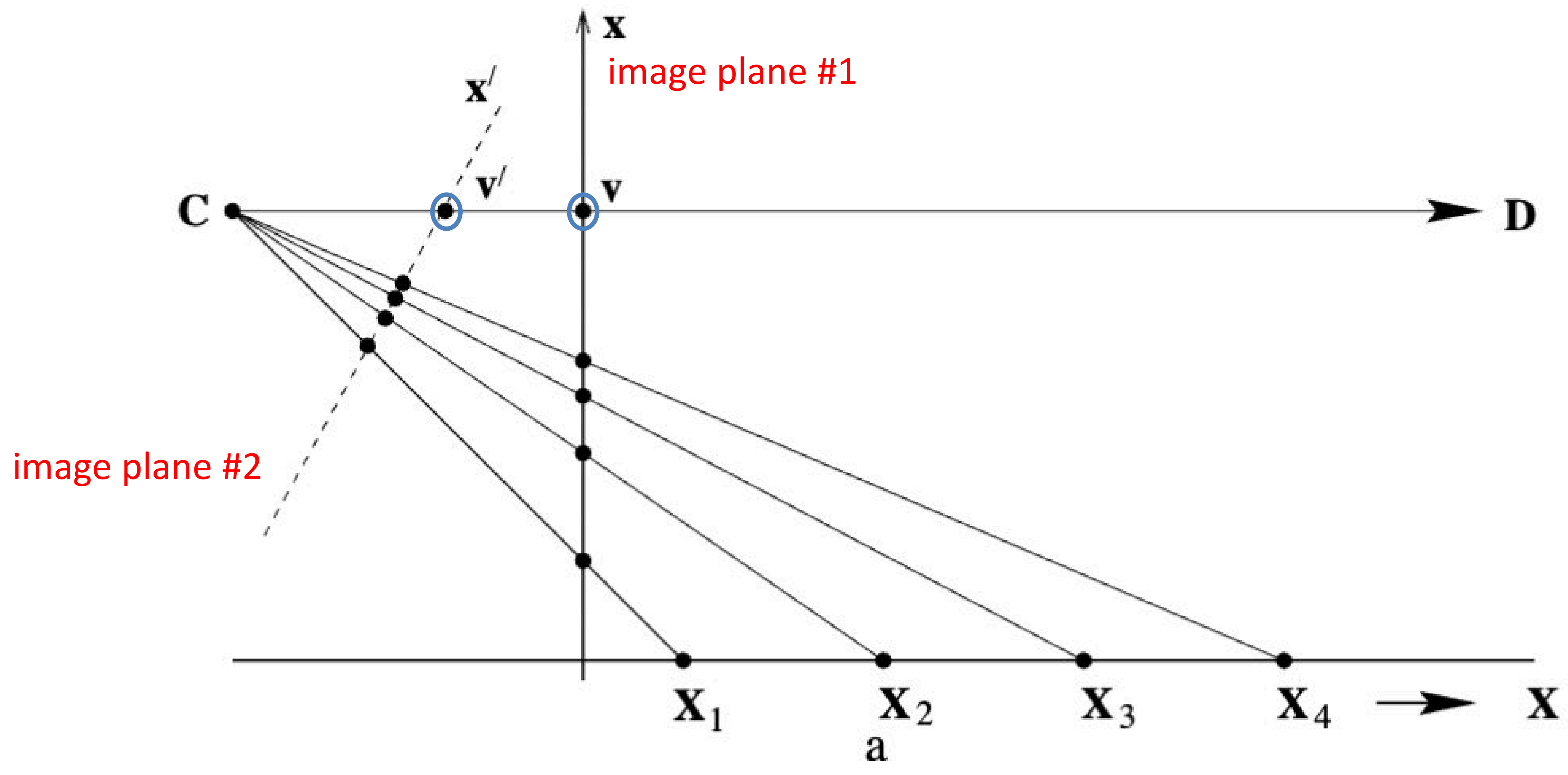
500
1000
1500
2000
2500
3000
500
1000
1500
2000
2500
3000
3500
D CLAIRE
LEVINE H

Where vanishing points come from



ZHCh8

The line connecting the camera origin and the vanishing point is parallel to all lines that share the same direction and converge at the vanishing point.

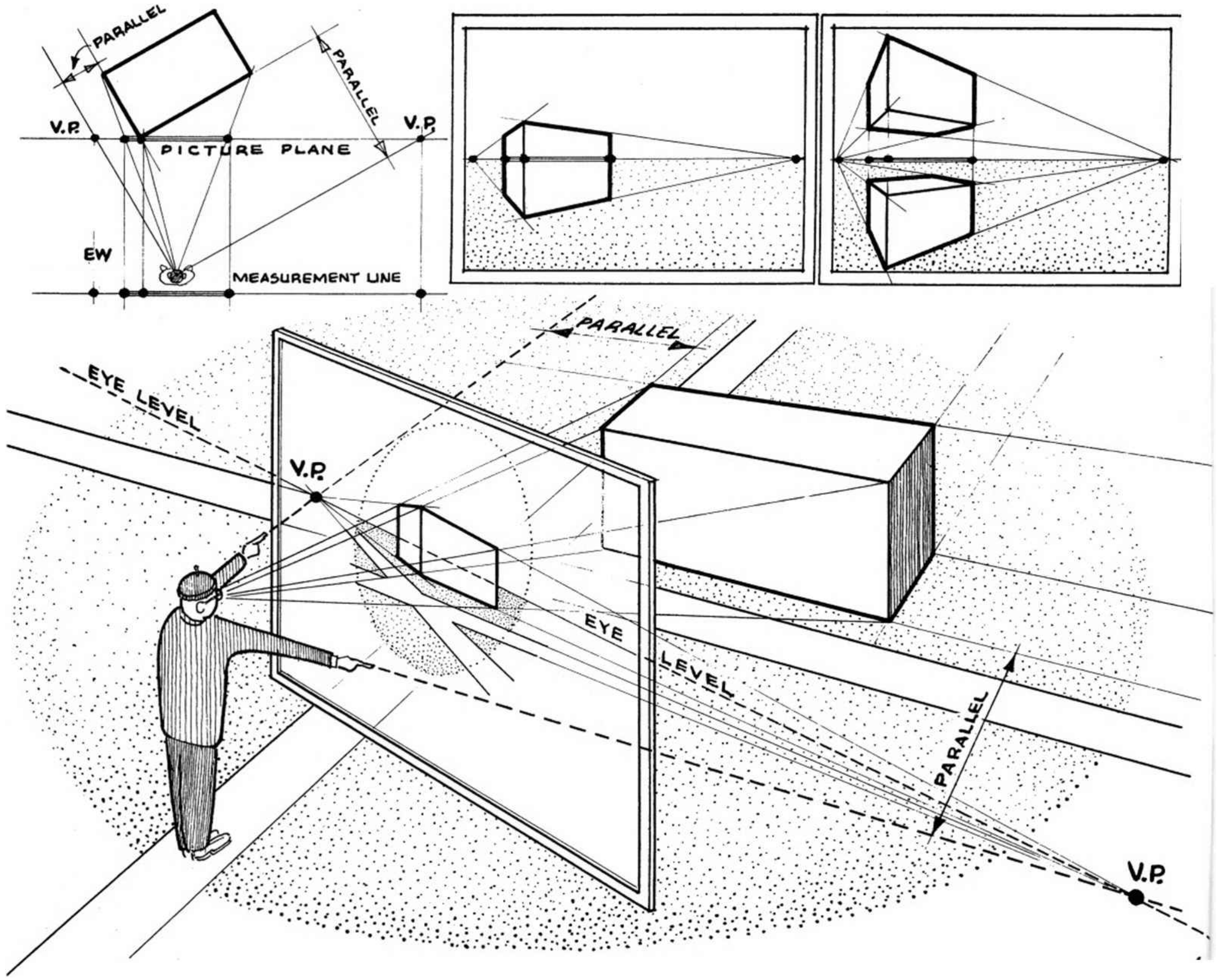


How Artists Find Vanishing Points

Find VP of a world line by:

- Standing at "camera center".
- Holding arm out parallel to the world line.
- Noting its intersection with the "canvas" or image plane. i.e. the arm represents the light ray.

"Vanishing rays of a world line" (camera rays through the VP) are just rays parallel to that line, passing through the camera center.

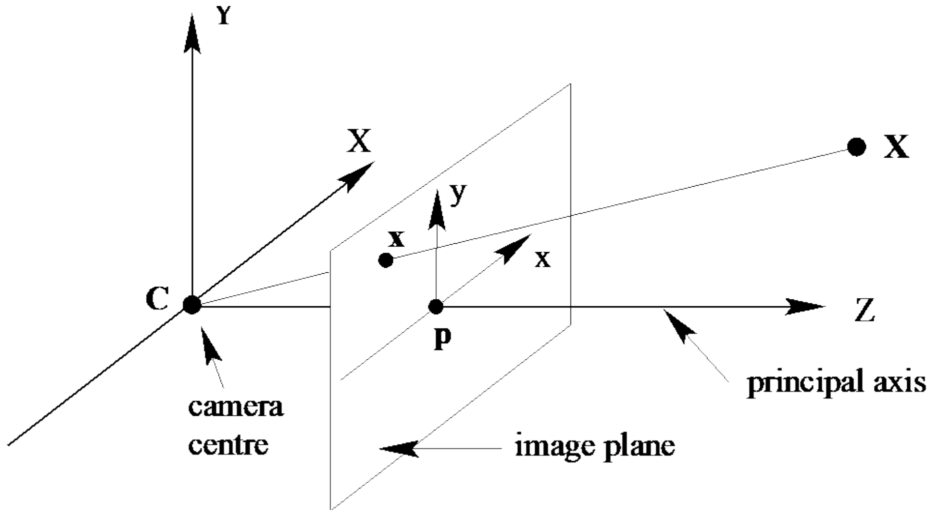


Perspective Projections are Linear in $\mathbb{P}$

Recall: $\boxed{x = f\frac{X}{Z}, y = f\frac{Y}{Z}}$

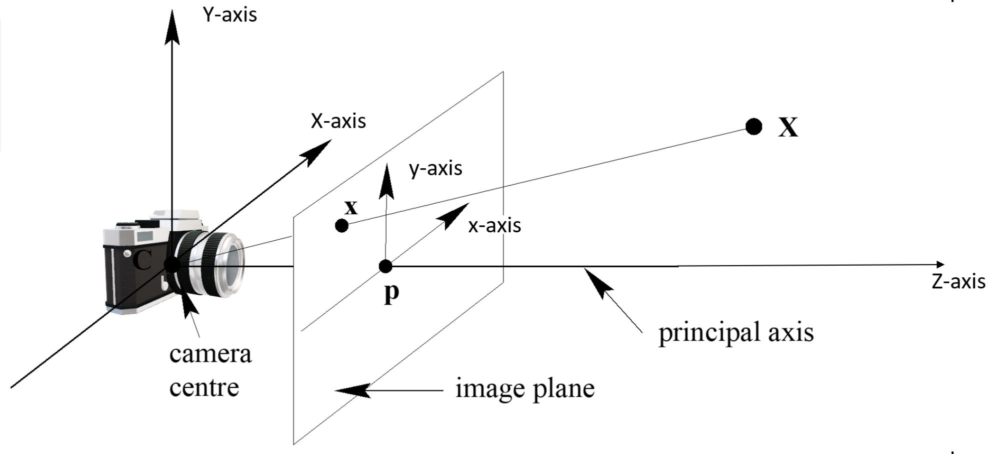
$\updownarrow$

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix}$$



Camera Projection Equation

$$x = f\frac{X}{Z}, y = f\frac{Y}{Z}$$

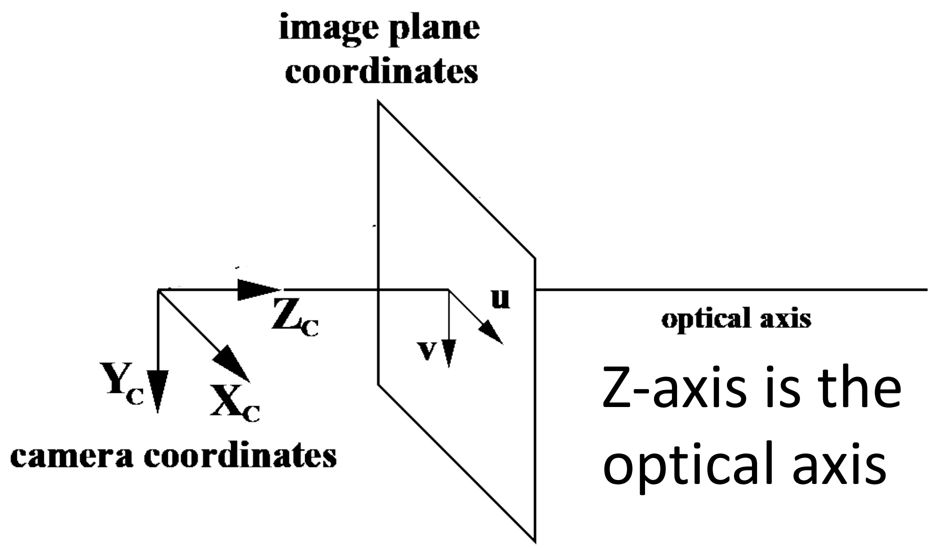


$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & \square & \square & 0 \\ \square & f & \square & 0 \\ \square & \square & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

This assumes that:

- the image coordinate system origin is the same as the “principal point” p where the principal / optical axis intersects image plane. Sometimes called “image center”
- Points in the 3D world are known in the *camera-centric* coordinate system.

Camera Coordinate System + Principal Point Offset



Z-axis is the optical axis

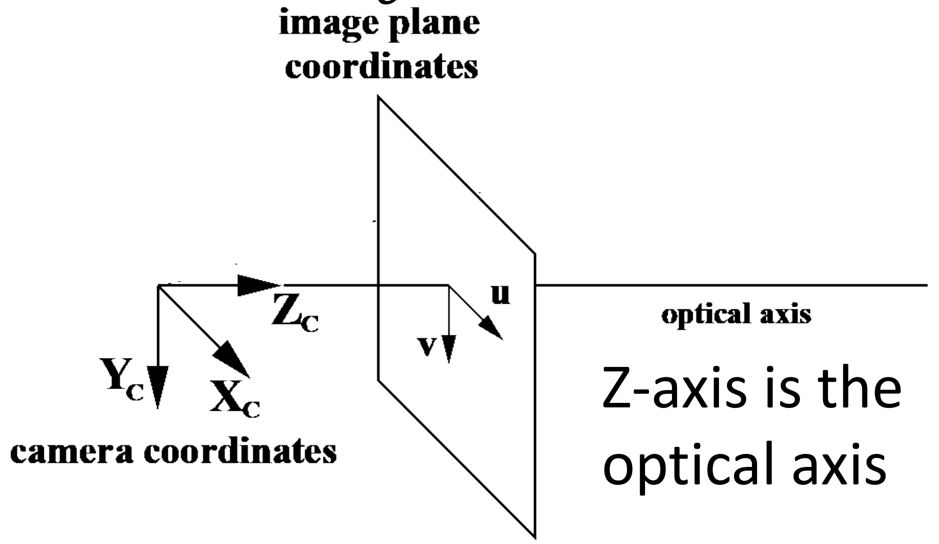
The image plane (u, v) is perpendicular to the optical axis.
Intersection of the image plane with the optical axis is the *image center* (u_o, v_o)

Projection in pixels

$$u = f\frac{X_c}{Z_c} + u_0, \qquad v = f\frac{Y_c}{Z_c} + v_0$$

Projection equation with image origin ≠ principal point

$$u = f\frac{X_c}{Z_c} + u_0, \qquad v = f\frac{Y_c}{Z_c} + v_0$$



Z-axis is the optical axis

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = K[I|0]\boldsymbol{X}_c$$

Generalizing intrinsics (1/2)

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} [R_{3\times3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

1. If pixels are not square?

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & & u_0 \\ & s_y & v_0 \\ & & 1 \end{bmatrix} [R_{3\times3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

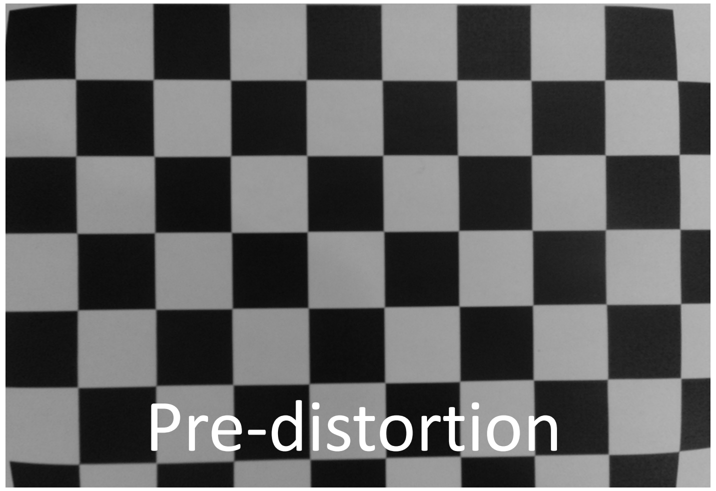
Generalizing intrinsics (2/2)

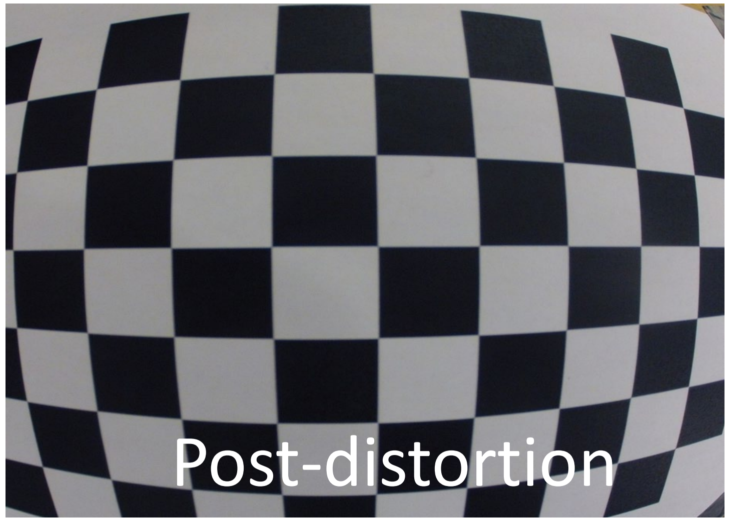
1. If pixels are not square?

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & & u_0 \\ & s_y & v_0 \\ & & 1 \end{bmatrix} [R_{3\times 3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

2. If “radial distortions”, then the intrinsics can no longer be represented as a linear operator any more

Parametrizing radial distortion in large field-of-view cameras





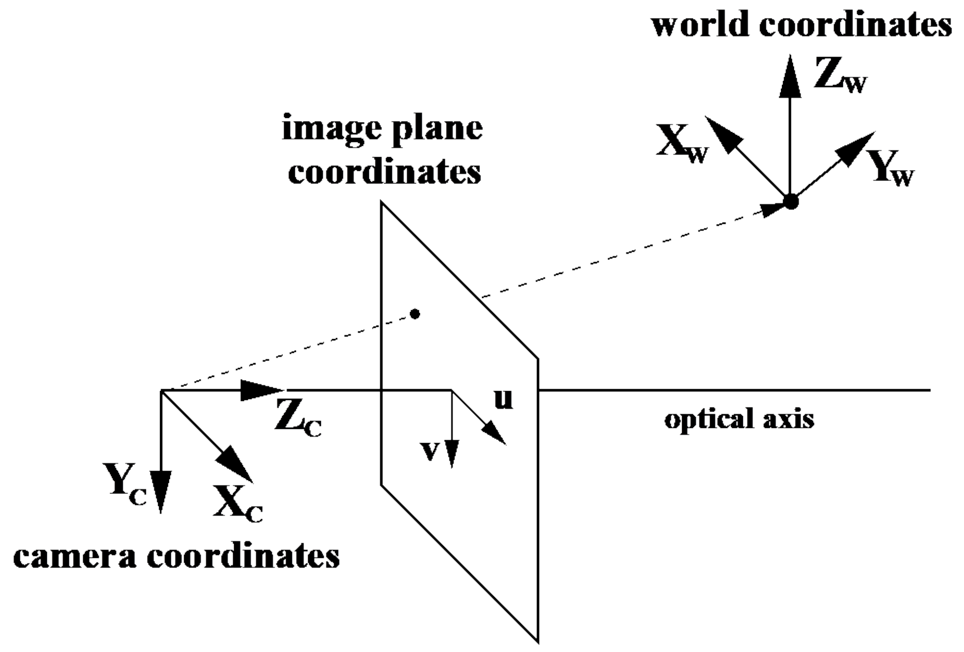
Then, correct for radial distortion:

$$u_{\text{pre-distortion}}(r) = u_{\text{post-distortion}}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots)$$
$$v_{\text{pre-distortion}}(r) = v_{\text{post-distortion}}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots)$$

where r is the distance from a (usually unknown) image center location (u_0, v_0).

Can choose the degree of the radial distortion to calibrate for. More => more accurate, but requires more images to fit well.

Putting the pieces together: Projection matrix P



$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix}}_{\text{camera 3D coords to pixels}} \underbrace{\begin{bmatrix} R_{3\times 3} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}}_{\text{Convert world to camera coordinates}} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

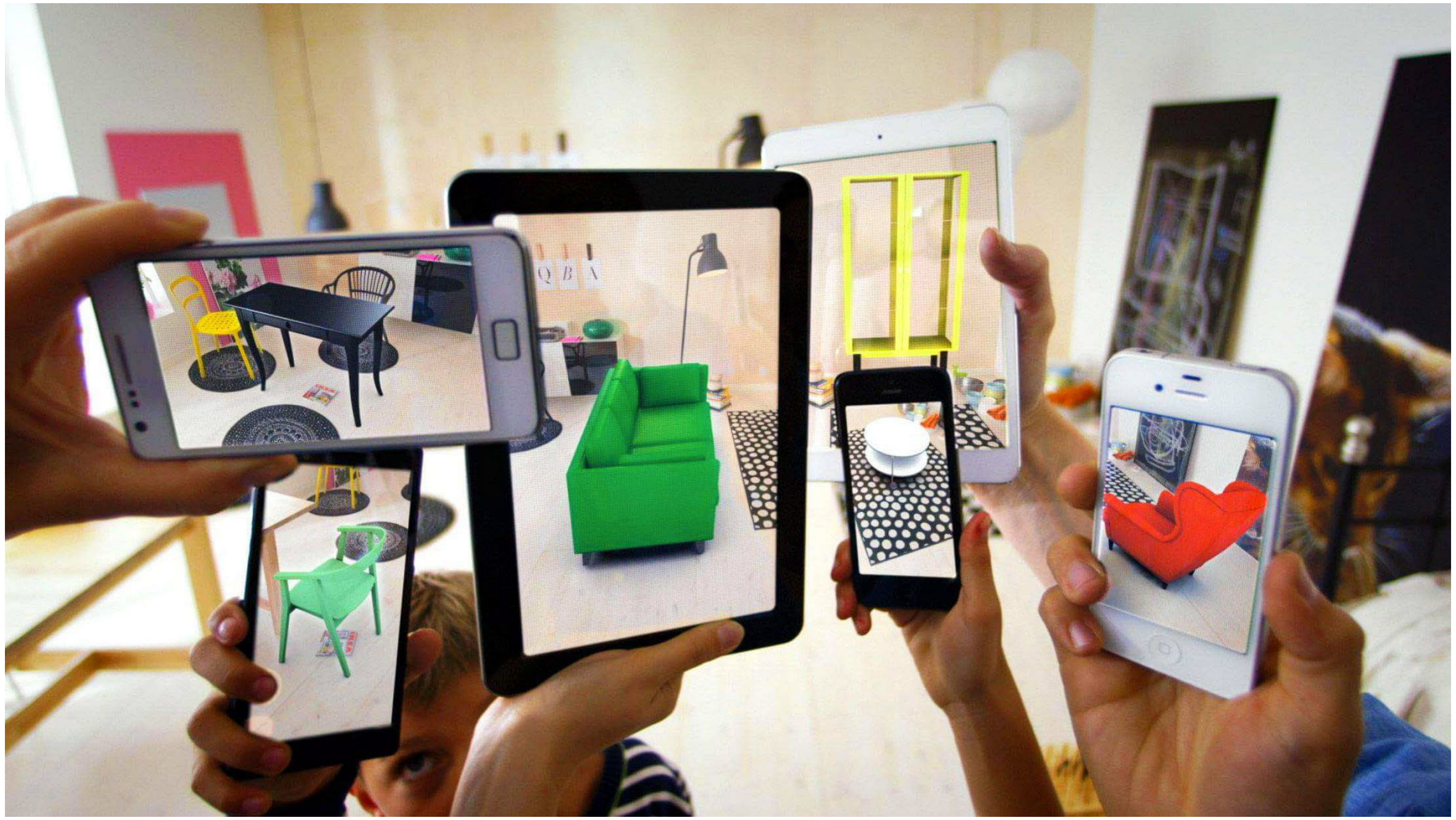
Perspective Projection in homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = P_{3\times 4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\underbrace{P_{3\times 4}}_{\text{“Camera projection matrix”}} = \underbrace{K_{3\times 3}}_{\text{“Intrinsics”}} \underbrace{[R_{3\times 3} | \boldsymbol{t}_{3\times 1}]}_{\text{“Extrinsics”}}$$

Applying extrinsics (and intrinsics) for 3D shape projection

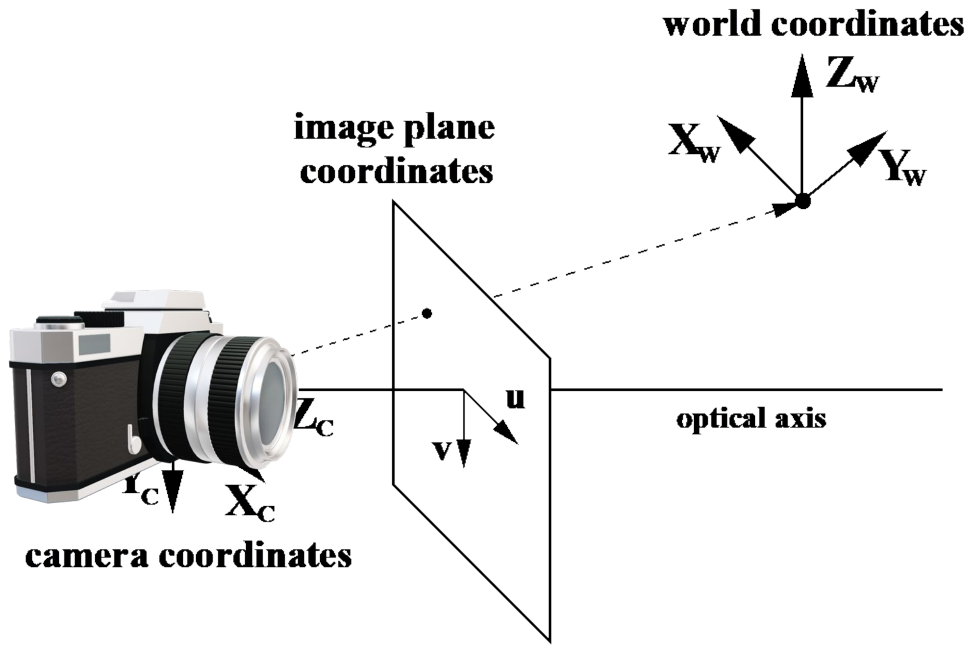
Can do AR-style projection of a 3D object onto the world plane once the full extrinsics and intrinsics are known!



