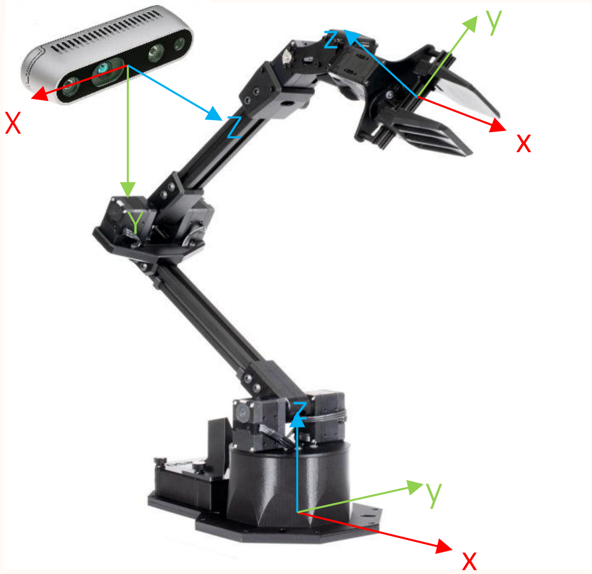
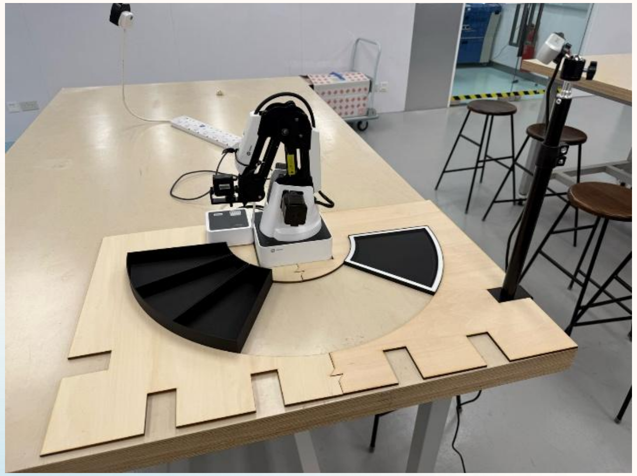
Tam Wing Fan Innovation Wing
FACULTY OF ENGINEERING

Introduction to Calibration





Presented to you by:
ArmStrong SIG
Ian Lo / Ryan Pang

4/6/2025 – Intern Training

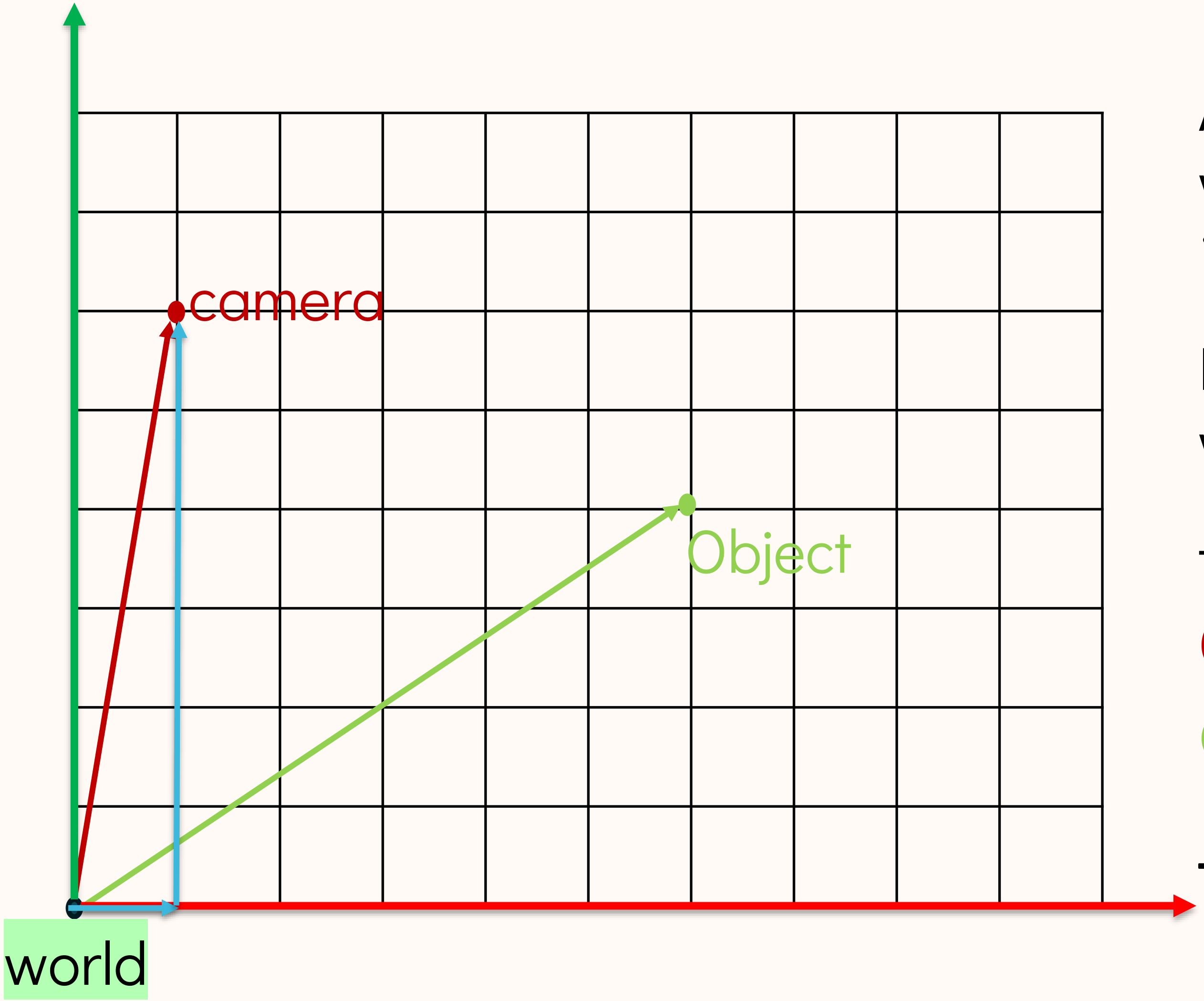
Outline

1. What is a Frame?
2. Camera Model
3. Calibration Concepts
4. Homogeneous Transformation Matrix
5. Eye-to-Hand Calibration (3 DoF)
6. Demonstration

01

What is a Frame

What is a Frame?



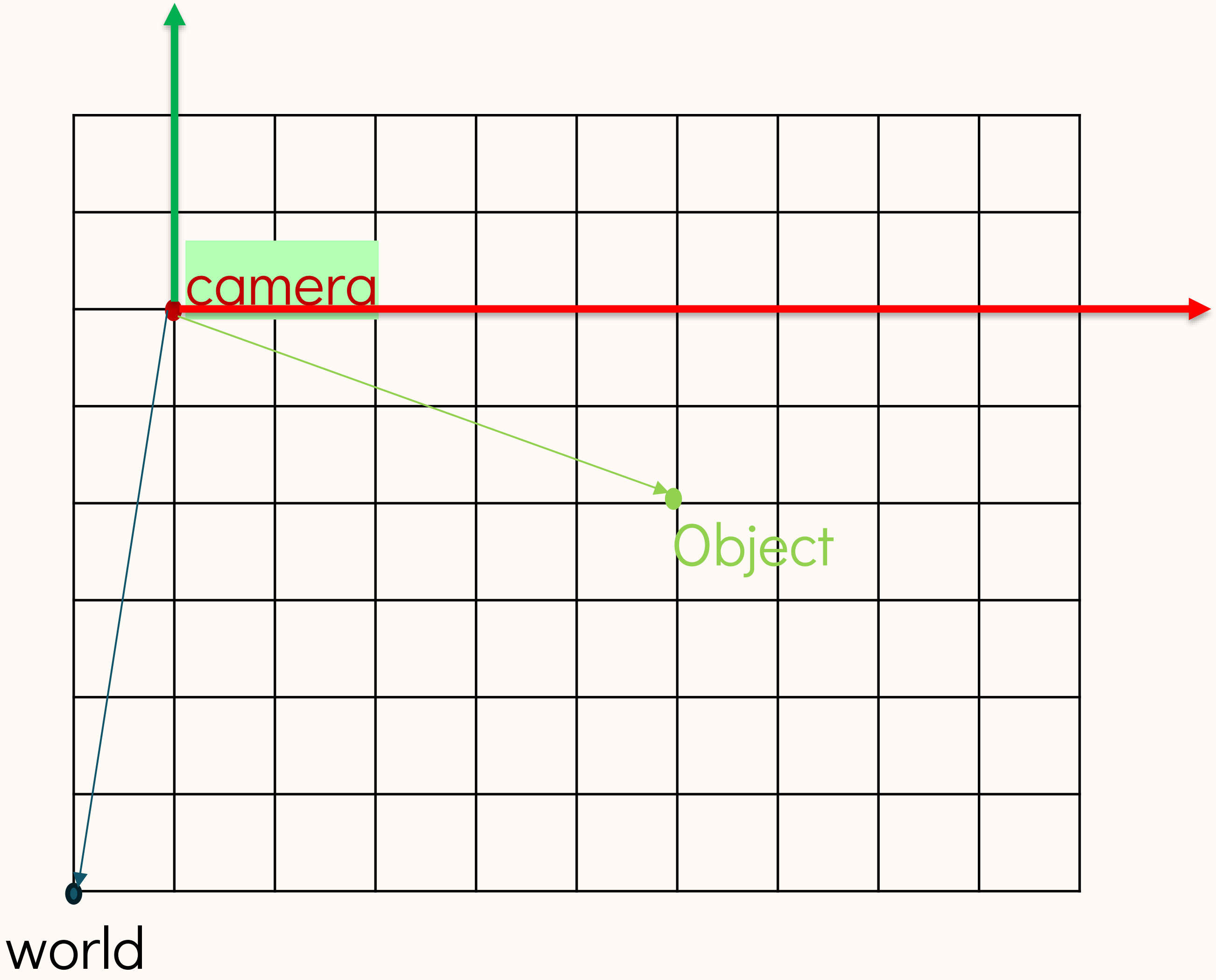
A frame is the reference point to which every other coordinate point is defined

