

CS4243
Computer Vision
&
Pattern Recognition

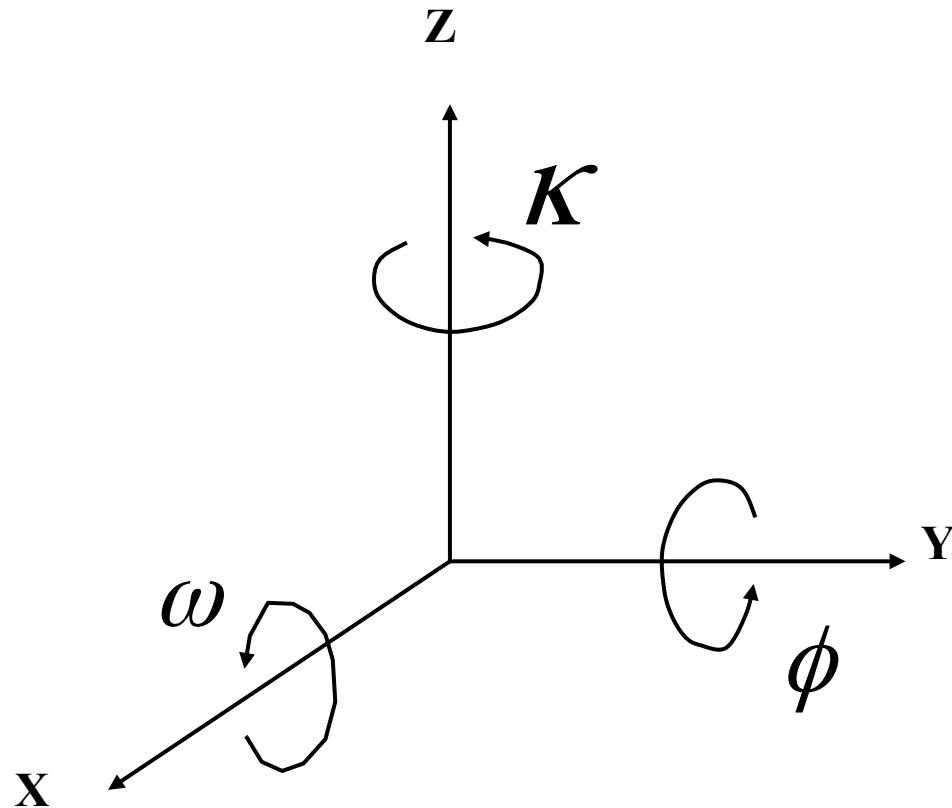
**Calibrating a Camera:
Intrinsic Parameters Calibration**

- Representation of rotation
- Camera intrinsic parameters calibration

Representation of Rotation -- Euler Angles

Euler angles

- pitch: rotation about x axis : ω
- yaw: rotation about y axis: ϕ
- roll: rotation about z axis: κ



Euler angles can be converted to rotation matrix (see “Machine Vision” pg 317).

$$R = R_z R_y R_x$$
$$= \begin{pmatrix} \cos(\kappa) \cos \phi & \cos(\kappa) \sin(\phi) \sin(\omega) - \sin(\kappa) \cos(\omega) & \cos(\kappa) \sin(\phi) \cos(\omega) + \sin(\kappa) \sin(\omega) \\ \sin(\kappa) \cos(\phi) & \sin(\kappa) \sin(\phi) \sin(\omega) + \cos(\kappa) \cos(\omega) & \sin(\kappa) \sin(\phi) \cos(\omega) - \cos(\kappa) \sin(\omega) \\ -\sin(\phi) & \cos(\phi) \sin(\omega) & \cos(\phi) \cos(\omega) \end{pmatrix}$$

Note: The above is for a right-handed system.
The matrix given in the book “Machine Vision”
is for a left handed system

Representation of Rotation -- Rotation Matrix

Rotation matrix R has two very important properties:

Property 1:

R is orthonormal, i.e. $R^T R = I$

In other words, $R^T = R^{-1}$

Property 2:

Determinant of R is 1.

Therefore, although the following matrix is orthonormal, it is not a rotation matrix because its determinant is -1.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \longleftarrow \text{this is a reflection matrix}$$

Representation of Rotation -- Axis-Angle Representation

- Rotation can be specified as a right-handed rotation by an angle θ about the axis specified by the unit vector

$$(\omega_x, \omega_y, \omega_z)$$

- This axis-angle representation has the same disadvantage as Euler angle representation:

It leads to algorithms that are not numerically well-conditioned. Therefore, although the axis-angle representation can be converted into a rotation matrix, the preferred way is to use “quaternions”.

Representation of Rotation -- Quaternions

- Quaternion yields well-conditioned numerical solutions
- A rotation is represented by a unit quaternion, which is a four-element unit vector

$$\mathbf{q} = (q_0, q_1, q_2, q_3)$$

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

- Conjugate of quaternion:

$$\mathbf{q}^* = (q_0, -q_1, -q_2, -q_3)$$

We can also denote the quaternion $\mathbf{q} = (q_0, q_1, q_2, q_3)$
by $\mathbf{q} = (s_q, \mathbf{v}_q)$

where $s_q = q_0$ is a scalar

and $\mathbf{v}_q = [q_1 \ q_2 \ q_3]$ is a vector

Similarly $\mathbf{q}^* = (s_q, -\mathbf{v}_q)$

Suppose we have a point $\mathbf{r} = [r_1 \ r_2 \ r_3]$ to be rotated by the rotation represented by the quaternion \mathbf{d}

