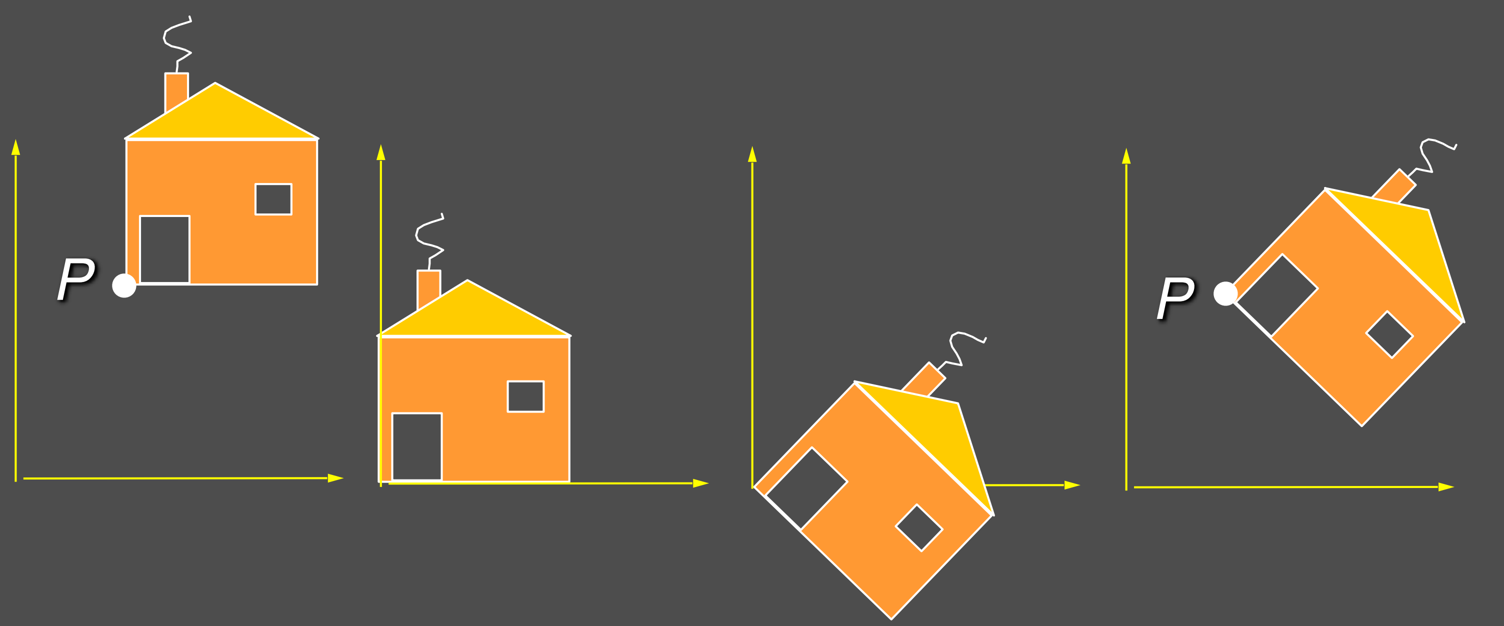
cs433-533
Transformation (cont)

Transformation Composition

- What operation rotates by θ around $P = (p_x, p_y)$?

  - Translate P to origin
  - Rotate around origin by θ
  - Translate back



Transformation Composition

$$T^{(-p_x,-p_y)} R^{\theta} T^{(p_x,p_y)} =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -p_x & -p_y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_x & p_y & 1 \end{bmatrix}$$

Let *Q* denote this matrix (computed once).

For every point *p* of the many points of the "house", we apply

p'=pQ

(read: The old corner *p* is transformed to the new corner *p'*)

Transformations Quiz

- What do these transformations do?

A: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

B: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

C: $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

D: $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$

E: $\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$

- And these homogeneous ones?

F: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ G: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$

