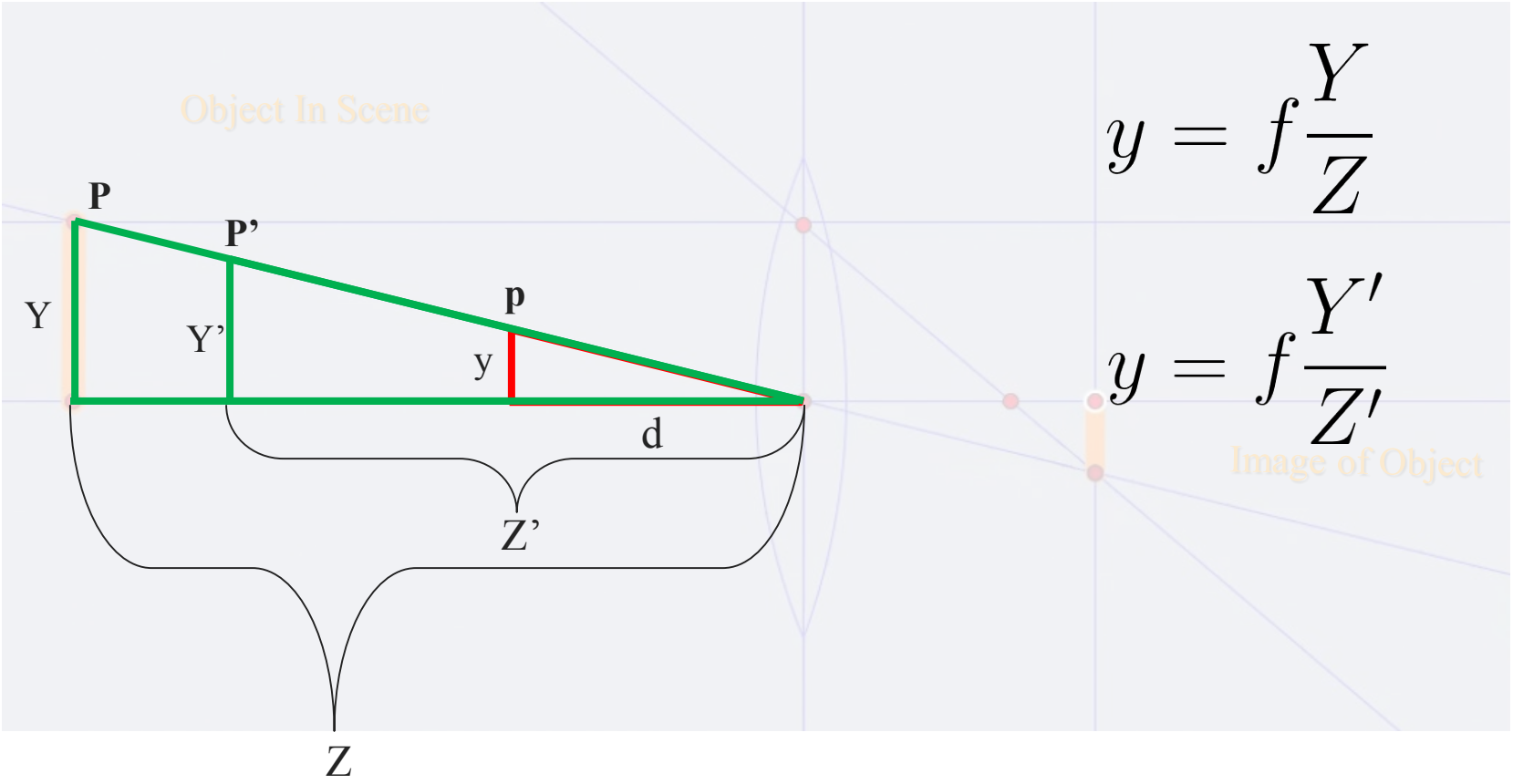
Perception:
Projective Geometry

Recall: Ambiguous Scale

Last time we looked at the problem of scale ambiguity
We said that there was no solution with vision alone



$$y = f\frac{Y}{Z}$$

$$y = f\frac{Y'}{Z'}$$

Characterizing Image Motion

We can still characterize 3D motion from images if we ignore the scale factor. We can later recover the scale factor if we have additional sensors or knowledge of the scene

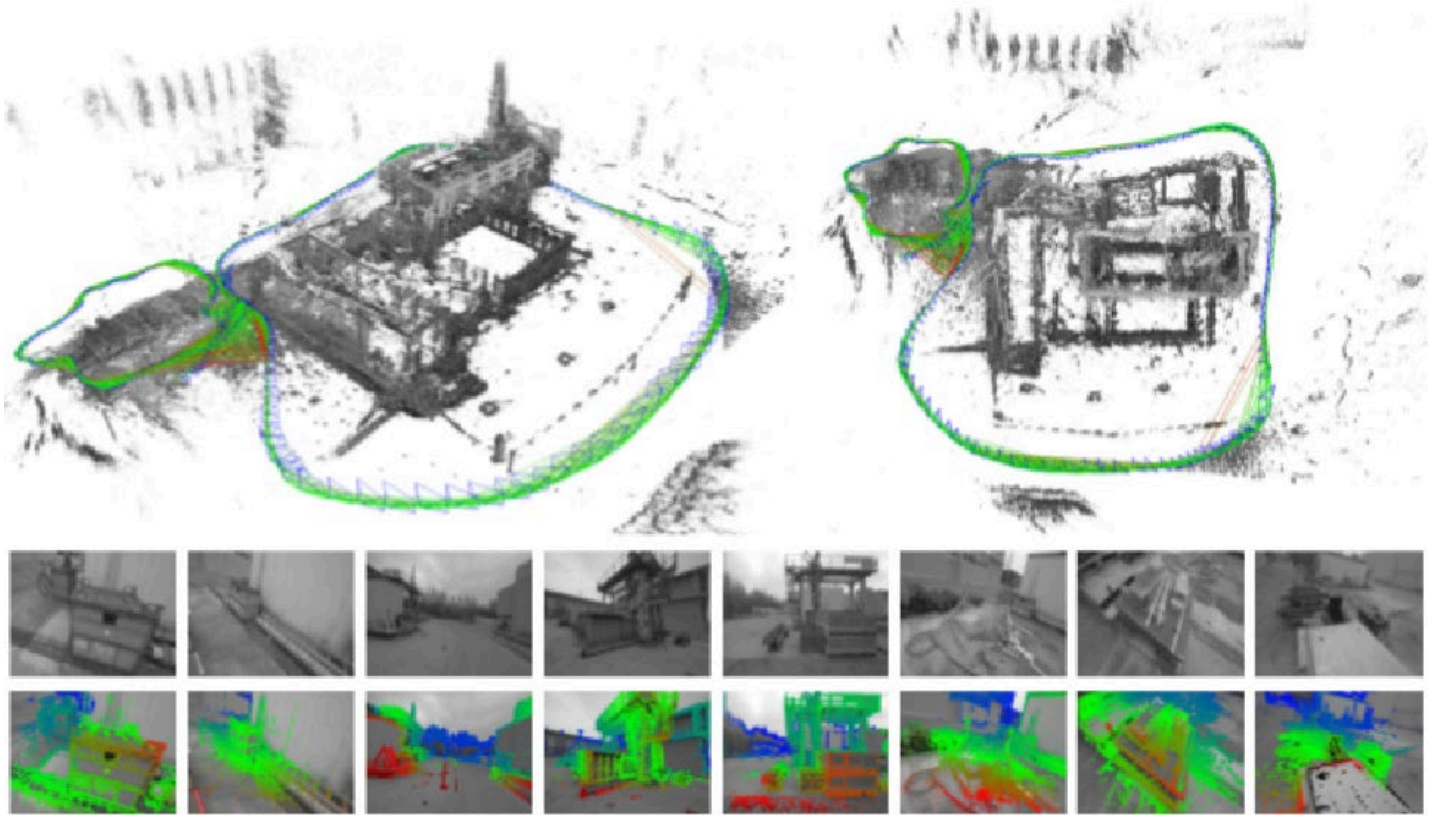
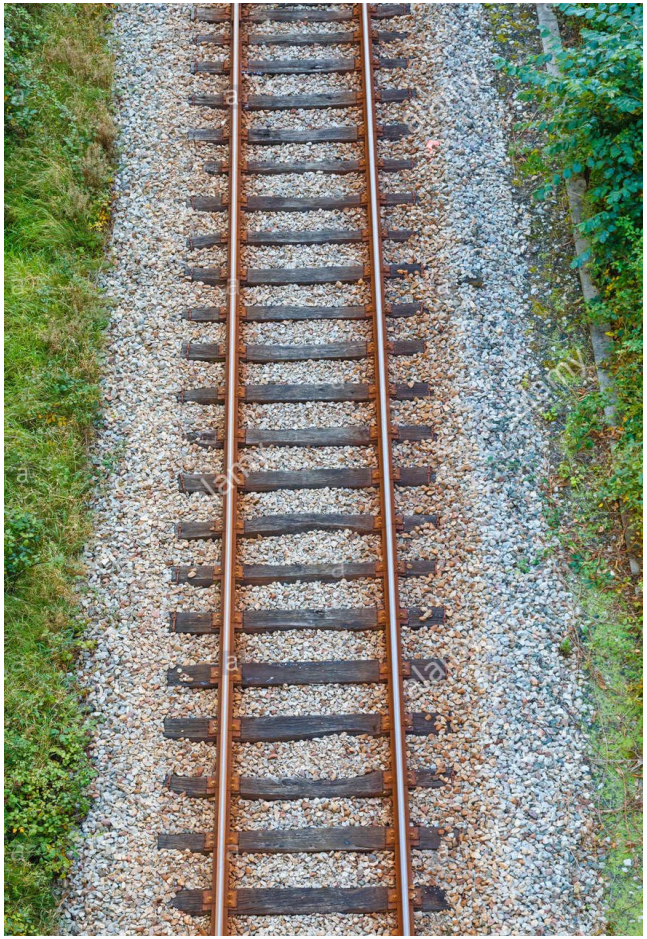


Image taken from LSD-SLAM Integration within AR System (https://arxiv.org/abs/1702.02514)

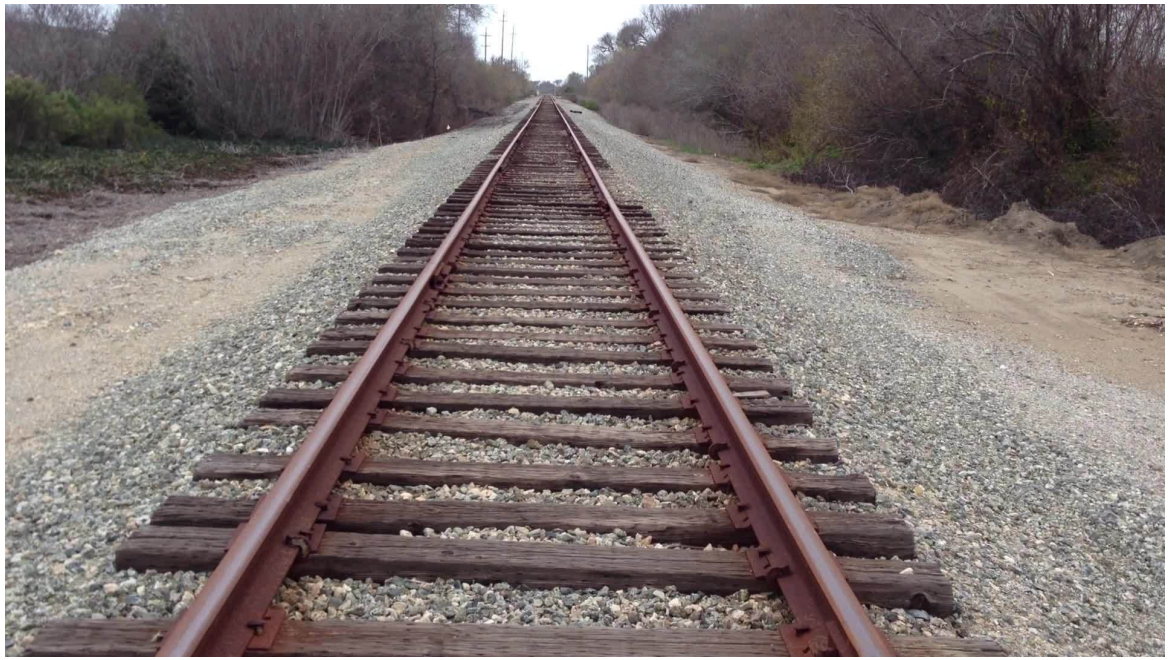
Perspective Projection - Parallel Lines

Consider the following example: We have parallel train tracks that go infinitely far. Parallel lines never intersect…



