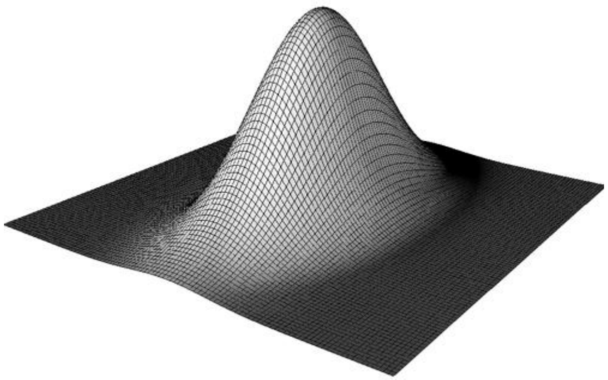
Computer Vision and Pattern Recognition

L22-23. Camera Geometry



Dr John Collomosse
J.Collomosse@surrey.ac.uk

Centre for Vision, Speech and Signal Processing
University of Surrey

Learning Outcomes

After attending this lecture you should be able to:

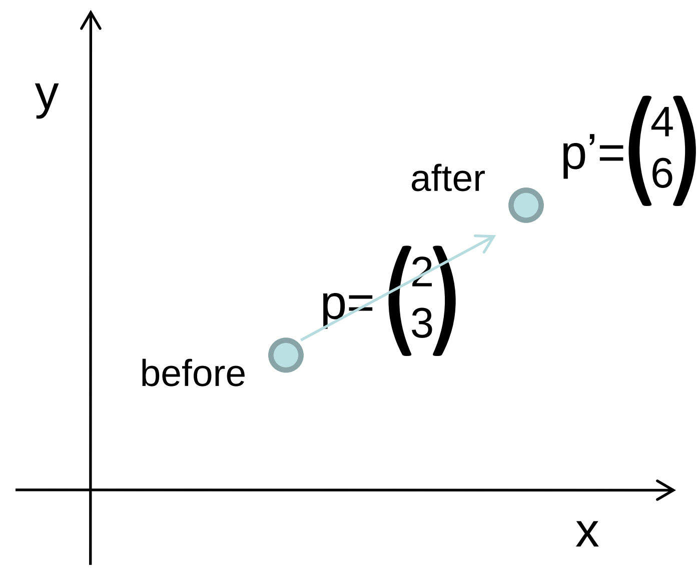
- Describe how matrices can be combined with homogeneous coordinates to perform rigid body transformations in 2D and 3D
- Combine matrices in 2D and 3D to create compound transformations
- Describe the pin-hole perspective projection and orthographic projection models
- Show how perspective projection can be written as a matrix in the framework of homogeneous coordinates
- Perform 3D simple reconstruction using a visual hull
- Describe how a pair of 2D points may be triangulated to recover a 3D point under a calibrated camera setup

2D Linear Transforms

A polygon **p**={p_1, p_2, ... , p_n} can be transformed using a 2x2 matrix **M**

$$\underline{p'} = \underline{\underline{M}}\,\underline{p}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}}_{\text{Scale by factor of 2}} \begin{pmatrix} x \\ y \end{pmatrix}$$



$$\underline{\underline{M}} = \underbrace{\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}}_{\text{Scale}} \qquad \underline{\underline{M}} = \underbrace{\begin{bmatrix} 1 & q \\ 0 & 1 \end{bmatrix}}_{\text{Shear}} \qquad \underline{\underline{M}} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{\text{Rotation anticlockwise about origin by } \theta}$$

Homogeneous coordinates

We can also perform translation with matrix , but to do so we have "invent" an extra coordinate – the **homogeneous coordinate**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous coordinate

To get the actual location of the point after the transform we must divide by the homogeneous coordinate i.e.

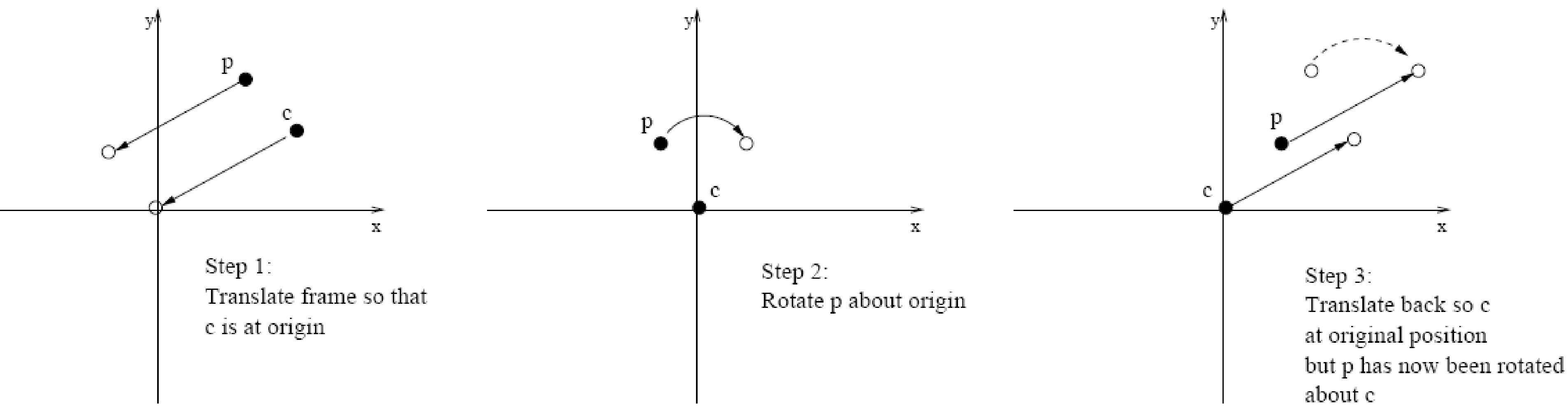
Output x = x' / 1

Output y = y' / 1

In this case the division has no effect as the bottom row of [0 0 1] means the homogeneous coordinate output will always equal the input i.e. = 1

Rotation about arbitrary point

Matrices can be multiplied to create compound operations



$$\underline{p'} = \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix} \underline{p}$$

$$\underline{p'} = \underline{\underline{T}}^{-1} \underline{\underline{R}}(\theta) \underline{\underline{T}}\underline{p}$$

order of operations

$$\underline{p'} = \underline{\underline{M}}\underline{p}$$

Transforming 3D points

3D points can also be transformed using matrices.

Our homogeneous coordinate is the 4th coordinate (and matrices 4x4)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\underline{\underline{T}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Translation}} \qquad \underline{\underline{S}} = \underbrace{\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Scale}}$$

3D Rotation (Euler angles)

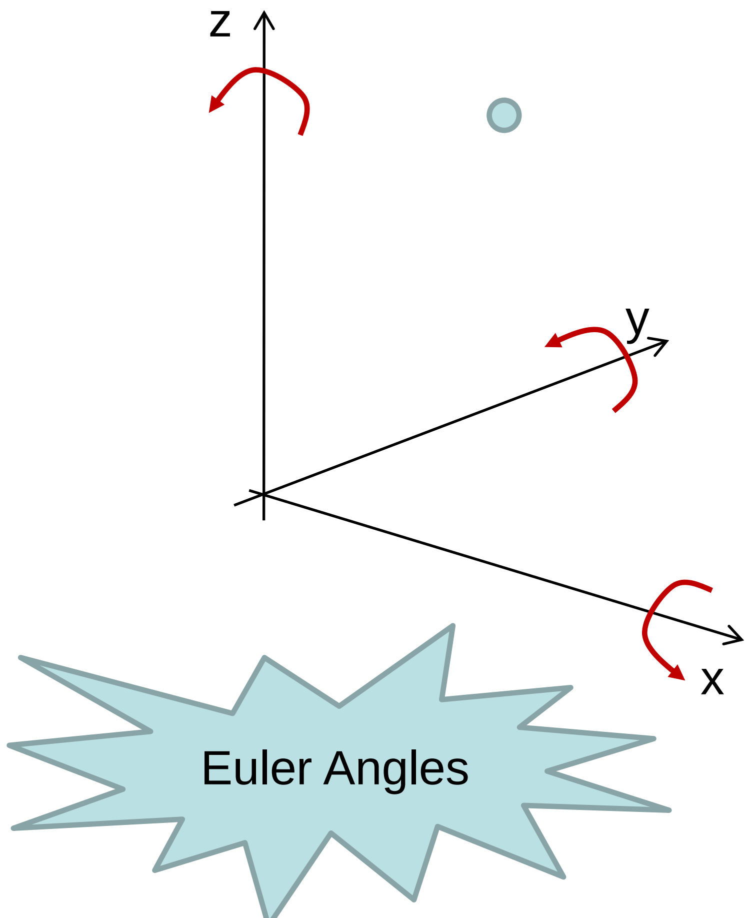
In 2D we rotate about a point (the matrix rotates about the origin)

In 3D we rotate **about an axis** (there are 3 matrices; i.e. for x, y and z)

$$\underline{\underline{R_x}}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\underline{R_y}}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\underline{R_z}}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

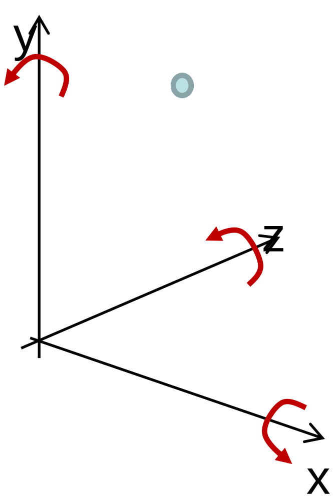


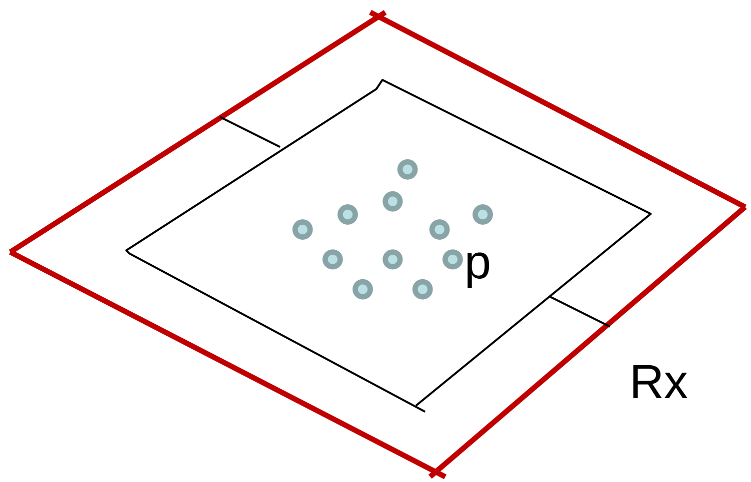
Euler Angles (Gimbal system)

With Euler angles we can rotate a certain amount around the x, y and z axis in turn to produce any rotation.

But which order to rotate in?

$$R_z R_y \boxed{R_x p}$$



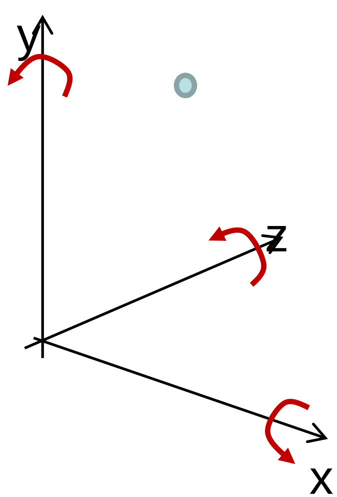


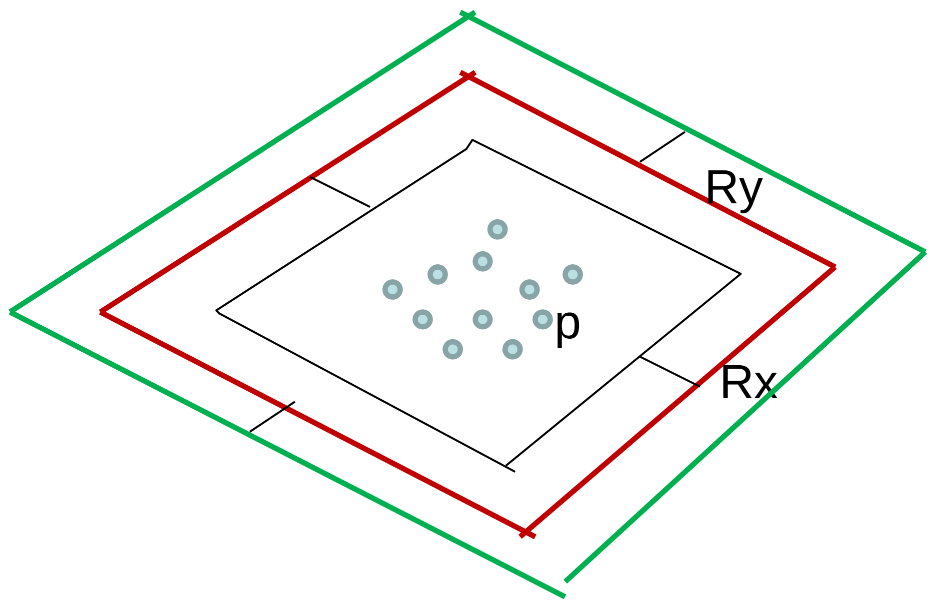
Euler Angles (Gimbal system)

With Euler angles we can rotate a certain amount around the x, y and z axis in turn to produce any rotation.

