Lecture 15

Rotations &

Homogeneous Coordinates

Slides courtesy Mike Langer

COMP 558 Overview

Part 1 : 2D Vision

RGB

Image filtering

Edge detection

Least Squares Estimation

Robust Estimation: Hough transform & RANSAC

Features 1: corners

Image Registration: the Lucas-Kanade method

Scale space

Histogram-based Tracking

Features 2: SIFT

Part 2 : 3D Vision

Perspective: projection, translation, vanishing points

Rotation, Homogeneous coordinates

Camera intrinsics and extrinsics

Least Squares methods (SVD)

Camera Calibration

Homographies & rectification

Stereo and Epipolar Geometry 1 & 2

Stereo correspondence

Features 3: CNN's

Object classification and detection

Segmentation (time permitting)

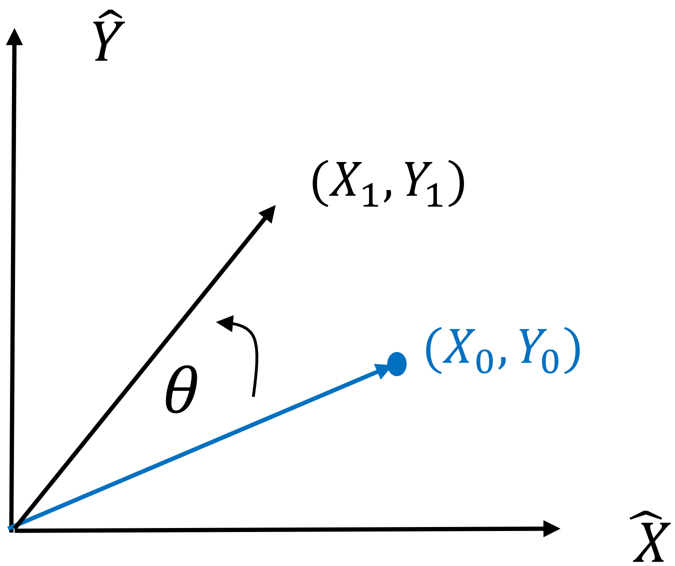
Cameras and Photography

RGBD Cameras

Rotations

- 2D rotations
- 3D rotations + projection (continuous)
- 3D rotations (discrete)
- review of cross product (left vs. right hand coordinates)

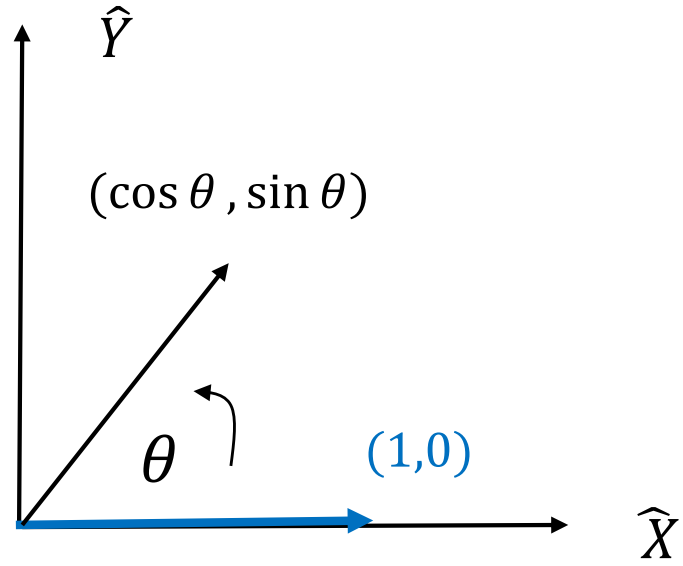
2D Rotation (discrete)

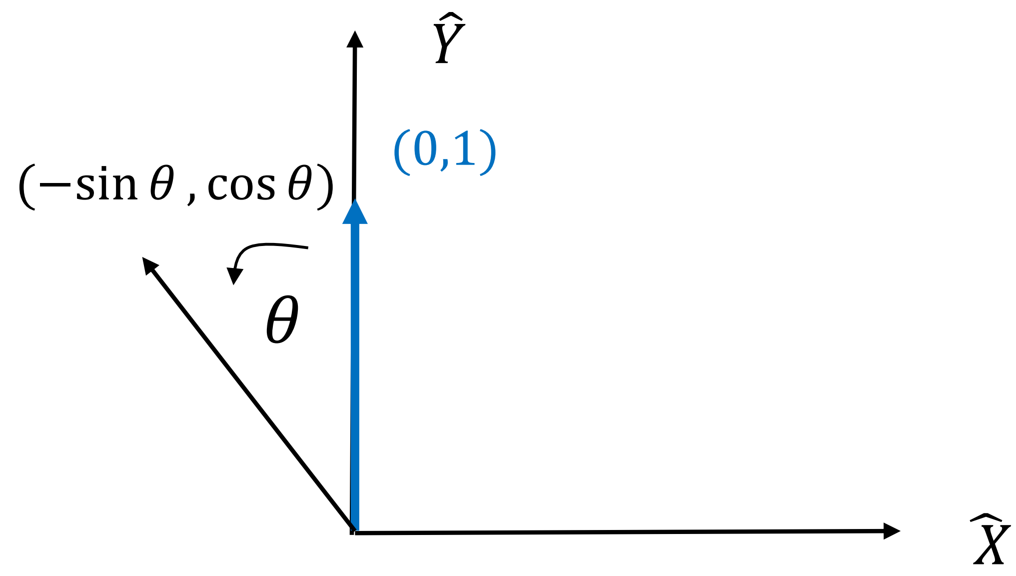


Update Nov. 25: in 2D, it doesn't matter if Y is pointing up or down. As long as a 90 degree rotation takes the unit x vector to the unit y vector and it takes the unit y vector to –x, we're fine. So I have not changed this figure or the following 2D rotation figures.