Perspective Projection - Parallel Lines

Consider the following example: We have parallel train tracks that go infinitely far. Parallel lines never intersect…
And yet they do when we look down them on the ground



Perspective Projection - Parallel Lines

Consider the following example: We have parallel train tracks that go infinitely far. Parallel lines never intersect…
And yet they do when we look down them on the ground

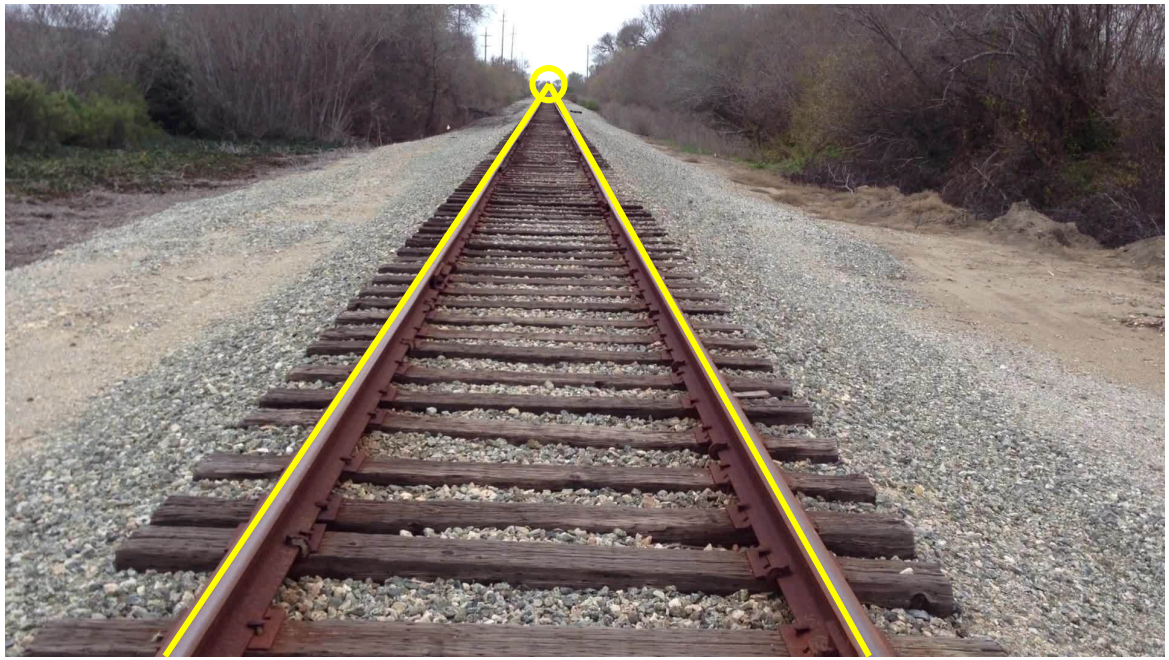
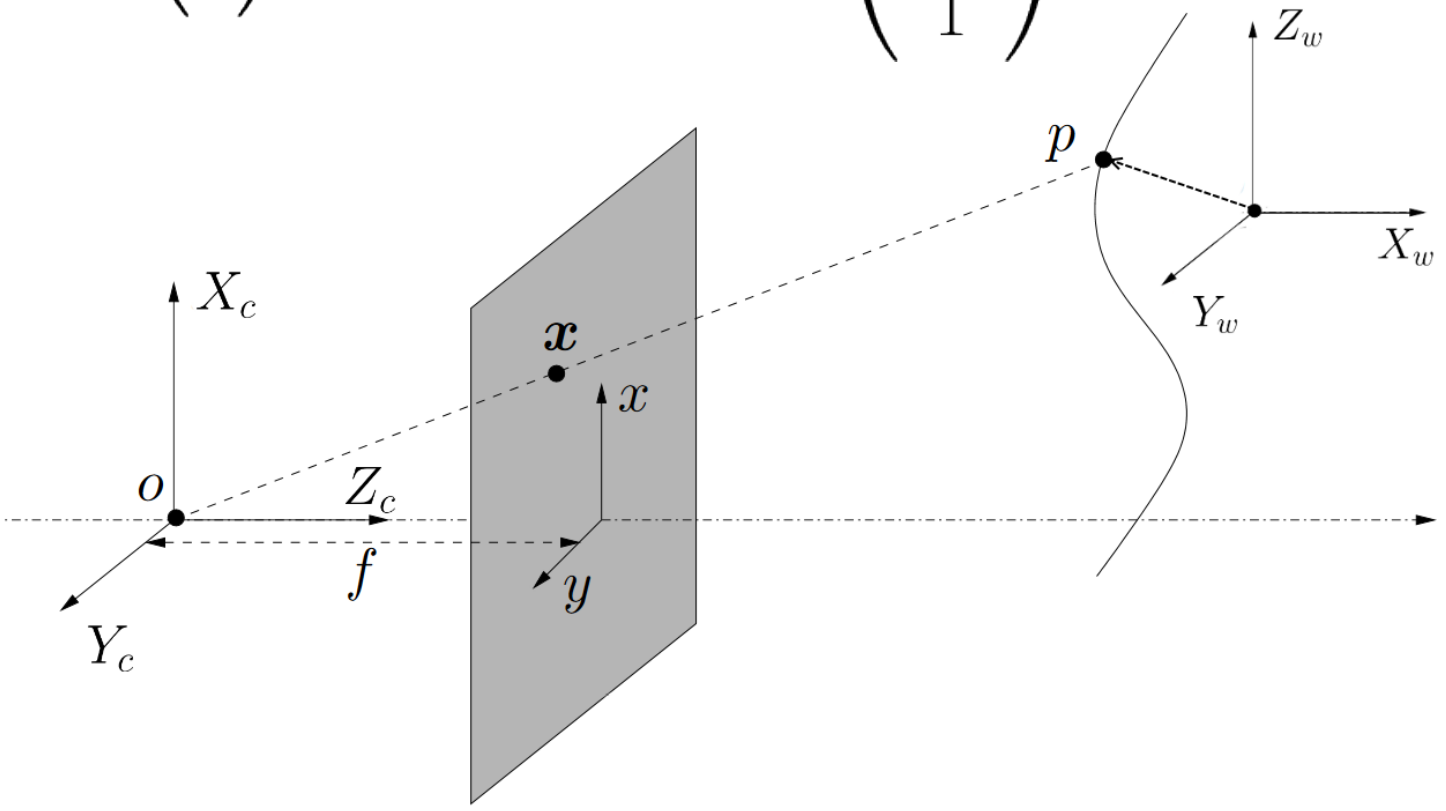


