Lecture 3: CS574

Aug 31, 2011

Review: Admin

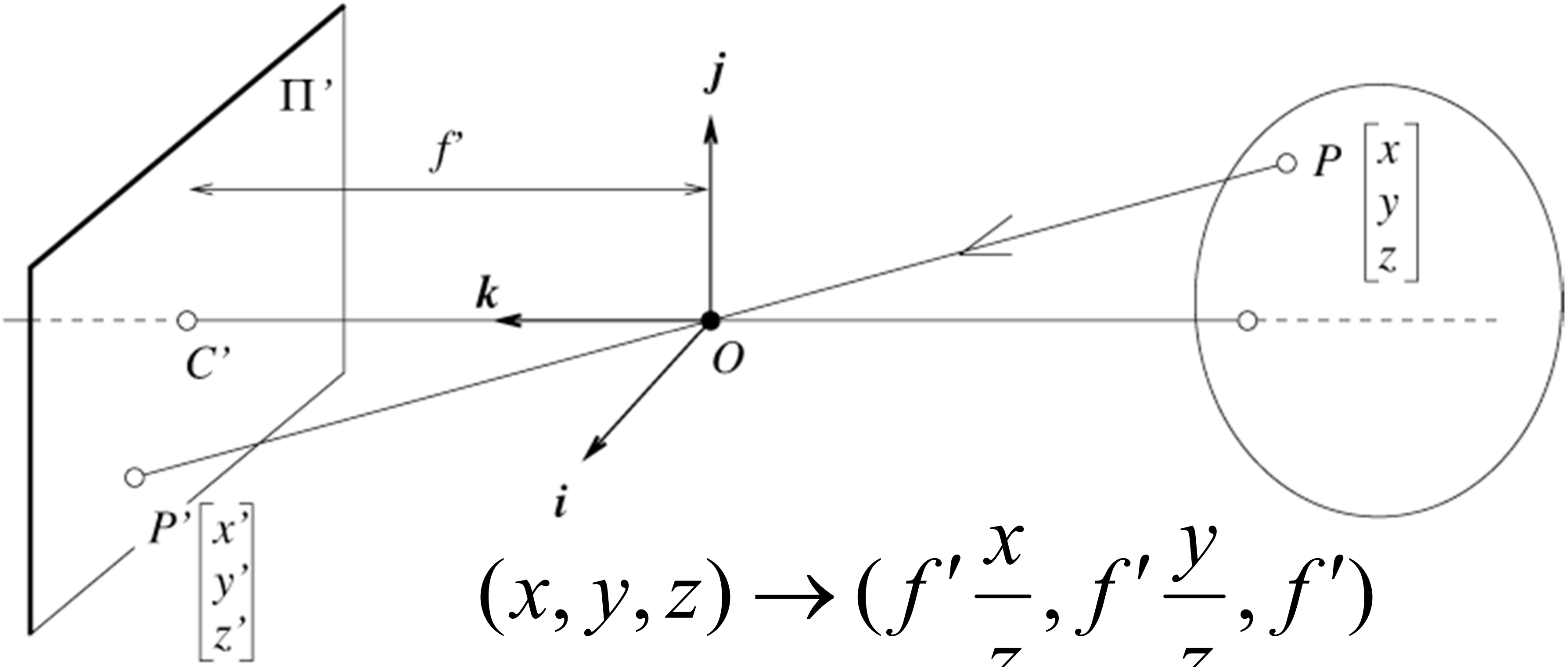
- Enrollment
  - Class remains full, must wait for some students to drop before adding more; possibly until next week
- Make up class: Sept 9, time and place to be announced
- Monday, Sept 5 is holiday; no class.

