Lecture 11

Camera calibration

Administrative

A3 is out

- Due Nov 12

Administrative

Recitation this friday

- Ontologies

So far: 2D Transformations with homogeneous coordinates

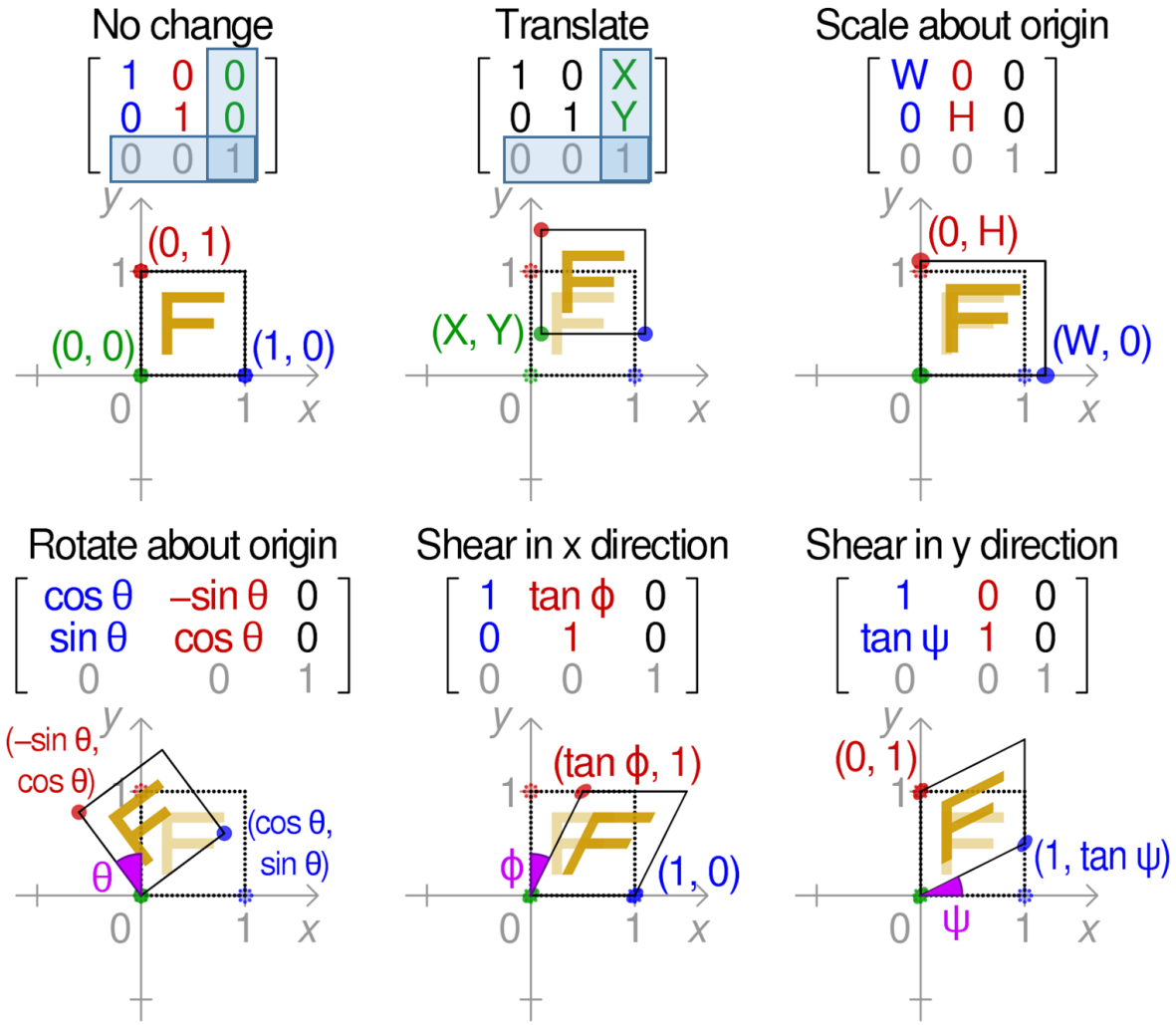
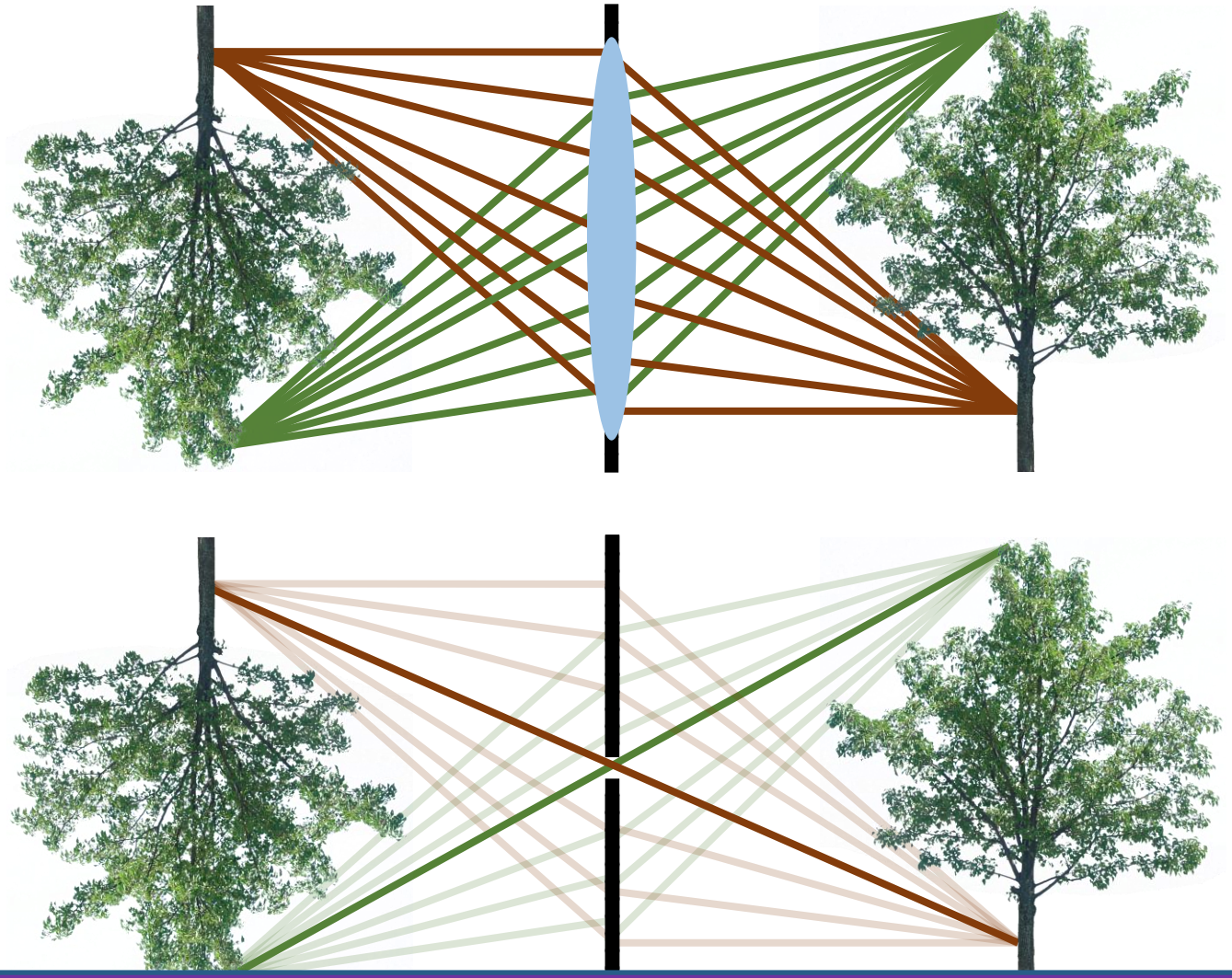


Figure: Wikipedia

So far: The pinhole camera



For this course, we focus on the pinhole model.

- Similar to thin lens model in Physics: central rays are not deviated.
- Assumes lens camera in focus.
- Useful approximation but ignores important lens distortions.

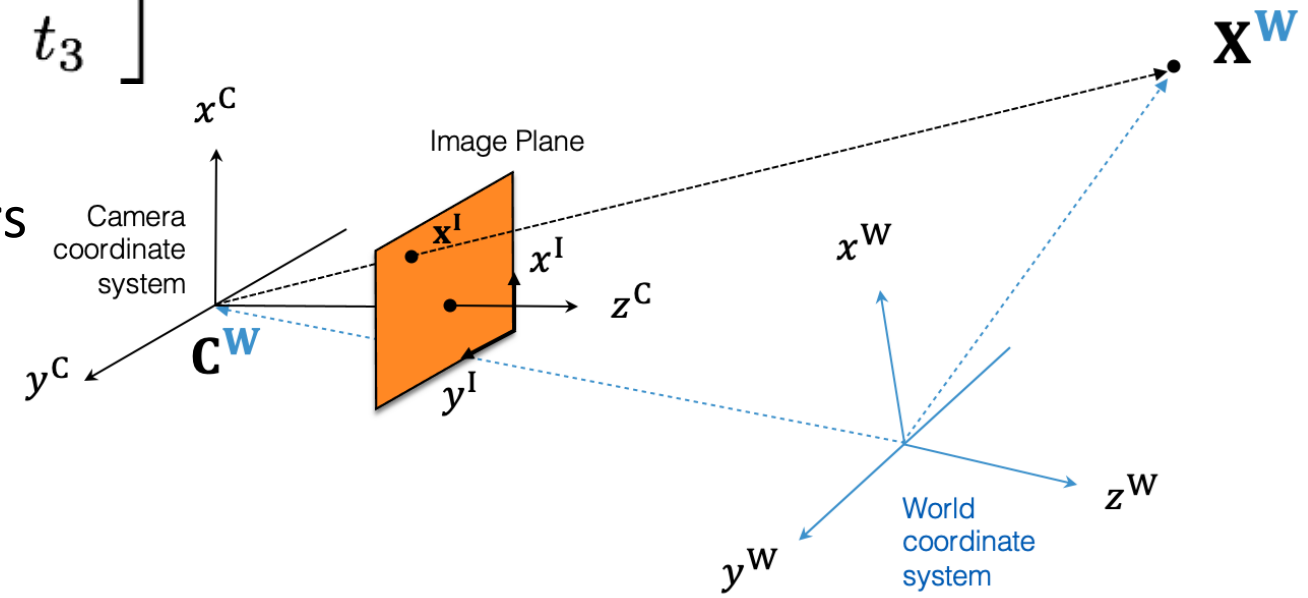
So far: General pinhole camera matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \qquad \text{where} \qquad \mathbf{t} = -\mathbf{RC}$$

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{bmatrix}$$

intrinsic parameters

extrinsic parameters



Today’s agenda

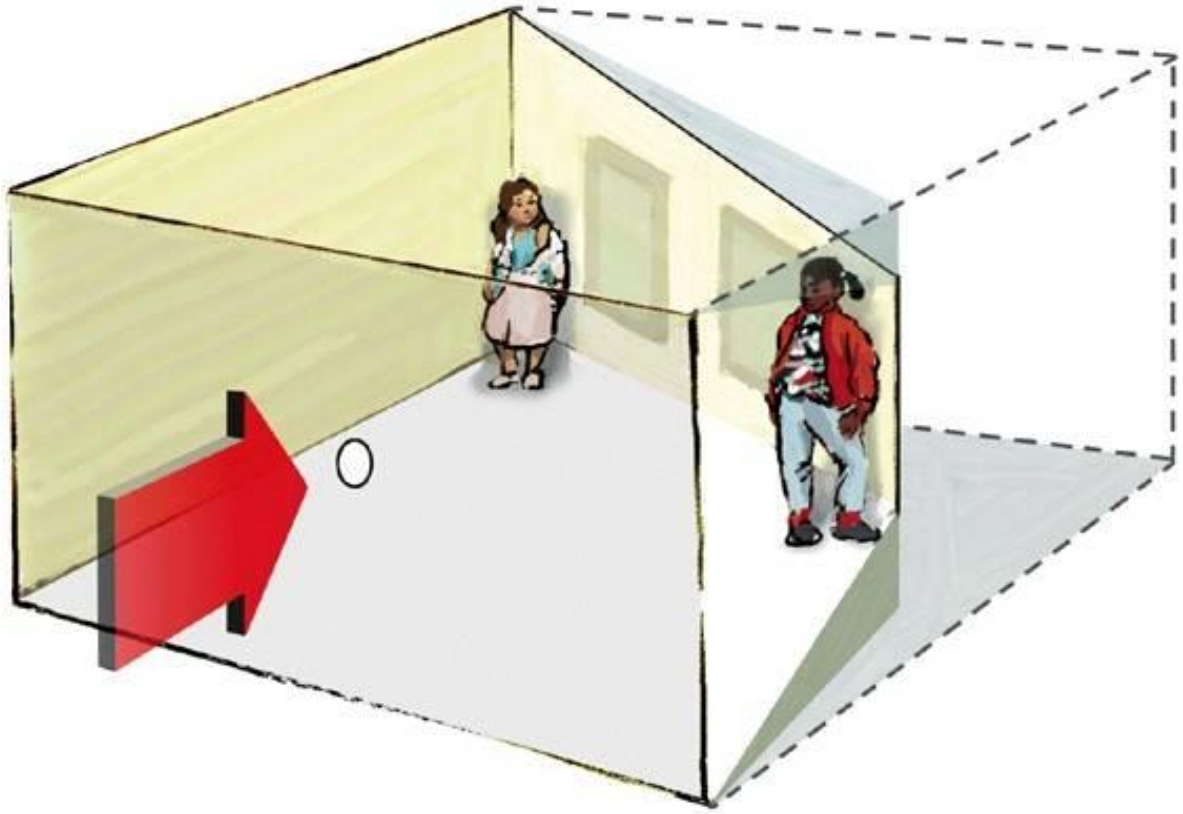
- Properties of Perspective transformations
- Introduction to Camera Calibration
- Linear camera calibration method
- Calculating intrinsics and extrinsics
- Depth estimation

