Visual Computing I:

Interactive Computer Graphics and Vision

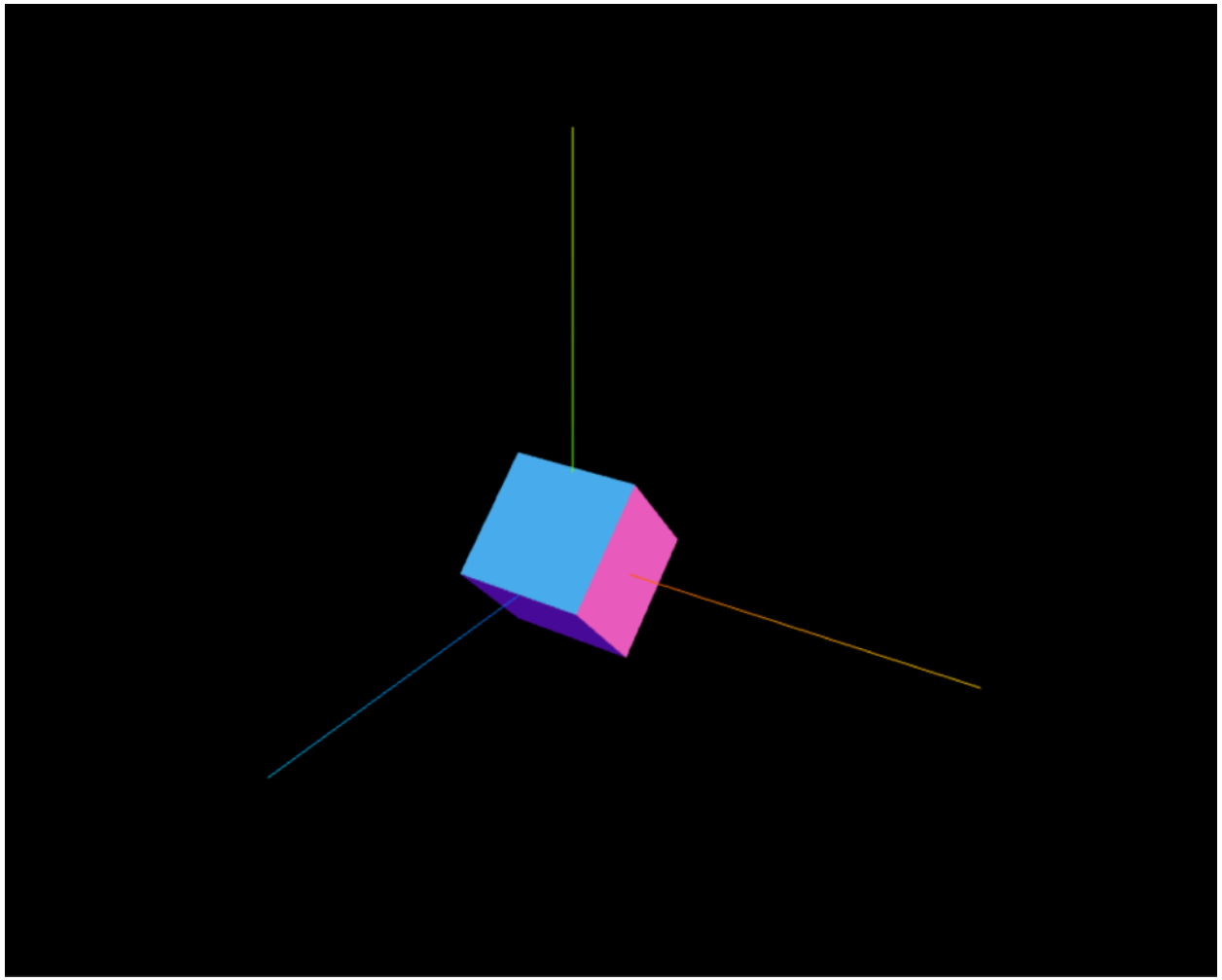
Camera Calibration and Stereo Vision

Stefanie Zollmann and Tobias Langlotz

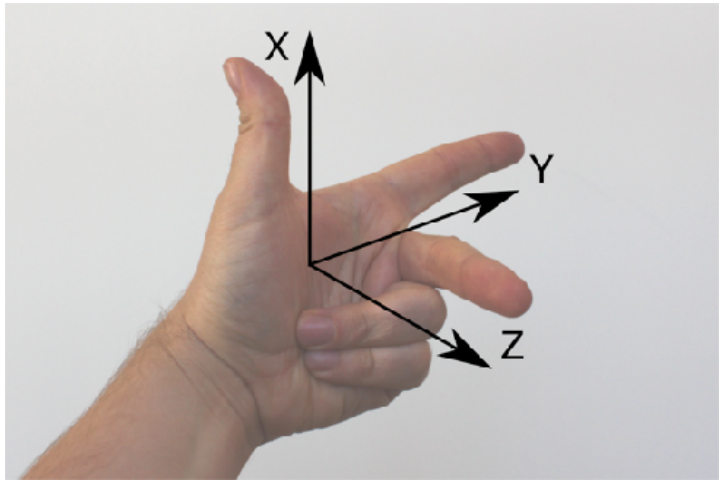
Last time..

Three Dimensions

- We moved from 2D to 3D
- Many ideas the same:
  - Homogeneous co-ordinates
  - Scaling and translation
- Some things get tricky
  - Choice of left- or right-handed coordinates
  - Rotations get complicated
- Projection from 3D to 2D