Review: Technical

- Image Formation: Geometry
- Pin hole camera model
- Real lenses
- Projection equations

The equation of projection

- Note: *k*-axis *towards* the camera (right handed coord system).



$$(x, y, z) \rightarrow (f'\frac{x}{z}, f'\frac{y}{z}, f')$$

Note: mapping is not invertible

Example: projection of a line

- How to determine equation of the projection of a line?
  - First need to define a 3-D line: One way is to define by two end points
  - Project the line
    - Could project the end points and determine 2-D line from them
    - Alternative: use a *parametric* form of 3-D line to derive *parametric* form of 2-D line
- 2-D line: $ax + by + c = 0$; or $x = at + x_0$, $y = bt + y_0$, t is a “parameter”
  - (line passes through (x0, y0), direction is given by (a,b))
- 3-D line: is given parametrically as $x = at + x_0$, $y = bt + y_0$, $z = ct + z_0$
  - Line passes through (x_0, y_0, z_0); direction is given by (a, b, c)
  - Can also be written as $(x-x_0)/a = (y-y_0)/b = (z-z_0)/c$
  - Sometimes also defined as intersection of two planes
- Map a point on the line, say (x,y,z) to image, say (x’,y’)

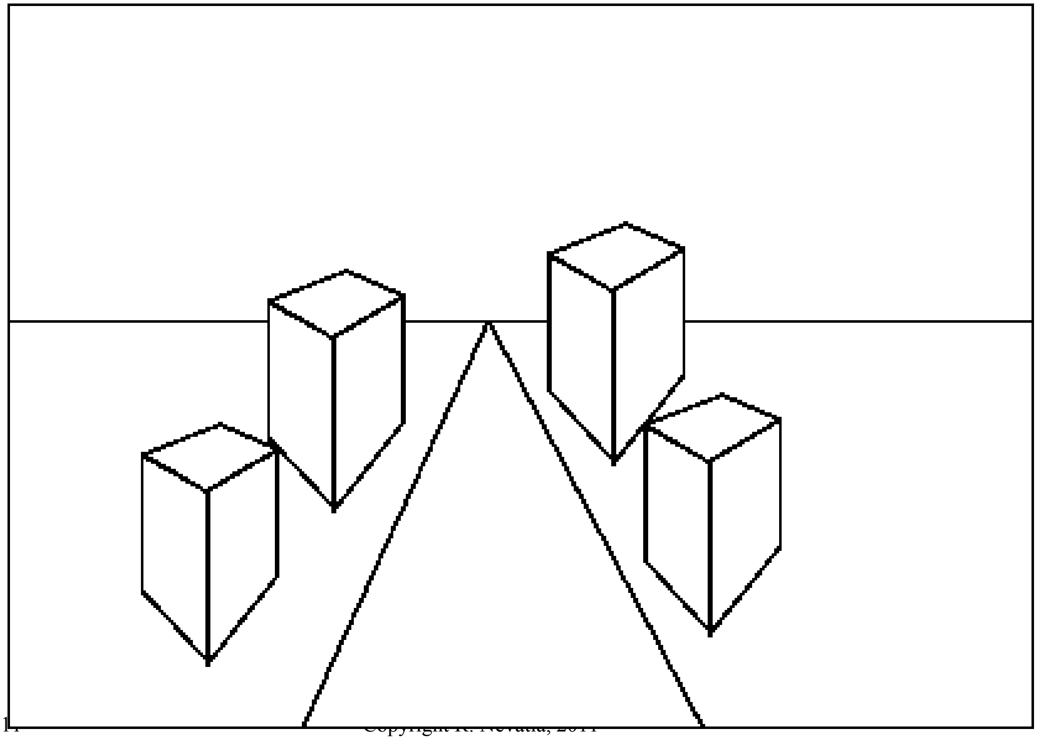
  use projection equations:

  $x' = f'*(x_0+at)/(z_0+ct)$, $y' = f'*(y_0+at)/(z_0+ct)$

We can show that the locus of the projected point is a straight line by eliminating parameter *t* from the above equations.