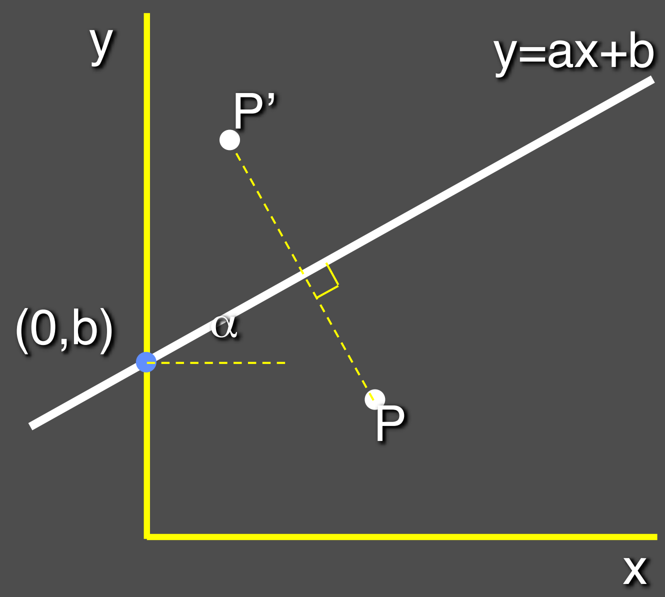
- How can one reflect a planar object through an arbitrary line in the plane?

- Can one rotate a planar object in the plane by reflection?

Matrix Form

- Why is it useful to use **Matrix** form to represent the **transformations** ?
- The answer depends if we think about CPU or about GPU
    - Interactive CG: We

Arbitrary Reflection



Shift by (0,-b)

Rotate by –α

$\alpha = \tan^{-1}(a)$

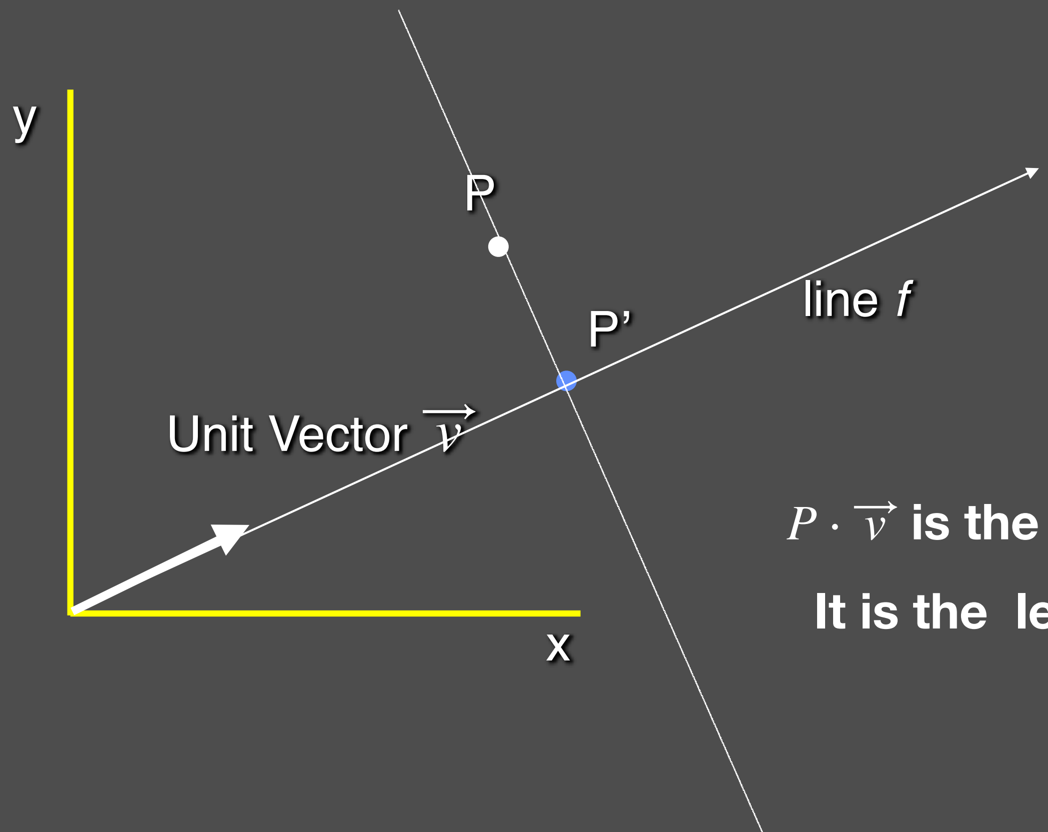
Reflect through x

Rotate by α

Shift by (0,b)