Left & Right-Handed Coordinates

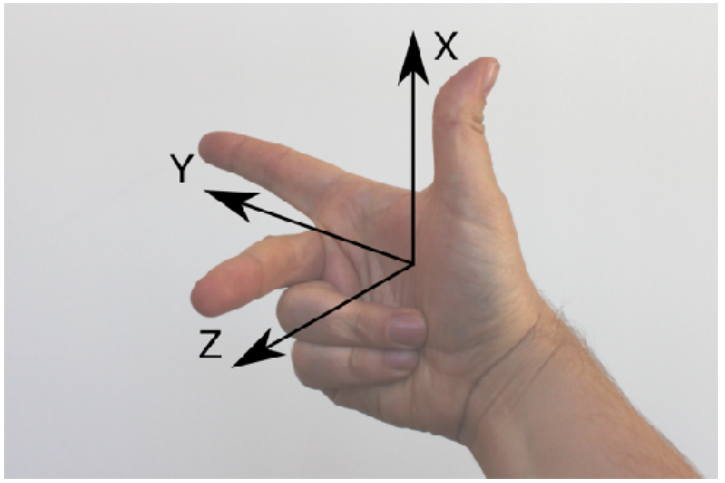


Left handed

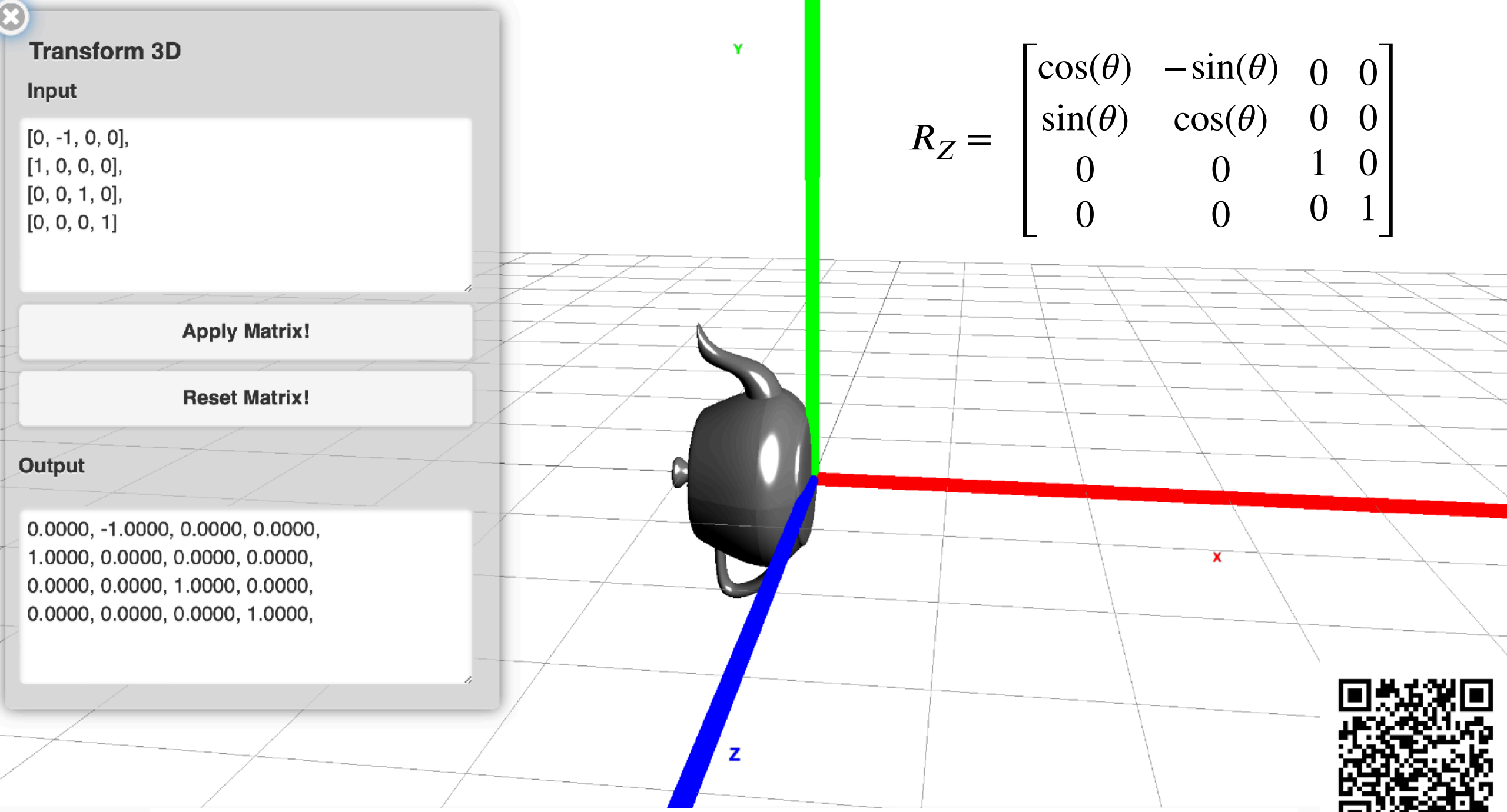
Thumb is axis

Forefinger is axis

Middle finger is axis



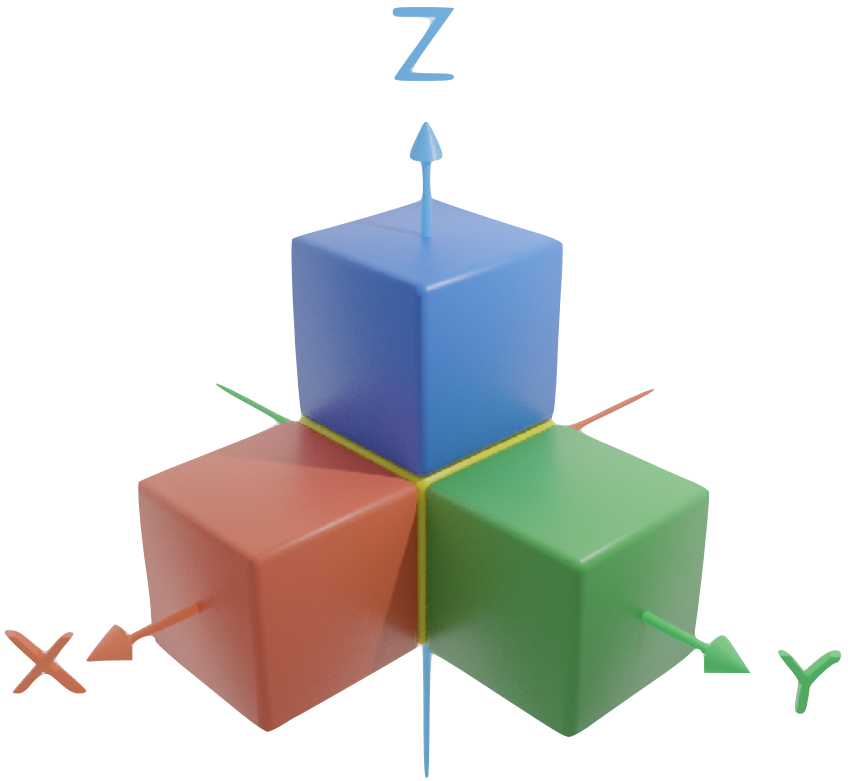
Right-Handed

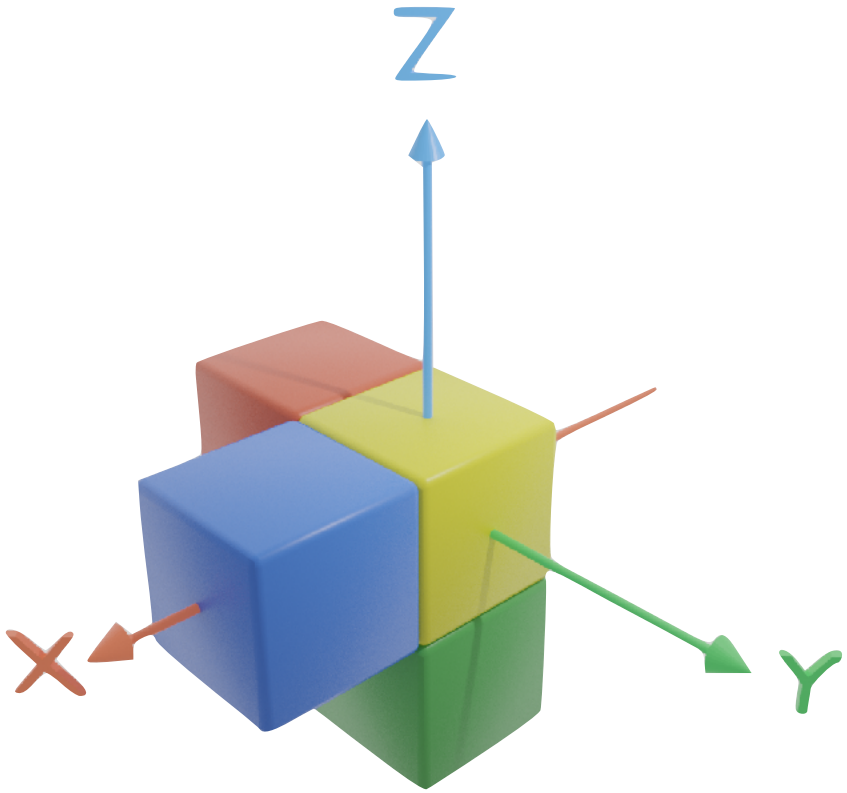


$$R_Z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

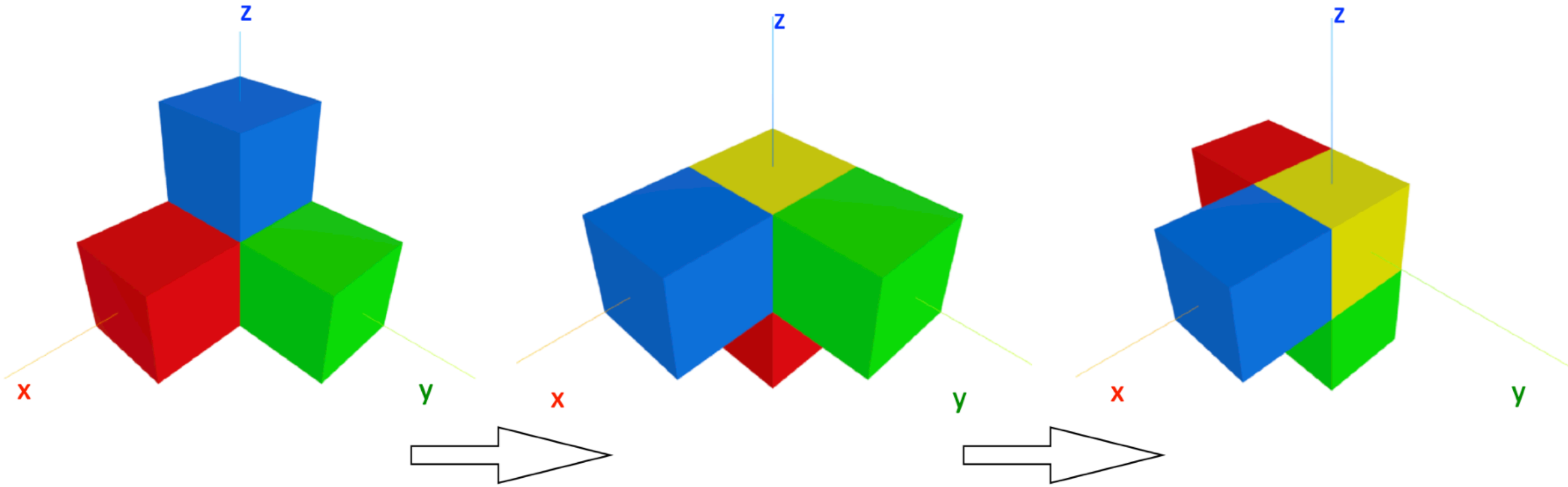
https://szollmann.github.io/LectureExamples/tutorials/transformations/Transformations3DTeapot/

Quiz: Rotation in 3D





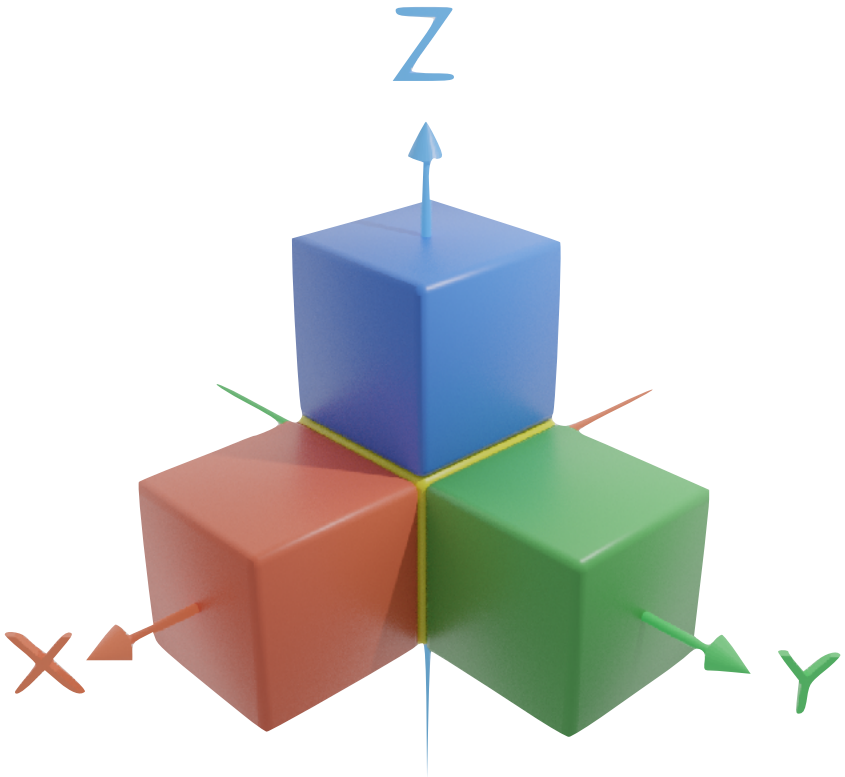
Answer

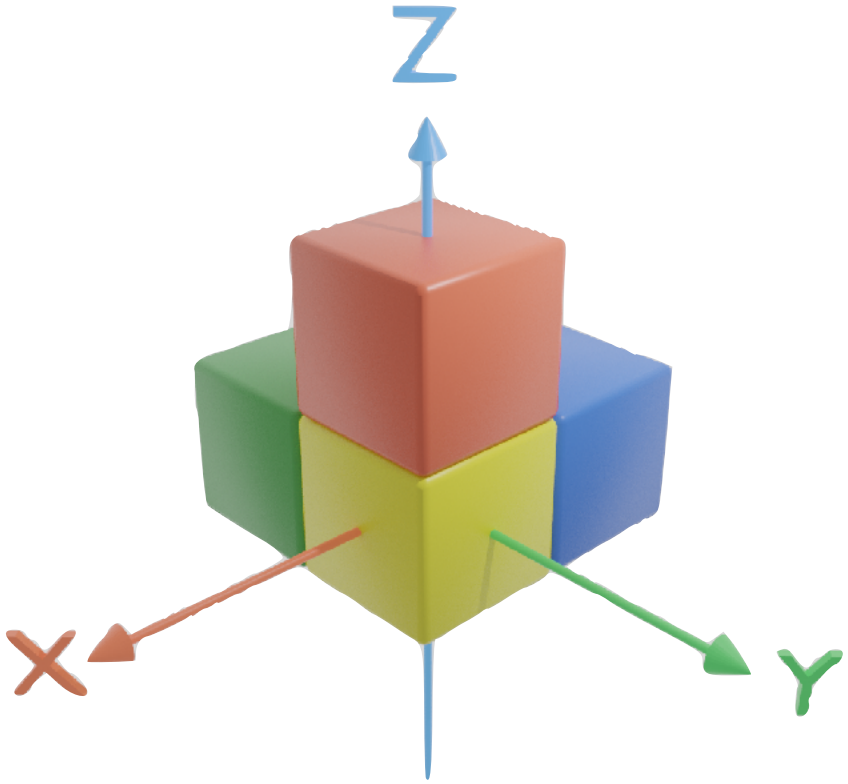


1. 90 Degree around y- Axis

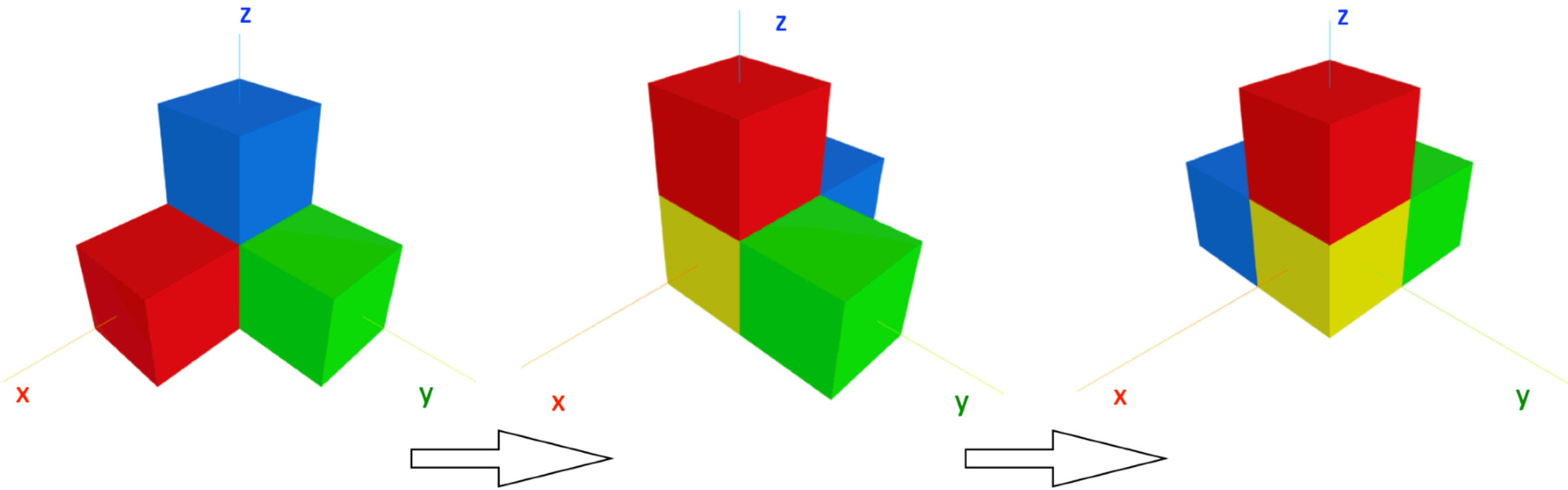
2. 270 Degree around y-Axis

Quiz: Rotation in 3D





Answer: Not Possible



1. 270 Degree around y- Axis

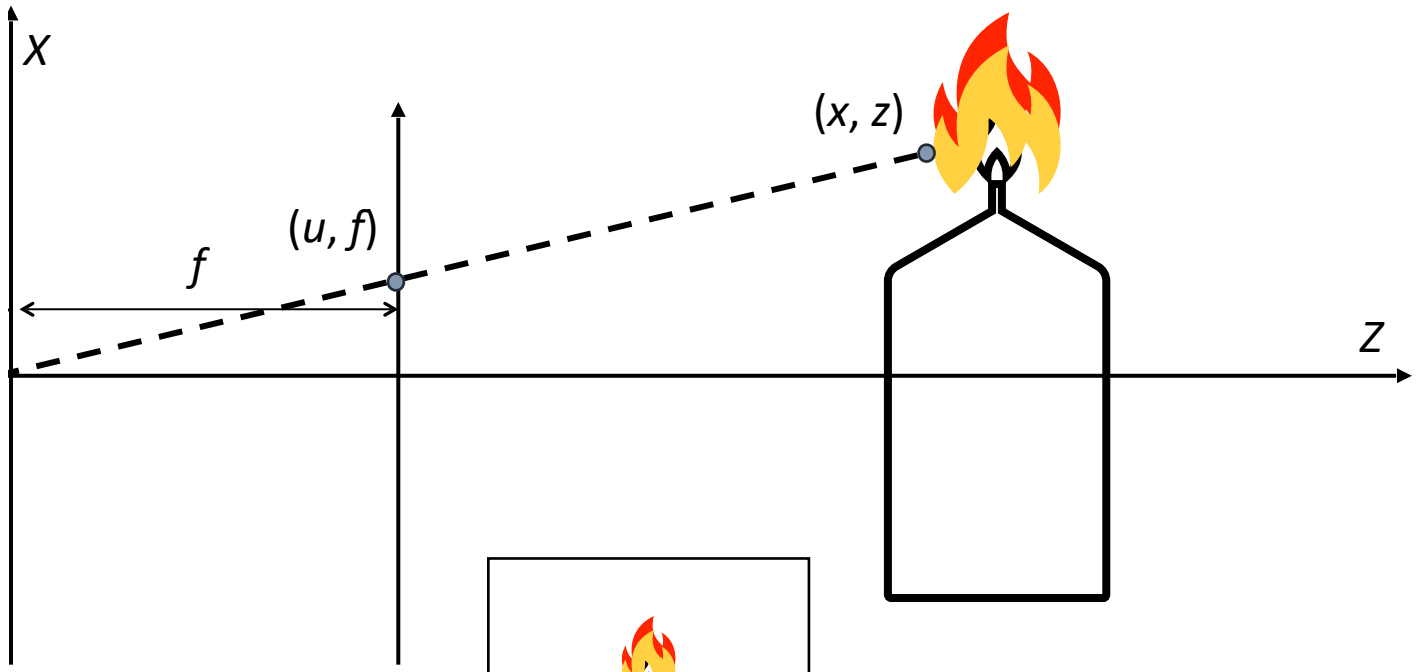
2. 90 Degree around z-Axis

Flip needed

The Pinhole Camera Model and Projective Geometry

- Removing the sign change

- We can put the image plane in front of the pinhole
  - Removes the sign change
  - Not practical for real cameras
  - The maths works out just fine