$$\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix}$$

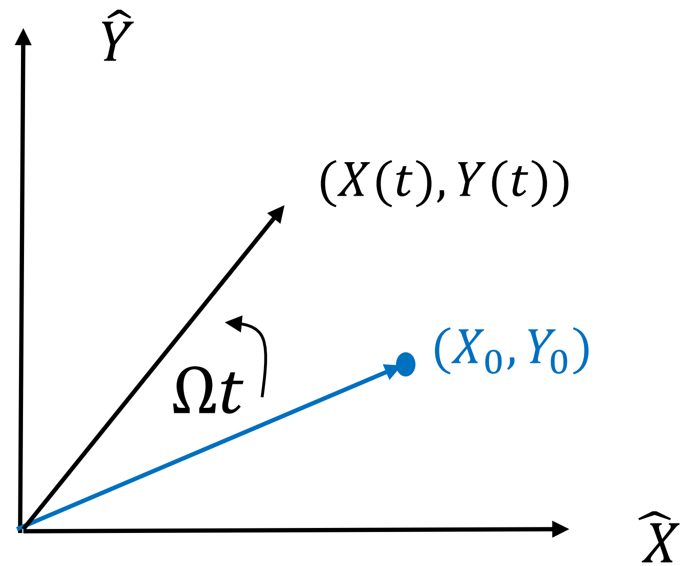
2D Rotation (discrete)





$$\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix}$$

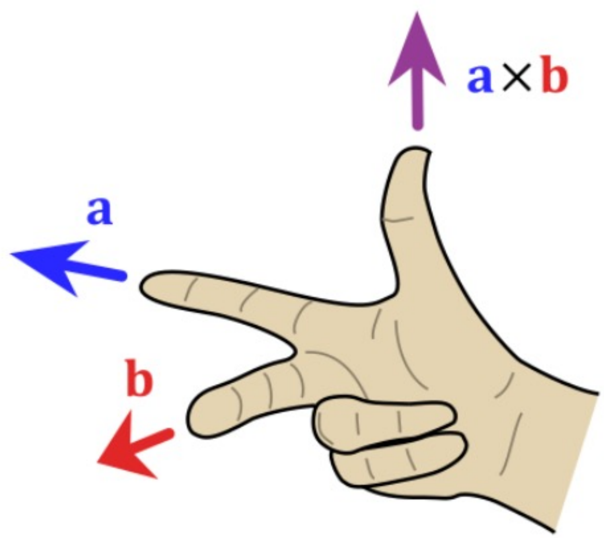
2D Rotation (continuous)



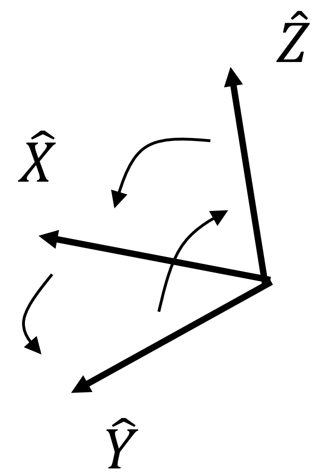
$$\begin{bmatrix} X(t) \\ Y(t) \end{bmatrix} = \begin{bmatrix} \cos(\Omega t) & -\sin(\Omega t) \\ \sin(\Omega t) & \cos(\Omega t) \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix}$$

where Ω is angular velocity (degrees or radians per unit time)

Cross Product



"Right hand"



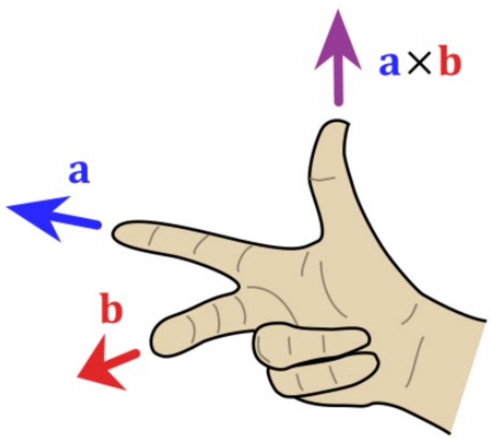
$$\hat{X} \times \hat{Y} = \hat{Z}$$

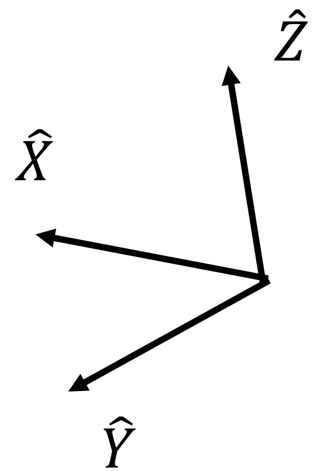
$$\hat{Y} \times \hat{Z} = \hat{X}$$

$$\hat{Z} \times \hat{X} = \hat{Y}$$

Cross Product

"Right hand"





$$\hat{X} \times \hat{Y} = \hat{Z}$$

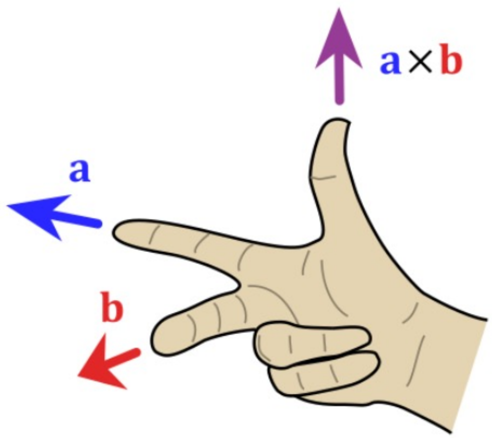
$$\hat{Y} \times \hat{Z} = \hat{X}$$

$$\hat{Z} \times \hat{X} = \hat{Y}$$

$$\left(a_X\hat{X} + a_Y\hat{Y} + a_Z\hat{Z}\right) \times \left(b_X\hat{X} + b_Y\hat{Y} + b_Z\hat{Z}\right) = \ ?$$

Cross Product

"Right hand"



