

Image Formation and Cameras

Computer Vision I
CSE 252A
Lecture 5

CS252A, Fall 2018

Computer Vision I

Announcements

- HW1 posted
- Wait list is cleared.
- About readings: You are responsible for the topics covered in class. The readings cover those topic in greater depth and breadth, but what you're responsible for during the final exam is what's covered in class and on homeworks.

CS252A, Fall 2018

Computer Vision I

Mapping from a Plane to a Plane under Perspective is given by a projective transform H

$x' = Hx$ H is a 3×3 matrix,
 x, x' are 3×1 vectors of
homogenous coordinates

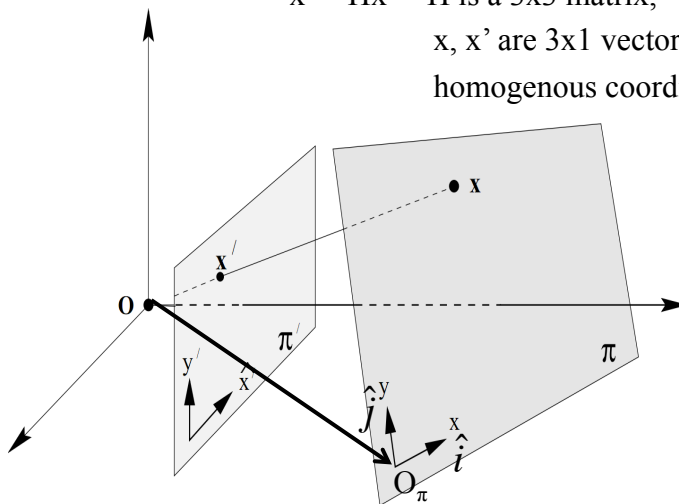


Figure borrowed from Hartley and Zisserman "Multiple View Geometry in computer vision"

CS252A, Fall 2018

Computer Vision I

Estimating a Homography from 4 or more matching points

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This is 3 equations. Noting that $\lambda = h_7x + h_8y + h_9$, we can eliminate λ , to get two equations in $h_1 \dots h_9$

$$\begin{bmatrix} 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \\ x & y & 1 & 0 & 0 & 0 & -xx' & -x'y & -x' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

CS252A, Fall 2018

Computer Vision I

Direct Linear Transform

$$\begin{bmatrix} 0 & 0 & 0 & -x_1 & -y_1 & -1 & y_1'x_1 & y_1'y_1 & y_1' \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & y_2'x_2 & y_2'y_2 & y_2' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

$$\mathbf{A}\mathbf{h} = \mathbf{0}$$

- Two rows for each point in a pair of images
- Four points, 8 rows
- \mathbf{h} is the Eigenvector corresponding to the zero (smallest) eigenvalue of \mathbf{A} .

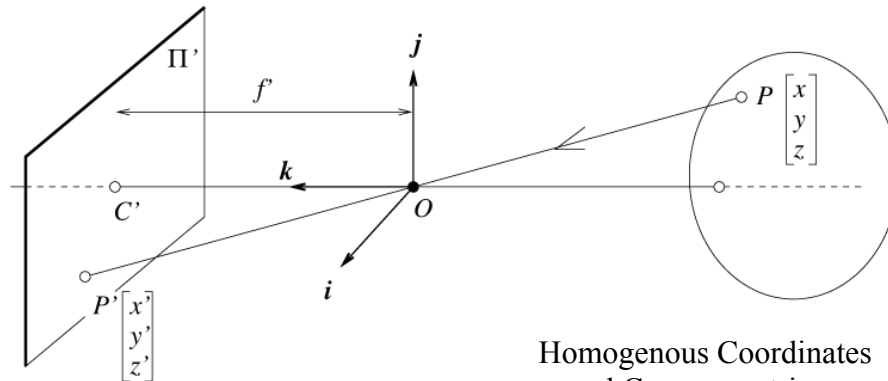
Computing H using SVD

- We can calculate the non-trivial solution to the equation $\mathbf{A}\mathbf{h} = \mathbf{0}$ by singular value decomposition

$$\text{Svd}(\mathbf{A}) = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

- \mathbf{h} is given by the singular vector corresponding to the smallest singular value, which is the last column of \mathbf{V} .
- Note: The SVD algorithm returns the singular values sorted from largest to smallest as the diagonal of \mathbf{S} .
- Reconstruct the matrix \mathbf{H} from the vector \mathbf{h}