Projection of Parallel Lines

- Projections of parallel lines are not parallel
  - They meet in a common point called the *vanishing point*
    - Can be proved by letting t => infinity in projected line equation
  - The vanishing point depends on the *direction* of the set of parallel lines
- Sets of parallel lines on the same plane lead to *collinear* vanishing points.
  - The line is called the *horizon* for that plane



Homogeneous Coordinates

- Add an extra coordinate
  - $(x,y,z) \Rightarrow (x_h, y_h, z_h, w_h) = (wx, wy, wz, w)$, w is any constant (in the book, w is set to 1)
- Advantage: allows perspective transformation to be *linearized*, *i.e.* expressed as a matrix equation

$$\begin{vmatrix} x_h' \\ y_h' \\ w_h' \end{vmatrix} = \begin{vmatrix} f' & 0 & 0 & 0 \\ 0 & f' & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} x_h \\ y_h \\ z_h \\ w_h \end{vmatrix}$$

$$x_h' = f' x_h \,,\ y_h' = f' y_h \,,\ w_h' = z_h$$

$$x' = x_h' / w_h' = f' * x/z,\ y' = f' * y/z$$

  - *NOTE:* If image plane in front, replace f' by $-f'$ in the above equations.

Coordinate Systems

- Previous transformation matrix requires object coordinates to be expressed in the *camera* coordinate system (with origin at lens center)
  - This, in general, is not very convenient
- *Object* coordinate system
  - Aligned with some components of the object, *e.g.* the three sides of a rectangular solid
- *World* coordinate system
  - Chosen for global convenience, *e.g.* lines forming corner of a room , or earth coordinates (latitude, longitude, height)
- Coordinate transformations define relations between different coordinate systems

Rigid Transformations

- Translation of origin (3 parameters)
- Rotation (3 parameters)
  - – Euler angles, about the 3 axes (e.g. rotate about z-axes, then about the new y-axis, then about the new x-axis) but other rotation axes may be chosen
- Notation

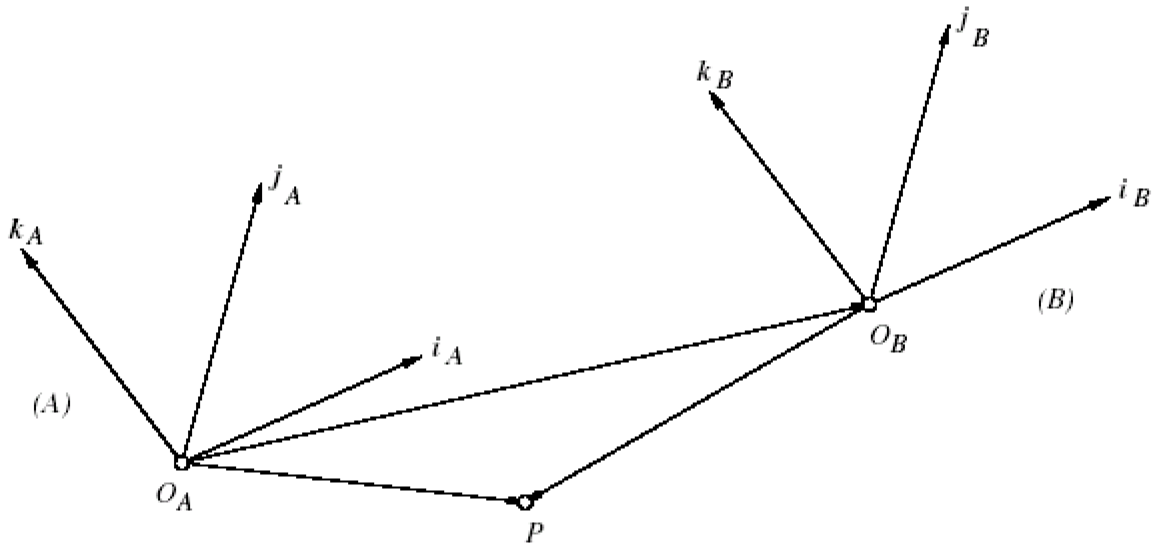
^{F}P Point *P* in Frame *F*

$$(A) = (O_A, \boldsymbol{i}_A, \boldsymbol{j}_A, \boldsymbol{k}_A)$$

$$(B) = (O_B, \boldsymbol{i}_B, \boldsymbol{j}_B, \boldsymbol{k}_B)$$

Translation

- figure 2.3 FP



- Assume vectors of two systems are parallel
  - Equations as in book, also in homogeneous coordinates