Let's define some points on the grid: world, camera, object

Then, with respect to the **world:**
Camera: (1,6) (1 units on x axis, 6 units on y)
Object: (6,4)

This is in the "world frame"

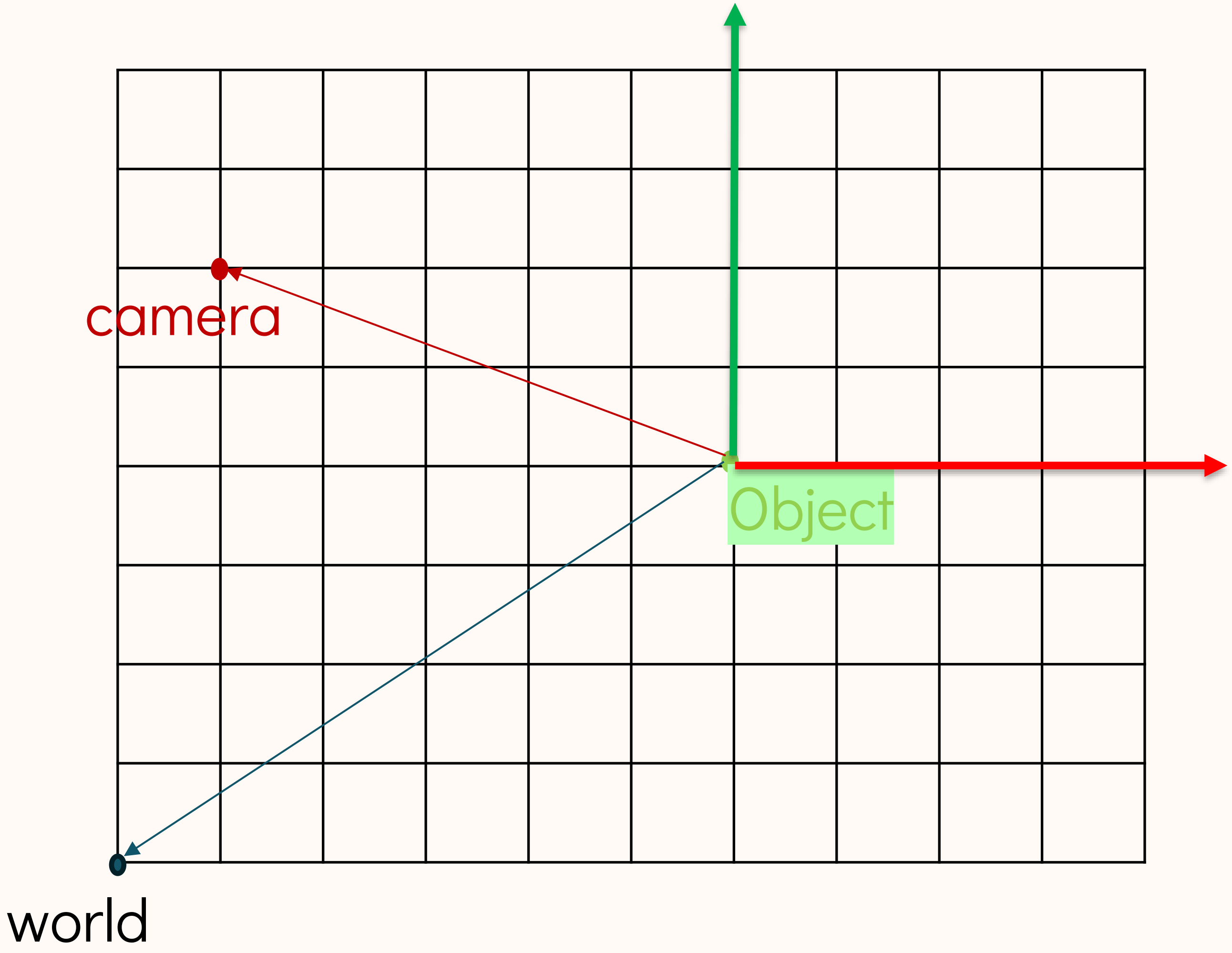
What is a Frame?



Similarly, with respect to the camera:
World: (-1,-6)
Object: (5,-2)

This is the “camera frame”

What is a Frame?



And with respect to the object:

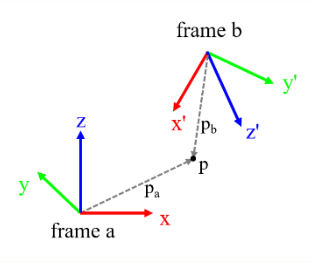
World: (-6,-4)

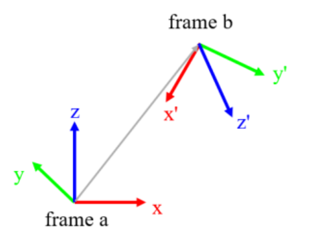
Camera: (-5,2)

This is the “object frame”

Notation:

$$H_b^a$$



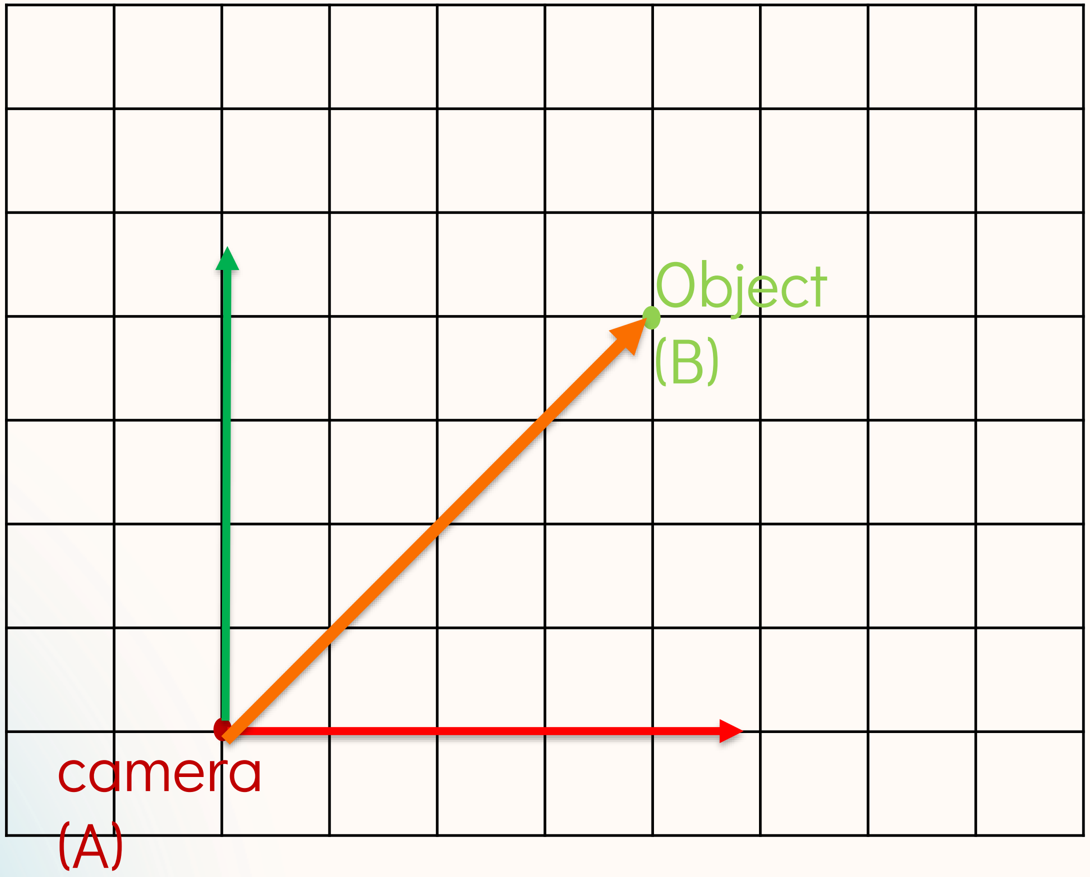


Interpretation	Meaning	Use case
Transform interpretation	$H_b^a \cdot \overrightarrow{p_b} = \overrightarrow{p_a}$	You have a point in frame **b**, and you want to express it in frame **a**
Pose interpretation	H_b^a is the pose of frame b in reference frame a	Frame **b** is located and oriented relative to frame **a**

Notation:

$$H_b^a$$

Interpretation	Meaning	Use case
Transform interpretation	$H_b^a \cdot \overrightarrow{p_b} = \overrightarrow{p_a}$	You have a point in frame **b**, and you want to express it in frame **a**
Pose interpretation	H_b^a is the pose of frame b in reference frame a	Frame **b** is located and oriented relative to frame **a**



*Q: A camera has **detected** an object and output its coordinates **relative to the camera frame**. Which matrix correctly express this relationship?*

$$H_B^A \text{ or } H_A^B$$

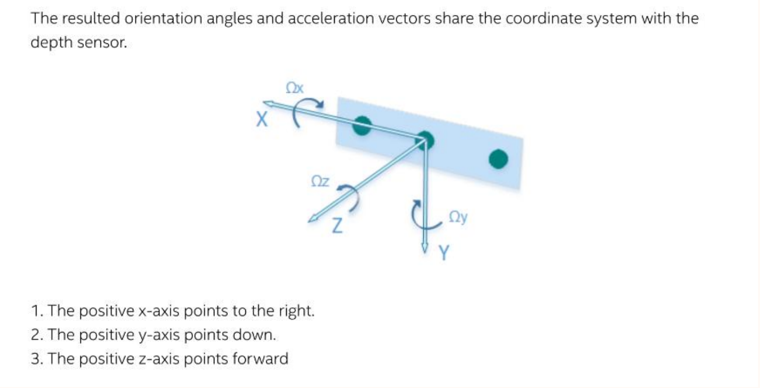
A: H_B^A because we want to express the object's pose (frame B) in the camera's frame (frame A).

Learn more: https://support.zivid.com/en/latest/reference-articles/position-orientation-coordinate-transform.html

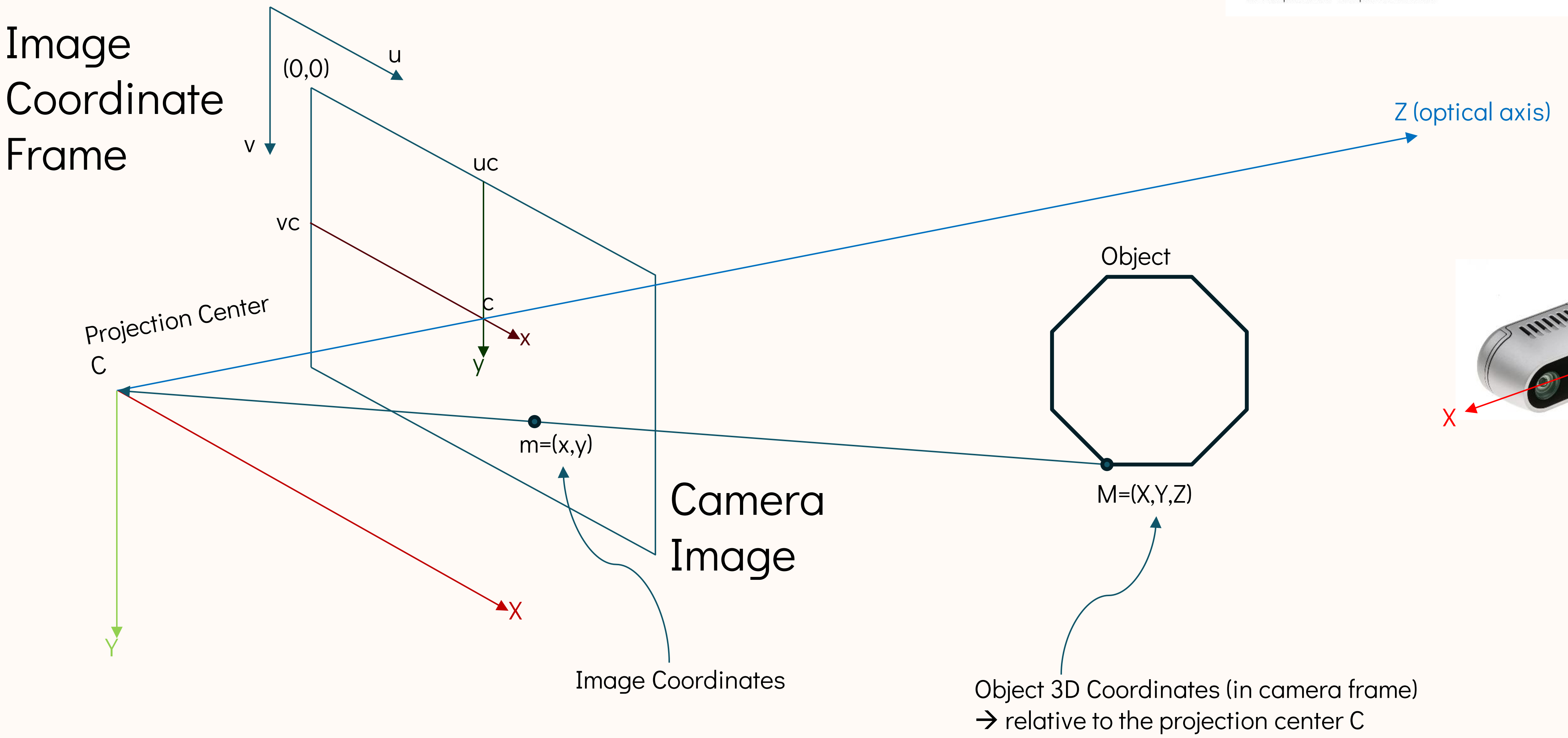
02

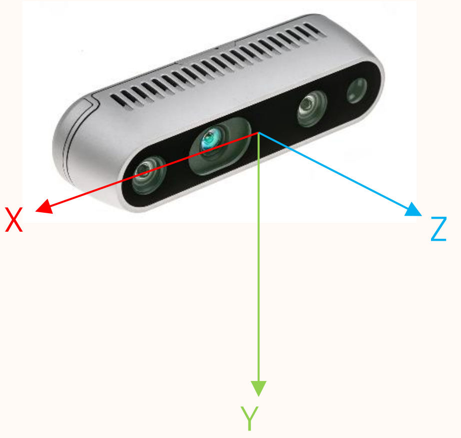
Camera Model

Camera Model

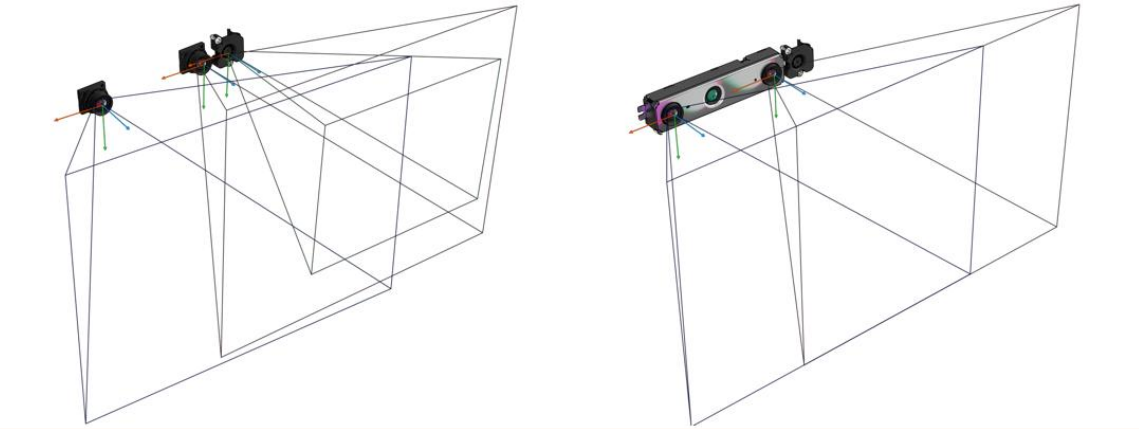

The resulted orientation angles and acceleration vectors share the coordinate system with the depth sensor.