Perspective Projection in a convenient coordinate system



Cartesian coordinates

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

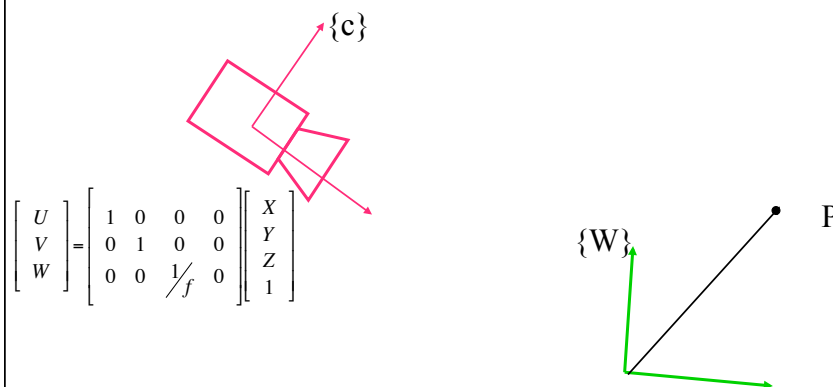
Homogenous Coordinates
and Camera matrix

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

CS252A, Fall 2018

Computer Vision I

What if camera coordinate system differs from object coordinate system

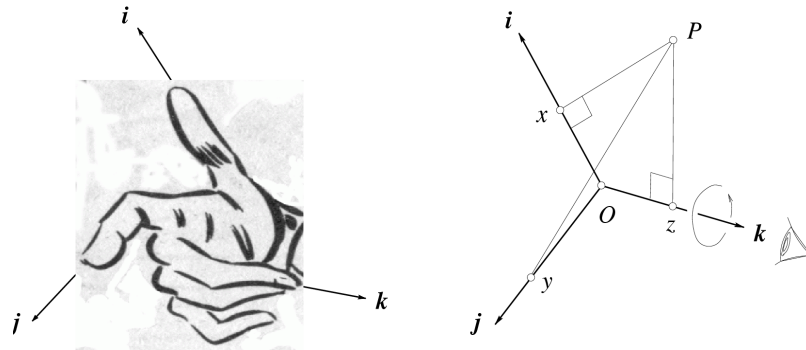


$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

CS252A, Fall 2018

Computer Vision I

Euclidean Coordinate Systems

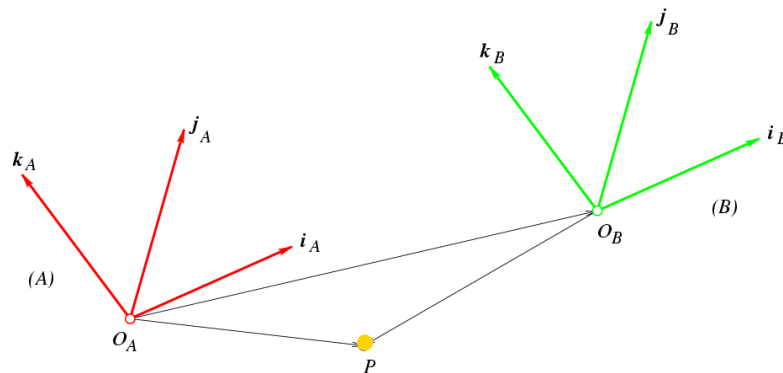


$$\begin{cases} x = \overrightarrow{OP} \cdot \mathbf{i} \\ y = \overrightarrow{OP} \cdot \mathbf{j} \\ z = \overrightarrow{OP} \cdot \mathbf{k} \end{cases} \Leftrightarrow \overrightarrow{OP} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \Leftrightarrow \mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