IKEA App, image from WIRED.

Application of pose: projecting a solid shape into the world

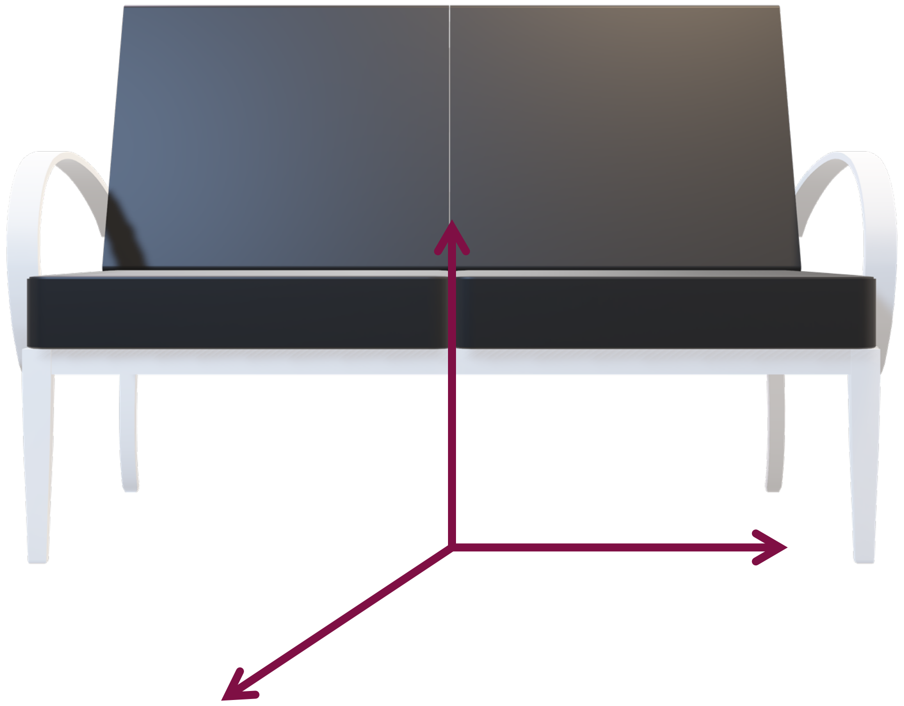
- Our normal projection equations tell us how world points in world coordinates project onto a camera, given camera pose (R, T) and intrinsics K



$$\boldsymbol{x} \sim K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

Application of pose: projecting a solid shape into the world

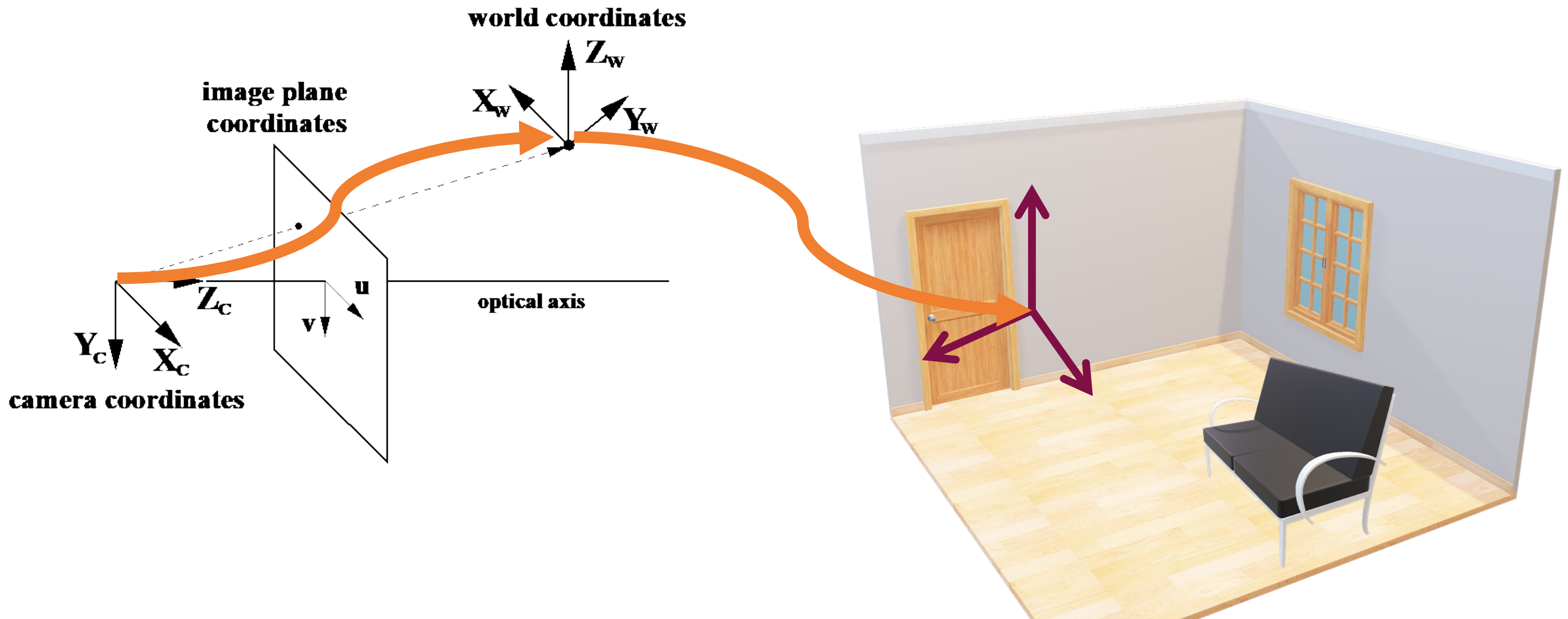
- Suppose the shape is expressed by the positions of points X_s in a "shape-coordinate system"


Coordinate system attached to the object

Application of pose: projecting a solid shape into the world

- First find R_{sw}, t_{sw} that convert object-centric coordinates X_s into world-centric coordinates $\boldsymbol{X}_w = R_{sw}\boldsymbol{X}_s + \boldsymbol{t}_{sw}$ to place the object at the right place in the world. (Think: what do R_{sw} and t_{sw} mean exactly?)

- Then just render the object points at $K[R|\boldsymbol{t}]\boldsymbol{X}_{\boldsymbol{w}}$



Projective Transformations

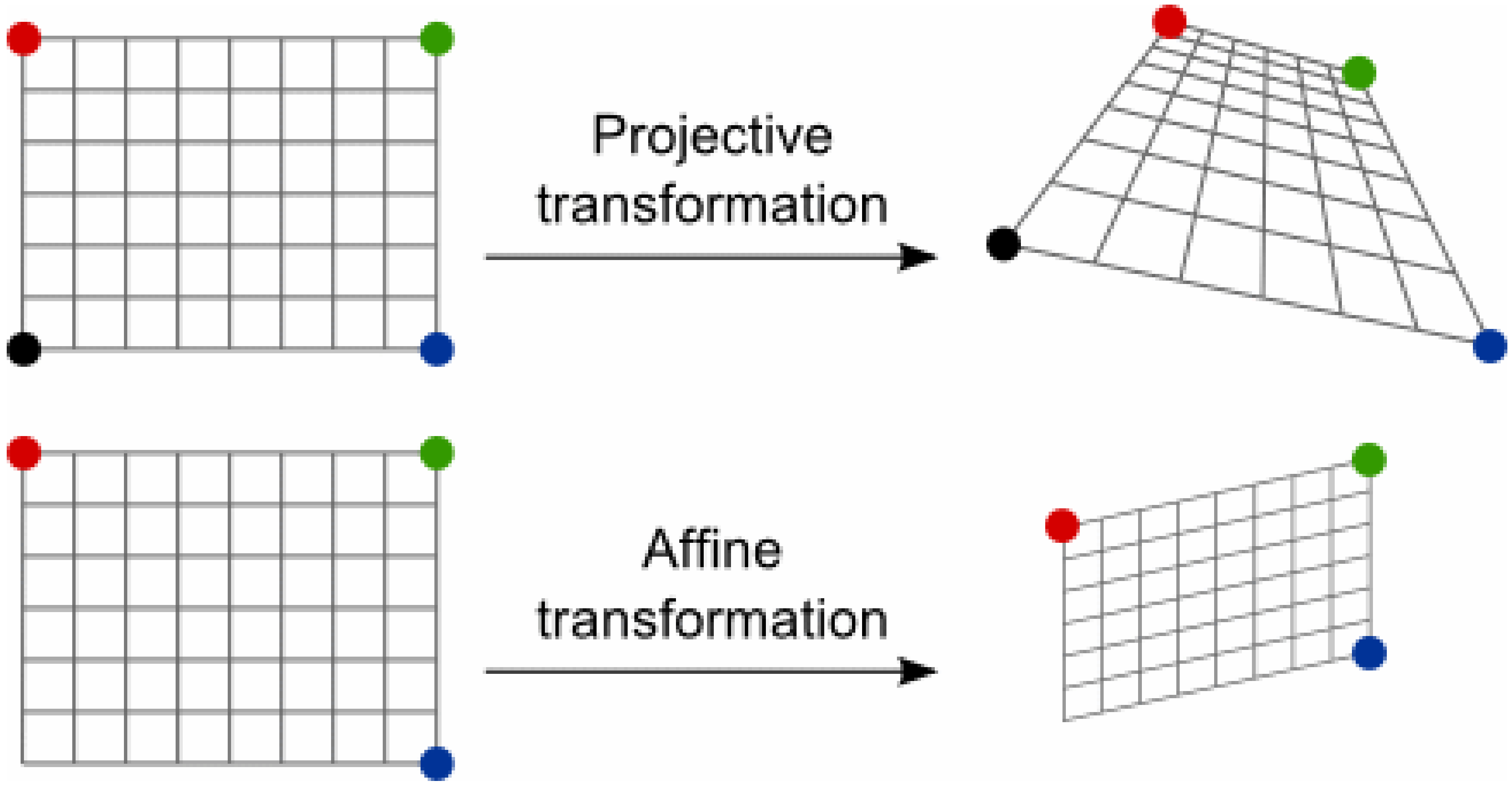
aka Collineations

aka Homographies

aka Projectivity

Example of Projective Transformation

Common notations: H
(Note that some books use A; however, we will avoid using A in this course, as A is commonly associated with Affine Transformations.)



Example of Projective Transformation

- A 2D point before H is represented as (X, Y) , after Projective transformation is (u, v) :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

or

$$\begin{aligned} u &= \frac{H_{11}X + H_{12}Y + H_{13}}{H_{31}X + H_{32}Y + H_{33}} \\ v &= \frac{H_{21}X + H_{22}Y + H_{23}}{H_{31}X + H_{32}Y + H_{33}} \end{aligned}$$

Projective Transformation = Homography = Collineation=Projectivity

Definition

A **projective transformation** is any invertible matrix transformation $\mathbb{P}^2 \to \mathbb{P}^2$.

A projective transformation H maps p to $p' \sim Hp$

Invertibility means that $\det(H) \neq 0$ and that there exists $\lambda \neq 0$ such that $\lambda p' = Hp$

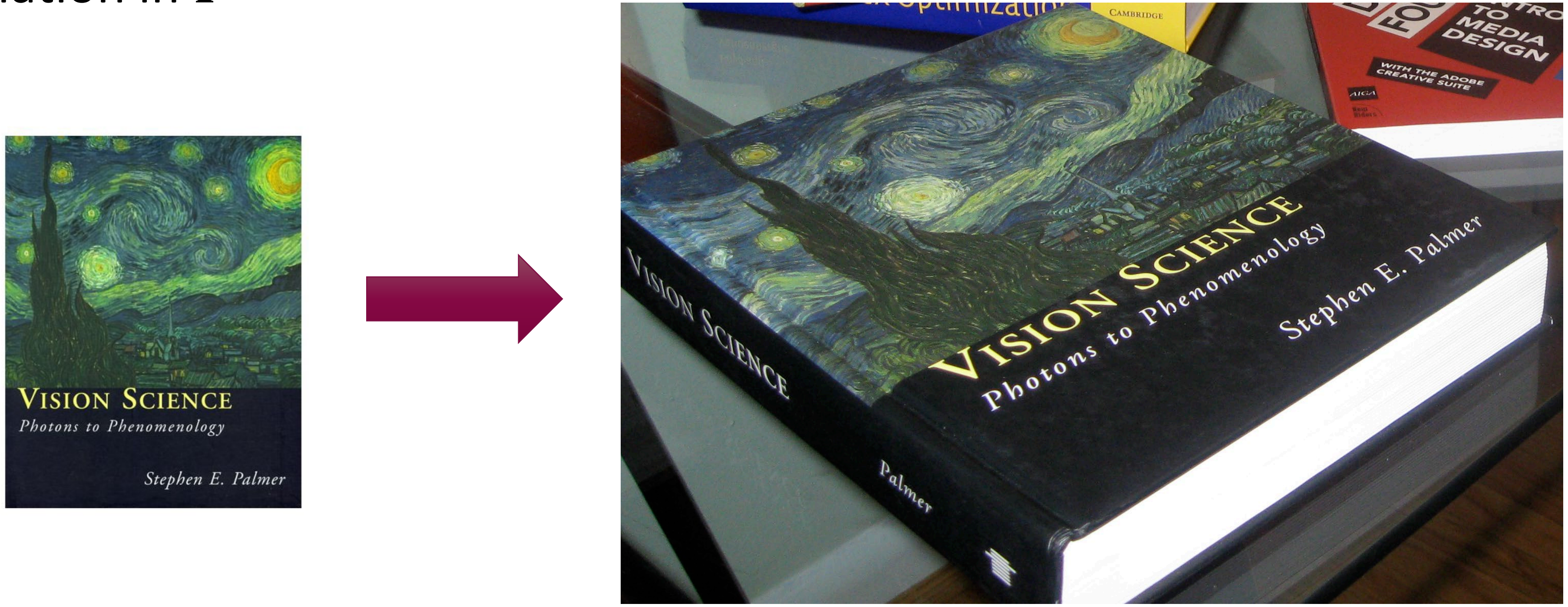
Observe that we will write either $p' \sim Hp$ or $\lambda p' = Hp$

Perspective Projection v.s. Projective Transformation

	Perspective Projection	Projective Transformation
Definition	A mapping from 3D space to a 2D plane (e.g., camera image)	A general mapping between projective space (e.g., P^2 to P^2)
Mathematical Formula	p′=K[R|T]P	p′=H·p
Input Space	R^3 (can also be P^3)	P^n (typically P^2 in this class)
Output Space	R^2 (can also be P^2)	P^n (typically P^2 in this class)
Applications	Image formulation, 3D rendering	Image registration, planar transformation, texture mapping

When Perspective Projection -> Projective Transformation?

A perspective camera projection of a plane (i.e., a camera image) is a projective transformation in $\mathbb{P}^2$



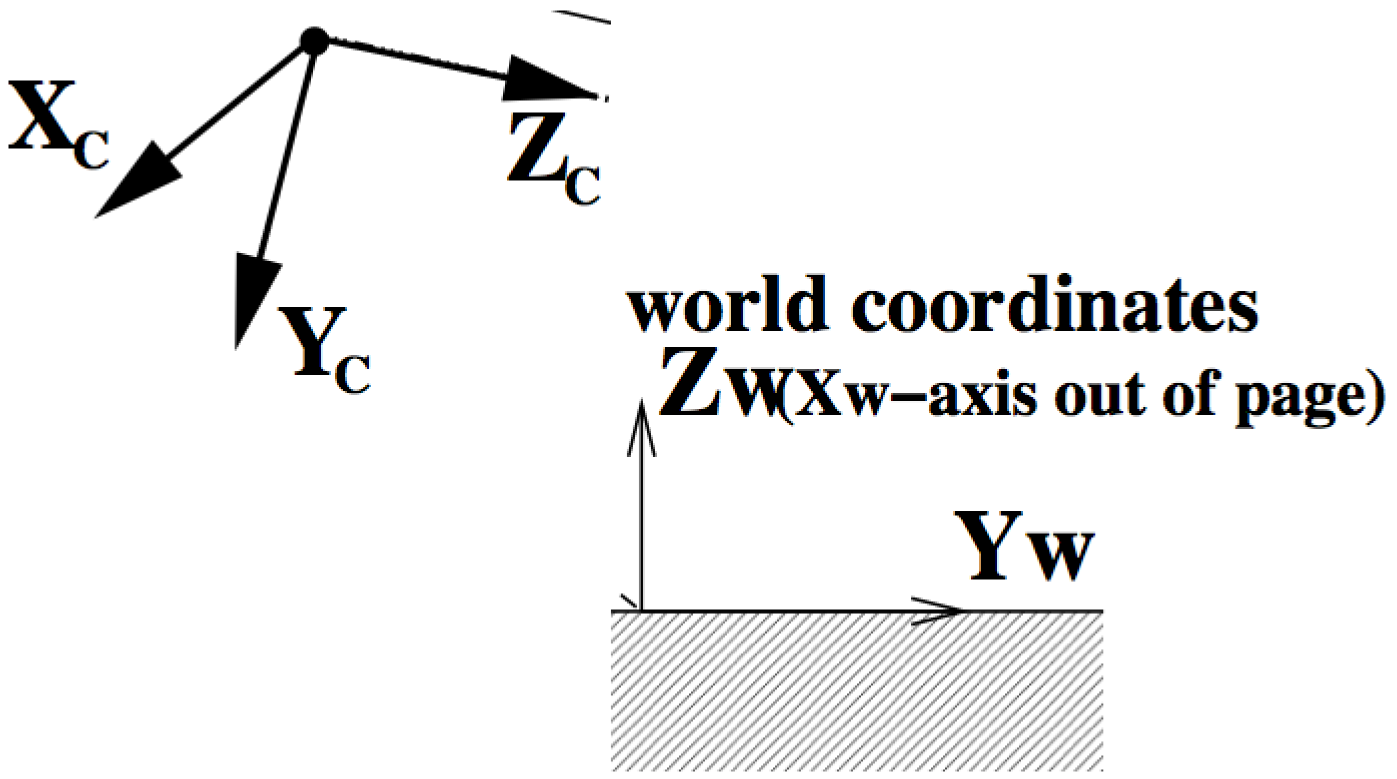
When Perspective Projection -> Projective Transformation?

- Can we show that the perspective camera projection from $\mathbb{P}^3 \rightarrow \mathbb{P}^2$ of a plane in the world is in fact a homography in $\mathbb{P}^2$ (i.e., projective transformation from $\mathbb{P}^2 \rightarrow \mathbb{P}^2$) when the world plane coordinates are expressed in $\mathbb{P}^2$?

- Remember:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \sim K_{3\times3}[R_{3\times3}|\boldsymbol{t}_{3\times1}]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Assume world plane $Z_w = 0$



When Perspective Projection -> Projective Transformation?

Recall the projection from world to camera

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim K \begin{pmatrix} r_1 & r_2 & r_3 & T \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

and assume that all points in the world lie in the ground plane $Z = 0$.

Pose From Homography

Recall the projection from world to camera

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim K \begin{pmatrix} r_1 & r_2 & r_3 & T \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

and assume that all points in the world lie in the ground plane $Z = 0$.

Then the transformation reads

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim \underbrace{K \begin{pmatrix} r_1 & r_2 & T \end{pmatrix}}_{} \begin{pmatrix} X \\ Y \\ W \end{pmatrix}$$

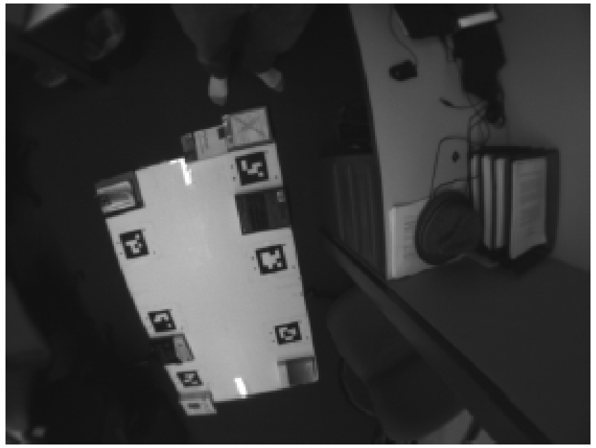
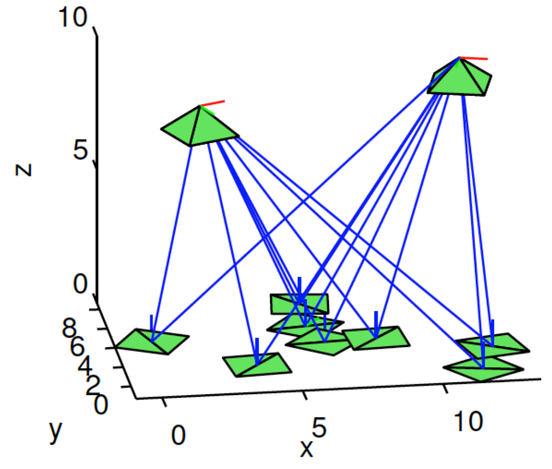
The planar homography
$H\colon \mathbb{P}^2 \rightarrow \mathbb{P}^2$

Computing the homography can tell us how the camera (and therefore, e.g. a robot attached to the camera) is oriented w.r.t. to a world plane! (assuming known K)

Q: Where do you get $\boldsymbol{r}_3$ from though?

A: $\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$

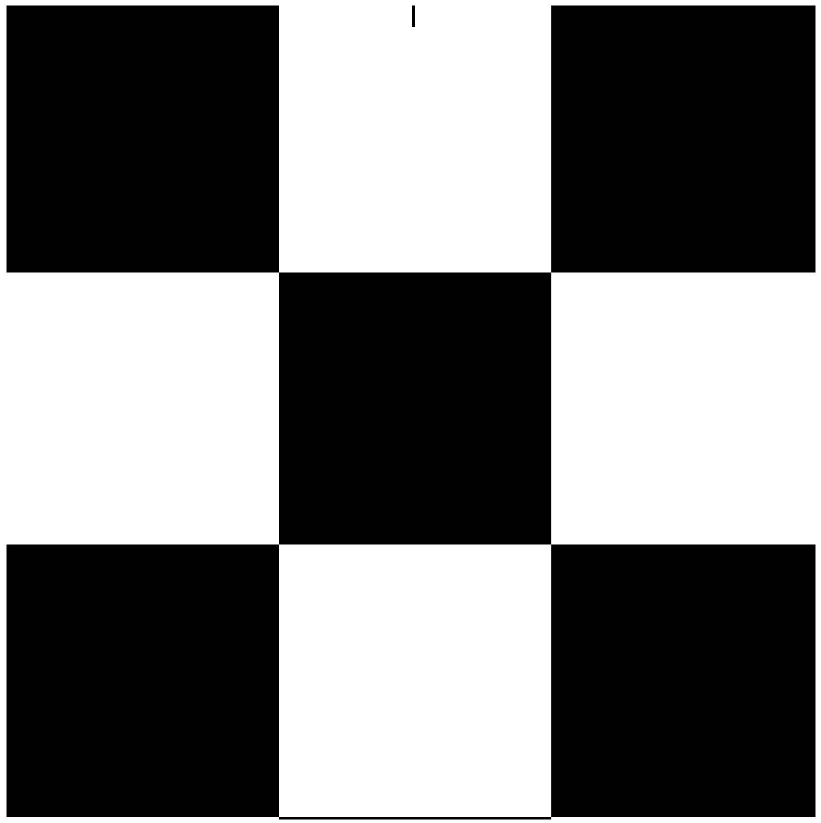
Localization w.r.t. known planes using homographies

Computing Homographies From 4 Point Correspondences

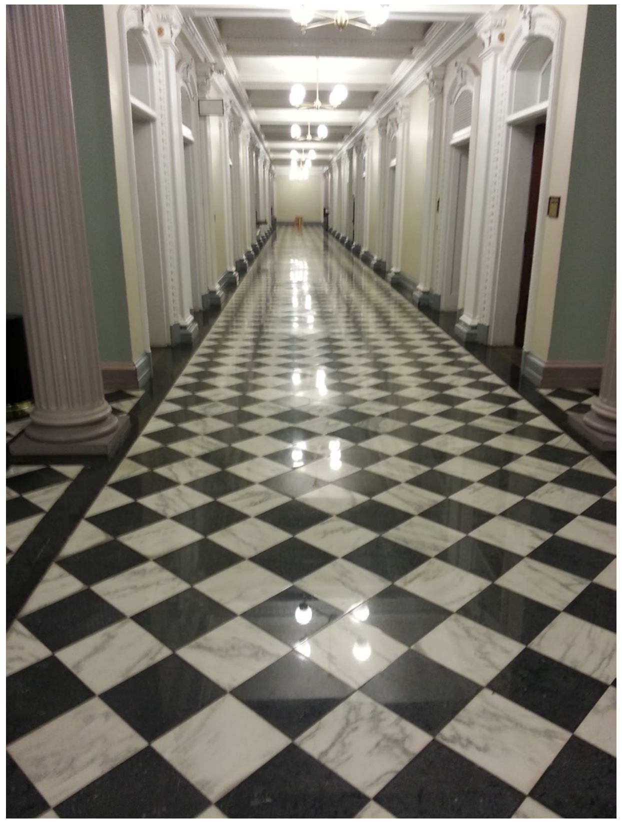
“4-point collineation”

How can we compute the projective transformation between a known pattern and its projection?