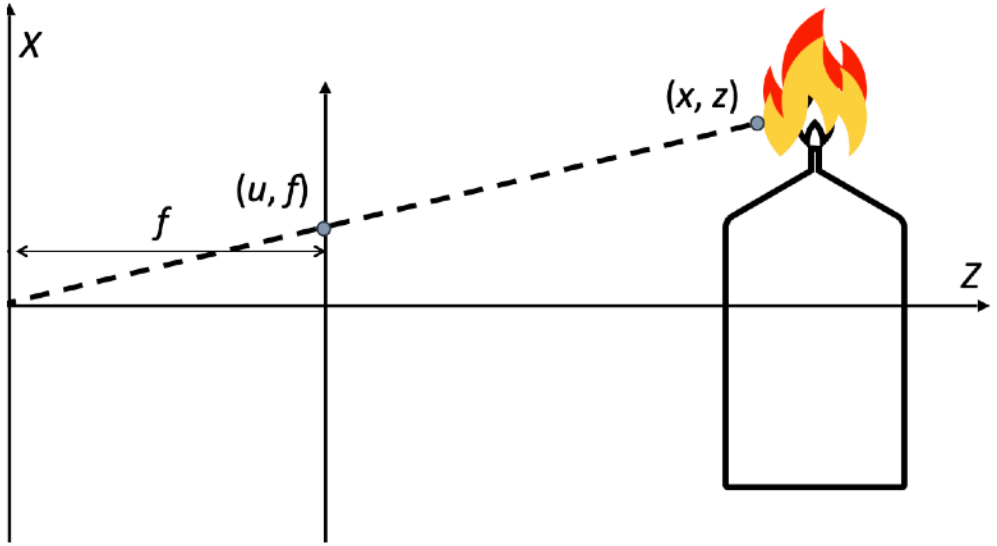


$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Camera Parameters and Calibration

Camera Parameters - Intrinsics and Extrinsics

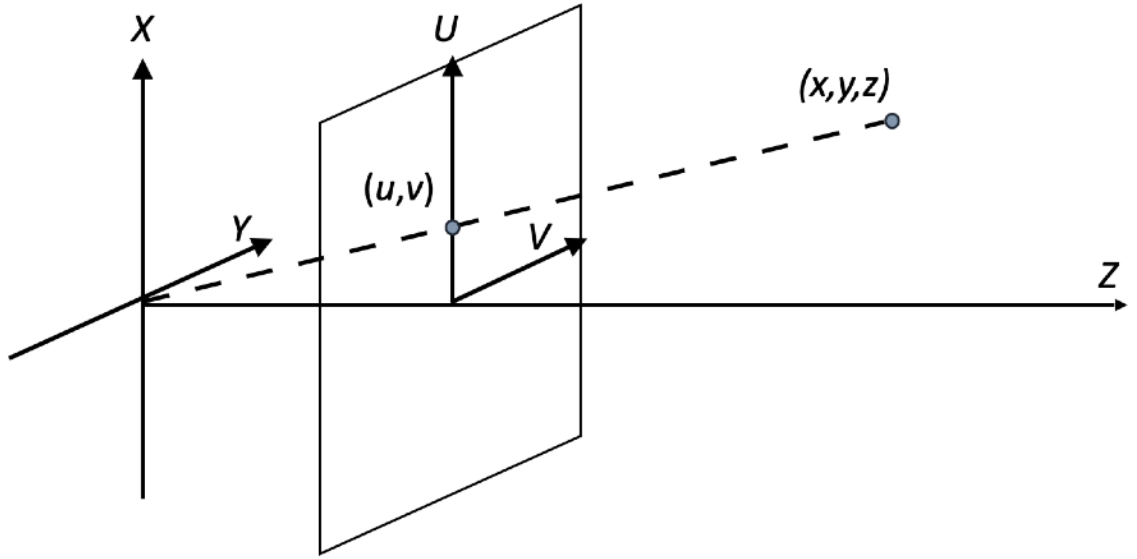
- Two main types of parameters:
  - **Intrinsic parameters** relate the camera's coordinate system to idealized coordinate system described earlier
  - **Extrinsic parameters** relate the camera's coordinate system to a world coordinate system
- Let's ignore lens issues for a moment
- Estimating intrinsic and extrinsic camera parameters is known as geometric camera calibration



Camera Parameters - Intrinsics and Extrinsics

- The pinhole camera model
  - Projects from 2D to 3D
  - Models perspective projection well
  - Basic model:

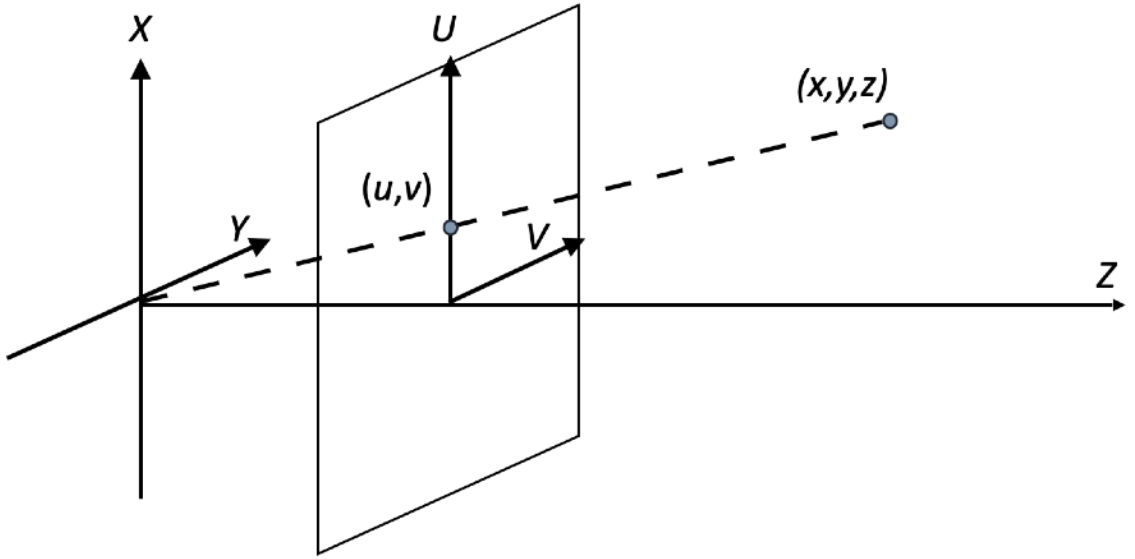
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Camera Parameters - Intrinsics and Extrinsics

- Often break this down into

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Extrinsic}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



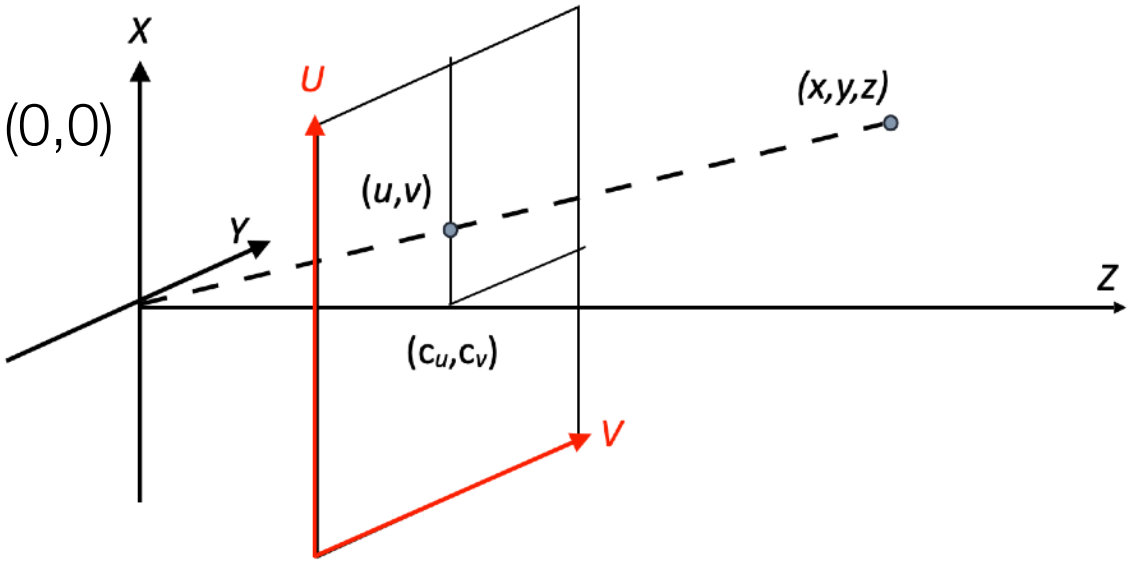
- Most simple case: $\mathbf{u} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{x}$

- $\mathbf{K}$: camera calibration or Intrinsic parameters

- $[\mathbf{I} \mid \mathbf{0}]$: camera pose or Extrinsic parameters

Camera Parameters - Intrinsics and Extrinsics

- Model has image origin in the centre
  - Remember in CV:
    - We usually put this at the top left corner (0,0)
  - Can fix this with a translation



- If the centre is at $(c_u,\ c_v)$ we get:

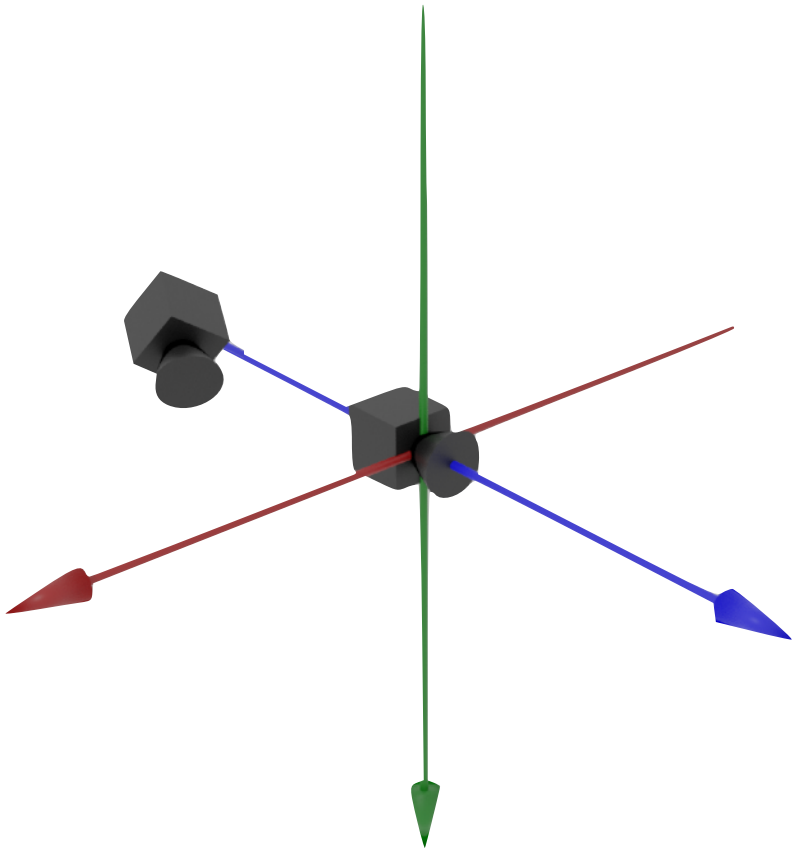
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \underbrace{\begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Extrinsic}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- $(c_u,\ c_v)$ is called principal point

- True geometric center of the image (For perfectly manufactured 1000x800 pixel sensor (500, 400) - Reality?)