Verify for yourself that

$\boldsymbol{a} \times \boldsymbol{a}$ is the 0 vector

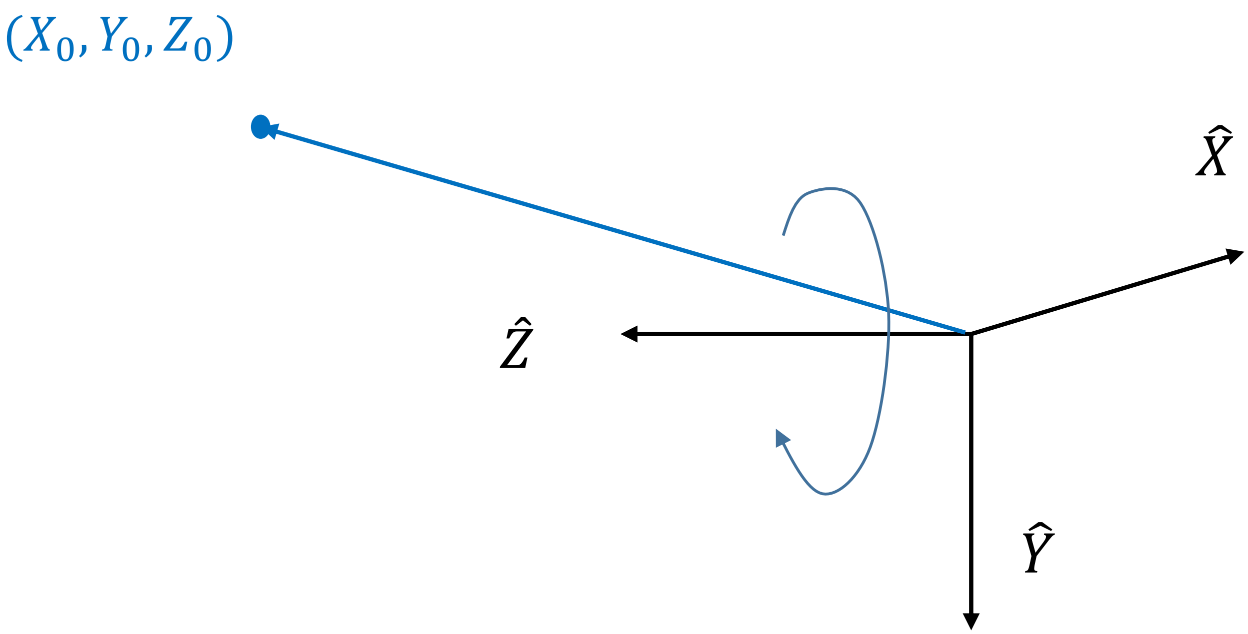
$\boldsymbol{a} \times \boldsymbol{b}$ is orthogonal to both $\boldsymbol{a}$ and $\boldsymbol{b}$

Cross Product

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} 0 & -a_Z & a_Y \\ a_Z & 0 & -a_X \\ -a_Y & a_X & 0 \end{bmatrix} \begin{bmatrix} b_X \\ b_Y \\ b_Z \end{bmatrix}$$

We will often write $\boldsymbol{a} \times \boldsymbol{b}$ as $[\boldsymbol{a}]_\times \boldsymbol{b}$. This treats this cross product as a linear transformation defined by $\boldsymbol{a}$ and applied to vector $\boldsymbol{b}$.

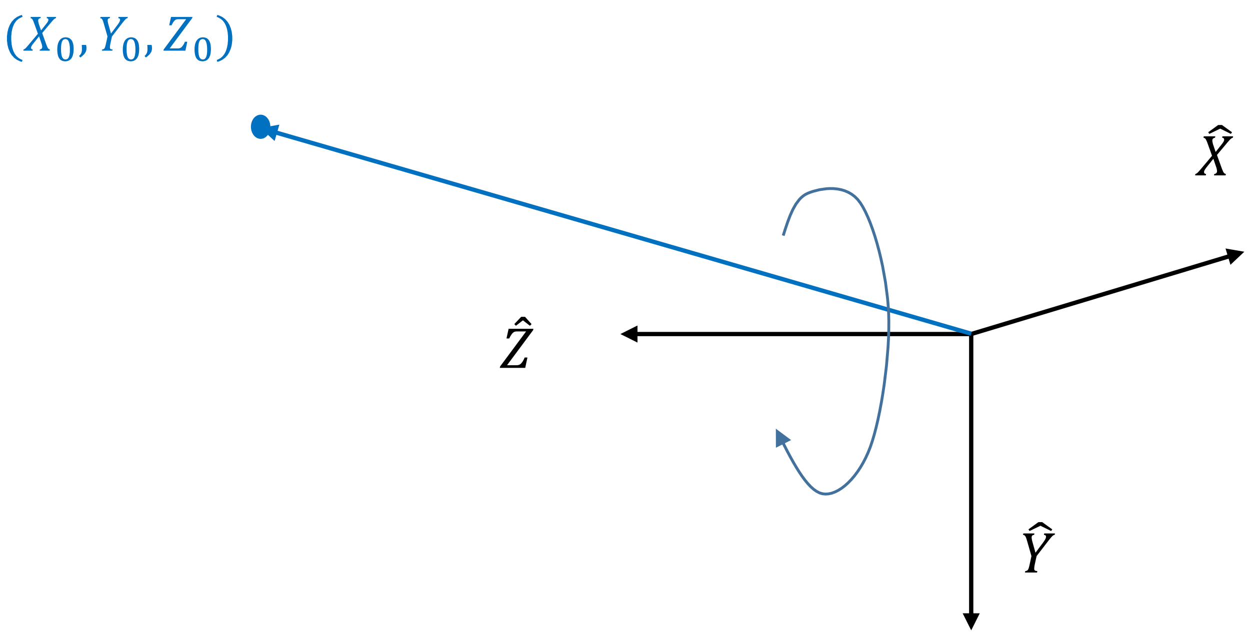
3D Camera Rotation (Z axis – “roll”)



When the camera rotates about the Z axis, what motion does the camera see?

$$\begin{bmatrix} X(t) \\ Y(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} \cos(\Omega t) & -\sin(\Omega t) & 0 \\ \sin(\Omega t) & \cos(\Omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