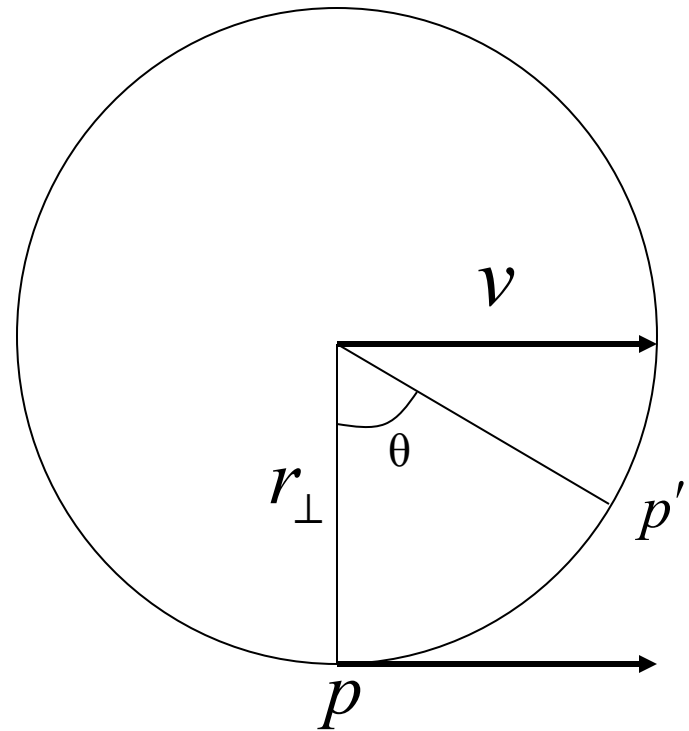
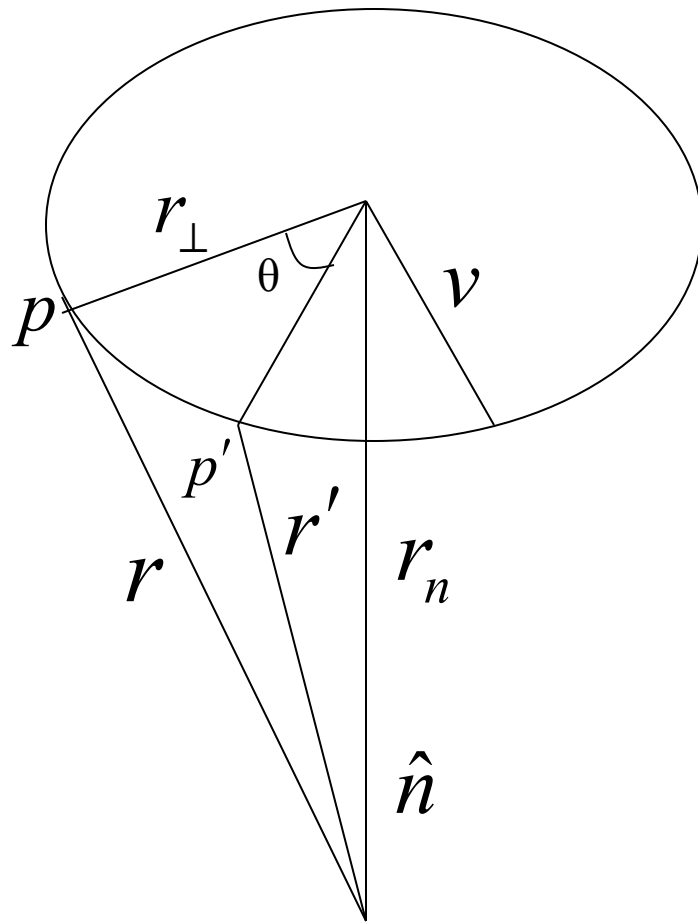
We represent the point \mathbf{r} with the quaternion $\mathbf{p} = (s_p, \mathbf{v}_p)$ with $s_p = 0$ and $\mathbf{v}_p = [r_1 \ r_2 \ r_3]$

Then the new location of the point \mathbf{r} after the rotation is given by

$$\mathbf{p}' = \mathbf{q} \mathbf{p} \mathbf{q}^*$$

where \mathbf{p}' is the quaternion representation of the new (i.e. after rotation) location of the point \mathbf{r} .

To see why $\mathbf{b}_i = \mathbf{d}\mathbf{b}\mathbf{d}_i^*$ represents a rotation, we quote below the proof given in *3D Computer Graphics* by Alan Watt, Addison-Wesley



We can write down the following based on the figure in the previous slide:

$$\begin{aligned}\mathbf{r}' &= \mathbf{r}_n + \mathbf{r}_\perp \cos \theta + \mathbf{v} \sin \theta \\ &= (\mathbf{r} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} + [\mathbf{r} - (\mathbf{r} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}] \cos \theta + [\hat{\mathbf{n}} \wedge \mathbf{r}] \sin \theta \\ &= \mathbf{r} \cos \theta + (\mathbf{r} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}(1 - \cos \theta) + [\hat{\mathbf{n}} \wedge \mathbf{r}] \sin \theta\end{aligned}$$

—— Eqn(*)

The new location of \mathbf{r} after rotation by angle θ is given by \mathbf{r}'

We will represent \mathbf{r}' by its quaternion representation \mathbf{p}' .

We will see how quaternion multiplications can give us the same result as in Eqn(*)).

First of all, we need to define quaternion multiplication. Multiplication of quaternions \mathbf{q} and \mathbf{p} is defined as

$$\mathbf{qp} = (s_q s_p - \mathbf{v}_q \cdot \mathbf{v}_p, \quad \mathbf{v}_q \wedge \mathbf{v}_p + s_q \mathbf{v}_p + s_p \mathbf{v}_q)$$

Alternatively, in terms of the individual components of \mathbf{p} and \mathbf{q} , the multiplication of quaternions \mathbf{p} and \mathbf{q} is given by

$$\mathbf{pq} = \begin{pmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2 \\ p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1 \\ p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0 \end{pmatrix} \quad \text{note: beware of} \\ \text{typo in book}$$

we set the following: $\mathbf{r} = (x, y, z)$



$$\mathbf{p} = (0, x, y, z)$$

i.e. $s_p = 0$

It can be shown that

$$\mathbf{q} \mathbf{p} \mathbf{q}^* = (0, -(\mathbf{v}_q \cdot \mathbf{v}_q) \mathbf{v}_p + 2(\mathbf{v}_q \cdot \mathbf{v}_p) \mathbf{v}_q + 2s_q (\mathbf{v}_q \wedge \mathbf{v}_p) + s_q^2 \mathbf{v}_p)$$

Note: You may want to verify the above. You may need the identity:

$$\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} = (\mathbf{a} \cdot \mathbf{c}) \mathbf{b} - (\mathbf{a} \cdot \mathbf{b}) \mathbf{c}$$

Now, if we let $s_q = \cos(\frac{\theta}{2})$

$$\mathbf{v}_q = [\sin(\frac{\theta}{2})\omega_x \quad \sin(\frac{\theta}{2})\omega_y \quad \sin(\frac{\theta}{2})\omega_z]$$

$$\text{i.e. } \mathbf{q} = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})\omega_x, \sin(\frac{\theta}{2})\omega_y, \sin(\frac{\theta}{2})\omega_z)$$

we will see that

$$\mathbf{q} \mathbf{p} \mathbf{q}^* = \mathbf{v}_p \cos \theta + (\mathbf{v}_p \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}) \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} (1 - \cos \theta) + \left[\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \wedge \mathbf{v}_p \right] \sin \theta$$

Comparing with Eqn(*), we have

$$\mathbf{v}_p = \mathbf{r} \quad \text{and} \quad (\omega_x, \omega_y, \omega_z) = \hat{\mathbf{n}}$$

So the geometrical interpretation of $\mathbf{q} \mathbf{p} \mathbf{q}^*$ is that the point \mathbf{r} has been rotated counter-clockwise (i.e. right-handed) by the angle θ w.r.t. the axis $(\omega_x, \omega_y, \omega_z)$

We can also express a rotation matrix using the components of a quaternion:

- Given a unit quaternion

$$\mathbf{q} = (q_0, q_1, q_2, q_3)$$

the equivalent rotation matrix is given by

$$R(\mathbf{q}) = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{pmatrix}$$

(note: there is a typo in the book)

so $\mathbf{r}' = R(\mathbf{q}) \mathbf{r}$

Camera Intrinsic Parameters Calibration

- Homogeneous coordinates
- A fast and accurate calibration method

Homogeneous Coordinates

Euclidean space

$$R^3$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Projective space

$$P^3$$

$$\begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

w is a scale factor.