CS252A, Fall 2018

Computer Vision I

Coordinate Changes: Pure Translations

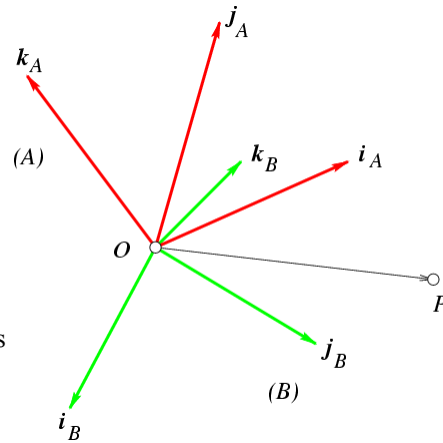


$$\overrightarrow{O_B P} = \overrightarrow{O_B O_A} + \overrightarrow{O_A P} \quad {}^B P = {}^A P + {}^B O_A$$

CS252A, Fall 2018

Computer Vision I

Rotation Matrix



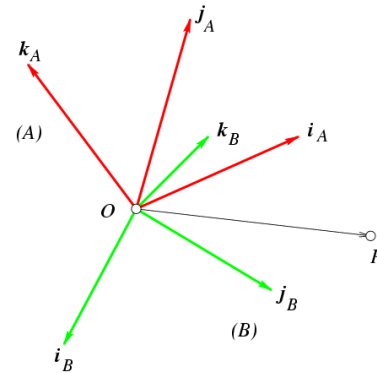
Dot Products between all pairs
of coordinate axis of both
systems

$${}^B_A R = \begin{bmatrix} \mathbf{i}_A \cdot \mathbf{i}_B & \mathbf{j}_A \cdot \mathbf{i}_B & \mathbf{k}_A \cdot \mathbf{i}_B \\ \mathbf{i}_A \cdot \mathbf{j}_B & \mathbf{j}_A \cdot \mathbf{j}_B & \mathbf{k}_A \cdot \mathbf{j}_B \\ \mathbf{i}_A \cdot \mathbf{k}_B & \mathbf{j}_A \cdot \mathbf{k}_B & \mathbf{k}_A \cdot \mathbf{k}_B \end{bmatrix} = \begin{bmatrix} \mathbf{i}_B & \mathbf{j}_B & \mathbf{k}_B \end{bmatrix}^T \begin{bmatrix} \mathbf{i}_A & \mathbf{j}_A & \mathbf{k}_A \end{bmatrix}$$

CS252A, Fall 2018

Computer Vision I

Coordinate Changes: Pure Rotations

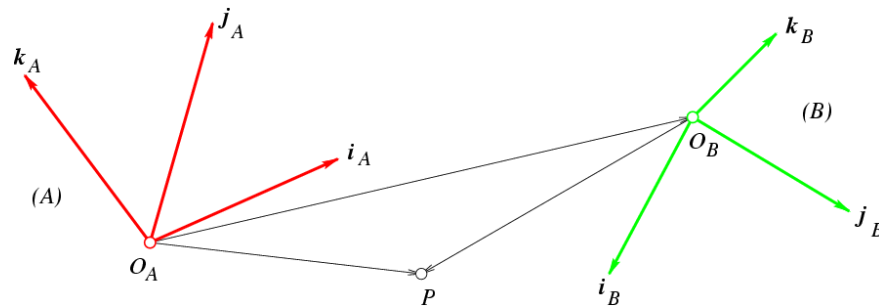


$$\begin{aligned} \overline{OP} &= \begin{bmatrix} \mathbf{i}_A & \mathbf{j}_A & \mathbf{k}_A \end{bmatrix} \begin{bmatrix} {}^A x \\ {}^A y \\ {}^A z \end{bmatrix} = \begin{bmatrix} \mathbf{i}_B & \mathbf{j}_B & \mathbf{k}_B \end{bmatrix} \begin{bmatrix} {}^B x \\ {}^B y \\ {}^B z \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{i}_A & \mathbf{j}_A & \mathbf{k}_A \end{bmatrix} {}^A P = \begin{bmatrix} \mathbf{i}_B & \mathbf{j}_B & \mathbf{k}_B \end{bmatrix} {}^B P \\ \Rightarrow {}^B P &= \begin{bmatrix} \mathbf{i}_B & \mathbf{j}_B & \mathbf{k}_B \end{bmatrix}^T \begin{bmatrix} \mathbf{i}_A & \mathbf{j}_A & \mathbf{k}_A \end{bmatrix} {}^A P = {}^B_A R {}^A P \end{aligned}$$