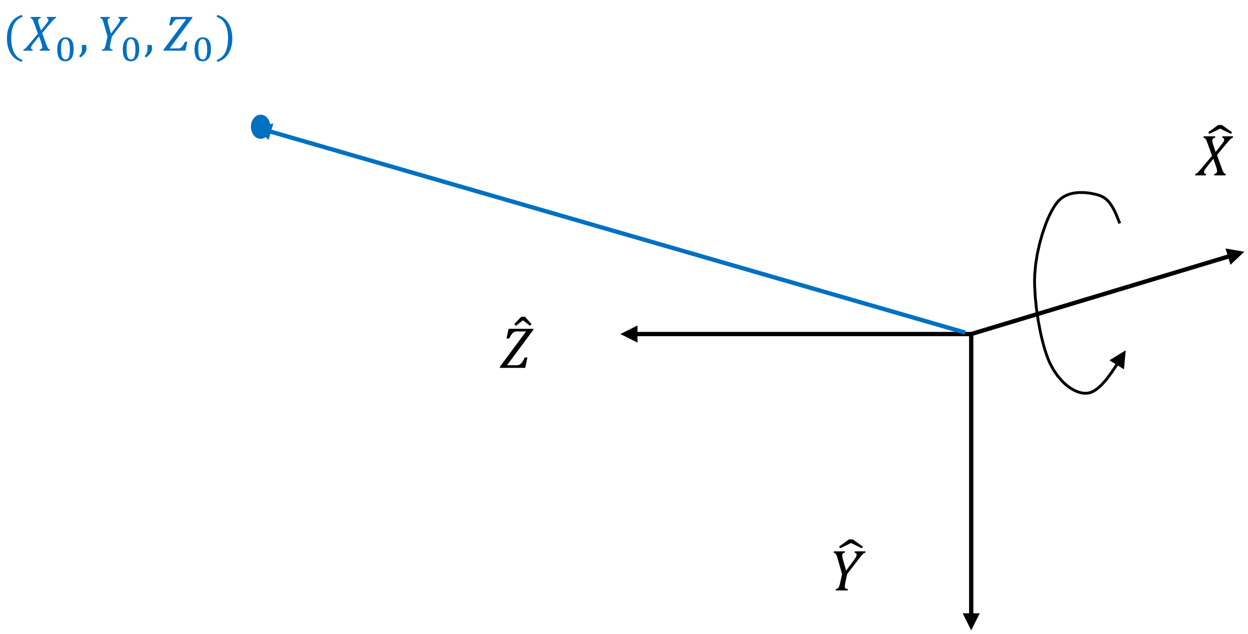
3D Camera Rotation (Z axis – “roll”)



The camera sees the same motion as when it is static and the scene rotates about the Z axis with the opposite velocity.

$$\begin{bmatrix} X(t) \\ Y(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} \cos(\Omega t) & -\sin(\Omega t) & 0 \\ \sin(\Omega t) & \cos(\Omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

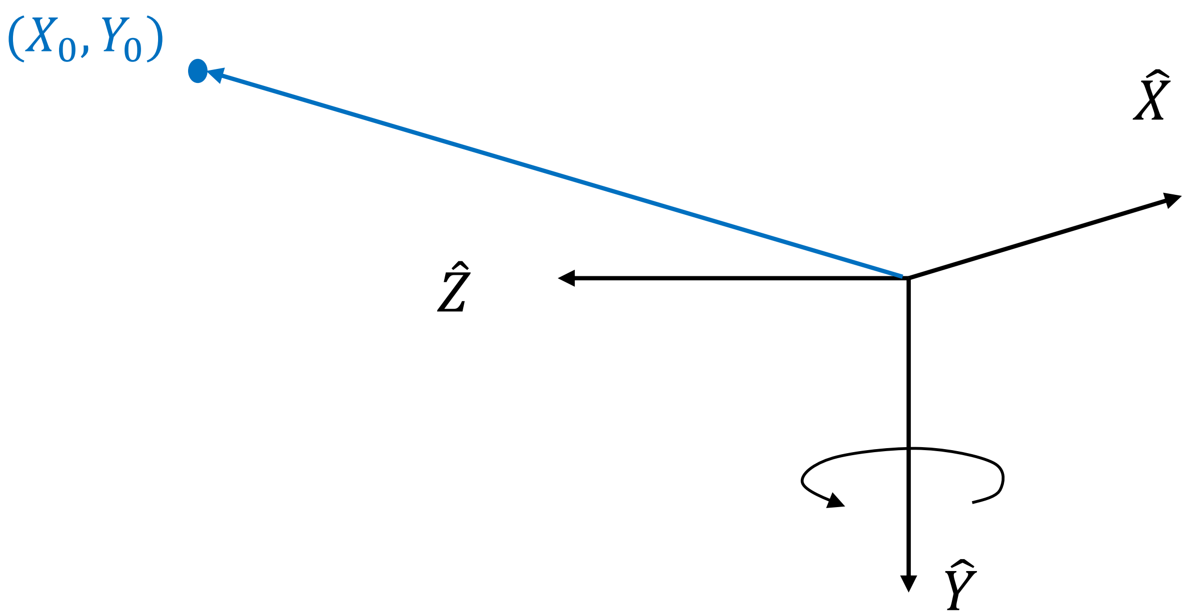
3D Camera Rotation (X axis – "tilt")



When the camera rotates about the X axis (tilt), the motion observed is the same as when the scene rotates about the X axis with the opposite velocity.

$$\begin{bmatrix} X(t) \\ Y(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Omega t) & -\sin(\Omega t) \\ 0 & \sin(\Omega t) & \cos(\Omega t) \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

3D Camera Rotation (Y axis – "pan")