$$^{B}P = {}^{A}P + {}^{B}O_{A}$$

Translation in Homogeneous Coordinates

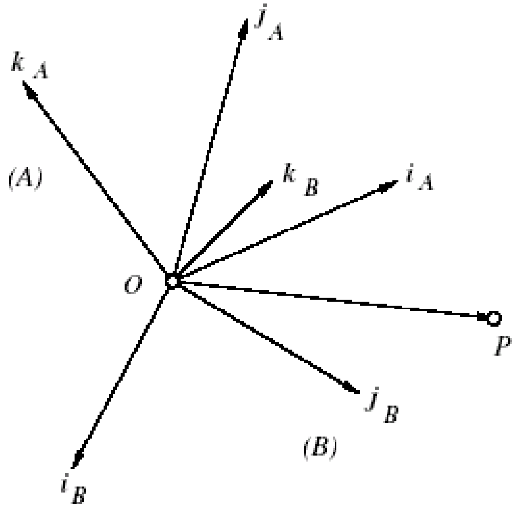
$$\begin{vmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{vmatrix} = \begin{vmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{vmatrix} + \begin{vmatrix} {}^{B}x_{OA} \\ {}^{B}y_{OA} \\ {}^{B}z_{OA} \end{vmatrix}$$

or $$\begin{vmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & {}^{B}x_{OA} \\ 0 & 1 & 0 & {}^{B}y_{OA} \\ 0 & 0 & 1 & {}^{B}z_{OA} \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \\ 1 \end{vmatrix}$$

or $$\begin{vmatrix} {}^{B}\boldsymbol{P} \\ 1 \end{vmatrix} = \begin{vmatrix} I & \boldsymbol{t} \\ 0^{T} & 1 \end{vmatrix} \begin{vmatrix} {}^{A}\boldsymbol{P} \\ 1 \end{vmatrix}$$

Rotation

- Figure 2.4 FP



- How to express rotation?
  - axis vectors of one system defined in the other also define the rotation matrix R

$${}^B_A\mathcal{R} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{i}_A \cdot \boldsymbol{i}_B & \boldsymbol{j}_A \cdot \boldsymbol{i}_B & \boldsymbol{k}_A \cdot \boldsymbol{i}_B \\ \boldsymbol{i}_A \cdot \boldsymbol{j}_B & \boldsymbol{j}_A \cdot \boldsymbol{j}_B & \boldsymbol{k}_A \cdot \boldsymbol{j}_B \\ \boldsymbol{i}_A \cdot \boldsymbol{k}_B & \boldsymbol{j}_A \cdot \boldsymbol{k}_B & \boldsymbol{k}_A \cdot \boldsymbol{k}_B \end{pmatrix} \qquad {}^B_A\mathcal{R} = \begin{pmatrix} {}^B\boldsymbol{i}_A & {}^B\boldsymbol{j}_A & {}^B\boldsymbol{k}_A \end{pmatrix} = \begin{pmatrix} {}^A\boldsymbol{i}_B^{\ T} \\ {}^A\boldsymbol{j}_B^{\ T} \\ {}^A\boldsymbol{k}_B^{\ T} \end{pmatrix}$$

$${}^BP = {}^B_A\mathcal{R}\,{}^AP \qquad {}^A_B\mathcal{R} = {}^B_A\mathcal{R}^T$$

Example

- Suppose that $\boldsymbol{k}_A = \boldsymbol{k}_B = \boldsymbol{k}$
- Vector $\boldsymbol{i}_B$ is obtained by rotating $\boldsymbol{i}_A$ by an angle θ in the counter-clockwise direction, then

$$\mathcal{R} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rigid Transformations

- Translation $$\begin{vmatrix} {}^{B}\boldsymbol{P} \\ 1 \end{vmatrix} = \begin{vmatrix} \mathrm{I} & \boldsymbol{t} \\ 0^{\mathrm{T}} & 1 \end{vmatrix} \begin{vmatrix} {}^{A}\boldsymbol{P} \\ 1 \end{vmatrix}$$