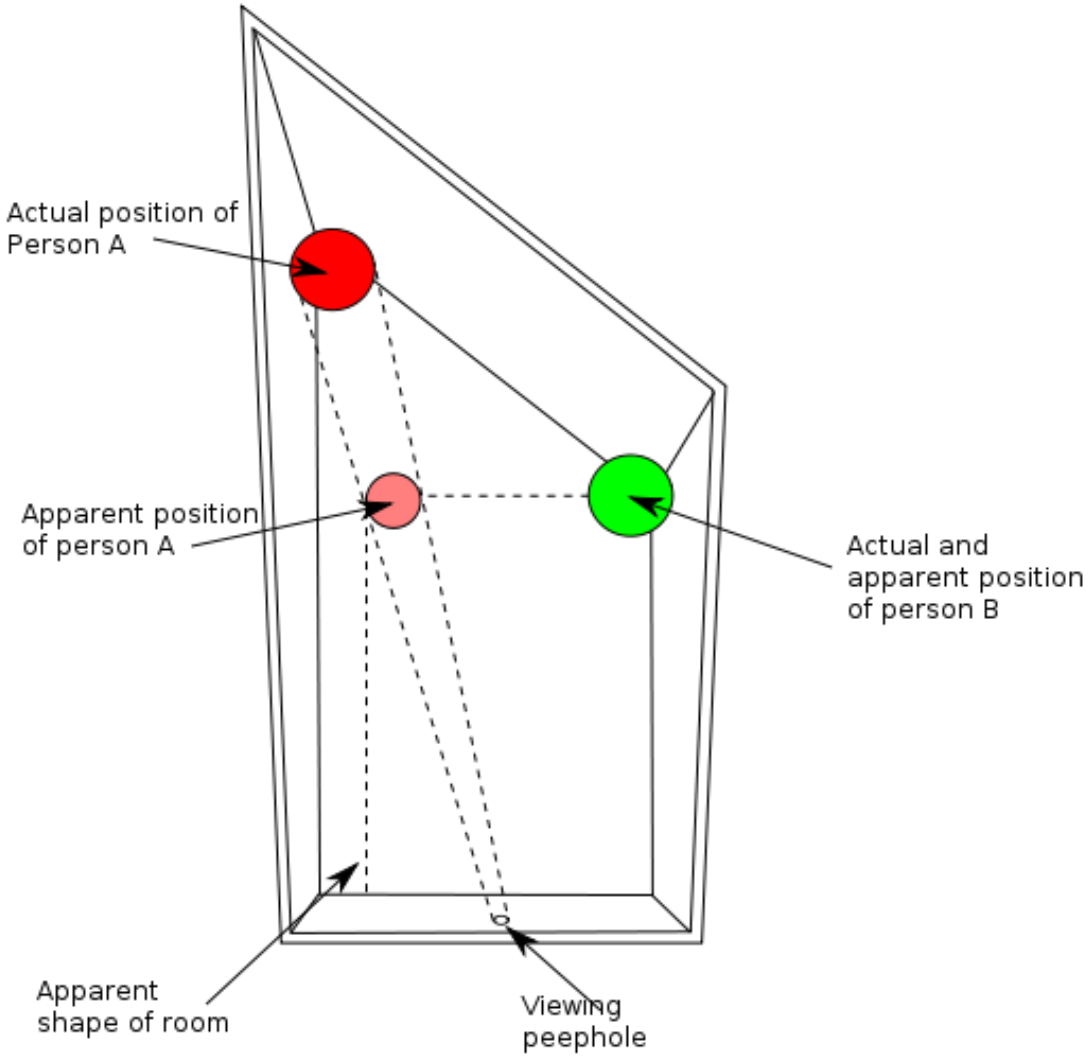
Today’s agenda

- Properties of Perspective transformations
- Introduction to Camera Calibration
- Linear camera calibration method
- Calculating intrinsics and extrinsics
- Depth estimation

Similar illusion as last lecture

The Ames room illusion

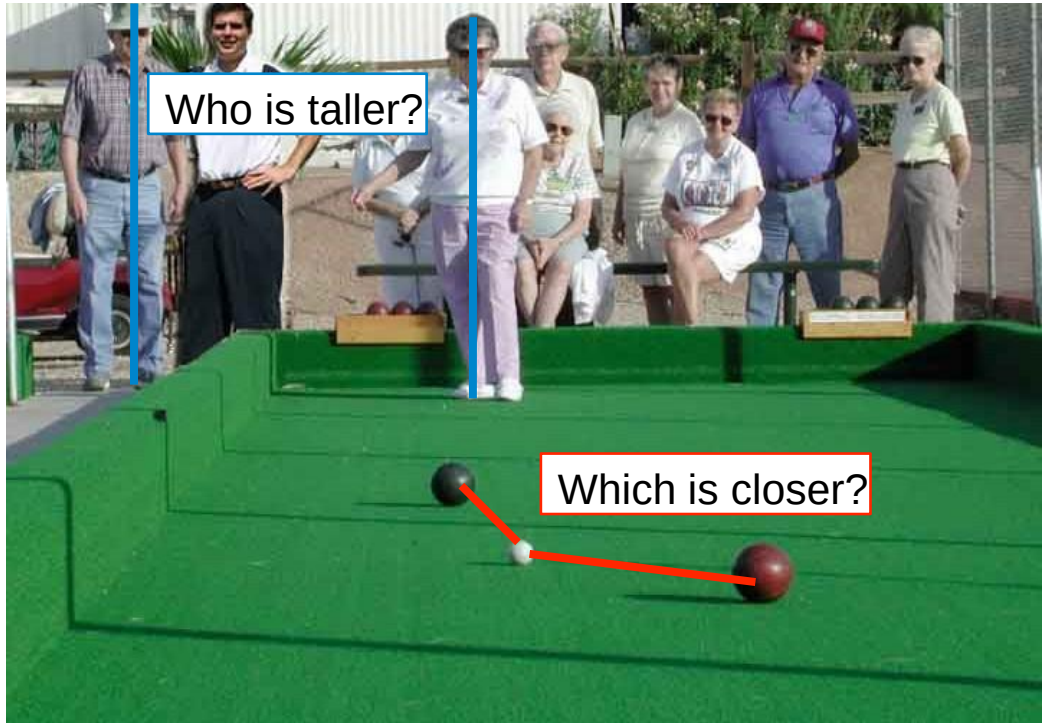




Projective Geometry

Q. Who is taller?

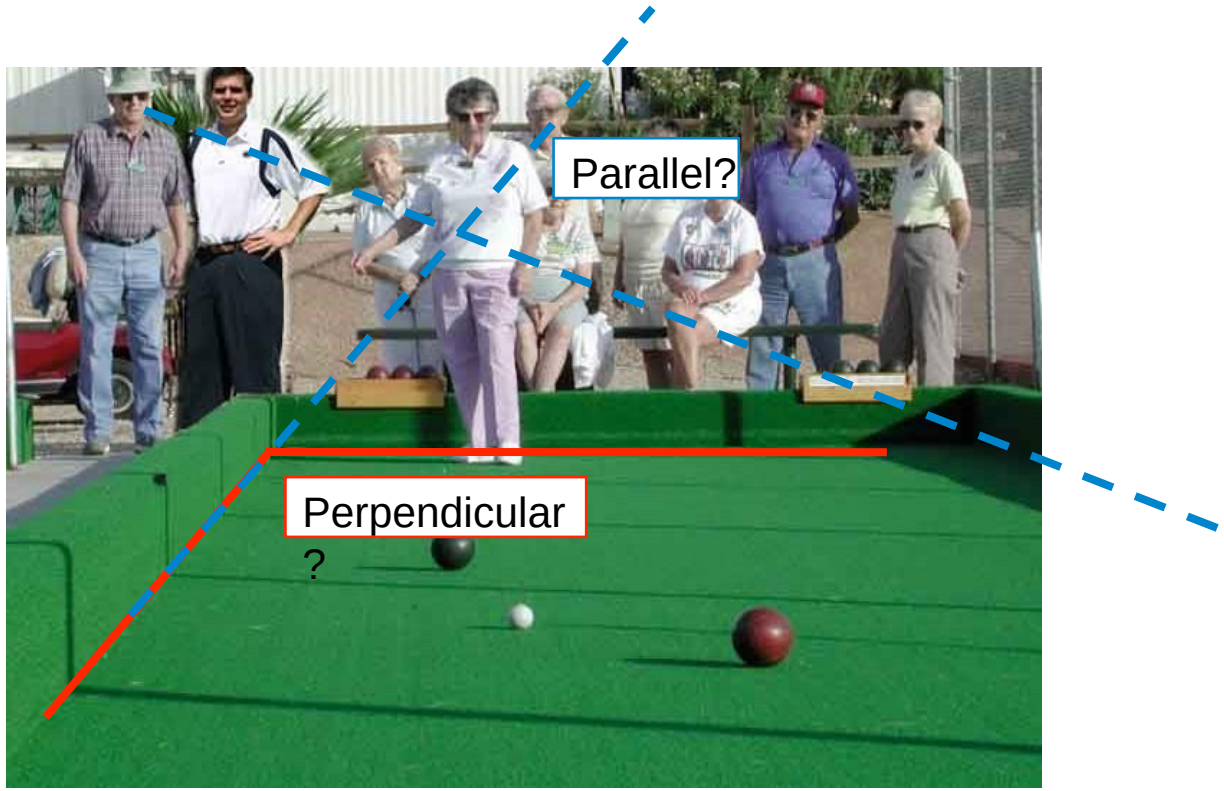
The two blue lines are the same length



Projective Geometry

What is not preserved?

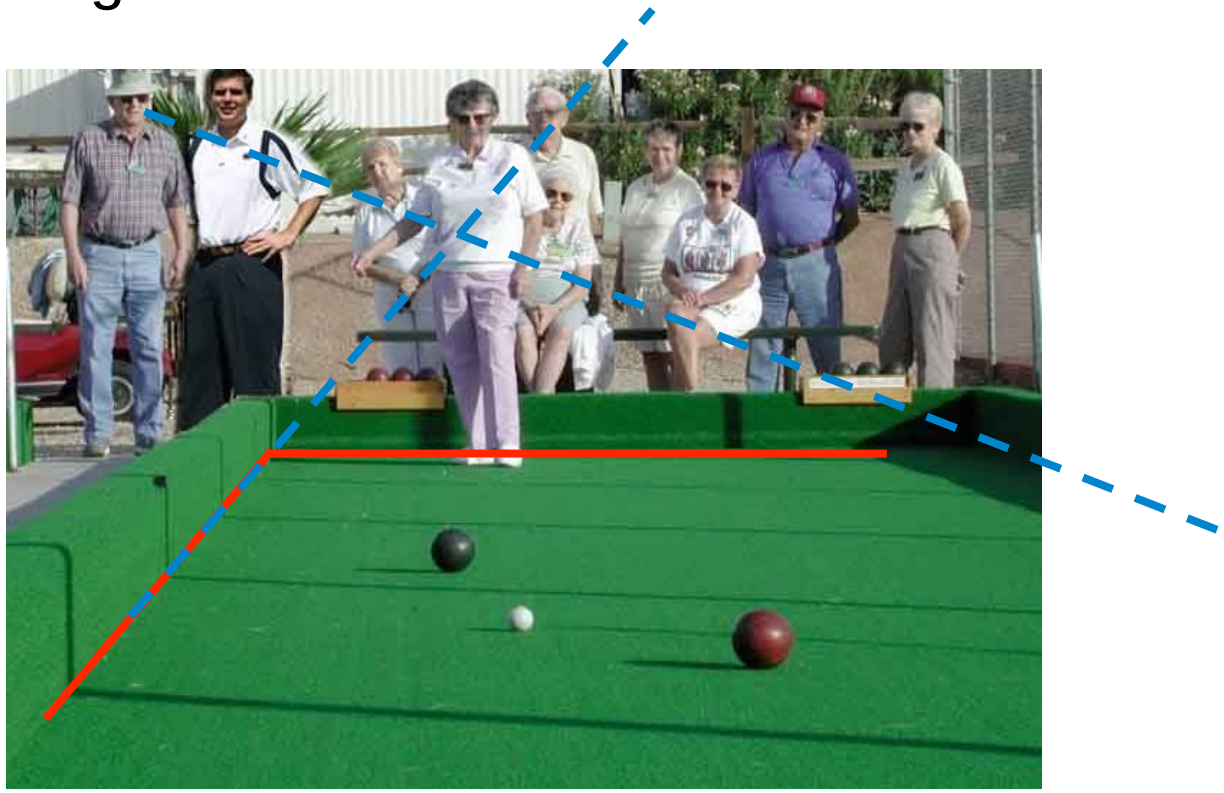
- Length
- Angles



Projective Geometry

What is preserved?

- Straight lines are still straight



Projection of lines

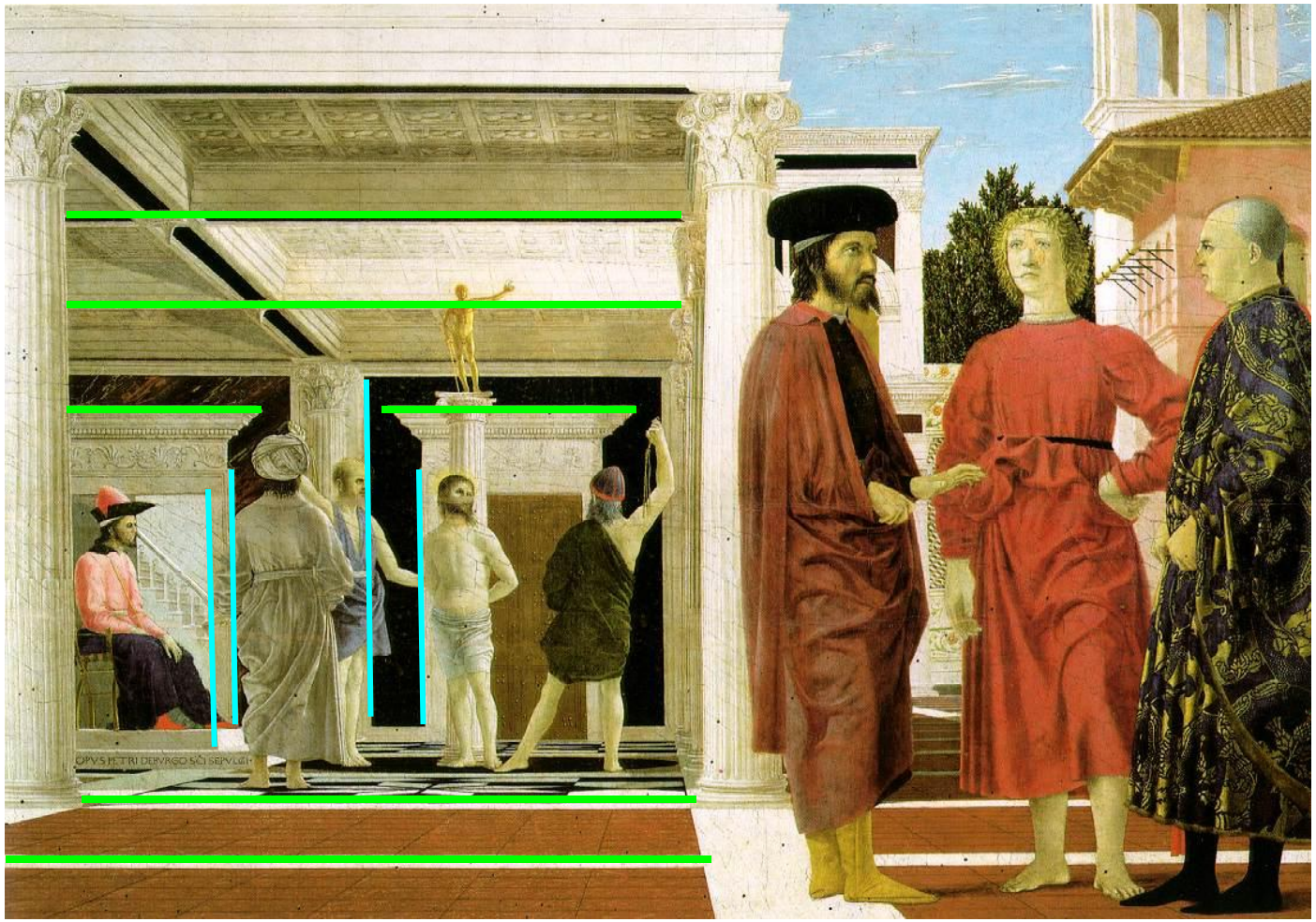
Q. When is parallelism preserved?