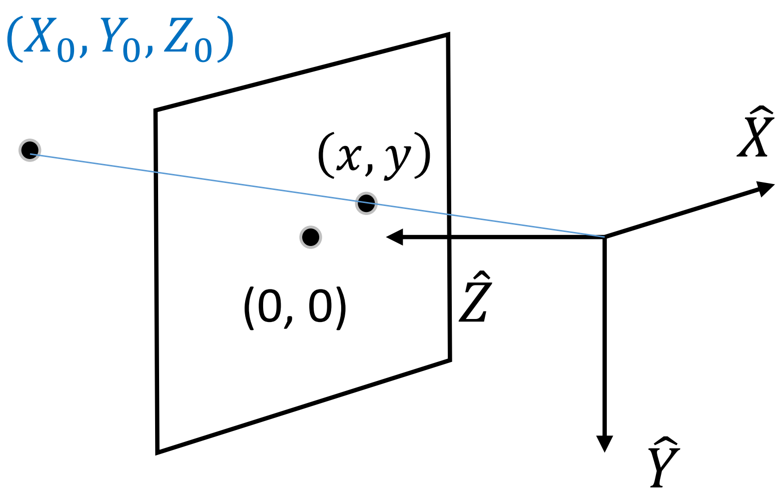


When the camera rotates about the Y axis (pan), the motion observed is the same as when the scene rotates about the Y axis with the opposite velocity.

$$\begin{bmatrix} X(t) \\ Y(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} \cos(\Omega t) & 0 & \sin(\Omega t) \\ 0 & 1 & 0 \\ -\sin(\Omega t) & 0 & \cos(\Omega t) \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

Image Projection (continued):

What is the image motion field seen by a rotating camera?

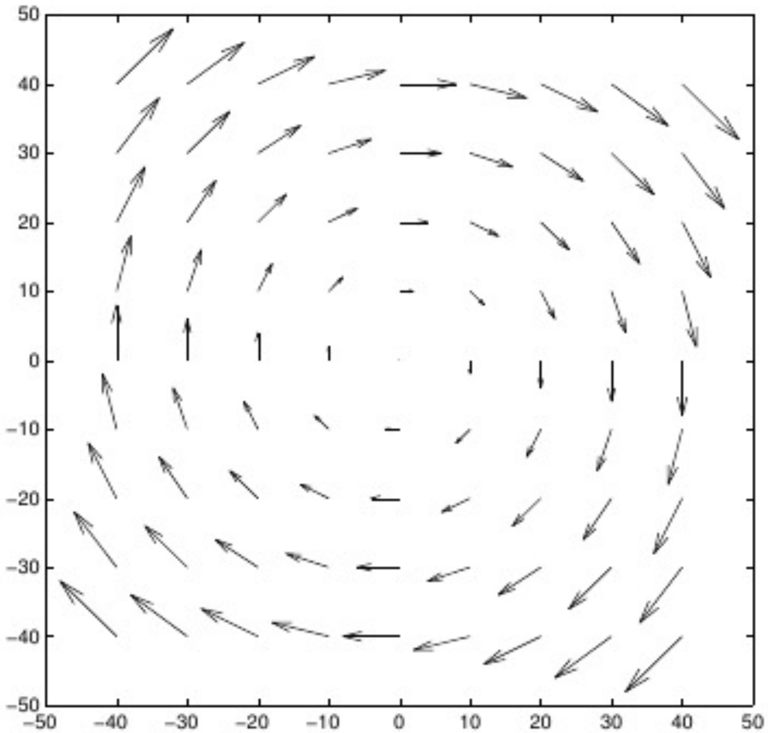


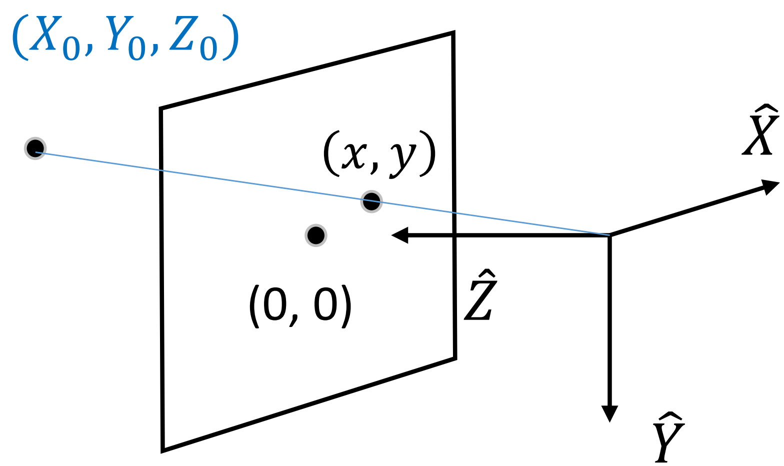
$$(x(t), y(t)) = (\frac{X(t)}{Z(t)}, \frac{Y(t)}{Z(t)})f$$

$$(v_x, v_y) = \frac{d}{dt}(x(t), y(t))|_{t=0}$$

[See Lecture Notes (Appendix TODO) for derivations. On the following slides, I will give results only.]

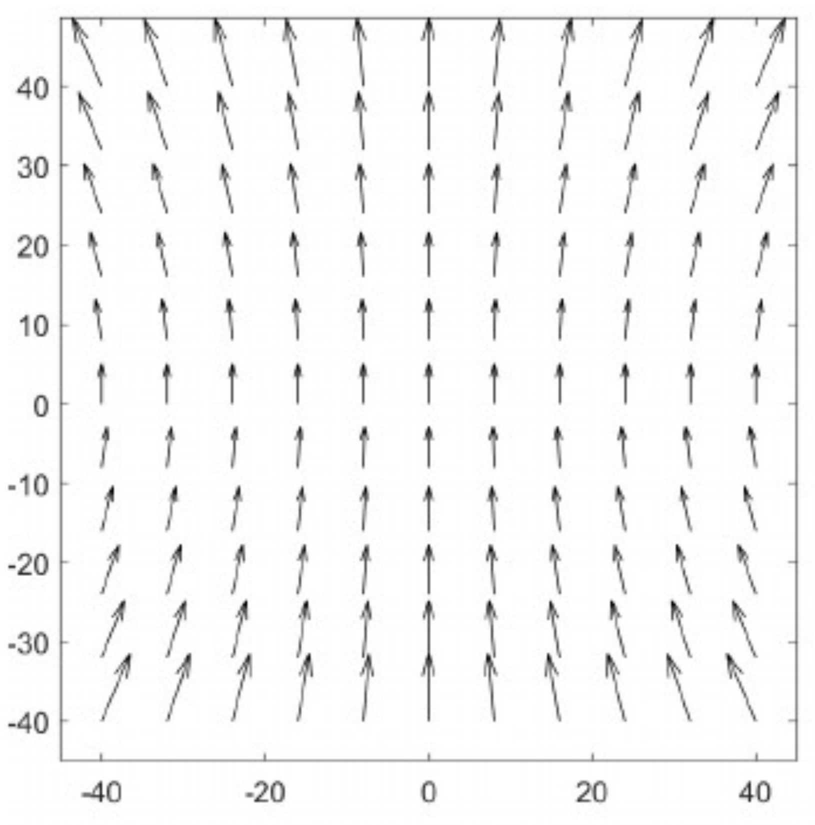
3D Camera Rotation about Z axis: "roll"