To transform from projective to Euclidean, divide the projective coordinates by the last element of the vector (w non-zero).

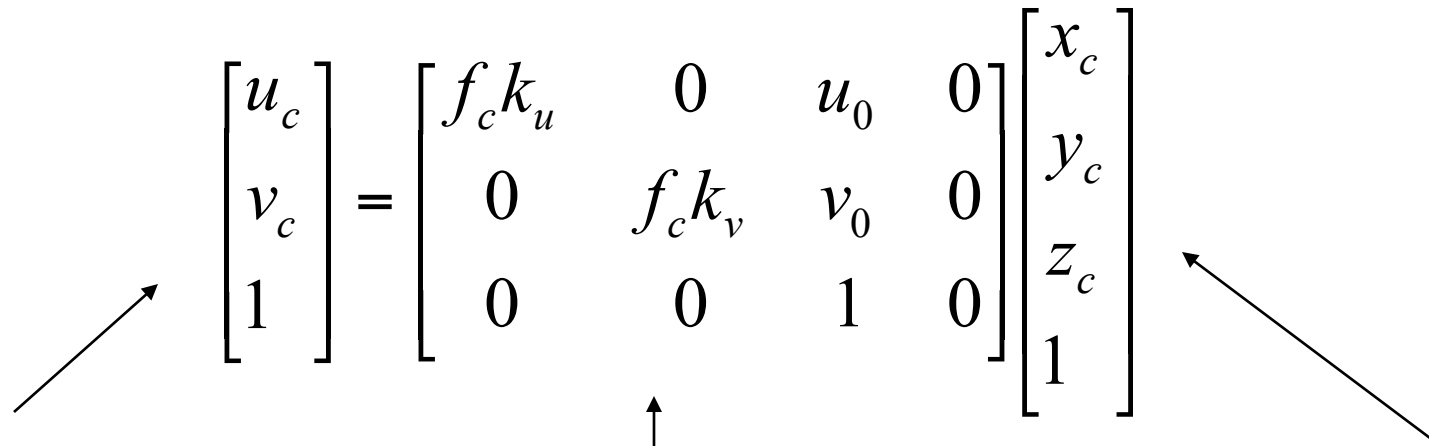
If $w=0$, the point in P^3 is the ideal point, or point at infinity.

The use of homogeneous coordinates allows the pin-hole projection to be represented using linear algebra

Camera Internal Parameters Calibration

*D. LaRose, “A Fast, Affordable System for Augmented Reality”
CMU-RI-TR-98-21, April 30, 1998.*

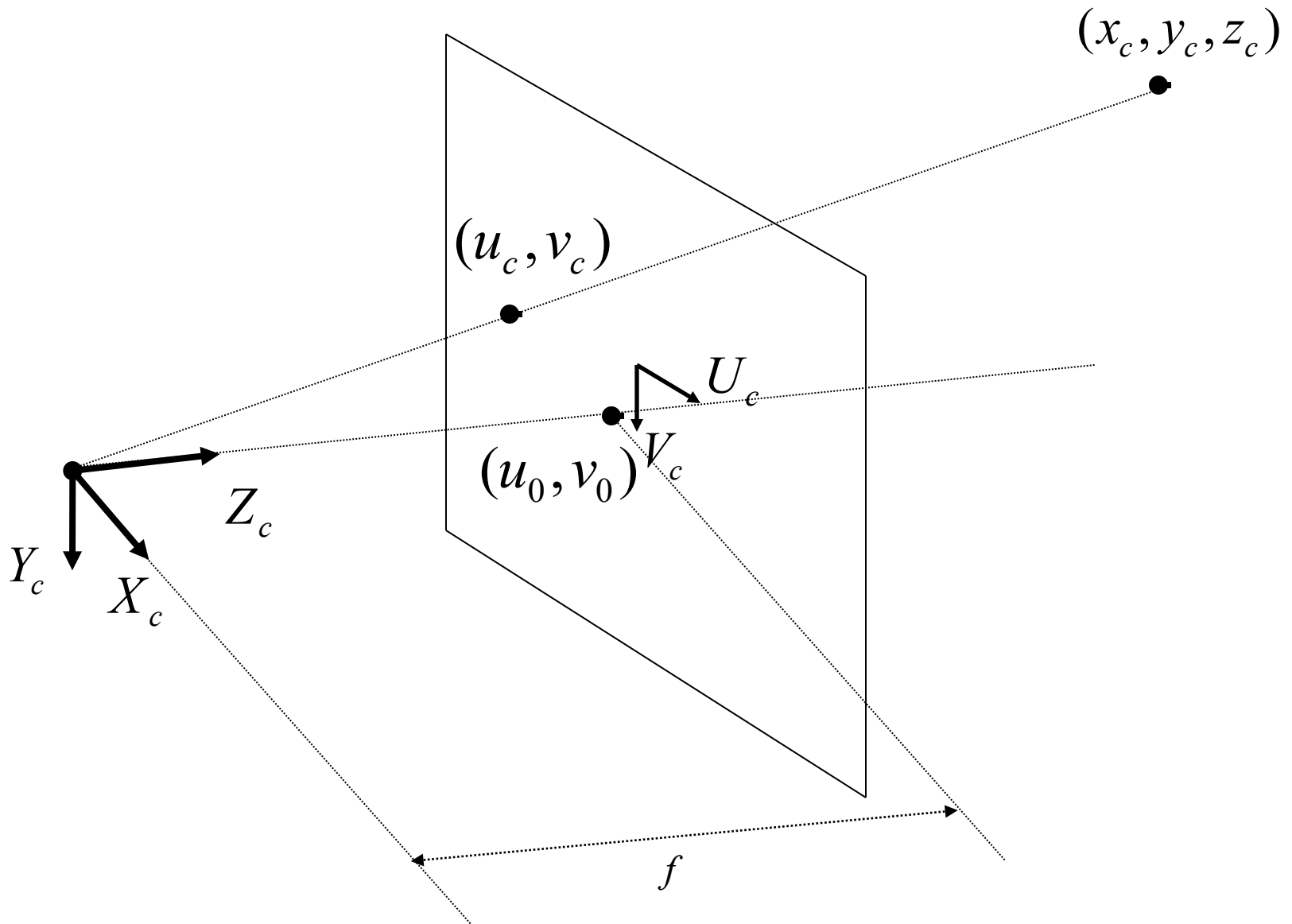
http://www.ri.cmu.edu/cgi-bin/tech_reports.cgi?year=1998&text=0

$$\begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} = \begin{bmatrix} f_c k_u & 0 & u_0 & 0 \\ 0 & f_c k_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$
The diagram shows the camera calibration equation. Three arrows point from descriptive text at the bottom to the components of the equation: one from the left vector to the 2D image point, one from the matrix to the intrinsic parameters, and one from the right vector to the 3D point.