But which order to rotate in?

$$R_z \boxed{R_y R_x p}$$



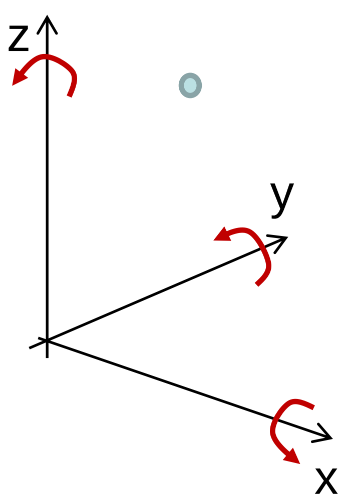


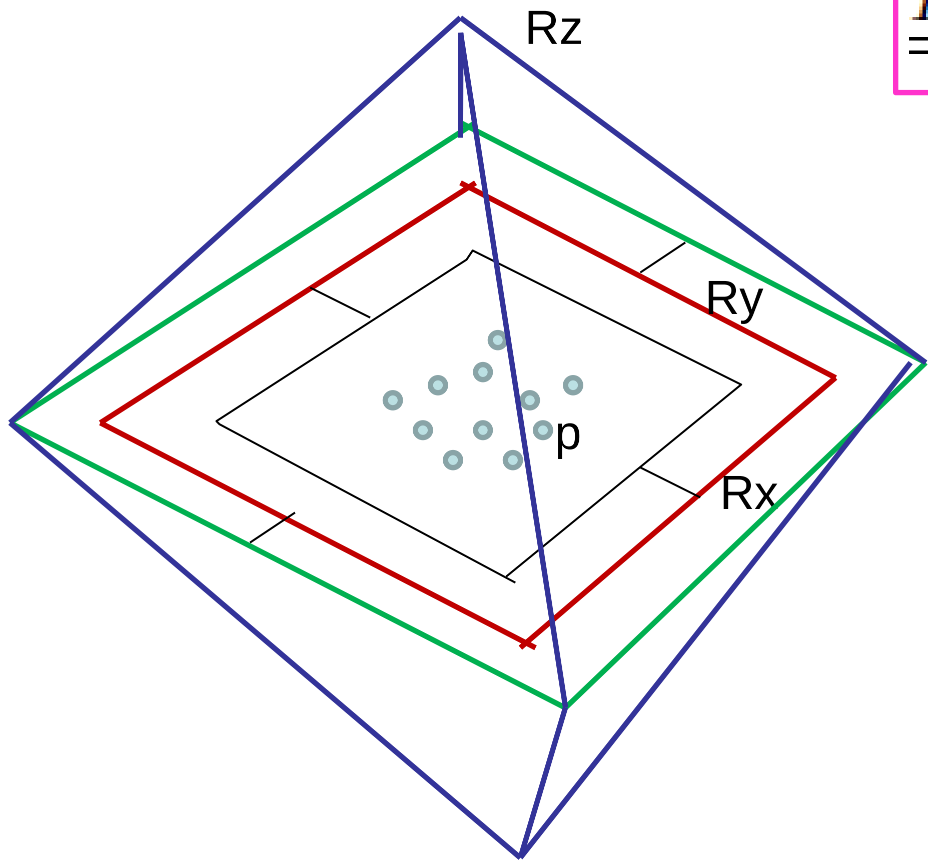
Euler Angles (Gimbal system)

With Euler angles we can rotate a certain amount around the x, y and z axis in turn to produce any rotation.

But which order to rotate in?

$$\boxed{R_z R_y R_x p}$$





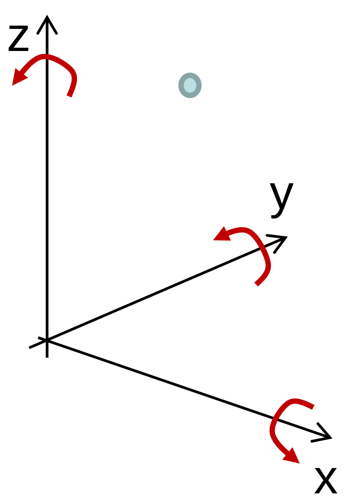
Euler Angles (Gimbal system)

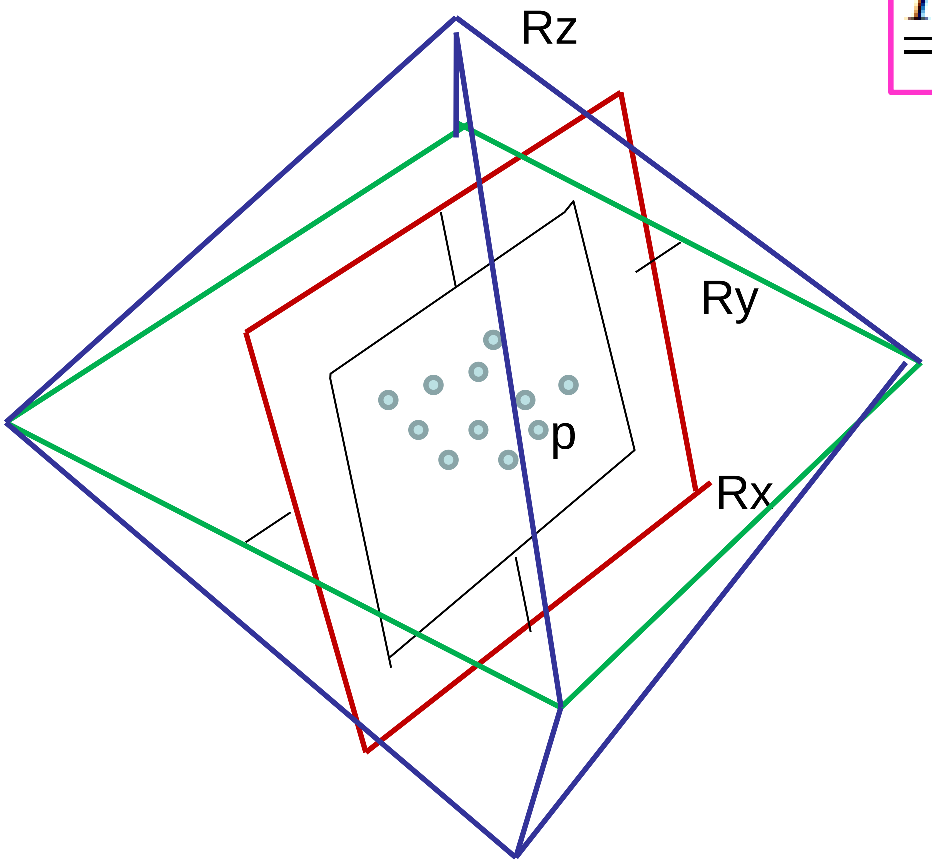
A Gimbal compass (nautical compass) using these principles

Gimbal Lock

When the middle Euler angle (**Ry** here) is rotated 90 degrees the inner reference frame (**Rx**) becomes aligned with the outer (**Rz**)

$$\boxed{\underline{\underline{R_z}}\,\underline{\underline{R_y}}\,\underline{\underline{R_x p}}}$$





Homogeneous 3D transforms

So, in homogeneous form 3D points can be manipulated to produce:

Linear transforms

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Affine transforms

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

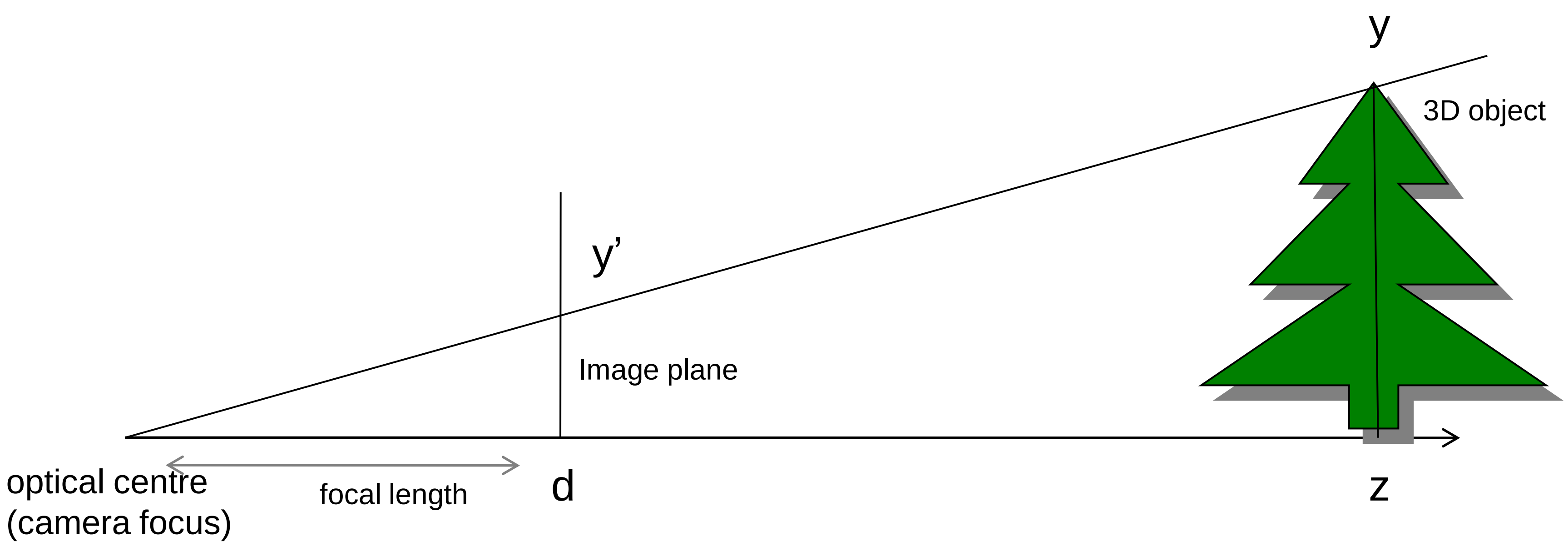
In these transformations the homogeneous coordinate will always be unchanges (i.e. 1)

But we can also create projective transformations using 4x4 matrices that deviate from [0 0 0 1] in their bottom row.

Pin-hole camera model

3D transformations are used frequently in Computer Graphics to move around 3D objects to build graphics scenes.

Eventually the 3D scene must be projected onto a 2D screen for viewing



$$\frac{y'}{d} = \frac{y}{z}$$

$$\boxed{y' = \frac{dy}{z}}$$

Camera Obscura

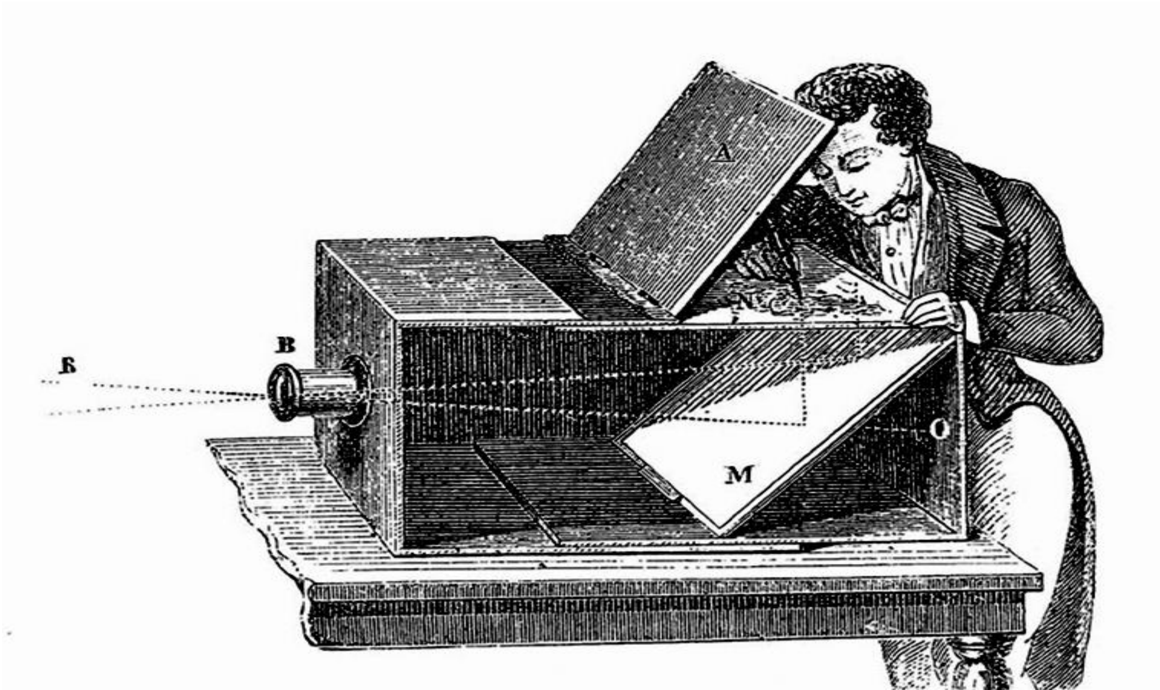
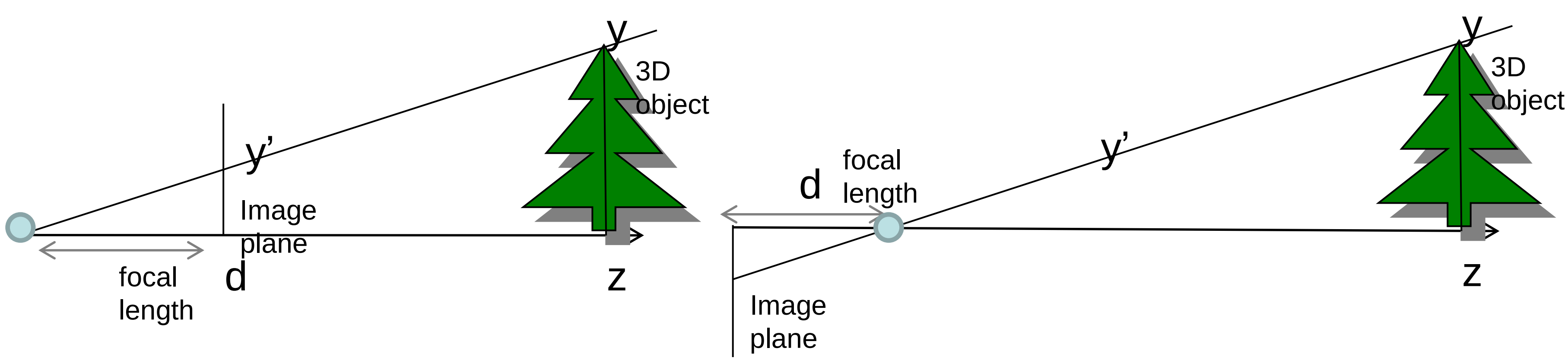


Fig. 434.