1. The positive x-axis points to the right.
2. The positive y-axis points down.
3. The positive z-axis points forward





The Camera Frame



Make sure you **align** the depth camera's:

1. Depth frame
2. Color frame

Before you read the depth of a point from a RGB image

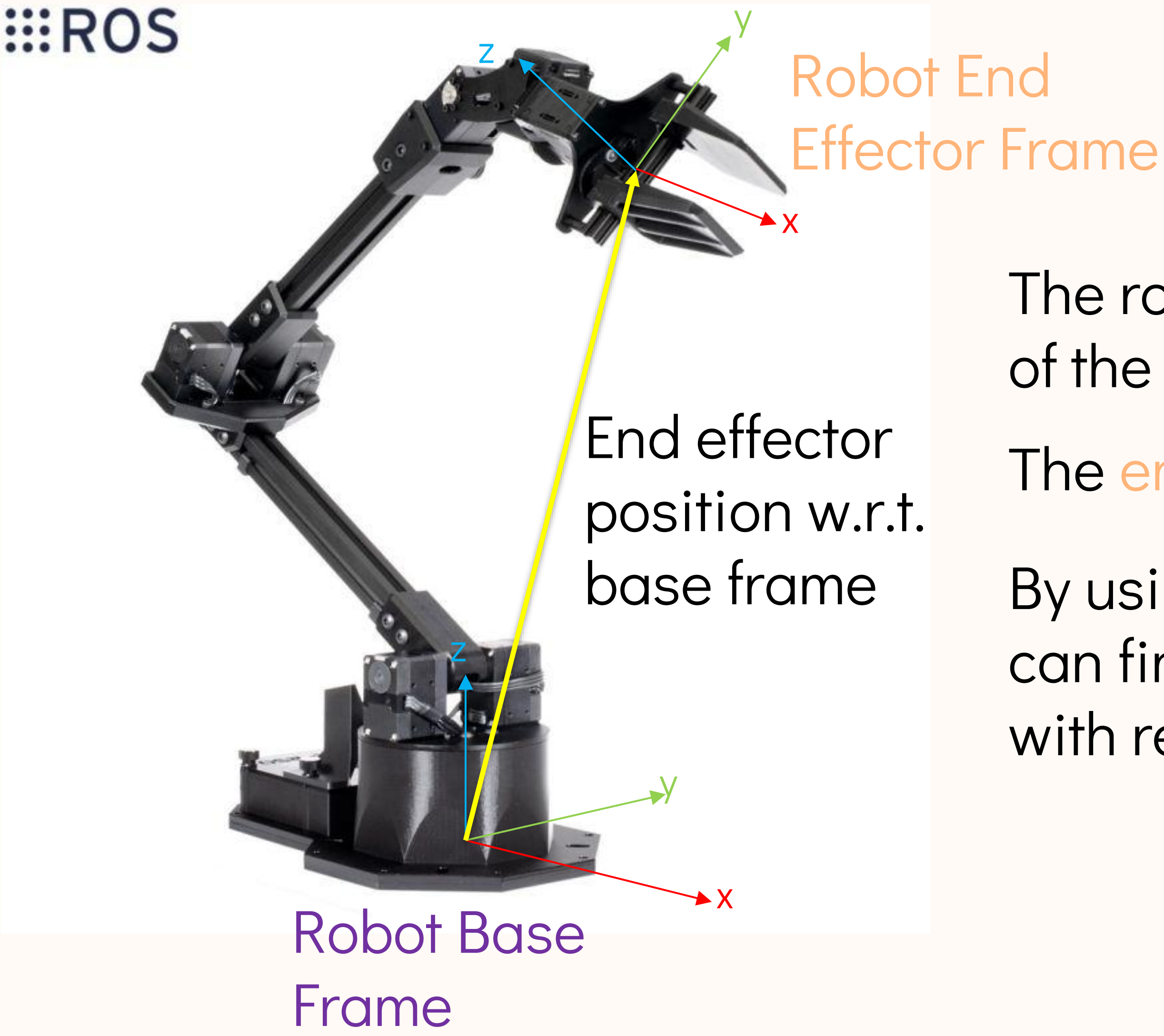
Coordinate system of Intel® RealSense™ Depth Cameras (D435)

```python
import pyrealsense2 as rs
import cv2 as cv
import numpy as np

pipeline = rs.pipeline()

cfg = rs.config()
cfg.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
cfg.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)

# Align depth to color
align_to = rs.stream.color
alignedFs = rs.align(align_to)

profile = pipeline.start(cfg)

try:
    while True:
        fs = pipeline.wait_for_frames()
        aligned_frames = alignedFs.process(fs)

        color_frame = aligned_frames.get_color_frame()
        depth_frame = aligned_frames.get_depth_frame()

        if not depth_frame or not color_frame:
            continue

        color_image = np.asanyarray(color_frame.get_data())
        depth_image = np.asanyarray(depth_frame.get_data())

        # Normalize the depth image for display
        depth_colormap = cv.applyColorMap(cv.convertScaleAbs(depth_image, alpha=0.03), cv.COLORMAP_JET)

        # Stack both images horizontally
        images = np.hstack((color_image, depth_colormap))

        # Display the result
        cv.imshow('Aligned Frames', images)

        if cv.waitKey(1) & 0xFF == ord('q'):
            break
finally:
    pipeline.stop()
    cv.destroyAllWindows()
```