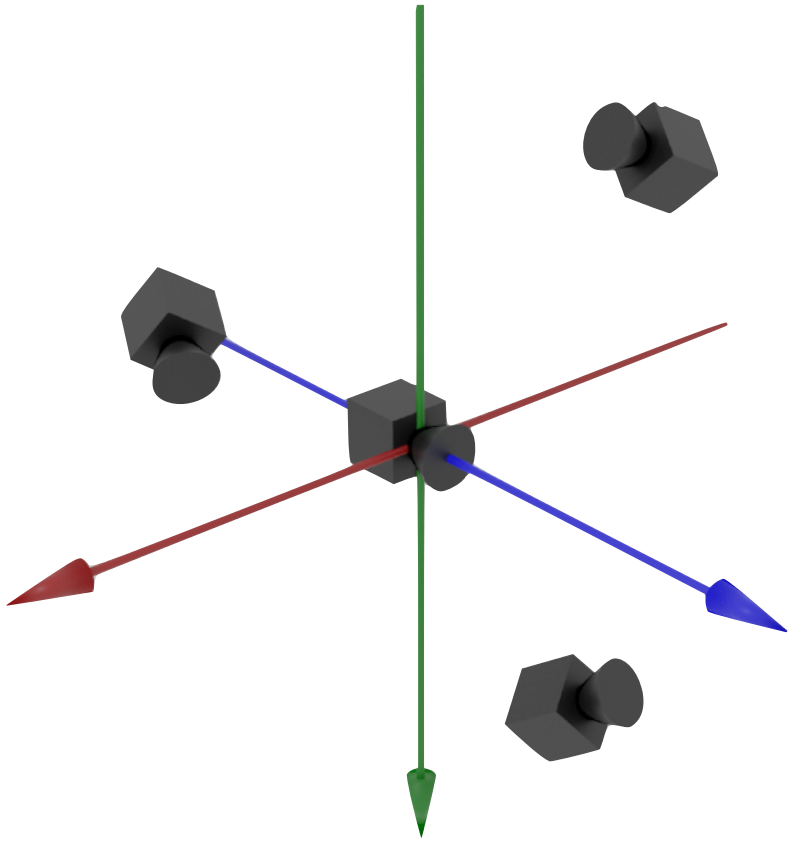
Camera Parameters - Intrinsics and Extrinsics

- We have assumed
  - A camera at the origin
  - Pointing along the positive Z axis
- Let's consider the case where camera coordinate system C does not match the world coordinate system W
- So far: Most simple case: $\mathbf{u} = K \left[I \mid \mathbf{0} \right] \mathbf{x}$
- We will need the general case: $\mathbf{u} = K \left[R \mid \mathbf{t} \right] \mathbf{x}$
  - Move the camera to any location (Translation $\mathbf{t}$)
  - Point the camera in any direction (Rotation $\mathbf{R}$)
- The **extrinsic parameters** consist of a 3x3 rotation matrix R and a 3-dimensional translation vector t



Camera Parameters - Intrinsics and Extrinsics

- To transform a camera by $\mathbf{T}$
  - Apply inverse, $\mathbf{T}^{-1}$, to points
  - To move the camera left 3 units, move the world right 3 units
  - To rotate the camera 45° about Z, rotate the world -45° about Z
  - The relative motion of the camera and the world is the same



Camera Parameters - Intrinsics and Extrinsics

- Want to rotate a camera by $\mathbf{R}$, then shift by $\mathbf{t}$
  - Equivalently shift the points by $-\mathbf{t}$, then rotate them by $\mathbf{R}^{-1} = \mathbf{R}^T$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{21} & r_{31} & 0 \\ r_{12} & r_{22} & r_{32} & 0 \\ r_{13} & r_{23} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  - Or, in more compact form:

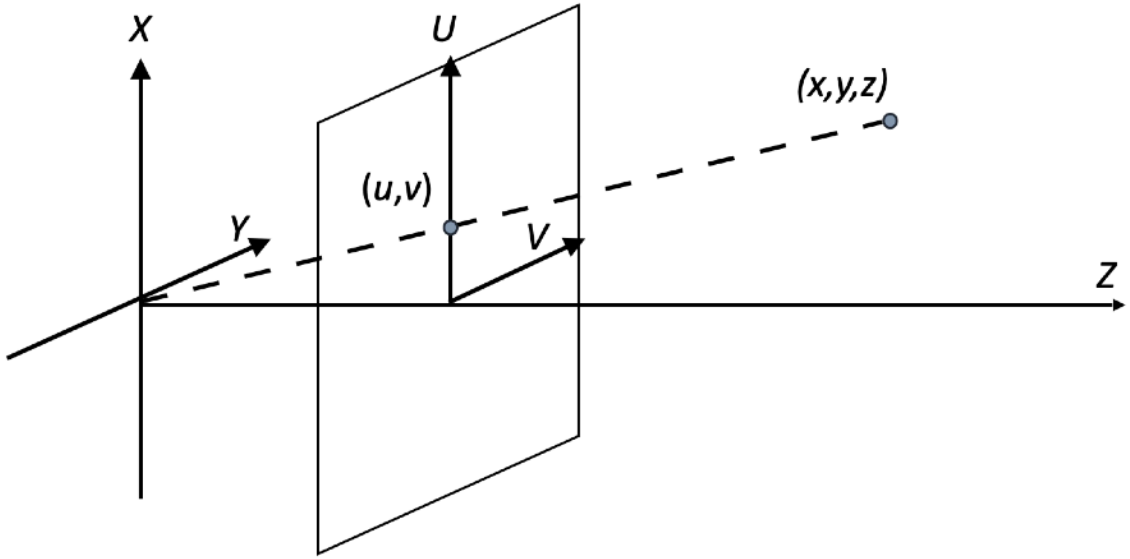
$$\mathbf{u} \equiv \mathrm{K}[\mathrm{I} \quad \mathbf{0}] \begin{bmatrix} \mathrm{R}^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathrm{I} & -\boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} = \mathrm{K} \begin{bmatrix} \mathrm{R}^T & -\mathrm{R}^{\mathrm{T}}\mathbf{t} \end{bmatrix} \mathbf{x}$$

Practical Camera Calibration

Pinhole Camera Review

$$\mathbf{u} \equiv \mathrm{K}[\mathrm{R} \quad \mathbf{t}]\mathbf{x}$$

- $\mathbf{u} = [u \quad v \quad 1]^T$ is a 2D image point
- K is a 3×3 calibration matrix
- R is a 3×3 rotation matrix
- $\mathbf{t}$ is a 3D translation vector
- $\mathbf{x} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ is a 3D point



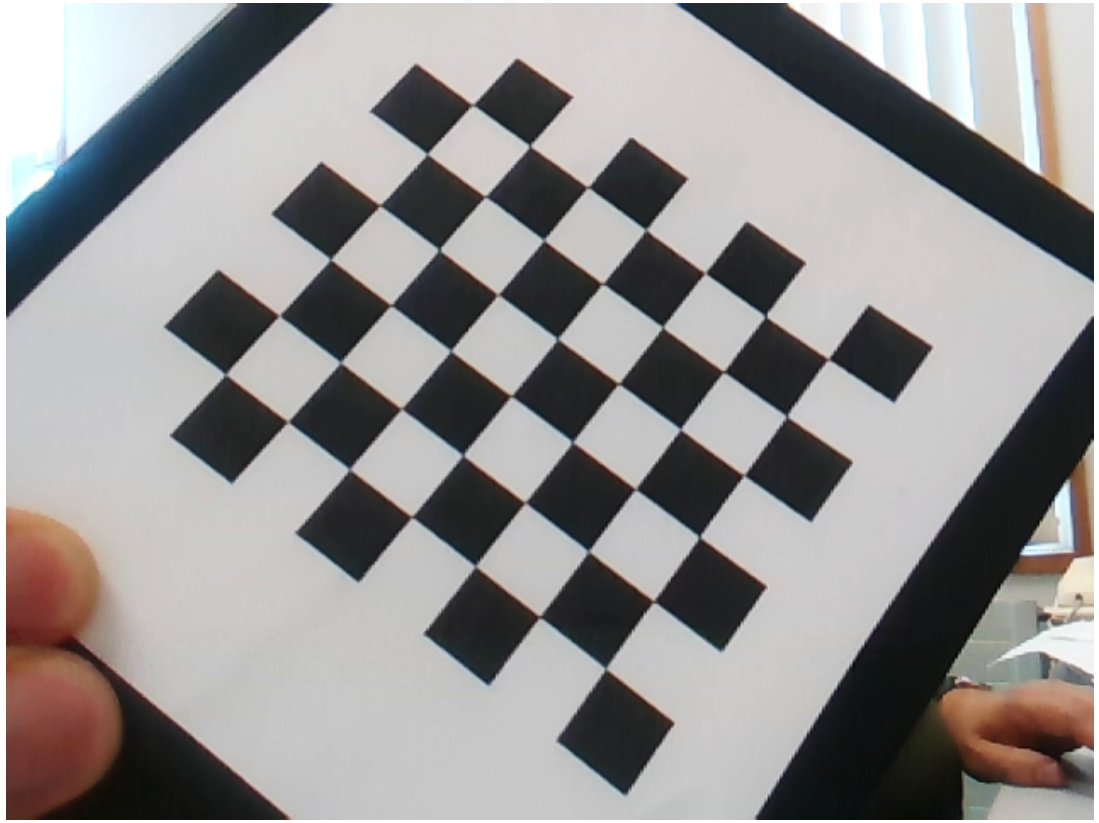
- Camera calibration: determine K

Calibration in OpenCV

```
double cv::calibrateCamera(
    // Input parameters
    //std::vector<std::vector<cv::Point3f>> objectPoints,
    //std::vector<std::vector<cv::Point2f>> imagePoints,
    //cv::Size imageSize,
    // Output parameters
    //cv::Mat cameraMatrix,
    //std::vector<double> distCoeffs,
    //std::vector<cv::Mat> rotationVectors,
    //std::vector<cv::Mat> translationVectors);
```

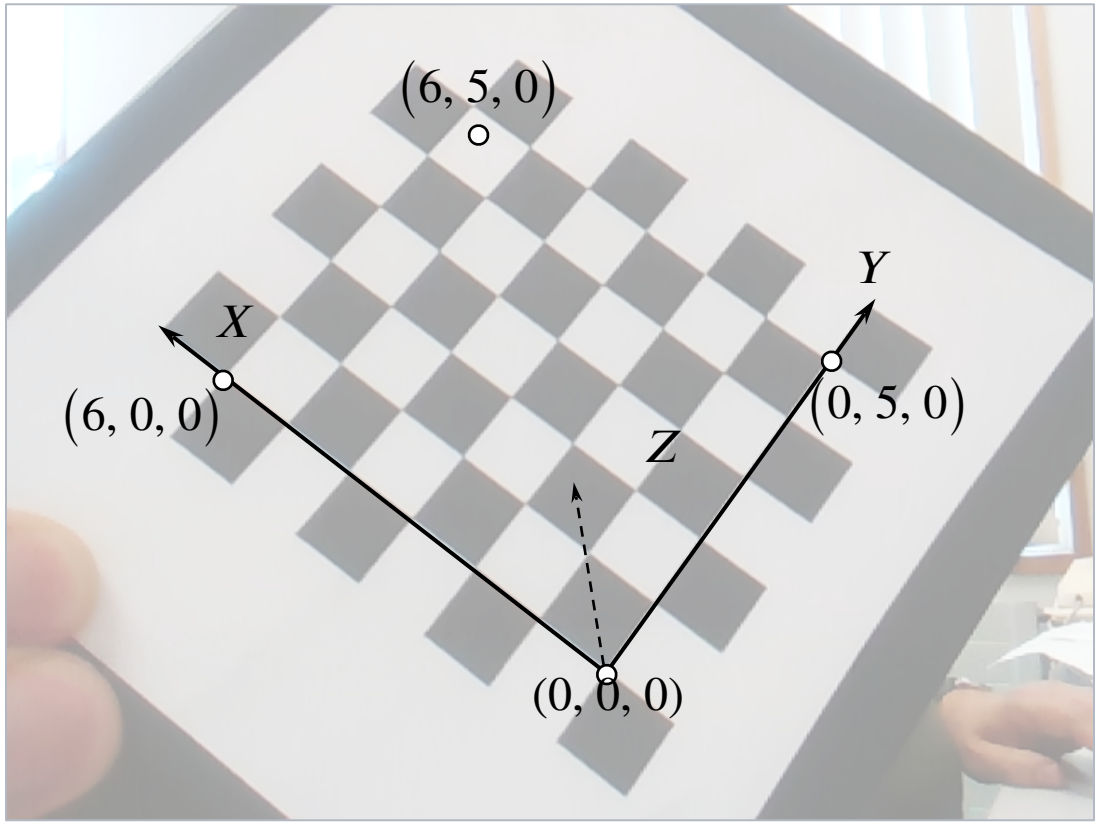
Calibration in OpenCV

- Calibration targets:
  - Input is 3D-2D matches
  - Want easy-to find 2D points
  - Need known 3D co-ordinates
- Planar targets common
  - Easy to make with a printer
  - Chess/Checkerboards
  - Grids of dots or lines
- Is a 2D pattern enough?