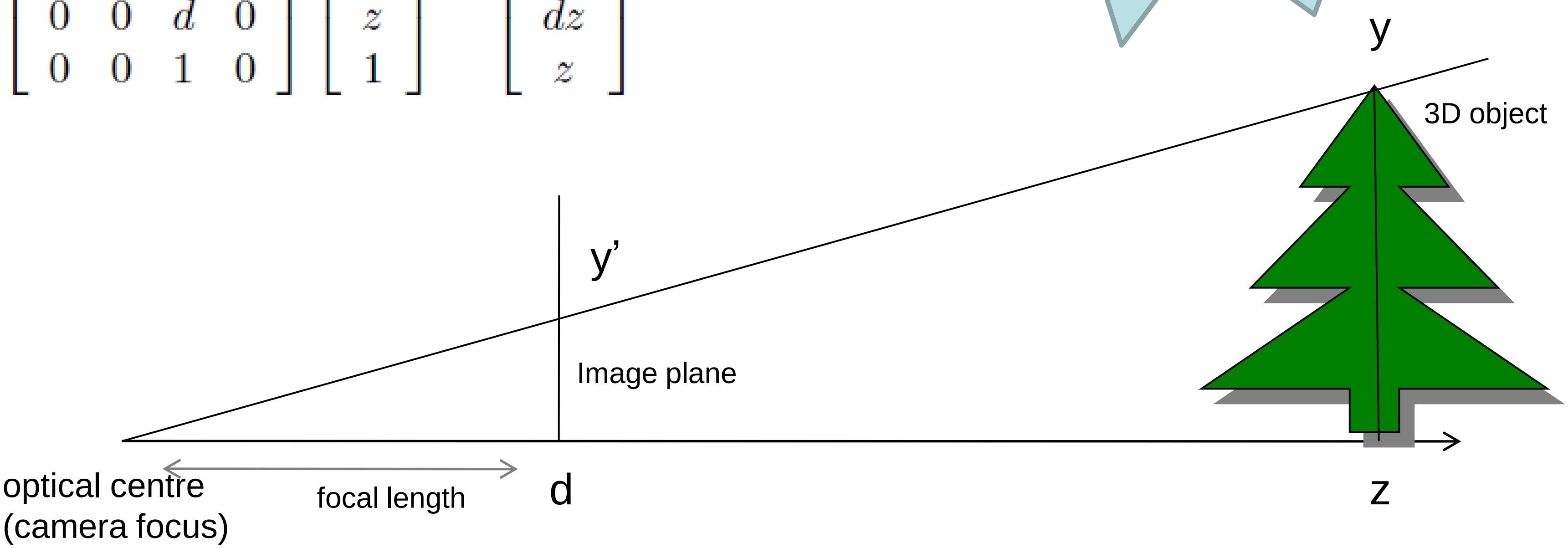


Pin-hole camera model

The perspective projection of the pin-hole camera can also be expressed via a 4x4 matrix.

$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ dz \\ z \end{bmatrix}$$

How do you think field (angle) of view varies with focal length?



$$\frac{y'}{d} = \frac{y}{z}$$

$$\boxed{y' = \frac{dy}{z}}$$

Focal Length vs Field of View

Field of view is a consequence of focal length (inverse proportional)

24mm

50mm

200mm

800mm

Perspective Projection Matrix

The perspective projection of the pin-hole camera can also be expressed via a 4x4 matrix.

$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ dz \\ z \end{bmatrix}$$

$$\underline{\underline{P}} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\underline{\underline{P}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

Shorthand versions that drop 'z'

$$\boxed{\underline{\underline{P}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} dx \\ dy \\ z \end{bmatrix}}$$

Don't forget to divide through by the homogeneous coordinate

Vanishing points

Orthographic Projection Matrix

Orthographic projection is the simplest form of 3D to 2D projection

You simply drop the z coordinate. This approximates

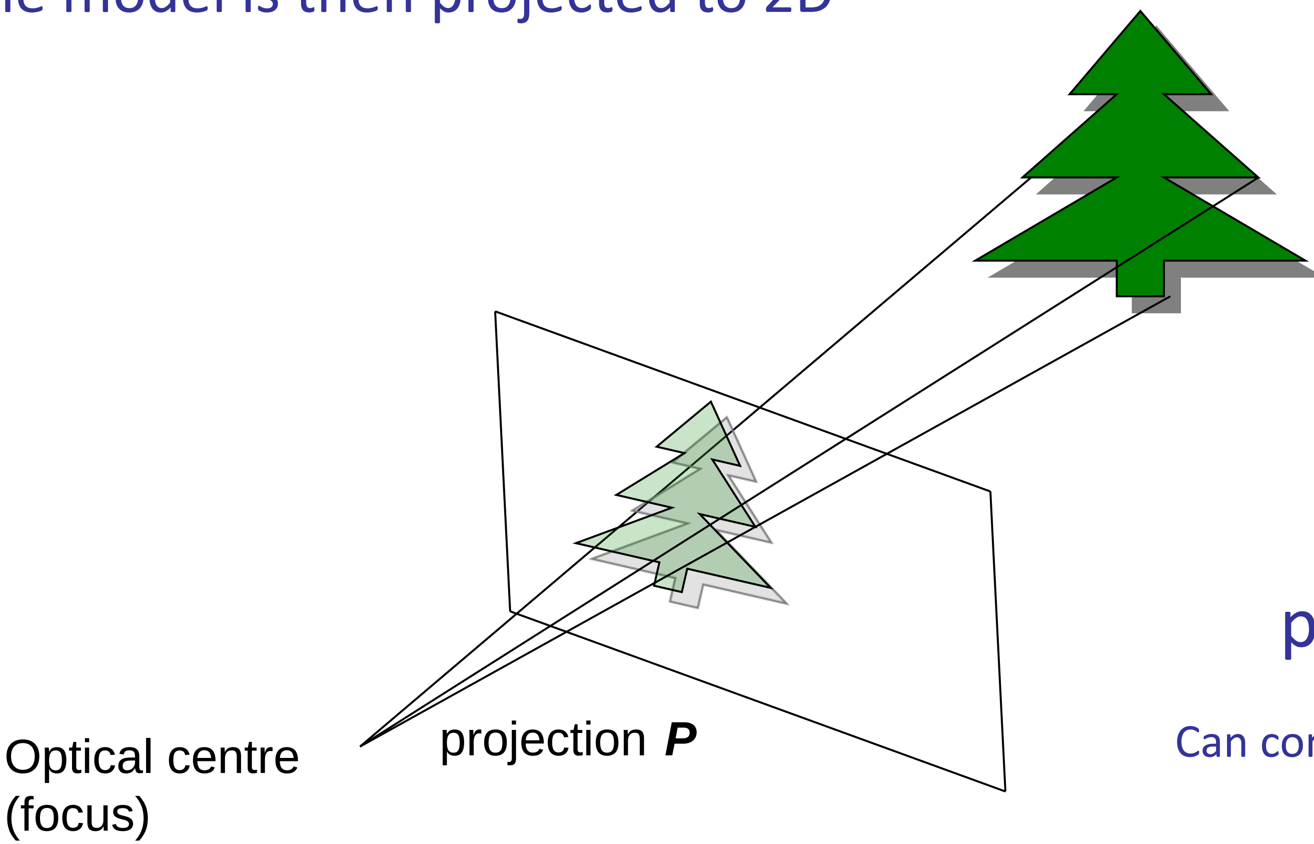
$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ? \end{bmatrix}$$

$$\underline{\underline{P}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ? \end{bmatrix}$$

Computer Graphics

In a standard computer graphics pipeline we position an object in 3D using some combination of rotation (orientation) and translation (shift)

The model is then projected to 2D



$$p' = \underbrace{P\ R\ T}_{\text{Can combined to a single matrix}}\ p$$

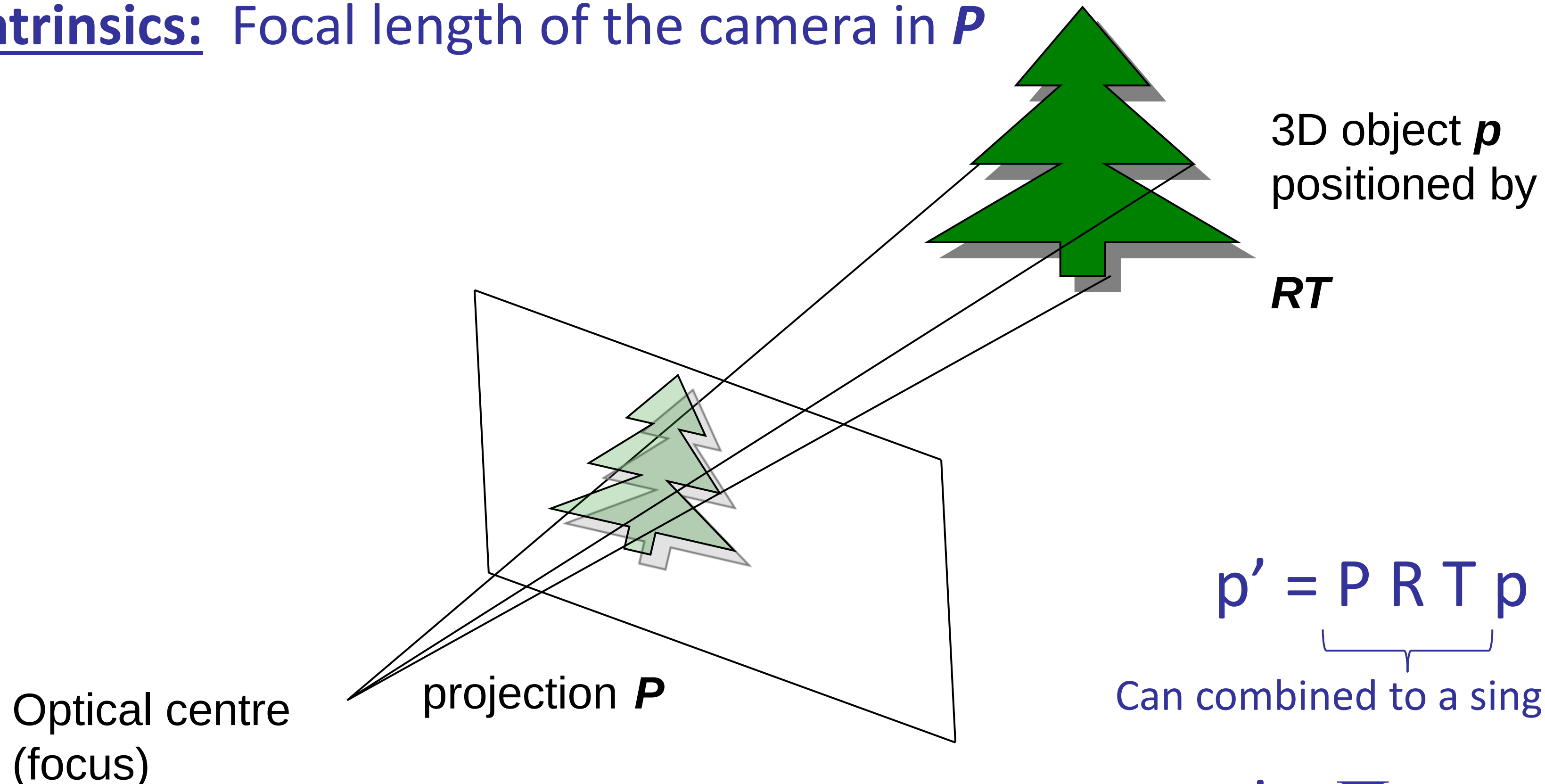
$$p' = \Pi\ p$$

Computer Vision

The imaging process can be considered mathematically identical.

Extrinsics: Relative position the camera and object in the world ***RT***

Intrinsics: Focal length of the camera in ***P***



$$p' = \underbrace{P\ R\ T}_{}\ p$$

Can combined to a single matrix

$$p' = \Pi\ p$$

Computer Vision

The imaging process can be considered mathematically identical.

Extrinsics: Relative position the camera and object in the world ***RT***

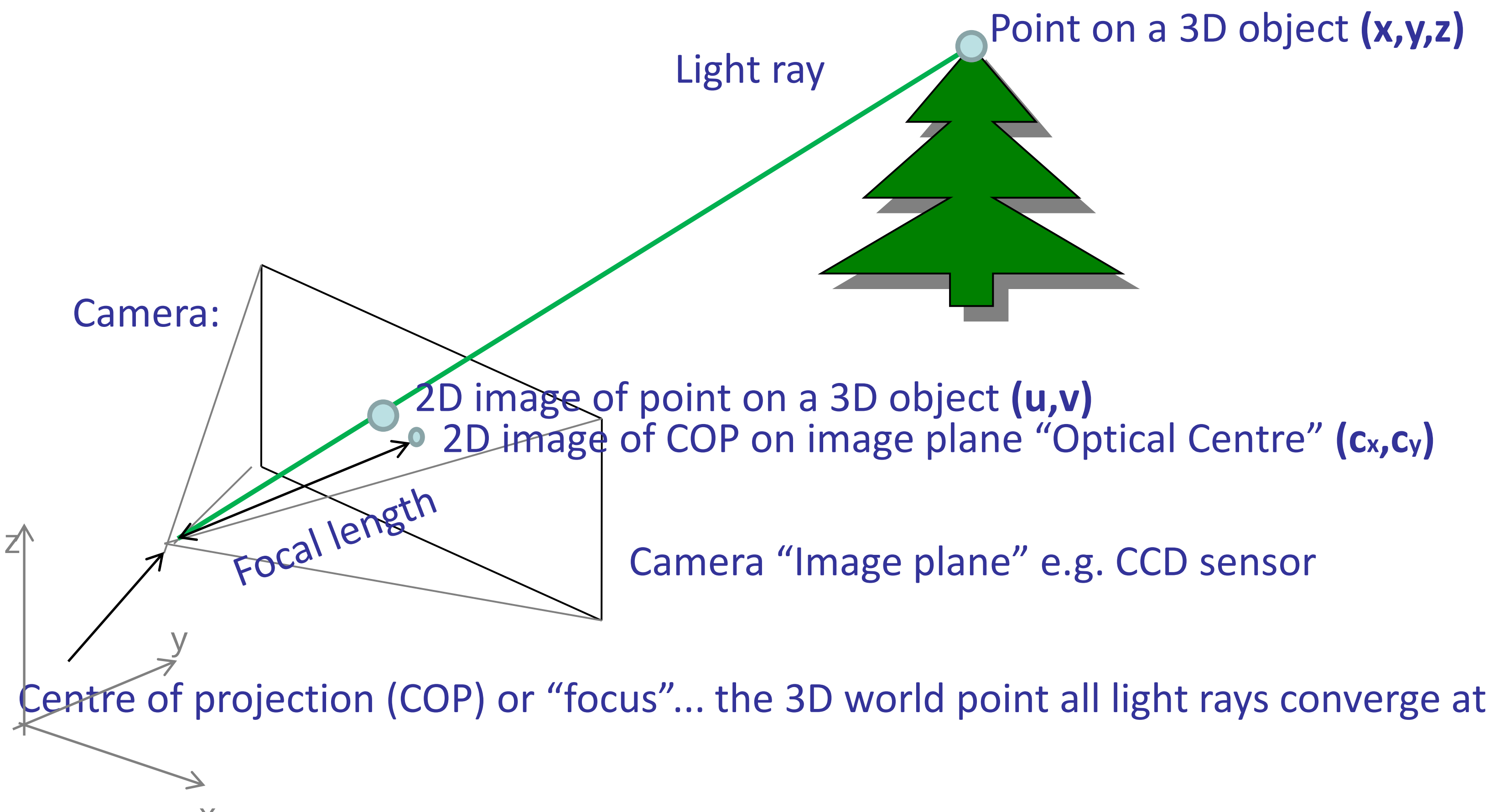
Intrinsics: Focal length of the camera in ***P***

$$\mathbf{\Pi} = \underbrace{\mathbf{K}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} & \mathbf{R} & & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} & \mathbf{I}_{3\times 3} & & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{translation}}$$