03

Calibration Concepts

Robot Frame



Forward Kinematics:
- *Known Joint Angles → Calculate EE coordinate*

Inverse Kinematics:
- *?*

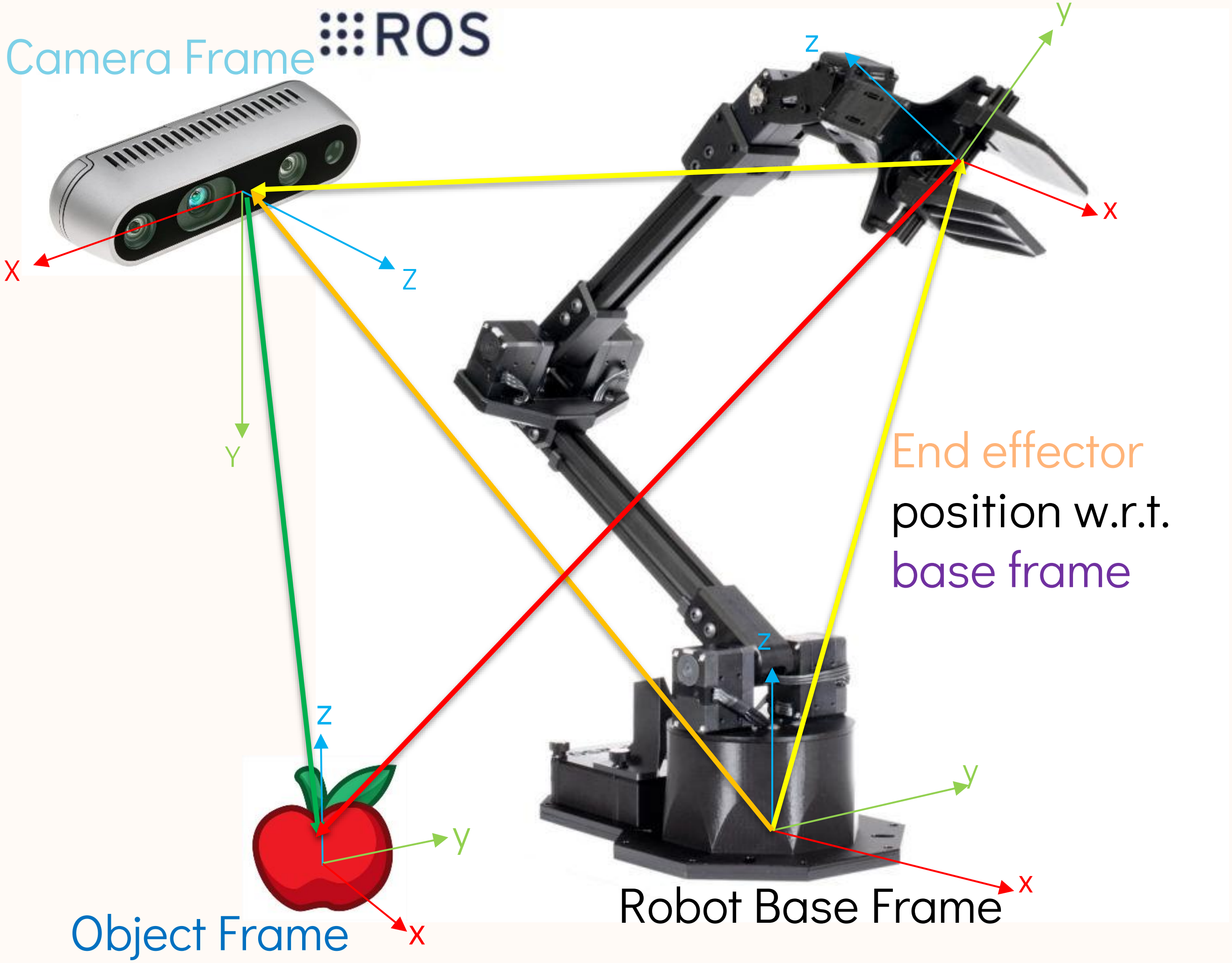
Inverse Kinematics:
- *Known EE coordinate → Find Joint Angles*

The robot's base frame is the same as that of the world frame, at (0,0,0)

The end effector has its own frame

By using the robot's forward kinematics, we can find the position of the end effector with respect to the base.

Camera Calibration Concept



Problem:
Given that the camera knows the position of an object, we don't know what is its coordinate in the robot base frame.
The camera and robot base have different frames

Calculate the position of the camera w.r.t. the robot base frame.

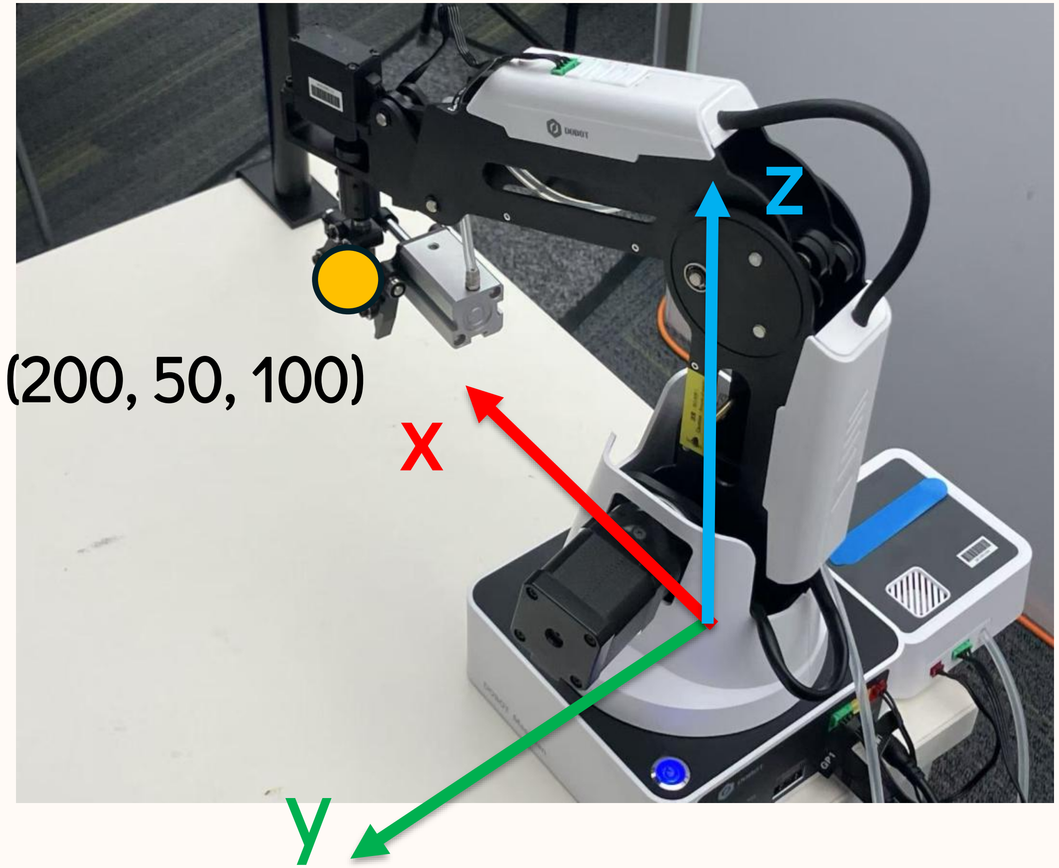
Then we can command the robot to move and manipulate according to what camera sees.

Hand-Eye Calibration in Practice

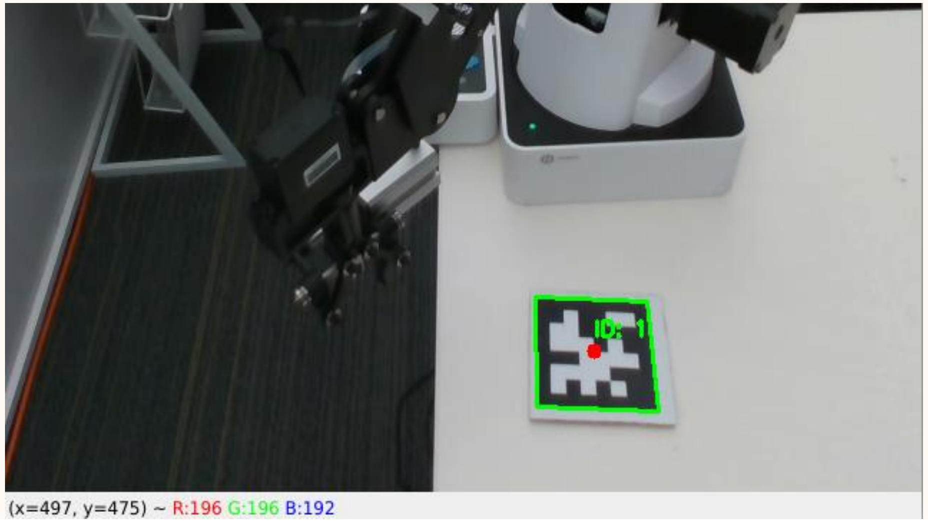
The gripper's pose in robot base frame is known.

Q: What is the name of this process?