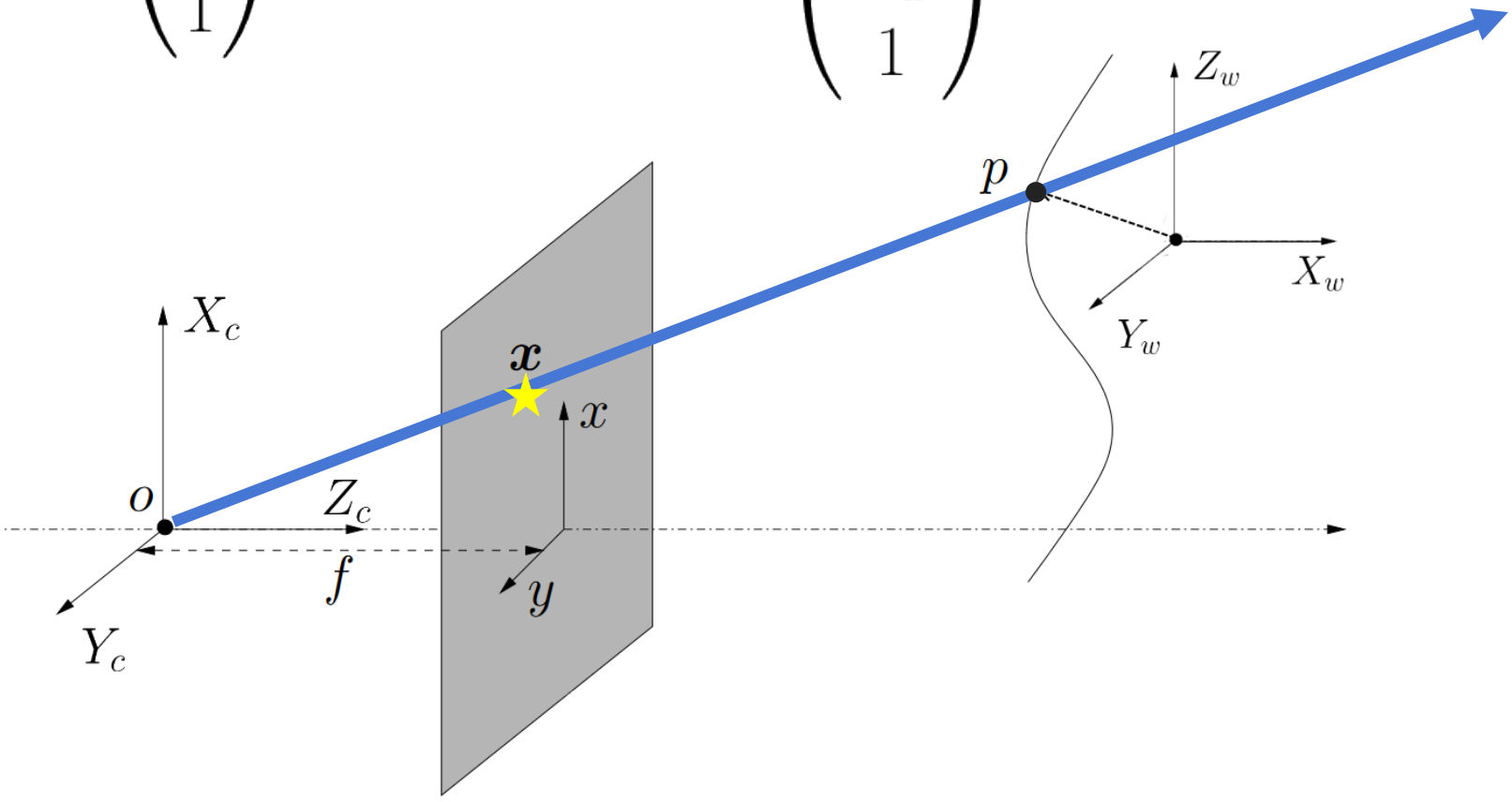
Diagram of Projection Equation

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$



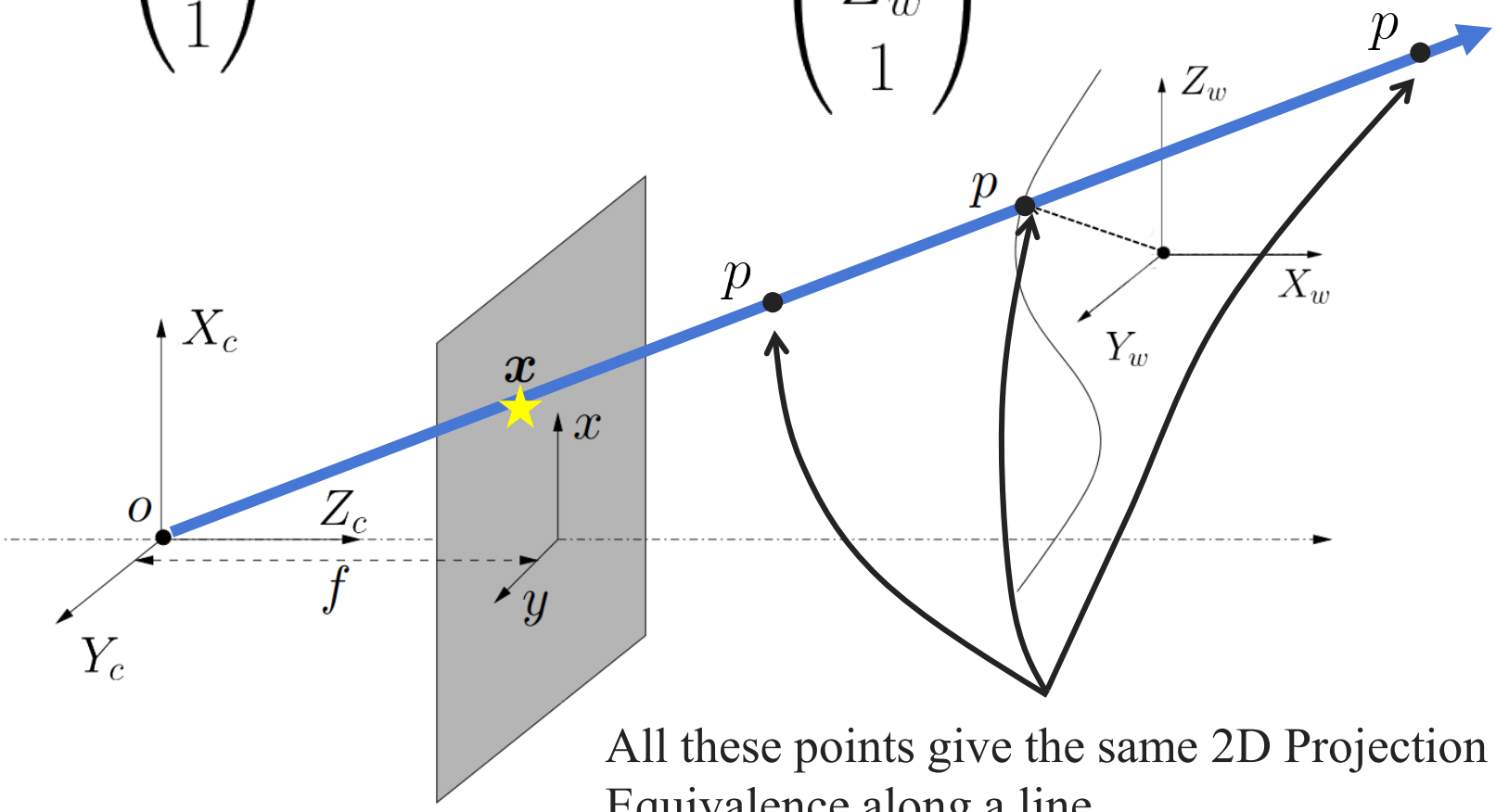
Projective equivalence

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$



Projective equivalence

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$



All these points give the same 2D Projection
Equivalence along a line

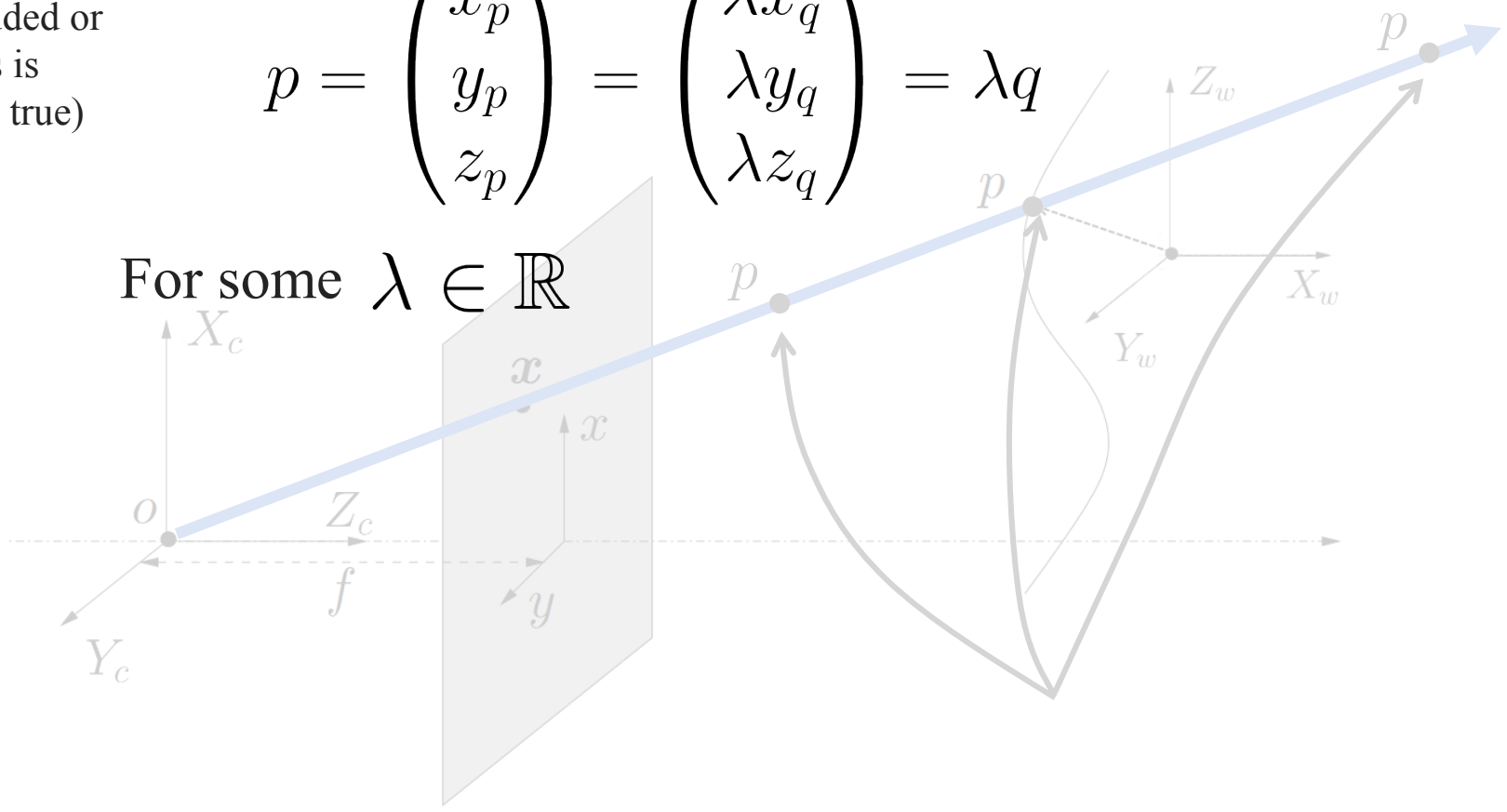
Projective equivalence

Two points p and q in $\mathbb{R}^3 \backslash \{(0,0,0)\}$ are **protectively equivalent** if:

(0 excluded or else this is trivially true)

$$p = \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} \lambda x_q \\ \lambda y_q \\ \lambda z_q \end{pmatrix} = \lambda q$$

For some $\lambda \in \mathbb{R}$



Projective equivalence

Two points p and q in $\mathbb{R}^3 \backslash \{(0,0,0)\}$ are **protectively equivalent** if:

$$p = \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} \lambda x_q \\ \lambda y_q \\ \lambda z_q \end{pmatrix} = \lambda q$$

For some $\lambda \in \mathbb{R}$

This is denoted $p \sim q$ and is an equivalence relation:

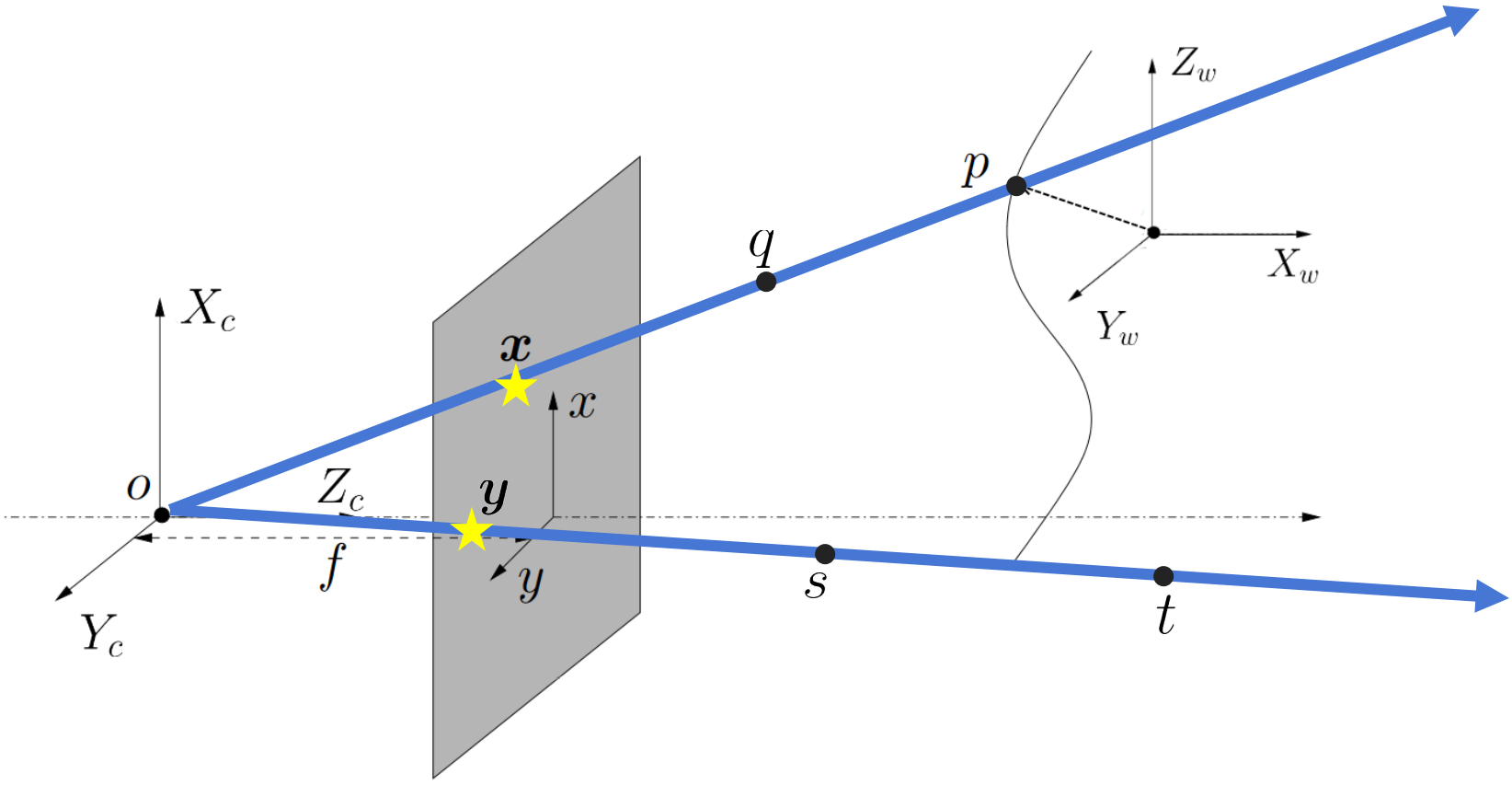
$$p \sim p$$
$$p \sim q \implies q \sim p$$
$$p \sim q, q \sim r \implies p \sim r$$

Note we will be using 'up to a scale' a lot – this is what we mean by that

For all points p, q, r - The set of points equivalent to each other are called an equivalence class