(We will be coming back to this later)

Camera Geometry (3D)

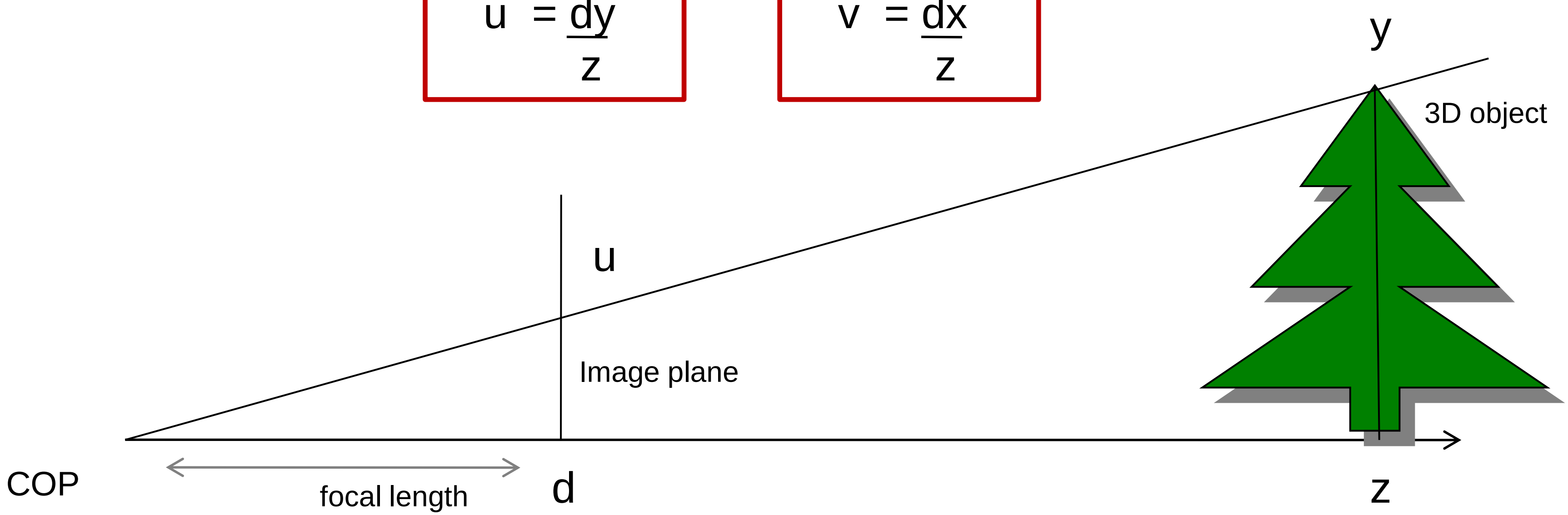
Recall perspective projection (for a pin-hole i.e. no lens camera)



Camera Geometry (simpler)

This simpler 2D diagram is equivalent to the last slide, only addressing the y coordinate. The ratio for x is identical in form.

$$\boxed{u = \frac{dy}{z}} \qquad \boxed{v = \frac{dx}{z}}$$



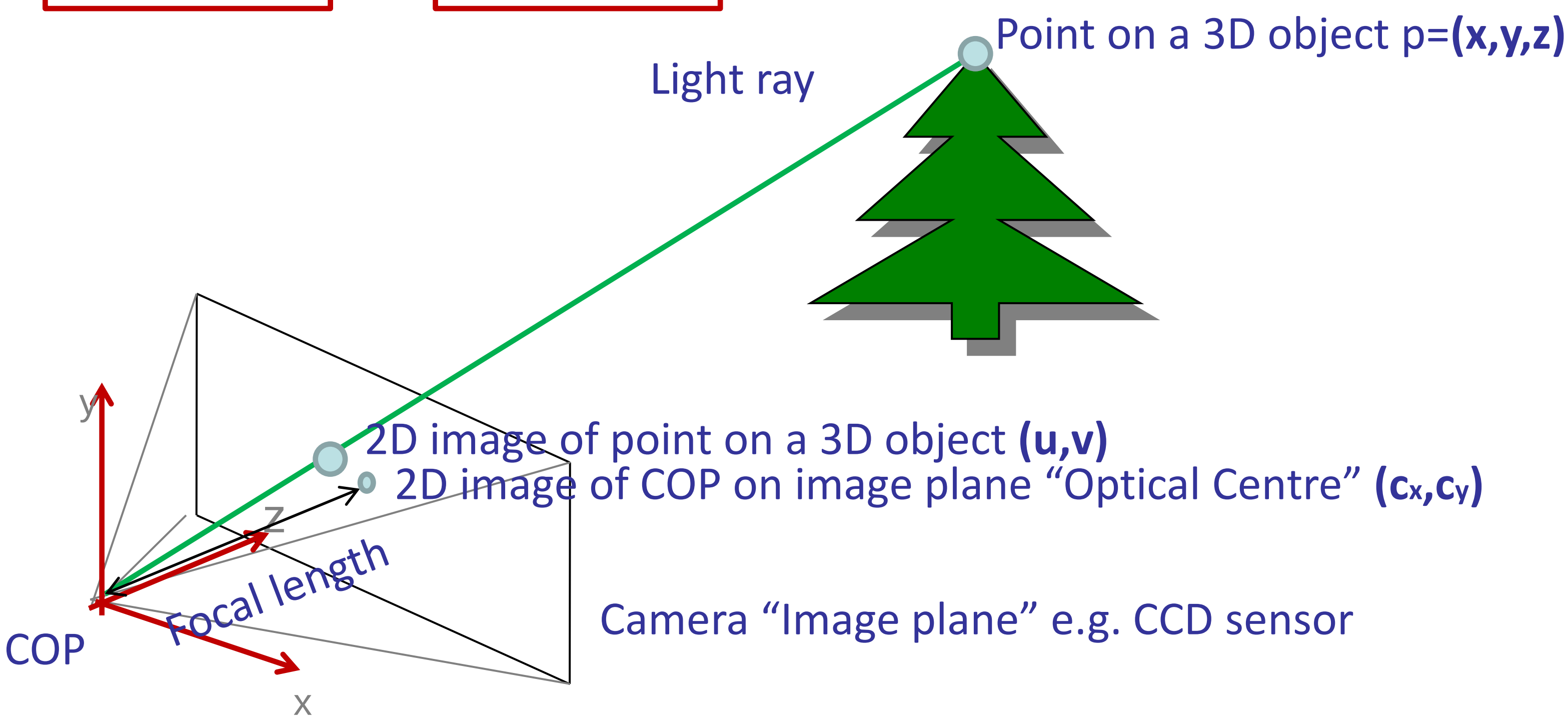
Camera Geometry

UNIVERSITY OF SURREY

So given a point x,y,z in 3D space it's 2D projection is (u,v)

$$u = \frac{dy}{z}$$

$$v = \frac{dx}{z}$$



BUT: it's only this simple if camera's COP is at the world origin, and camera aligned as above

AND: the (u,v) coordinate system is relative to the Optical Centre, not the image top-left

Camera Geometry

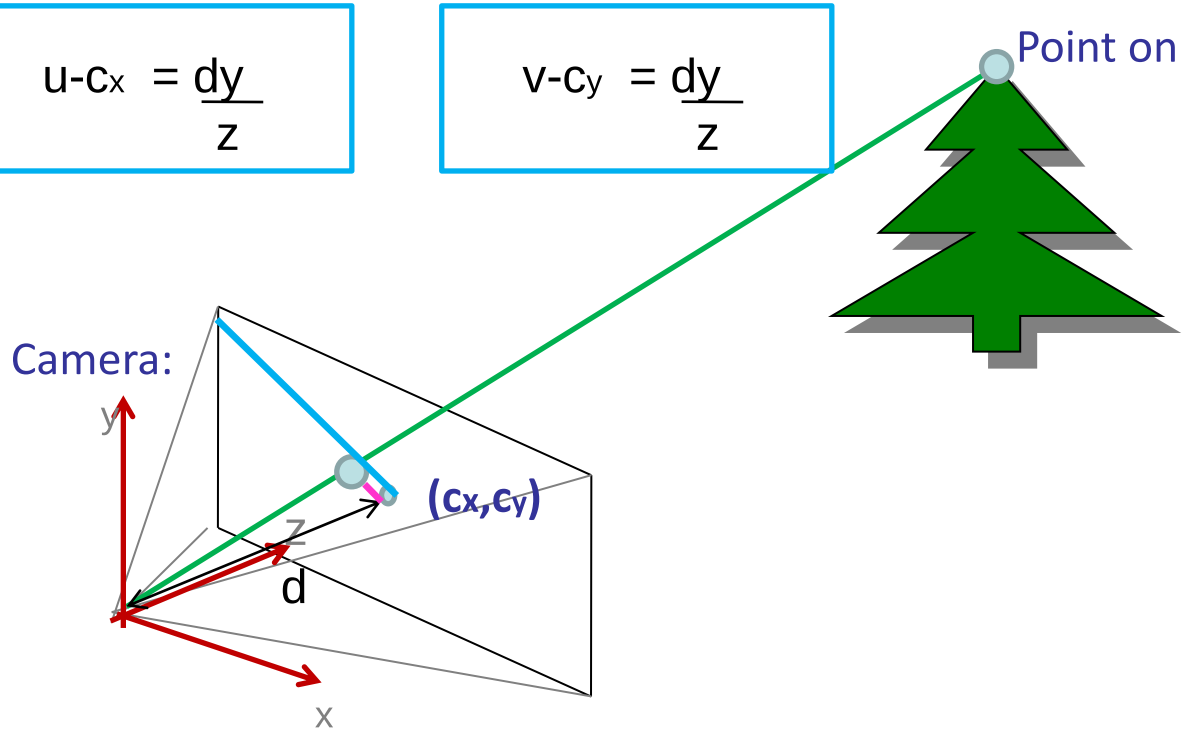
Dealing with the optical centre...

$$u = \frac{dy}{z}$$

$$v = \frac{dx}{z}$$

$$u - c_x = \frac{dy}{z}$$

$$v - c_y = \frac{dy}{z}$$



$$u = \frac{dx}{z} + c_x$$

$$v = \frac{dy}{z} + c_y$$

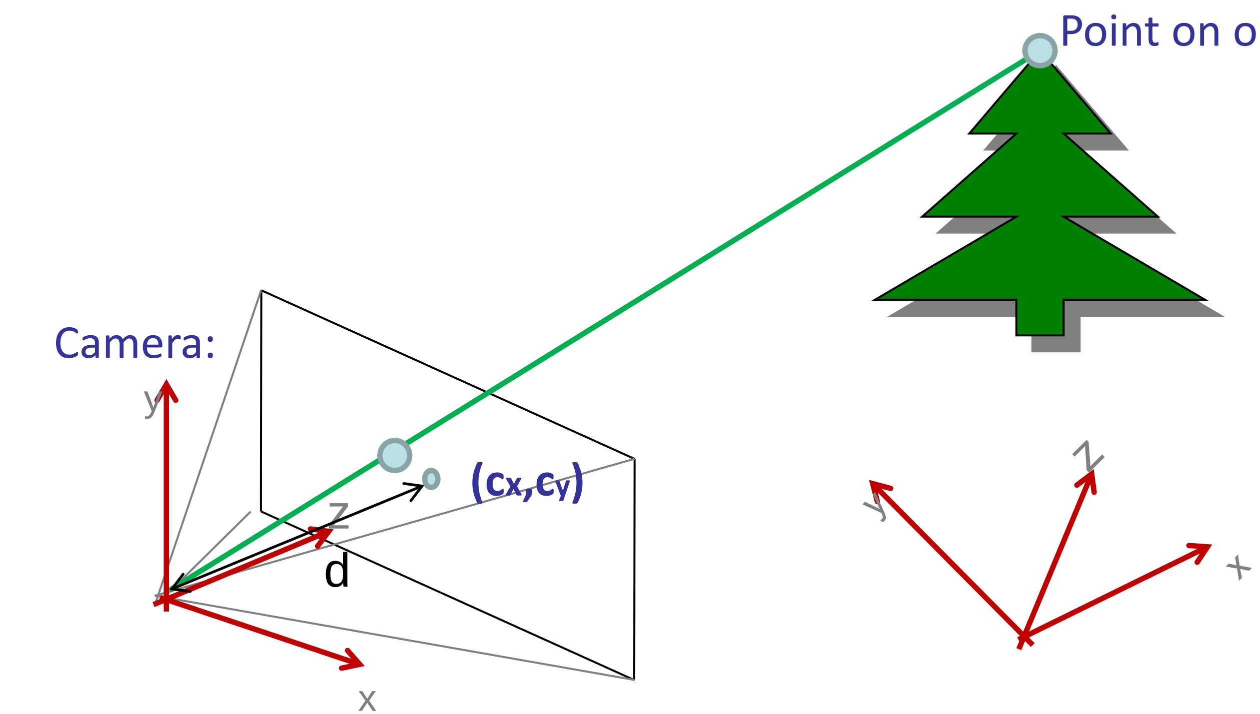
Camera Geometry

UNIVERSITY OF SURREY

Dealing with the camera being anywhere in the space

It could be translated any where... (T)

It could also be oriented any way... (R)