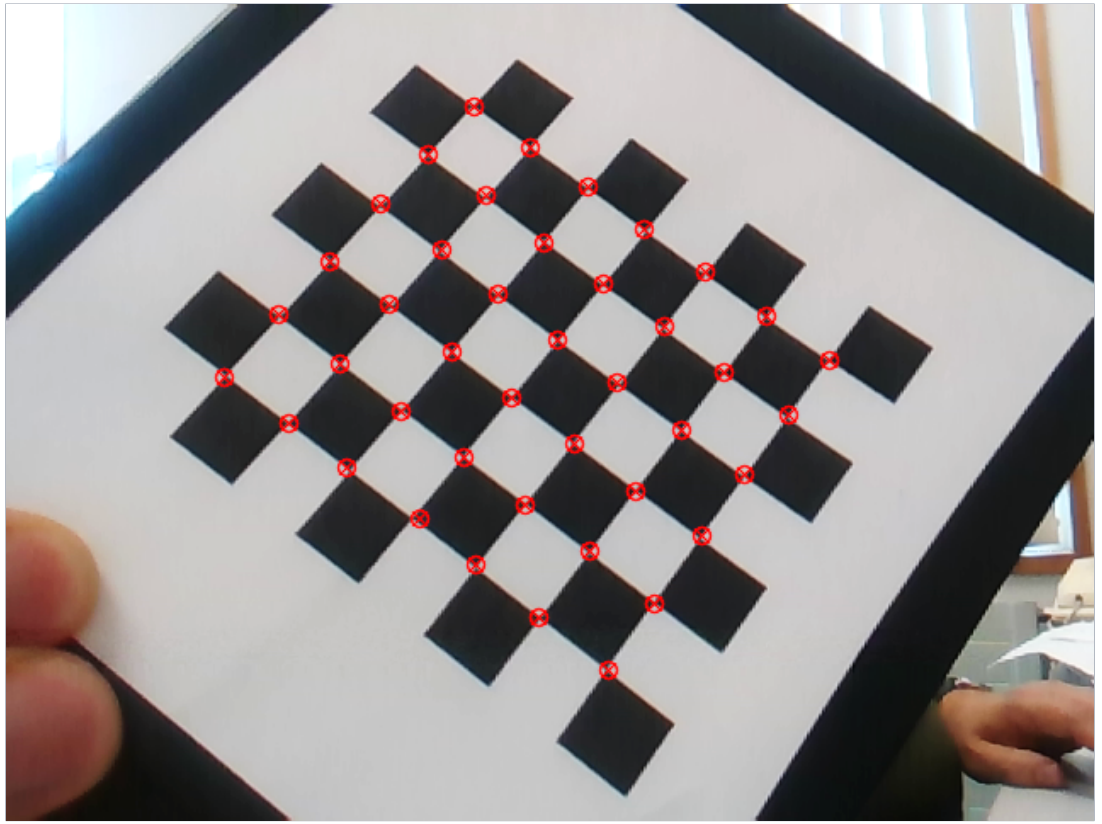
Calibration in OpenCV

- 3D Point Locations
- Choice of co-ordinates
  - X-Y plane is the target
  - Origin at one internal corner
  - Z runs into the target plane
- Need to decide on units
  - Best to use real-world units
  - Here 1 square = 1cm
  - Can use 1 square = 1 unit



Calibration in OpenCV

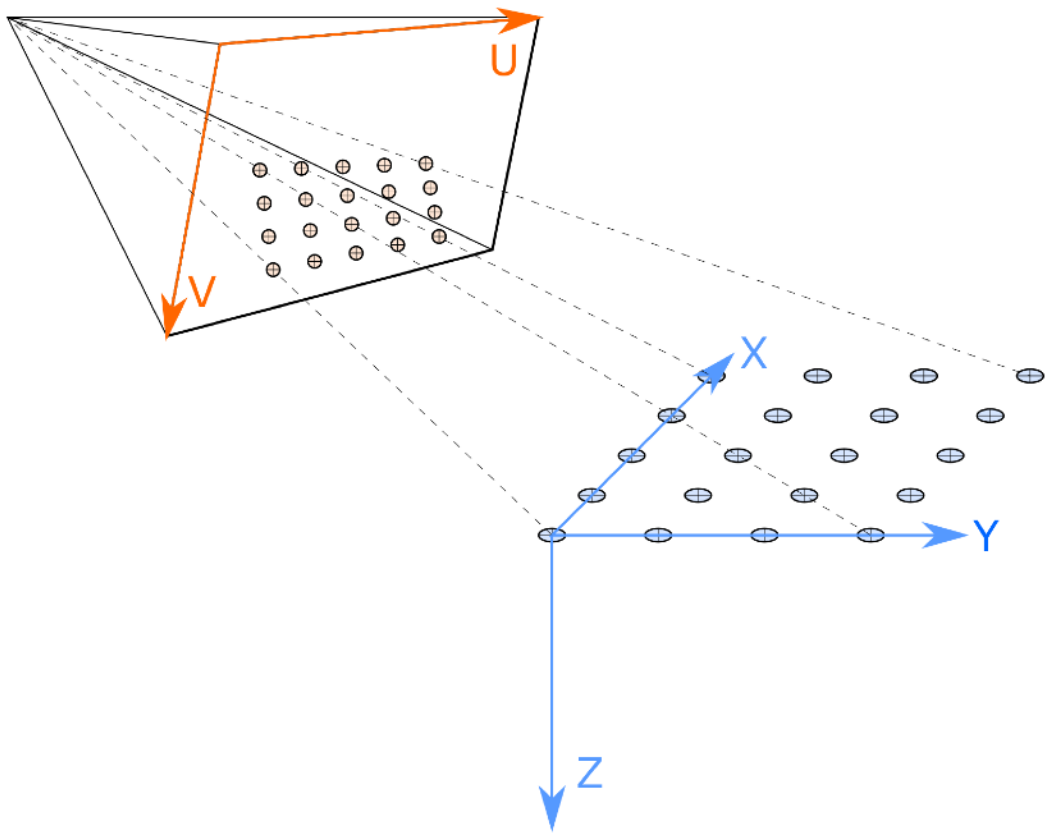
- Finding Checkerboard Corners
- OpenCV method (in brief)
  - Threshold image to black and white
  - Look for quads
  - Link quads → checkerboard
  - Sub-pixel refinement



Camera Calibration Problem

- Have k images, $i = 1 \ldots k$
  - Each has a set of 2D points
    $$\mathbf{u}_{i,j} = \begin{bmatrix} u_{i,j} & v_{i,j} & 1 \end{bmatrix}^T$$
  - These match 3D points
    $$\mathbf{x}_{i,j} = \begin{bmatrix} x_{i,j} & y_{i,j} & z_{i,j} & 1 \end{bmatrix}^{\mathrm{T}}$$
  - Related by
    $$\mathbf{u}_{i,j} \equiv \mathrm{K} \begin{bmatrix} \mathrm{R}_i & \mathbf{t}_i \end{bmatrix} \mathbf{x}_{ij}$$

- Find K, get $\mathbf{R}_i$s and $\mathbf{t}_i$s too



Modelling Lens Distortion

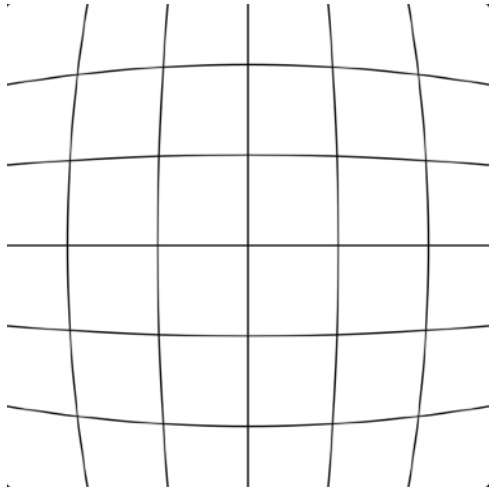
- Lens Distortion: Real lenses are imperfect and cause lens distortion, making straight lines in the world appear curved in the image

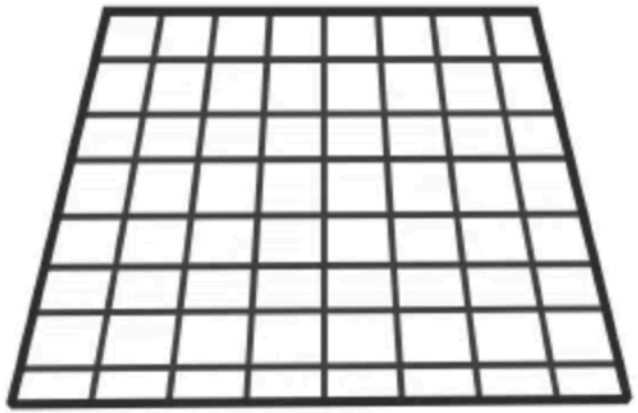
- Two main types:

  - Radial distortion (barrel or pincushion effect, where lines curve towards/away from the image center)

  - Tangential distortion (when the lens is not perfectly parallel to the sensor).

- The function calculates a set of distortion coefficients (k_1, k_2, p_1, p_2, k_3...) that mathematically describe how to correct for these curves



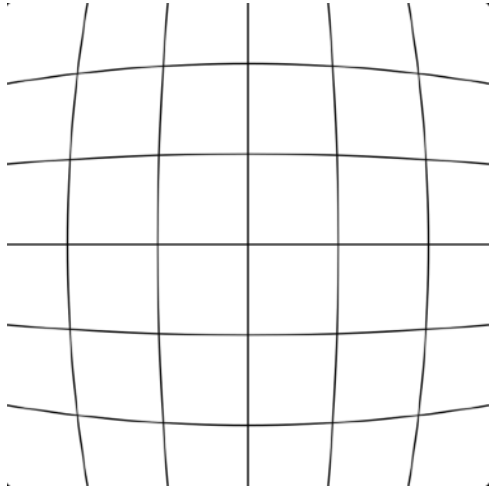


Modelling Lens Distortion

- Barrel/pincushion distortion

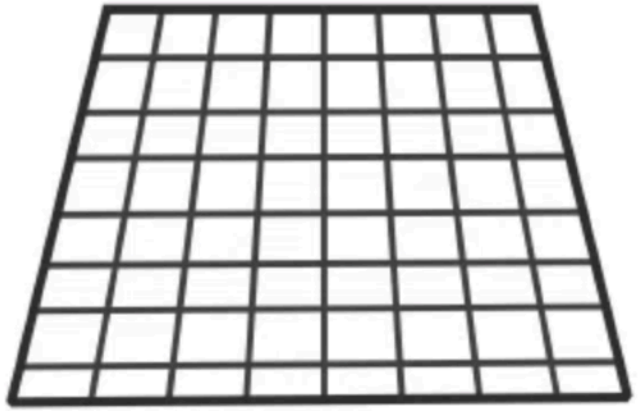
  $\mathbf{u}' = \mathbf{u}(1 + k_1 r^2 + k_2 r^4 + \ldots)$

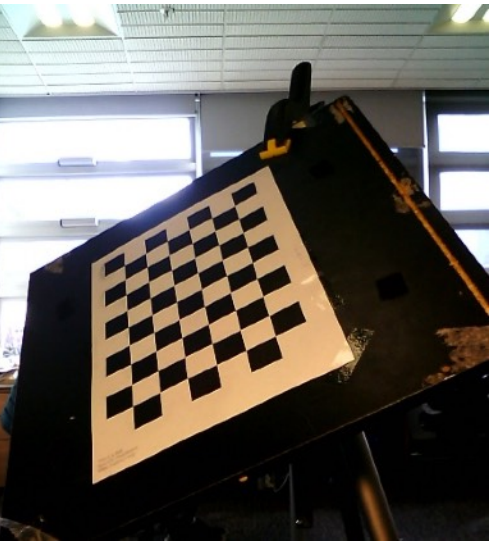
  - (u, v) measured from image centre
  - $r = \sqrt{u^2 + v^2}$: distance from centre

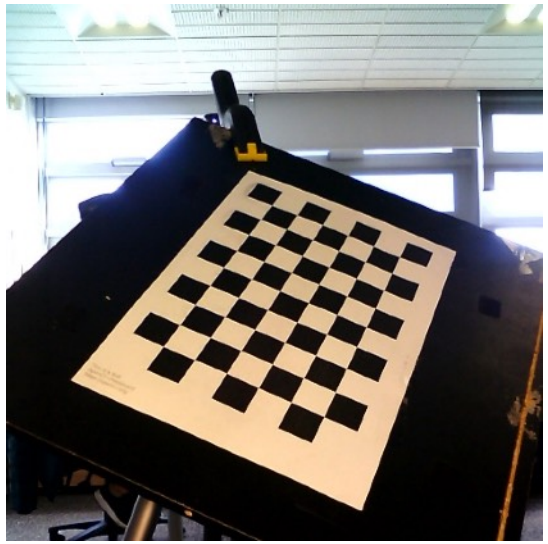


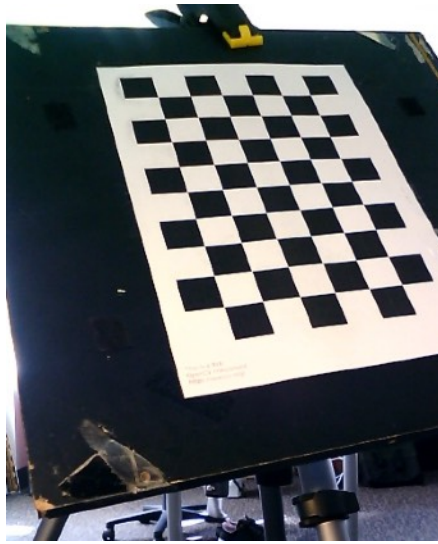
- Tangential distortion
  - Lens not parallel to image plane

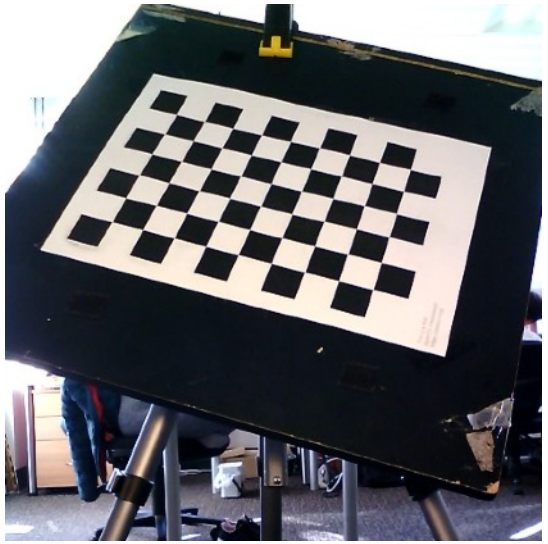
$$u' = u + \left(2p_1 uv + p_2\left(r^2 + 2u^2\right)\right)$$
$$v' = v + \left(2p_2 uv + p_1\left(r^2 + 2u^2\right)\right)$$