Piero della Francesca, *Flagellation of Christ*, 1455-1460

Projection of lines

Q. When is parallelism preserved?

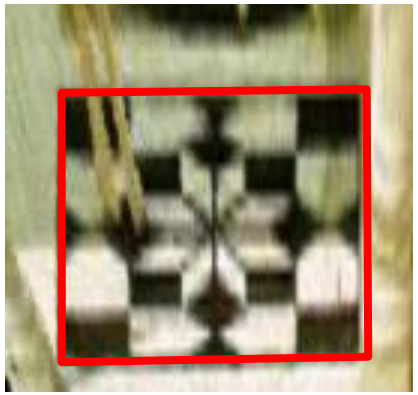
When the parallel lines are also parallel to the image plane

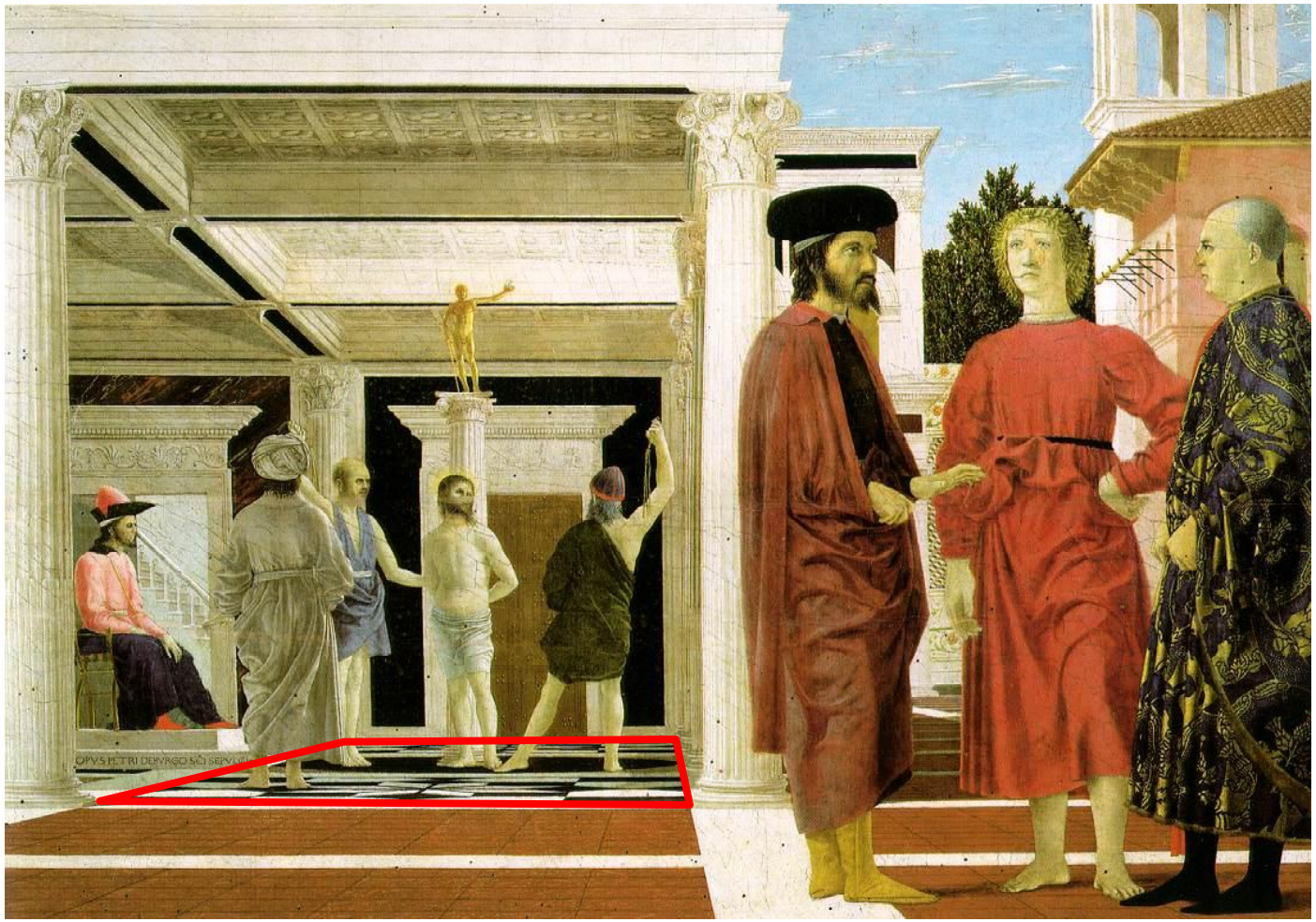


Piero della Francesca, *Flagellation of Christ*, 1455-1460

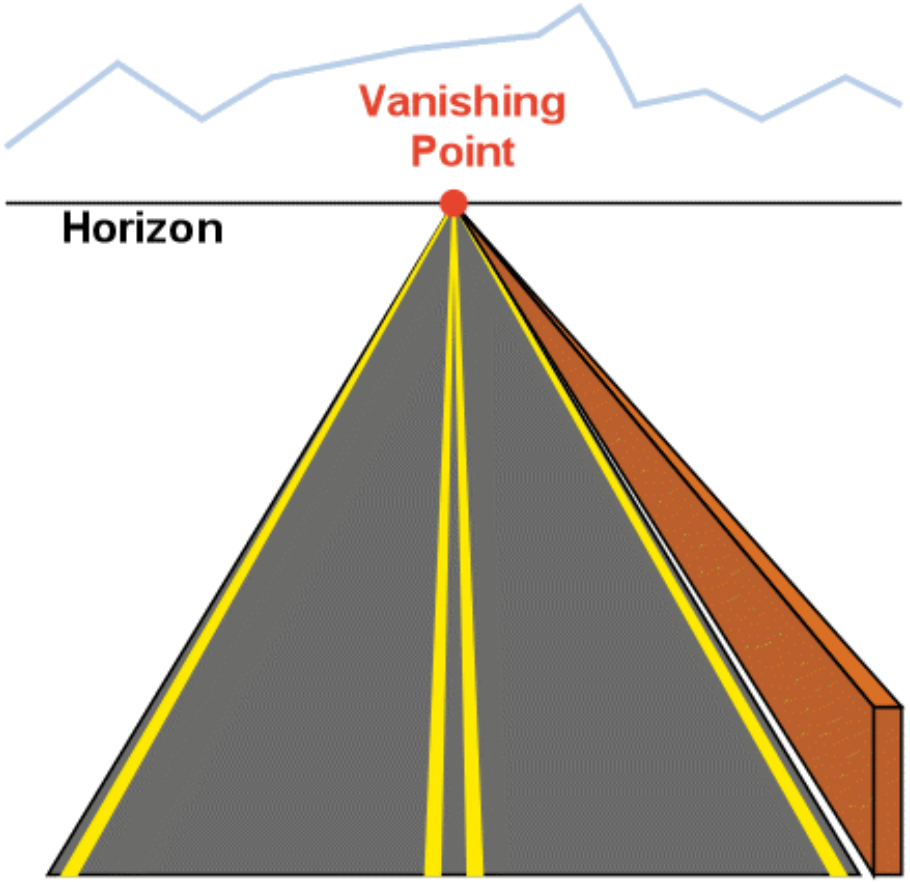
Projection of lines

Patterns on *non-fronto-parallel* planes are distorted by a *homography*

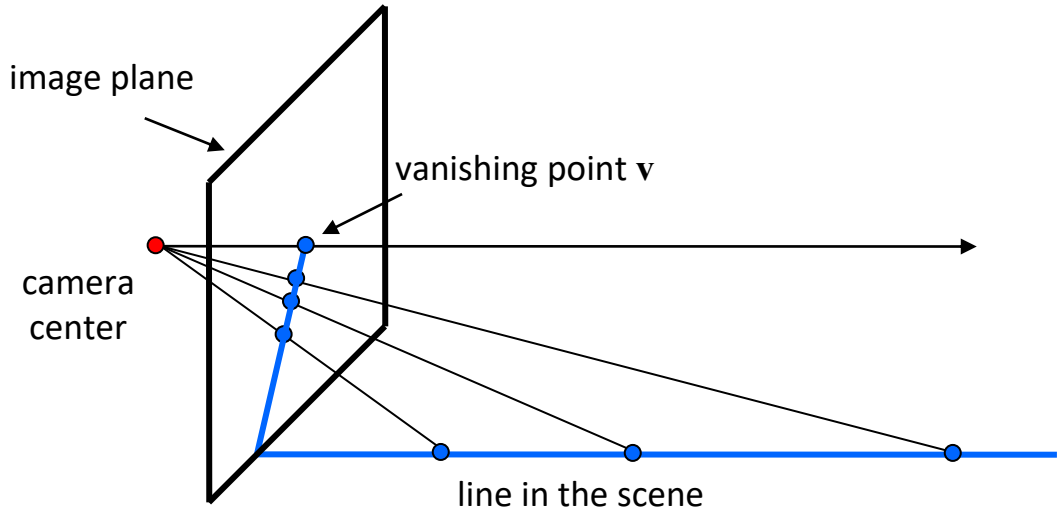




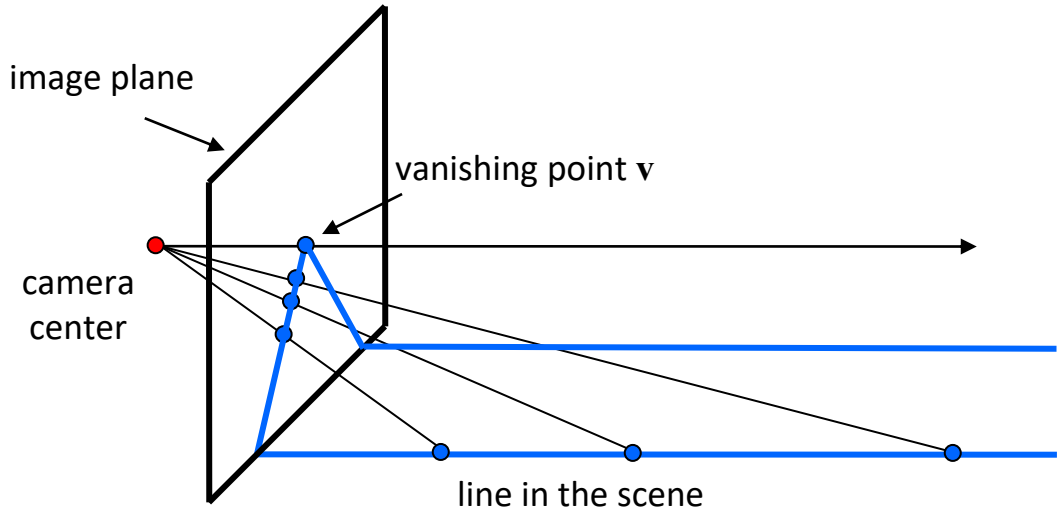
If parallel lines are no longer parallel, where do they intersect?



Vanishing Points & Lines



Vanishing Points & Lines

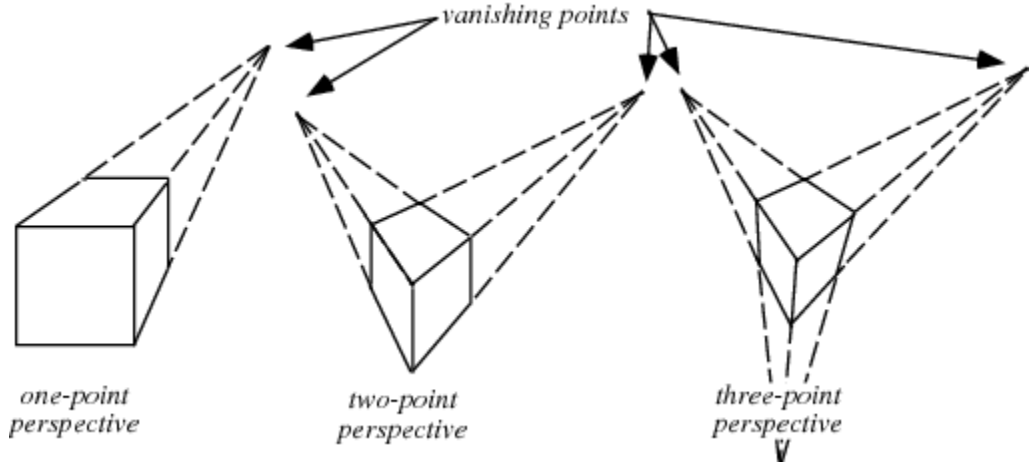


Vanishing Points & Lines

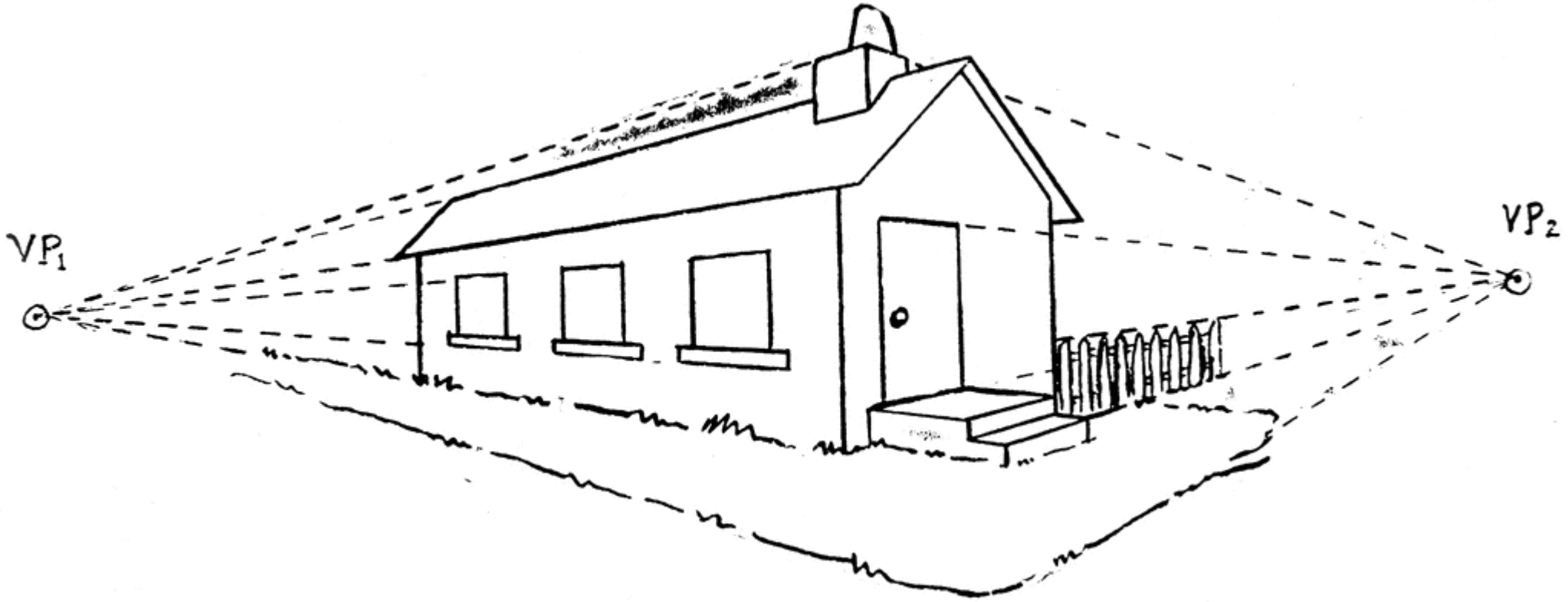
1-, 2-, 3-perspective art instruction

3D objects can have 1, 2, or 3 vanishing points depending on the camera location.