A: Forward Kinematics



Pose of some object (AprilTag) in camera frame is known using some software.



$$
{}^{c}H_{t} = \begin{bmatrix} 0.043 & 0.999 & -0.003 & 75 \\ -0.743 & 0.034 & 0.668 & 128 \\ 0.667 & -0.027 & 0.744 & 525 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

04

Homogeneous Transformation Matrix

Tutorial on Transformation

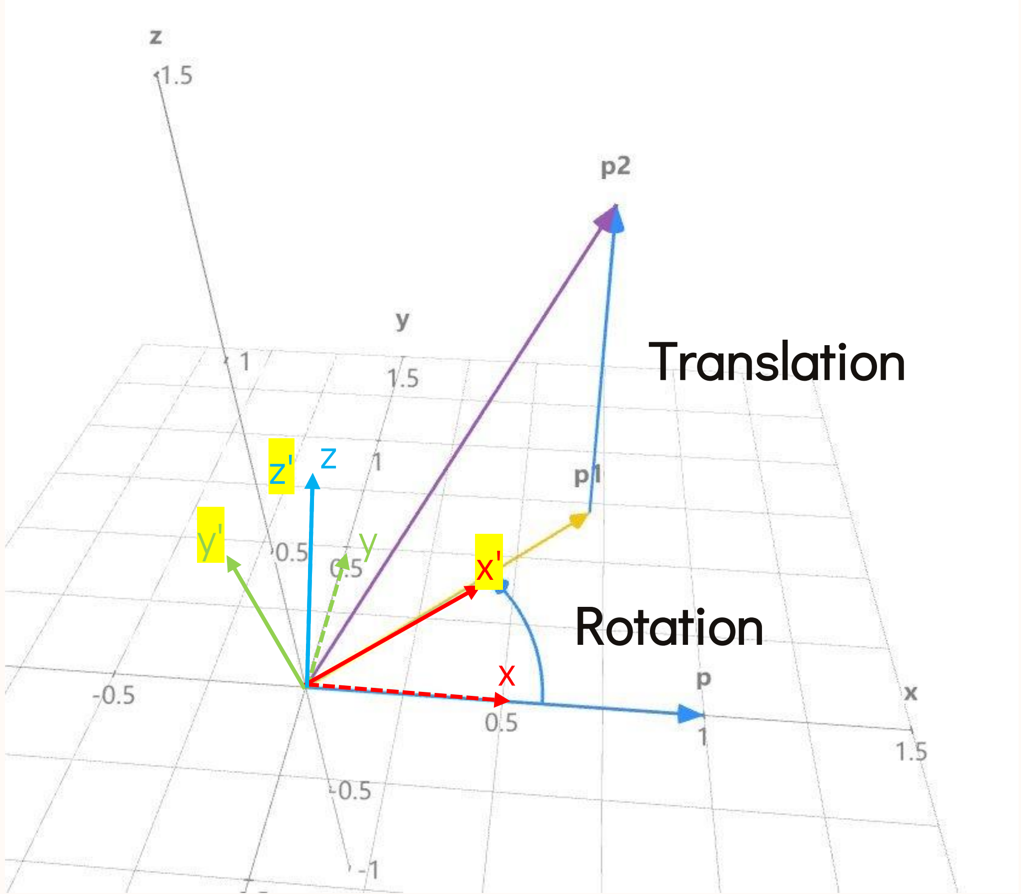
- A 3D rigid **transformation** is composed of a rotation and a translation.
- The rotation can be expressed by a 3*3 matrix, and the translation can be expressed by a 3-element vector.

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad \boldsymbol{t} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- The homogeneous transformation matrix is a 4*4 matrix in this form:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- A **pose** is composed of a position and an orientation, which can be analogized to the translation and rotation. So, a pose can be expressed by a transformation matrix.
- Example: Rotate $p = [1,0,0]$ around z-axis for 45 degrees to get p_1 and translate p_1 by $[0,0,1]$ to get p_2.



$$\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{t} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

$$R = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{p}_1 = R\mathbf{p}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}$$

$$\mathbf{p}_2 = \mathbf{p}_1 + \mathbf{t}$$

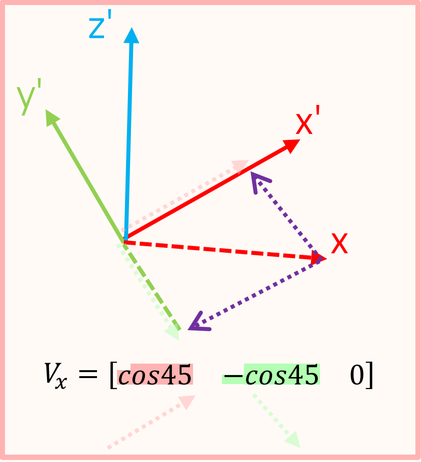
$$= \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \end{bmatrix}$$

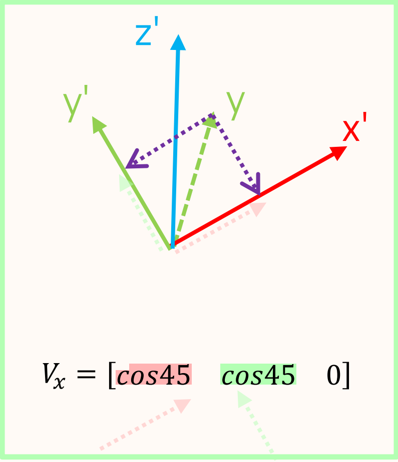
$[1\ 0\ 0]^T$ is under the **old** frame

From equation $p_{new} = Rp_{old}$, we know that R is in a form of a matrix that can transform a point from old frame to new frame, H_{old}^{new}

For R_{old}^{new}, it should be in form of $\begin{bmatrix} -V_x- \\ -V_y- \\ -V_z- \end{bmatrix}$, which

$[-V_x-]$ is a unit vector representing <u>how old x-axis projects onto the new coordinate system</u>. Or, you can understand it





Homogeneous Transformation Matrix

In homogeneous form:

Note that in 3D homogeneous coordinate system, we append a '1' after column vector $[x, y, z]$ to make it homogeneous.

$$H = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

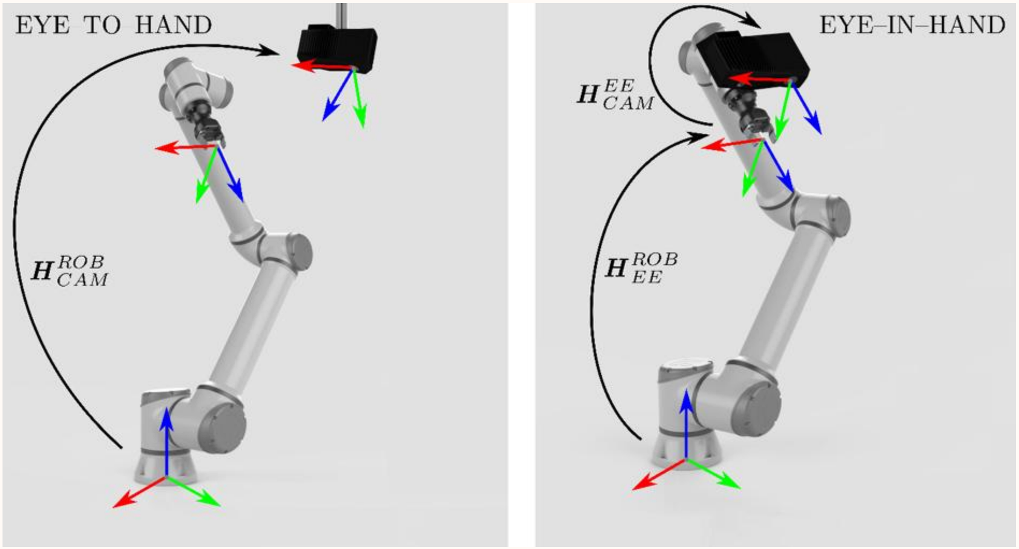
$$H^{-1} = \begin{bmatrix} R^T & -R^T * T \\ 0 & 1 \end{bmatrix} = H^T$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 1 \\ 1 \end{bmatrix}$$

Hand-Eye Calibration



- Terms
  - The base of robotic arm: *base, robot*
  - The hand of robotic arm: *end-effector (ee), gripper, tool, hand*
  - The camera: *camera, eye, sensor*
  - The target object or the calibration board: *obj, target, cal*

- Eye-in-Hand: camera is attached on the hand
- Eye-to-Hand: camera is separate from the robot and stationary