$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{TR}p$$

$$u = \frac{dx}{z} + c_x$$

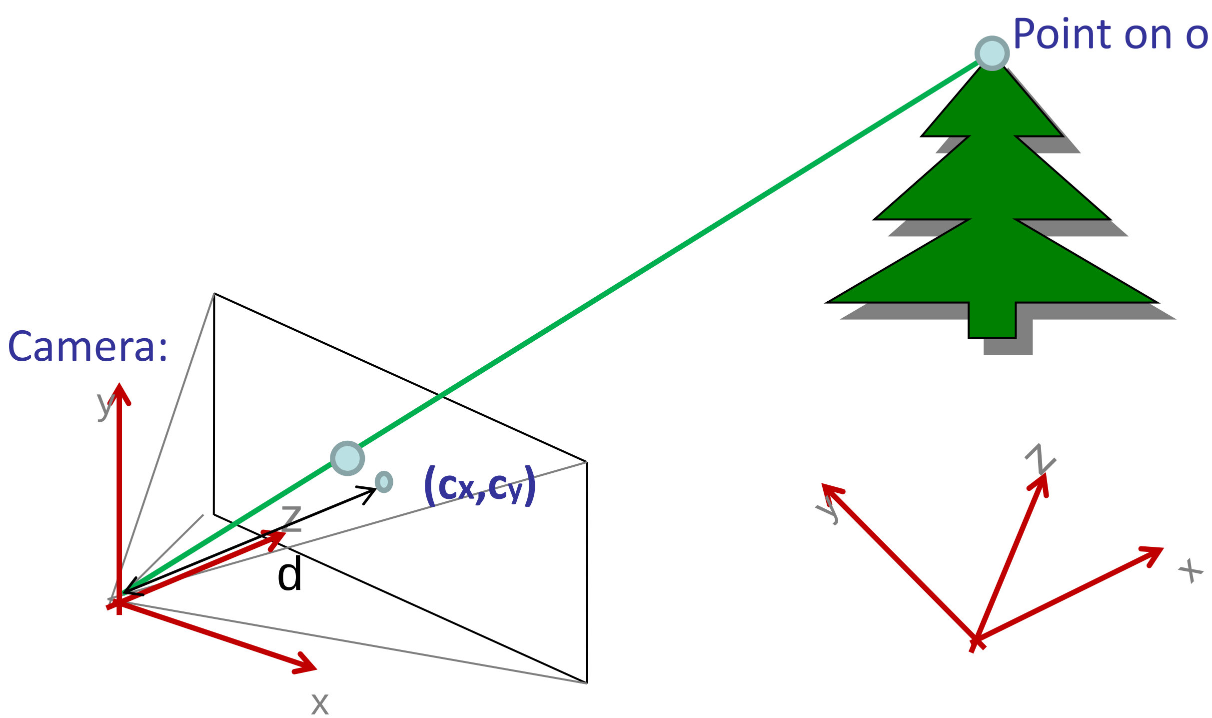
$$v = \frac{dy}{z} + c_y$$

Camera Geometry

To do this we need to know where the camera is... **Extrinsics** (R,T)

And it's internal parameters... **Intrinsics** (d, c_x, c_y)

<u>If we know the intrinsics and extrinsics we have a calibrated camera</u>



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{TR}p$$

$$u = \frac{dx}{z} + c_x$$

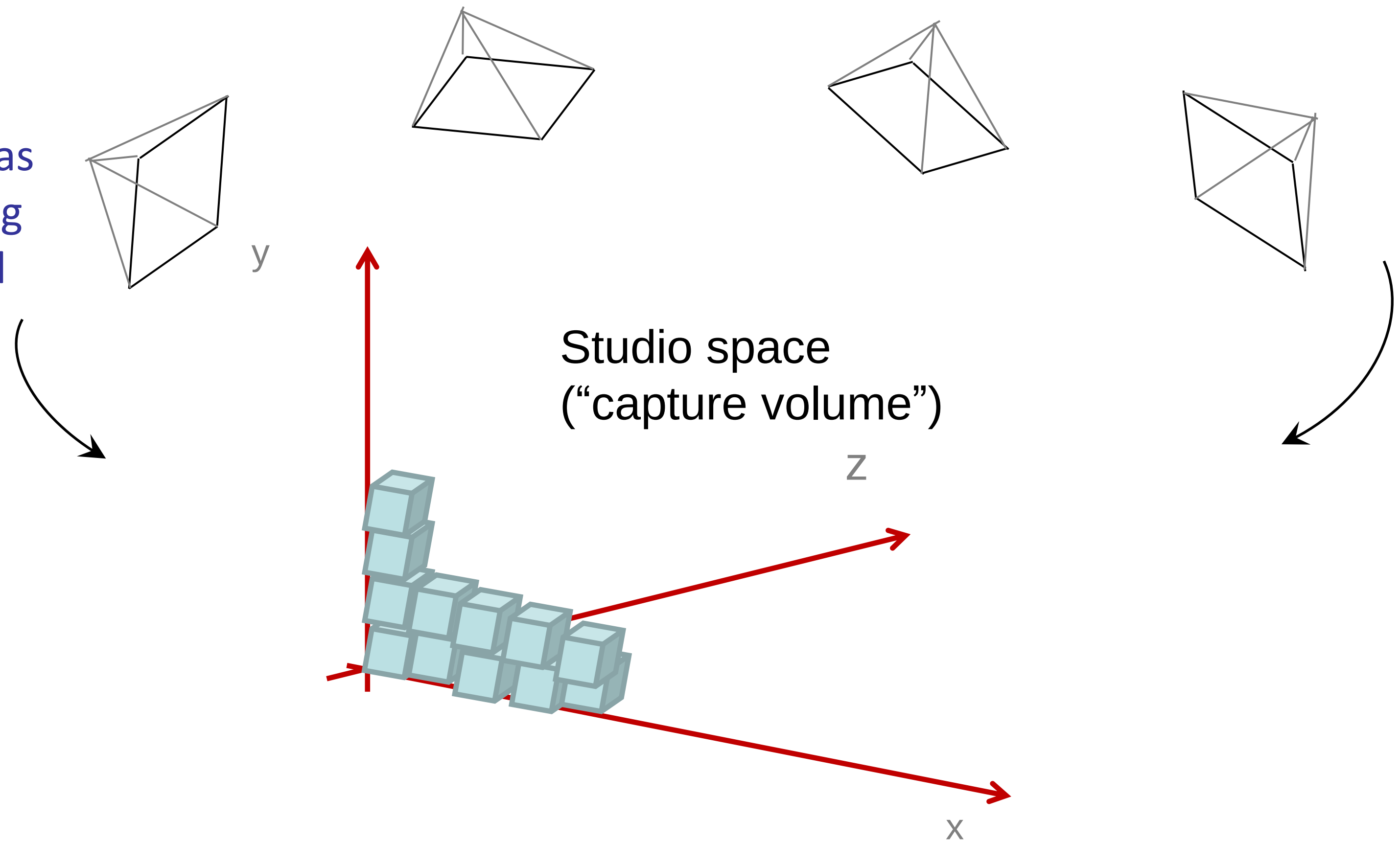
$$v = \frac{dy}{z} + c_y$$

Basic 3D reconstruction

If we have more than 1 camera e.g. in a studio we can use these equations to make a 3D model of objects in the studio.

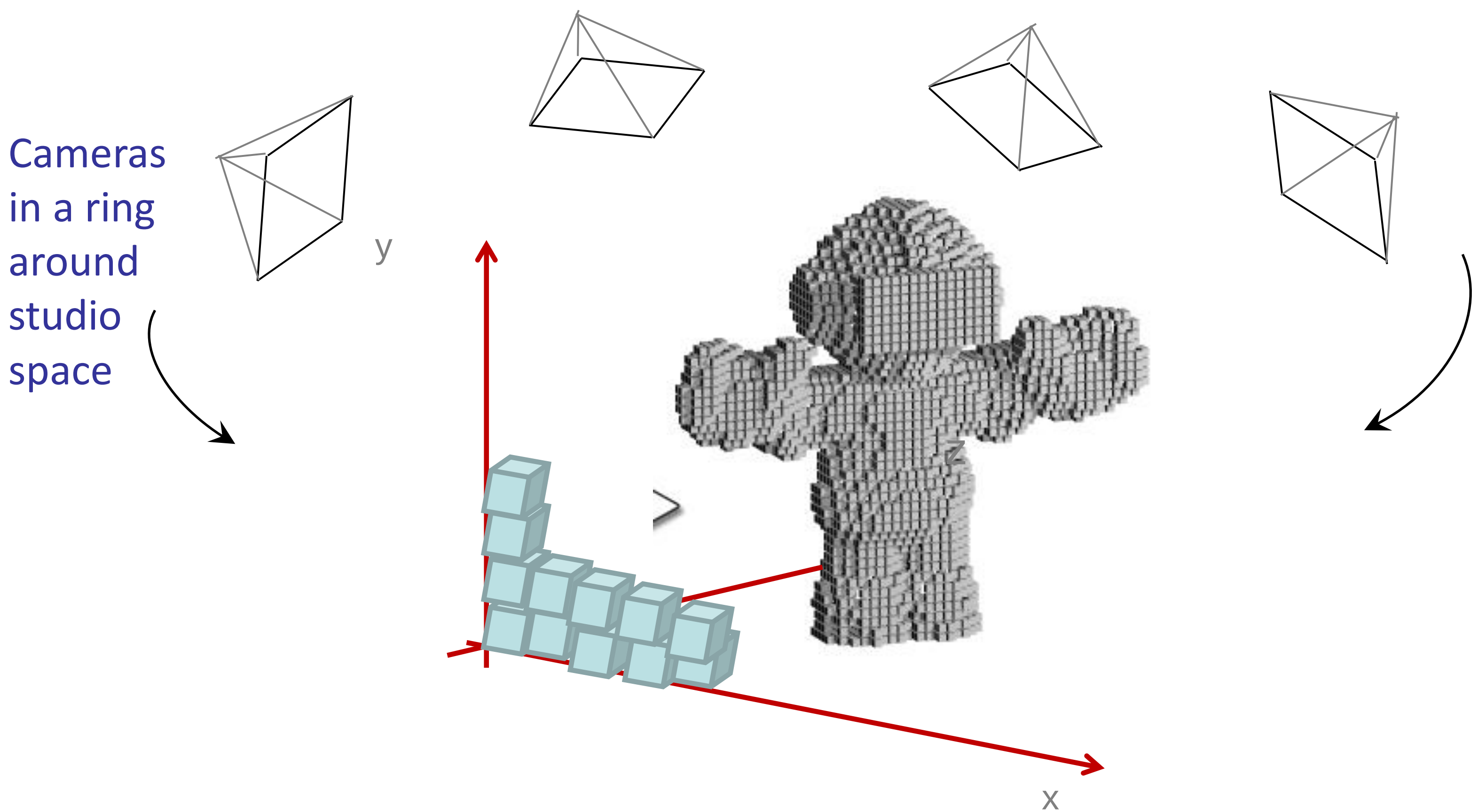
Imagine **carving the studio into many small cubes** (**voxels** = 3D pixels)

Cameras in a ring around studio space



Visual Hull

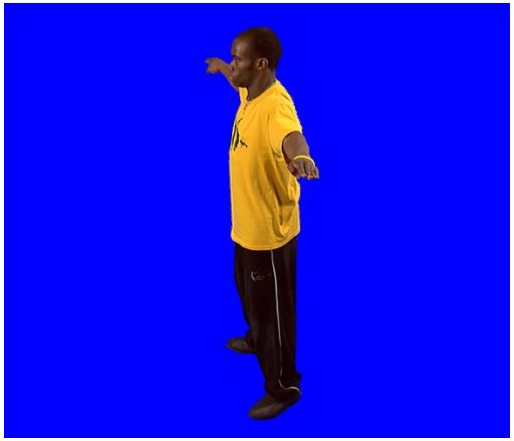
If we **work out which voxels are occupied** by a person/object then we will have produced a 3D model. We call this the **visual hull.**



Visual hull

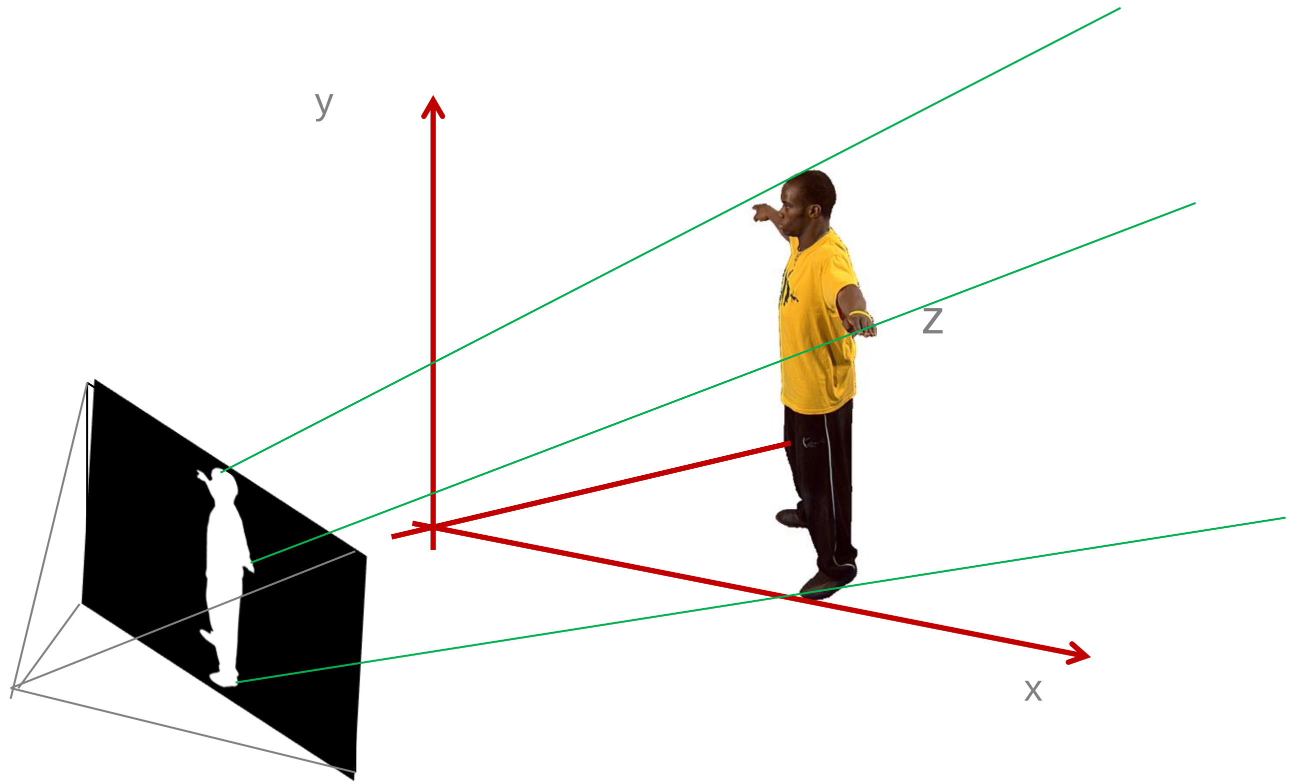
Consider an image of an actor against a blue background.

We already know how to extract a mask of the actor.