$$T^{(0,-b)} R^{-\alpha} \, \mathrm{Ref}^{x} \, R^{\alpha} T^{(0,b)}$$

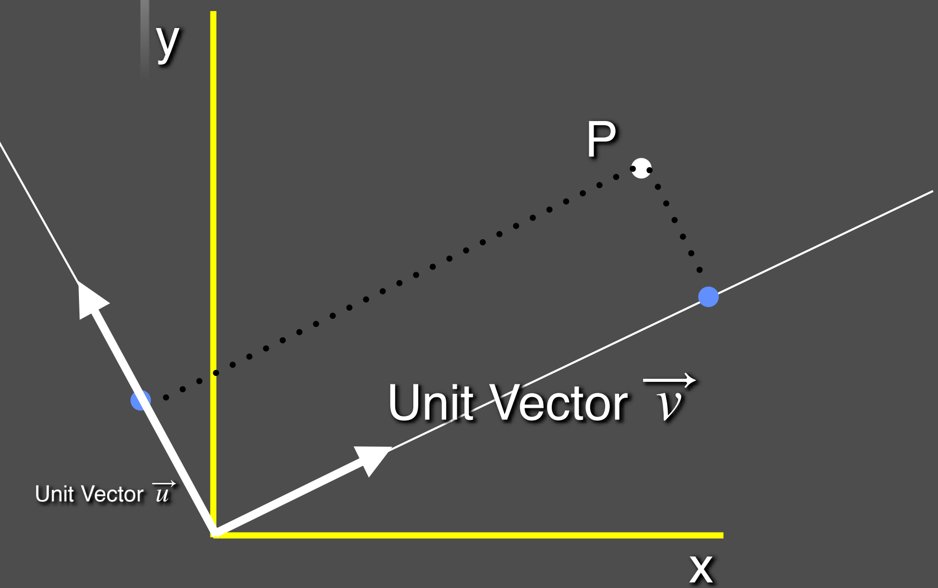
Another way to think about rotation



$P \cdot \vec{v}$ **is the projection of P on the line f**

It is the length from the origin to P',

$$R^{\theta} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$



Instead of rotating P CCW, lets tilt the camera CW

What are the coordinate of P in the new system
Let u be a unit vector orthogonal v.

u and v define an coordinates system (just like x and y. This is the camera coordinates system.

Pv is the first coordinate of P in the coordinator system v,u

Pu is the second coordinate of P in the new system

So $R^{-\theta} \cdot P$ are the coordinates of P, in the new system

3D Linear Transformations

- We can transform points in a 3D coordinate system by multiplying the point (a vector) by a matrix (the transformation), just like in 2D!

- The only difference is we will use 3x3 matrices A by **x** = (x,y,z), or A**x**, e.g. for scale and shear:

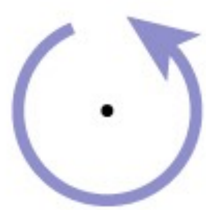
$$\text{scale}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

$$\text{shear-x}(d_y, d_z) = \begin{bmatrix} 1 & d_y & d_z \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