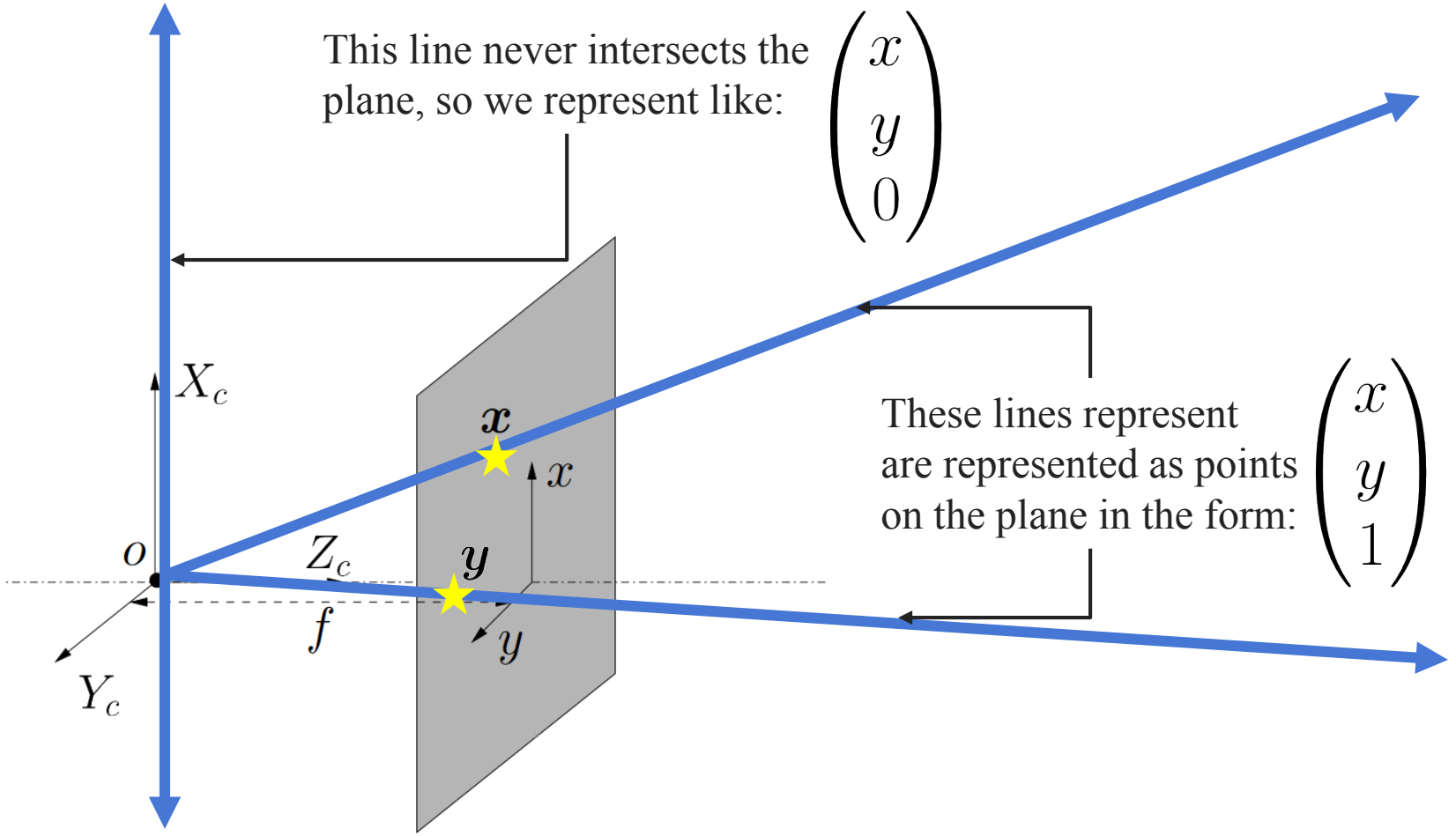
Projective plane

The projective plane (denoted $\mathbb{P}^2$) is the set of all equivalence classes. Alternatively, it is the set of all lines



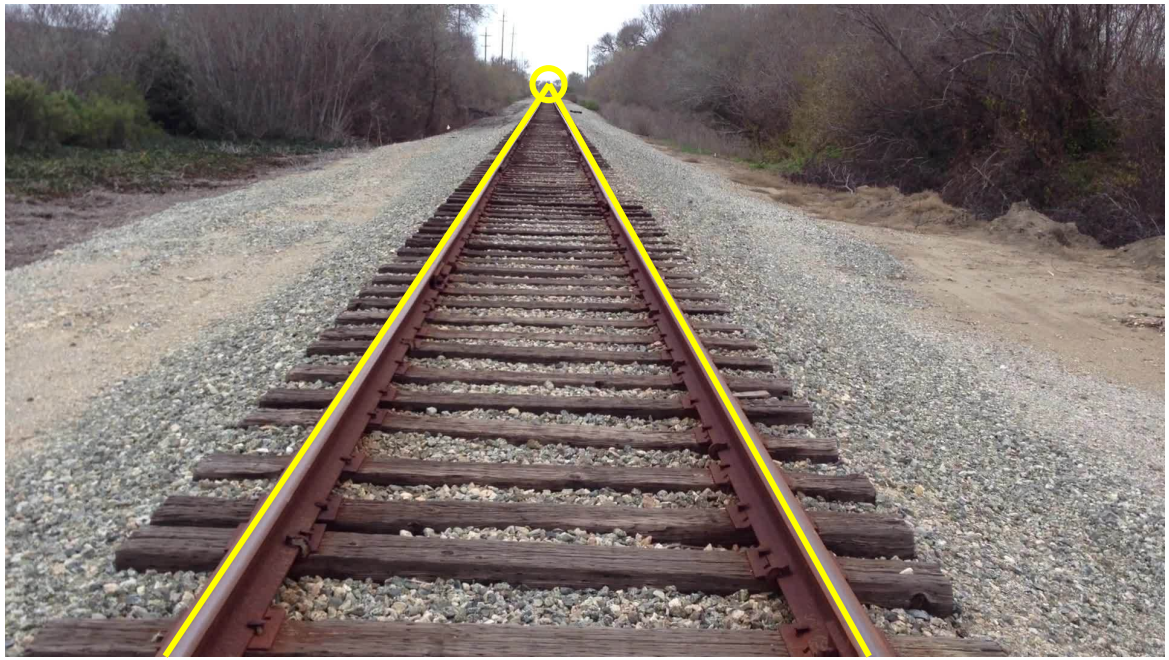
Representing the Projective plane

We can represent all the points on the 2D plane (plane at Z = 1) with the addition of points at infinity



Representing the Projective plane

Recall our example from earlier – from this perspective our lines intersect at some point $(x, y, 1)$



Representing the Projective plane

These lines however intersect at a point at infinity, something of the form $(x', y', 0)$ (Note that $(-x', -y', 0)$ would work as well)

Another View of Projective Space

We can also look at the projective plane as a sphere where antipodal (opposite side) points are considered the same

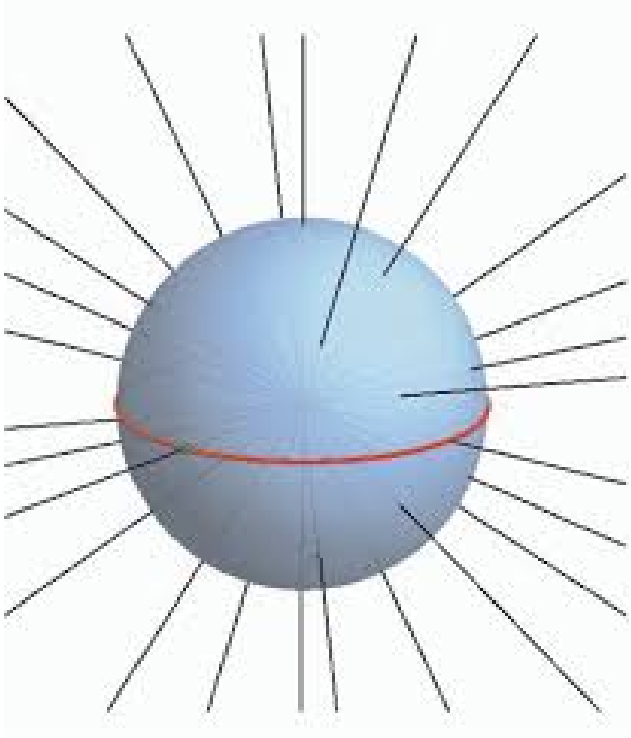


Image from Demiralp et. al. "Coloring 3D line fields using Boy's real projective plane immersion." *IEEE transactions on visualization and computer graphics* 15, no. 6 (2009): 1457-1464.

Equivalence of Points and Lines

In this second view we can more easily see that in projective space, lines in projective space are equivalent to points in projective space

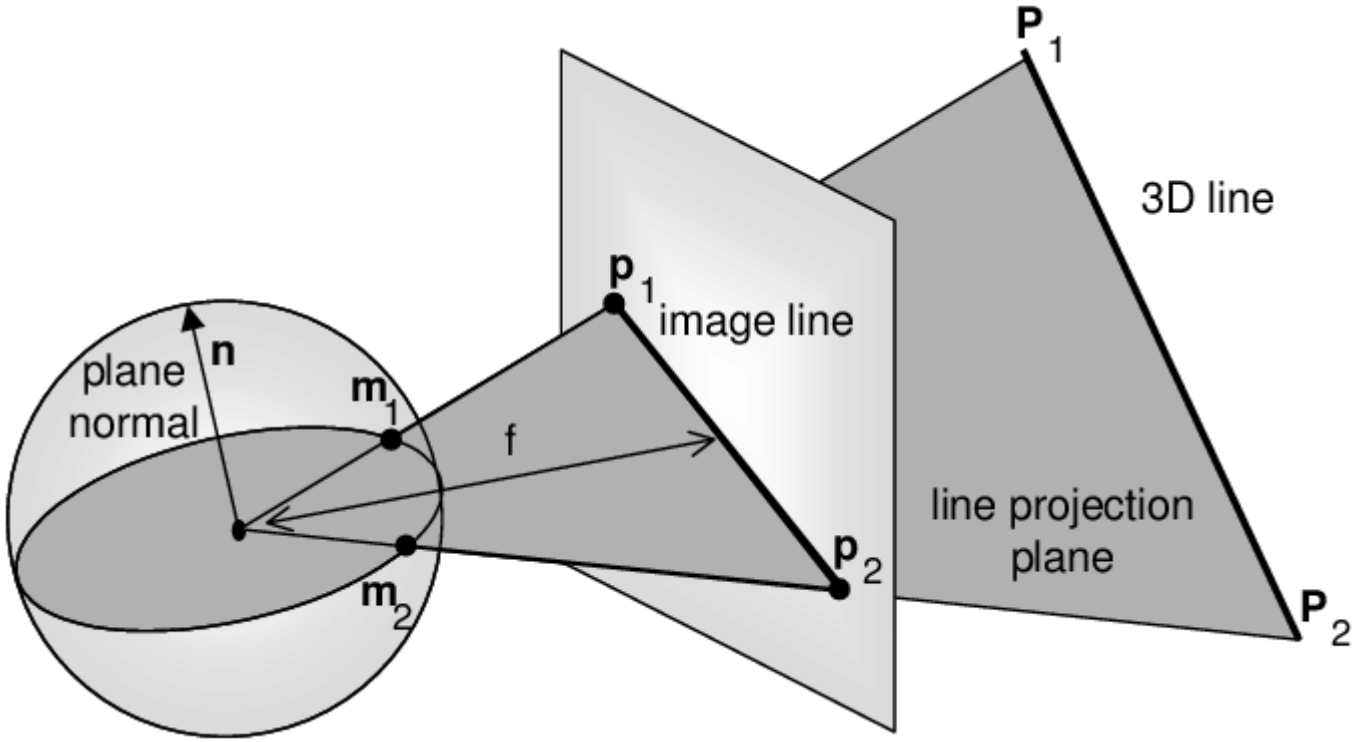
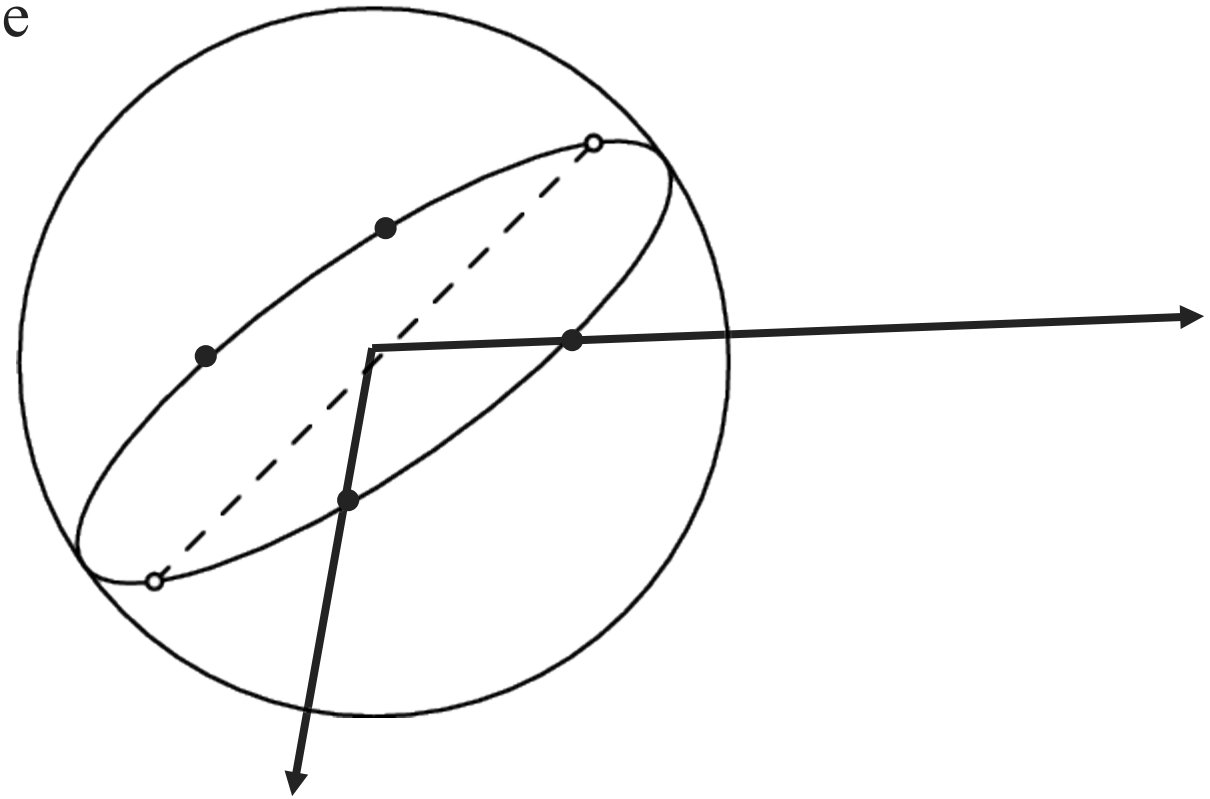


Image from Alves, Joao & Lobo, Jorge & Dias, Jorge. (2019). Camera-Inertial Sensor Modeling and Alignment for Visual Navigation.