Visual hull

Consider a mask captured from a single camera

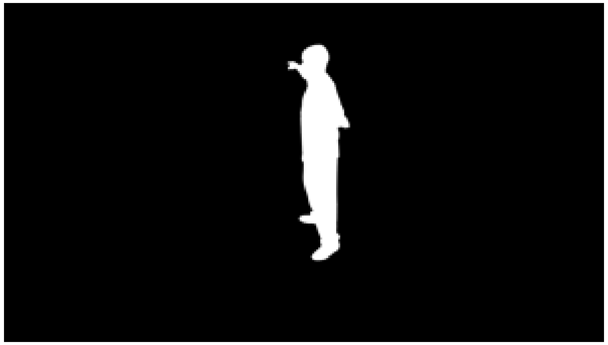
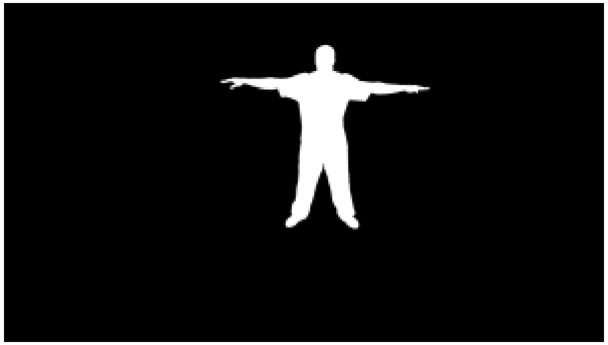
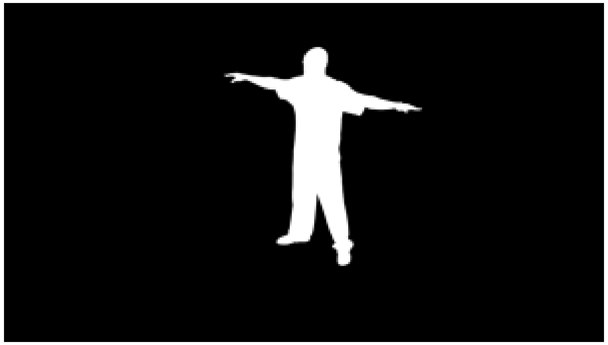


Consider all voxels in the studio that might project onto the camera’s image plane to cast this silhouette .

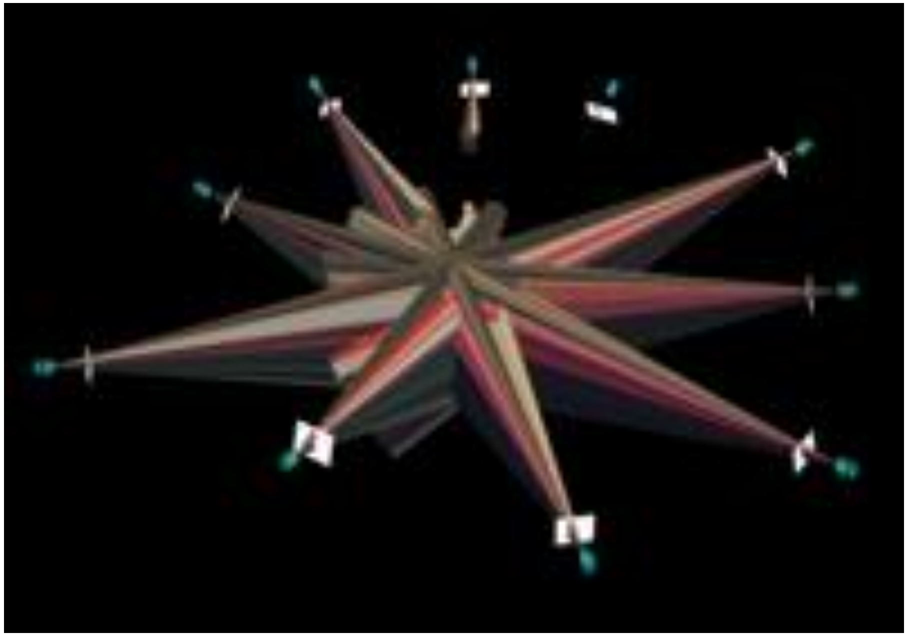
This resembles a “prism” through the studio space

Visual hull

Now consider the silhouettes from multiple views of the same object

The prisms from each camera will intersect at the object



Visual hull

Reconstruction algorithm:

For each voxel (x,y,z) in the capture volume

    Let counter=0

    For each calibrated camera

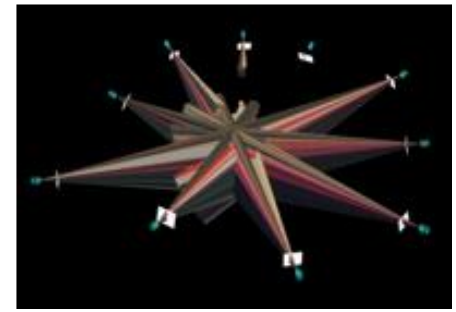
        Project (x,y,z) to (u,v)

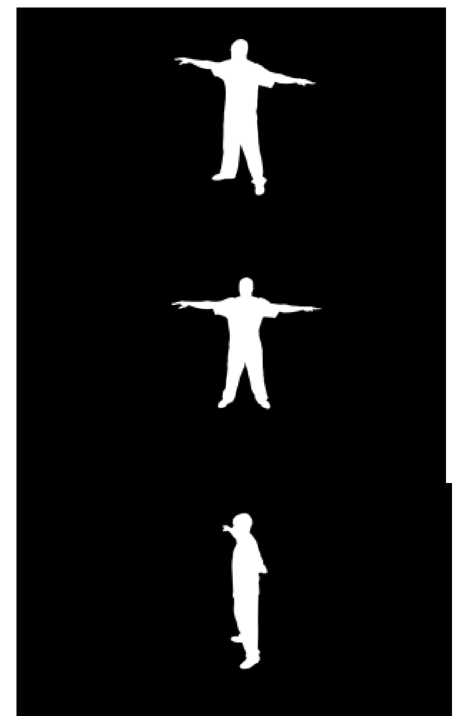
        If (u,v) is set, then counter =counter+1

    End

    If counter == total cameras, then voxel is part of hull

End





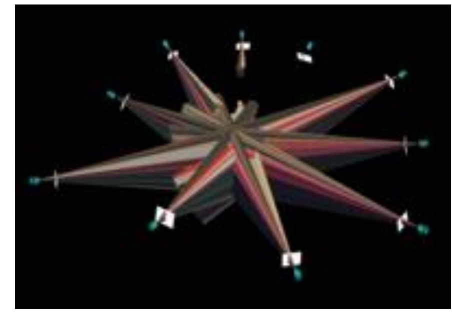
Projecting (x,y,z) to (u,v)

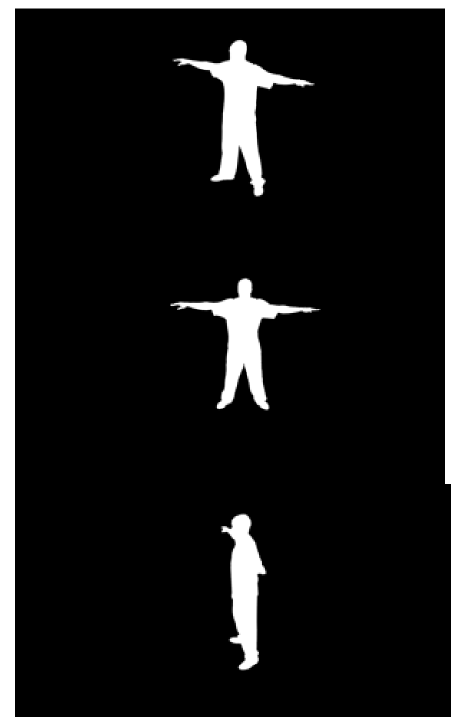
Use our knowledge of **extrinsics** and **intrinsics** for each camera

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{TR}p$$

$$u = \frac{dx}{z} + c_x$$

$$v = \frac{dy}{z} + c_y$$





Recall:

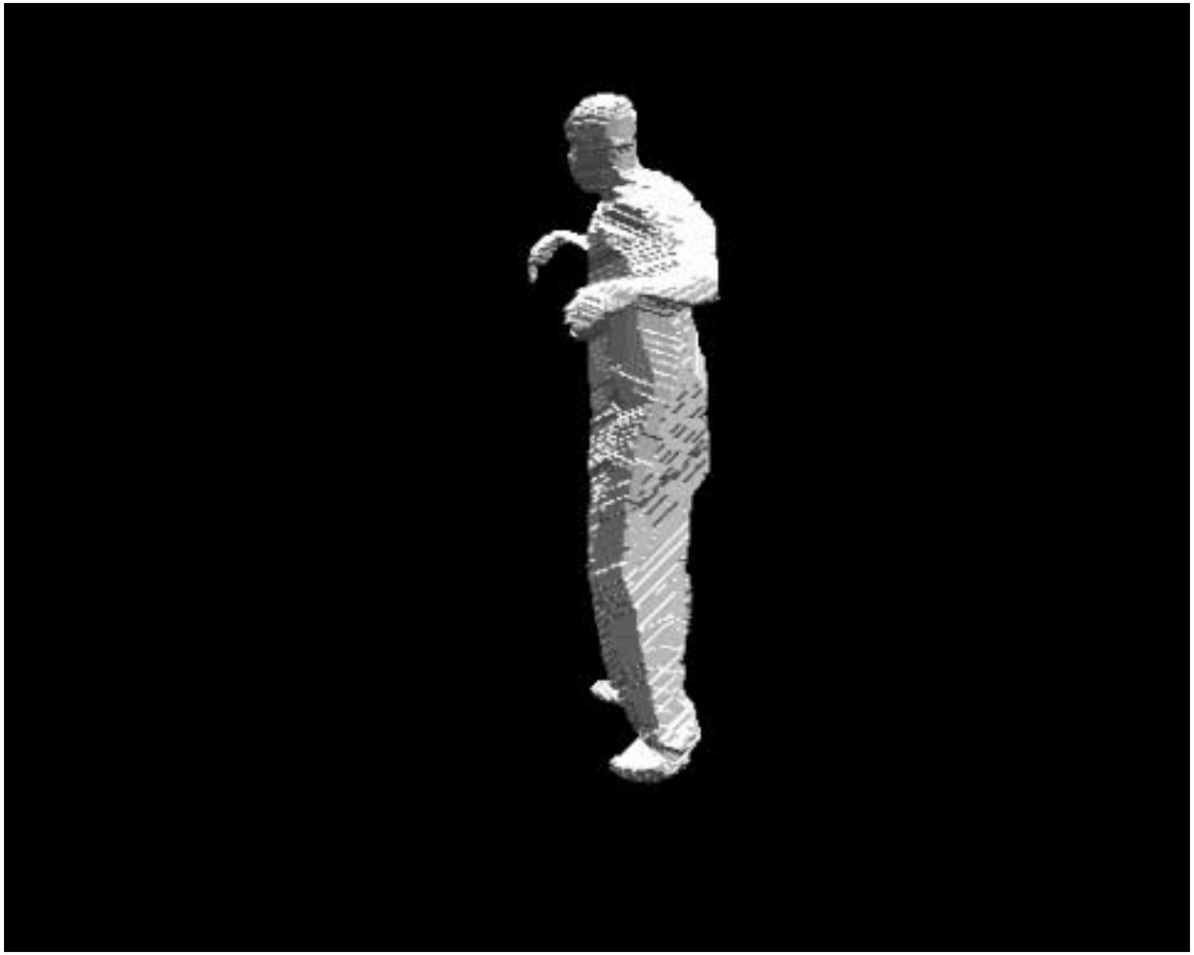
R,T - the extrinsics

d, c_x, c_y - the intrinsics

Visual Hull - Result

UNIVERSITY OF SURREY

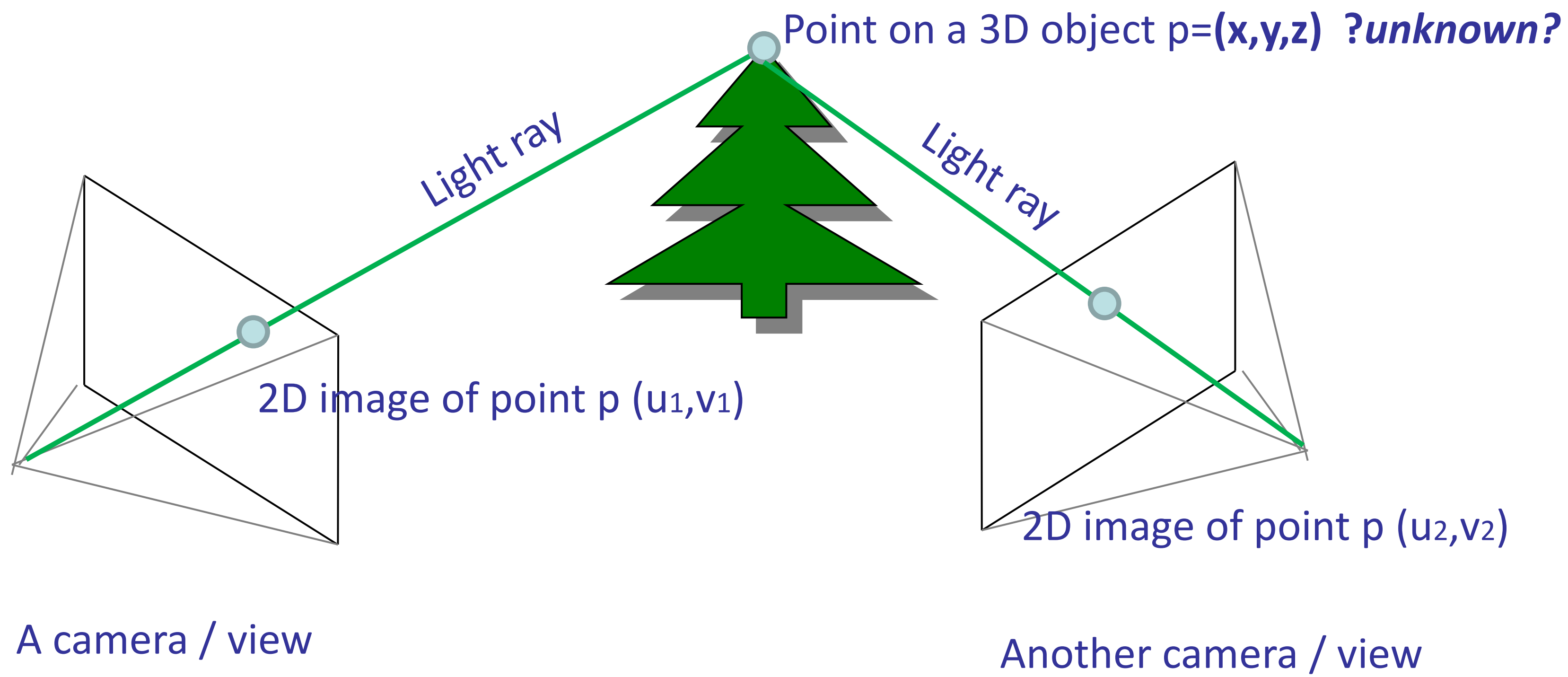
If we capture synchronised video from multiple cameras we can use this approach to make a 3D video i.e. a sequence of visual hulls