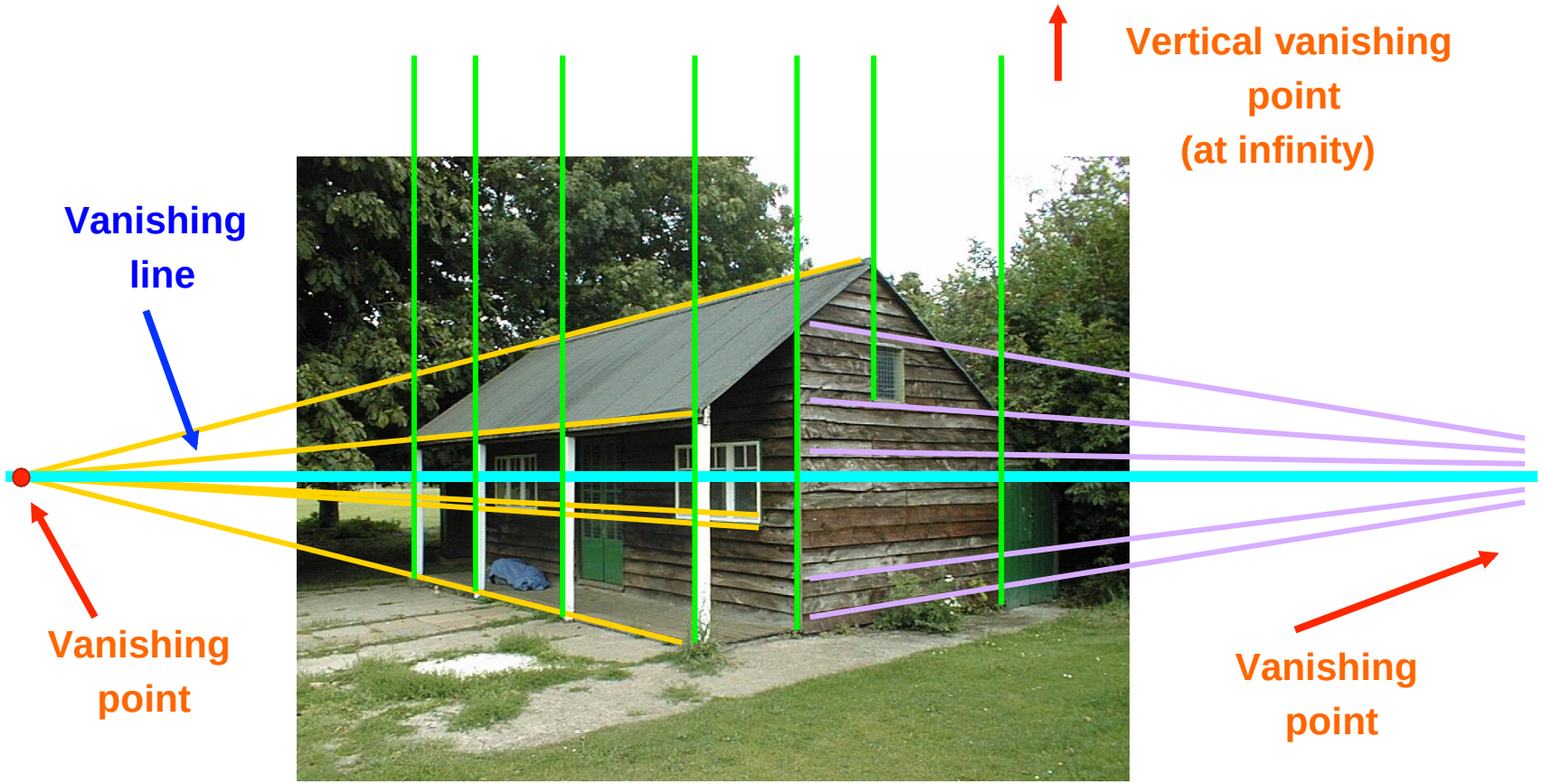
An image with multiple objects can have an arbitrary number of vanishing points.



Vanishing lines for a house

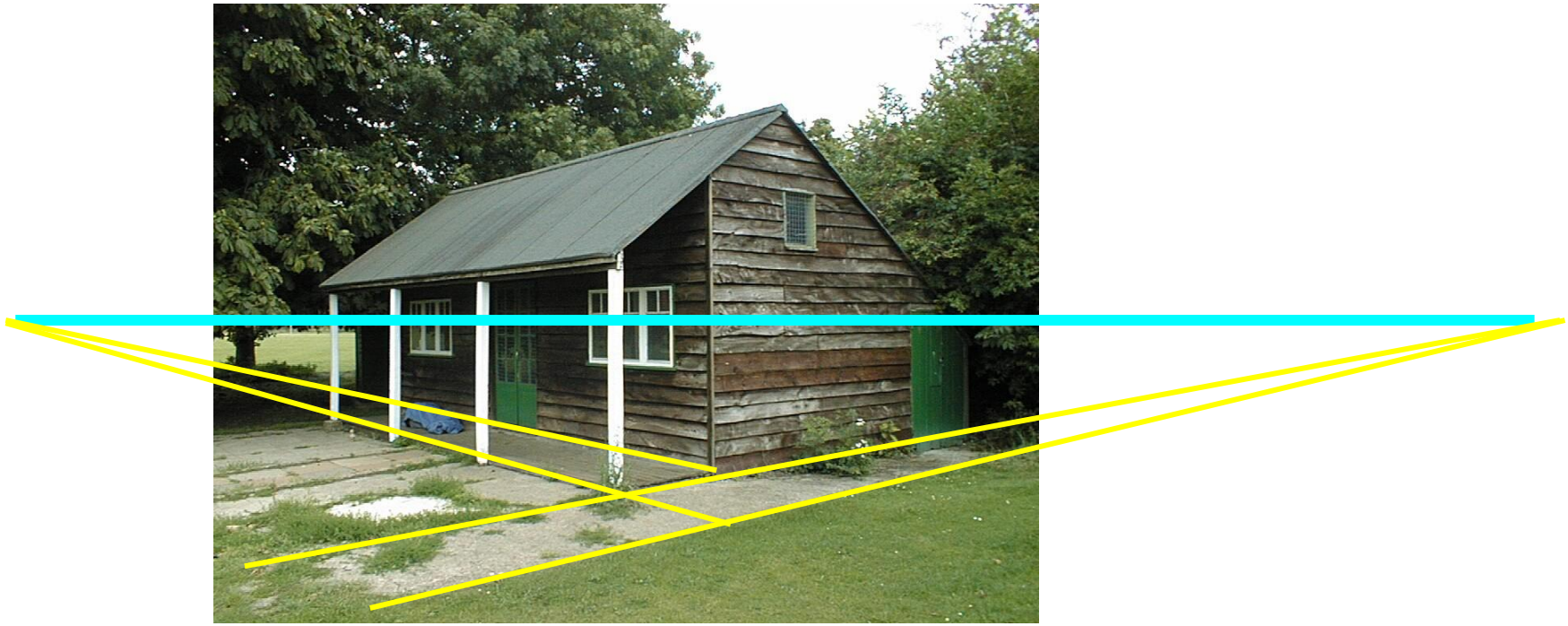


Parallel lines in the world intersect in the image at a vanishing point

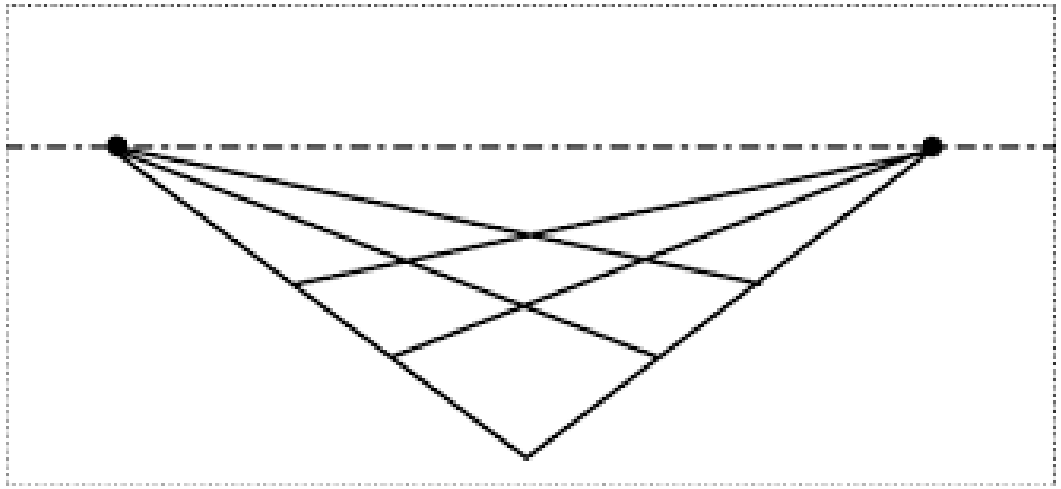


Not all lines that intersect in 2D are parallel in 3D!

Horizon: vanishing line of the ground plane (and planes parallel to it)



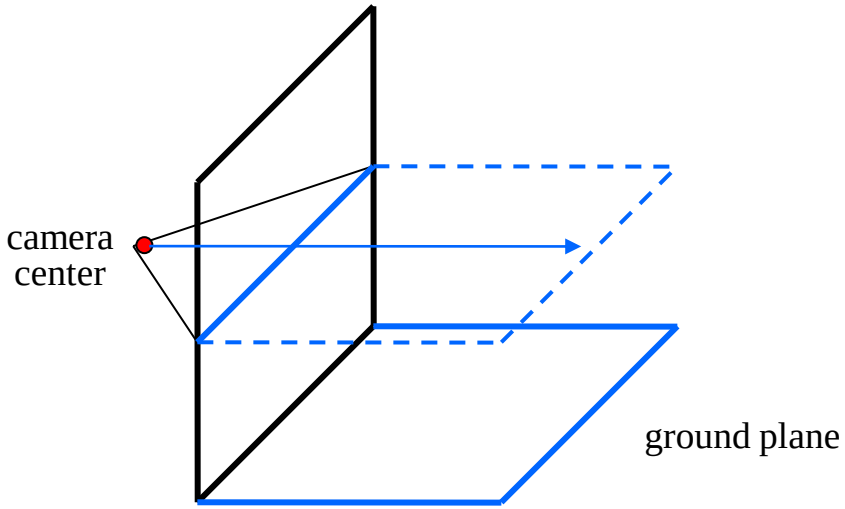
Parallel planes *in the world* intersect *in the image* at a **vanishing line**



Vanishing lines of planes

Horizon: vanishing line of the ground plane

Q. Is this parachutist **higher** or **lower** than the person taking this picture?

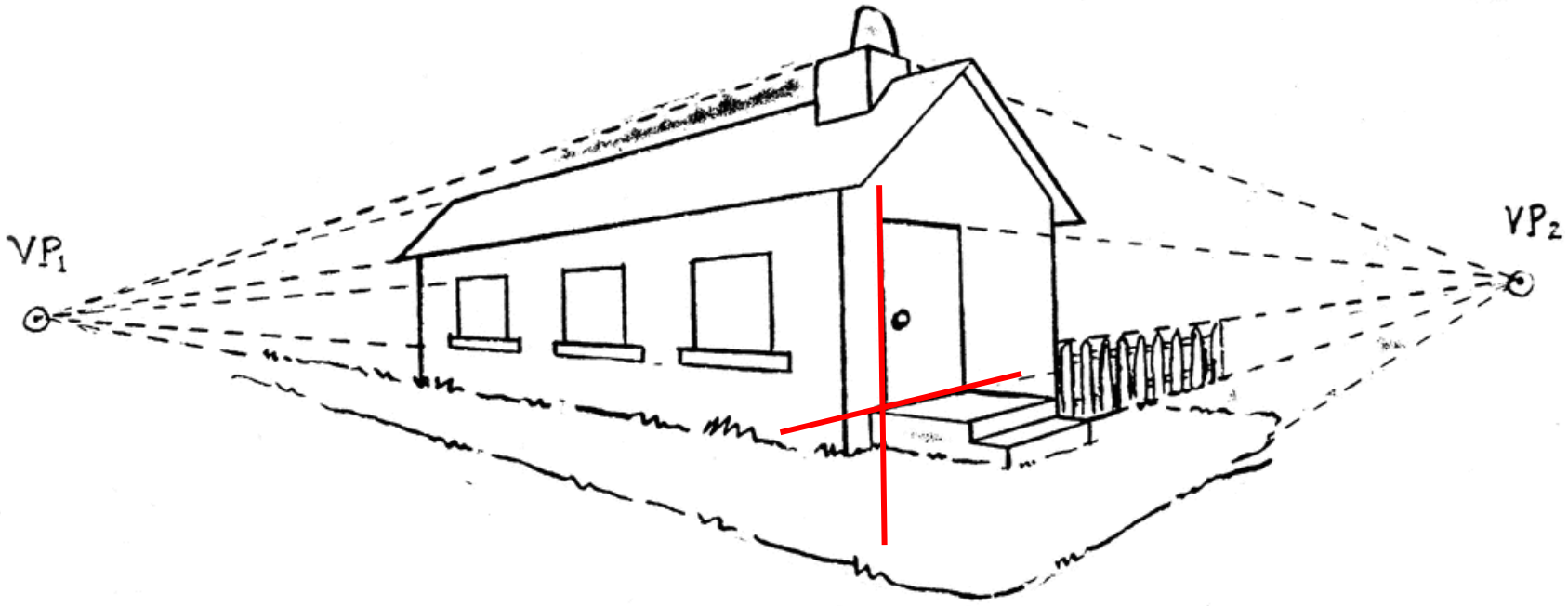


Q. Let’s try and answer these together with a neighbor

1. All lines that intersect in 2D are parallel in 3D
2. Lines in 3D always intersect each other in 2D at a single point
3. All non-parallel 3D planes in the real world intersect each other
4. 3D lines in the real world always intersect each other
5. All 3D lines intersect each other in the 2D image
6. All parallel lines in 3D meet at the same vanishing point
7. Non-intersecting lines in 3D meet at the same vanishing point
8. If a set of parallel 3D lines are also parallel to a particular plane, their vanishing point will lie on the vanishing line of the plane

1. All lines that intersect in 2D are parallel in 3D

Not true.



2. Lines in 3D always intersect each other in 2D at a single point

Lines in 3D are still lines in 2D.

Any two distinct lines meet in exactly one point (which may lie ‘at infinity’ if the lines are parallel in the Euclidean sense).