2D image point
in camera image
plane coordinates

Matrix containing
intrinsic parameters
to be calibrated

3D point in camera
coordinate system



$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \approx \begin{bmatrix} k_p & 0 & 0 \\ 0 & k_p & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

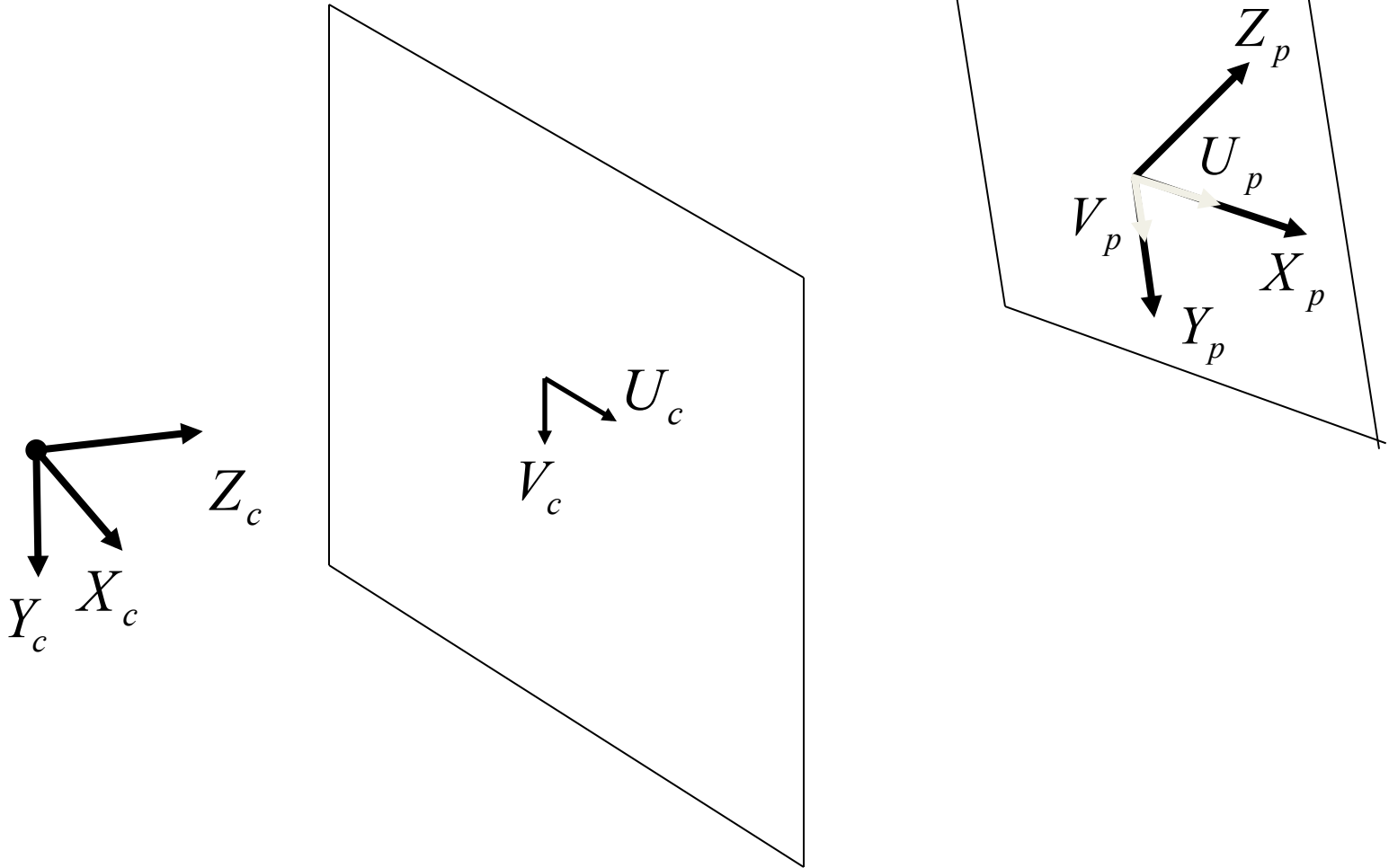
3D coordinates of points.
Notice that z_p is 0,
because we define
the flat plane pattern to
lie on xy plane.

Matrix that
does scaling

2D coordinates of
points on the flat
plane pattern.

pattern

camera



$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \approx \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

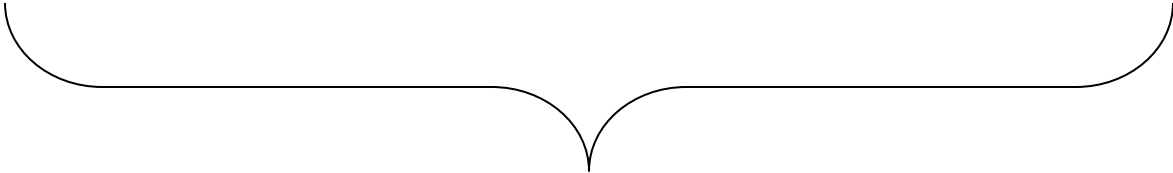


Transformation matrix that aligns the 3D coordinate system of the flat plane pattern to the 3D coordinate system of the camera

Combining the equations in the previous slides, we can relate the 2D coordinate of the points on the flat plane pattern to the 2D coordinate of the points on the image plane of the camera. The relationship is through a matrix

$$\begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} \approx \underbrace{\begin{bmatrix} f_c k_u & 0 & u_0 & 0 \\ 0 & f_c k_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k_p & 0 & 0 \\ 0 & k_p & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Matrix}} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

This matrix can be simplified, and will be called ${}_c H_p$, see the next slide

$$\begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} \approx \begin{bmatrix} f_c k_u & 0 & u_0 \\ 0 & f_c k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} k_p & 0 & 0 \\ 0 & k_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$


$${}_c H_p$$

$${}_c H_p \approx \begin{bmatrix} f_c k_u & 0 & u_0 \\ 0 & f_c k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} k_p & 0 & 0 \\ 0 & k_p & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{f_c k_u} & 0 & \frac{-u_0}{f_c k_u} \\ 0 & \frac{1}{f_c k_v} & \frac{-v_0}{f_c k_v} \\ 0 & 0 & 1 \end{bmatrix} {}_c H_p \begin{bmatrix} \frac{1}{k_p} & 0 & 0 \\ 0 & \frac{1}{k_p} & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

Let ${}_c H_p = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$

${}_c H_p$ is known as a homography matrix. We will see later how it can be found. For the time being, let's assume we know

${}_c H_p$

$$\begin{bmatrix} \frac{h_{11} - u_0 h_{31}}{f_c k_u} & \frac{h_{12} - u_0 h_{32}}{f_c k_u} & \frac{(h_{13} - u_0) k_p}{f_c k_u} \\ \frac{h_{21} - v_0 h_{31}}{f_c k_v} & \frac{h_{22} - v_0 h_{32}}{f_c k_v} & \frac{(h_{23} - v_0) k_p}{f_c k_v} \\ h_{31} & h_{32} & k_p \end{bmatrix} \approx \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

This matrix consists of h_{ij} , which are the elements of the matrix ${}_c H_p$, as well as the intrinsic camera parameters that we want to solve. (it also contains k_p , but we won't use it).