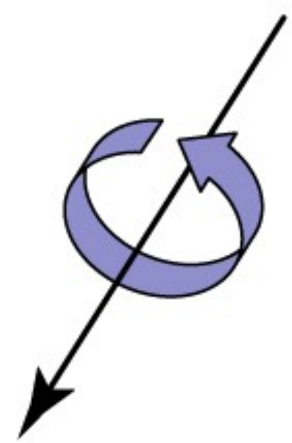
Rotations in 3D

- In 2D, a rotation is about a point
- In 3D, a rotation is about an axis

convention: positive rotation is CCW



2D



3D

convention: positive rotation is CCW when axis vector is pointing at you

Rotations about 3D Axes

- In 3D, we need to pick an axis to rotate about

$$\text{rotate-z}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

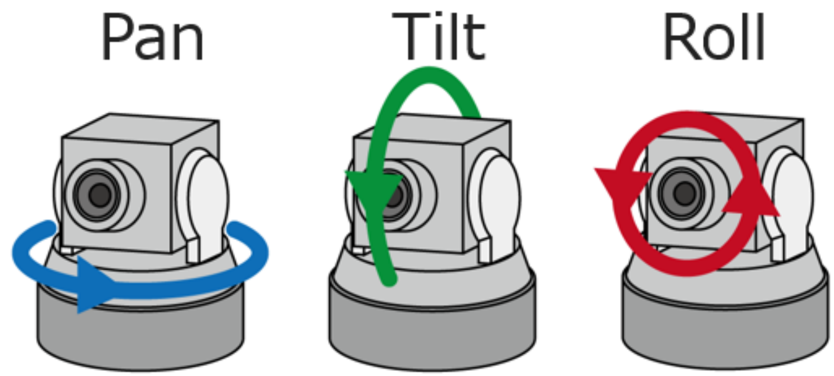
- And we can pick any of the three axes

$$\text{rotate-x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

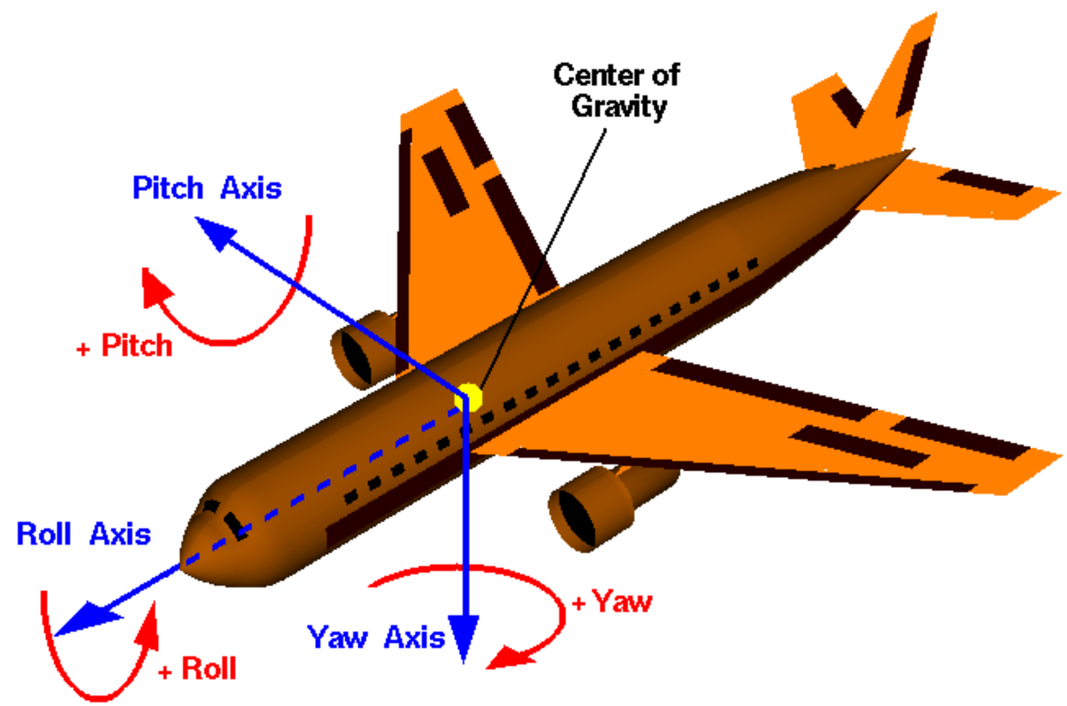
$$\text{rotate-y}(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}$$

Building Complex Rotations from Axis-Aligned Rotations

- Rotations about **x, y, z** are sometimes called **Euler angles**
- Build a combined rotation using matrix composition



Ishikawa Watanabe Laboratory



Wikipedia

Arbitrary Rotations

- To rotate about any axis: we change the coordinate space we are working in, using orthogonal matrices.
- Consider orthogonal matrix R_{uvw}, form by taking three orthogonal vectors **u**, **v**, and **w**:

Property of orthogonal vectors:

$$\mathbf{u}\cdot\mathbf{u} = \mathbf{v}\cdot\mathbf{v} = \mathbf{w}\cdot\mathbf{w} = 1$$
$$\mathbf{u}\cdot\mathbf{v} = \mathbf{v}\cdot\mathbf{w} = \mathbf{w}\cdot\mathbf{u} = 0$$

$$\mathrm{R}_{uvw} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

Arbitrary Rotations

- What happens when we apply R_{uvw} to any of the basis vectors, e.g.:

$$\mathbf{R}_{uvw}\mathbf{u} = \begin{bmatrix} \mathbf{u} \cdot \mathbf{u} \\ \mathbf{v} \cdot \mathbf{u} \\ \mathbf{w} \cdot \mathbf{u} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{x}$$

- But this means that if we apply R_{uvw}^{T} to the Cartesian coordinate vectors, e.g.:

$$\mathbf{R}_{uvw}^{\mathrm{T}}\mathbf{y} = \begin{bmatrix} x_u & x_v & x_w \\ y_u & y_v & y_w \\ z_u & z_v & z_w \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = \mathbf{v}$$

Arbitrary Rotations

- This means that if we want to rotation around an arbitrary axis, we need only to use a change of coordinates
- E.g. to rotate around a direction $\mathbf{w}$, we
  - Compute orthogonal directions $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$
  - Change the $\mathbf{uvw}$ axes to be **xyz** (R_{uvw})
  - Apply a rotate-z()
  - Finally, change the axes back to $\mathbf{uvw}$ ($R_{uvw}^{\mathbf{T}}$)

$$\underbrace{\begin{bmatrix} x_u & x_v & x_w \\ y_u & y_v & y_w \\ z_u & z_v & z_w \end{bmatrix}}_{R_{uvw}^{\mathbf{T}}} \underbrace{\begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{rotate-z()}} \underbrace{\begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{bmatrix}}_{R_{uvw}}$$