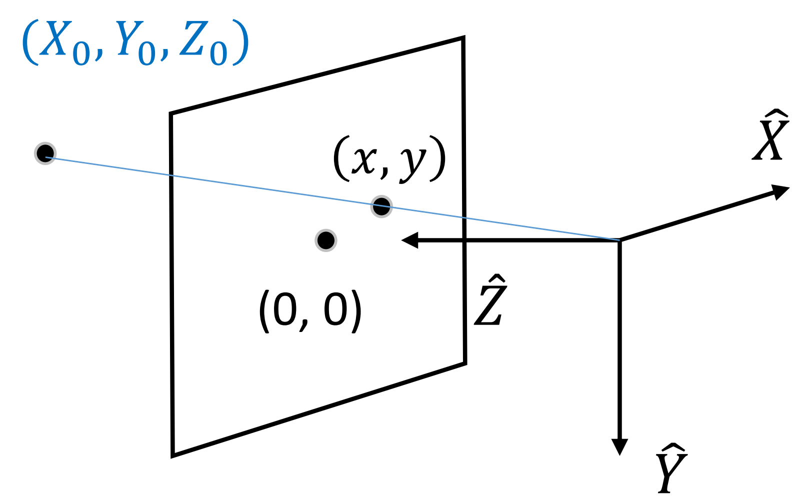




$$(v_x, v_y) = \Omega_Z(-y, x)$$ where Ω_Z is rotational velocity about Z axis

3D Camera Rotation about X axis: "pitch" or "tilt"

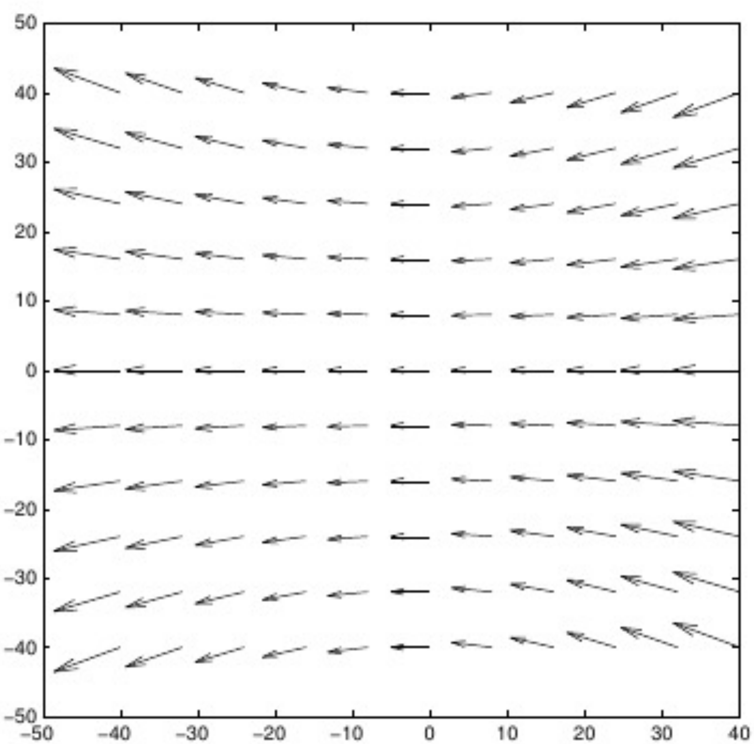


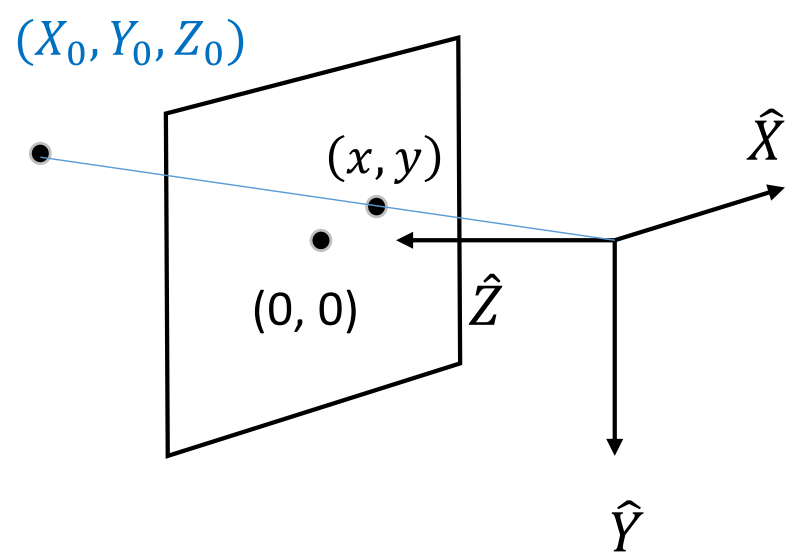


$$(v_x, v_y) = \Omega_X\left(\frac{xy}{f}, f\left(1 + \left(\frac{y}{f}\right)^2\right)\right)$$

where Ω_X is rotational velocity about X axis

3D Camera Rotation about Y axis: “pan”





$$(v_x, v_y) \;=\; \Omega_Y(f(1 + (\frac{x}{f})^2),\; \frac{xy}{f})$$ where Ω_Y is rotational velocity about Y axis

Note:

- One can define motion fields from rotation about arbitrary axis $(\Omega_X, \Omega_Y, \Omega_Z)$. Details omitted.