CS252A, Fall 2018

Computer Vision I

Coordinate Changes: Rigid Transformations



$${}^B P = {}^B_A R {}^A P + {}^B O_A$$

CS252A, Fall 2018

Computer Vision I

A convenient notation

$${}^B P = {}^B_A R {}^A P + {}^B O_A$$

- Points, vectors: ${}^A P_1$
 - Leading superscript indicates the coordinate system that the coordinates are with respect to
 - Subscript – an identifier
- To add vectors, coordinate systems must agree
- Rotation Matrices ${}^B_A R$
 - Lower left (Going from this system)
 - Upper left (Going to this system)
- To rotate a vector, point's coordinate system must agree with lower left of rotation matrix

CS252A, Fall 2018

Computer Vision I

More about rotations matrices

A rotation matrix R has the following properties:

- Its inverse is equal to its transpose $R^{-1} = R^T$ or $R^T R = I$
- Its determinant is equal to 1 $\det(R)=1$
- Rows (or columns) of R form a right-handed orthonormal coordinate system.
- Even though a rotation matrix is 3x3 with nine numbers, it only has three degrees of freedom, it can be parameterized with three numbers.
- There are many ways to parameterize rotation matrices.

Some points about SO(n)

- $SO(n) = \{ R \in \mathbb{R}^{n \times n} : R^T R = I, \det(R) = 1 \}$
 - $SO(2)$: rotation matrices in plane \mathbb{R}^2
 - $SO(3)$: rotation matrices in 3-space \mathbb{R}^3
- $R^{-1} = R^T$
- Forms a Group under matrix product operation:
 - Identity
 - Inverse
 - Associative
 - Closure
- Bounded $R_{i,j} \in [-1, +1]$
- Does not form a vector space.
- Not commutative!
- Manifold of dimension $n(n-1)/2$
 - $\dim(SO(2)) = 1$
 - $\dim(SO(3)) = 3$

Parameterizations of SO(3)

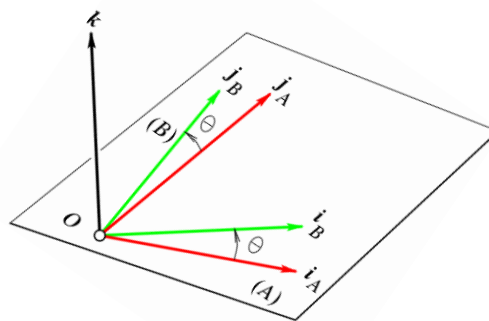
Even though a rotation matrix is 3x3 with nine numbers, it only has three degrees of freedom, it can be parameterized with three numbers. There are many parameterizations.

- 3-D manifold, so between 3 parameters and $2n+1$ parameters (Whitney's Embedding Thm.)
 - Roll-Pitch-Yaw
 - Euler Angles
 - Axis Angle (Rodrigues formula)
 - Cayley's formula
 - Matrix Exponential
 - Quaternions (four parameters + one constraint)

CS252A, Fall 2018

Computer Vision I

Rigid Transformations: Rotation about the **k** Axis



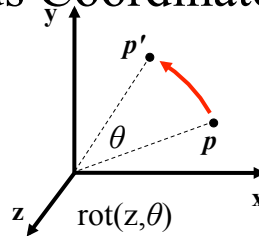
$${}^F P' = \mathcal{R}^F P, \quad \text{where} \quad \mathcal{R} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \text{rot}(\mathbf{k}, \theta)$$

CS252A, Fall 2018

Computer Vision I

Rotation: Homogenous Coordinates

- About z axis



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

CS252A, Fall 2018

Computer Vision I

Rotation

- About x axis:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- About y axis:

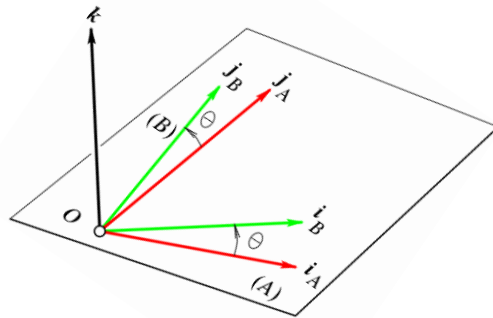
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

CS252A, Fall 2018

Computer Vision I

Roll-Pitch-Yaw

$$R = \text{rot}(\hat{i}, \alpha) \text{rot}(\hat{j}, \beta) \text{rot}(\hat{k}, \varphi)$$



Euler Angles

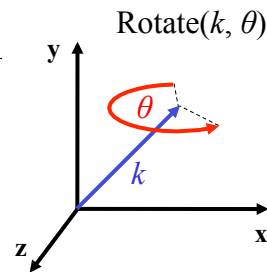
$$R = \text{rot}(\hat{k}'', \alpha) \text{rot}(\hat{j}', \beta) \text{rot}(\hat{k}, \varphi)$$

CS252A, Fall 2018

Computer Vision I

Rotation

- Rotate about $k = (k_x, k_y, k_z)$ by angle θ , a unit vector for an arbitrary axis (Rodrigues Formula)



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} k_x k_x (1-c) + c & k_x k_y (1-c) - k_z s & k_x k_z (1-c) + k_y s & 0 \\ k_y k_x (1-c) + k_z s & k_y k_y (1-c) + c & k_y k_z (1-c) - k_x s & 0 \\ k_z k_x (1-c) - k_y s & k_z k_y (1-c) - k_x s & k_z k_z (1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where $c = \cos \theta$ & $s = \sin \theta$

CS252A, Fall 2018

Computer Vision I

Quaternions

$$q = (a, \alpha)$$

q is a quaternion (generalization of imaginary numbers)
 $a \in \mathbb{R}$ is its real part
 $\alpha \in \mathbb{R}^3$ is its imaginary part.

Operations on quaternions:

- Sum of quaternions: $(a, \alpha) + (b, \beta) = ((a+b), (\alpha+\beta))$
- Multiplication by a scalar: $\lambda (a, \alpha) = (\lambda a, \lambda \alpha)$
- Quaternion product:
 $(a, \alpha) (b, \beta) = ((a b - \alpha \cdot \beta), (a \beta + b \alpha + \alpha \times \beta))$
- Conjugate: $q = (a, \alpha) \quad q' = (a, -\alpha)$
- Norm: $|q|^2 = q q' = q' q = a^2 + |\alpha|^2$

CS252A, Fall 2018

Computer Vision I

Unit Quaternions and Rotations

- Let R denote the rotation of angle θ about the unit vector u .
- Define unit quaternion $q = (\cos \theta/2, \sin \theta/2 u)$.
- Note $|q| = 1$ (i.e., q lies on unit sphere for any u and θ).
- Then for any vector α ,
 $R \alpha = \text{imaginary}(q \alpha^* q')$
 where $\alpha^* = (0, \alpha)$
- q and $-q$ define the same rotation matrix.

If $q = (a, (b, c, d)^T)$ is a unit quaternion, the corresponding rotation matrix is:

$$\mathcal{R} = \begin{pmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & a^2 - b^2 + c^2 - d^2 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & a^2 - b^2 - c^2 + d^2 \end{pmatrix}$$

CS252A, Fall 2018

Computer Vision I

Interlude: Multiplying Block Matrix

Block Matrix Multiplication

Given

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Where A_{ij} and B_{ij} are themselves matrices (blocks) of appropriate dimension.