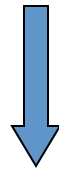
In the next two slides, we will see how we can make use of the orthonormal property of rotation matrix (i.e. property 1) to solve for the intrinsic parameters.

Two constraints

Constraint 1:

Dot product of columns of rotation matrix are is 0

$$\begin{bmatrix} r_{11} & r_{21} & r_{31} \end{bmatrix} \begin{bmatrix} r_{12} \\ r_{22} \\ r_{32} \end{bmatrix} = 0$$



$$\left(\frac{1}{f_c k_u} \right)^2 (h_{11} - u_0 h_{31})(h_{12} - u_0 h_{32}) + \left(\frac{1}{f_c k_v} \right)^2 (h_{21} - v_0 h_{31})(h_{22} - v_0 h_{32}) + h_{31} h_{32} = 0$$

Constraint 2:

Magnitude of columns of rotation matrix is 1 (and therefore equal to each other)

$$r_{11}^2 + r_{21}^2 + r_{31}^2 - r_{12}^2 - r_{22}^2 - r_{32}^2 = 0$$



$$\left(\frac{1}{f_c k_u}\right)^2 ((h_{11} - u_0 h_{31})^2 - (h_{12} - u_0 h_{32})^2) + \left(\frac{1}{f_c k_v}\right)^2 ((h_{21} - v_0 h_{31})^2 - (h_{22} - v_0 h_{32})^2) + h_{31}^2 - h_{32}^2 = 0$$

$$\text{Let } \theta_1 = f_c k_u \quad \theta_2 = f_c k_v \quad \theta_3 = u_0 \quad \theta_4 = v_0$$

We want to find θ such that the following is minimized:

$$e(\theta) = \sum_{i=1}^N (g_1(H_i, \theta)^2 + g_2(H_i, \theta)^2)$$

where

$$g_1(\theta, H) = \left(\frac{1}{\theta_1} \right)^2 (h_{11} - \theta_3 h_{31})(h_{12} - \theta_3 h_{32}) + \left(\frac{1}{\theta_2} \right)^2 (h_{21} - \theta_4 h_{31})(h_{22} - \theta_4 h_{32}) + h_{31} h_{32}$$

$$g_2(\theta, H) = \left(\frac{1}{\theta_1} \right)^2 ((h_{11} - \theta_3 h_{31})^2 - (h_{12} - \theta_3 h_{32})^2) + \left(\frac{1}{\theta_2} \right)^2 ((h_{21} - \theta_4 h_{31})^2 - (h_{22} - \theta_4 h_{32})^2) + h_{31}^2 - h_{32}^2$$

N is total number of frames used.

Optimization function is non-linear. Use non-linear optimization techniques like Levenberg Marquardt to solve for the $\hat{\theta}$.

See the book “Numerical Recipes in C” for the explanation and code for Levenberg Marquardt technique.

Once the $\hat{\theta}$ are calculated, obtain the intrinsic parameters using

$$f_c k_u = \hat{\theta}_1$$

$$f_c k_v = \hat{\theta}_2$$

$$u_0 = \hat{\theta}_3$$

$$v_0 = \hat{\theta}_4$$

In the next two slides, we will see how the homography matrix H can be calculated.

Since pattern is on a flat plane, there exists a linear relationship called homography between the pattern coordinates and image coordinates.

$$\begin{bmatrix} \alpha u_c \\ \alpha v_c \\ \alpha \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

$$u_c = \frac{h_{11}u_p + h_{12}v_p + h_{13}}{h_{31}u_p + h_{32}v_p + h_{33}}$$

$$v_c = \frac{h_{21}u_p + h_{22}v_p + h_{23}}{h_{31}u_p + h_{32}v_p + h_{33}}$$

$$\begin{bmatrix} u_p & v_p & 1 & 0 & 0 & 0 & -u_c u_p & -u_c v_p & -u_c \\ 0 & 0 & 0 & u_p & v_p & 1 & -v_c u_p & -v_c v_p & -v_c \\ & & \vdots & & & & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Solve for h_{ij} using the above equation.