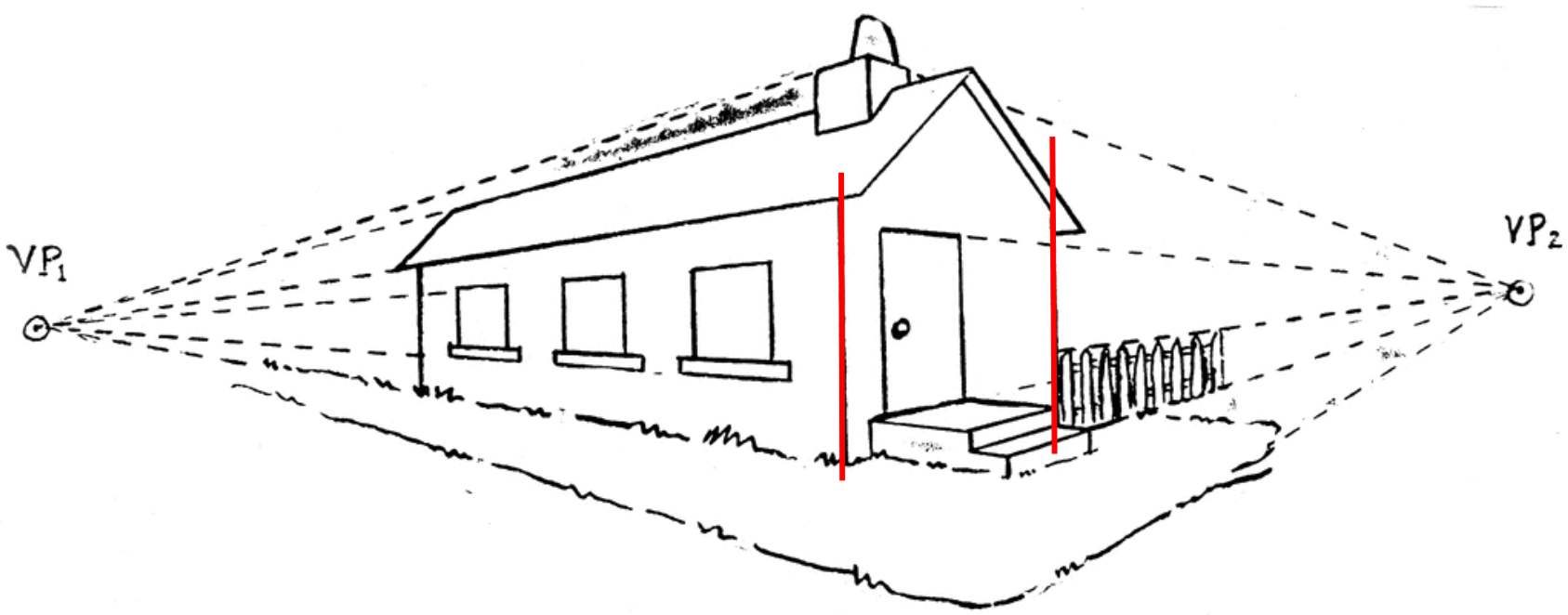
3. All non-parallel 3D planes in the real world intersect each other

True. They intersect at a line.

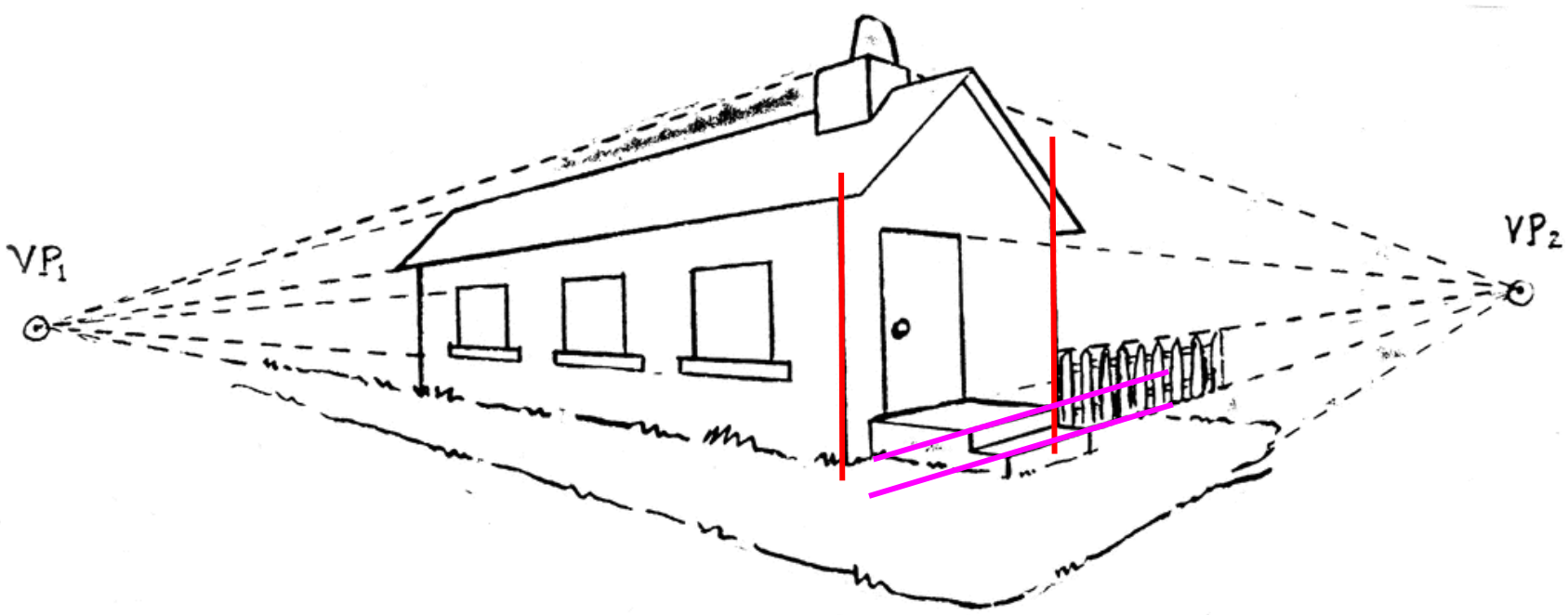
Parallel planes could potentially intersect at infinity (but don’t have to)

The horizon line is where the ground planes intersect

4. 3D lines in the real world always intersect each other



5. All 3D lines intersect each other in the 2D image



6. All parallel lines in 3D meet at the same vanishing point

Q. how would you go about proving this?

6. All parallel lines in 3D meet at the same vanishing point

consider a simple camera with its camera z-axis aligned with the world z-axis:

For line $\ell_i(t) = \big(p_{ix} + t\,d_x,\ p_{iy} + t\,d_y,\ p_{iz} + t\,d_z\big)$,
its projected image in the 2D plane is:

$$\big(x_i(t),\ y_i(t)\big) \;=\; \pi\big(\ell_i(t)\big) \;=\; \left(\frac{p_{ix} + t\,d_x}{p_{iz} + t\,d_z},\ \frac{p_{iy} + t\,d_y}{p_{iz} + t\,d_z}\right),$$

6. All parallel lines in 3D meet at the same vanishing point

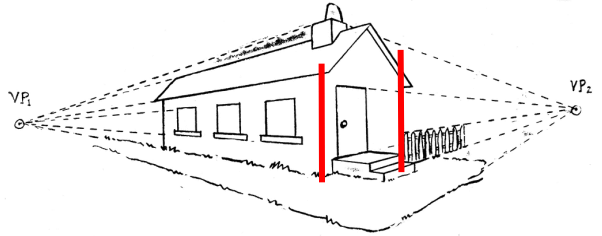
consider a simple camera with its camera z-axis aligned with the world z-axis:

For line $\ell_i(t) = \big(p_{ix} + t\,d_x,\ p_{iy} + t\,d_y,\ p_{iz} + t\,d_z\big)$,
its projected image in the 2D plane is:

$$\big(x_i(t),\ y_i(t)\big) \;=\; \pi\big(\ell_i(t)\big) \;=\; \left(\frac{p_{ix} + t\,d_x}{p_{iz} + t\,d_z},\ \frac{p_{iy} + t\,d_y}{p_{iz} + t\,d_z}\right),$$

as long as $p_{iz} + t\,d_z \neq 0$.

when dz = 0, the lines are parallel in both 3D and 2D and never intersect



6. All parallel lines in 3D meet at the same vanishing point

Since the lines are parallel in 3D, they intersection

As $t \to \infty$, the ratio

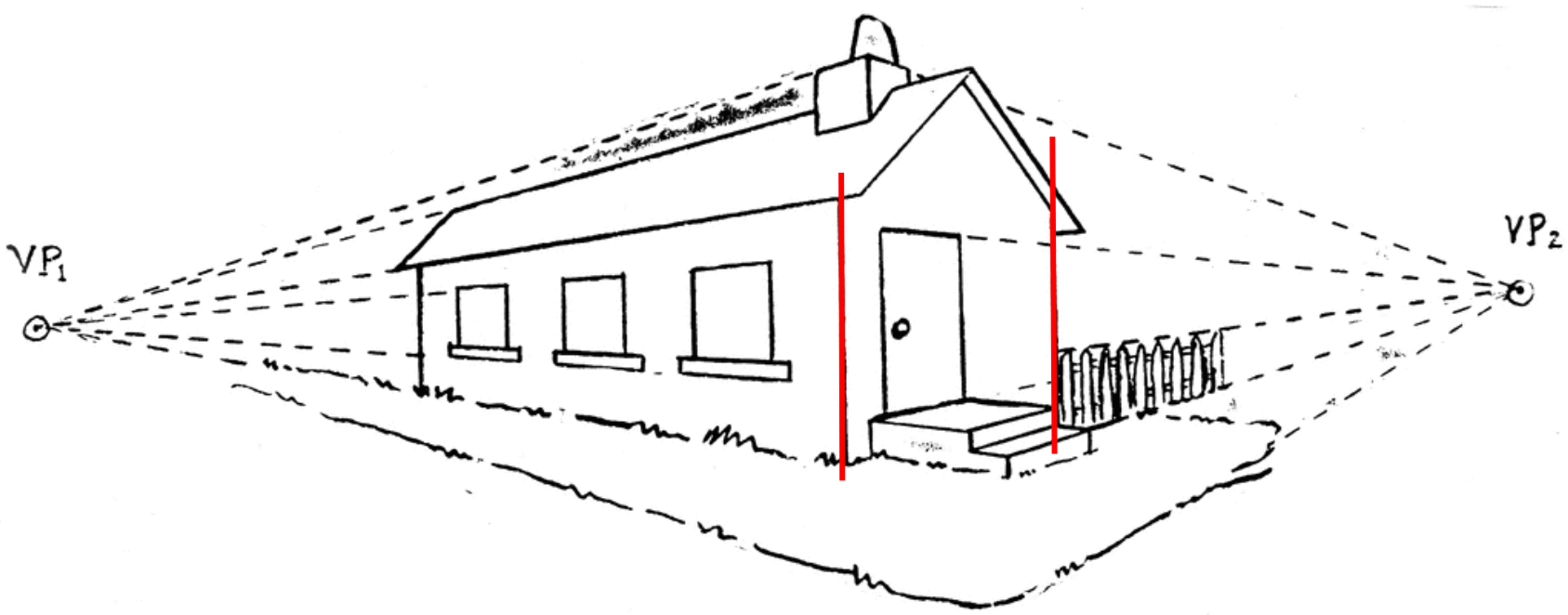
$$x_i(t) = \frac{p_{ix} + t\,d_x}{p_{iz} + t\,d_z} \quad \text{and} \quad y_i(t) = \frac{p_{iy} + t\,d_y}{p_{iz} + t\,d_z}$$

behaves like

$$x_i(t) \approx \frac{t\,d_x}{t\,d_z} = \frac{d_x}{d_z}, \quad y_i(t) \approx \frac{d_y}{d_z}.$$

In the limit $t \to \infty$, $\big(x_i(t),\, y_i(t)\big)$ approaches the **same** coordinate $\left(\frac{d_x}{d_z},\, \frac{d_y}{d_z}\right)$, regardless of $\mathbf{p_i}$.

7. Non-intersecting lines in 3D meet at the same vanishing point

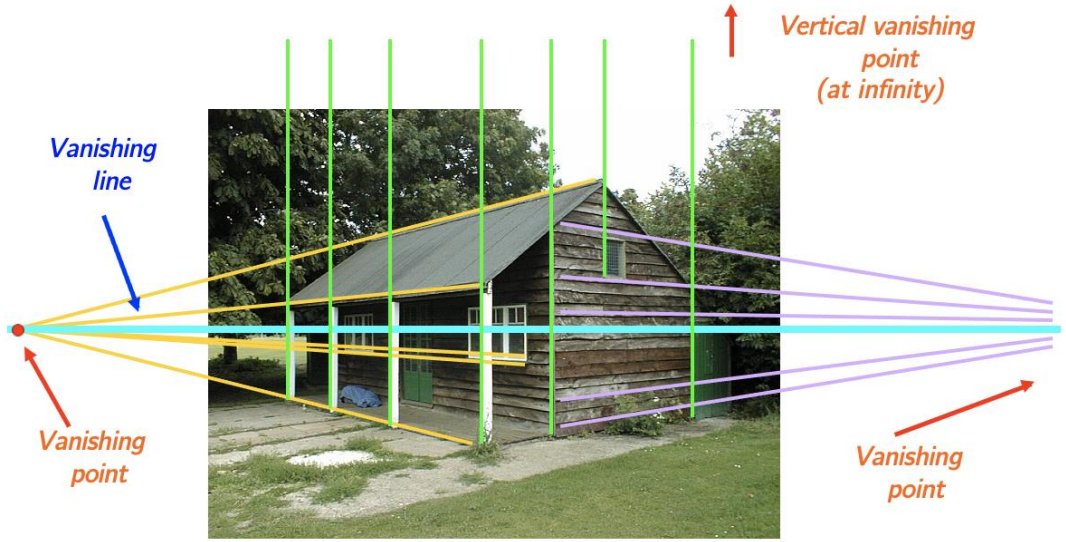


8. If a set of parallel 3D lines are also parallel to a particular plane, their vanishing point will lie on the vanishing line of the plane

Same proof as 6.

Properties of Vanishing Points & Lines

- The projections of parallel 3D lines intersect at a vanishing point
- The projection of parallel 3D planes intersect at a vanishing line
- Vanishing point <-> 3D direction of a line
- Vanishing line <-> 3D orientation of a surface



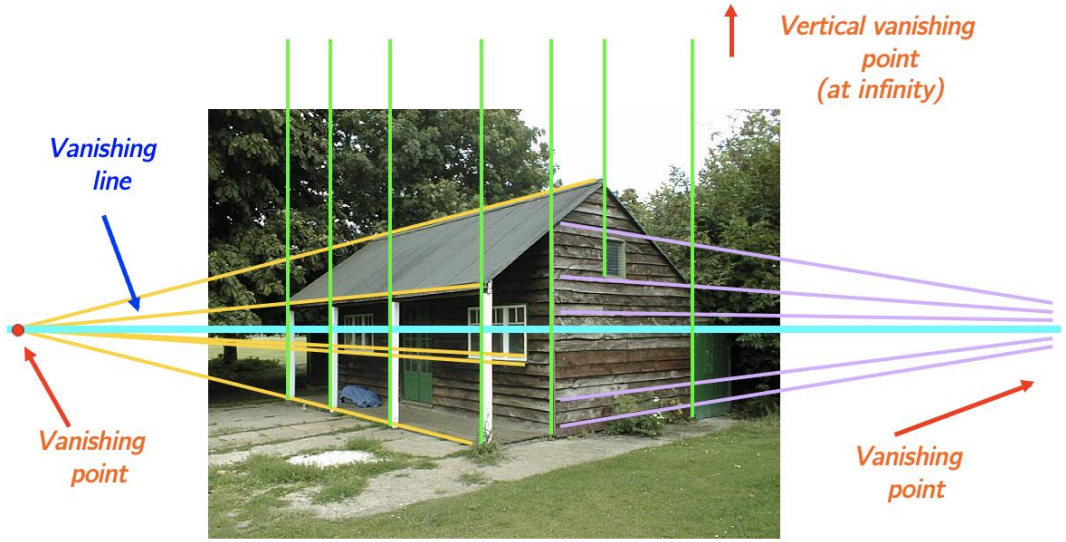
Vanishing Points are a key perspective tool:

from making realistic drawings, to measuring

3D from 2D, and even for camera calibration!