Equivalence of Points and Lines

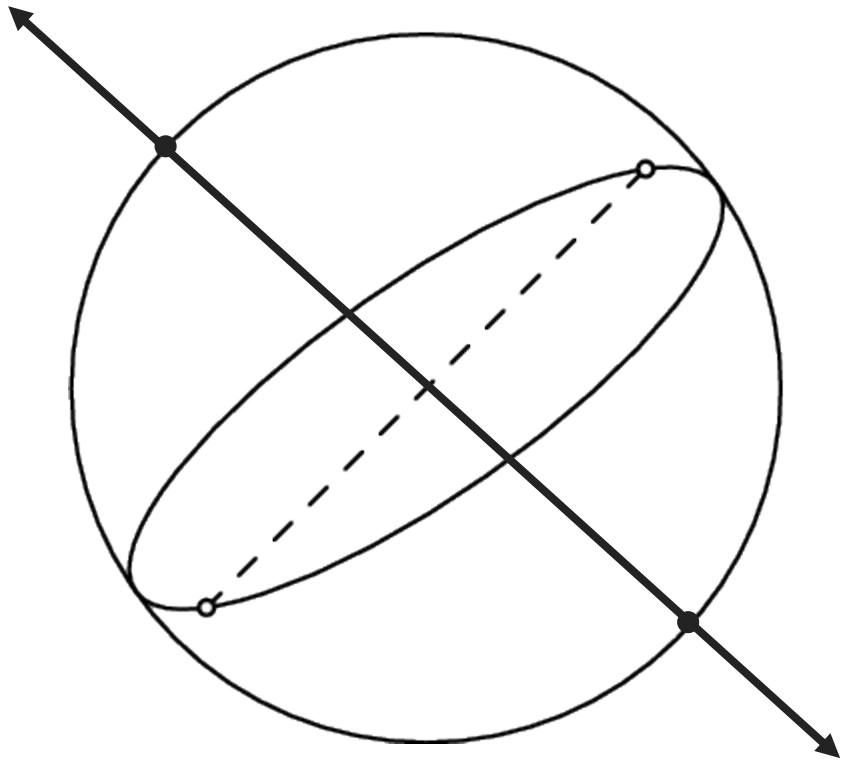
A line on the plane looks like a great circle on the sphere – line is defined by two points and you can draw a great circle through any two points (in our case 2 pairs of antipodal points)
Thus lines correspond to great circles in this view of Projective Space



Equivalence of Points and Lines

A great circle defines two antipodal points on the sphere by its axis – which in projective space is a point

And this is a 1 to 1 relation – every great circle has an axis and every axis has a great circle



Representing Points and Lines

As points and lines are equivalent, they can be represented in the same way – context will typically make it clear which is which.

Recall we represent points using homogenous coordinates – thus so are lines

z is typically 1 when the coordinates are normalized but can be 0 if it is a point at infinity

$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad l = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

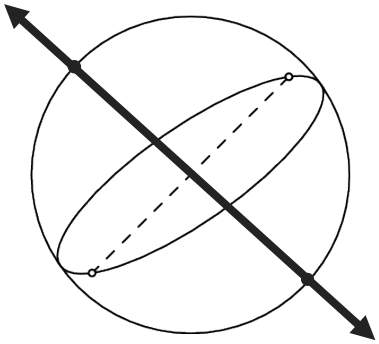
The line represents the axis discussed earlier for the spherical view, or a orientation and offset in the plane view

To find all the points that lie on the line, we can take the dot product to see which are perpendicular to the axis

$$l^T p = 0$$

Finding Points and Lines

With this view in mind, we can more easily see how to get lines from points – find the axis of that great circle



We can do this by finding a vector perpendicular to both points – the easiest way to do that is the cross product

$$l \sim p \times q$$

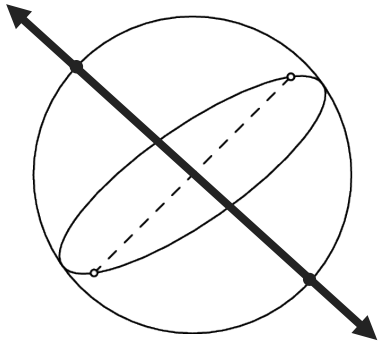
To find the intersection of two lines, as it turns out you can do the same thing – find the point that is perpendicular to both axes of the great circles

$$p \sim l \times l'$$

p, q - points, l, l' - lines

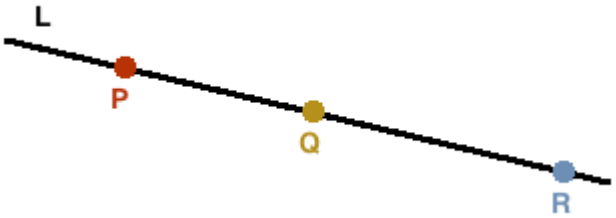
Finding Points and Lines

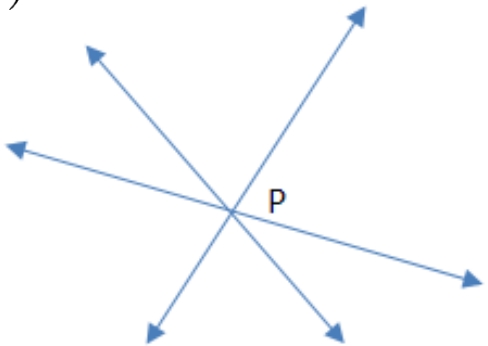
We can also find if 3 points are collinear – check to see if a third point is perpendicular to the same axis as the other two. Similarly with checking if 3 lines are concurrent (match at a point)



We can do this by testing the points using the triple product:

$$p^T(q \times r) = q^T(r \times p) = r^T(p \times q) = 0$$
$$l^T(l' \times l'') = l'^T(l'' \times l) = l''^T(l \times l') = 0$$



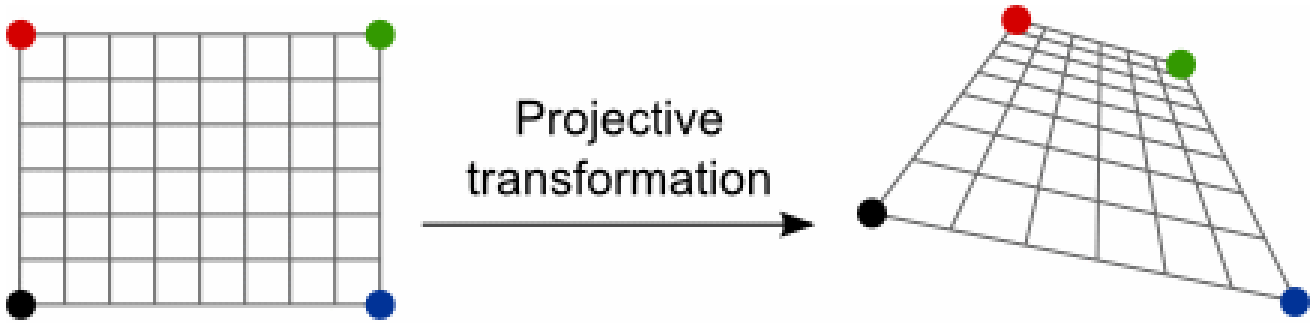


p, q, r - points, l, l', l'' - lines

Projective Transformations

A **projective transformation** or **homography** is any invertible matrix transformation

$$Ap \sim p'$$



Projective Trans. Properties

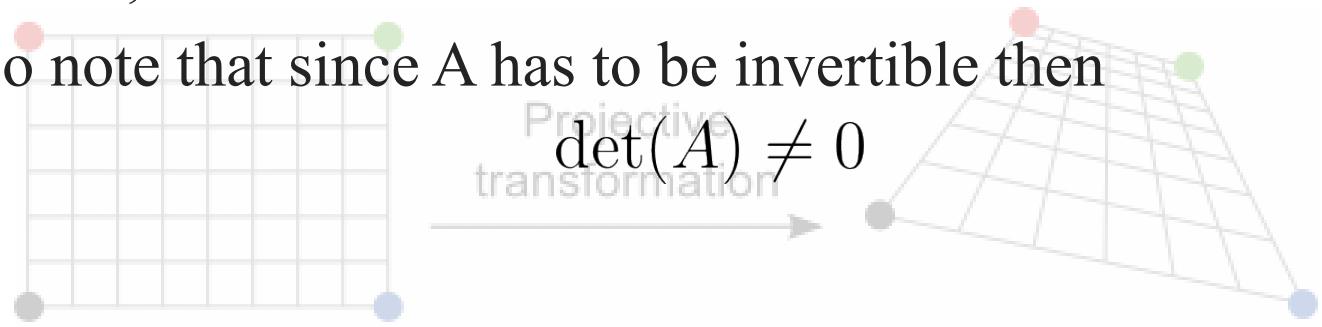
Note that the matrix A can only be determined up to a scale, since

$$Ap = \lambda p' \implies \alpha Ap = \alpha \lambda p', \quad \forall \alpha \neq 0$$

Meaning A and αA represent the same transformation. This implies that our transformation has only 8 degrees of freedom, not 9.

Also note that since A has to be invertible then

$$\det(A) \neq 0$$



Projective Trans. Properties

One fundamental property (and an alternative way of defining it) is that it maps lines to lines.

$$l^T p = 0, \quad p' \sim Ap \implies l^T A^{-1} p' \implies l \to A^{-T} l$$

It also preserves incidence relationships i.e. three colinear points remain collinear, three concurrent lines remain concurrent (proof same for lines and points)

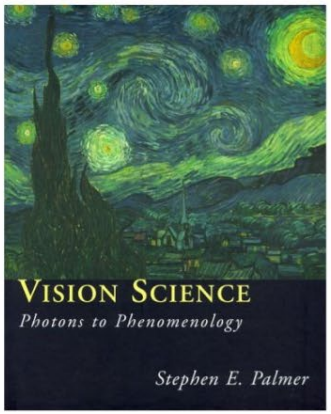
$$\begin{aligned} & p^T (q \times r) = 0 \\ \implies & (Ap)^T (Aq \times Ar) \\ = & \, p^T A^T (\det(A) A^{-T})(q \times r) \\ = & \det(A) p^T (q \times r) = 0 \end{aligned}$$

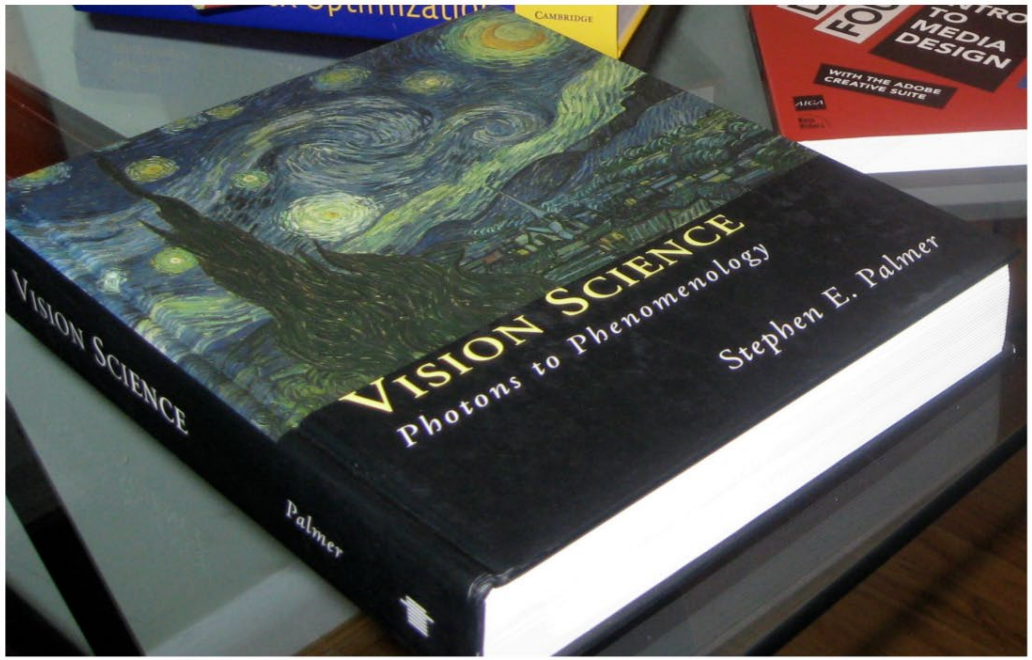
A useful property of cross products

Perspective Projection of Planes

Why we care:

The perspective projection of a plane is always a projective transformation





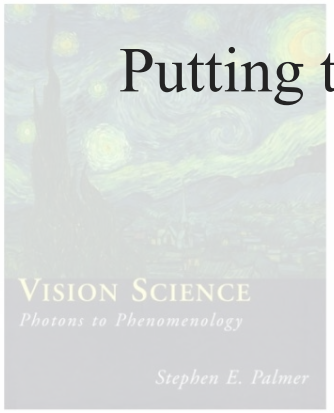
Perspective Projection of Planes

Why is this?