- The rotation field does *not* depend on depth.

  Recall the translation field does depend on depth as we saw last lecture.

Rotations

- 2D rotations
- 3D rotations + projection (continuous)
- 3D rotations (discrete)
- review of cross product (left vs. right hand coordinates)

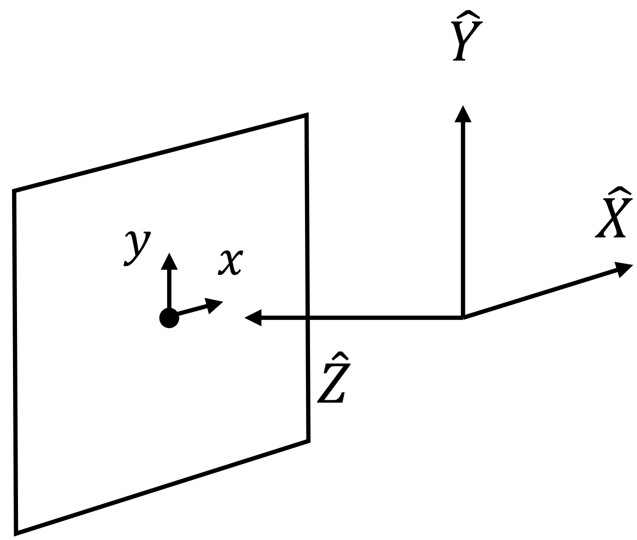
Classic computer vision problem:

- Given two image frames $I(x, y)$ and $J(x, y)$ taken by a two nearby cameras, estimate the "image motion" (v_x, v_y) between frames.
- Estimate the relative camera translation (T_X, T_Y, T_Z) and rotation $(\Omega_X, \Omega_Y, \Omega_Z)$ and the depth map $Z(x, y)$ that best explains the image motion.

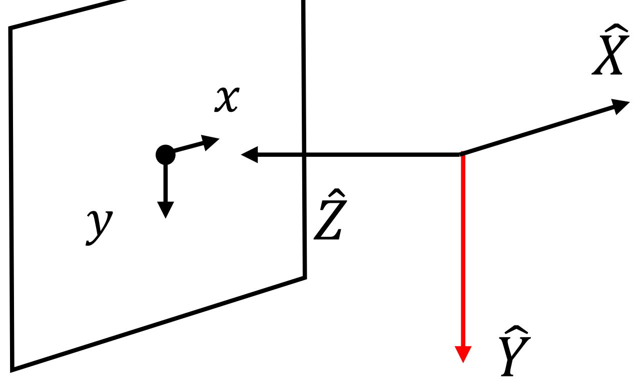
We will cover fundamental elements of this problem in the coming weeks…

Left versus right hand coordinate systems

Left hand coordinates

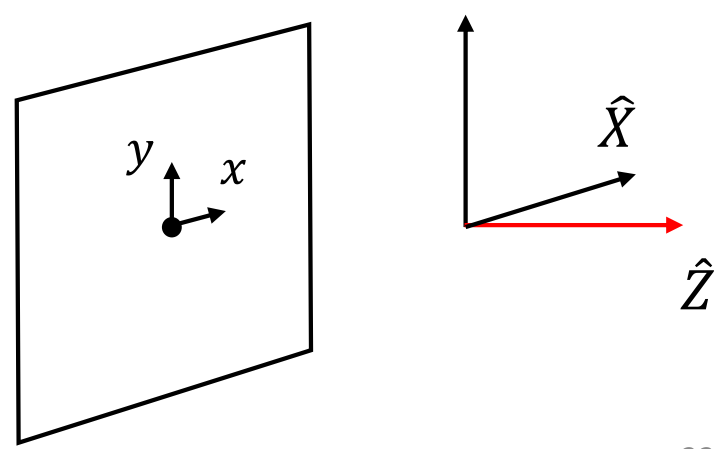


Right hand coordinates
(what I will use from now on)



Right hand coordinates
(used in computer graphics)

$\hat{Y}$

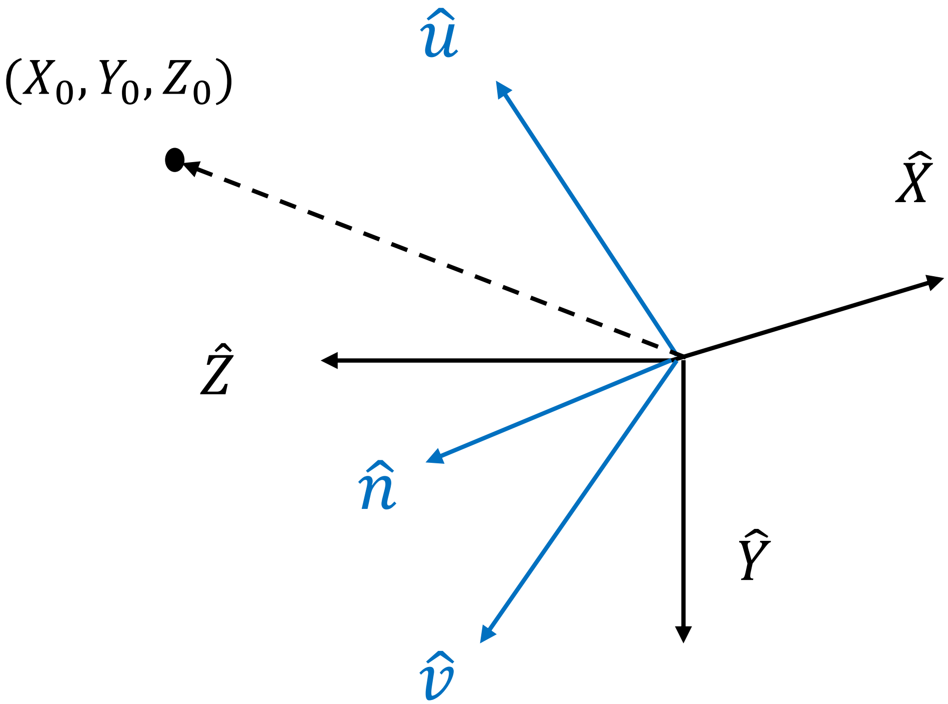


3D Camera Rotation (discrete)

A 3D rotation matrix is a 3x3 matrix that has orthonormal rows and columns and its determinant is 1.

$$\mathbf{R}^T = \mathbf{R}^{-1}$$

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$$



The matrix $\boldsymbol{R}$ rotates a 3D point into a different coordinate system, whose axes are the rows of $\boldsymbol{R}$.