Floor tiles measured in [m]

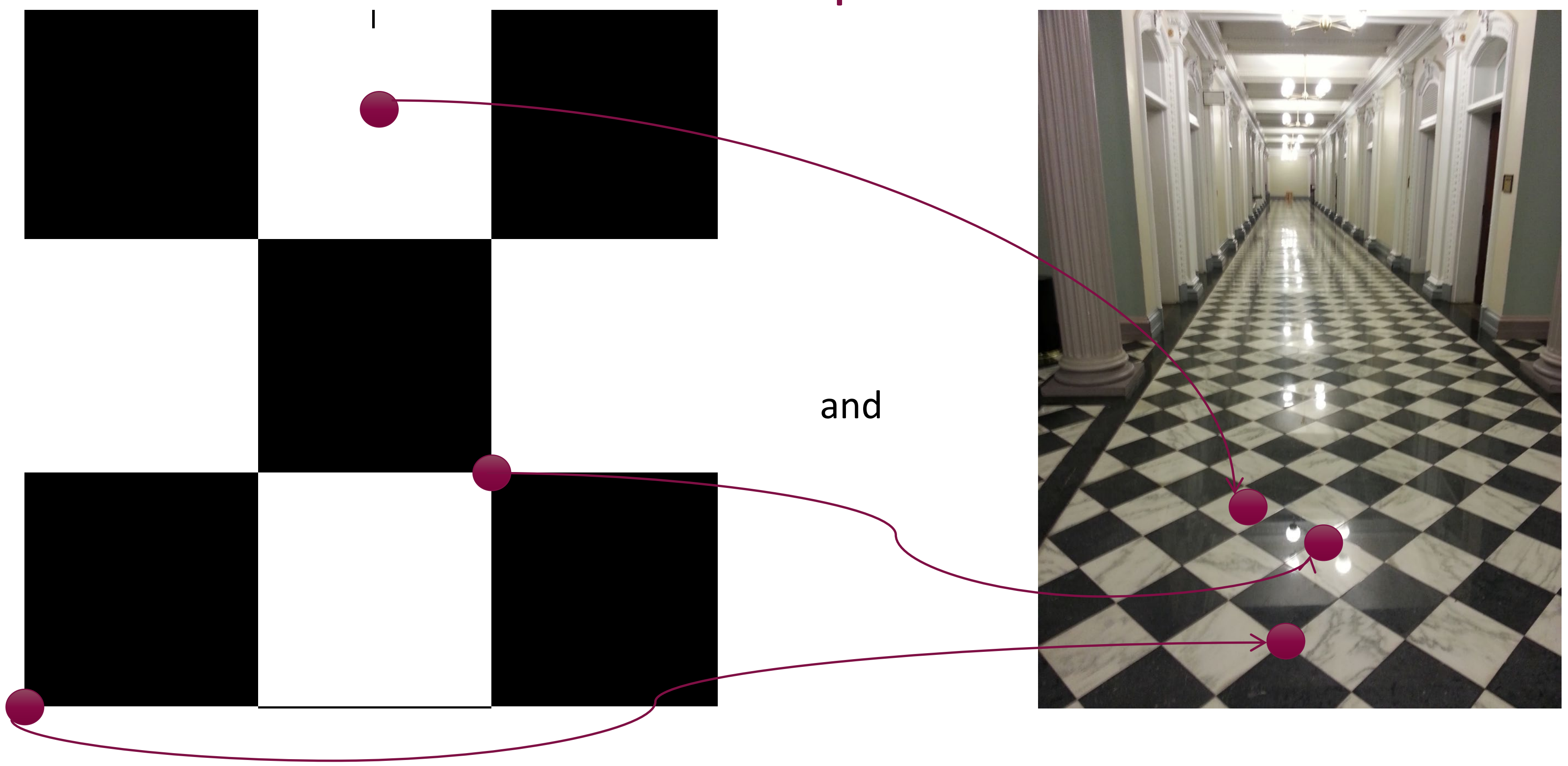
and



Points in pixel coordinates

The result of such a transformation would map any point in one plane to the corresponding point in the other

“correspondences”



Floor tiles measured in [m]

Points in pixel coordinates

How many unknowns are in a projective transformation H ?

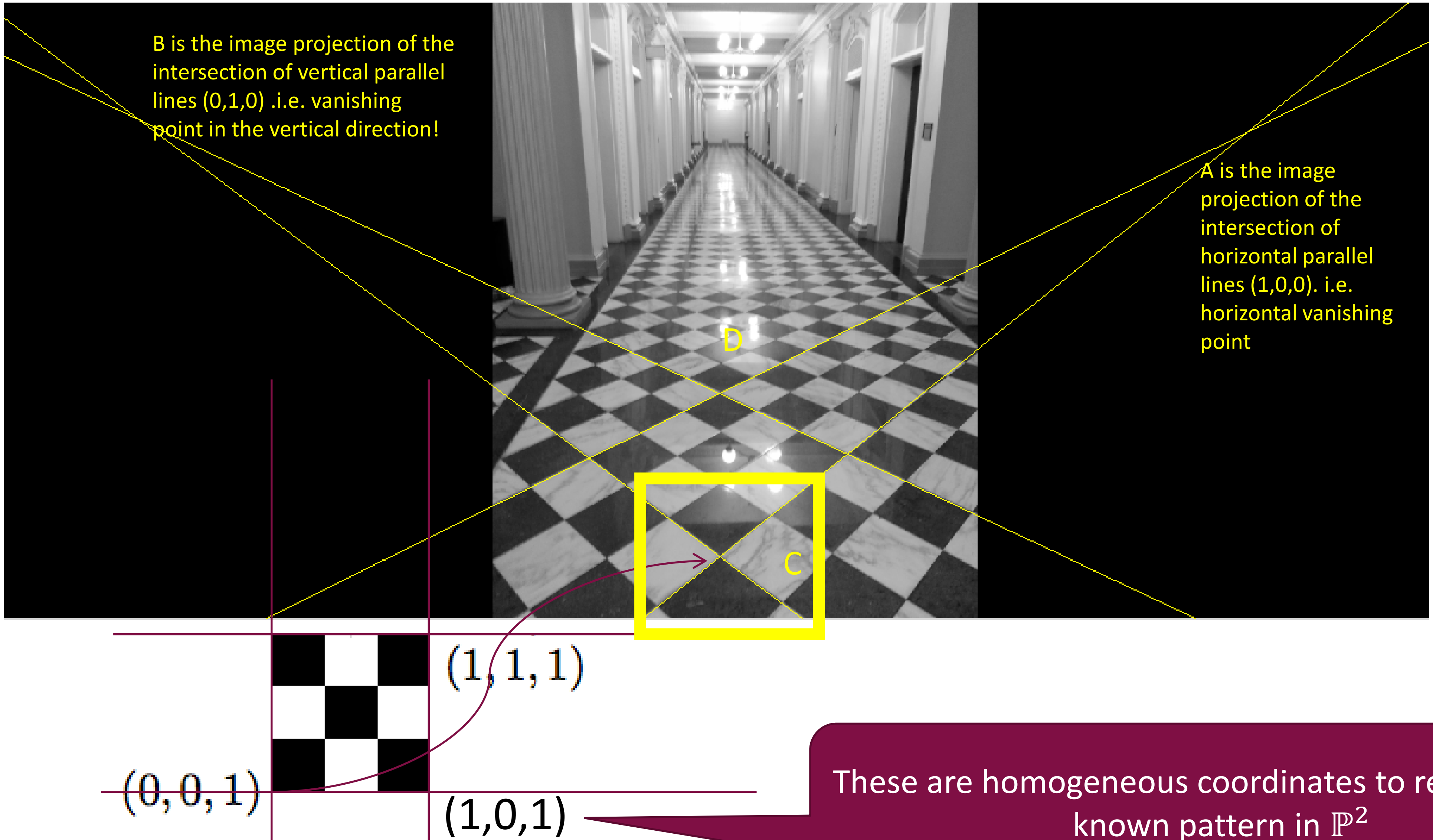
$(\mathbb{P}^2 \rightarrow \mathbb{P}^2)$

A projective transformation $\mu\ H$ is the same as H since they map to projectively equivalent points:

$$\mu\lambda p' = \mu\ Hp$$

We will be able to determine a projective transformation only up to a scale factor. Hence the 3x3 invertible matrix H will have only EIGHT independent unknowns.

How can we compute the projective transformation between a known pattern and its projection?

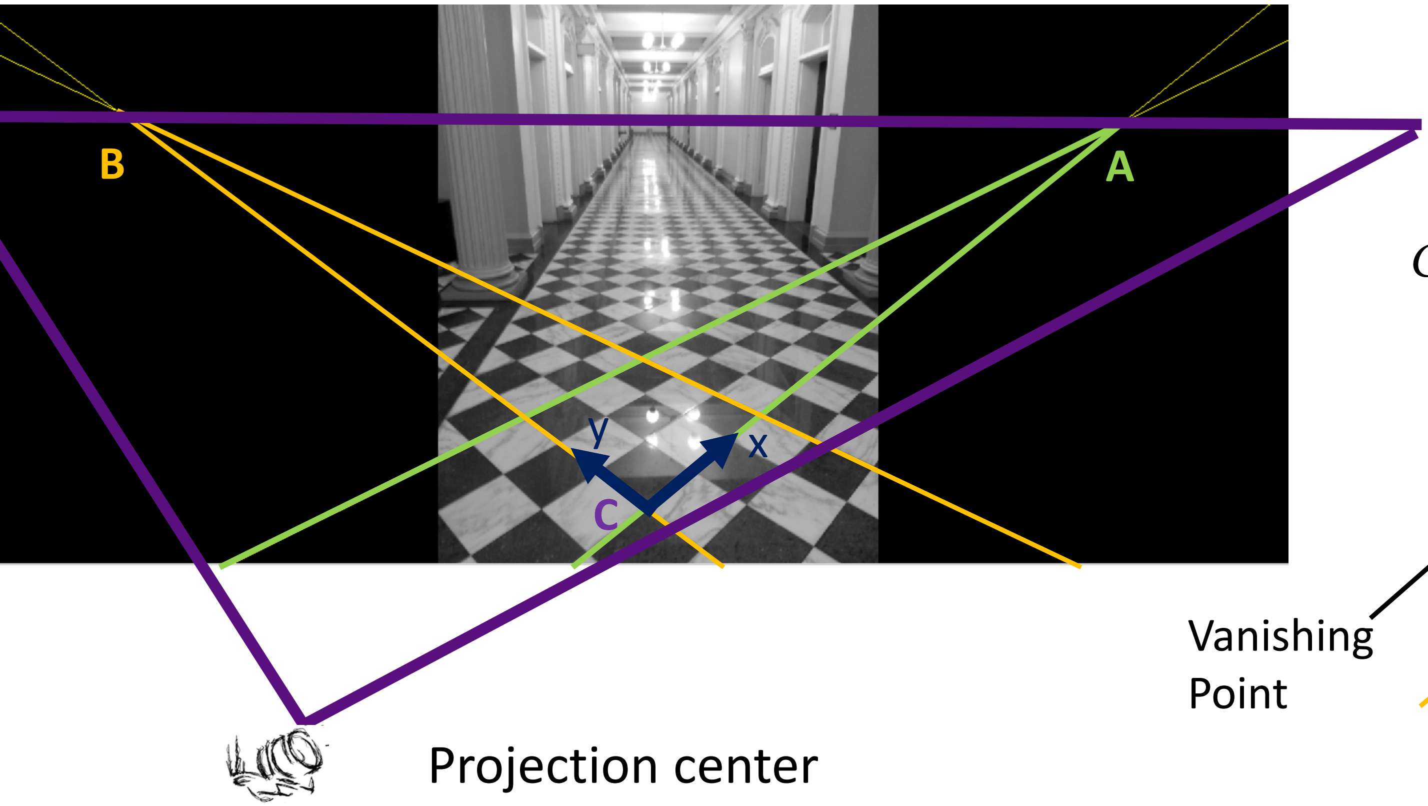


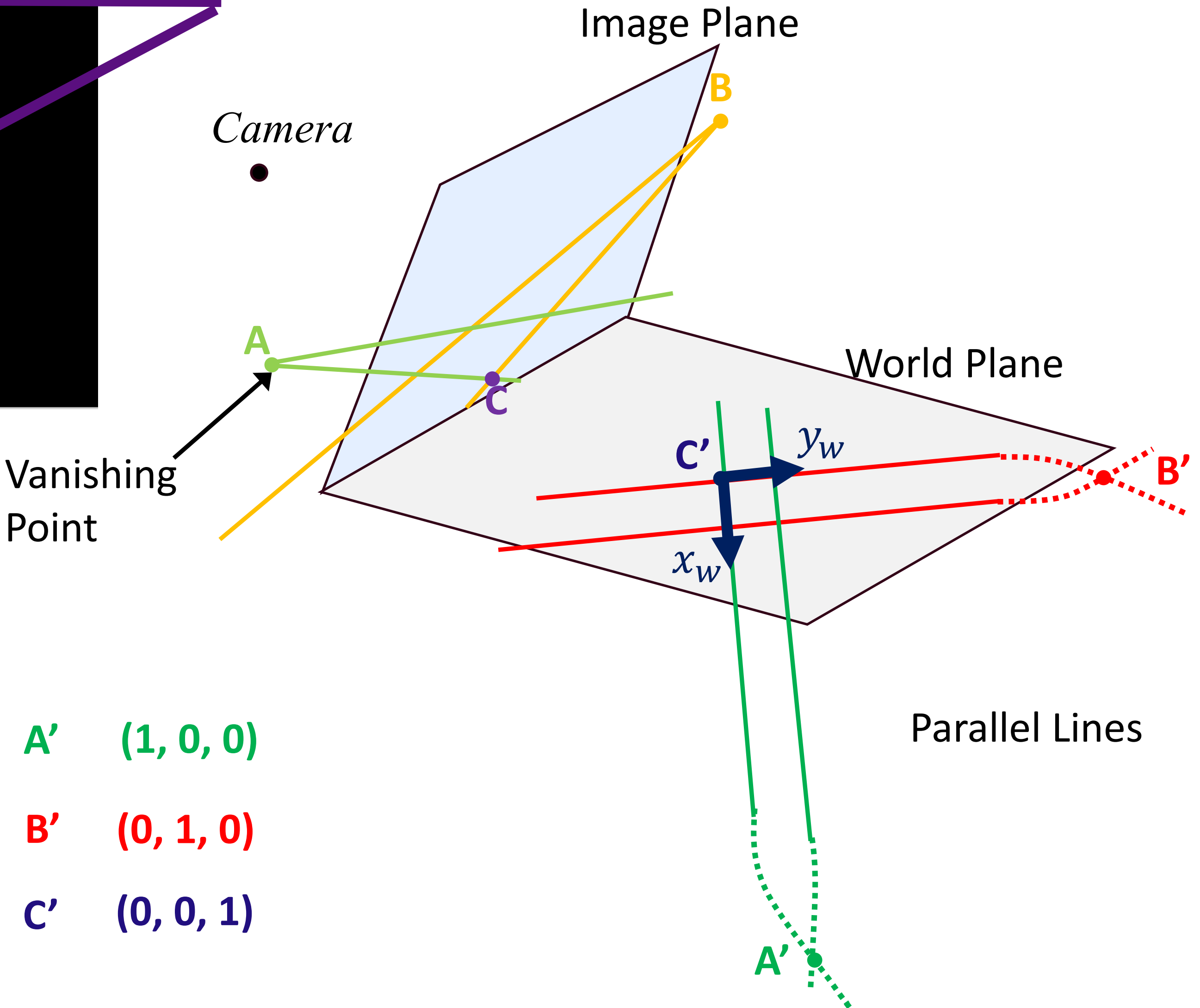
B is the image projection of the intersection of vertical parallel lines (0,1,0) .i.e. vanishing point in the vertical direction!

A is the image projection of the intersection of horizontal parallel lines (1,0,0). i.e. horizontal vanishing point

These are homogeneous coordinates to represent the known pattern in $\mathbb{P}^2$

Cont.





Correspondences:
($P^2 \rightarrow P^2$)

A	a	$\longleftrightarrow$	A'	(1, 0, 0)
B	b	$\longleftrightarrow$	B'	(0, 1, 0)
C	c	$\longleftrightarrow$	C'	(0, 0, 1)

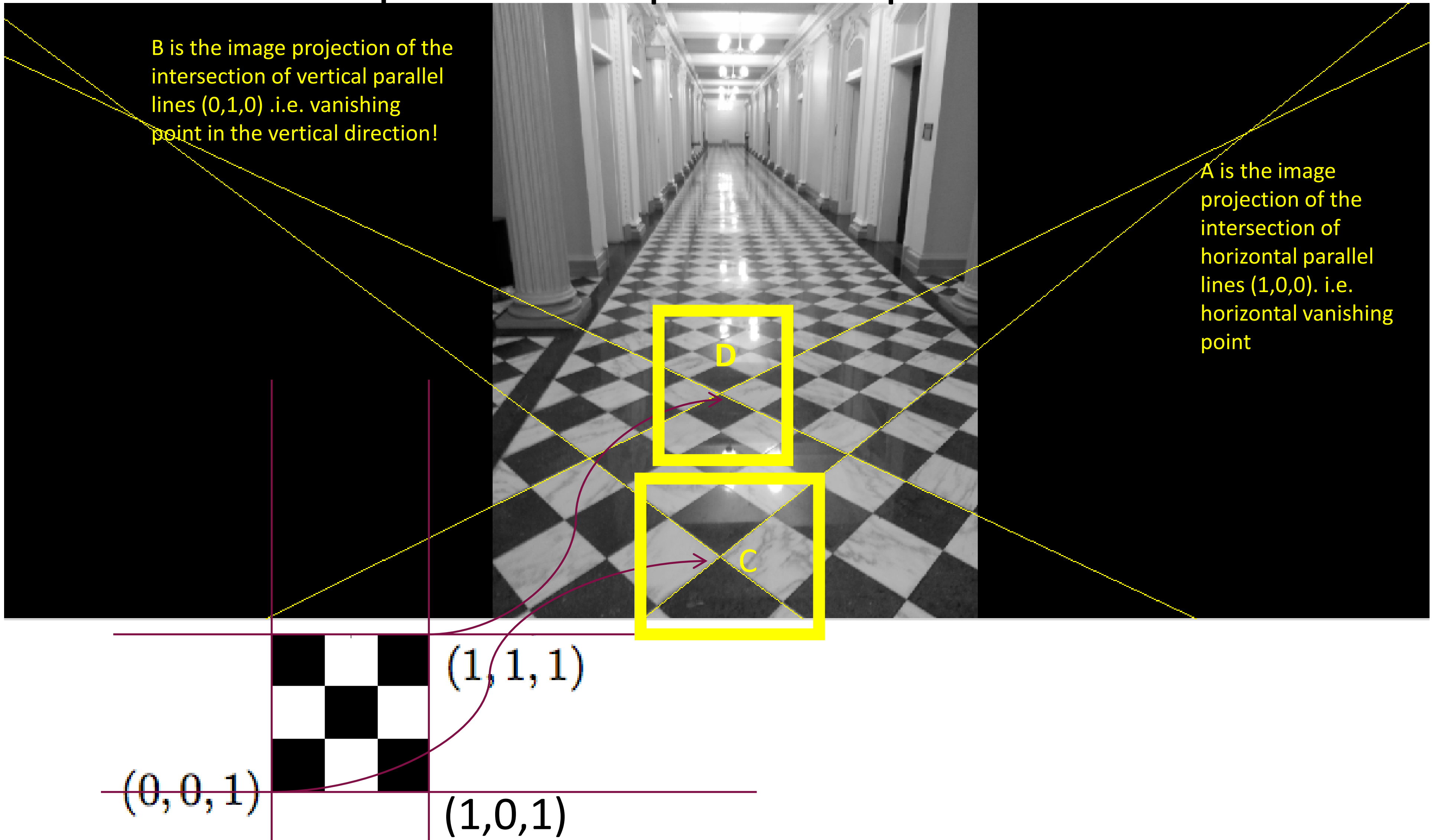
Assume that a mapping H maps the three points $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ to the non-collinear points A,B,C

with coordinate vectors a, b and $c \in \mathbb{P}^2$. Then the following is a possible projective transformation:

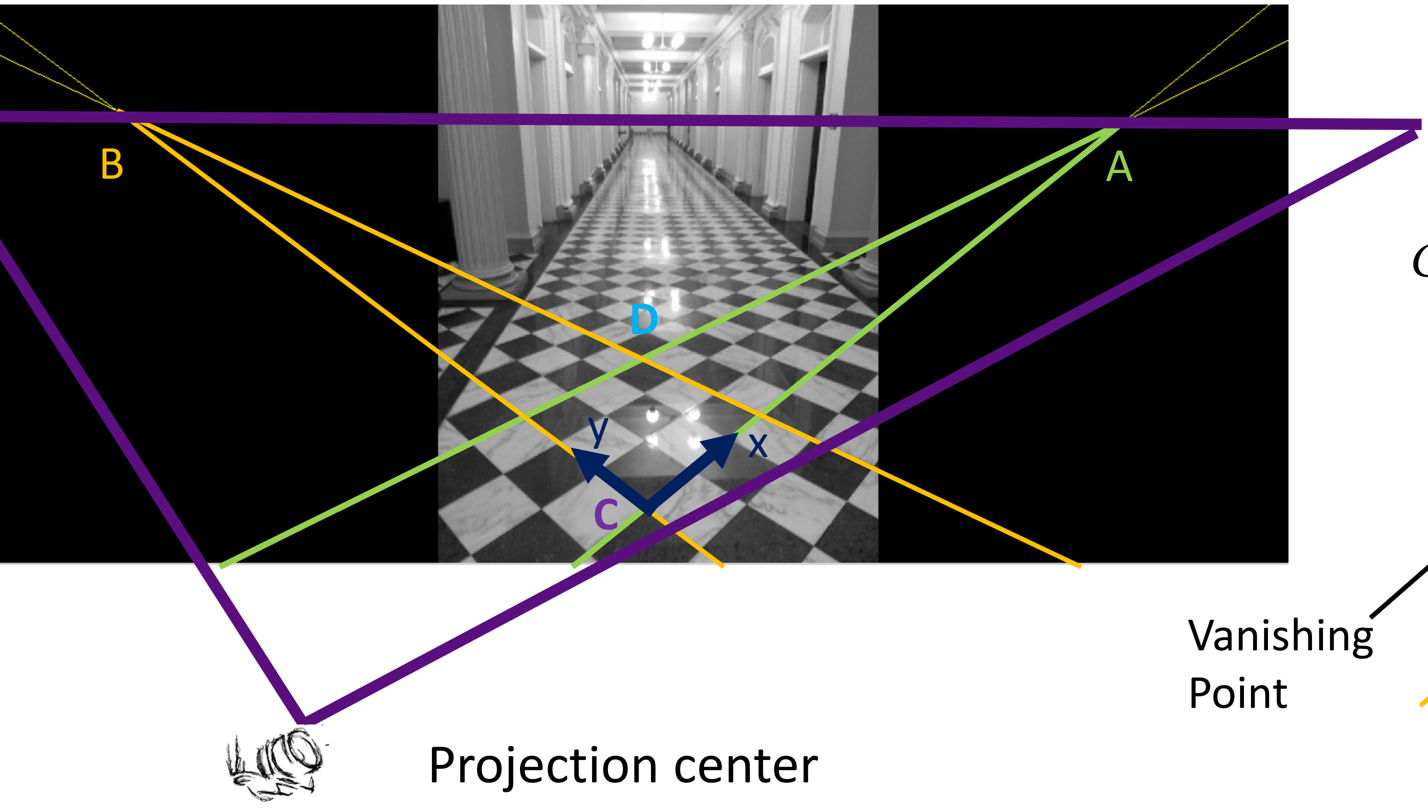
$$\begin{array}{ccc} A & B & C \\ \left(\boxed{a}\right. & \boxed{b} & \left.\boxed{c}\right) \end{array} \sim H_{3\times3} \begin{array}{ccc} A' & B' & C' \\ \left(\boxed{\begin{matrix}1\\0\\0\end{matrix}}\right. & \boxed{\begin{matrix}0\\1\\0\end{matrix}} & \left.\boxed{\begin{matrix}0\\0\\1\end{matrix}}\right) \end{array}$$

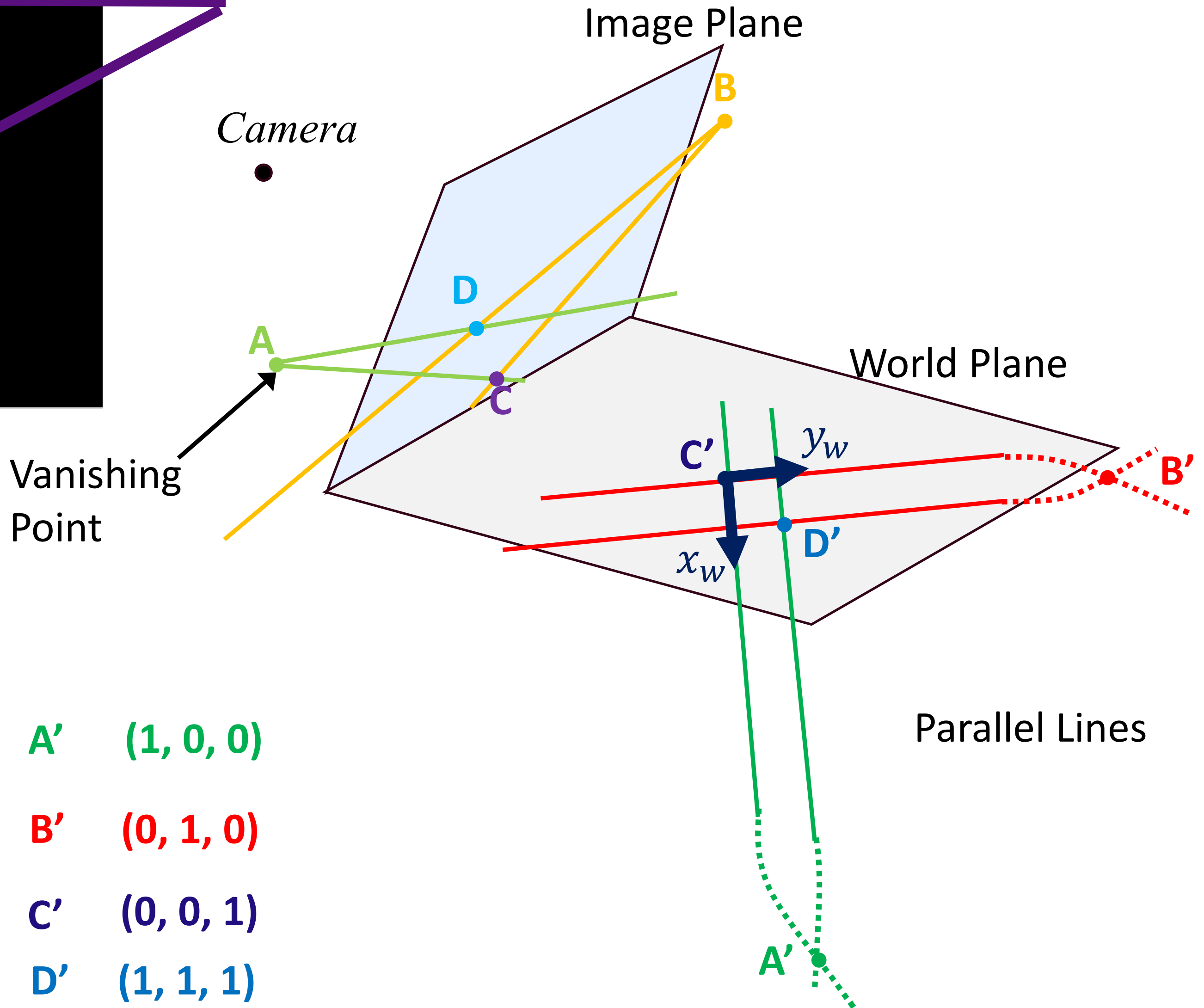
Solution: Introduce a 4th point correspondence D

Note: makes sense, because after all, H has 8 degrees of freedom, and each 2D point correspondence pins down 2DOF.