Generic Proof (if plane is not centered at origin):

Equation of a plane: $$n^T P_w = d \iff \frac{n^T P_w}{d} = 1$$

Point in Camera Frame: $$P_c = {}^cR_w P_w + {}^cT_w$$



Putting them together:

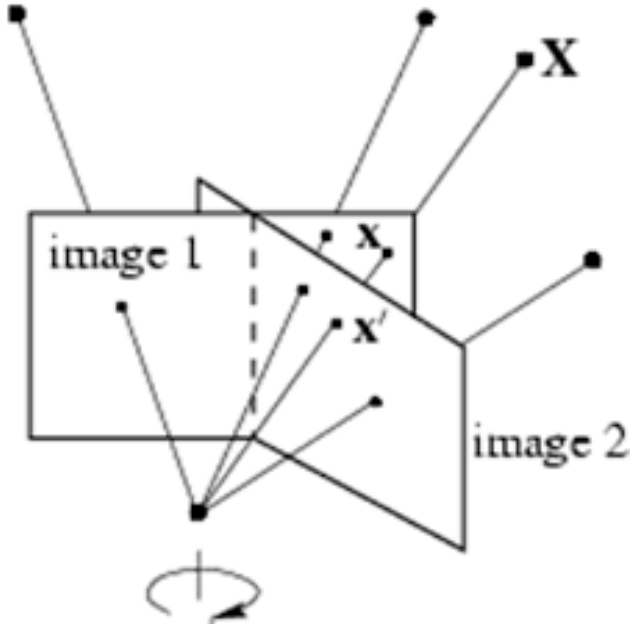
$$\begin{aligned} P_c &= {}^cR_w P_w + {}^cT_w(1) \\ &= {}^cR_w P_w + {}^cT_w \frac{n^T P_w}{d} \\ &= \left({}^cR_w + \frac{{}^cT_w n^T}{d} \right) P_w \end{aligned}$$

Perspective Proj. with Pure Rotation

Pure rotations also are described by Projective Transformations

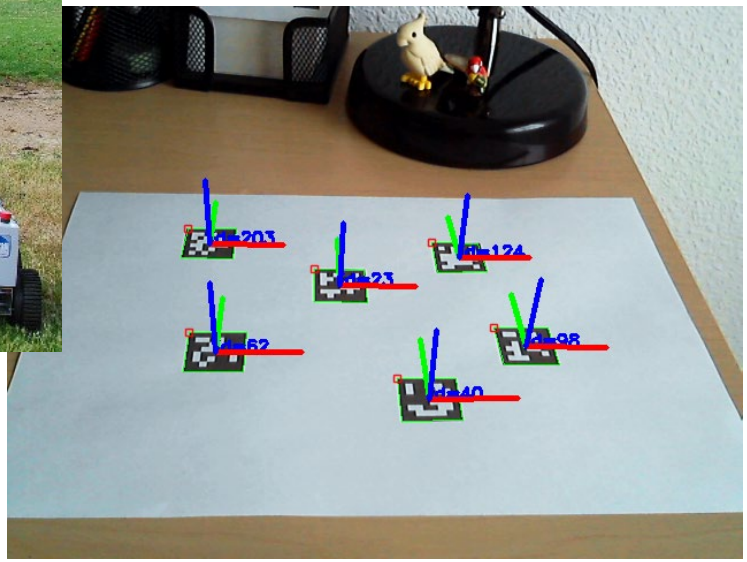
$$P_c = {}^cR_w P_w$$

Only a matrix transform – this is the projective transformation



Application: AprilTags

Can use easily identifiable markers (similar to QR codes) with known dimensions. We can use that to find the position and orientation of things in the world, or ourselves relative to it

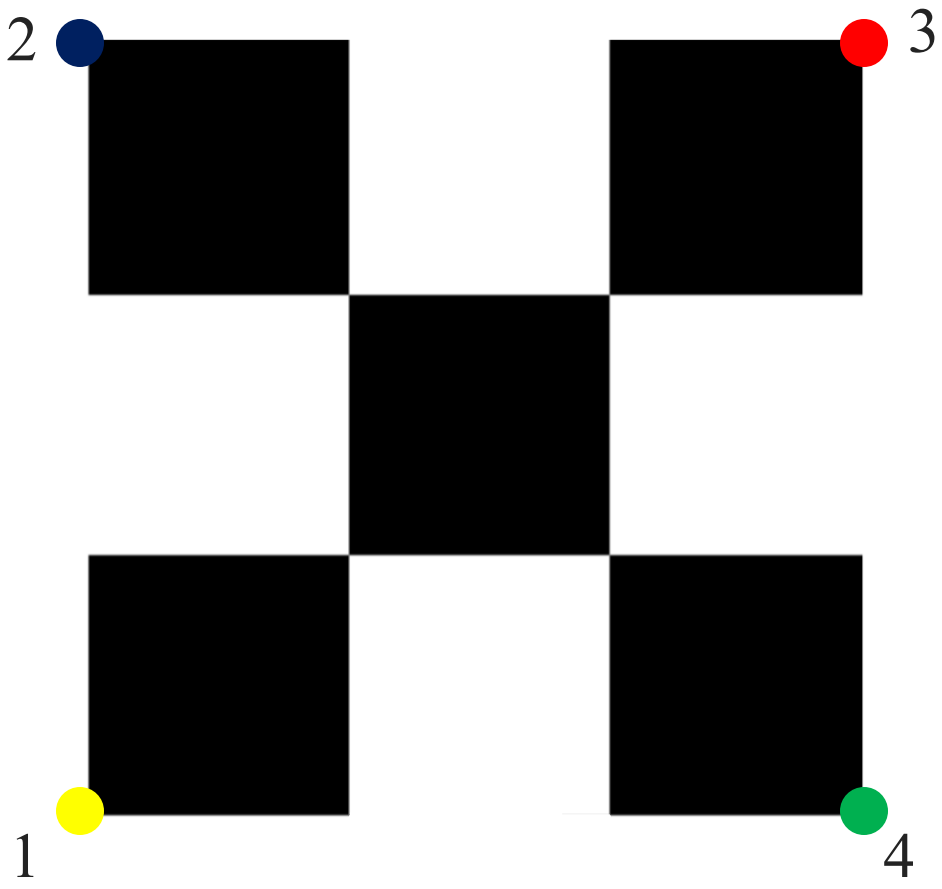


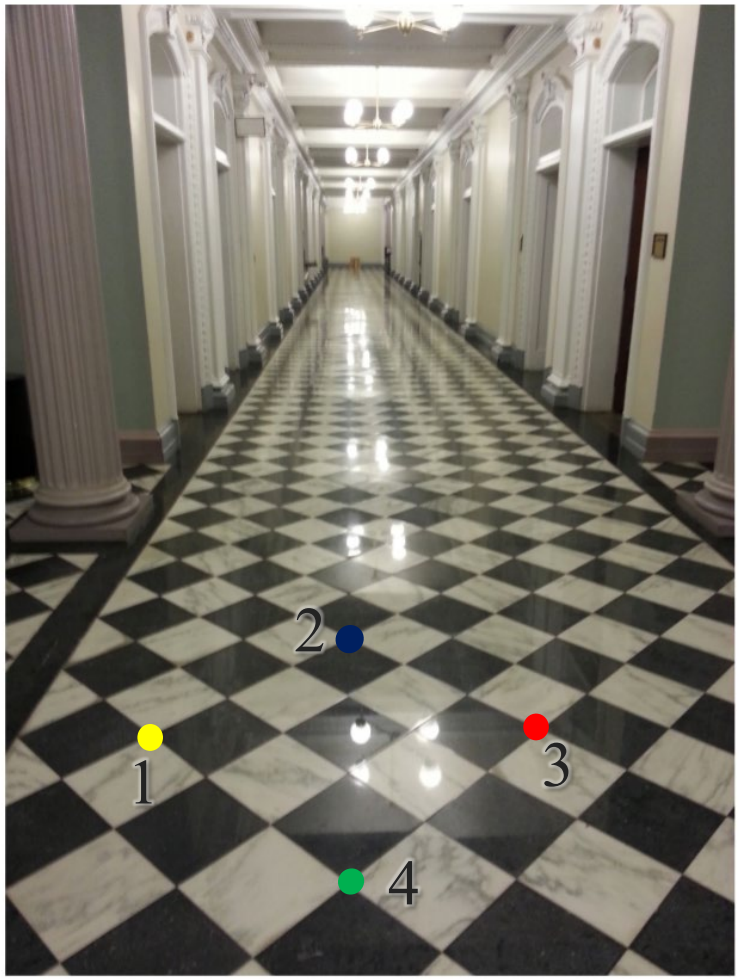
Application: Panoramas

If you are looking at objects far enough away, rotating your camera around can get you an approximately pure rotation, thus allowing you to stitch together images via Projective Transforms

Computing a Transformation

If we want to compute the transformation between a reference image and a new image





Computing a Trans.: Version #1

If we have direct point correspondences we can compute it that way

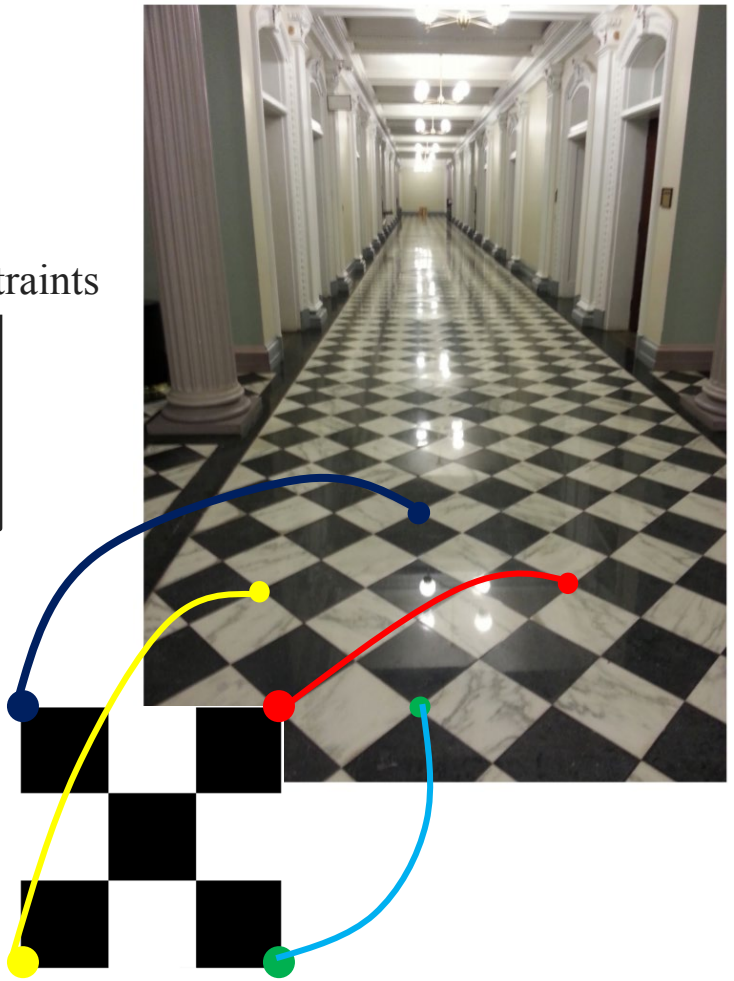
Each point correspondence gives 2 constraints (3 linear constraints, minus 1 for scale ambiguity)