CS4243
Computer Vision
&
Pattern Recognition

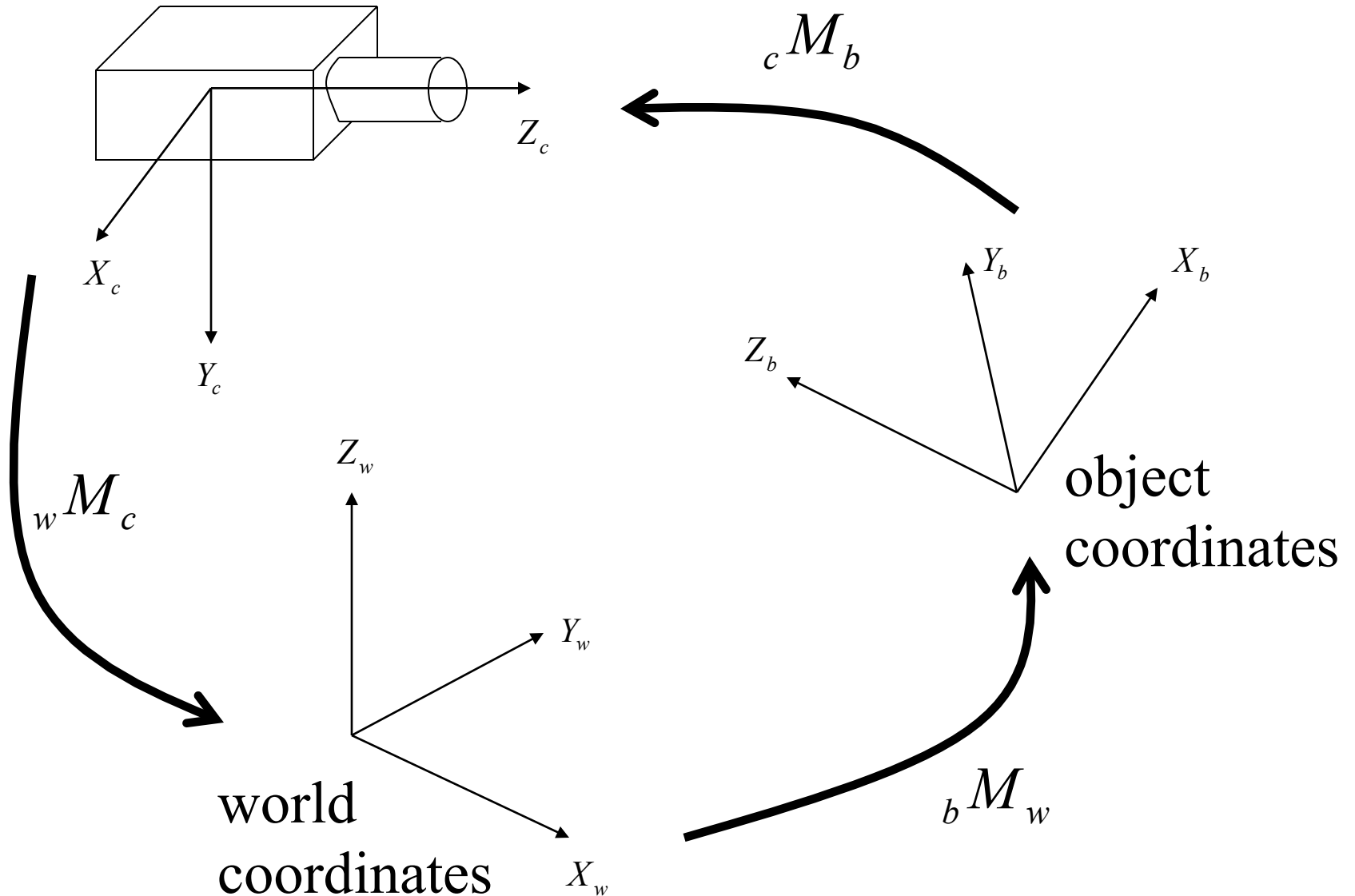
Calibrating a Camera:
Extrinsic Parameters Calibration

Exterior, Relative, Absolute Orientation

“Machine Vision” chapter 12

“Robot Vision” by B.K.P. Horn chapter 13

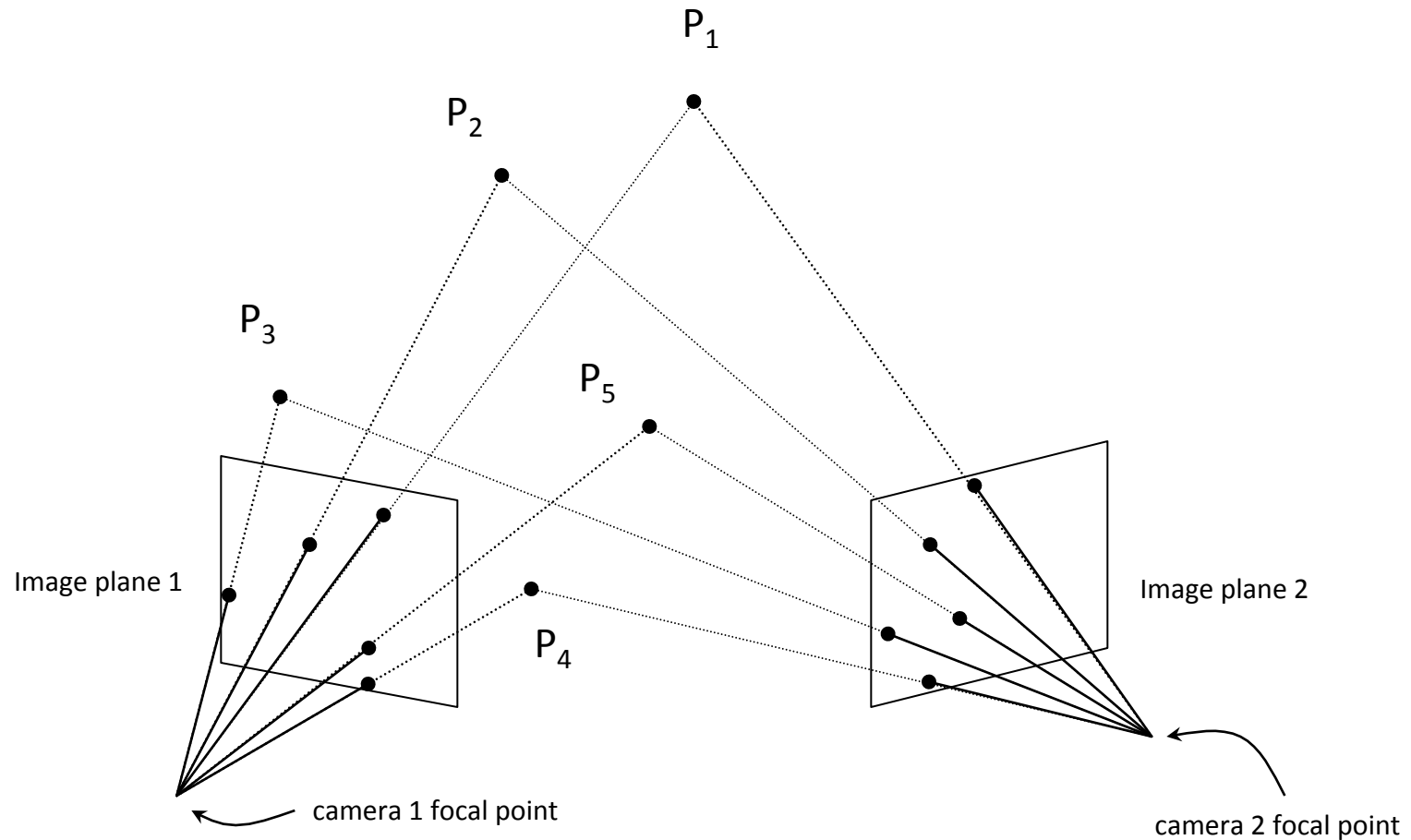
camera coordinates



Camera External Parameters Calibration

- To calibrate the relationship between image plane coordinates and 3D scene point coordinates:
The **exterior** orientation problem
- To calibrate the position and orientation between 2 cameras :
The **relative** orientation problem
- To calibrate the rigid body transformation between 2 coordinate systems:
The **absolute** orientation problem