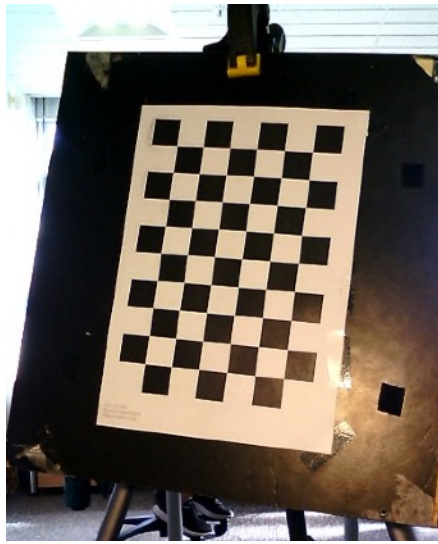










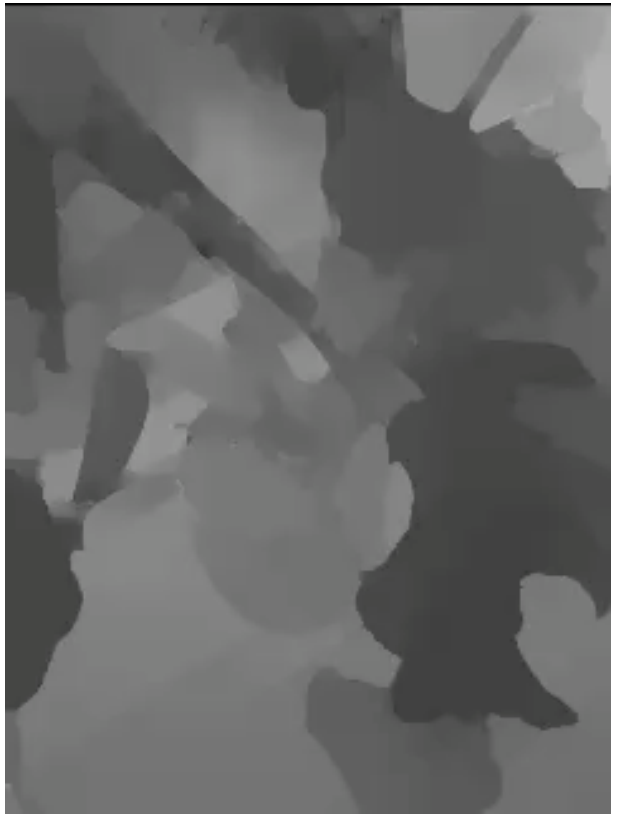
The Geometry of Multiple Views and Scene Reconstruction

The Geometry of Multiple Views and Scene Reconstruction

- Scene Reconstruction
- Epipolar Geometry
    - Epipolar Constraint
    - Baseline, Epipolar Plane, Epipoles, Epipolar Lines
- The Uncalibrated Case
    - Fundamental Matrix
    - Eight-Point Algorithm
    - Seven-Point Algorithm
- The Calibrated Case
    - Essential Matrix
- More cameras -> Multi-View Geometry

Scene Reconstruction

- So far, we have estimated the parameters of a single camera
- The depth of a scene point is not directly accessible in a single image
- What can we do?
    - With at least two pictures, depth can be measured through triangulation
    - Parameters of all cameras have to be known
    - Search for point correspondence!

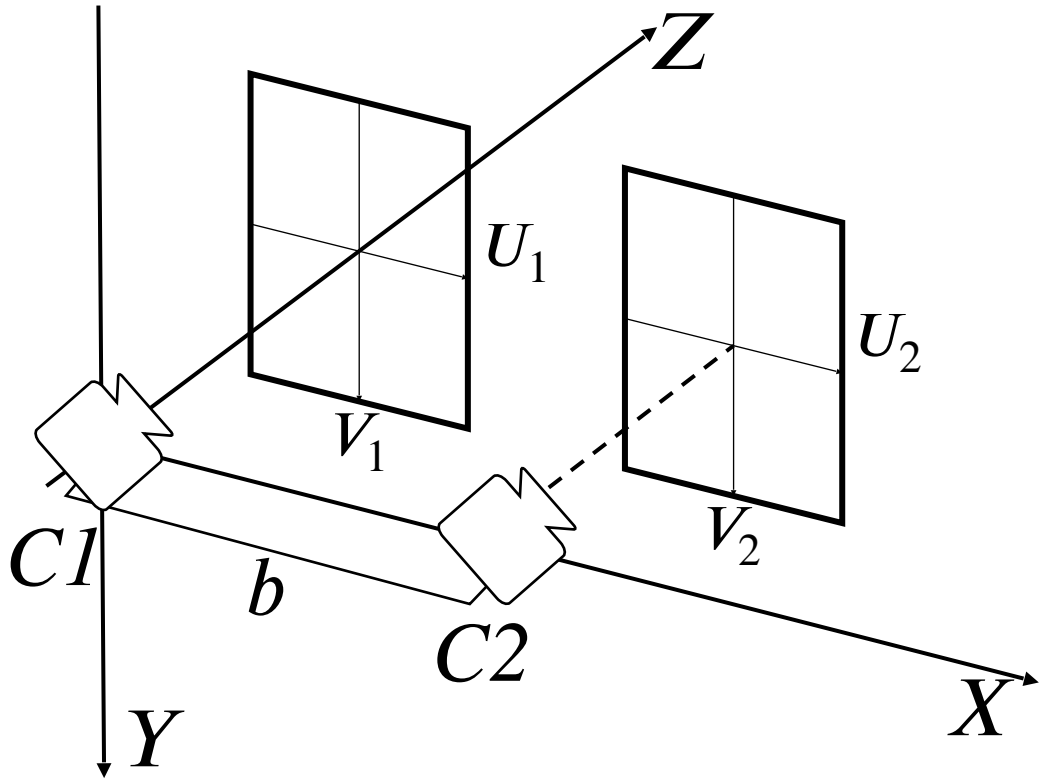


Stereo Computation

A Simple Stereo System

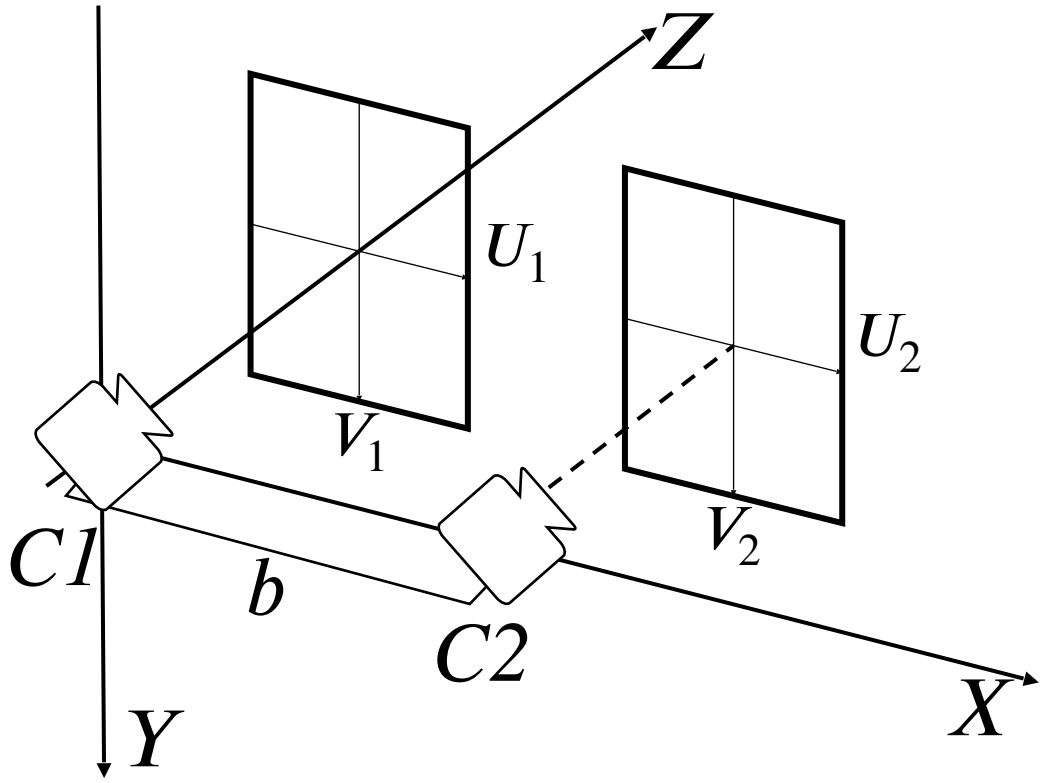
- Assume a set of 2 pinhole cameras
  $\mathbf{u} = \mathrm{K}[\mathrm{R} \quad \mathbf{t}]\mathbf{x}$
  - Simplify for now: $\mathrm{K} = \mathrm{I}$
- Camera 1 (C1): $\mathrm{R} = \mathrm{I}, \mathbf{t} = \mathbf{0}$
  - No rotation, looks along Z
- Camera 2 (C2): shifted along X

$$\mathrm{R} = \mathrm{I}, \; t = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}$$



A Simple Stereo System

- Perfectly Parallel: The principal axes (the lines pointing straight out from the center of the lenses) of both cameras are parallel to each other and perpendicular to the baseline
- Perfectly Aligned: Image planes of the two cameras lie on the same plane. This means there is no vertical shift between the cameras; their y-axes are aligned
- Known Baseline: The distance between the centres of the two cameras, called the baseline (b), is precisely known



Project a Point into the Cameras

- Camera 1:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y \\ z \end{bmatrix} \equiv \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

- Camera 2:

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 & -b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x - b \\ y \\ z \end{bmatrix} \equiv \begin{bmatrix} (x - b)/z \\ y/z \\ 1 \end{bmatrix}$$

Depth and Disparity

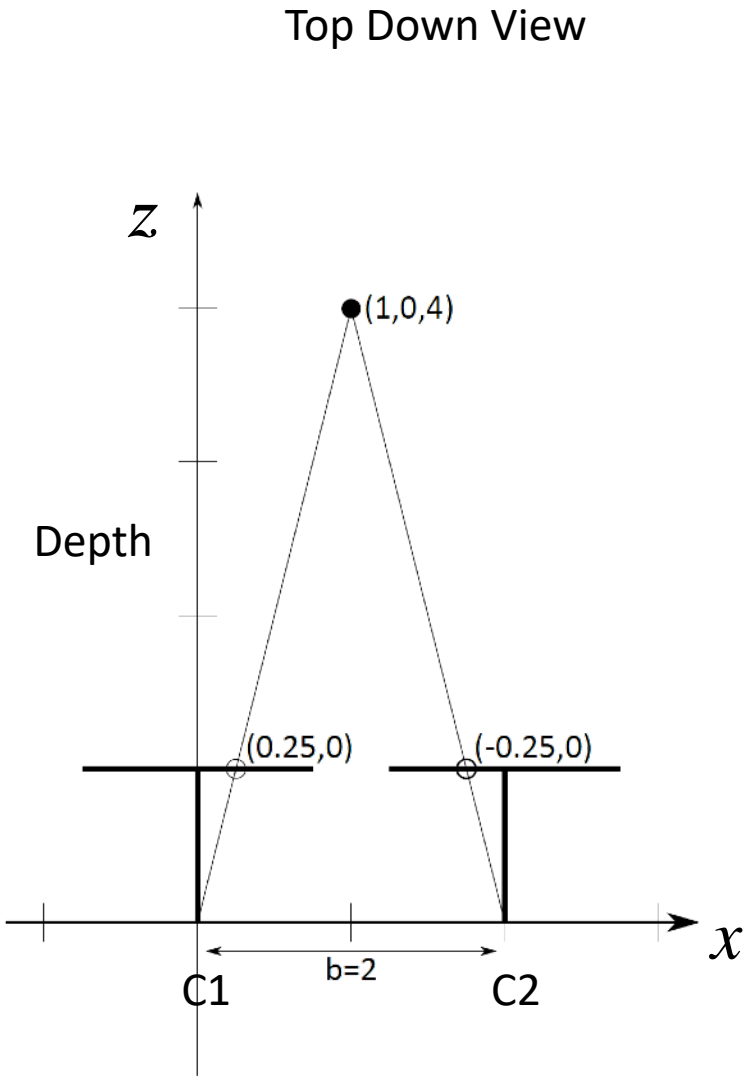
- Difference between the views is the disparity
  - In this case it is in u:

$$u_1 - u_2 = \frac{x}{z} - \frac{x-b}{z} = \frac{b}{z}$$

  - The v values are the same

- If we know b, can find z:

$$z = \frac{b}{u_1 - u_2}$$



Depth and Disparity

https://juniorxsound.github.io/THREE.SixDOF/examples/video.html

More General Camera Model

- Camera 1:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f & 0 & c_u & 0 \\ 0 & f & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} fx + c_u z \\ fy + c_u z \\ z \end{bmatrix} \equiv \begin{bmatrix} \frac{fx}{z} + c_u \\ \frac{fy}{z} + c_v \\ 1 \end{bmatrix}$$

- Camera 2:

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f & 0 & c_u & -fb \\ 0 & f & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\equiv \begin{bmatrix} \frac{f(x-b)}{z} + c_u \\ \frac{fy}{z} + c_v \\ 1 \end{bmatrix}$$