$$\lambda_i \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

3 Constraints

Unknown scale

Unknown transformation parameters



Computing a Trans.: Version #1

We can simplify this to remove the unknown scale parameter:

$$\lambda_i \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

Can solve this directly

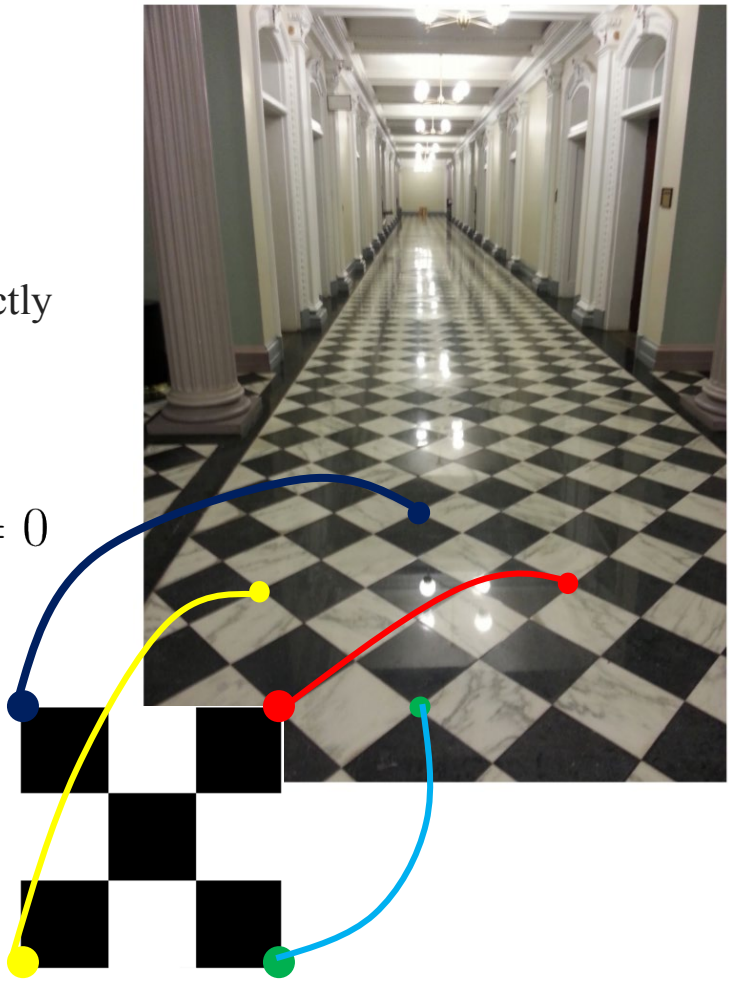
A few algebra tricks later…

$$\begin{pmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{pmatrix} \boxed{h} = 0$$

Vector of unknown transformation parameters

Need 4 of these to get all 8 degrees of freedom

In general a Projective transformation can map any 4 points to any 4 points



Computing a Trans.: Version #1

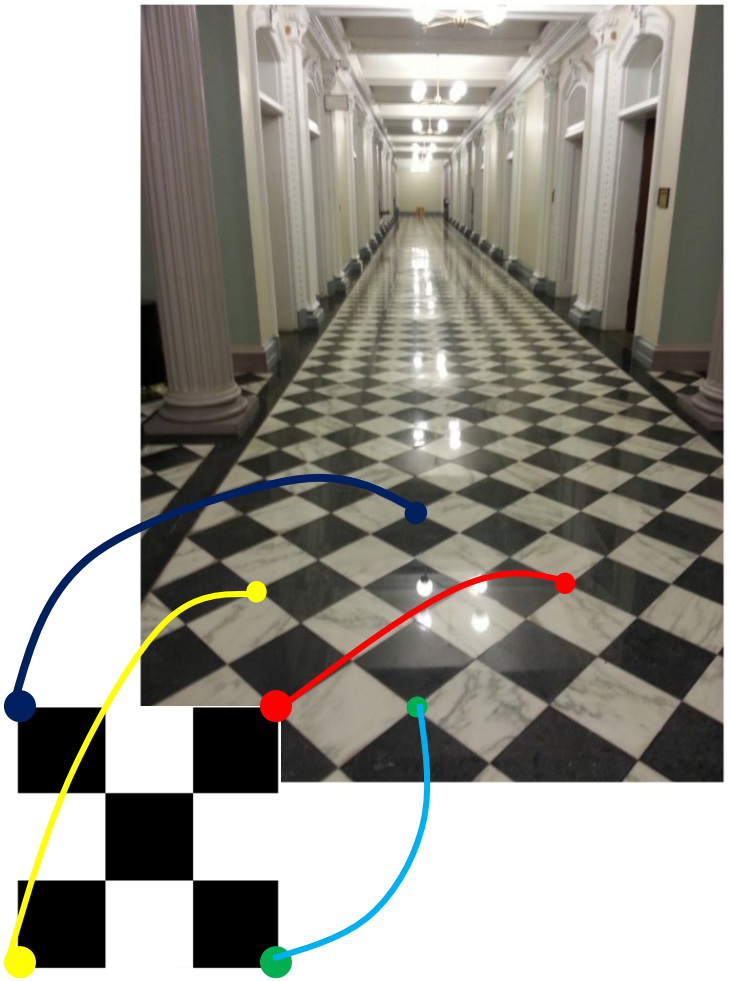
After getting 4 correspondences we solve an equation like the following:

$$Ah = 0$$

Solve using SVD with the smallest singular value

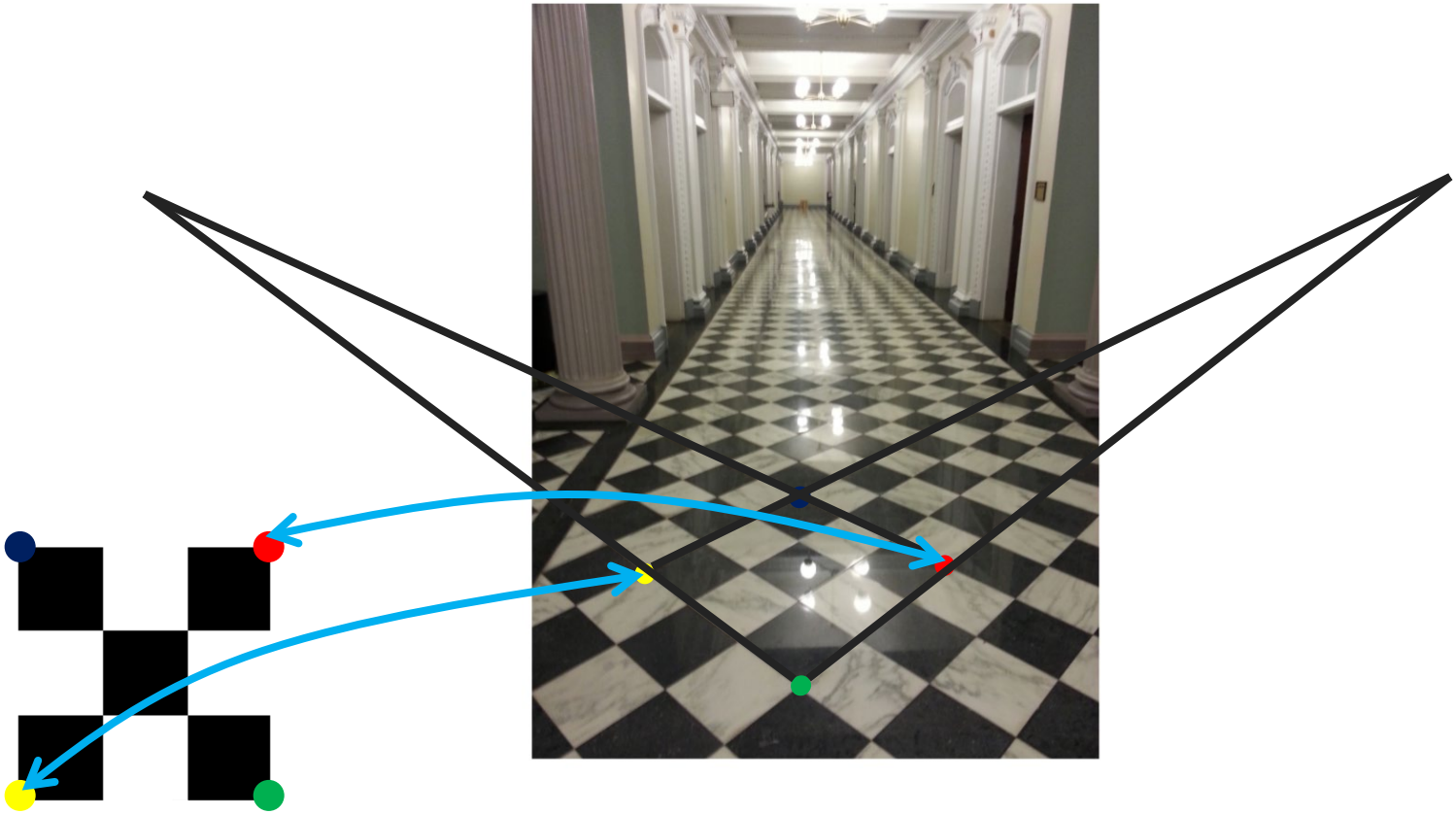
$$A = USV^T \implies h = \boxed{V_9}$$

9^{th} Column of V



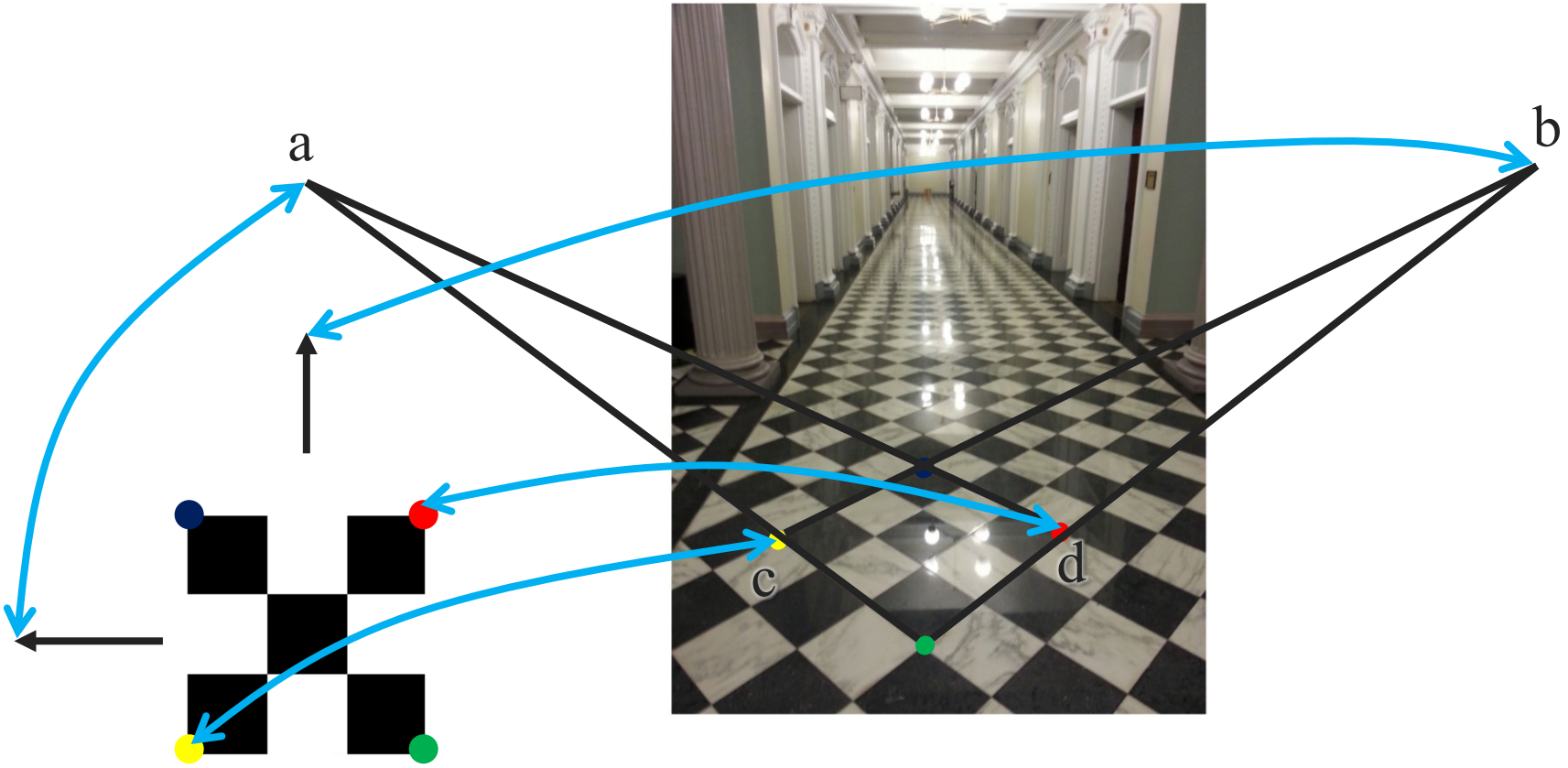
Computing a Trans.: Version #2

Maybe we don't have all direct point correspondences but have some correspondences for points at infinity (vanishing points)