How do you calculate vanishing points?

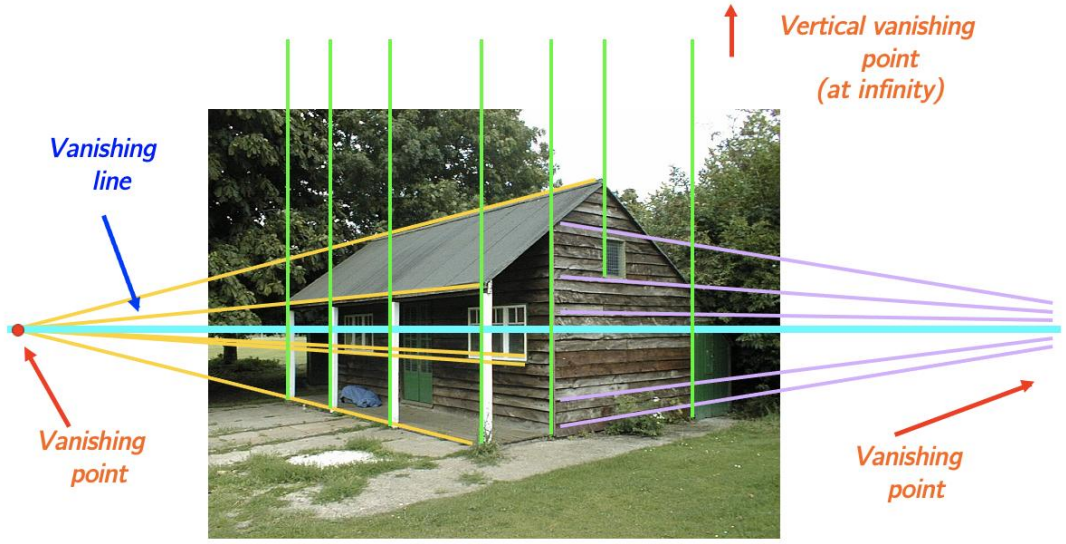
Q. What do you need first?



How do you calculate vanishing points?

Lines!

Q. How do you get the lines?



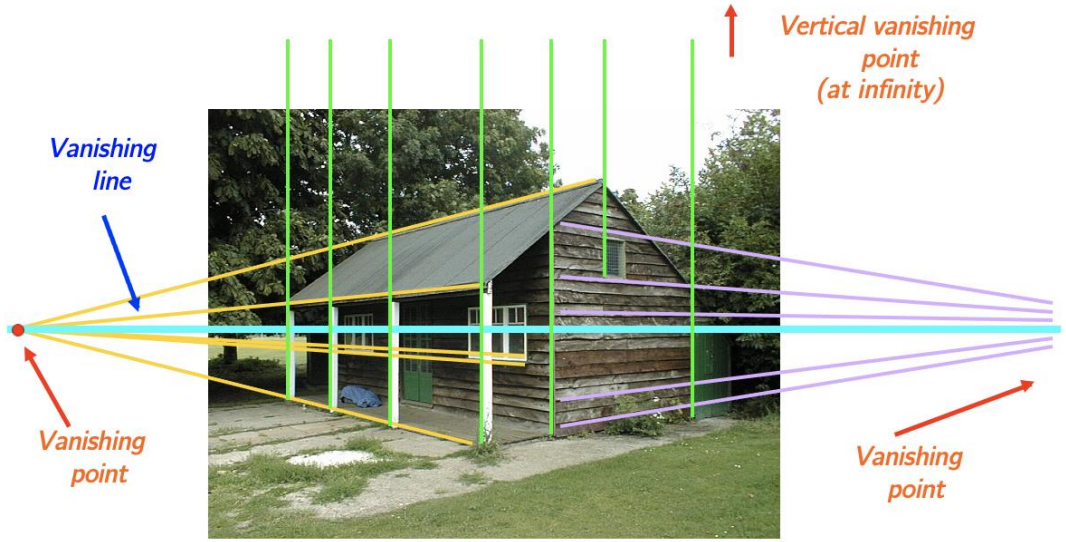
How do you calculate vanishing points?

Lines!

First calculate edges using Canny Edge Detector.

Then use RANSAC or Hough transforms to get the lines.

Q. Once you have the lines, how do you find the vanishing points?

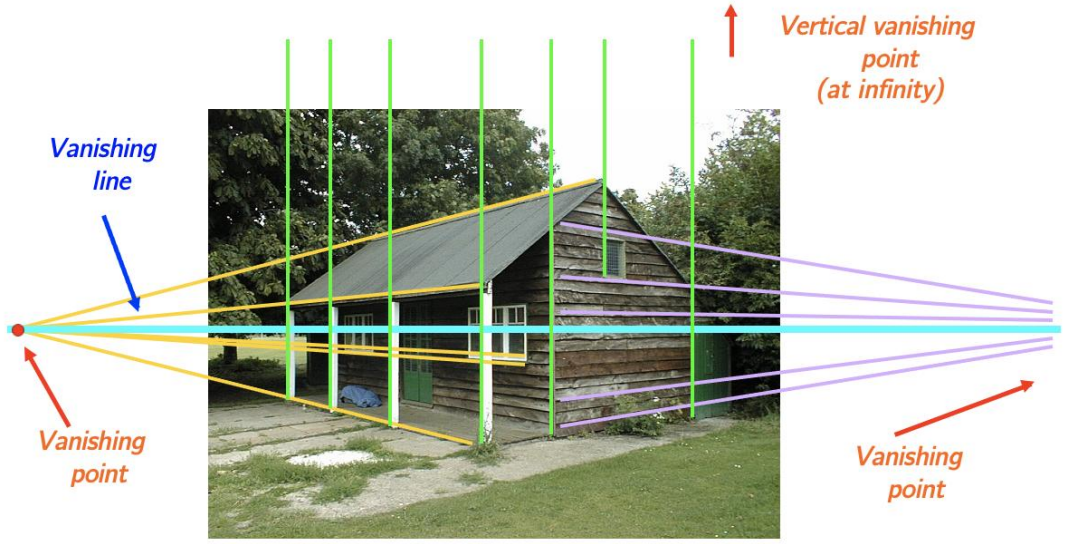


How do you calculate vanishing points?

Lines!

First calculate edges using Canny Edge Detector.

Then use RANSAC or Hough transforms to get the lines.



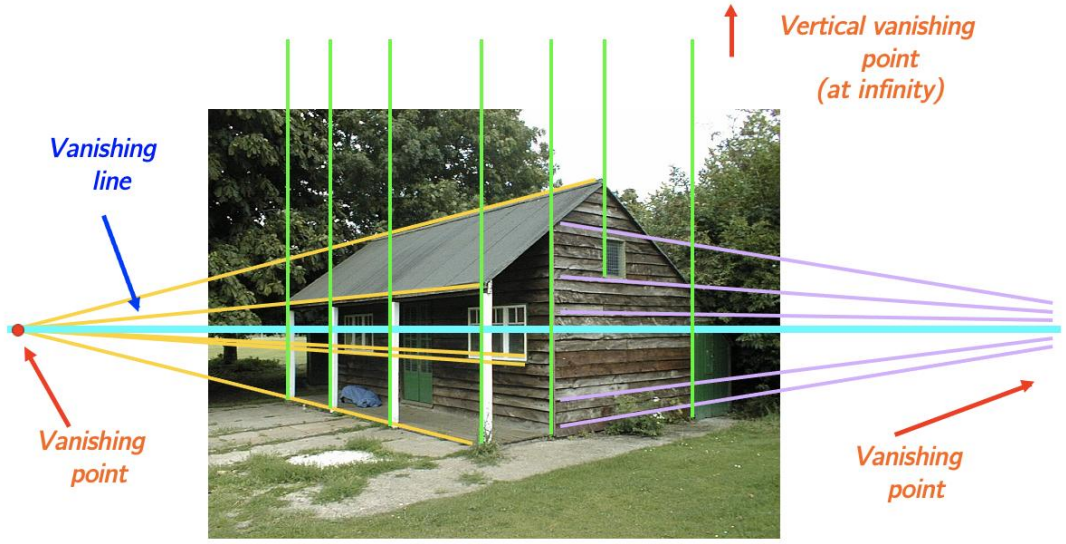
Q. Once you have the lines, how do you find the vanishing points?

RANSAC again, but in pixel space to count the intersections!

What can you do with the vanishing points?

You can calculate camera intrinsics.

Let v_1, v_2 be the *homogeneous* coordinates of two distinct vanishing points.



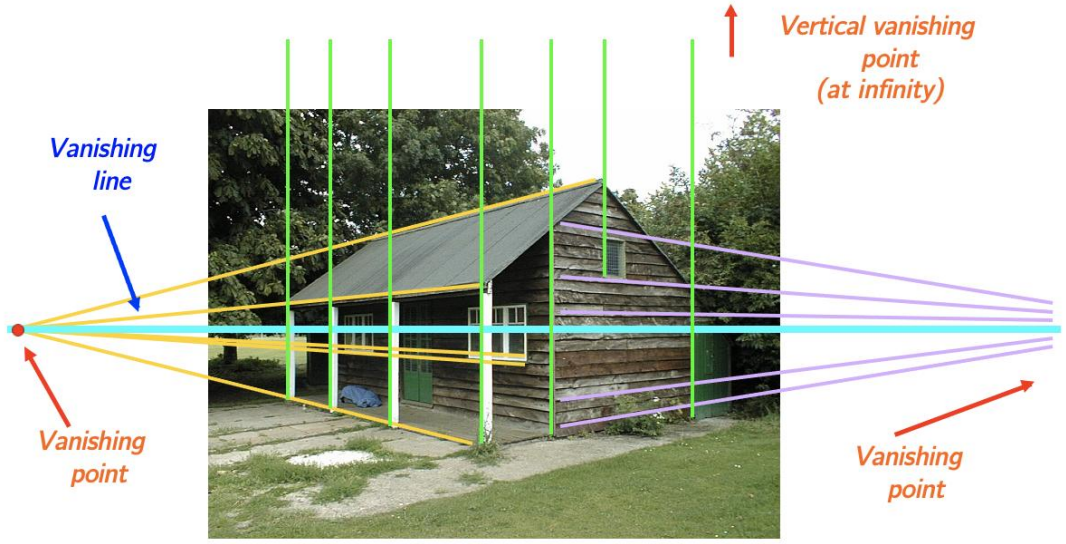
Calculating the lines from the vanishing points *(assuming we already know intrinsics K)*

You can calculate camera intrinsics.

Let v_1,v_2 be the *homogeneous* coordinates two distinct vanishing points.

Then the 3D directions of these lines—after undoing the intrinsics—are:

$$\mathbf{d}_1 \propto K^{-1}\mathbf{v}_1, \quad \mathbf{d}_2 \propto K^{-1}\mathbf{v}_2$$



Calculating the lines from the vanishing points *(assuming we already know intrinsics K)*

You can calculate camera intrinsics.

Let v_1,v_2 be the *homogeneous* coordinates two distinct vanishing points.

Then the 3D directions of these lines—after undoing the intrinsics—are:

$$\mathbf{d}_1 \propto K^{-1}\mathbf{v}_1, \quad \mathbf{d}_2 \propto K^{-1}\mathbf{v}_2$$

Because these directions are orthogonal in 3D, we have: $\mathbf{d}_1^\top \mathbf{d}_2 = 0$.

Calculating the lines from the vanishing points *(assuming we already know intrinsics K)*

You can calculate camera intrinsics.

Let v_1,v_2 be the *homogeneous* coordinates two distinct vanishing points.

Then the 3D directions of these lines—after undoing the intrinsics—are:

$$\mathbf{d}_1 \propto K^{-1}\mathbf{v}_1, \quad \mathbf{d}_2 \propto K^{-1}\mathbf{v}_2$$

Because these directions are orthogonal in 3D, we have: $\mathbf{d}_1^\top \mathbf{d}_2 = 0$.

Substituting $d_i=K^{-1}v_i$, we get:

$$(K^{-1}\mathbf{v}_1)^\top (K^{-1}\mathbf{v}_2) = \mathbf{v}_1^\top (K^{-1})^\top (K^{-1})\,\mathbf{v}_2 = 0$$

Calculating the lines from the vanishing points *(assuming we already know intrinsics K)*

You can calculate camera intrinsics.

Let v_1,v_2 be the *homogeneous* coordinates two distinct vanishing points.

Then the 3D directions of these lines—after undoing the intrinsics—are:

$$\mathbf{d}_1 \propto K^{-1}\mathbf{v}_1, \quad \mathbf{d}_2 \propto K^{-1}\mathbf{v}_2$$

Because these directions are orthogonal in 3D, we have:

$$\mathbf{d}_1^\top \mathbf{d}_2 = 0$$

Substituting $d_i=K^{-1}v_i$, we get:

$$(K^{-1}\mathbf{v}_1)^\top (K^{-1}\mathbf{v}_2) = \mathbf{v}_1^\top (K^{-1})^\top (K^{-1})\,\mathbf{v}_2 = 0$$

K has only 3 values:

$$\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Each pair of vanishing points gives us 1 equation. We need at least 3 pairs

Calculating the lines from the vanishing points *(assuming we already know intrinsics K)*

You can calculate camera intrinsics.

after undoing the intrinsics—are:

$$\mathbf{d}_1 \propto K^{-1}\mathbf{v}_1, \quad \mathbf{d}_2 \propto K^{-1}\mathbf{v}_2$$

Because these directions are orthogonal in 3D, we have:

$$\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Each pair of vanishing points gives us 1 equation. We need at least 3 pairs

We will use this method in your assignments. We will explore another way in lecture to calculate camera intrinsics

Today’s agenda

- Properties of Perspective transformations
- Introduction to Camera Calibration
- Linear camera calibration method
- Calculating intrinsics and extrinsics
- Depth estimation

An improved method from Roger Tsai used in industry

- Notation
  - P = [x, y, z] is a point in the 3D world coordinate system
  - p = [u, v] is a point in the real image plane (u axis is horizontal and v axis is vertical)
  - a = [r, c] is a point in the image array

- Camera calibration will recover both intrinsic and extrinsic parameters.

Intrinsic Camera Parameters

The *intrinsic* parameters are truly camera parameters, as they depend on the particular device being used. They include the following parameters:

- principal point $[u_0, v_0]$: the intersection of the optical axis with the image plane.
- scale factors $\{d_x, d_y\}$ for the x and y pixel dimensions.
- aspect distortion factor τ_1: a scale factor used to model distortion in the aspect ratio of the camera.
- focal length f: the distance from the optical center to the image plane.
- lens distortion factor (κ_1): a scale factor used to model radial lens distortion.

External Camera Parameters

The *extrinsic* parameters describe the position and orientation (pose) of the camera system in the 3D world. They include:

- translation:

$$\mathbf{t} = [t_x \quad t_y \quad t_z]^T \tag{13.44}$$

- rotation:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.45}$$

There are only 3 independent rotation parameters, not nine.

Calibration Object

i	x_i	y_i	z_i	u_i	v_i
1	0.00	5.00	0.00	-0.58	0.00
2	10.00	7.50	0.00	1.73	1.00
3	10.00	5.00	0.00	1.73	0.00
4	5.00	10.00	0.00	0.00	1.00
5	5.00	0.00	0.00	0.00	-1.00

Algorithm

Given $\{(\ [x_i, y_i, z_i],\ [u_i, v_i]\ ) \mid i = 1, \ldots, n\}$.