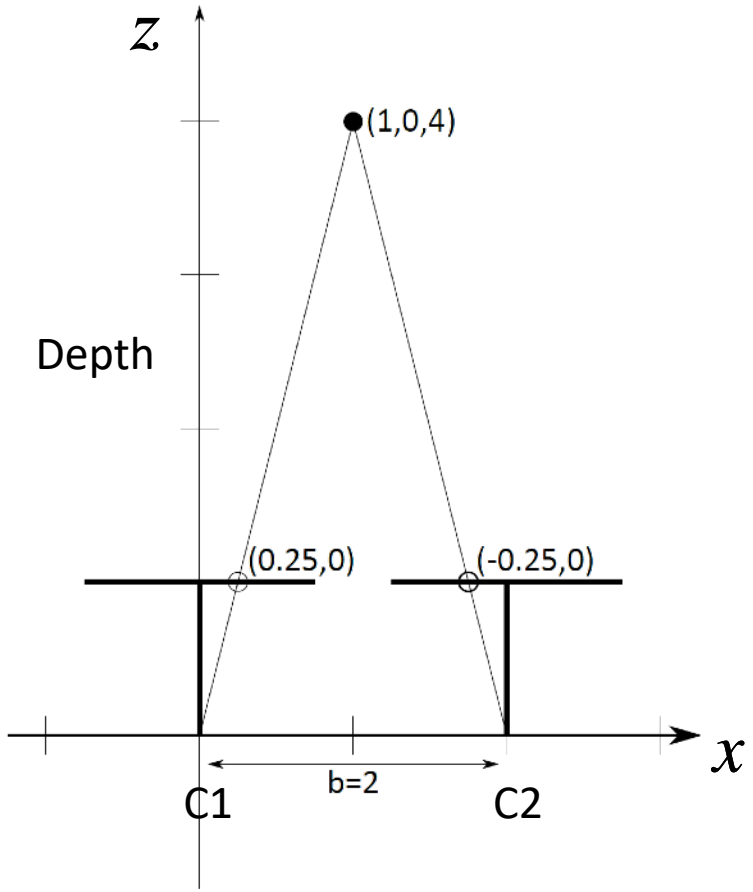
Depth Estimation

- And we solve for z:

$$u_1 - u_2 = \frac{fx}{z} - \frac{f(x-b)}{z}$$

$$z = \frac{fb}{u_1 - u_2}$$

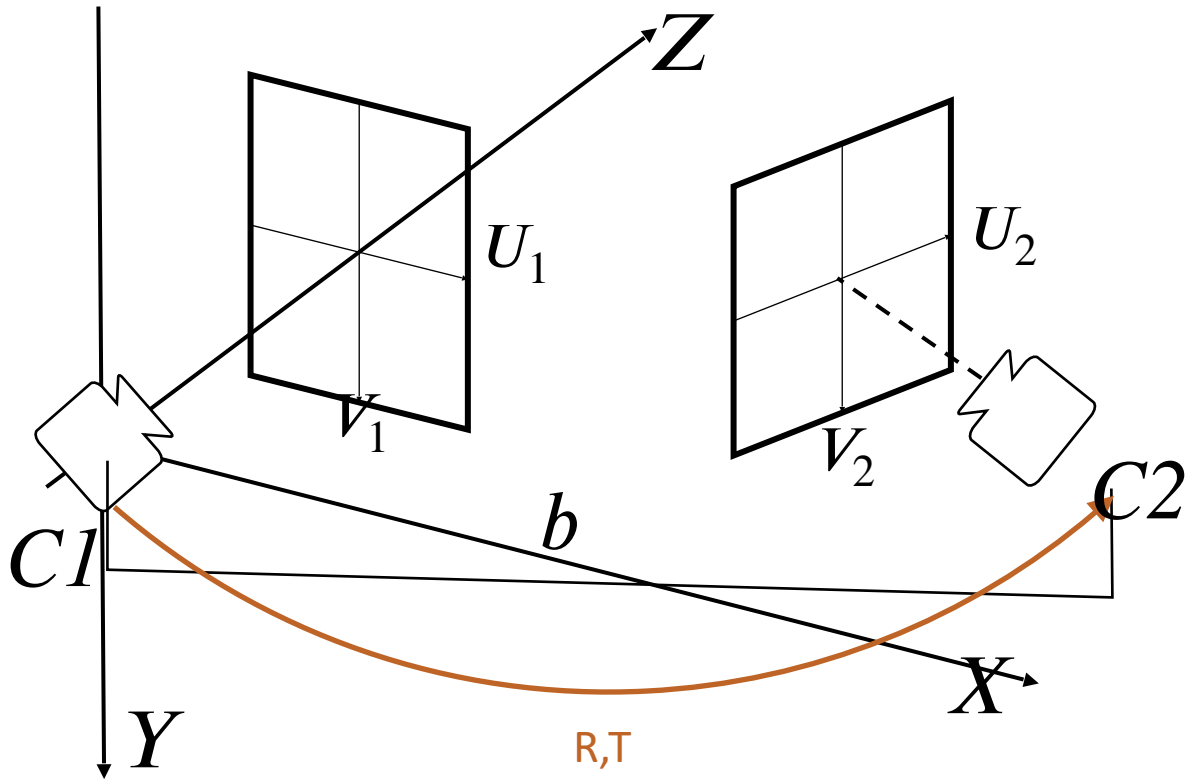
Top Down View



More General Stereo

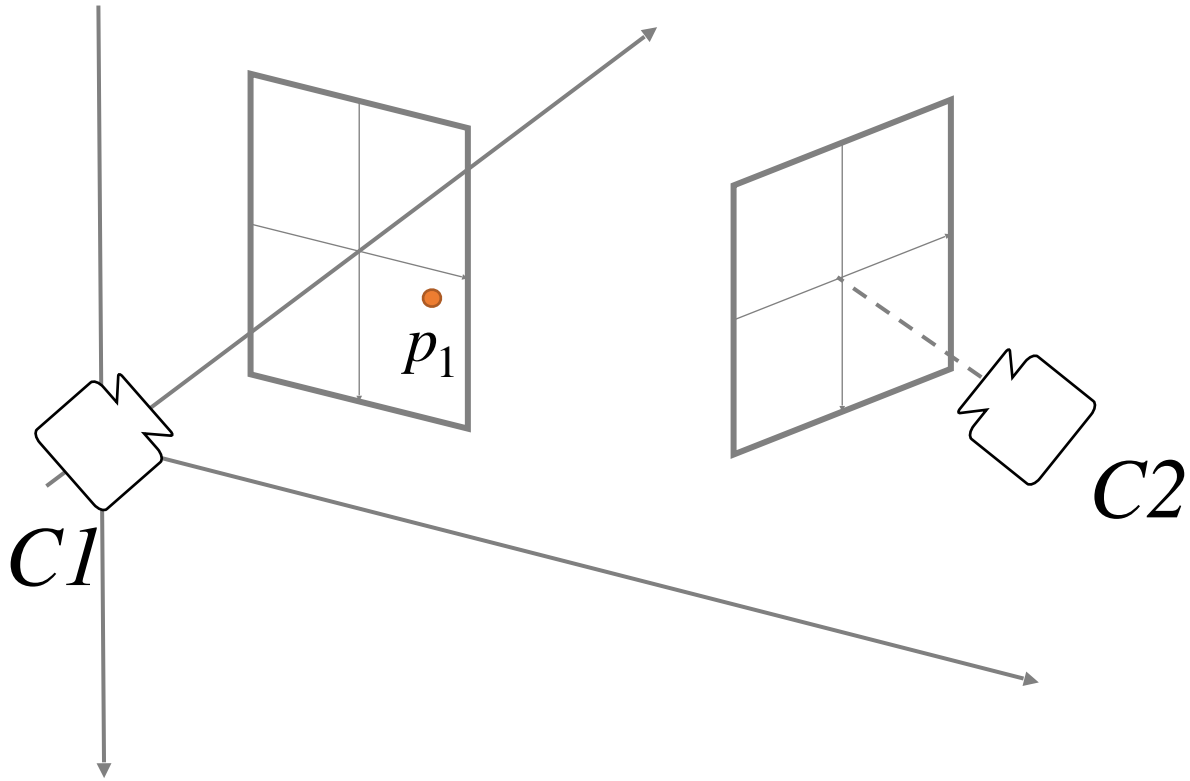
More General Stereo

- In the more general case:
  - The cameras have different calibration parameters
  - There is a rotation between the cameras
  - There is translation along all three axes
- We can still align the XYZ coordinate system with camera C1



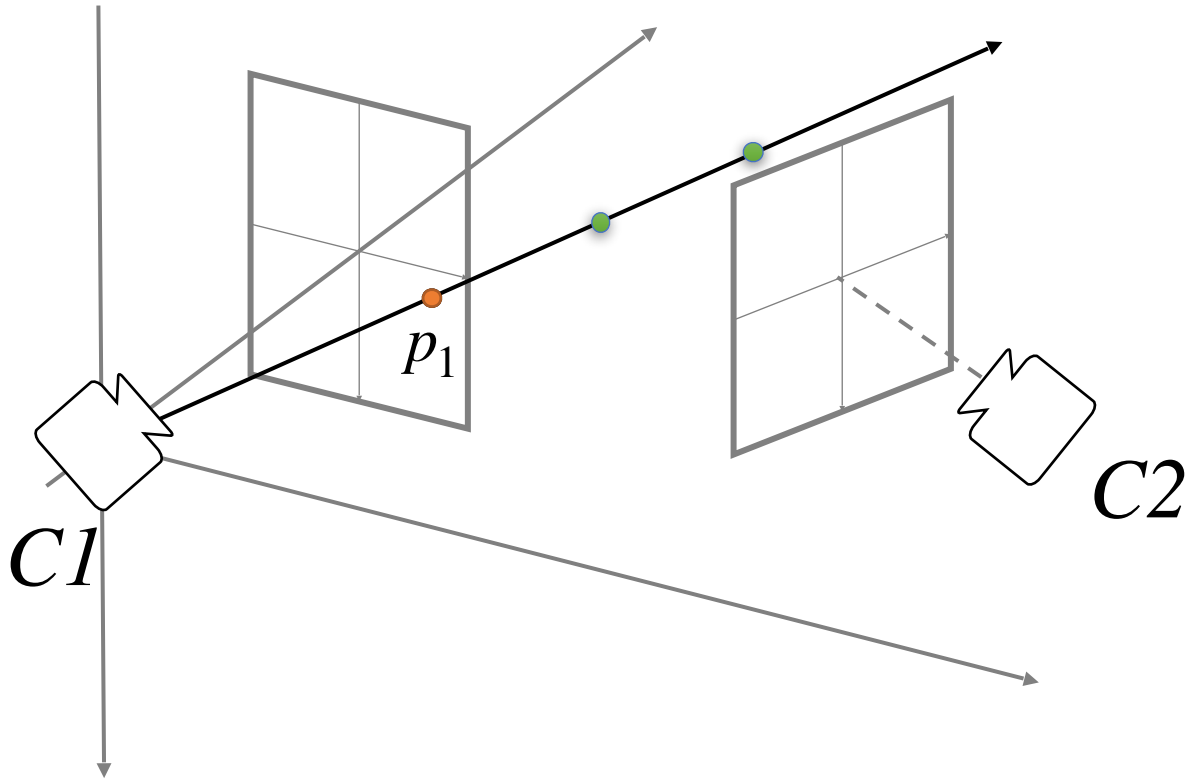
Epipolar Geometry

- Consider a point p_1 in the first camera



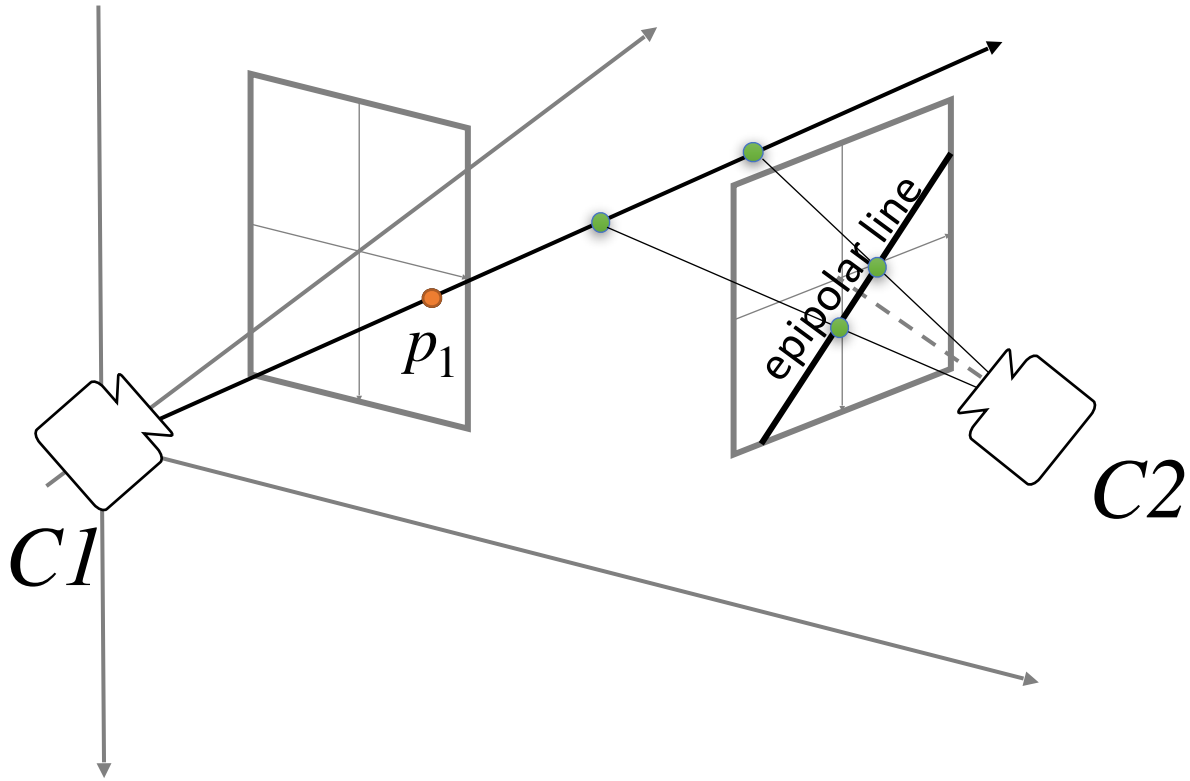
Epipolar Geometry

- Consider a point p_1 in the first camera
  - This gives a 3D line, along which the point must lie