- Rotation $${}^{B}P = {}^{B}_{A}\mathcal{R}\,{}^{A}P$$

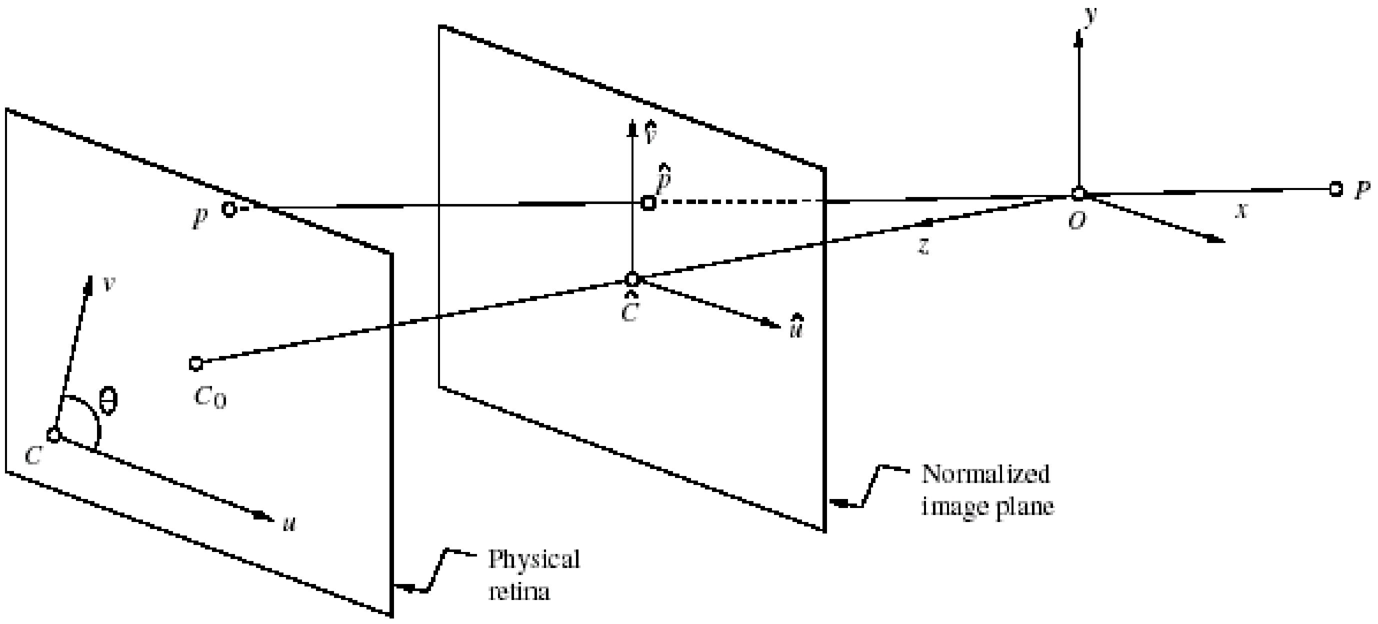
- Combine rotation and translation

$$\begin{pmatrix} {}^{B}P \\ 1 \end{pmatrix} = {}^{B}_{A}\mathcal{T} \begin{pmatrix} {}^{A}P \\ 1 \end{pmatrix}, \quad \text{where} \quad {}^{B}_{A}\mathcal{T} \overset{\text{def}}{=} \begin{pmatrix} {}^{B}_{A}\mathcal{R} & {}^{B}O_{A} \\ 0^{T} & 1 \end{pmatrix}$$

- Note: **t** = ${}^{B}O_{A}$, from object to camera coordinates, it may be more convenient to be given ${}^{A}O_{B}$ in which case **t**= -R ${}^{A}O_{B}$

Intrinsic Camera Parameters

- Figure 2.8



- Measurement in image coordinate system may be in “pixel” units (u,v), pixels may not be rectangular, origin of image coordinate system may not be at the center of *image* (projection of lens center), axis may be *skewed* .
- *Normalized* image plane: parallel to physical retina but unit distance from lens center