The rotation takes the inner (dot) product with the rows of $\boldsymbol{R}$.

$$\mathbf{R} = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{bmatrix}$$

3D Camera Rotation (discrete)

A 3D rotation matrix preserves the length of a vector.
It also preserves the angles between vectors.

Why?

$$(\mathbf{R}\mathbf{p}_1) \cdot (\mathbf{R}\mathbf{p}_2) = \mathbf{p}_1^T \mathbf{R}^T \mathbf{R} \mathbf{p}_2 = \mathbf{p}_1^T \mathbf{p}_2 = \mathbf{p}_1 \cdot \mathbf{p}_2$$

3D Reflection

A 3D rotation matrix is a 3x3 matrix that has orthonormal rows and columns and its determinant is 1.

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The above matrices are reflections.
Their determinant is -1.

Axis of Rotation

For any rotation matrix $\boldsymbol{R}$, one can show there is a unique vector $\boldsymbol{v}$ such that

$$\boldsymbol{R}\boldsymbol{v} = \boldsymbol{v}.$$

This (eigen)vector defines the *axis of rotation*.

Lecture 15

Rotations &

Homogeneous Coordinates

Homogenous Coordinates

To represent a 3D point, (X, Y, Z) we write the point in 4D as $(X, Y, Z, 1)$.

This allows us to represent various transformations in a similar way, namely using 4D matrix multiplication.

Translation

$$\begin{bmatrix} X + T_x \\ Y + T_y \\ Z + T_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} \boxed{} \\ 1 \end{bmatrix} = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

3 x 3 rotation matrix

result of 3D rotation

Scaling

$$\begin{bmatrix} \sigma_X X \\ \sigma_Y Y \\ \sigma_Z Z \\ 1 \end{bmatrix} = \begin{bmatrix} \sigma_X & 0 & 0 & 0 \\ 0 & \sigma_Y & 0 & 0 \\ 0 & 0 & \sigma_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

What if we have a value different than 1 in the 4^{th} coordinate?

$$\{\ (wX, wY, wZ, w)\ \ :\ \ w \neq 0\ \}$$

No problem. By association, we will let these 4D points all represent the same 3D point, namely (X, Y, Z).

Consider a 3D point (X, Y, Z) and scale the 3D coordinates of this point by $s > 0$:

$$(sX, sY, sZ, 1) \equiv (X, Y, Z, \frac{1}{s})$$

The equivalence holds because you can rescale by 1/s on the left. For different values s, we get 3D points that all lie along a line from the origin through (X, Y, Z).

$$(sX, sY, sZ, 1) \equiv (X, Y, Z, \frac{1}{s})$$

As $s \rightarrow \infty$, we get a "point at infinity" in direction (X, Y, Z).

$$lim_{s\rightarrow\infty}\ (sX, sY, sZ, 1) = (X, Y, Z, 0)$$

What happens if we apply a translation or rotation or scaling transformation to a point at infinity?

Translating a point at infinity

$$? \quad = \begin{bmatrix} 1 & 0 & 0 & T_X \\ 0 & 1 & 0 & T_Y \\ 0 & 0 & 1 & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix}$$

Translating a point at infinity

$$\begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_X \\ 0 & 1 & 0 & T_Y \\ 0 & 0 & 1 & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix}$$

Rotating a point at infinity

$$\begin{bmatrix} \\ \\ \\ 0 \end{bmatrix} = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix}$$

result of 3D rotation

3 x 3 rotation matrix

So, it behaves similarly to the rotation of a finite point.

Scaling

$$\begin{bmatrix} \sigma_X X \\ \sigma_Y Y \\ \sigma_Z Z \\ 0 \end{bmatrix} = \begin{bmatrix} \sigma_X & 0 & 0 & 0 \\ 0 & \sigma_Y & 0 & 0 \\ 0 & 0 & \sigma_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix}.$$

Note the direction of the point at infinity will be changed when axes are scaled by different amounts.

Exercise:

How are (3D) points at infinity related to vanishing points?

Homogeneous Coordinates in 2D

Translation:

$$\begin{bmatrix} x + T_x \\ y + T_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation:

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

Homogeneous Coordinates in 2D

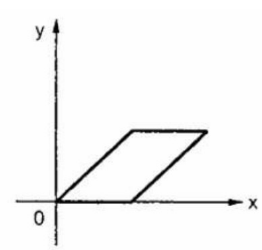
scaling

$$\begin{bmatrix} \sigma_x x \\ \sigma_y y \\ 1 \end{bmatrix} = \begin{bmatrix} \sigma_X & 0 & 0 \\ 0 & \sigma_Y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

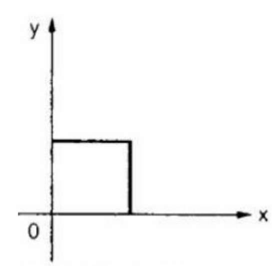
shear

$$\begin{bmatrix} x + sy \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Recall motion field of ground plane from last lecture



(b) Object after x shear



(a) Original object

Points at infinity in 2D homogenous coordinates

$$lim_{s\rightarrow\infty}(sx, sy, 1) = lim_{s\rightarrow\infty}\ (x, y, \frac{1}{s}) = (x, y, 0)$$

You can think of this as a *direction vector*.
(Its magnitude is undefined.)

We will use 2D points at infinity in coming weeks.