B is the image projection of the intersection of vertical parallel lines (0,1,0) .i.e. vanishing point in the vertical direction!
A is the image projection of the intersection of horizontal parallel lines (1,0,0). i.e. horizontal vanishing point
D
C
(1,1,1)
(0,0,1)
(1,0,1)

Cont.





Correspondences:
($P^2 \rightarrow P^2$)

A	a	⟷	A'	(1, 0, 0)
B	b	⟷	B'	(0, 1, 0)
C	c	⟷	C'	(0, 0, 1)
D	d	⟷	D'	(1, 1, 1)

Let us assume that the same H maps $(1,1,1)$ to the point d. Then, the following should hold:

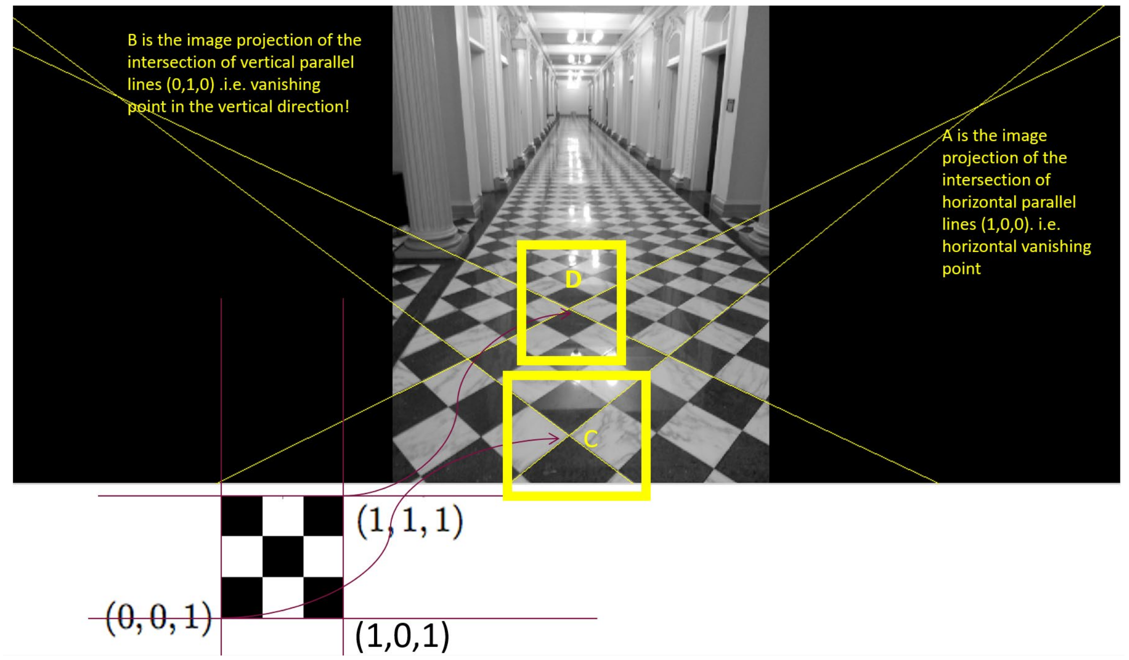
$$d \sim \underbrace{\begin{pmatrix} \alpha a & \beta b & \gamma c \end{pmatrix}}_{H_{3\times 3}} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

hence

$$d \sim \alpha a + \beta b + \gamma c. \quad \text{or} \quad \lambda d = \alpha a + \beta b + \gamma c$$

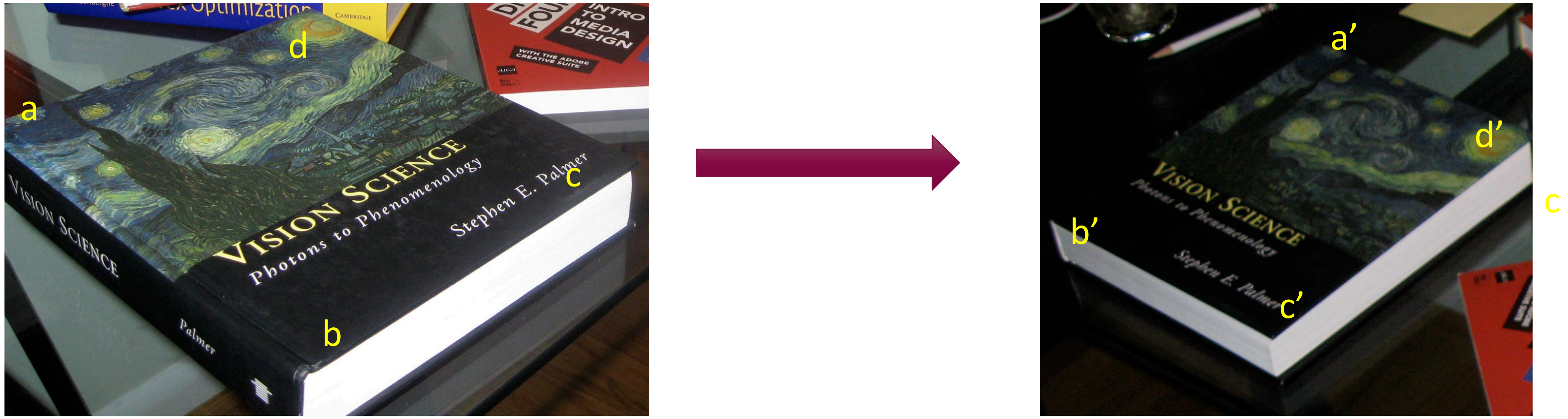
Because a, b, c are not collinear, there exist unique $\alpha/\lambda, \beta/\lambda, \gamma/\lambda$ for writing this linear combination.

Four points, no three of them collinear, suffice to unambiguously recover a homography


B is the image projection of the intersection of vertical parallel lines (0,1,0) .i.e. vanishing point in the vertical direction!
A is the image projection of the intersection of horizontal parallel lines (1,0,0). i.e. horizontal vanishing point
D
C
(1,1,1)
(0,0,1)
(1,0,1)

Choosing the points to be the horizontal and vertical vanishing points (1,0,0), (0,1,0) plus origin (0,0,1) and the diagonal (1,1,1) is particularly “nice” especially if you have a square to start from, but really, any four non-collinear points will do.
(coming up next)

What happens when the original set of points is not a square?



Find projective transformation mapping $(a, b, c, d) \rightarrow (a', b', c', d')$:

To determine this mapping we go through the four canonical points.

We find the mapping from $(1, 0, 0), etc$ to (a, b, c, d) and we call it T:

$$a \sim T(1, 0, 0)^T, etc$$

We find the mapping from $(1, 0, 0), etc$ to (a', b', c', d') and we call it T':

$$a' \sim T'(1, 0, 0)^T, etc$$

Then, back-substituing $(1, 0, 0)^T \sim T^{-1}a, etc$ we obtain that

$$a' = T'T^{-1}a, etc$$

General Form for Computing homographies

$$\mathbf{x}' \sim H\mathbf{x}$$

$$\lambda \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{aligned} \lambda x' &= h_{11}x + h_{12}y + h_{13} \\ \lambda y' &= h_{21}x + h_{22}y + h_{23} \\ \lambda &= h_{31}x + h_{32}y + h_{33} \end{aligned}$$

Converting to a linear system of equations

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$
$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

$$\Rightarrow$$

$$\begin{aligned} -h_{11}x - h_{12}y - h_{13} + h_{31}xx' + h_{32}yx' + h_{33}x' &= 0 \\ -h_{21}x - h_{22}y - h_{23} + h_{31}xy' + h_{32}yy' + h_{33}y' &= 0 \end{aligned}$$

$$\Downarrow$$

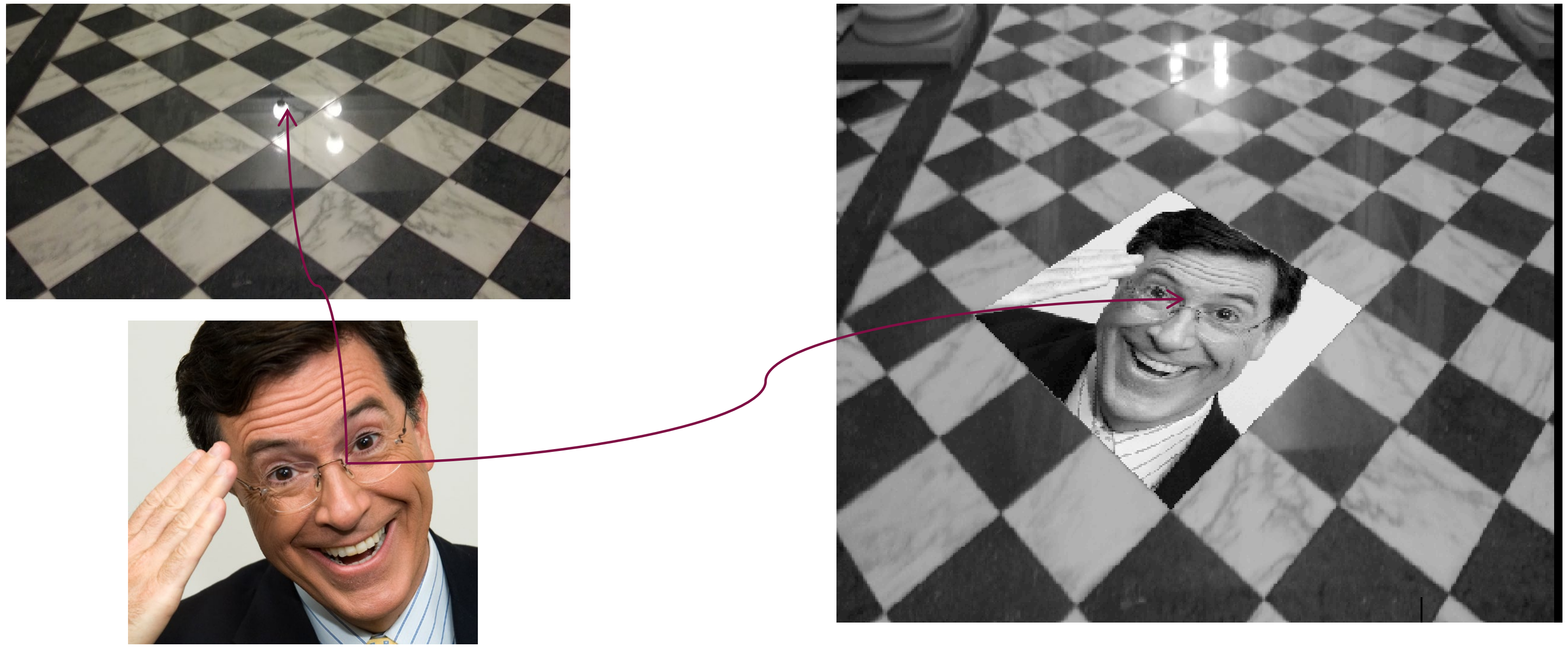
$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} h = 0$$

Two linear equations for each point correspondence!

$$a_x = \begin{pmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \end{pmatrix}$$
$$a_y = \begin{pmatrix} 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{pmatrix}$$
$$h = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} & h_{33} \end{pmatrix}^T$$

Homography -> Virtual Billboards

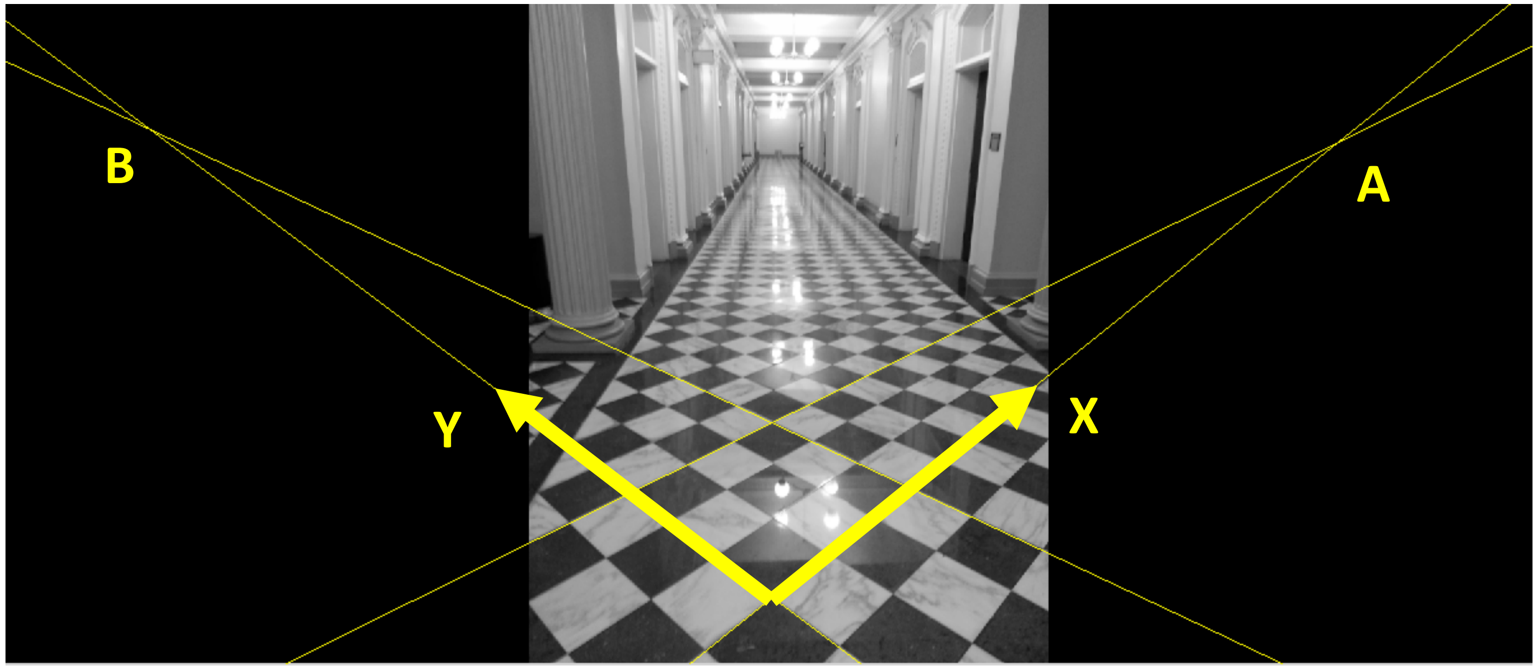
- For virtual billboards, we just treat the desired billboard pattern as the world pattern.
- Recall that the homography is a mapping from world to image pixel coordinates. So, just the homography can directly find the image pixel coordinate corresponding to every image in the world pattern.



Homography -> Vanishing Points, Horizon

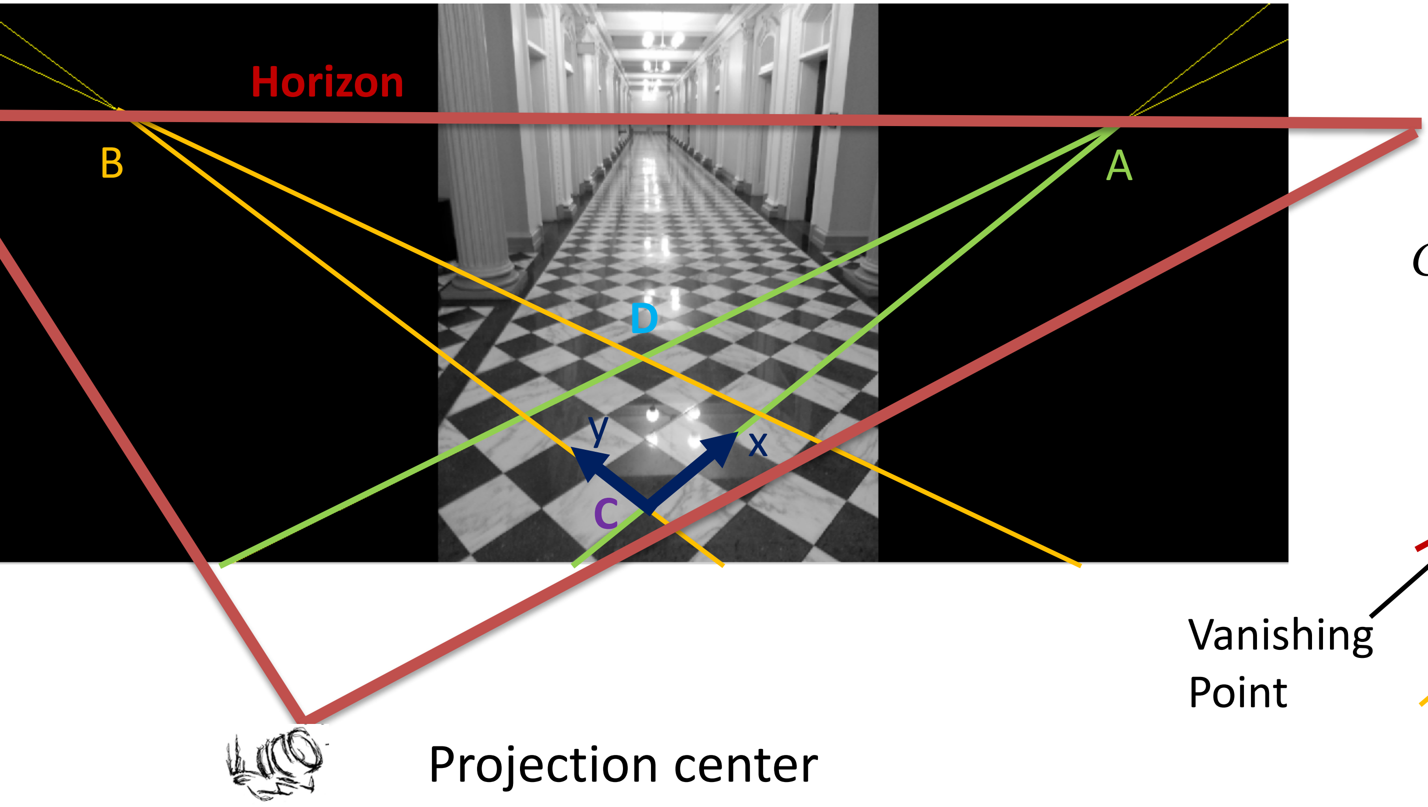
Homography columns are vanishing points

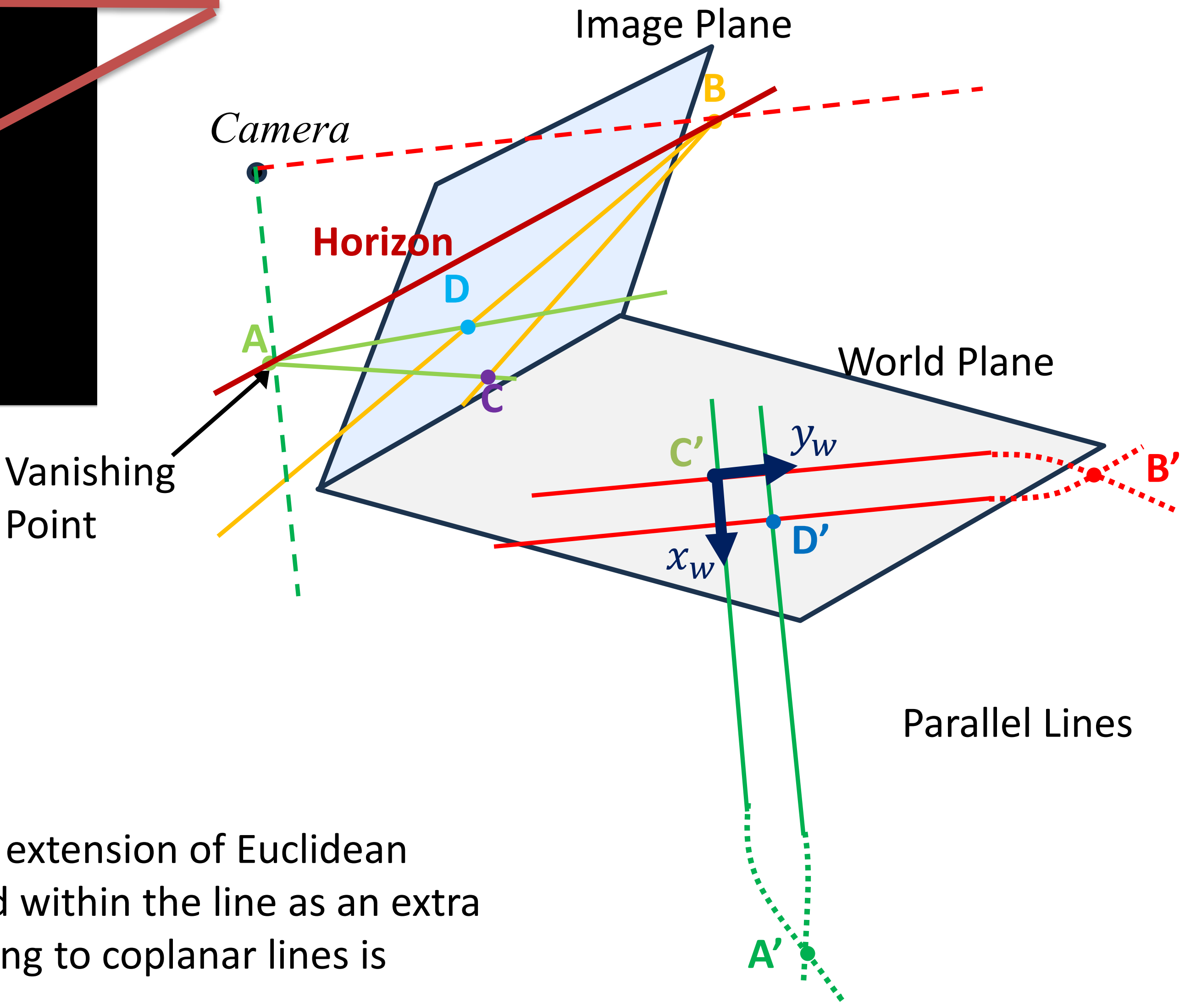
If $H = \begin{pmatrix} h_1 & h_2 & h_3 \end{pmatrix}$ then $h_1 \sim A$ and $h_2 \sim B$.