Intrinsic Camera Equations

- *Normalized* image plane: parallel to physical retina but unit distance from lens center, coordinate system as in fig 2.8

$$\begin{cases} \hat{u} = \dfrac{x}{z} \\ \hat{v} = \dfrac{y}{z} \end{cases} \iff \hat{p} = \frac{1}{z}(\mathrm{Id} \quad 0)\begin{pmatrix} P \\ 1 \end{pmatrix}$$

- Projection Equations for different u,v scales (2.10)

$$\begin{cases} u = kf\dfrac{x}{z}, \\ v = lf\dfrac{y}{z}. \end{cases}$$

- Now shift the origin (2.11)

$$\begin{cases} u = \alpha\dfrac{x}{z} + u_0, \\ v = \beta\dfrac{y}{z} + v_0. \end{cases}$$

Intrinsic Camera Equations

- With skewed axes (2.12)

$$\begin{cases} u = \alpha \dfrac{x}{z} - \alpha \cot\theta \dfrac{y}{z} + u_0, \\ v = \dfrac{\beta}{\sin\theta} \dfrac{y}{z} + v_0. \end{cases}$$

- All intrinsic parameters combined

$$\boldsymbol{p} = \mathcal{K}\hat{\boldsymbol{p}}, \quad \text{where} \quad \boldsymbol{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

$$\mathcal{K} \stackrel{\text{def}}{=} \begin{pmatrix} \alpha & -\alpha\cot\theta & u_0 \\ 0 & \dfrac{\beta}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Projection equation

$$\boldsymbol{p} = \frac{1}{z}\mathcal{M}\boldsymbol{P}, \quad \text{where} \quad \mathcal{M} \stackrel{\text{def}}{=} (\,\mathcal{K} \quad \boldsymbol{0}\,)$$

  - Note that division by z is present because ***p*** = (u, v, 1)^T ; if we allow ***p***= (u', v', w')^T ; we can just write ***p*** = M***P*, ** leaving the step of division to be performed only when non-homogeneous coordinates need to be computed.

Including *Extrinsic* Camera Parameters

- Relate camera frame,***C***, to world (object) frame, ***W***

$$\begin{pmatrix} {}^{C}P \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{C}_{W}\mathcal{R} & {}^{C}O_{W} \\ \mathbf{0}^{T} & 1 \end{pmatrix} \begin{pmatrix} {}^{W}P \\ 1 \end{pmatrix}$$

$$\boldsymbol{p} = \frac{1}{z}\mathcal{M}\boldsymbol{P}, \quad \text{where} \quad \mathcal{M} = \mathcal{K}\,(\mathcal{R} \quad \boldsymbol{t})$$

- Fully expanded, matrix *M* (3x4)

$$\mathcal{M} = \begin{pmatrix} \alpha\boldsymbol{r}_1^T - \alpha\cot\theta\,\boldsymbol{r}_2^T + u_0\boldsymbol{r}_3^T & \alpha t_x - \alpha\cot\theta\, t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0\boldsymbol{r}_3^T & \dfrac{\beta}{\sin\theta}t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix}$$

where $\boldsymbol{r}_1^T$, $\boldsymbol{r}_2^T$ and $\boldsymbol{r}_3^T$ denote the three rows of the matrix $\mathcal{R}$

Note M is 3 x4, first column includes vectors ($\boldsymbol{r}^t_1$, $\boldsymbol{r}^t_2$ and $\boldsymbol{r}^t_3$), *z* is not in the original world coordinate system (is $\boldsymbol{m}_3 \,.\, \boldsymbol{P}$)

- Equivalent to:

  $u = (\boldsymbol{m}_1 \,.\, \boldsymbol{P})/(\boldsymbol{m}_3 \,.\, \boldsymbol{P})$, $v = (\boldsymbol{m}_2 .\, \boldsymbol{P})/(\boldsymbol{m}_3 \,.\, \boldsymbol{P})$, where $\boldsymbol{m}_1$, $\boldsymbol{m}_2$ and $\boldsymbol{m}_3$ are transposes of three rows of *M*

Next Class

- Chapter 2, section 2.3
- Chapter 3, section 3.1