Triangulation

The visual hull projected **from 3D to 2D** to make 3D reconstructions

But it is also useful to go **from 2D to 3D** to triangulate points



<u>i.e. Knowing **(u_1,v_1)** and **(u_2,v_2)** and the **camera calibrations**, we can work out **p**</u>

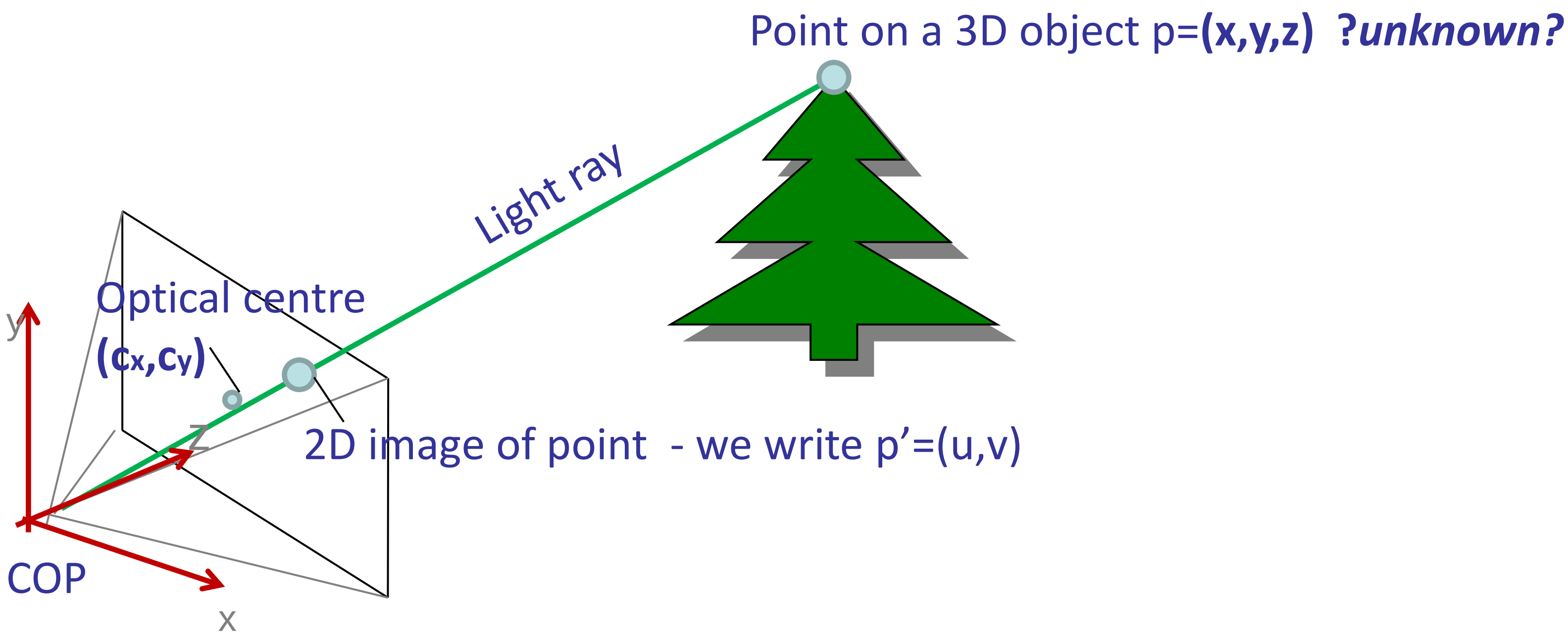
Single camera

Consider a single camera and the ray of light from object to COP

We know the 3D coordinates of COP (0,0,0).

We also know the 2D coordinates of p' = (u,v) with respect to top-left of image.

2D coordinates of p' with respect to **optical centre** are **(u – cx, v – cy)**

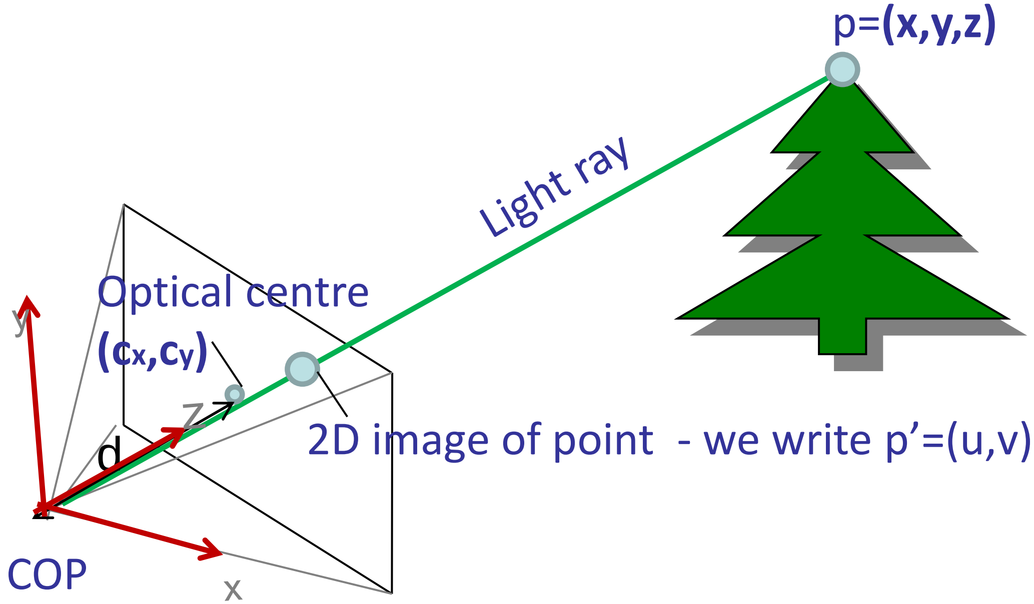


Single camera

2D coordinates of p' with respect to **optical centre** are **(u – cx, v – cy)**

3D coordinates of p' with respect to **COP** are :

$$\begin{pmatrix} (u-c_x) \\ (v-c_y) \\ d \end{pmatrix}$$



Single camera

3D coordinates of p' with respect to **COP** are :

$$\begin{pmatrix} (u-c_x) \\ (v-c_y) \\ d \end{pmatrix}$$

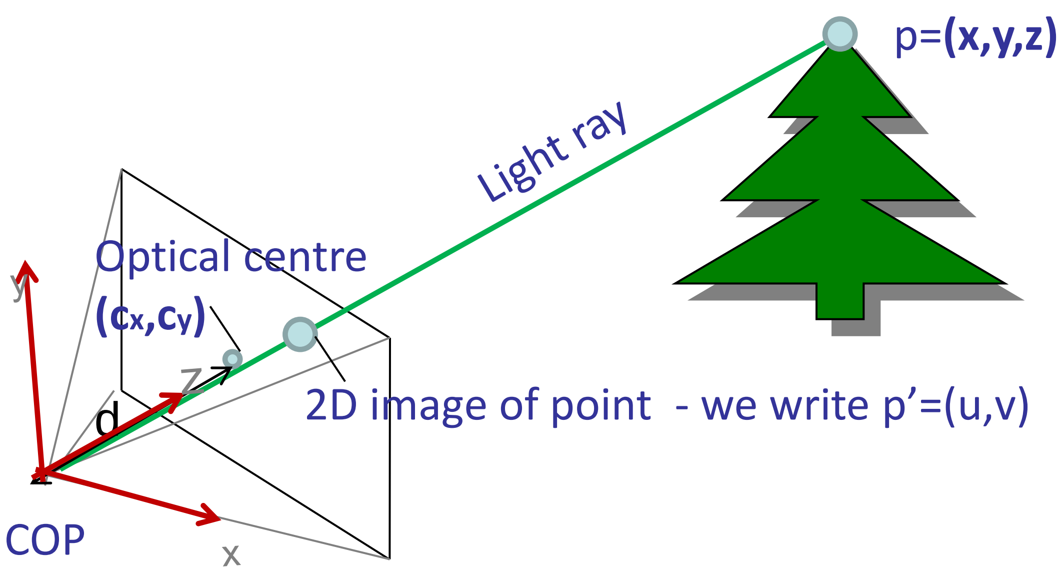
which is (0,0,0)

So the light ray starts at **COP** and travels in **direction D:** **D=**

$$D = \begin{pmatrix} (u-c_x)/d \\ (v-c_y)/d \\ 1 \end{pmatrix}$$

The parametric equation of the ray is: **p(s) = COP + s D**

i.e. Just **p(s) = s D**



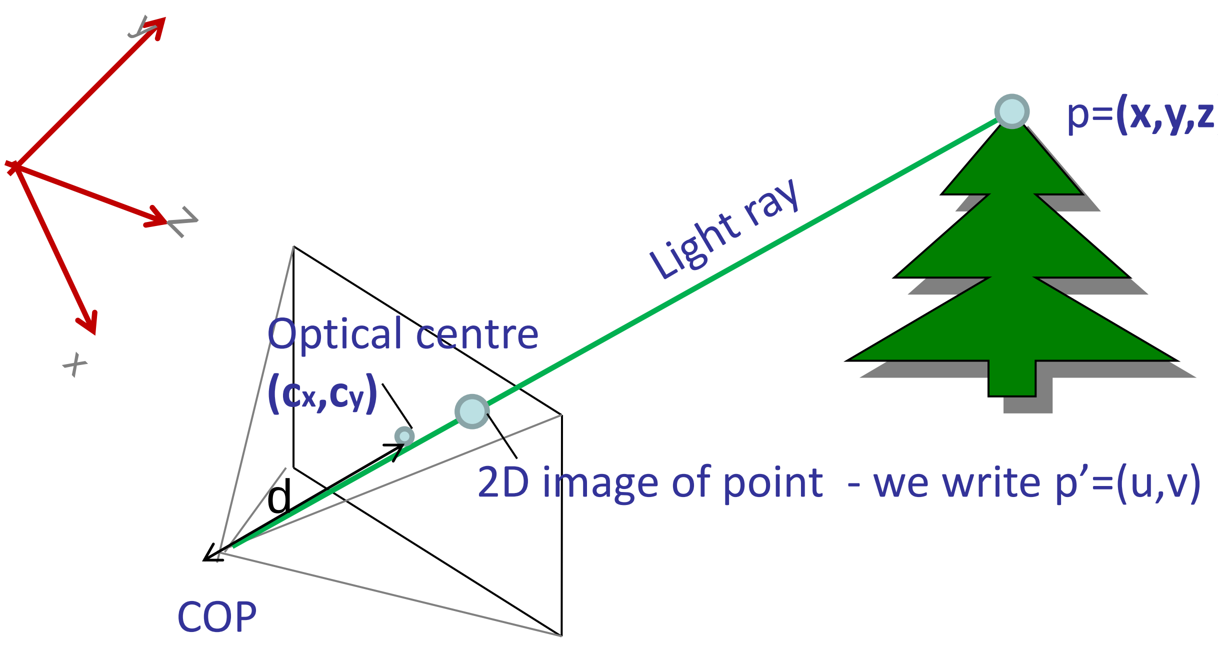
Single camera

The parametric equation of the ray is: **p(s) = COP + s D**

But remember the origin could be anywhere with respect to the camera(translated by some **T**, rotated by some **R**)

So **COP** becomes **-t** .... But also rotated by inv(R) so COP is **($-R^{-1}t$)**

D becomes **(R^{-1} D)**



Summary

Projecting from **3D point** to **2D point**

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{TR}p$$

$$\mathbf{u} = \frac{dx}{z} + c_x$$

$$\mathbf{v} = \frac{dy}{z} + c_y$$

(x, y, z) to **(u,v)**

Going from **2D point** to **3D ray**

$$\mathbf{a} = - R^{-1} t$$

$$u' = u - c_x$$
$$v' = v - c_y$$

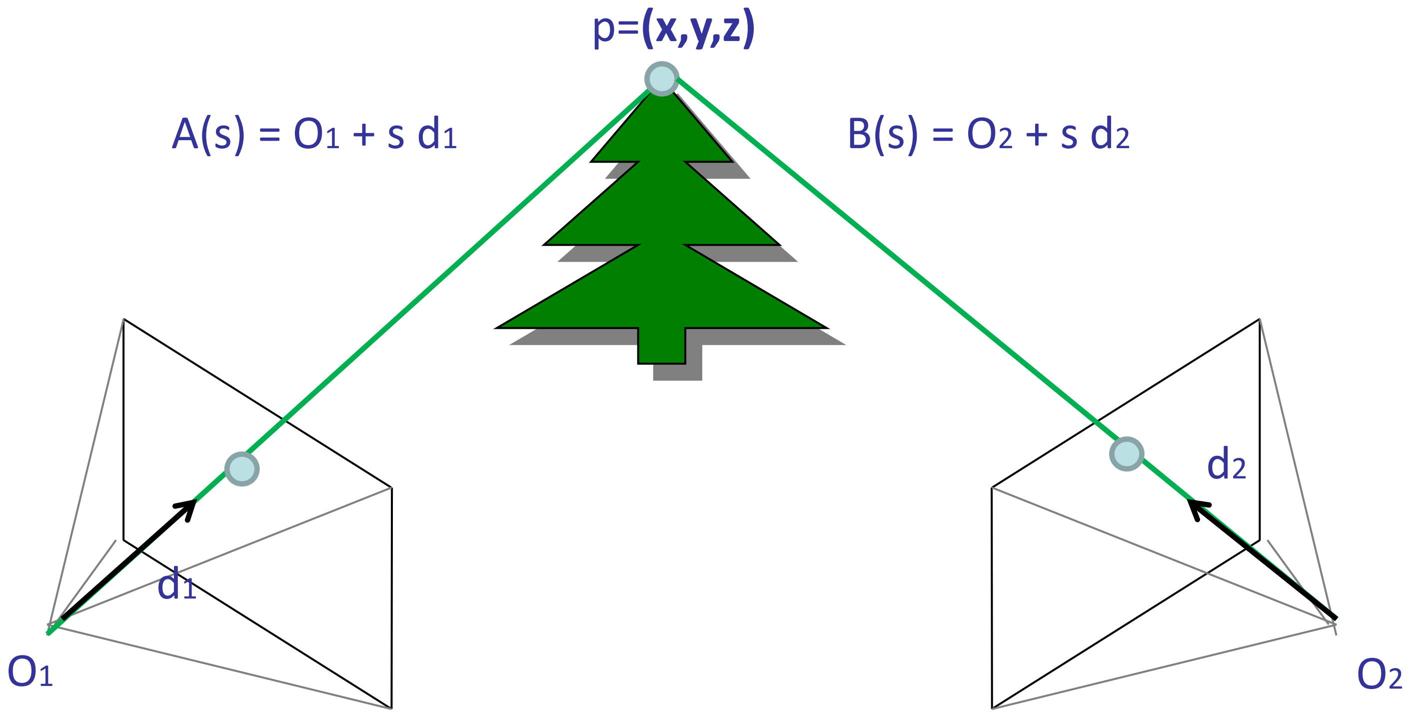
$$\mathbf{b} = R^{-1} \begin{pmatrix} \mathbf{(u')/d} \\ \mathbf{(v')/d} \\ \mathbf{1} \end{pmatrix}$$