So the first two columns are two Orthogonal vanishing points

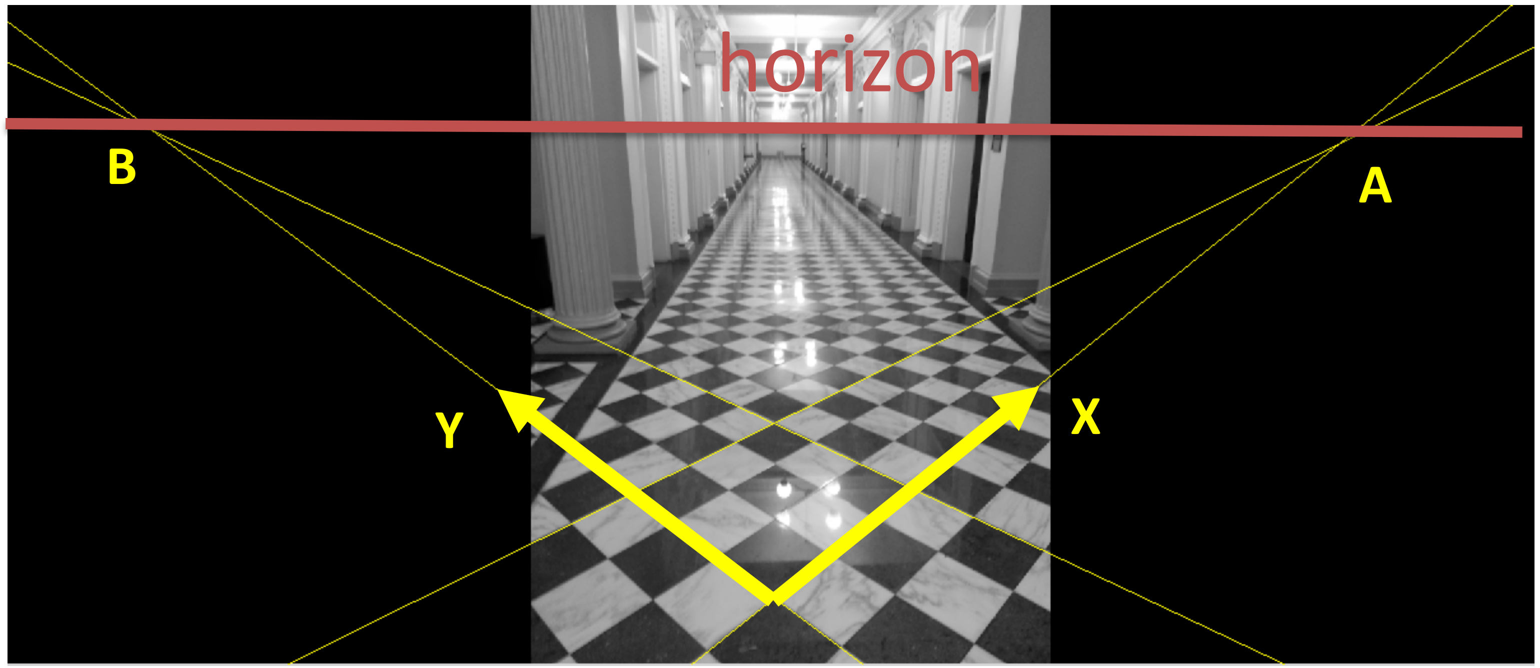
Horizon





In essence, a projective geometry may be thought of as an extension of Euclidean geometry in which the "direction" of each line is subsumed within the line as an extra "point", and in which a "horizon" of directions corresponding to coplanar lines is regarded as a "line"

Equation of horizon: $(h_1 \times h_2)^T \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$



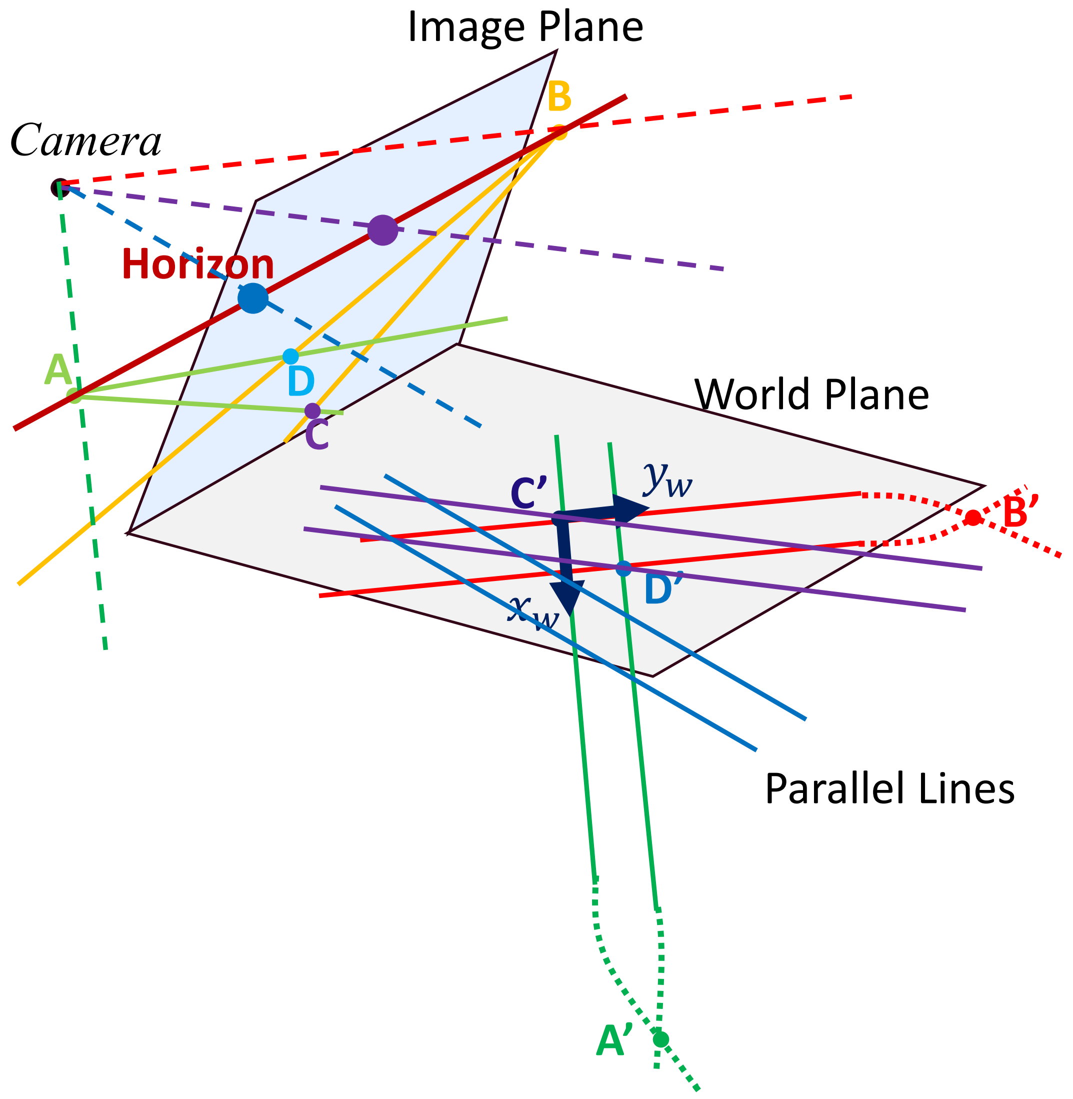
We will encounter another way to derive this equation of the horizon very soon.

Projecting the line at infinity to compute the horizon

Points at infinity in the world plane look like $(X, Y, W = 0)^T$

The "line" connecting them is $W = 0$, the "line at infinity". The image of this line is the horizon, which contains all vanishing points. Expressed in world plane $\mathbb{P}^2$, this line's coefficients are $(0,0,1)^T$.

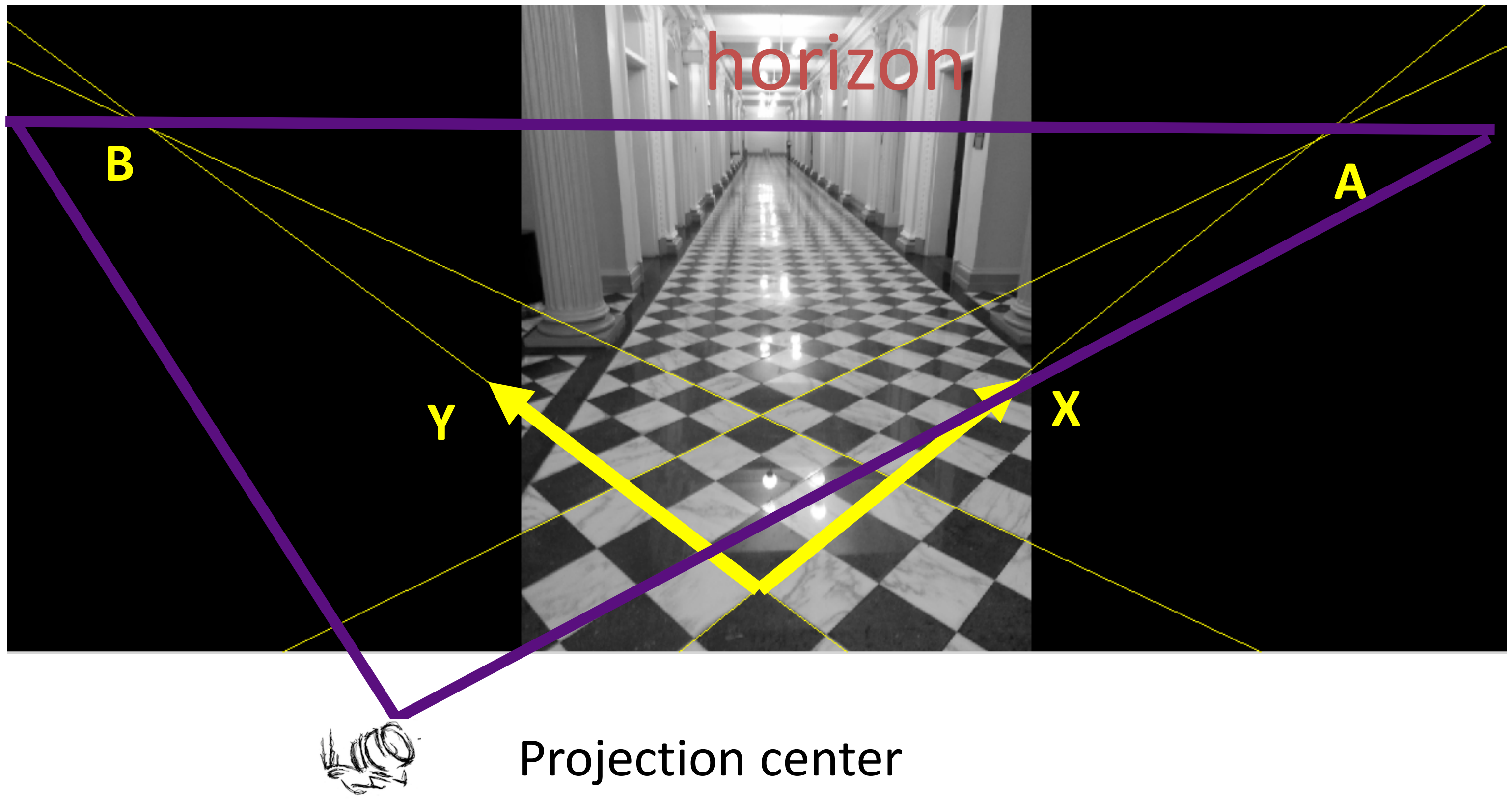
So if we could find the projection of this line, we could find the horizon



Summary

Vanishing rays/planes through the camera center are parallel to the world lines/planes

So, the horizon plane is parallel to the ground plane and hence $h_1 \times h_2$ is the normal to the ground plane!



Projection center

Horizon plane = Vanishing plane = Viewing plane

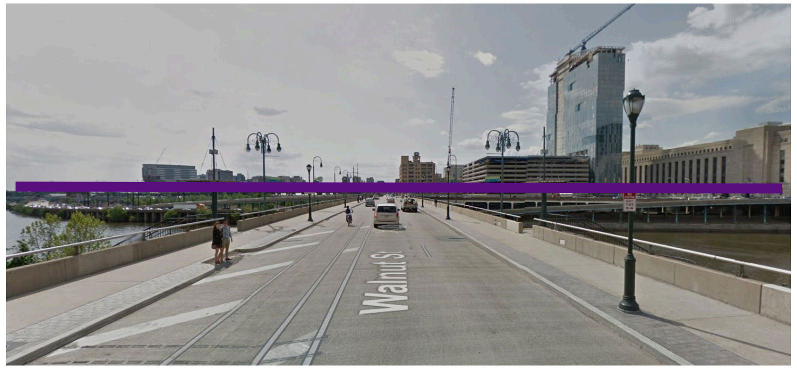
World plane // vanishing plane

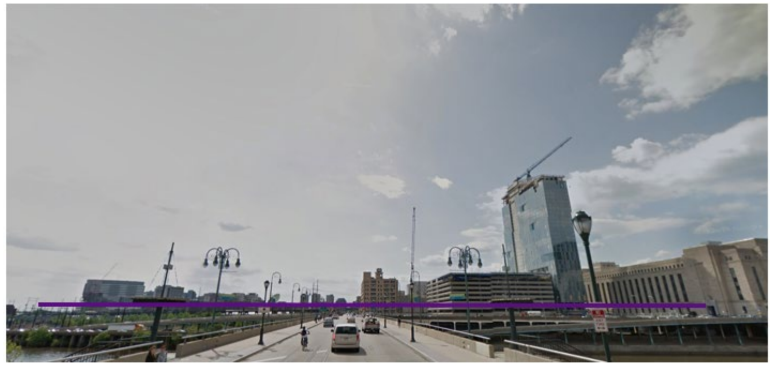
World plane = Ground plane in this case

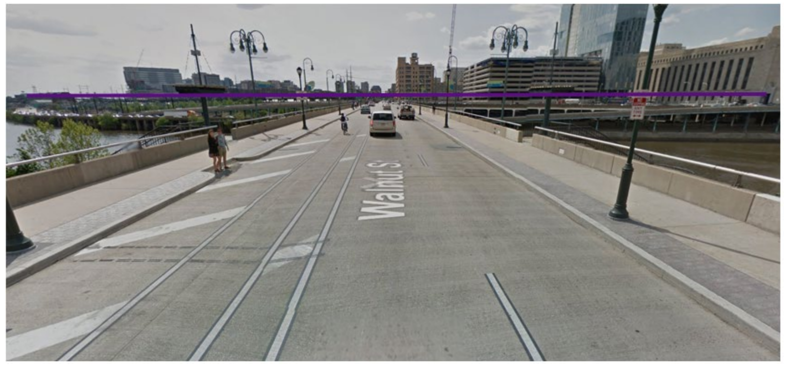
Using the horizon to orient the camera

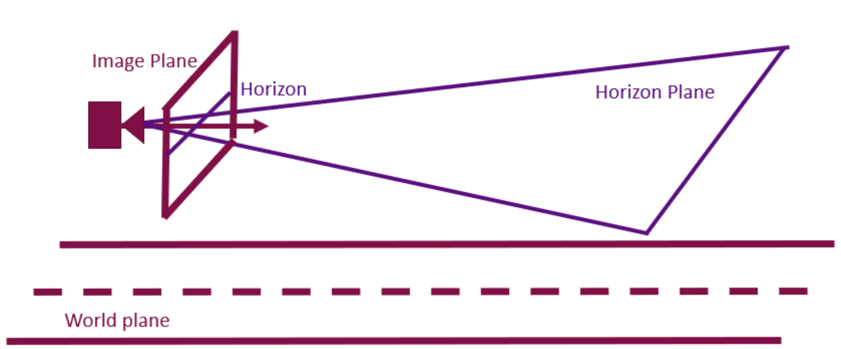
Horizon gives complete info about how camera is oriented w.r.t. world plane*!

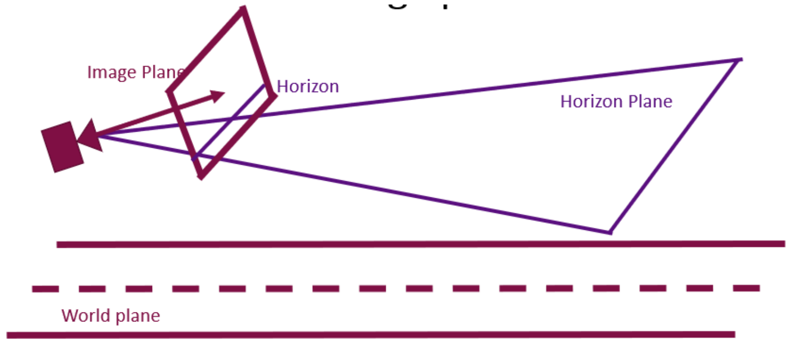
Thumb rule: “If horizon is horizontal & central, camera is correctly vertical & principal axis is parallel to world plane**!”

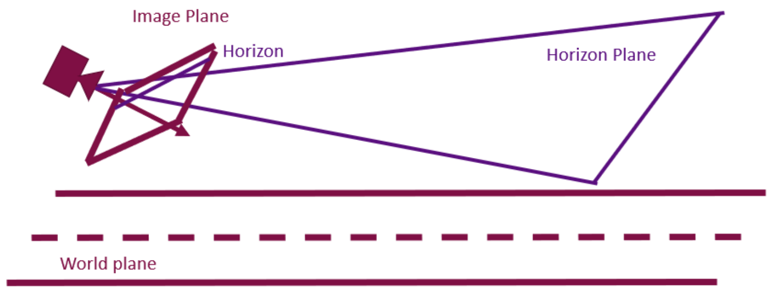












*caveat: assuming known K

**caveat: assuming that principal axis passes through image center, and camera axes are horizontal. (usually approximately true)

Homography -> Camera Pose

Recap: Pose From Homography

Recall the projection from world to camera

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim K \begin{pmatrix} r_1 & r_2 & r_3 & T \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

and assume that all points in the world lie in the ground plane $Z = 0$.

Then the transformation reads

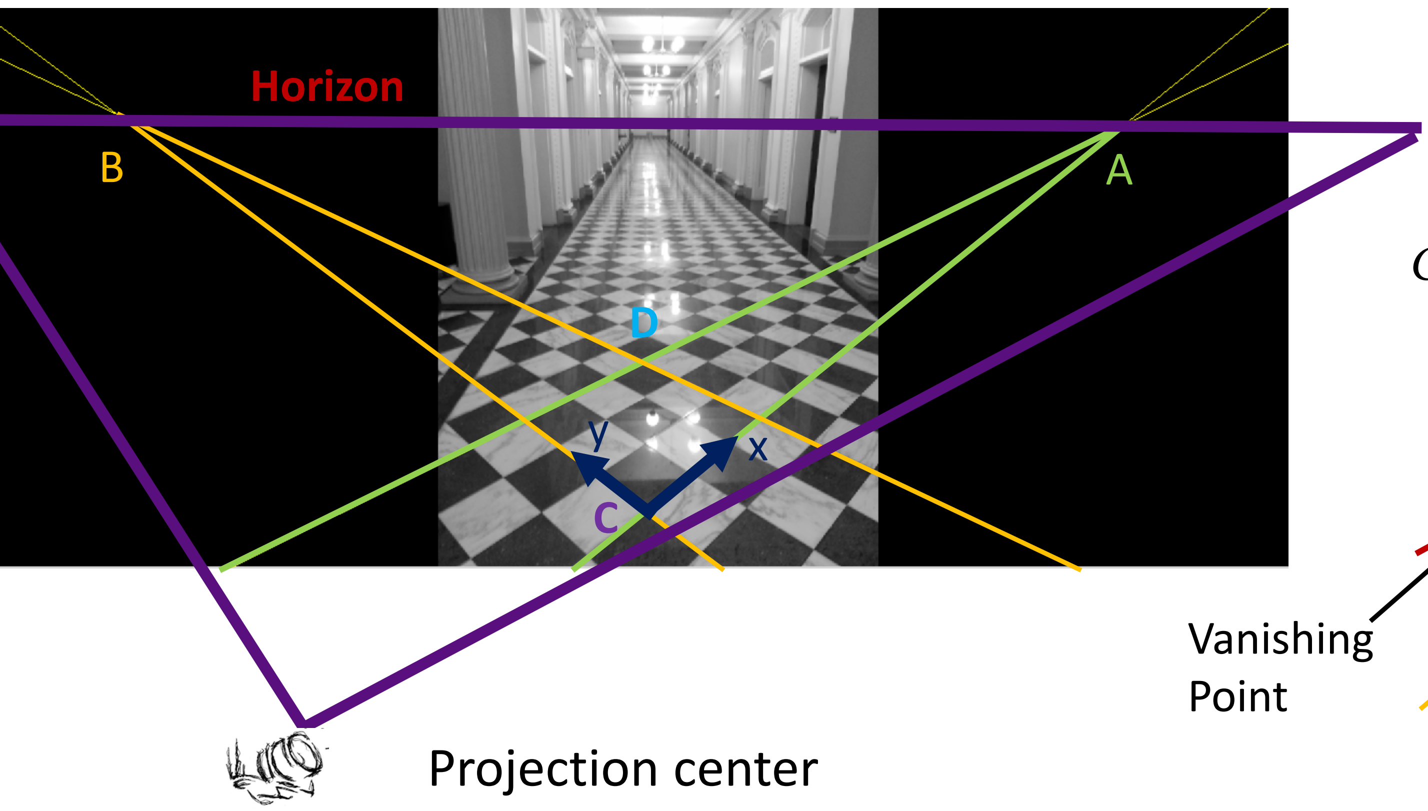
$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim \underbrace{K \begin{pmatrix} r_1 & r_2 & T \end{pmatrix}}_{} \begin{pmatrix} X \\ Y \\ W \end{pmatrix}$$

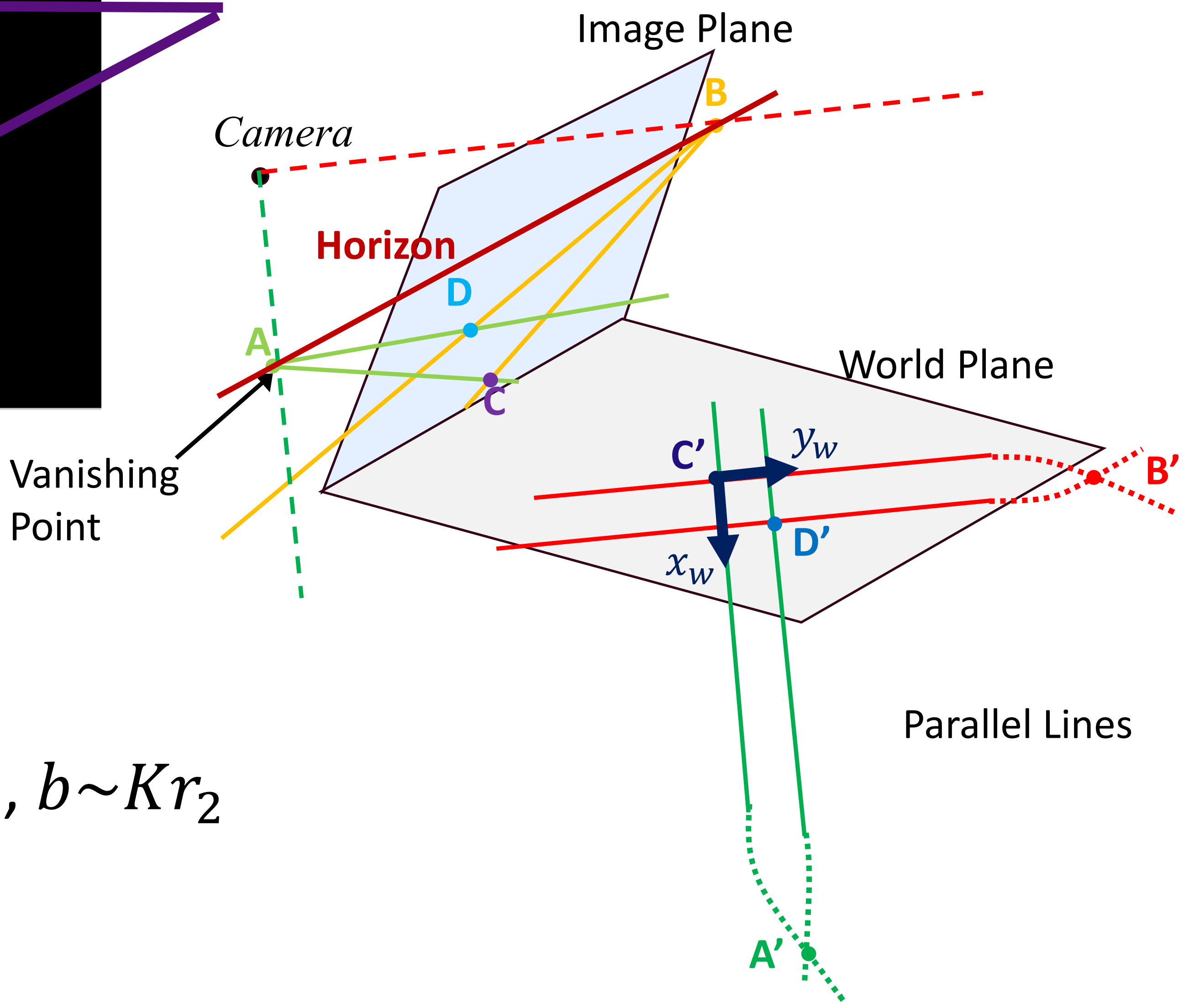
The planar homography

$H : \mathbb{P}^2 \to \mathbb{P}^2$

Computing the homography can tell us how the camera (and therefore, e.g. a robot attached to the camera) is oriented w.r.t. to a world plane! (assuming known K)

Q: Where do you get $\boldsymbol{r}_3$ from though? A: $\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$





$$H \sim K(r_1, r_2, T)$$

$$H \sim (a, b, c)$$

$$a \sim K r_1,\ b \sim K r_2$$

- We can get the rotation of camera from 2 orthogonal vanishing points on a plane, **assuming known intrinsics matrix** K.

$$\begin{tikzcd}
H \sim K(r_1, r_2, T) \arrow[dr] & & \\
 & a \sim K r_1,\ b \sim K r_2 \arrow[r] & r_1 \sim K^{-1} a,\ r_2 \sim K^{-1} b \\
H \sim (a, b, c) \arrow[ur] & &
\end{tikzcd}$$

But actually, not quite!