What is AB ?

$$AB = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

CS252A, Fall 2018

Computer Vision I

Homogeneous Representation of Rigid Transformations

$$\begin{bmatrix} {}^B P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^B R {}^A P + {}^B O_A \\ 1 \end{bmatrix} = \begin{bmatrix} {}^B R & {}^B O_A \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} {}^A P \\ 1 \end{bmatrix} = {}^B T_A \begin{bmatrix} {}^A P \\ 1 \end{bmatrix}$$

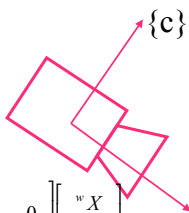
Rigid transformation represented by 4 by 4 Matrix

$${}^B T_A = \begin{bmatrix} {}^B R & {}^B O_A \\ \mathbf{0}^T & 1 \end{bmatrix}$$

CS252A, Fall 2018

Computer Vision I

What if camera coordinate system differs from object coordinate system



$${}^cT_w = \begin{bmatrix} {}^cR_w & {}^cO_w \\ \mathbf{0}^T & 1 \end{bmatrix}$$

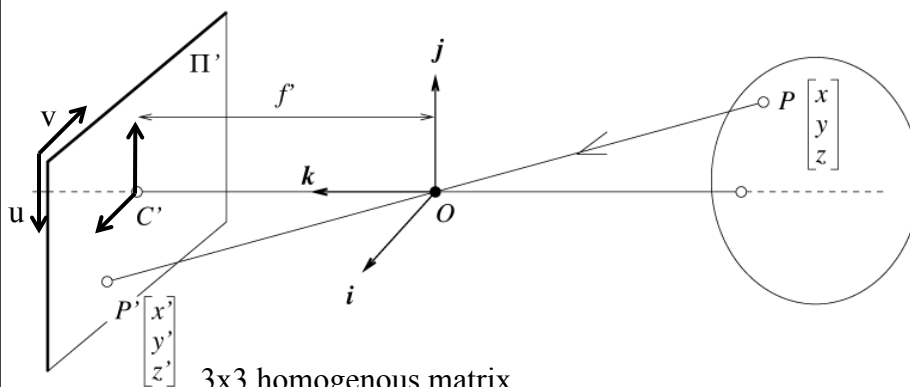
$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} {}^wX \\ {}^wY \\ {}^wZ \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} {}^cR_w & {}^cO_w \\ \mathbf{0}^T & 1 \end{bmatrix} {}^wP$$

CS252A, Fall 2018

Computer Vision I

Intrinsic parameters



3x3 homogenous matrix

Focal length: f

Principal Point: C'

Units (e.g. pixels)

Orientation and position of image coordinate system

Pixel Aspect ratio

CS252A, Fall 2018

Computer Vision I

Camera parameters

- Extrinsic Parameters: Since camera may not be at the origin, there is a rigid transformation between the world coordinates and the camera coordinates
- Intrinsic parameters: Since scene units (e.g., cm) differ image units (e.g., pixels) and coordinate system may not be centered in image, we capture that with a 3x3 transformation comprised of focal length, principal point, pixel aspect ratio, angle between axes, etc.

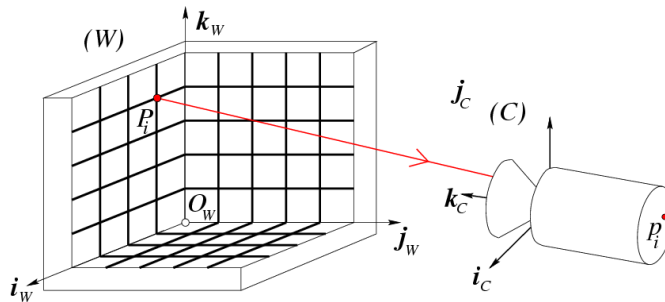
$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} \text{Transformation} \\ \text{represented by} \\ \text{intrinsic parameters} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \text{Rigid Transformation} \\ \text{represented by} \\ \text{extrinsic parameters} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

3×3
 4×4
 cT_w

CS252A, Fall 2018

Computer Vision I

Camera Calibration



Given n points P_1, \dots, P_n with *known* positions and their images p_1, \dots, p_n , estimate intrinsic and extrinsic camera parameters

- See Text book for how to do it.
- Camera Calibration Toolbox for Matlab (Bouguet)
http://www.vision.caltech.edu/bouguetj/calib_doc/
- OpenCV

CS252A, Fall 2018

Computer Vision I