The real-valued image coordinates $[u, v]$ are computed from their pixel position $[r, c]$ by the formulas

$$u = \tau_1 d_x (c - u_0) \quad (13.46)$$

$$v = -d_y (r - v_0) \quad (13.47)$$

where d_x and d_y are the center-to-center distances between pixels in the horizontal and vertical directions, and τ_1 is the scale factor for distortions in the aspect ratio of the camera.

Algorithm (2)

Instead of directly solving for all unknowns, first solve for a set of computed parameters from which the extrinsic parameters can be derived. Define matrix A with rows:

$$a_i = [v_i x_i,\ v_i y_i,\ -u_i x_i,\ -u_i y_i,\ v_i]. \tag{13.48}$$

Let $\mu = [\mu_1, \mu_2, \mu_3, \mu_4, \mu_5]$ be a vector of unknown (computed) parameters defined with respect to the rotation parameters r_{11}, r_{12}, r_{21}, and r_{22} and the translation parameters t_x and t_y as:

$$\mu_1 = \frac{r_{11}}{t_y} \tag{13.49}$$

$$\mu_2 = \frac{r_{12}}{t_y} \tag{13.50}$$

$$\mu_3 = \frac{r_{21}}{t_y} \tag{13.51}$$

$$\mu_4 = \frac{r_{22}}{t_y} \tag{13.52}$$

$$\mu_5 = \frac{t_x}{t_y} \tag{13.53}$$

Algorithm (3)

Let the vector $\mathbf{b} = [u_1, u_2, \ldots, u_n]$ contain the u_i image coordinates of the n correspondences. Since $\mathbf{A}$ and $\mathbf{b}$ are known, the system of linear equations

$$\mathbf{A}\mu = \mathbf{b} \tag{13.54}$$

can be solved for the unknown parameter vector μ. (See *Introduction to Linear Algebra, 2nd edition* by Johnson, Riess, and Arnold for techniques on solving linear systems of equations.) Now μ can be used to compute the rotation and translation parameters as follows.

Algorithm (4)

1. Let $U = \mu_1^2 + \mu_2^2 + \mu_3^2 + \mu_4^2$. Calculate the square of the y component of translation t_y as:

$$t_y^2 = \begin{cases} \frac{U - [U^2 - 4(\mu_1\mu_4 - \mu_2\mu_3)^2]^{1/2}}{2(\mu_1\mu_4 - \mu_2\mu_3)^2} & \text{if } (\mu_1\mu_4 - \mu_2\mu_3) \neq 0 \\ \frac{1}{\mu_1^2 + \mu_2^2} & \text{if } (\mu_1^2 + \mu_2^2) \neq 0 \\ \frac{1}{\mu_3^2 + \mu_4^2} & \text{if } (\mu_3^2 + \mu_4^2) \neq 0 \end{cases} \quad (13.55)$$

Algorithm (5)

2. Let $t_y = (t_y^2)^{1/2}$ (the positive square root) and compute four of the rotation parameters and the x-translation t_x from the known computed value of μ:

$$r_{11} = \mu_1 t_y \quad (13.56)$$

$$r_{12} = \mu_2 t_y \quad (13.57)$$

$$r_{21} = \mu_3 t_y \quad (13.58)$$

$$r_{22} = \mu_4 t_y \quad (13.59)$$

$$t_x = \mu_5 t_y \quad (13.60)$$

Algorithm (6)

3. To determine the true sign of t_y, select an object point $\mathbf{P}$ whose image coordinates $[u, v]$ are far from the image center (to avoid numerical problems). Let $\mathbf{P} = [x, y, z]$, and compute

$$\xi_x = r_{11}x + r_{12}y + t_x \quad (13.61)$$

$$\xi_y = r_{21}x + r_{22}y + t_y \quad (13.62)$$

This is like applying the computed rotation parameters to the x and y coordinates of $\mathbf{P}$. If ξ_x has the same sign as u and ξ_y has the same sign as v, then t_y already has the correct sign, else negate it.

Algorithm (7)

4. Now the remaining rotation parameters can be computed as follows:

$$r_{13} = (1 - r_{11}^2 - r_{12}^2)^{1/2} \quad (13.63)$$

$$r_{23} = (1 - r_{21}^2 - r_{22}^2)^{1/2} \quad (13.64)$$

$$r_{31} = \frac{1 - r_{11}^2 - r_{12}r_{21}}{r_{13}} \quad (13.65)$$

$$r_{32} = \frac{1 - r_{21}r_{12} - r_{22}^2}{r_{23}} \quad (13.66)$$

$$r_{33} = (1 - r_{31}r_{13} - r_{32}r_{23})^{1/2} \quad (13.67)$$

The orthonormality constraints of the rotation matrix $\mathbf{R}$ have been used in deriving these equations. The signs of r_{23}, r_{31}, and r_{32} may not be correct due to the ambiguity of the square root operation. At this step, r_{23} should be negated if the sign of

$$r_{11}r_{21} + r_{12}r_{22}$$

is positive, so that the orthogonality of the rotation matrix is preserved. The other two may need to be adjusted after computing the focal length.

Algorithm (8)

5. The focal length f and the z component of translation t_z are now computed from a second system of linear equations. First a matrix $\mathbf{A}'$ is formed whose rows are given by

$$a'_i = (r_{21}x_i + r_{22}y_i + t_y,\ v_i) \tag{13.68}$$

where d_y is the center-to-center distance between pixels in the y direction.

Next a vector $\mathbf{b}'$ is constructed with rows defined by

$$b'_i = (r_{31}x_i + r_{32}y_i)vi. \tag{13.69}$$

We solve the linear system

$$\mathbf{A}'\mathbf{v} = \mathbf{b}' \tag{13.70}$$

for $\mathbf{v} = (f, t_z)^T$. We obtain only estimates for f and t_z at this point.

Algorithm (9)

6. If $f < 0$ then change the signs of r_{13}, r_{23}, r_{31}, r_{32}, f, and t_z. This is to force the use of a right-handed coordinate system.

7. The estimates for f and t_z can be used to compute the lens distortion factor κ_1 and to derive improved values for f and t_z. The simple distortion model used here is that the true image coordinates $[\hat{u}, \hat{v}]$ are obtained from the measured ones by the following equations:

$$\hat{u} = u(1 + \kappa_1 r^2) \quad (13.71)$$

$$\hat{v} = v(1 + \kappa_1 r^2) \quad (13.72)$$

Algorithm (10)

where the radius of distortion r is given by

$$r = (u^2 + v^2))^{1/2} \tag{13.73}$$

Using the perspective projection equations, modified to include the distortion, we derive a set of nonlinear equations of the form

$$\left\{ v_i(1 + \kappa_1 r^2) = f \frac{r_{21}x_i + r_{22}y_i + r_{23}z_i + t_y}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_z} \right\} \; i = 1, \ldots, n \tag{13.74}$$

Solving this system by nonlinear regression gives the values for f, t_z, and κ_1.

Example you can look at

A fully worked out numeric example from my book is given at
https://courses.cs.washington.edu/courses/cse576/book/ch13.pdf

Today’s agenda

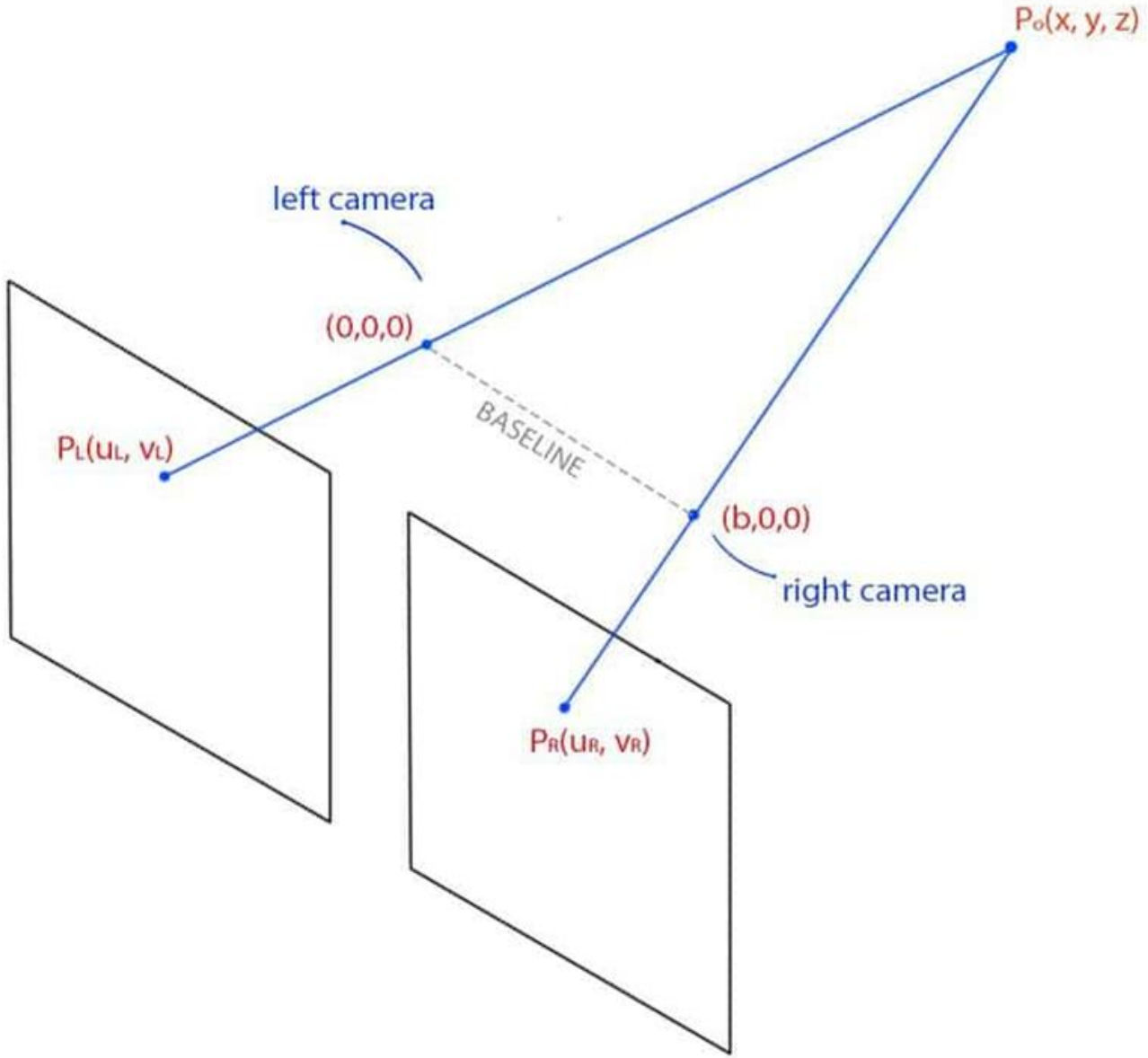
- Properties of Perspective transformations
- Introduction to Camera Calibration
- Linear camera calibration method
- Calculating intrinsics and extrinsics
- Depth estimation

Estimating depth

b is the translation from camera at location L and camera at R

We want to estimate z

We know (u_L, v_L) and (u_R, v_R)



Estimating depth

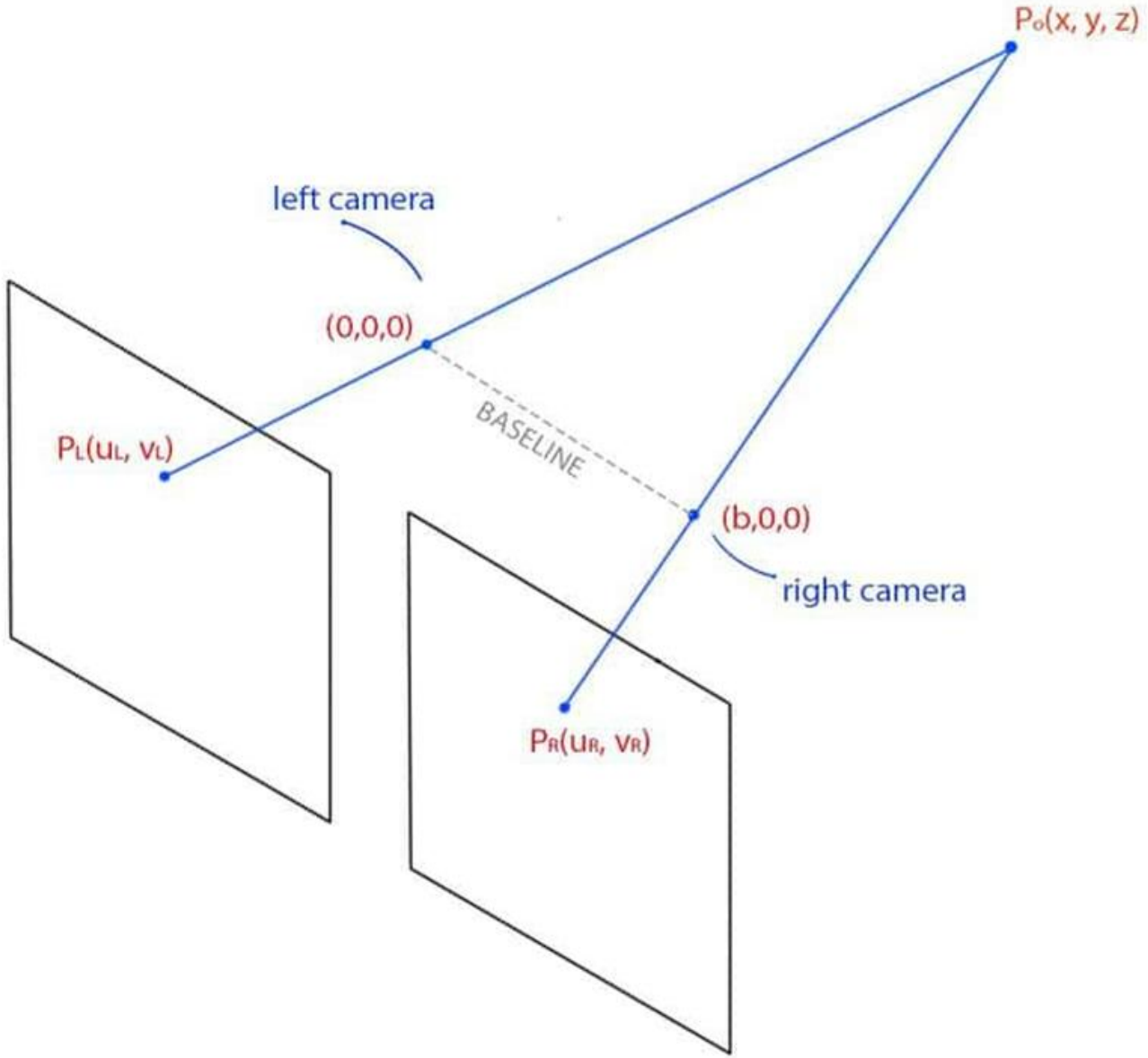
We can estimate the intrinsics of the camera from the previous sections