- According to the previous slide $K(r_1\ r_2\ T) = H$, or in other words,

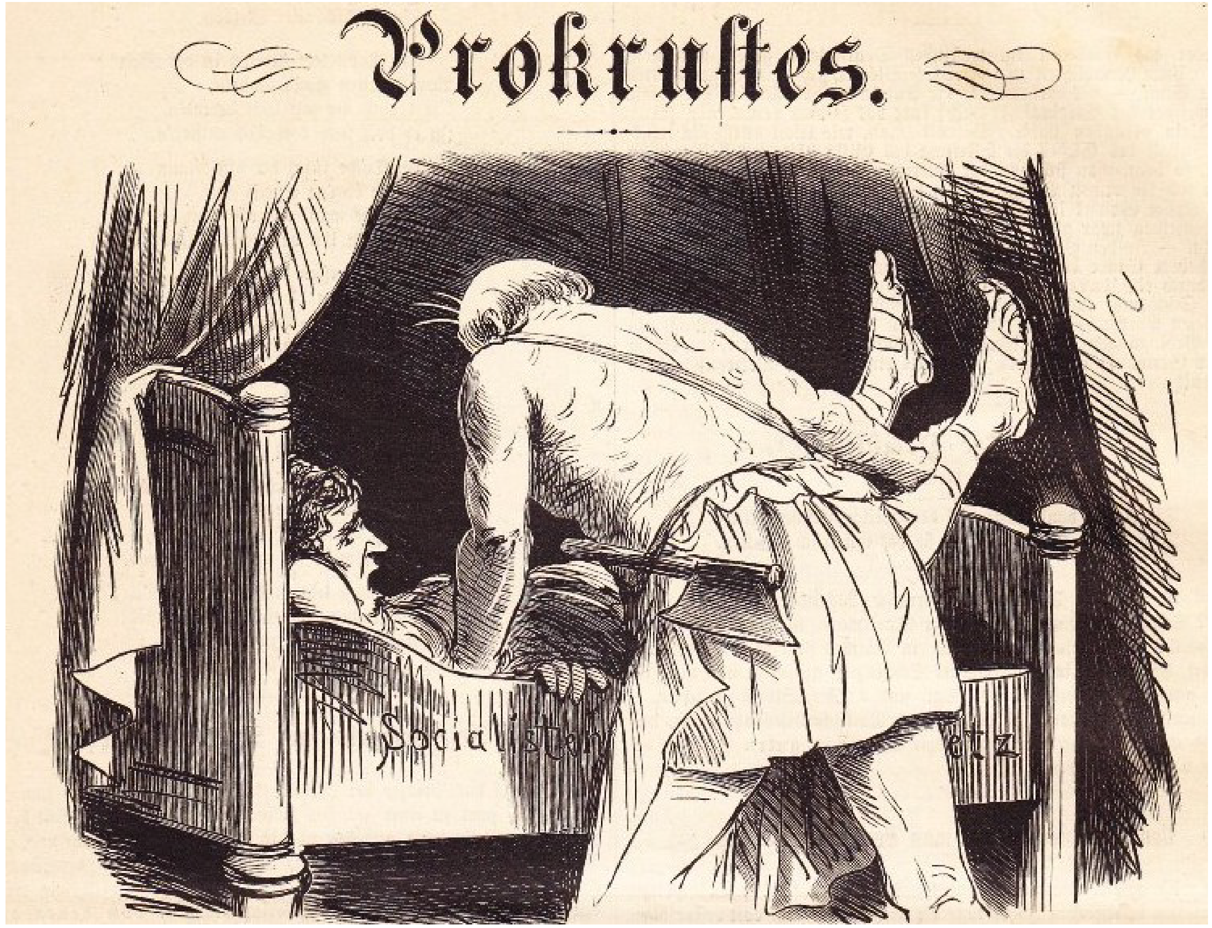
  $K^{-1}H = (r_1, r_2, T)$ and $r_3 = r_1 \times r_2$

  If only life were so simple!

- Problem: when we ***estimate*** homographies (e.g. through solving linear systems with 2n equations from $n >= 4$ point correspondences), and then compute $K^{-1}H$, we aren't guaranteed to find a *valid* r_1 and r_2 pair. i.e. an orthonormal pair.
  - So, we need to find a way to first "correct" $(K^{-1}H)_{3\times 3}$ to get orthonormal r_1 and r_2. Often called the "Procrustes", or "special orthogonal (SO) Procrustes" problem.
  - And we must solve this in real-time for robotics applications, so preferably an inexpensive approach.

The macabre Greek legend of Procrustes

We are trying to get every $(K^{-1}H)$ “traveler” to fit the “bed” of valid rotation matrices by stretching it or chopping it off.



Let us name the columns of $K^{-1}H$:

$$K^{-1}H = \begin{pmatrix} h'_1 & h'_2 & h'_3 \end{pmatrix}$$

We seek orthogonal r_1 and r_2 that are the closest to h'_1 and h'_2. The solution to this problem is given by the Singular Value Decomposition.

We find the orthogonal matrix R that is the closest to $\begin{pmatrix} h'_1 & h'_2 & h'_1 \times h'_2 \end{pmatrix}$:

$$\arg\min_{R \in SO(3)} \| R - \begin{pmatrix} h'_1 & h'_2 & h'_1 \times h'_2 \end{pmatrix} \|_F^2$$

Kabsch algorithm for Procrustes

$$\boxed{\arg\min_{R \in SO(3)} \| R - \begin{pmatrix} h_1' & h_2' & h_1' \times h_2' \end{pmatrix} \|_F^2}$$

If the SVD of

$$\begin{pmatrix} h_1' & h_2' & h_1' \times h_2' \end{pmatrix} = USV^T$$

then the solution is

$$R = U \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(UV^T) \end{pmatrix} V^T$$

The diagonal matrix is inserted to guarantee that $\det(R) = 1$.

To find the translation : $T = h_3' / \|h_1'\|$

(In case original columns were not even unit norm)

proof in supp readings- Kabsch-Algorithm-RT-from-H-proof.pdf. We will also prove it in the next class.

Proof in class notes, optional

Full Kabsch algorithm for finding pose via homography

1. Find H up to a scale factor from the point coorrespondences

2. Compute $H' = K^{-1}H$. Let H''s columns be $\begin{pmatrix} a & b & c \end{pmatrix}$

3. Minimize

$$\| \begin{pmatrix} a & b & c \end{pmatrix} - \lambda \begin{pmatrix} r_1 & r_2 & T \end{pmatrix} \|_F$$

w.r.t. $\lambda \in \mathbb{R}, r_1, r_2, T \in \mathbb{R}^3$
s.t. $r_1^T r_2 = 0$ and $\|r_1\| = \|r_2\| = 1$

Let

$$\left.\begin{aligned} (a\ b) = U_{3x2} \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix} V_{2x2}^T. \end{aligned}\right\}$$

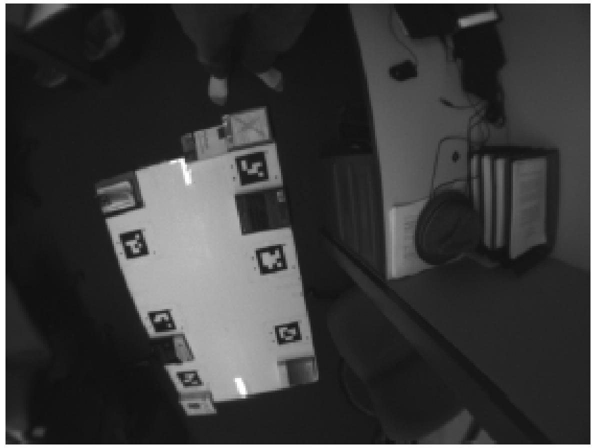
Alternative to running Kabsch including the 3rd column $c = h_1' \times h_2'$ as on last slide

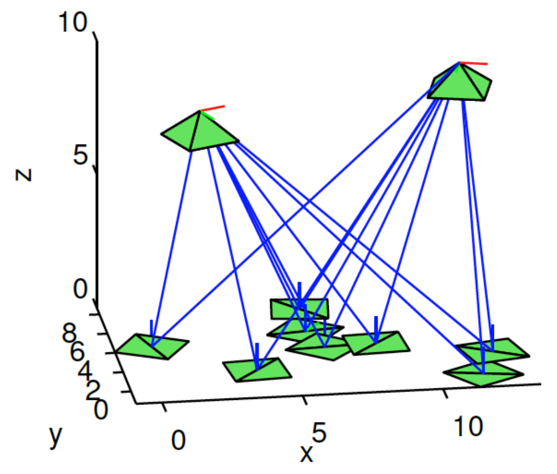
Then

$$\begin{pmatrix} r_1 & r_2 \end{pmatrix} = U_{3x2} V_{2x2}^T \quad \text{and} \quad \lambda = \frac{s_1 + s_2}{2}$$

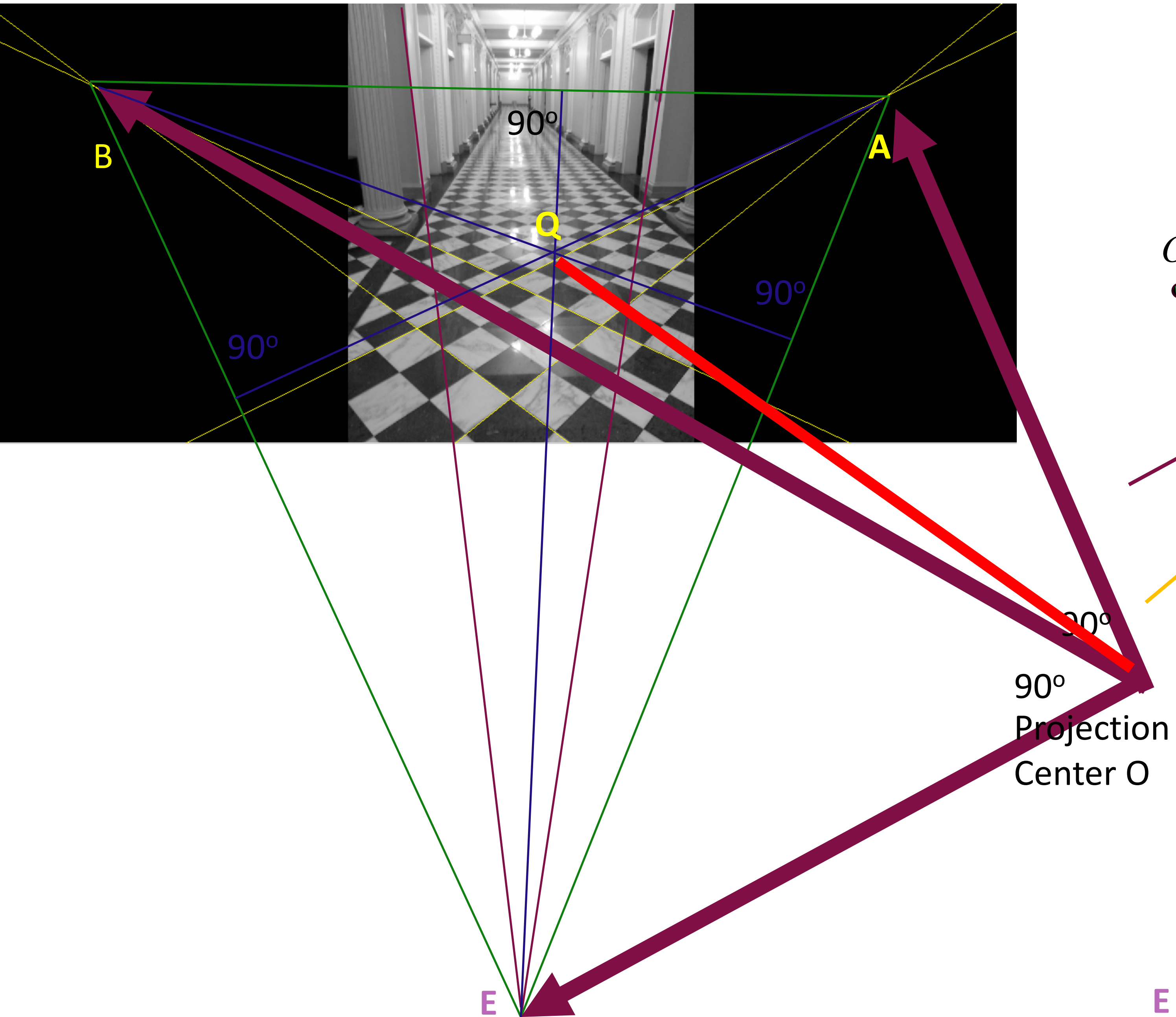
4. $T = c/\lambda$ and $R = \begin{pmatrix} r_1 & r_2 & r_1 \times r_2 \end{pmatrix}$. Scale R to have determinant 1 if needed.

So now, camera pose (actually) known w.r.t world plane!





Vanishing Points -> intrinsics K


B
90º
A
Q
90º
90º
90º
90º
Projection Center O
E

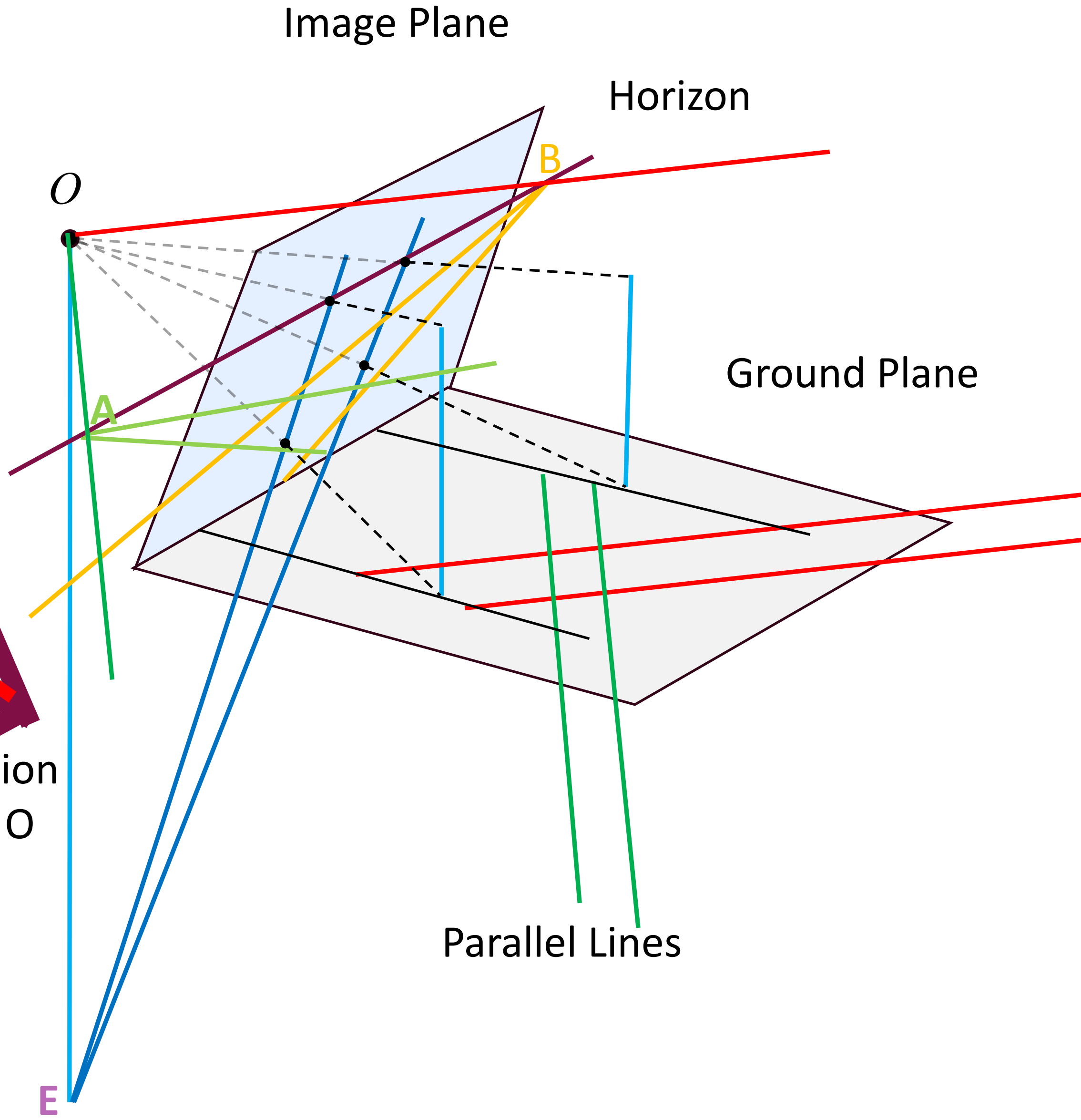
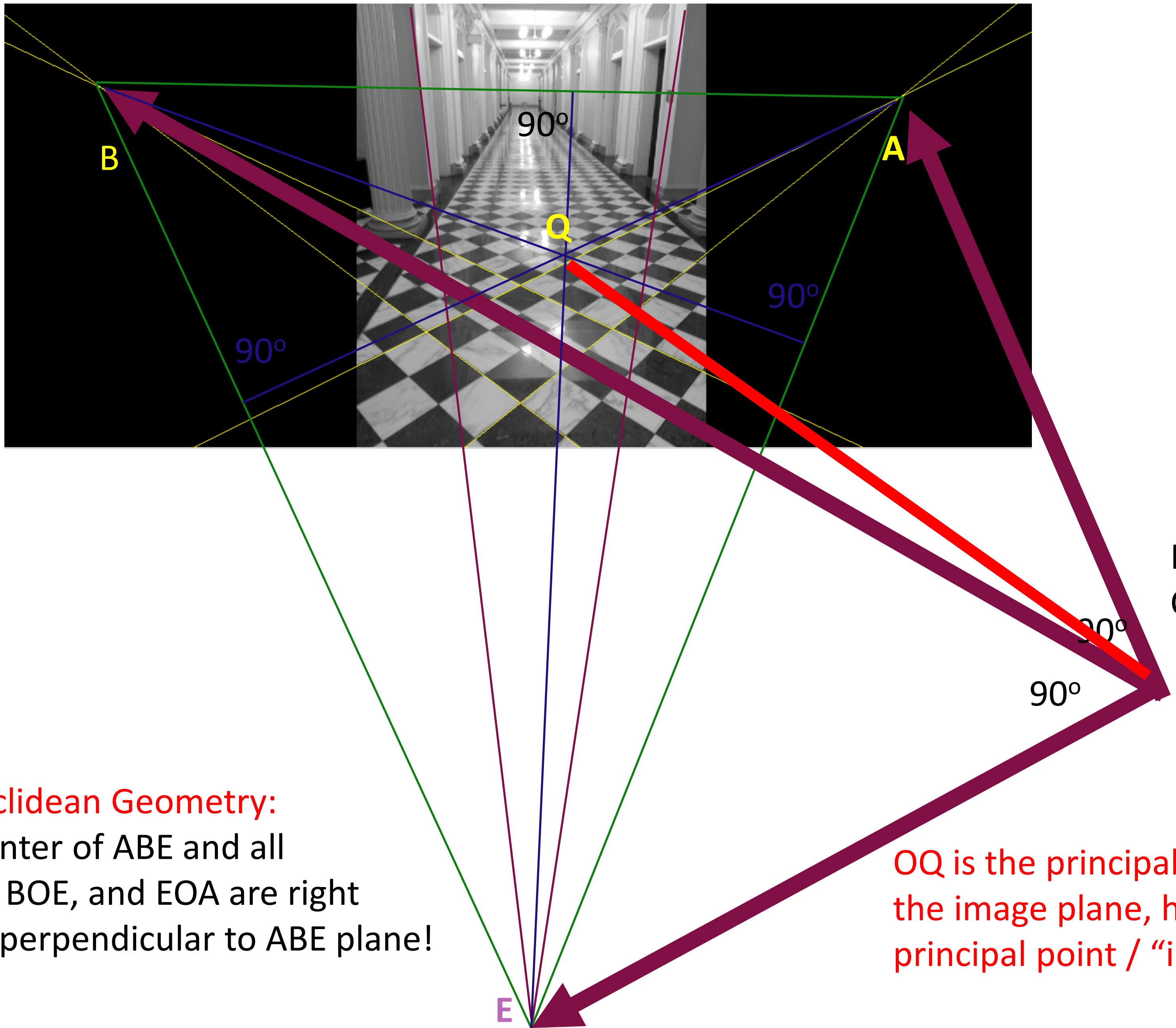


Image Plane
Horizon
O
B
A
Ground Plane
Parallel Lines
E

Let Q be the orthocenter of the triangle ABC



Theorem from Euclidean Geometry:
If Q is the orthocenter of ABE and all three angles AOB, BOE, and EOA are right angles , the OQ is perpendicular to ABE plane!

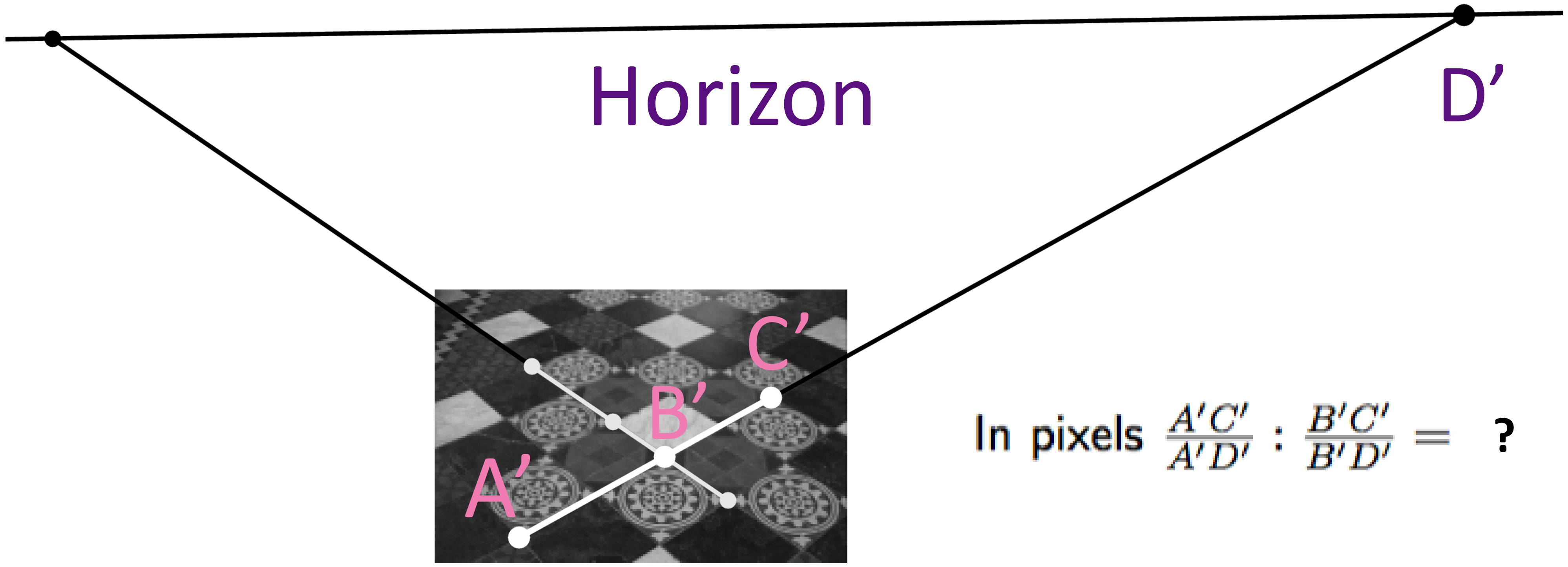
OQ is the principal axis and ABE is the image plane, hence, Q is the principal point / “image center”

Summary

- If we have 3 orthogonal VPs, we can get full intrinsics K (focal length and image center), and also extrinsics R.
  - What’s missing? Just the translation t.
    - And that information is not contained in VPs, because camera translations don’t affect the VPs!
    - Which is why, when we found homographies (that do contain full information about translation), we used more than just VPs.

Cross Ratios & Length Measurements from Single Images ("Single View Metrology")

What happens when one of the points is at infinity?



In pixels $\frac{A'C'}{A'D'} : \frac{B'C'}{B'D'} =$?

When a point D is at infinity, the cross-ratio becomes a ratio !