(u,v) to **p(s) = a + sb**

Triangulation – Intersecting Rays

Given a pair of 2D points, captured with calibrated cameras, we can write down the equations of the related light rays

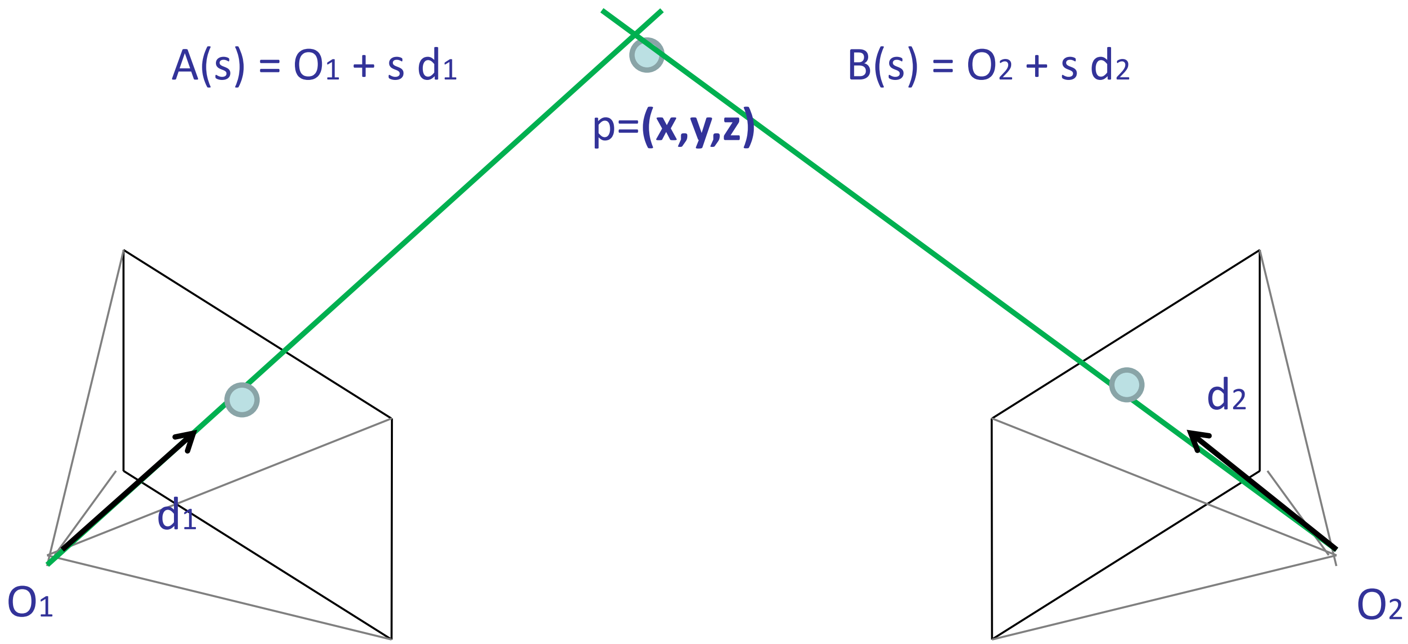
The 3D point we seek is at the intersection of the two rays



Triangulation – Intersecting Rays

In practice, errors may mean the rays don't perfectly intersect

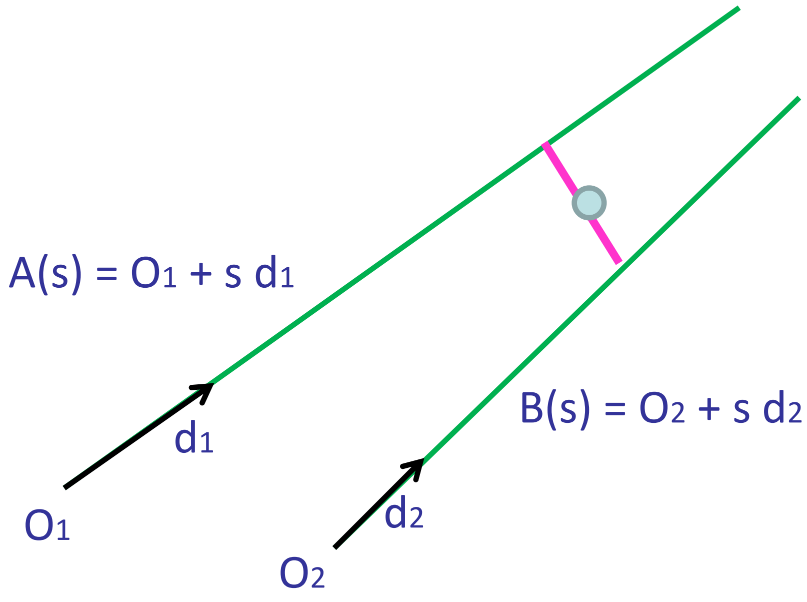
So really we are looking for the point in the space closest to both lines



Triangulation – Intersecting Rays

In practice, errors may mean the rays don't perfectly intersect

So really we are looking for the point in the space closest to both rays



The shortest distance between two infinitely long lines (rays) is the length of the line orthogonal to both rays

So the point closest to both lines is halfway along that line

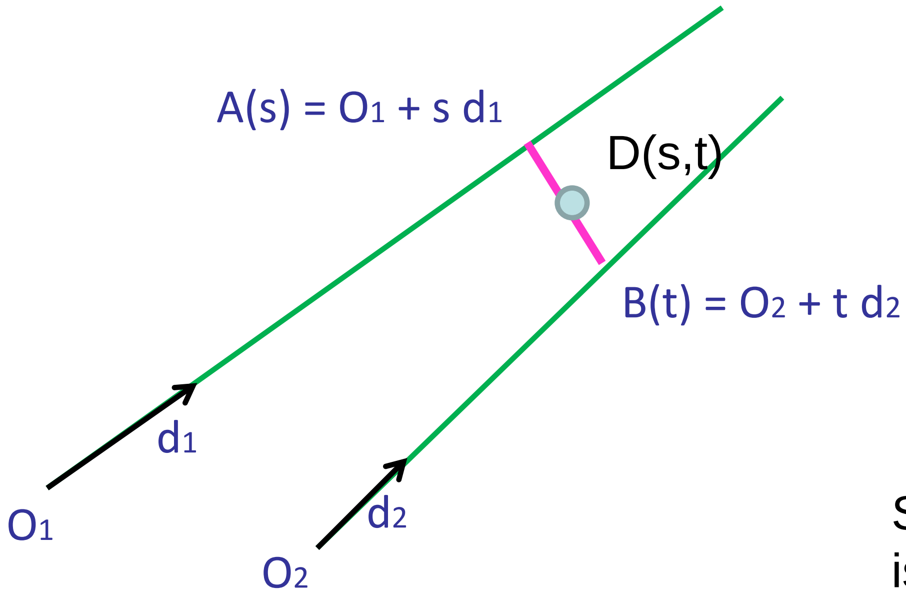
Triangulation – Intersecting Rays

Any line between **A(s)** and **B(t)** can be written **D(s,t) = B(t) – A(s)**

We are looking for the **s** and **t** that give us a line **D(s,t)** where:

$$\mathbf{d_1 \cdot D(s,t) = 0}$$

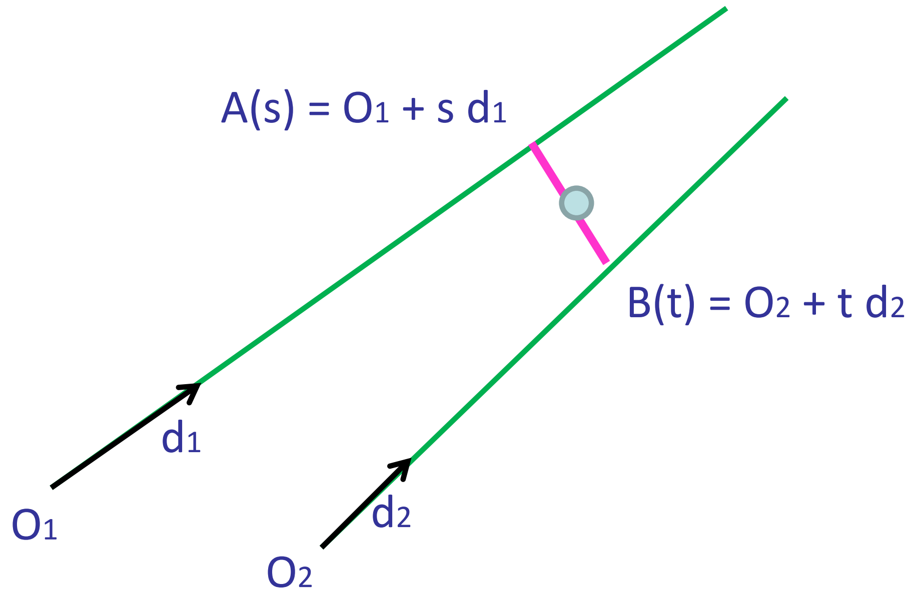
$$\mathbf{d_2 \cdot D(s,t) = 0}$$



So the point closest to both lines is halfway along that line

Triangulation – Intersecting Rays

From this simple constraint we can derive the values of **s** and **t**



$$\mathbf{d_1 \cdot (B(t) - A(s)) = 0}$$

$$\mathbf{d_2 \cdot (B(t) - A(s)) = 0}$$

$$\mathbf{d_1 \cdot (O_2 + t\, d_2 - O_1 - s\, d_1) = 0}$$

$$\mathbf{d_2 \cdot (O_2 + t\, d_2 - O_1 - s\, d_1) = 0}$$

$$\mathbf{d_1 \cdot O_2 + t\, d_1 \cdot d_2 - d_1 \cdot O_1 - s\, d_1 \cdot d_1 = 0}$$

$$\mathbf{d_2 \cdot O_2 + t\, d_2 \cdot d_2 - d_2 \cdot O_1 - s\, d_1 \cdot d_2 = 0}$$

$$\mathbf{d_1 \cdot (O_2 - O_1) = (d_1 \cdot d_1)\, s - (d_1 \cdot d_2)\, t}$$

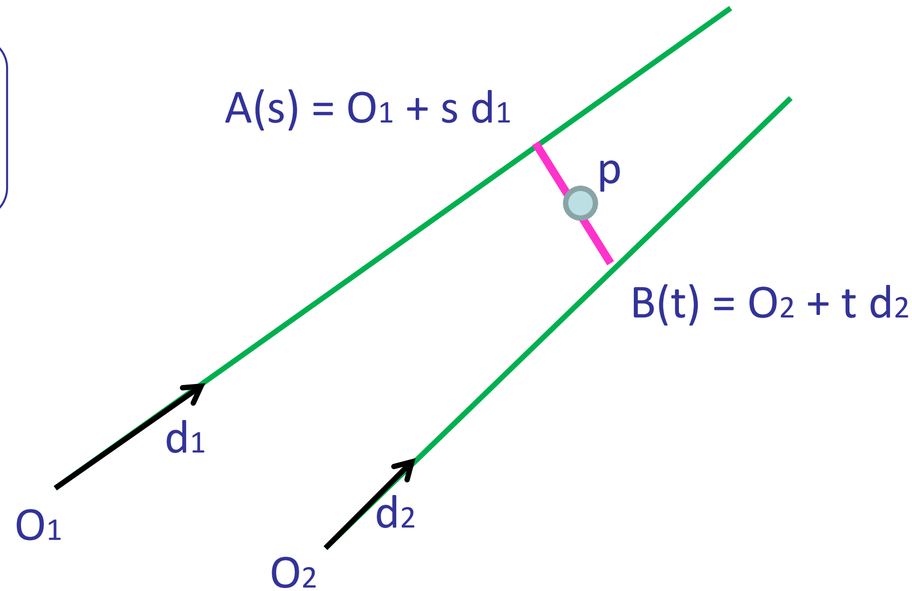
$$\mathbf{d_2 \cdot (O_2 - O_1) = (d_1 \cdot d_2)\, s - (d_2 \cdot d_2)\, t}$$

Triangulation – Intersecting Rays

Rearrange into a matrix and solve the linear system

$$\begin{pmatrix} (d_1 \cdot d_1) & -(d_1 \cdot d_2) \\ (d_1 \cdot d_2) & -(d_2 \cdot d_2) \end{pmatrix} \begin{bmatrix} s \\ t \end{bmatrix} = \begin{pmatrix} d_1 \cdot (O_2 - O_1) \\ d_2 \cdot (O_2 - O_1) \end{pmatrix}$$

$$A \quad\quad x \quad = \quad b$$



$$A\,x = b$$

$$x = A^{-1}\,b$$

Once we have **s** and **t** we can plug these back into $\left\{ \begin{array}{l} A(s) = O_1 + sd_1 \\ B(t) = O_2 + td_2 \end{array} \right.$

So the closest point $\mathbf{p} = \mathbf{A(s)} + \dfrac{\mathbf{B(t)-A(s)}}{\mathbf{2}}$

Summary

After attending this lecture you should be able to:

- Describe how matrices can be combined with homogeneous coordinates to perform rigid body transformations in 2D and 3D
- Combine matrices in 2D and 3D to create compound transformations
- Describe the pin-hole perspective projection and orthographic projection models
- Show how perspective projection can be written as a matrix in the framework of homogeneous coordinates
- Perform 3D simple reconstruction using a visual hull
- Describe how a pair of 2D points may be triangulated to recover a 3D point under a calibrated camera setup