05

Eye-to-Hand Calibration

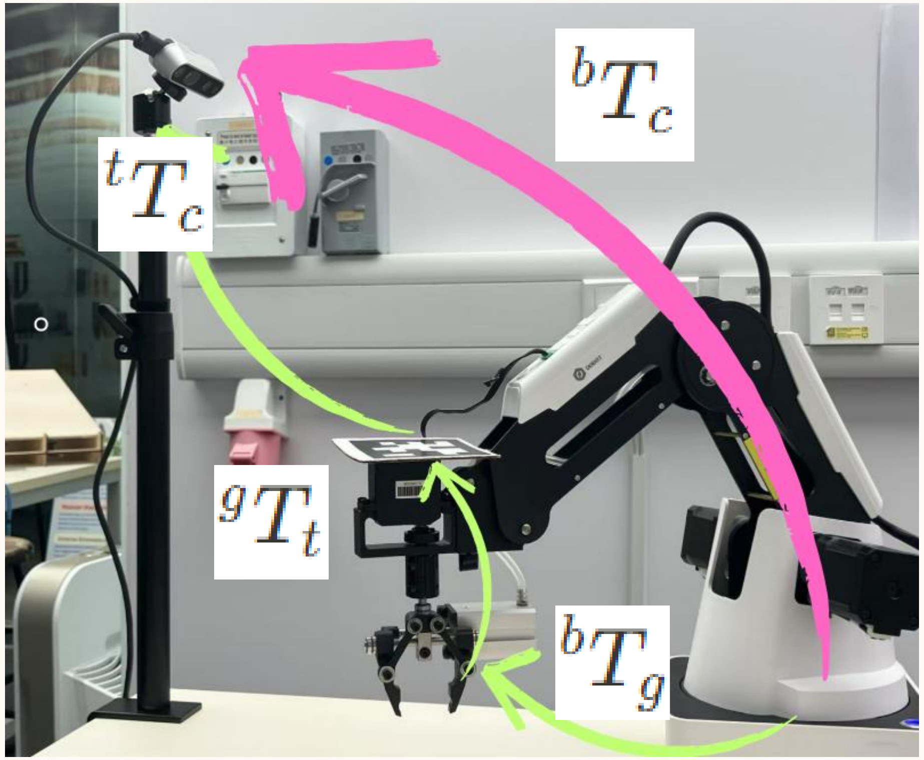
Eye-to-Hand Calibration

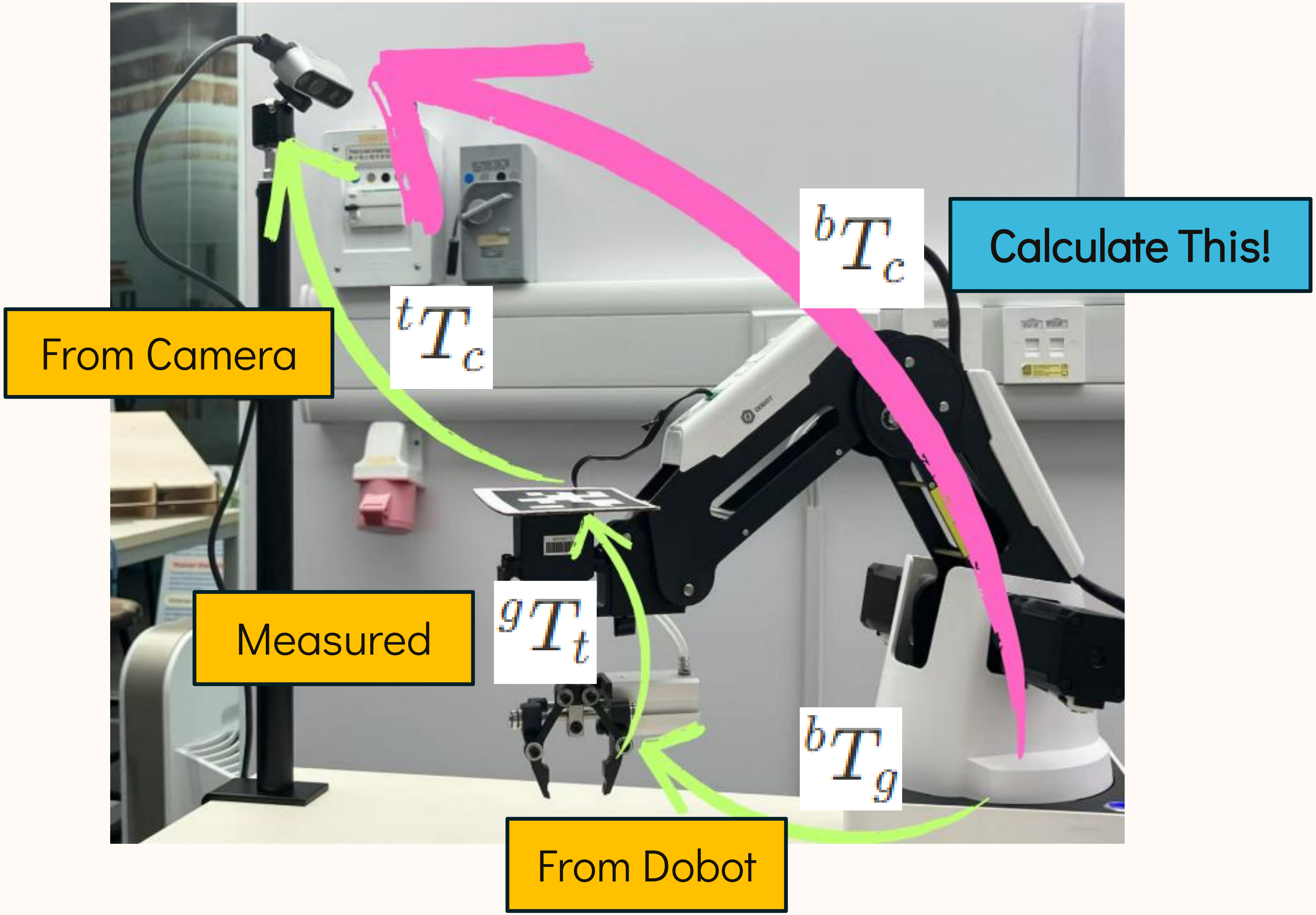
- The AprilTag will be attached on the gripper.
- There is an unknown transformation from camera to the robot base T_c^b
- There is an constant transformation from the target to the gripper
  - In common practice, we don't care about this transformation because it won't be involved in the calculation. But for simplification, we measure this matrix on Dobot and use a simpler method to calibrate.

$$ {}^{g}T_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 140 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Eye-to-Hand Calibration

- We are to find out the transformation from camera to robot base so that we can bring a point from camera frame to robot base frame.
  - $T_c^b \cdot \overrightarrow{p_c} = \overrightarrow{p_a}$
- The transformation follows a loop:
  - $T_c^b = T_g^b \cdot T_c^g \cdot T_c^t = T_g^b \cdot T_c^g \cdot {T_t^c}^{-1}$
- The camera obtains T_t^c and its inverse will be T_c^t





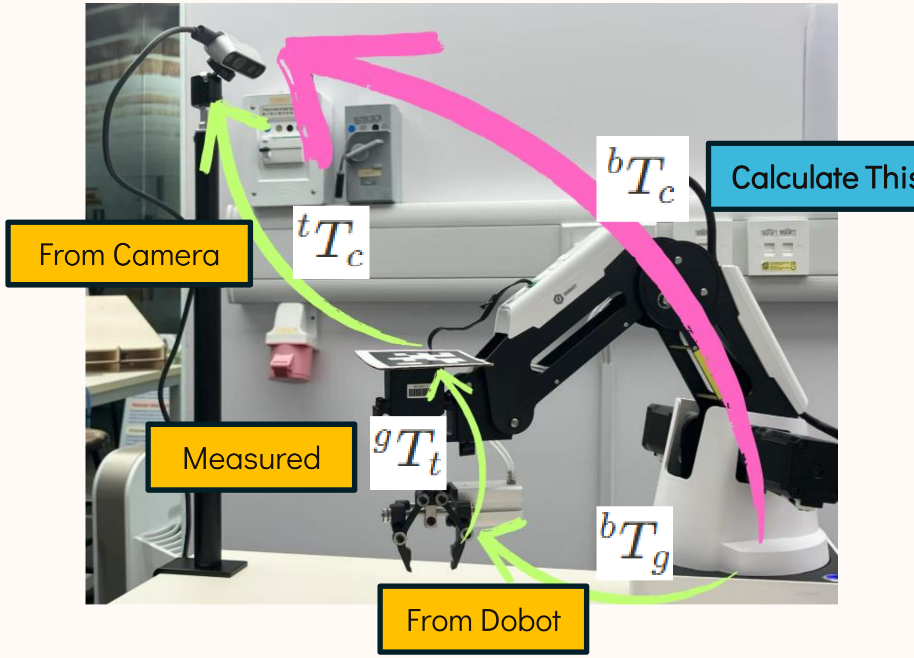
${}^{b}T_{c}$
Calculate This!
${}^{t}T_{c}$
From Camera
Measured
${}^{g}T_{t}$
${}^{b}T_{g}$
From Dobot

Eye-to-Hand Calibration

Q:
$$H_c^b = H_g^b \,?\, H_t^g \,?\, H_c^t$$

A:
$$H_c^b = H_g^b \,@\, H_t^g \,@\, H_c^t$$

1. **Data Collection:** Record transformations of T_t^c and T_g^b from various robot poses
2. **Calculation:** Get transformations of T_c^t using matrix inverse. Calculate the T_c^b transformation matrices.
3. Average: Derive the **rotation angles** and **translations** from the matrices and take the averages for them.
4. Save the final transformation in any form you like as a file for later use.