Computing a Trans.: Version #2

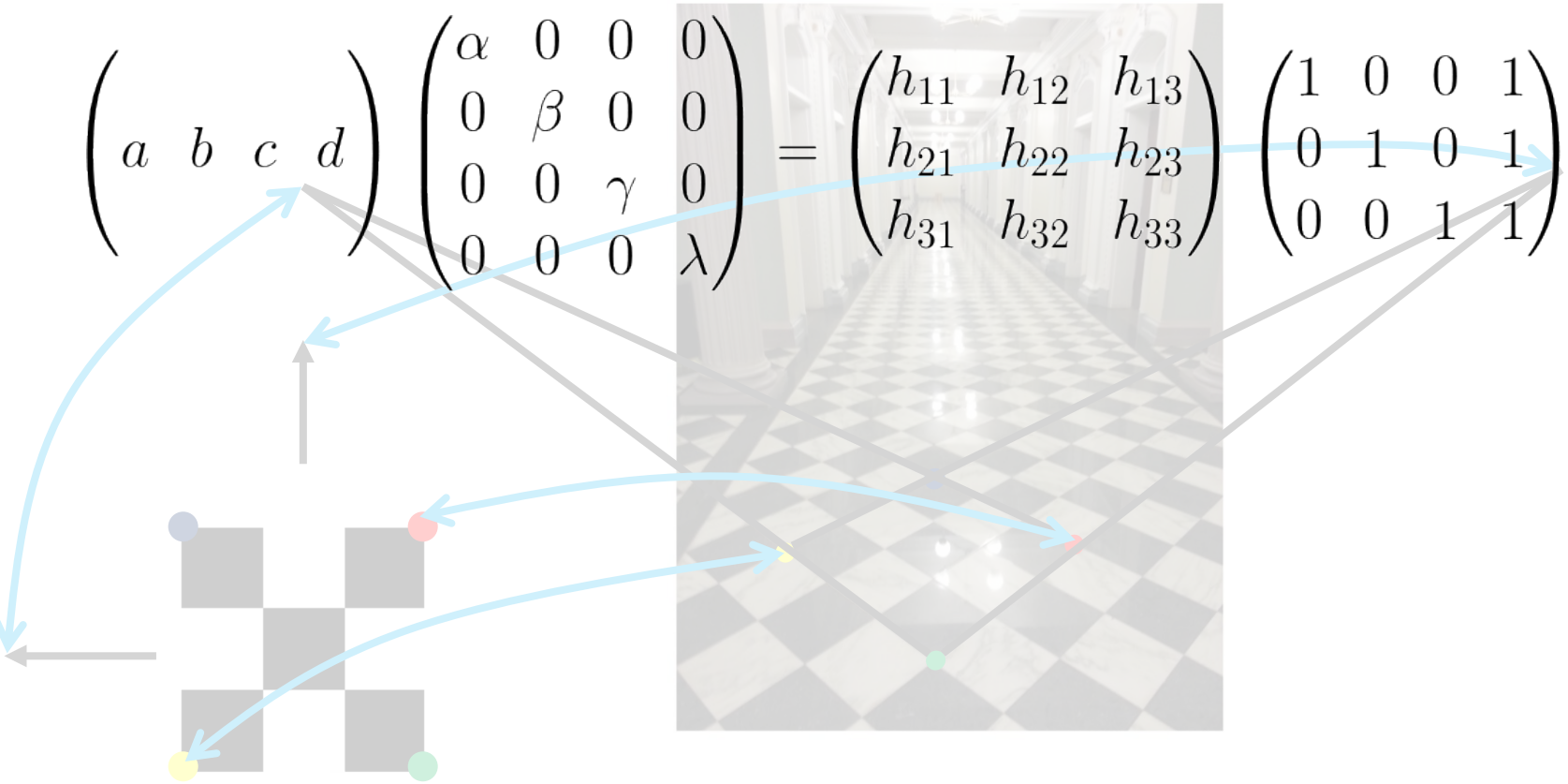
Maybe we don't have all direct point correspondences but have some correspondences for points at infinity (vanishing points)



Computing a Trans.: Version #2

With the vanishing points we need to take a slightly different approach, since the last element of the vector is zero. Assume we have the vanishing points corresponding to the system:

$$\begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & \lambda \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$



Computing a Trans.: Version #2

With the vanishing points we need to take a slightly different approach, since the last element of the vector is zero. Assume we have the vanishing points corresponding to the system:

$$\begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & \lambda \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

From this we can see that:

$$\begin{pmatrix} \alpha a & \beta b & \gamma c \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

So we just need to solve for the three scales, up to a global scale factor

Computing a Trans.: Version #2

Our particular choice of *d* in this case is just the sum of the other vectors:

$$\lambda d = H \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = H \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + H \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + H \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \alpha a + \beta b + \gamma c$$

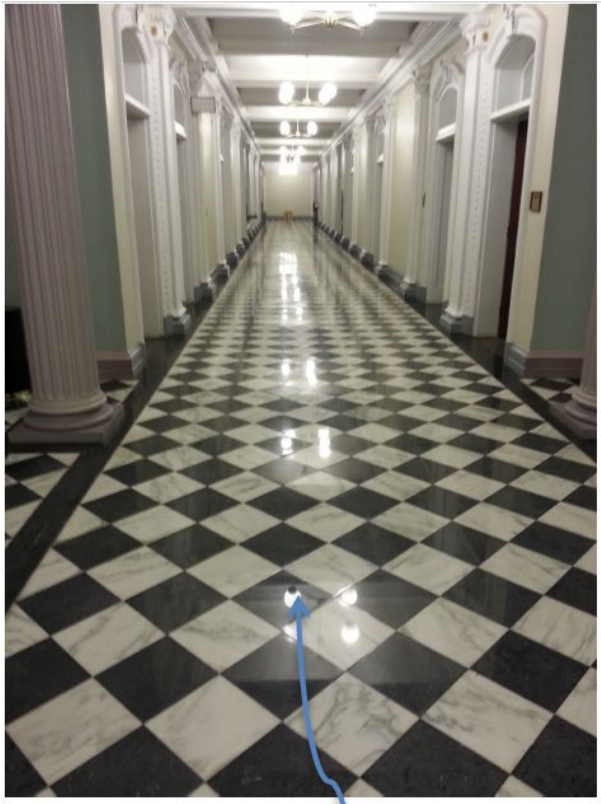
$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

Since all the scale factors are ambiguous up to a global scale, we set $\lambda = 1$

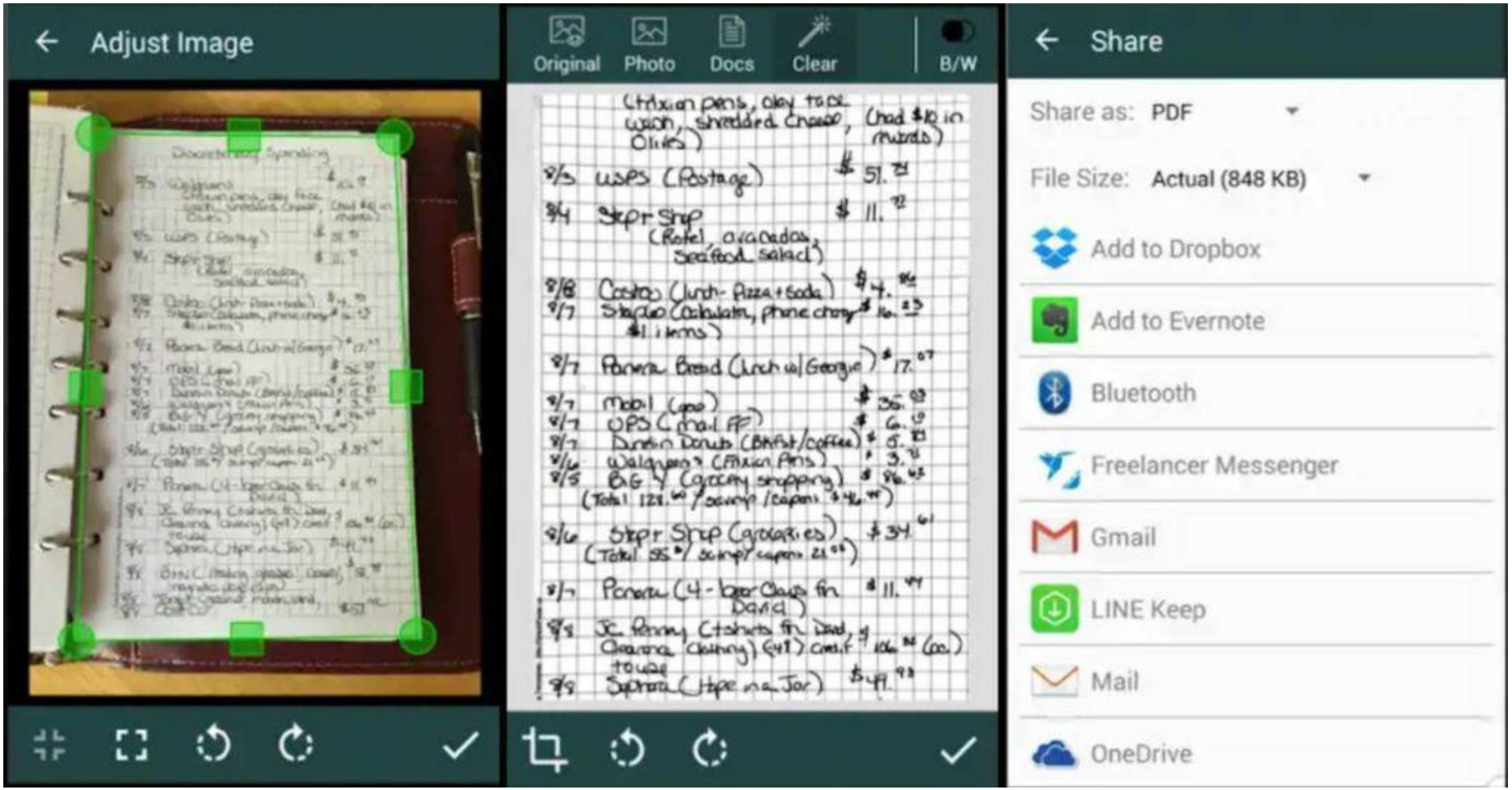
We can write this as solving a linear system, and we have all we need:

$$d = \begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

Applying a Transformation



Applying a Transformation



Next Time

Question: What can we do with the projective transformation once we have it?

Answer: We can get our pose relative to the plane!

We'll be going over how to compute that next time.

Perspective Projection of Planes

Alternative Proof (applicable in all scenarios):

Equation of a plane: $P_w = \alpha U_w + \beta V_w + T_{plane}$

Point in Camera Frame: $P_c = {}^cR_w P_w + {}^cT_w$

Putting them together:

$$P_c = {}^cR_w P_w + {}^cT_w$$

$$\Longrightarrow P_c = {}^cR_w(\alpha U_w + \beta V_w + T_{plane}) + {}^cT_w$$

$$\Longrightarrow P_w = \begin{pmatrix} {}^cR_w U_w & {}^cR_w V_w & {}^cT_w + T_{plane} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ 1 \end{pmatrix}$$