$\frac{AC}{AD} : \frac{BC}{BD} = \frac{AC}{BC}$ (Think $\frac{AC}{\infty} : \frac{BC}{\infty} = \frac{AC}{BC} \times \frac{\infty}{\infty} = \frac{AC}{BC}$)

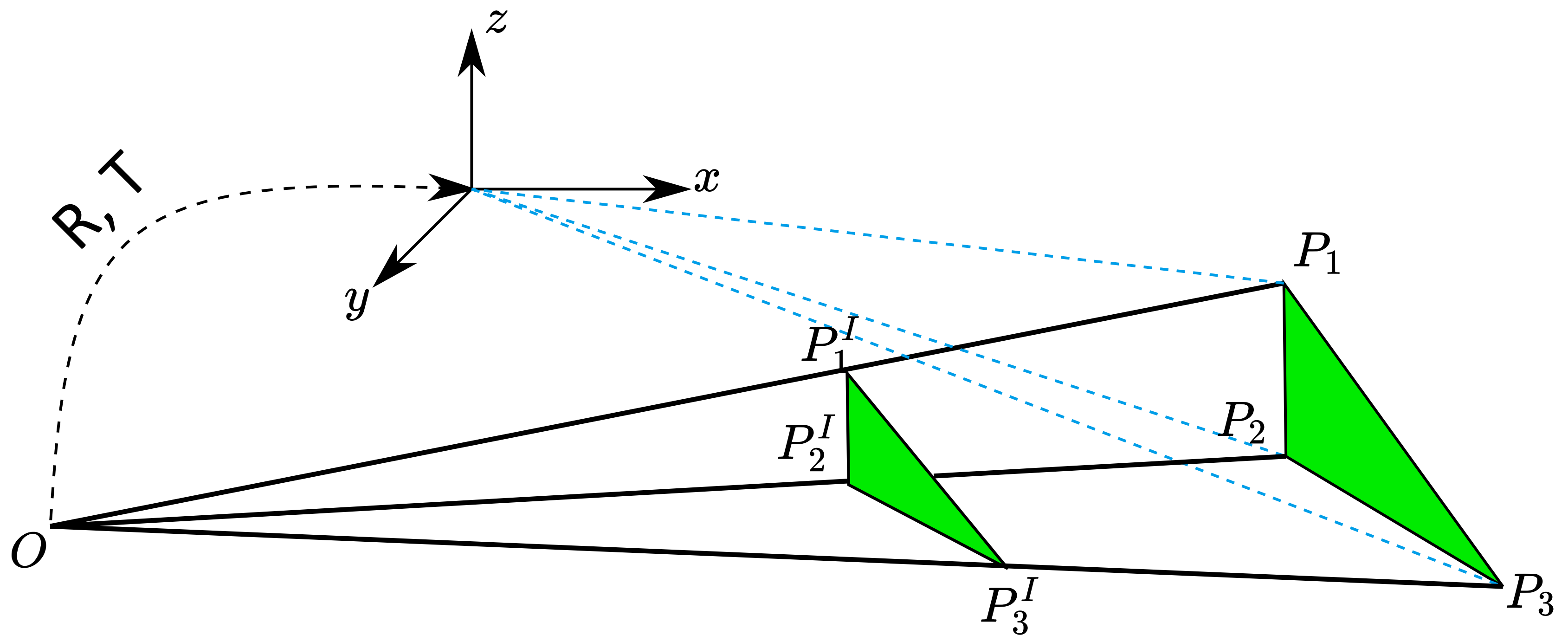
Pose from Point Correspondences, the Perspective N Point Problem (PnP)

Localization by observing known 3D points from the world?

A real problem for autonomous cars, for example!
GPS: ~ a few feet accuracy. Just not good enough.

Instead, autonomous cars rely on 3D maps of the world to localize!

The *Perspective* 3-Point Problem



- Given the point correspondences, find camera pose R, T

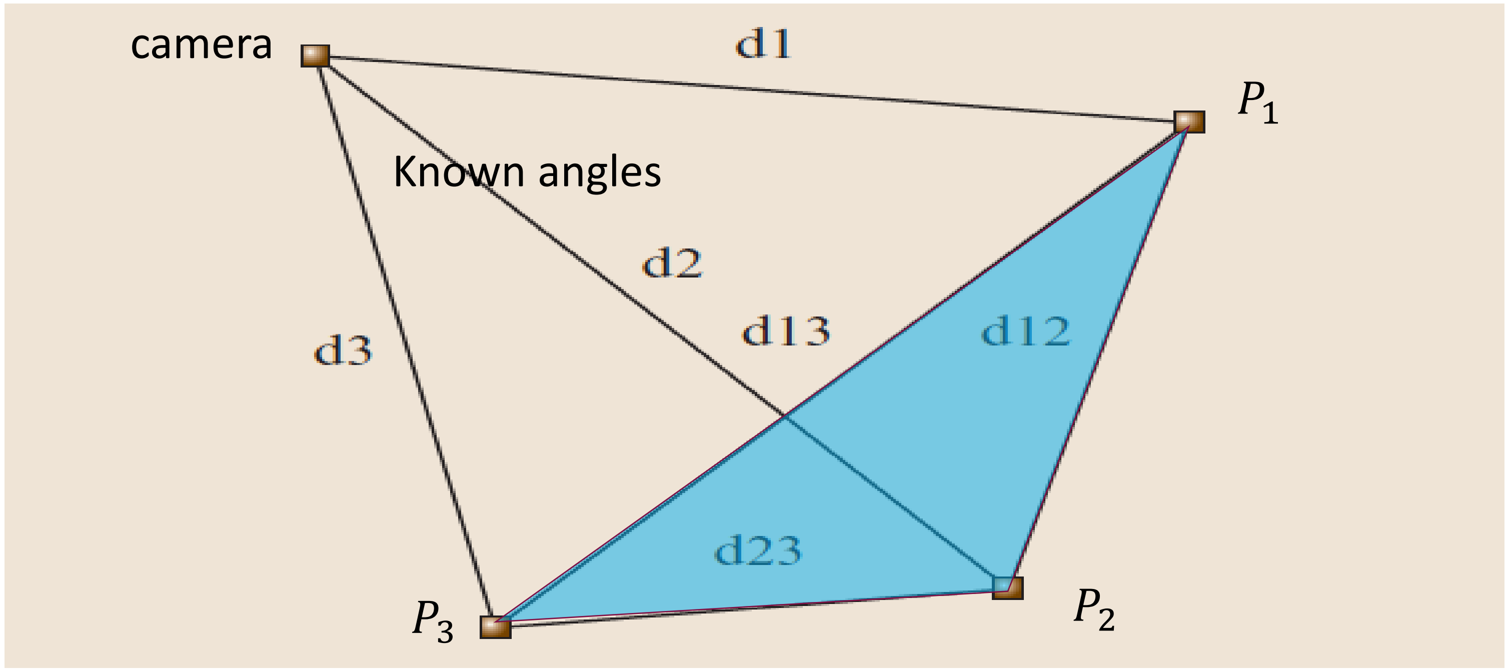
What are the differences from 4-Point Algorithm?

Jingnan Shi, https://jingnanshi.com/blog/pnp_minimal.html

P3P v.s. Homography

- Why P3P needs only three point correspondences, while computing Homography needs four point correspondences?

P3P from Pixels



A triangle's world coordinates P_i are known, and its pixel coordinates are known

Pixels → “Calibrated coordinates”

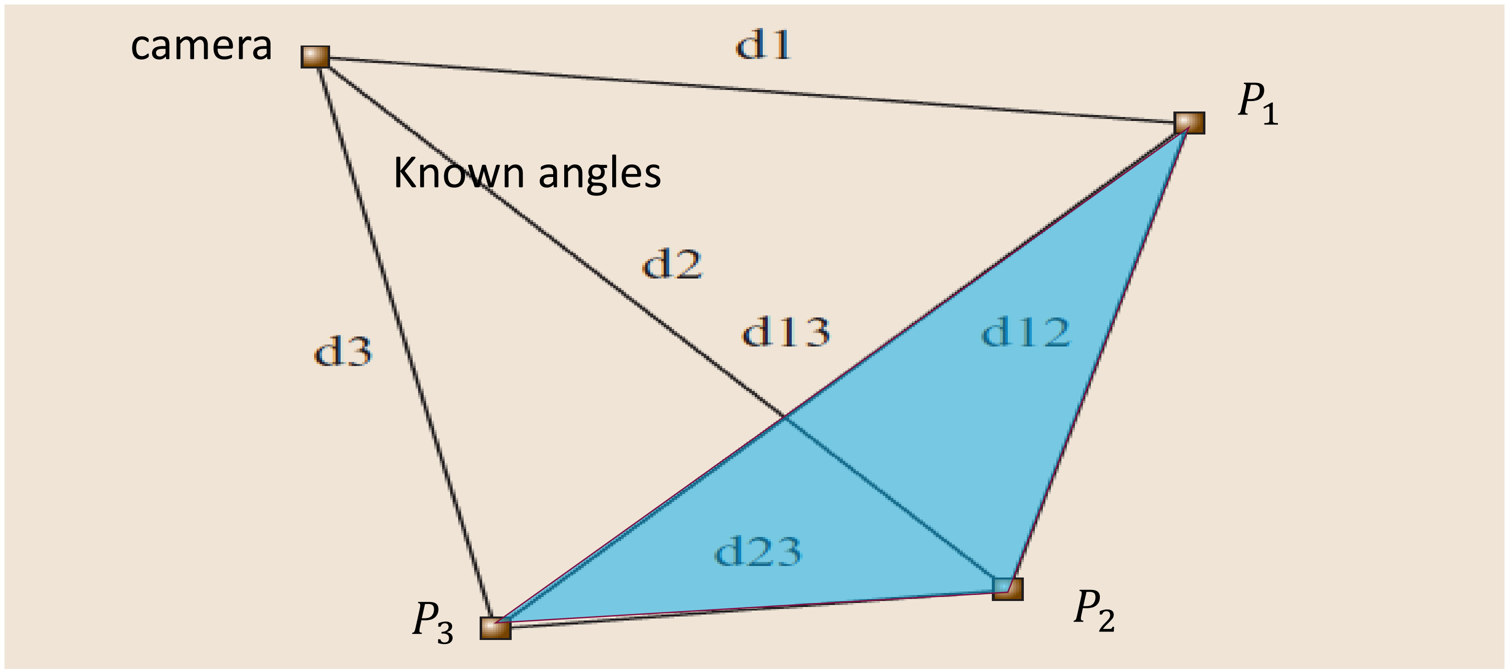
RGB images only provide pixel coordinates u, v for each vertex, not camera-centric 3D coordinates. Convert these to:

“Calibrated coordinates”: $\boldsymbol{p_i} \sim K^{-1}(u_i \quad v_i \quad 1)^T$

These are essentially image plane coordinates with principal point as origin, and focal length set to 1.

Euclidean interpretation: $\boldsymbol{p_i}$ is a vector in camera coordinates, originating from the camera center and pointing toward the 3D point corresponding to pixel coordinates (u_i, v_i).

P3P from ~~Pixels~~ Calibrated Coordinates



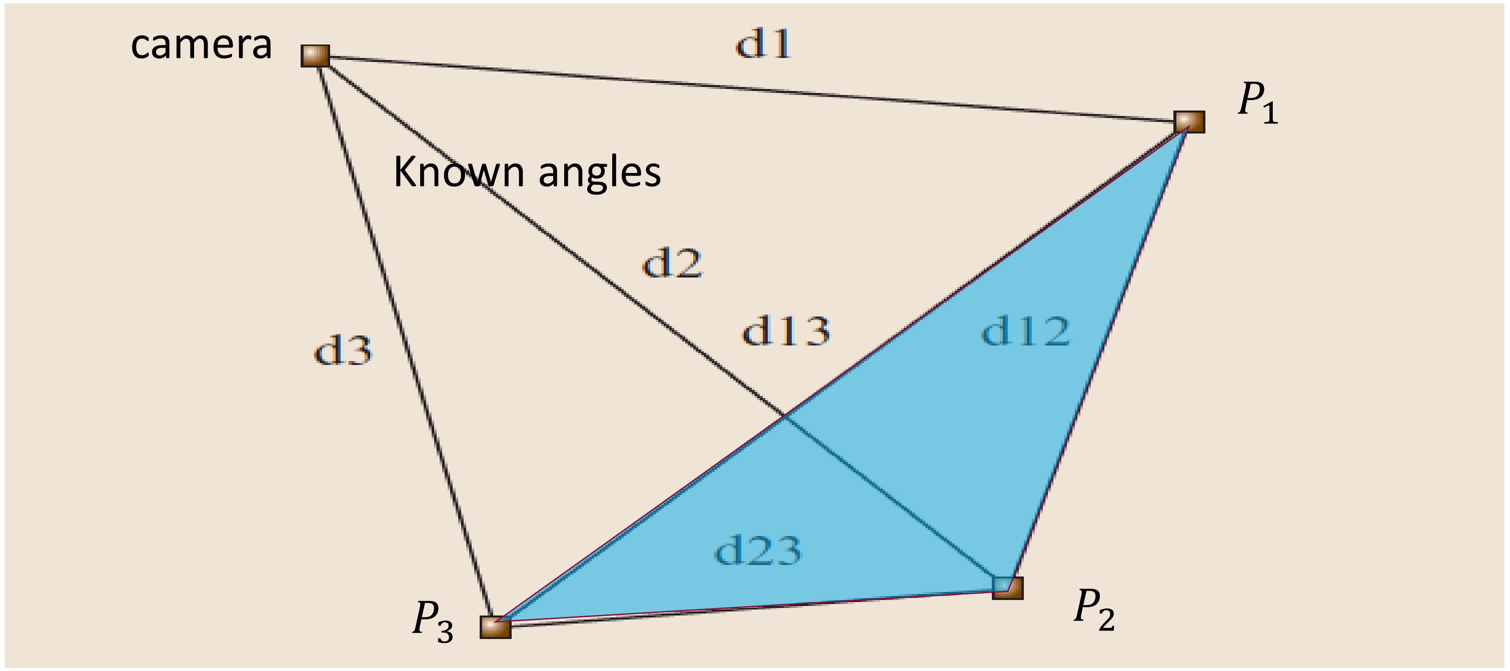
A triangle's world coordinates $\boldsymbol{P_i}$ are known, and its camera-centric calibrated coordinates $\boldsymbol{p_i}$ are known

The P3P problem: Find λ_i, R, T such that

$$\lambda_1 p_1 = RP_1 + T$$
$$\lambda_2 p_2 = RP_2 + T$$
$$\lambda_3 p_3 = RP_3 + T$$

Q: Are λ_i the same as "depths" d_i?
A: No, because p_i are not unit vectors.

P3P from Calibrated Coordinates



A triangle's world coordinates $\boldsymbol{P_i}$ are known, and its camera-centric calibrated coordinates $\boldsymbol{p_i}$ are known

The P3P problem: Find d_i, R, T such that

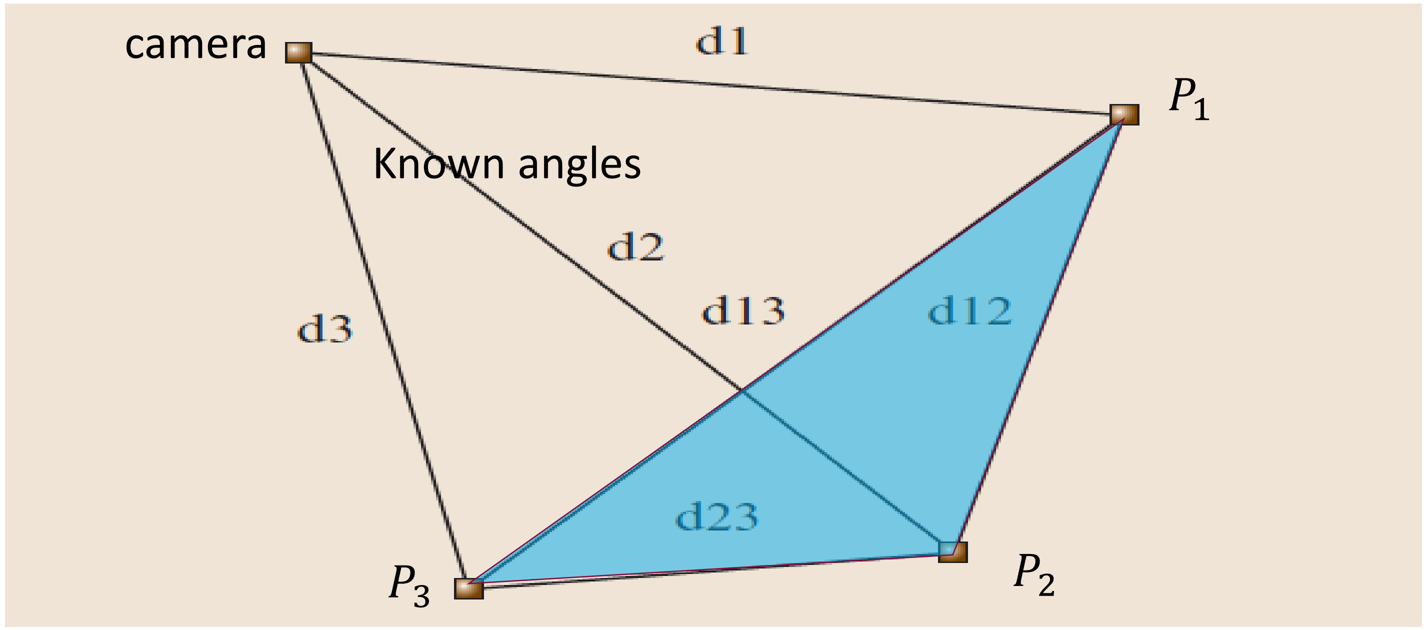
$$\frac{d_i}{||p_i||_2} p_i = R P_i + T, \qquad \forall i = 1,2,3$$

P3P Step 1: Finding depths d_i of triangle vertices

Let δ_{ij} denote the observed angle between the calibrated coordinates p_i and p_j

Then cosine law reads

$$d_i^2 + d_j^2 - 2d_i d_j \cos\delta_{ij} = d_{ij}^2$$



The cosine law

$$d_i^2 + d_j^2 - 2d_i d_j \cos\delta_{ij} = d_{ij}^2$$

applies for each point pair. With 3 points we could solve 3 quadratic equations for $d_{i=1\ldots3}$.

Set $d_2 = ud_1$ and $d_3 = vd_1$ and solve all three equations for d_1:

$$\begin{aligned}
d_1^2 &= \frac{d_{23}^2}{u^2 + v^2 - 2uv\cos\delta_{23}} \\
d_1^2 &= \frac{d_{13}^2}{1 + v^2 - 2v\cos\delta_{13}} \\
d_1^2 &= \frac{d_{12}^2}{u^2 + 1 - 2u\cos\delta_{12}}
\end{aligned}$$

P3P Step 1: The algebraic drudgery

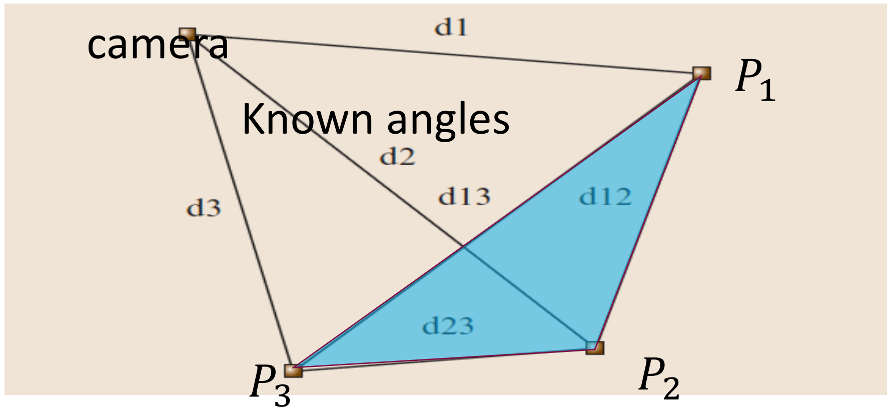
Reduces to two quadratic equations in u and v.

$$\begin{aligned} d_{13}^2(u^2+v^2-2uv\cos\delta_{23}) &= d_{23}^2(1+v^2-2v\cos\delta_{13}) && (1)\\ d_{12}^2(1+v^2-2v\cos\delta_{13}) &= d_{13}^2(u^2+1-2u\cos\delta_{12}) && (2)\end{aligned}$$

a) Solve Eqn (1) for u^2 in terms of u, v, v^2 (and constants).
b) Plug this solution into Eqn (2), so that it has no u^2 term. Solve for u in terms of v, v^2, and constants.
c) Plug this solution for u back into Eqn (1), so that it has no more u or u^2. Instead, it is a 4th degree equation in v. *Get the 4 real solutions analytically.*
d) Then plug back into the solution found in step b) above, to get u.
e) Then get d_1 from the quadratic equations on the last page.
f) Then plug back into $d_2 = ud_1$ and $d_3 = vd_1$ from the last page to get d_2, d_3.

Grunert 1841

P3P Step 2: 3D->3D Pose/ 3D Registration. Find R&T!



The P3P problem has reduced to:

Find R, T such that

$$\frac{d_i}{||p_i||_2} p_i = RP_i + T, \qquad \forall i = 1,2,3$$

But naïve direct solution of the linear system is perilous, because rotation matrix R might not be valid.

(Does this remind you of something?)

Kabsch Algorithm for 3D->3D

- Compute centroids $\bar{A}$ and $\bar{B}$ of the two sets of 3D points.
- Create matrices $A_{3\times n},\ B_{3\times n}$ after subtracting $\bar{A}$ and $\bar{B}$ from all points in the two sets.
- To find R, we must solve: $\underset{R\in SO(3)}{\operatorname{argmin}} ||A - RB\,||_F = \underset{R\in SO(3)}{\operatorname{argmin}} ||R - AB^T||_F^2$
  - First set $\hat{R} = (AB^T)_{3\times 3}$
  - Then decompose $\hat{R} = \mathrm{U}\Sigma V^T$
  - Set $R = U\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{pmatrix}V^T$
- Set $T = \bar{A} - R\bar{B}$

PnP produces non-unique solutions $(n > 3)$?

- Recall that P3P Step 1 produced non-unique solutions for the distances d_i.
- But if we have $n > 3$ point correspondences, we only need Step 2. This is the **"Perspective-N-Point" problem, or simply PnP.**

Direct solution for PnP ($n > 3$): Steps

Again, first switch to calibrated coordinates:

Pose from N points in space given intrinsic parameters K and correspondences $(X_i, Y_i, Z_i, x_i, y_i)_{i=1\dots N}$
where

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \sim K^{-1} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix}$$

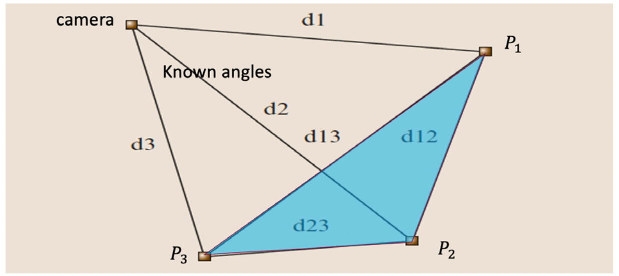
where u_i, v_i are pixel coordinates.

Direct solution for PnP: Steps

Given $(X_i, Y_i, Z_i, x_i, y_i)_{i=1\ldots N}$
find R, T such that

$$\lambda_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = R \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + T$$

P3P from Pixels



A triangle's world coordinates P_i are known, and its camera-centric calibrated coordinates p_i are known

The P3P problem: Find λ_i, R, T such that

$$\lambda_1 p_1 = RP_1 + T$$
$$\lambda_2 p_2 = RP_2 + T$$
$$\lambda_3 p_3 = RP_3 + T$$

Identical to the stage we reached with P3P, shown above.

But now, we are after a *direct* solution.

No need to solve explicitly for depths as we did in P3P, so need to find unit vectors etc.

Direct solution for PnP: Steps

Instead substitute λ and get 2 linear equations per point correspondence

$$\lambda_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = R \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + T \quad \Longrightarrow \quad \begin{aligned} x_i &= \frac{r_{11}X_i + r_{12}Y_i + r_{13}Z_i + T_1}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + T_3} \\ y_i &= \frac{r_{21}X_i + r_{22}Y_i + r_{23}Z_i + T_2}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + T_3} \end{aligned}$$

Get linear equations by cross-multiplying ...

$$A_{2n\times 12}\boldsymbol{x}_{12} = 0,$$

where $\boldsymbol{x}$ is a vector of all the unknowns including R and T.

Need at least 11 equations = at least 6 point correspondences to solve.

Q: Why do we need more points for the direct method, when P3P was able to work with 3 points?

We meet Procrustes and Kabsch yet again

Again, just solving the linear system won't give good rotations. So full steps:

1. Solve $A_{2n\times 12}\boldsymbol{x}_{12} = 0$ from $n \geq 6$ correspondences.
2. Then, assemble the rotation matrix $\hat{R}$ from the solution for $\boldsymbol{x}$.
3. Then, find the closest valid rotation matrix by Kabsch!