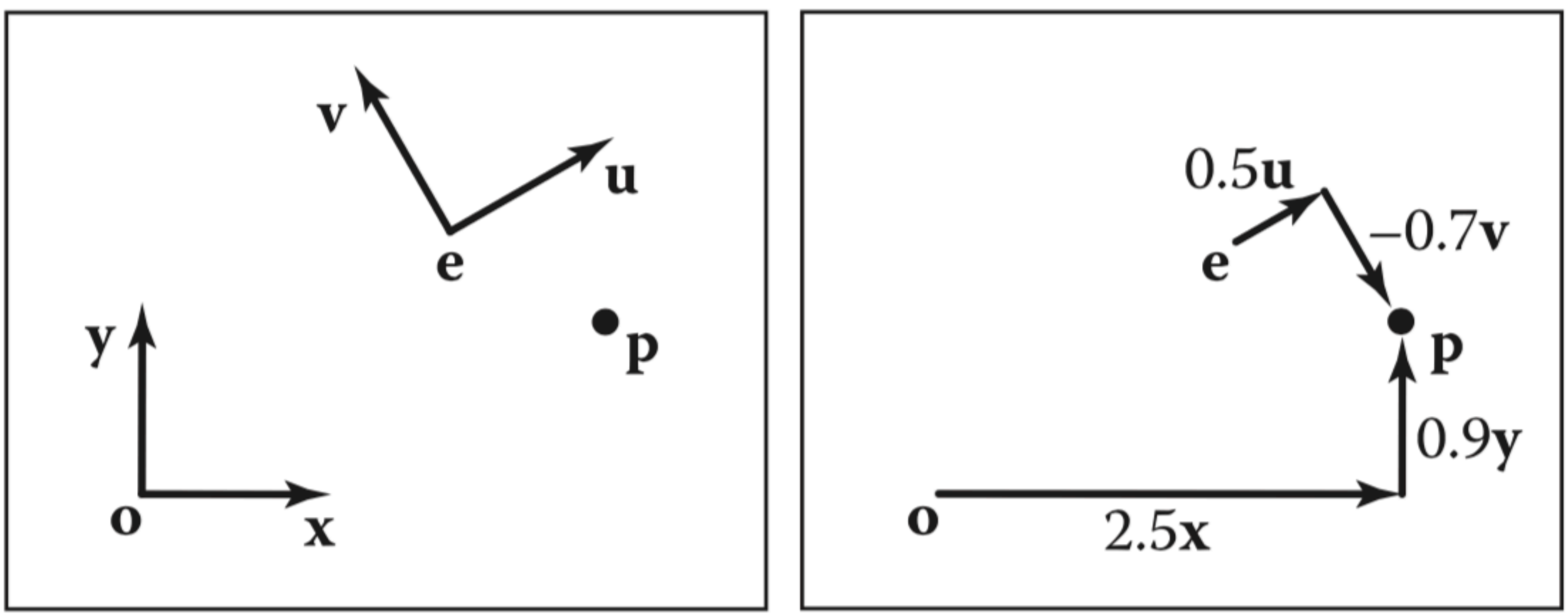
Coordinate Transformations

Coordinate Systems

- Points in space can be represented using an origin position and a set of orthogonal basis vectors:

$$\mathbf{p} = (x_p, y_p) \equiv \mathbf{0} + x_p\mathbf{x} + y_p\mathbf{y} \qquad \mathbf{p} = (u_p, v_p) \equiv \mathbf{e} + u_p\mathbf{u} + v_p\mathbf{v}$$

- Any point can be described in either coordinate system



Matrices for Converting Coordinate Systems

- (Remember: Rotating the world CCW is equivalent to rotating the coordinate system CW)

- Using homogenous coordinates and affine transformations, we can convert between coordinate systems:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_e \\ 0 & 1 & y_e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_u & x_v & 0 \\ y_u & y_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_e \\ y_u & y_v & y_e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

- More generally, any arbitrary coordinate system transform:

$$\mathbf{P}_{uv} = \begin{bmatrix} \mathbf{x}_{uv} & \mathbf{y}_{uv} & \mathbf{o}_{uv} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{xy}$$