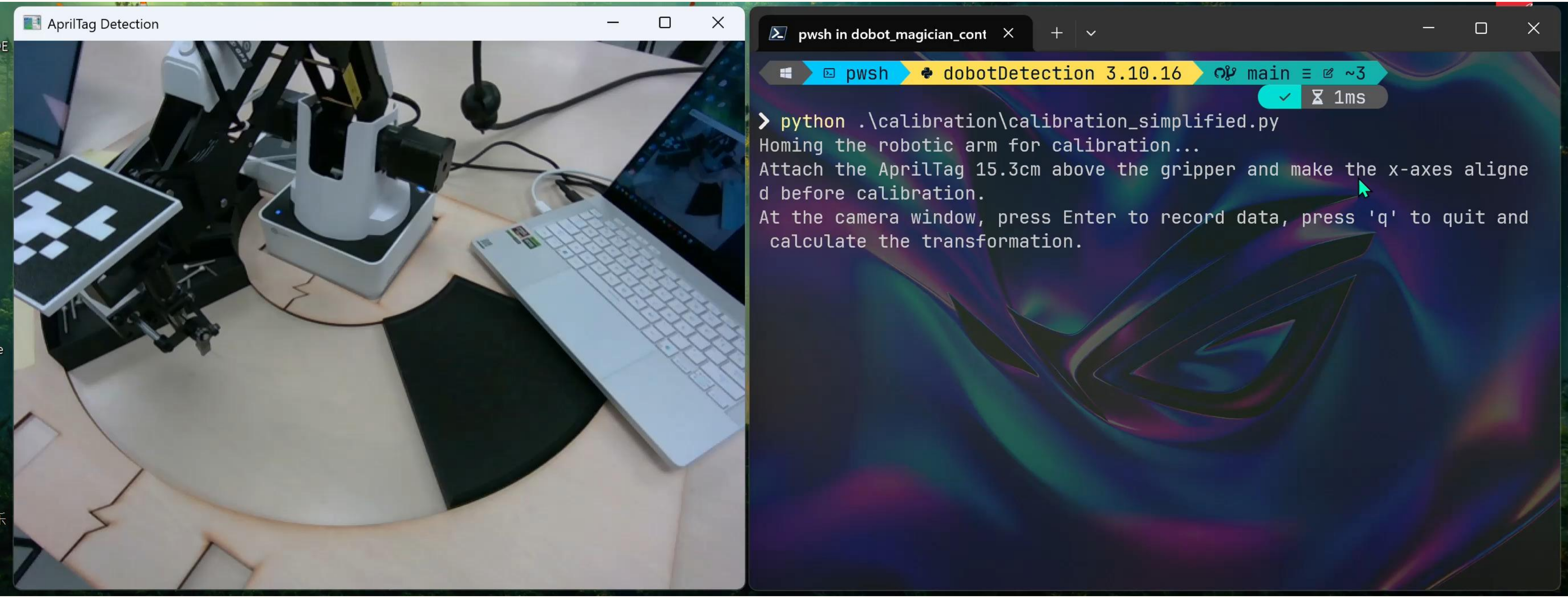
06

Demonstration

Using calibration_simplified.py

1. Run the program, wait for Dobot to calibrate itself.
2. Unlock Dobot and move to a different pose such that AprilTag can be detected by the camera.
3. Press Enter at camera view window to record the transformation (T_t^c and T_g^b)
4. Repeat step 2 and 3 to collect more data.
5. Press 'q' to calculate, average and save the transformation.
6. Check "config/camera_to_robot_transformation.yaml"

Demonstration



Validation of Calibration

Check whether the robotic arm could reach the point that the camera sees.

1. Use AprilTag and camera to obtain a position from camera view and this position should be reachable by the arm. (Same as the calibration)

2. Load the transformation you just saved. Apply it to the AprilTag's position to get the coordinate in robot base frame.

   Homogeneous

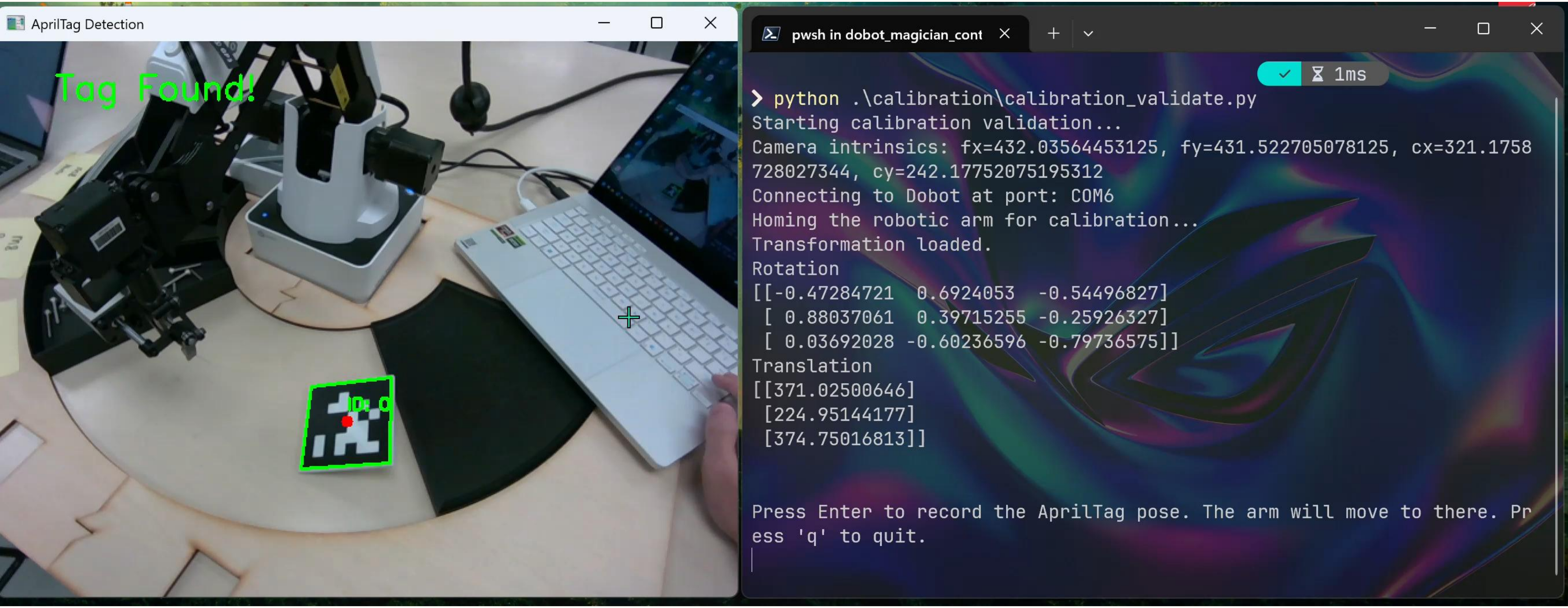
   $$ {}^{b}H_{c} \cdot \mathbf{P_c} = \mathbf{P_b} $$

   Cartesian

   $$ R \cdot \mathbf{p_c} + \mathbf{t} = \mathbf{p_b} $$

3. Move the gripper to the position, see if it goes to the correct place.

Demonstration



Practice makes perfect ☺

You will find TODO sections in the provided programs

- *calibration_simplified.py*
- *calibration_validate.py*

```
if len(cHt_list) >= 1:
    ##################################################
    # TODO: Your Code Here
    # Get the inverse of each cHt matrix
    ##################################################

    # gripper to tag transformation (x-axes aligned)
    # [1,  0,  0,  30]
    # [0, -1,  0,   0]
    # [0,  0, -1, 153]
    # [0,  0,  0,   1]
    # tag size = 0.0792 meters

    ##################################################
    # TODO: Your Code Here
    # Define the gripper to tag transformation matrix (gHt)
    ##################################################


    ##################################################
    # TODO: Your Code Here
    # Calculate bHc_list, which is the transformation from base to camera frame
    ##################################################
```

https://connecthkuhk.sharepoint.com/:u:/s/RoboticArmSIG/EROQGSheXNpAvu0l3XcMbj0BHbxLpqIoJBPJN1aFICBVXw?e=e0n1y3

https://shorturl.at/SLoJ2