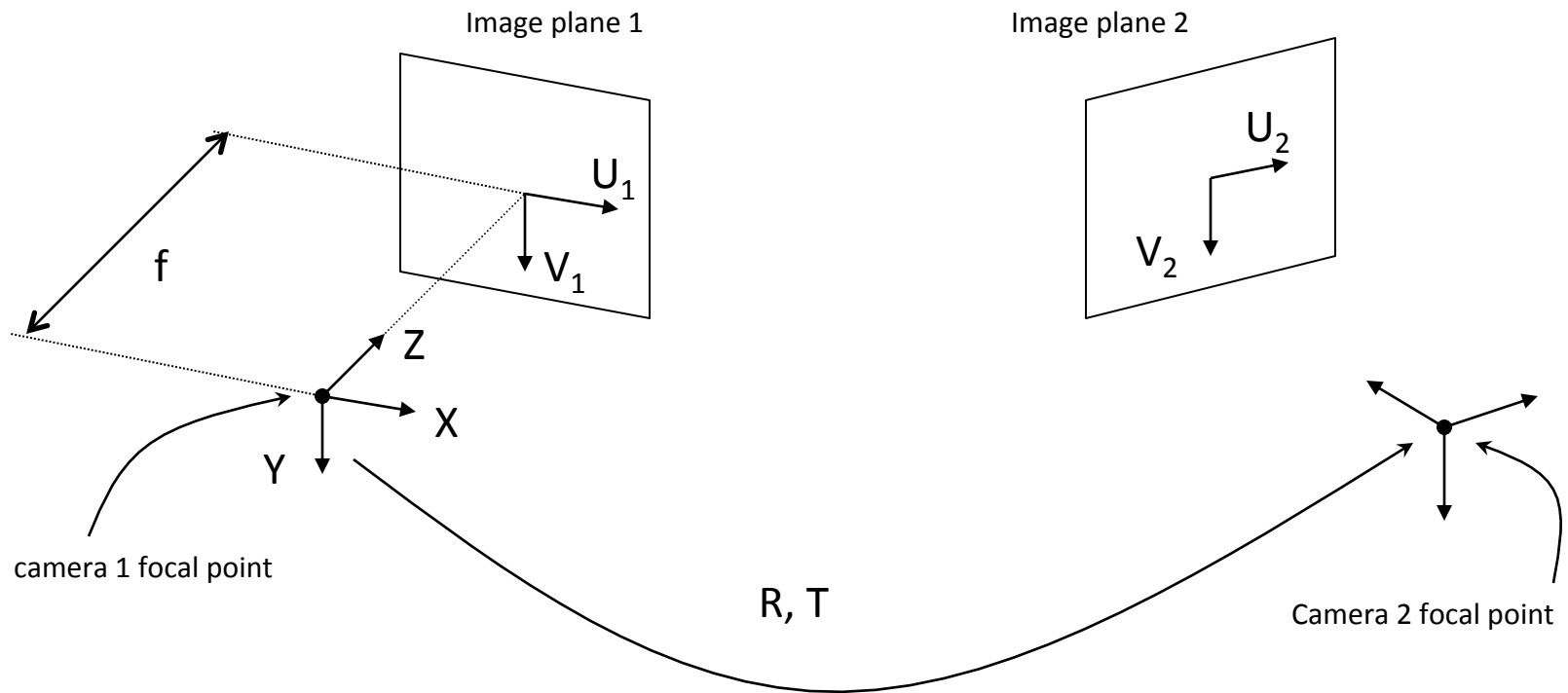
Coordinate Transformation: Relative Orientation



From the projections (image points) of the 3D points P_i , we want to recover the relative rotation and translation between image planes 1 and 2.

Coordinate Transformation: Relative Orientation

Let image plane1 be the reference frame. We fix (w.l.o.g.) the global coordinate system (X,Y,Z) to align with image plane1 coordinate system (U_1,V_1) .



Coordinate Transformation: Relative Orientation

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = R \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$

$$\left. \begin{aligned} u_1 &= \frac{f_1 x_1}{z_1} \\ v_1 &= \frac{f_1 y_1}{z_1} \\ u_2 &= \frac{f_2 x_2}{z_2} \\ v_1 &= \frac{f_1 y_1}{z_1} \end{aligned} \right\} \longrightarrow \begin{pmatrix} \frac{u_1 z_1}{f_1} \\ \frac{v_1 z_1}{f_1} \\ z_1 \end{pmatrix} = R \begin{pmatrix} \frac{u_2 z_2}{f_2} \\ \frac{v_2 z_2}{f_2} \\ z_2 \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$

Coordinate Transformation: Relative Orientation

suppose $R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$

we have

$$\frac{u_1}{f_1} = \frac{r_{11} \frac{u_2}{f_2} + r_{12} \frac{v_2}{f_2} + r_{13} + \frac{T_x}{z_2}}{r_{31} \frac{u_2}{f_2} + r_{32} \frac{v_2}{f_2} + r_{33} + \frac{T_z}{z_2}}$$

$$\frac{v_1}{f_1} = \frac{r_{21} \frac{u_2}{f_2} + r_{22} \frac{v_2}{f_2} + r_{23} + \frac{T_y}{z_2}}{r_{31} \frac{u_2}{f_2} + r_{32} \frac{v_2}{f_2} + r_{33} + \frac{T_z}{z_2}}$$

Coordinate Transformation: Relative Orientation

No. of constraints

So for each correspondence pair (i.e. conjugate pair), we can write 2 independent equations. We can also write the 6 orthonormal constraints and 1 constraint to fix the baseline length ambiguity. Therefore, for n correspondence points, we have $6+1+2n = 7+2n$ constraints

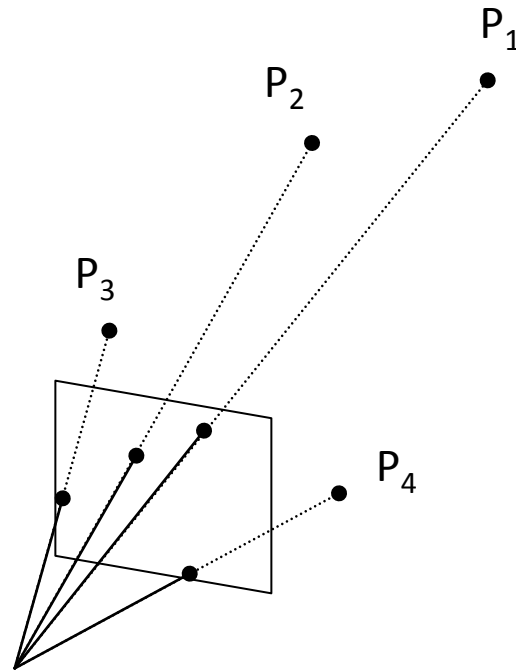
No. of unknowns

For each correspondence pair, we have 1 ambiguity in depth. We also have 9 unknowns in the rotation matrix R and 3 unknowns in the translation vector T . Therefore, we have altogether $9+3+n = 12+n$ unknowns