So, we know:

focal length: f_x, f_y

translation: p_x, p_y

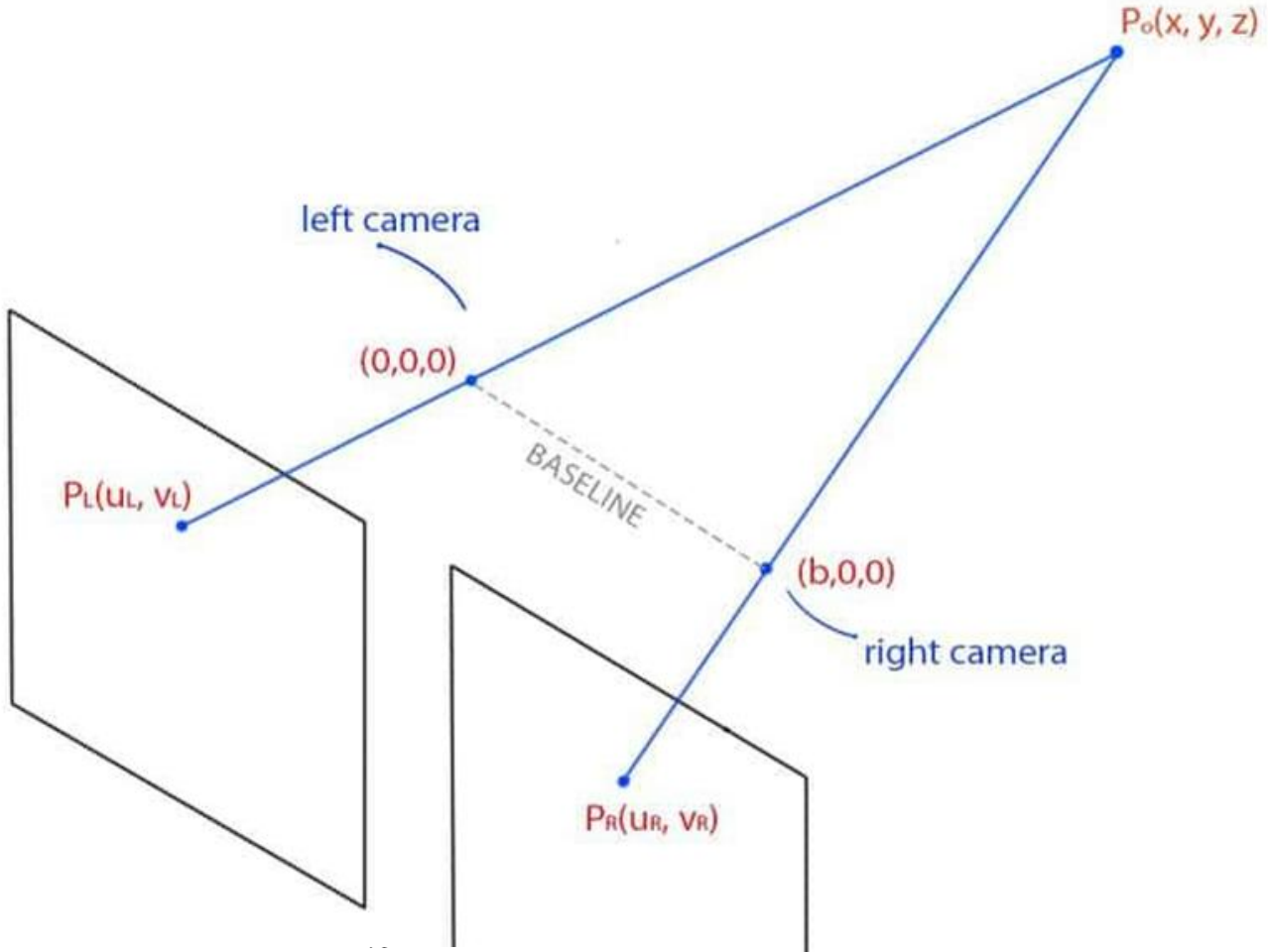
$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$



Estimating depth

$$u_L = f\frac{x}{z} + p_x$$

$$v_L = f\frac{y}{z} + p_y$$



$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

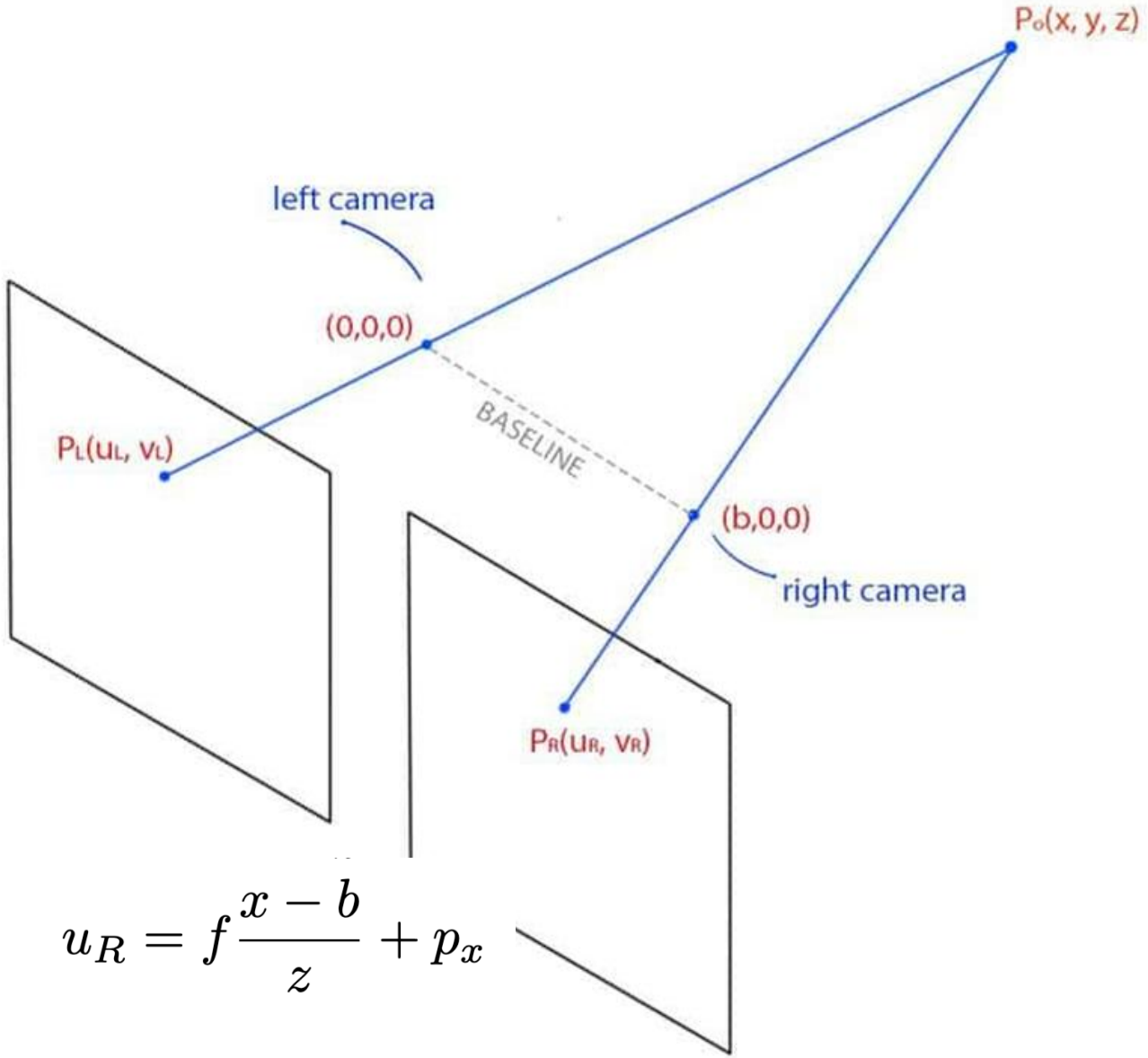
$$u_R = f\frac{x-b}{z} + p_x$$

$$v_R = f\frac{y}{z} + p_y$$

Estimating depth

$$u_L = f\frac{x}{z} + p_x$$

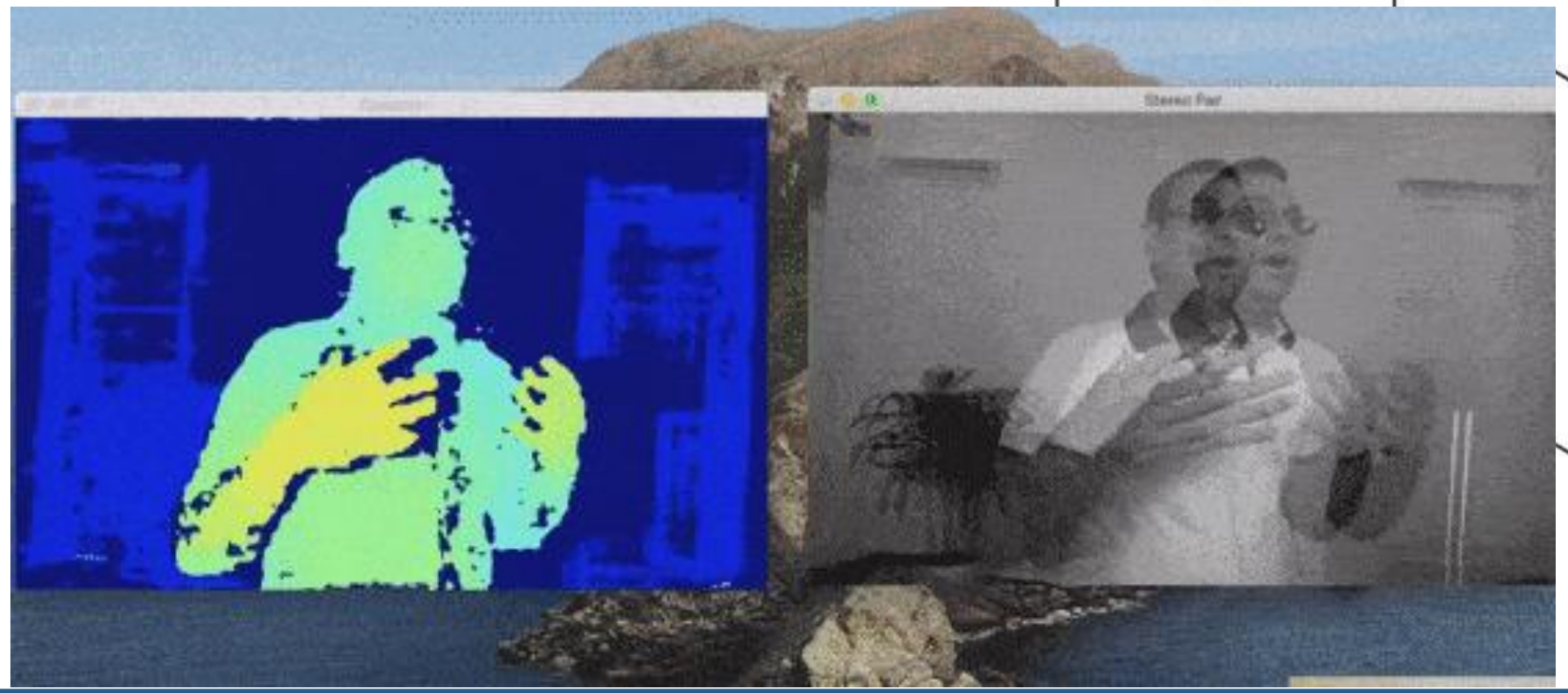
$$z = \frac{fb}{u_L - u_R}$$

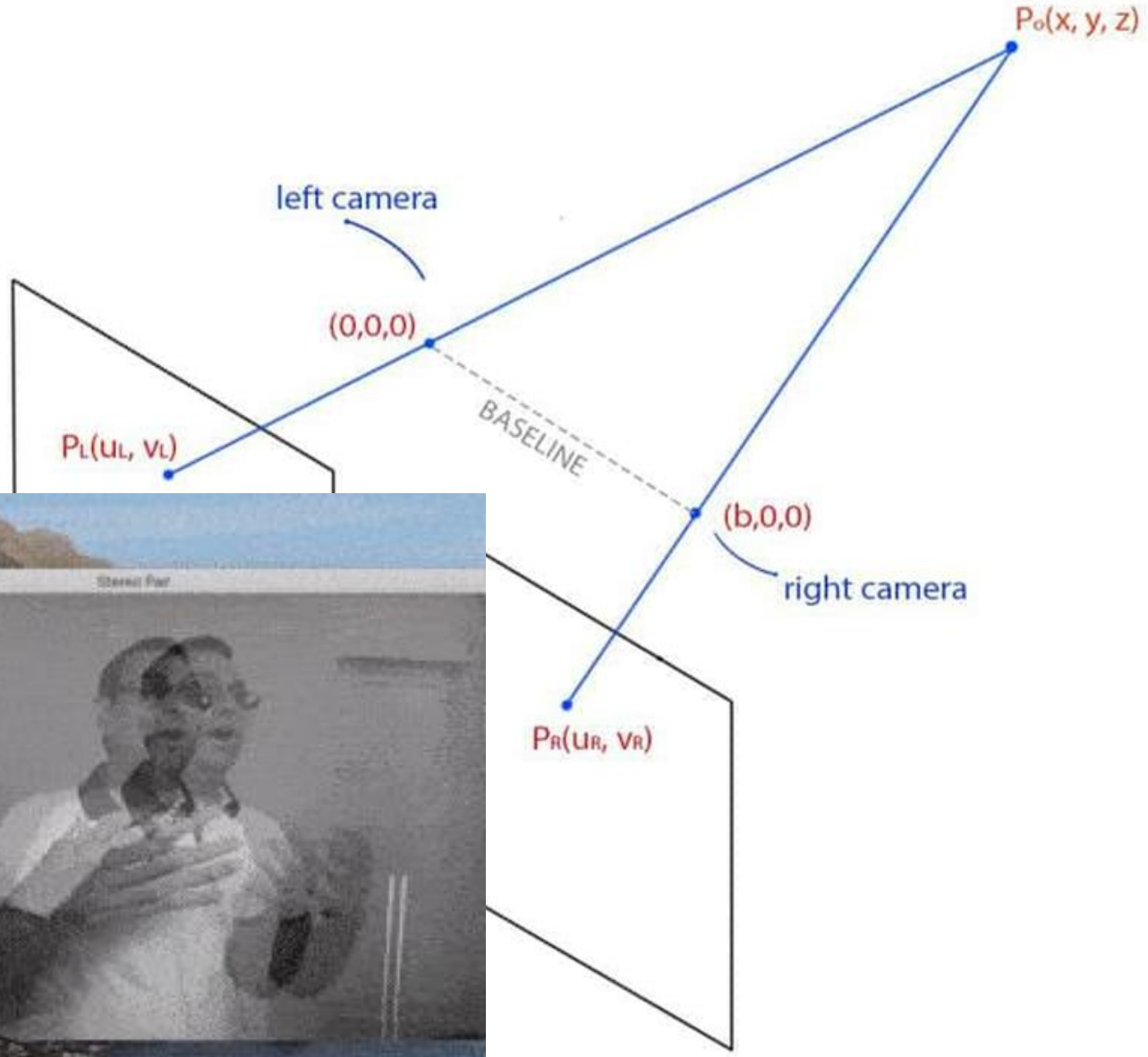


$$u_R = f\frac{x-b}{z} + p_x$$

Estimating depth

$$z = \frac{fb}{u_L - u_R}$$





Today’s agenda

- Properties of Perspective transformations
- Introduction to Camera Calibration
- Linear camera calibration method
- Calculating intrinsics and extrinsics
- Depth estimation

Next lecture

Recognition