Decompose $\hat{R} = \mathrm{U}\Sigma V^T$

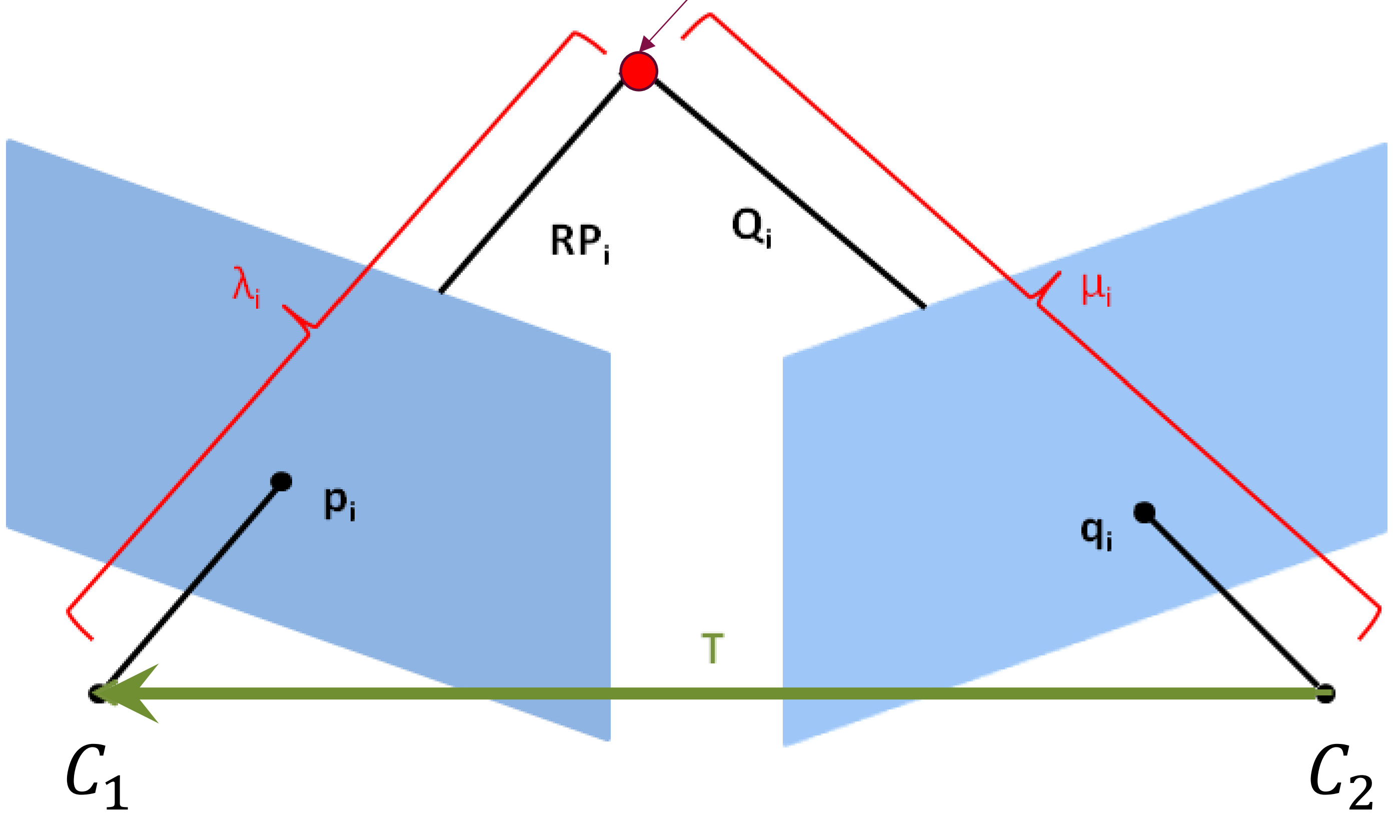
Then, set $R = U \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{pmatrix} V^T$

4. May now need to adjust translation T to be consistent with new R, e.g., by solving $A_{2n\times 12}\boldsymbol{x}_{12} = 0$, for only t_1, t_2, t_3 (elements of $\boldsymbol{x}$), holding R fixed.

3D Motion from Two Views or Structure from Motion (SfM)

Input: Two Calibrated Views of the Same 3D Scene

$$\boxed{[\boldsymbol{\lambda p}]_{\text{in } C1 \text{ camera coords}} = [\boldsymbol{\mu q}]_{\text{in C2 camera coords}}}$$



$$[X]_{C2} = R[X]_{C1} + T$$

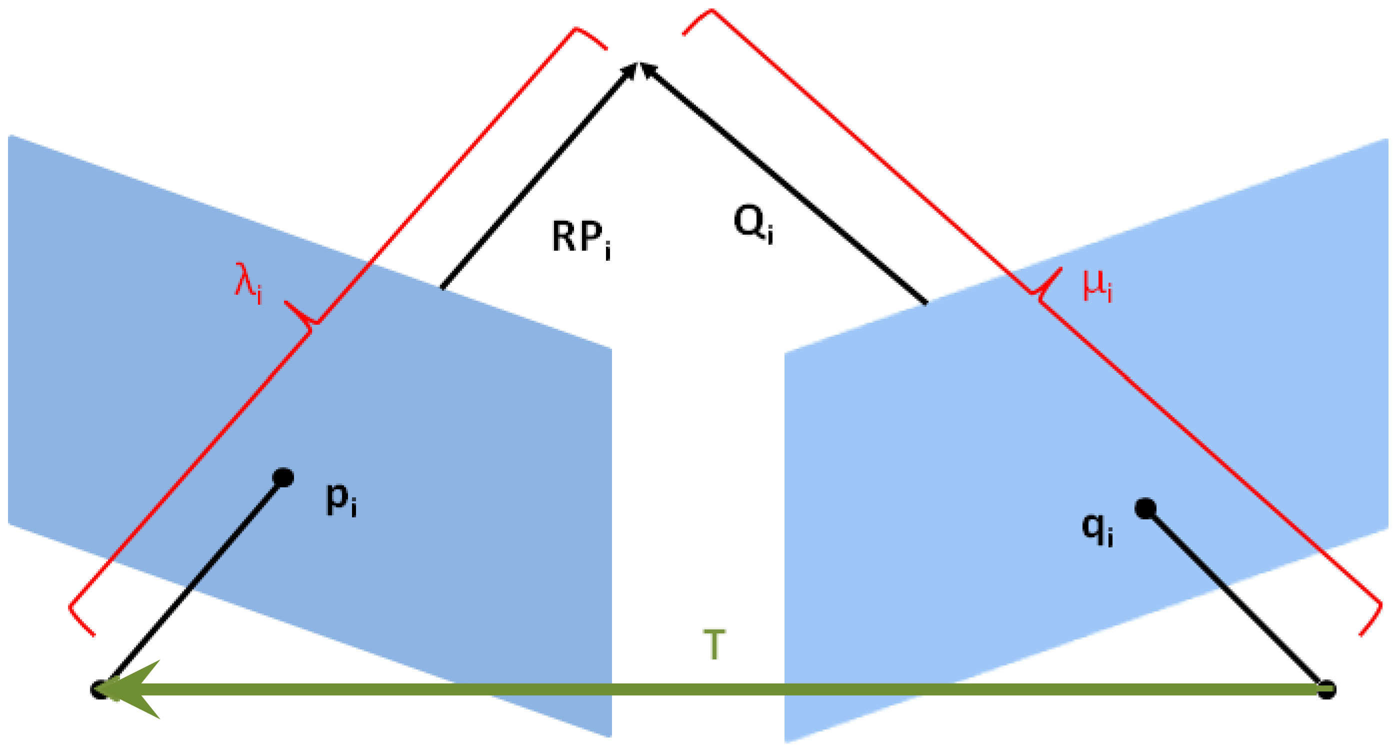
Now, any 3D coordinates in C_1 frame:
$[X]_{C1} \rightarrow [R[X]_{C1} + T]_{C2}$

So $[\lambda p]_{C1} \rightarrow [R(\lambda p) + T]_{C2} = [\mu q]_{C2}$

Or now that we are in the same coordinate system:

$$R(\lambda p) + T \ = \mu q$$

Two Calibrated Views of the Same 3D Scene

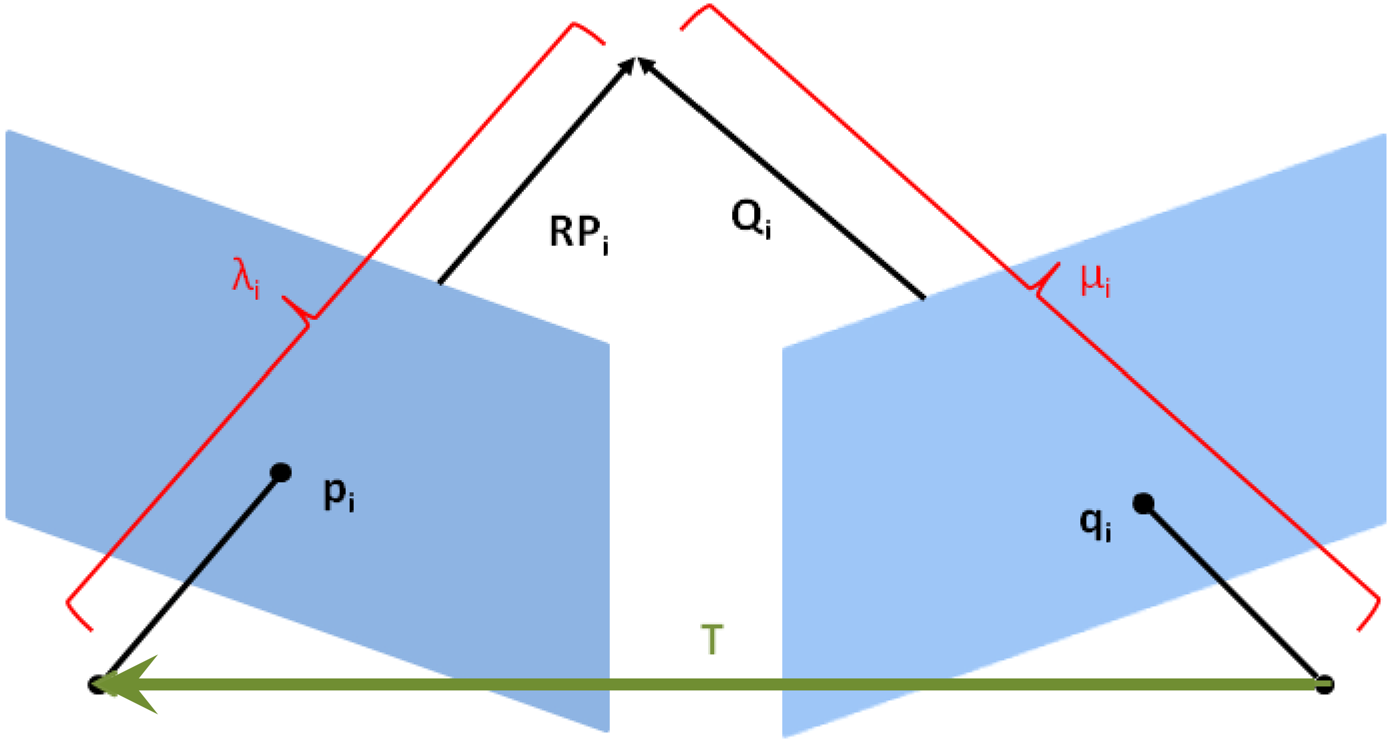


$$R(\lambda p) + T \; = \mu q$$

Given 2D correspondences (p,q)

Find motion R, T and depths λ, μ.

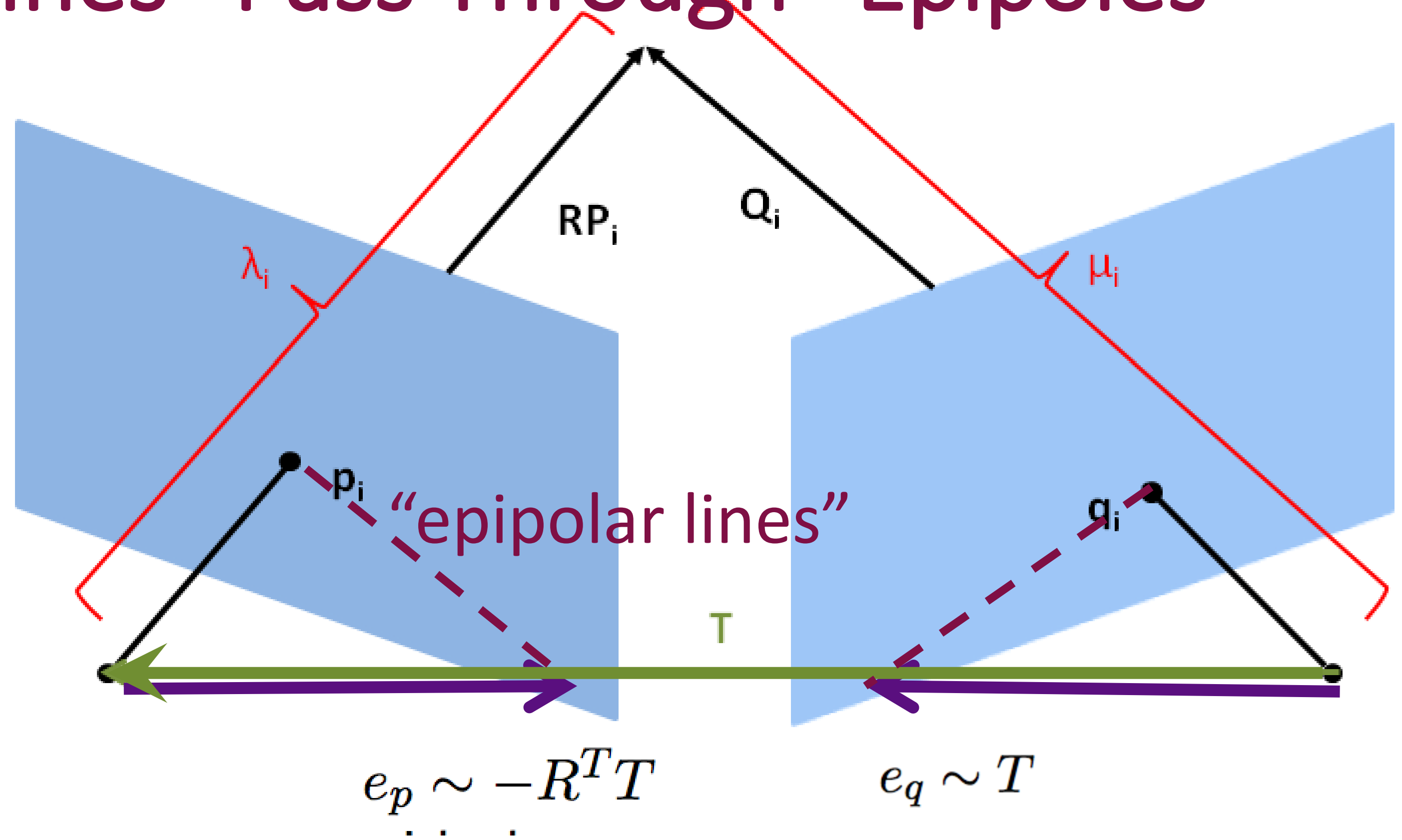
"Epipolar Constraints" Between Two Views of a Scene



We can eliminate the depths from $R(\lambda p) + T = \mu q$ and obtain the epipolar constraint:

$$\boldsymbol{q}_i^T (T \times R\boldsymbol{p}_i) = 0$$

“Epipolar Lines” Pass Through “Epipoles”



$e_p \sim -R^T\ T$ and $e_q \sim T$ are the “epipoles” = images of the other camera center on each plane = intersections of baseline T with the two planes = VP of the translation direction in each plane.

All epipolar lines in each image plane pass through its epipole.

The Essential Matrix E

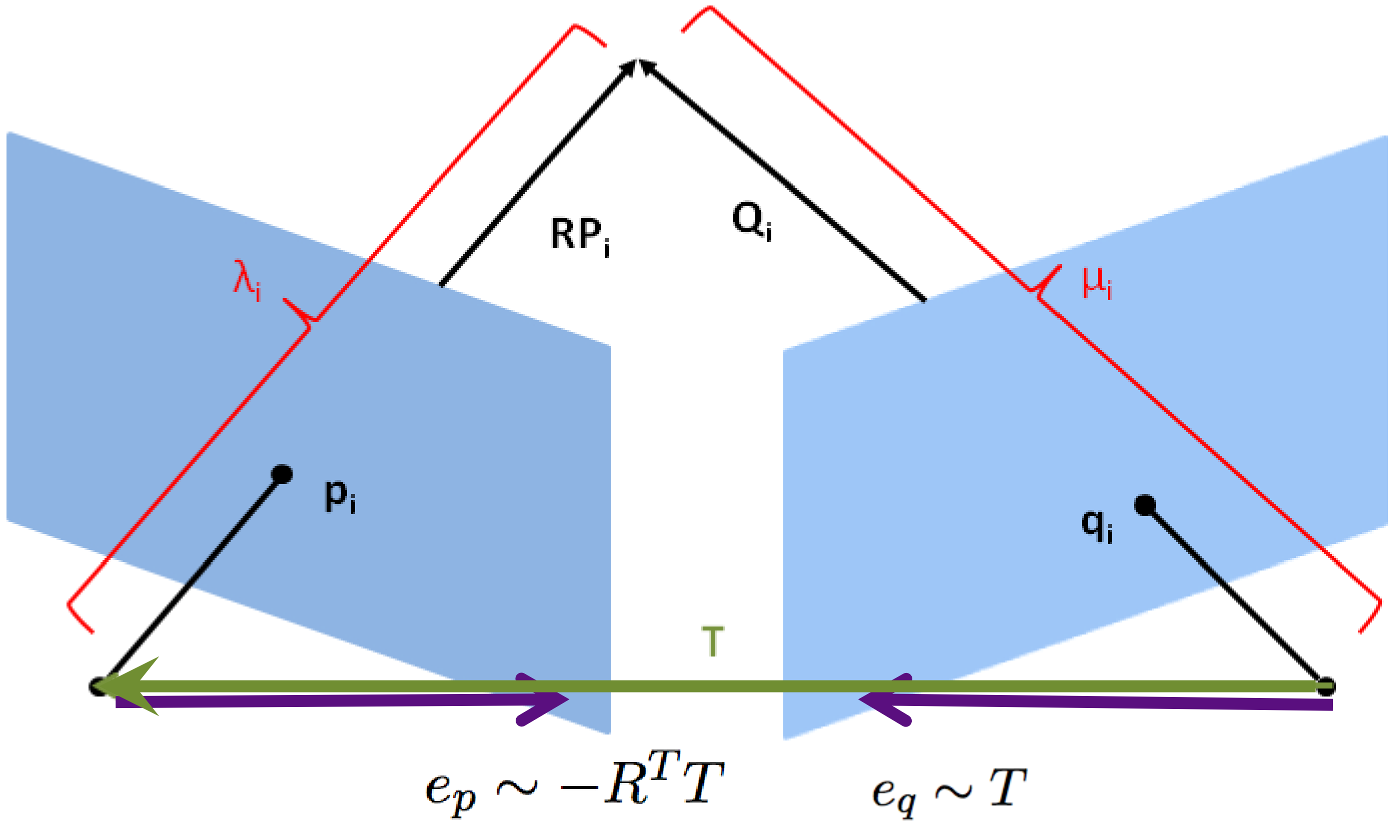
We had: $\boldsymbol{q}_i^T(T \times R\boldsymbol{p}_i) = 0$

$$\Rightarrow \boldsymbol{q}_i^T(\hat{T}R)\ \boldsymbol{p}_i = 0$$

Renaming $E = (\hat{T}R)$:

$$\boldsymbol{q}_i^T E\ \boldsymbol{p}_i = 0$$

“Essential matrix”



Now linear in the new unknowns $E_{3\times3}$! But will need to recover $T_{3\times1}$, $R_{3\times3}$ later.

Is the epipolar constraint really linear in E?

$q_i^T E_{3\times 3} p_i = 0$ is a single equation that is linear in the elements of E
Can write this out explicitly as below.

If

$$E = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix}$$

then epipolar constraint can be rewritten as

$$\begin{aligned} q^T \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} &= q^T \left(p_x e_1 + p_y e_2 + p_z e_3 \right) \\ &= \begin{pmatrix} p_x q^T & p_y q^T & p_z q^T \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = 0 \end{aligned}$$

This equation is linear

Longuet-Higgins 1981

8-Point Algorithm

Let $\vec{a} = \begin{pmatrix} p_x q^T & p_y q^T & p_z q^T \end{pmatrix}$

$$\begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{pmatrix}_{n \times 9} E' = 0$$

One row per point correspondence

where a_i is the known 1 x 9 vector of image points and E' is the essential matrix re-organized into a 9 x 1 column vector.

E' has to be in the null-space of $\begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{pmatrix}$.

Does this remind you of something?
Hint: 4-Point Collineations, PnP, …

Solution: As before, set E' to the last right singular vector of $A_{n\times 9}$

Longuet-Higgins 1981

After solving for E, not Quite Done Yet!

$E = \hat{T}R$ has fewer than 8 DOF. T has 3 DOF $(+3)$, R has 3 DOF $(+3)$, and E is scale invariant (-1), so total 5 DOF. **So not any 3x3 matrix is a valid essential matrix.**

- **Problem:** **Given the above, how to ensure that the estimated $\boldsymbol{E}$ is a valid essential matrix?**
- **Problem:** **How to decompose $\boldsymbol{E}$ into the $\boldsymbol{\hat{T}}, \boldsymbol{R}$ required in SfM?**

*https://tutorial.math.lamar.edu/classes/calciii/quadricsurfaces.aspx

Constructing Valid Essential Matrices and Decomposing Them

Necessary and sufficient condition: E is essential **iff** $\sigma_1(E) = \sigma_2(E) \neq 0$ and $\sigma_3(E) = 0$.

Part 1: Proving ‘necessary’ (“If E is essential, then …”) will tell us about properties of essential matrices, so we can correct the E matrices from the direct method to become valid.

Part 2: Proving ‘sufficient’ (“If singular values …, then …”) will help us solve R, T from E for a particular pair of cameras.

Proof Sketch Part 1: Singular Values of A Valid Essential Matrix

Hence, we have proved that **if a matrix is essential, namely, can be decomposed as the product of an antisymmetric $\hat{T}$ and a special orthogonal R then its singular values are $\sigma_1 = \sigma_2 > 0$ and $\sigma_3 = 0$.**

$$\boxed{\sigma_1(E) = \sigma_2(E) \neq 0 \text{ and } \sigma_3(E) = 0.}$$

Condition A

Utility: Having obtained an initial estimate E through direct solution of >=8 epipolar constraints, we may enforce "condition A" above on it, by:

1. Compute SVD $E = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} V^T$
2. Then, set $E_{new} = U \begin{bmatrix} (\sigma_1+\sigma_2)/2 & & \\ & (\sigma_1+\sigma_2)/2 & \\ & & 0 \end{bmatrix} V^T$

This satisfies $\underset{E_{new}}{\operatorname{argmin}} ||E_{new} - E||_F^2$ s.t. E_{new} meets "condition A".

Since E is scale-invariant, optionally, can also just set Σ to $diag(1,1,0)$

We don't yet know how to get $\boldsymbol{R}, \boldsymbol{T}$ from $\boldsymbol{E}$

Proof Part 2: Construction of E as $\hat{T}R$

Necessary and sufficient condition: E is essential **iff** $\sigma_1(E) = \sigma_2(E) \neq 0$ and $\sigma_3(E) = 0$.

We have to prove the sufficient condition:

If the singular values of a matrix are are $\sigma_1 = \sigma_2 > 0$ and $\sigma_3 = 0$ then the matrix can be decomposed into the product of an antisymmetric $\hat{T}$ and a $\underbrace{\textbf{special orthogonal } R.}_{\text{i.e. valid rotation matrix } R\text{, orthonormal with determinant +1 (right-handed coordinate system)}}$

Proof Part 2: Construction of E as $\hat{T}R$

$$\begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

Consider the simplest matrix satisfying $\sigma_1 = \sigma_2$ and $\sigma_3 = 0$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

It indeed can be decomposed into antisymmetric / skew and rotation, e.g.:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

#1

skew-symmetric $\widehat{T_{-z}}$ x rotation $R_{z,\pi/2}$

$T_{-z} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$ and rotation by 90° about z axis

The Four Possible R, T decompositions of $\pm E$

If $E = U\Sigma V^T = U \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$, there are four solutions for the pair $(\hat{T}, R)$

Note: Both R matrices are not guaranteed to have determinant +1. Could be -1. But at least one with +1.

$$
\begin{array}{ccc}
\pm E & = & \hat{T} \quad R \\
 & & \boxed{\sigma}\left(\widehat{UT_{-z}}\right)\left(UR_{z,+\pi/2}V^T\right) \\
 & & \sigma\left(\widehat{UT_{+z}}\right)\left(UR_{z,+\pi/2}V^T\right) \\
 & & \sigma\left(\widehat{UT_{+z}}\right)\left(UR_{z,-\pi/2}V^T\right) \\
 & & \sigma\left(\widehat{UT_{-z}}\right)\left(UR_{z,-\pi/2}V^T\right)
\end{array}
$$

Can get rid of the scale σ, no harm done.

Last left singular vector of E, with either + or - sign

Can disambiguate by enforcing positive depths for all points (most points if noisy) in both cameras.

But how to get depths?

Computing depths λ_i, μ_i through "triangulation"

Triangulation is possible if we have computed R and T but again up to a scale factor. Set $\|T\| = 1$:

$$\boxed{\underbrace{(q_i - Rp_i)}_{3\times 2} \underbrace{\begin{pmatrix} \mu_i \\ \lambda_i \end{pmatrix}}_{2\times 1} = \underbrace{T}_{3\times 1}}$$

There are then 3 equations with 2 unknowns λ_i and μ_i for each point.

Solve with pseudo-inverse.

The full two-view 8-point algorithm

Direct solution of E

❶ Build the homogeneous linear system by stacking epipolar constraints $q_i^T(T \times Rp_i) = 0,\ i = 1, \ldots, 8$:

$$\underbrace{\begin{bmatrix} \vdots \\ (q_i \otimes p_i)^T \\ \vdots \end{bmatrix}}_{A\ (8\times 9)} \begin{bmatrix} e_1' \\ e_2' \\ e_3' \end{bmatrix}$$

❷ Let $\begin{bmatrix} e_1' \\ e_2' \\ e_3' \end{bmatrix}$ be the nullspace of A (if $\sigma_8 \approx 0$ give up)

The full two-view 8-point algorithm

Make E valid

❸ $\begin{bmatrix} e_1' & e_2' & e_3' \end{bmatrix} = U\text{diag}\,(\sigma_1' \; \sigma_2' \; \sigma_3')V^T$. Then use the following estimate of the essential matrix:

$$E = U\text{diag}\left(\frac{\sigma_1' + \sigma_2'}{2}, \frac{\sigma_1' + \sigma_2'}{2}, 0\right) V^T$$

Decompose Into T, R

❹ $T = \pm \hat{u}_3 \quad R = UR_{Z,\pi/2}V^T$ or $R = UR_{Z,-\pi/2}V^T$

❺ Try all four pairs (T, R) to check if reconstructed points are **in front** of the cameras $\boxed{\lambda q = \mu R p + T}$ give $\lambda, \mu > 0$.