No. of points required to solve for R and T

$$7 + 2n \geq 12 + n \quad \longrightarrow \quad n \geq 5$$

Coordinate Transformation: Exterior Orientation

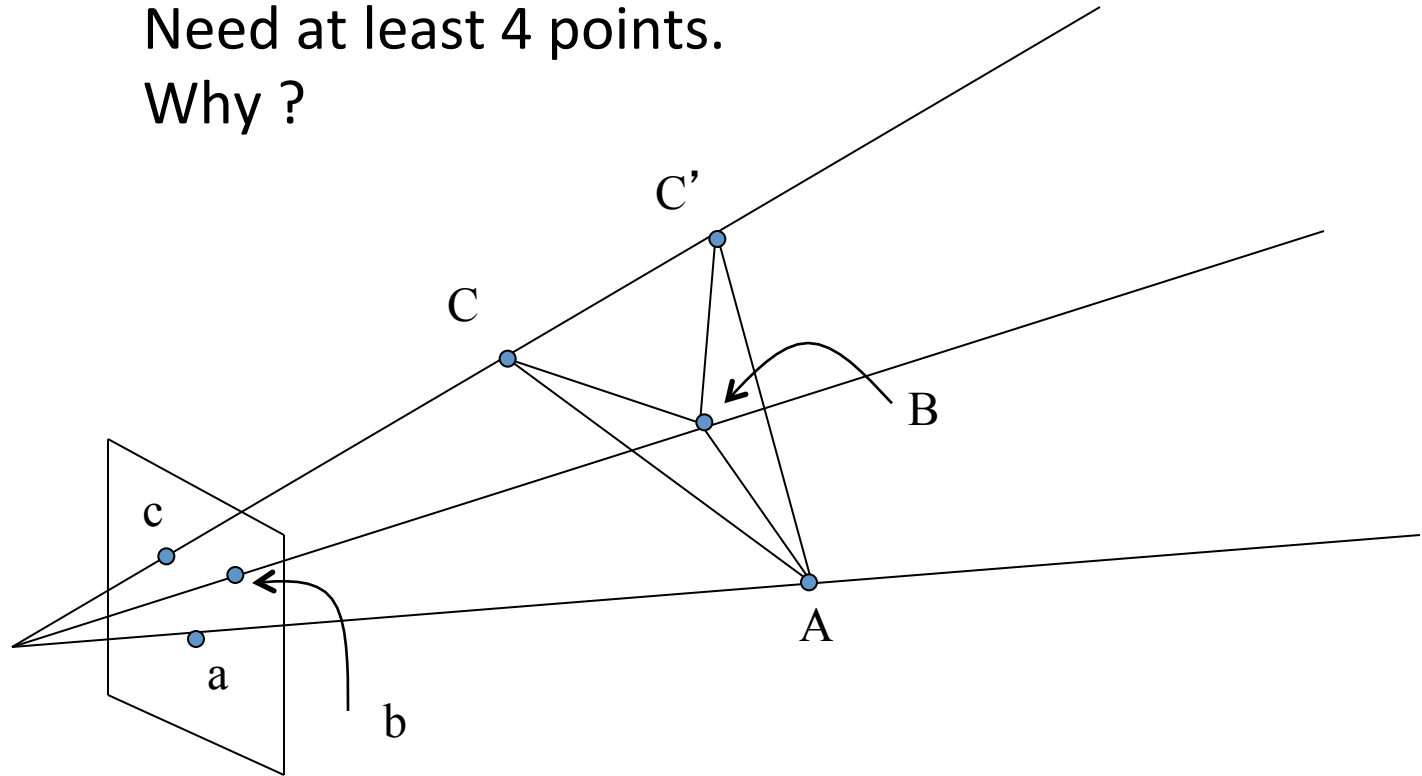


Exterior orientation is also known as the hand-eye problem. Given a set of 3D scene points of known coordinates, the task is to compute the orientation of the image plane given the image coordinates of these 3D points.

In other words, the exterior orientation problem is to determine the position and orientation of the bundle of rays corresponding to image points with respect to the absolute coordinate system of the 3D scene.

Coordinate Transformation: Exterior Orientation

Need at least 4 points.
Why ?



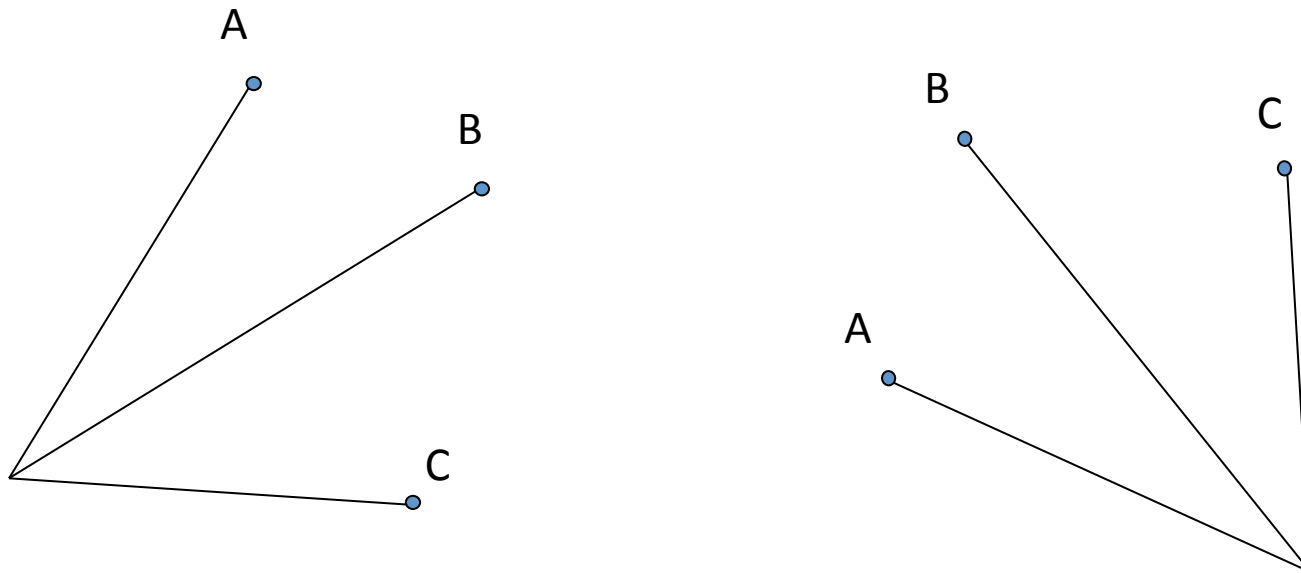
So three points are not enough because both the triangles ABC and ABC' project to the same image points abc .

Therefore, there are more than one ways the image plane could orientate with respect to a 3D absolute coordinate system, if A and B are fixed.

The fourth point removes the uncertainty.
Therefore, the minimum number of points is 4.

Coordinate Transformation: Absolute Orientation

Need at least 3 points.



3 points are required to align the two coordinate systems