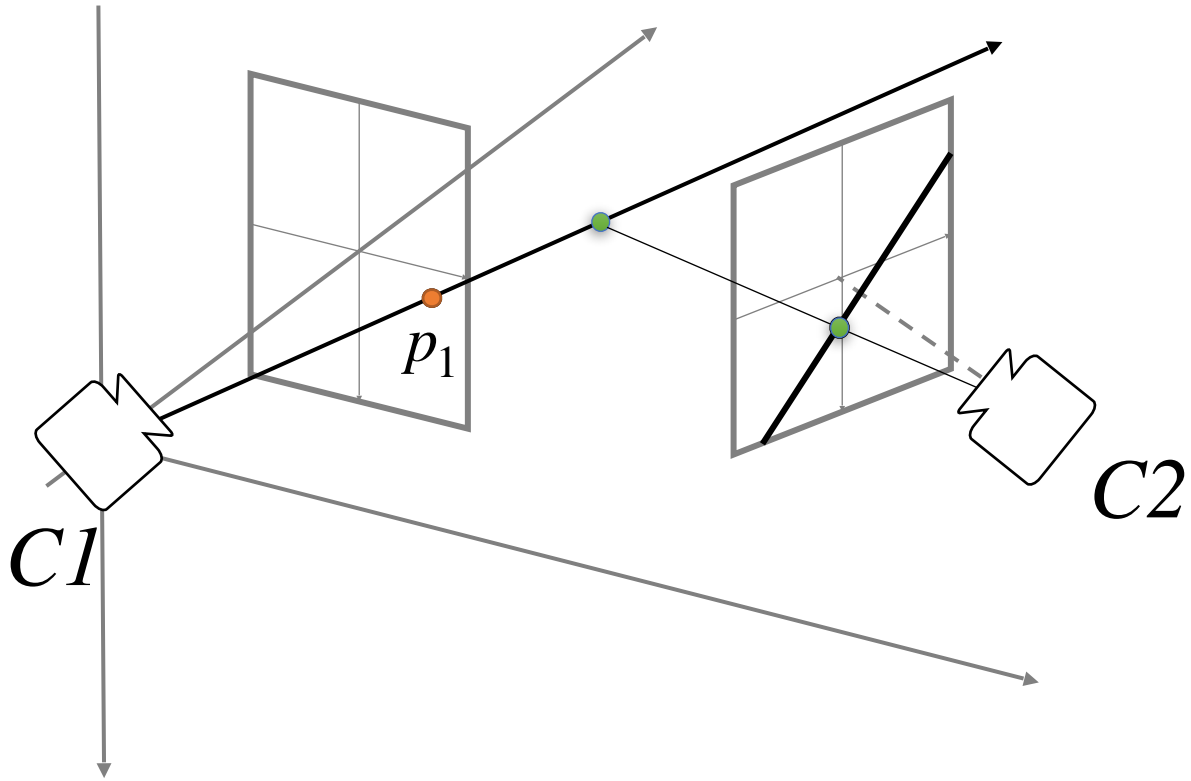
Epipolar Geometry

- Consider a point p_1 in the first camera
  - This gives a 3D line, along which the point must lie
  - The 3D line projects to a 2D line in the second image – this is called an **epipolar line**



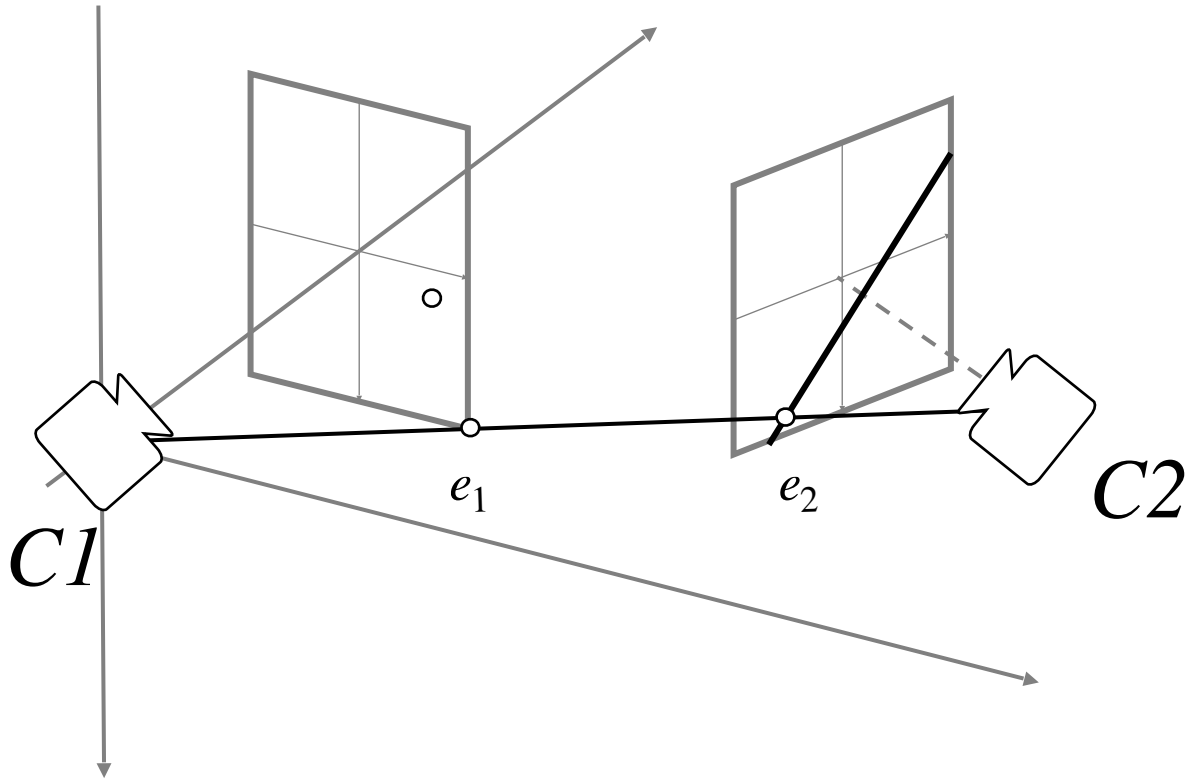
Epipolar Geometry

- Consider a point p_1 in the first camera
  - This gives a 3D line, along which the point must lie
  - The 3D line projects to a 2D line in the second image – this is called an **epipolar line**
  - The matching point must lie on this line



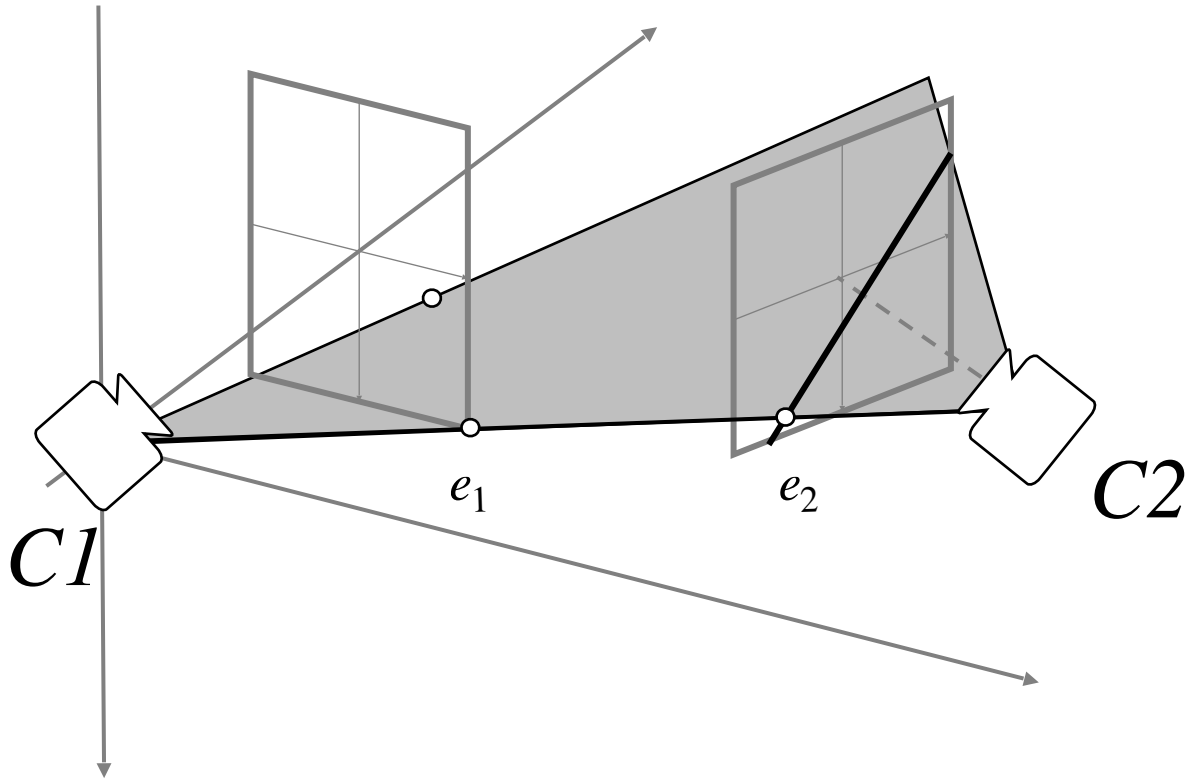
Epipoles and Epipolar line

- Draw a line between the camera centres
  - This crosses the image planes at the epipoles (e_1, e_2)



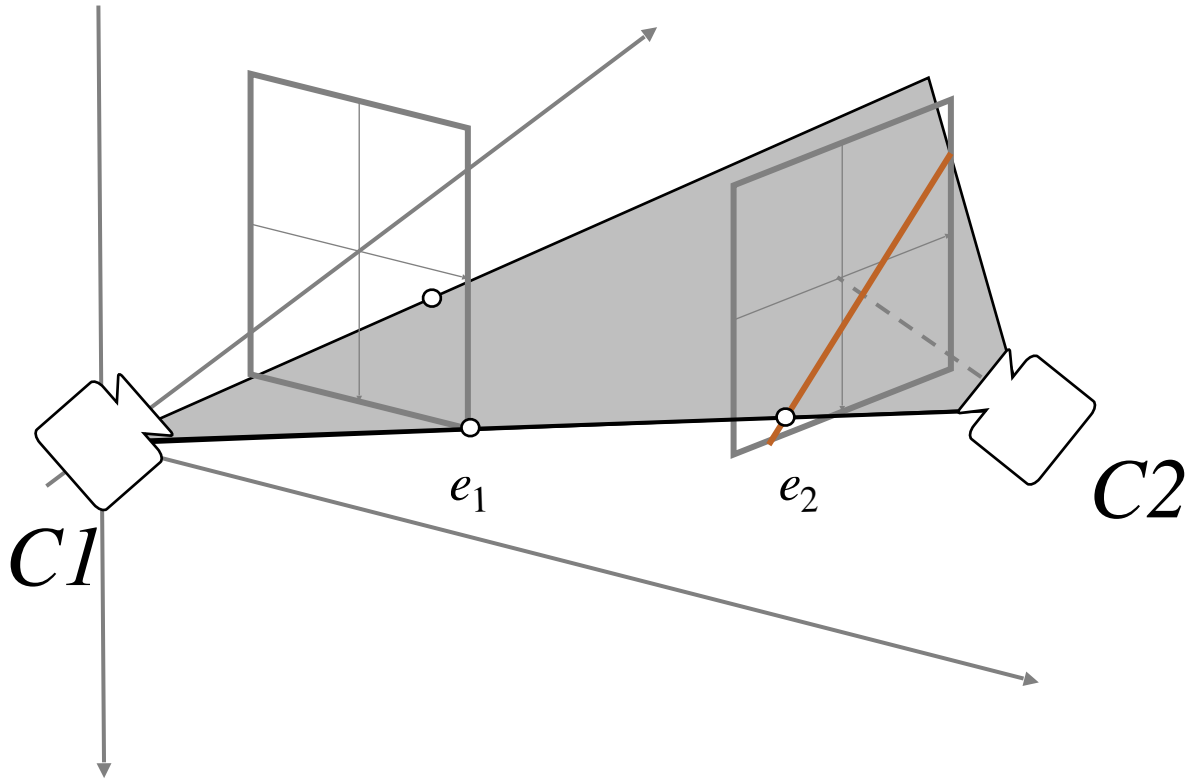
Epipoles and Epipolar line

- Draw a line between the camera centres
  - This crosses the image planes at the epipoles (e_1, e_2)
  - Make a plane through a point in one image and the two camera centres: epipolar plane



Epipoles and Epipolar line

- Draw a line between the camera centres
  - This crosses the image planes at the epipoles (e_1, e_2)
  - Make a plane through a point in one image and the two camera centres: epipolar plane
  - This intersects the second image at the epipolar line



Epipoles and Epipolar line

- Each point in one image makes an epipolar line
  - These all meet at the epipole in the other image
- For simple (parallel) case
  - Line through camera centres is the X axis
  - Epipoles are at infinity
  - Epipolar lines are the rows