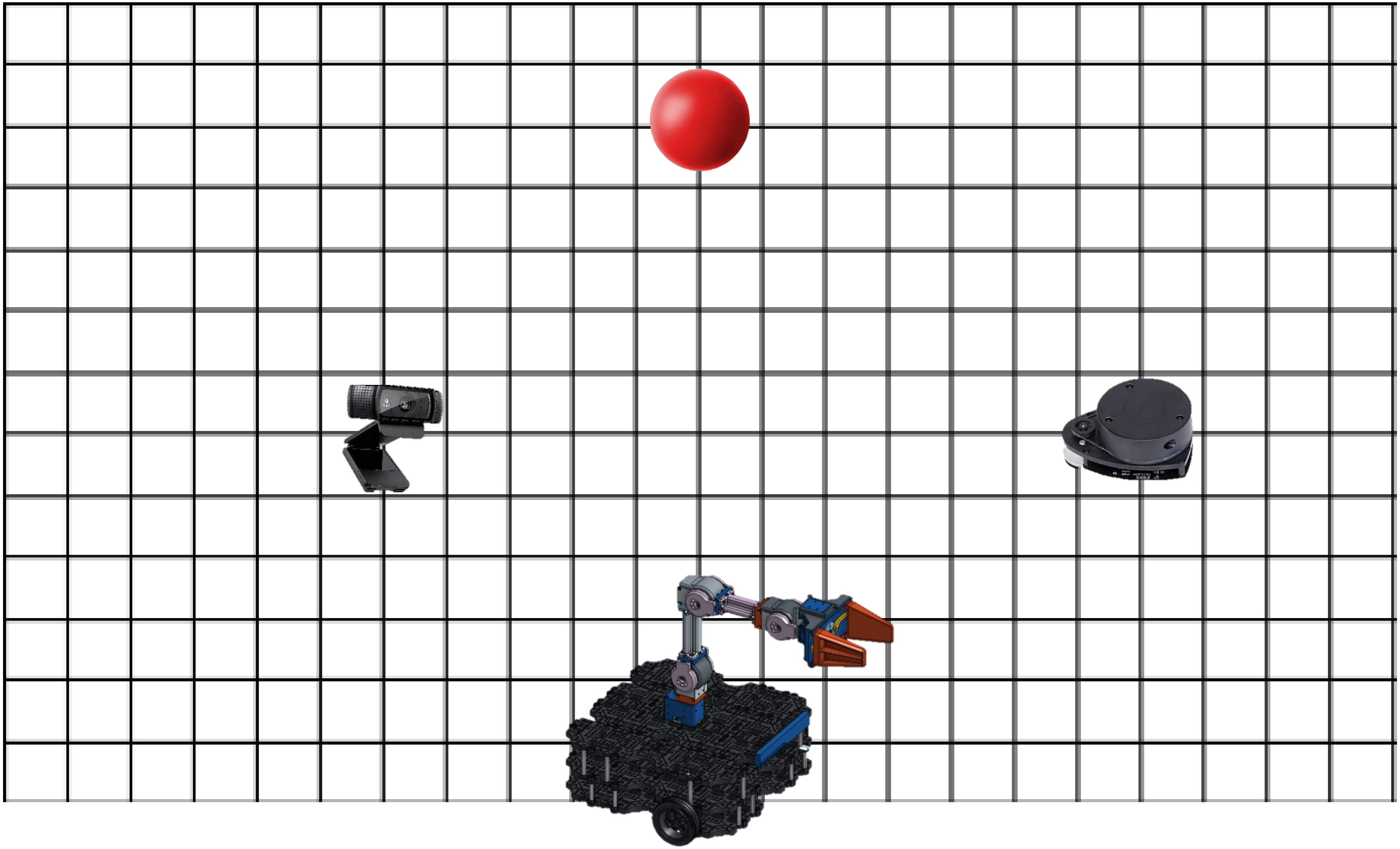
ME400 – Capstone Design
Supplementary Material – About TF(Transformation)

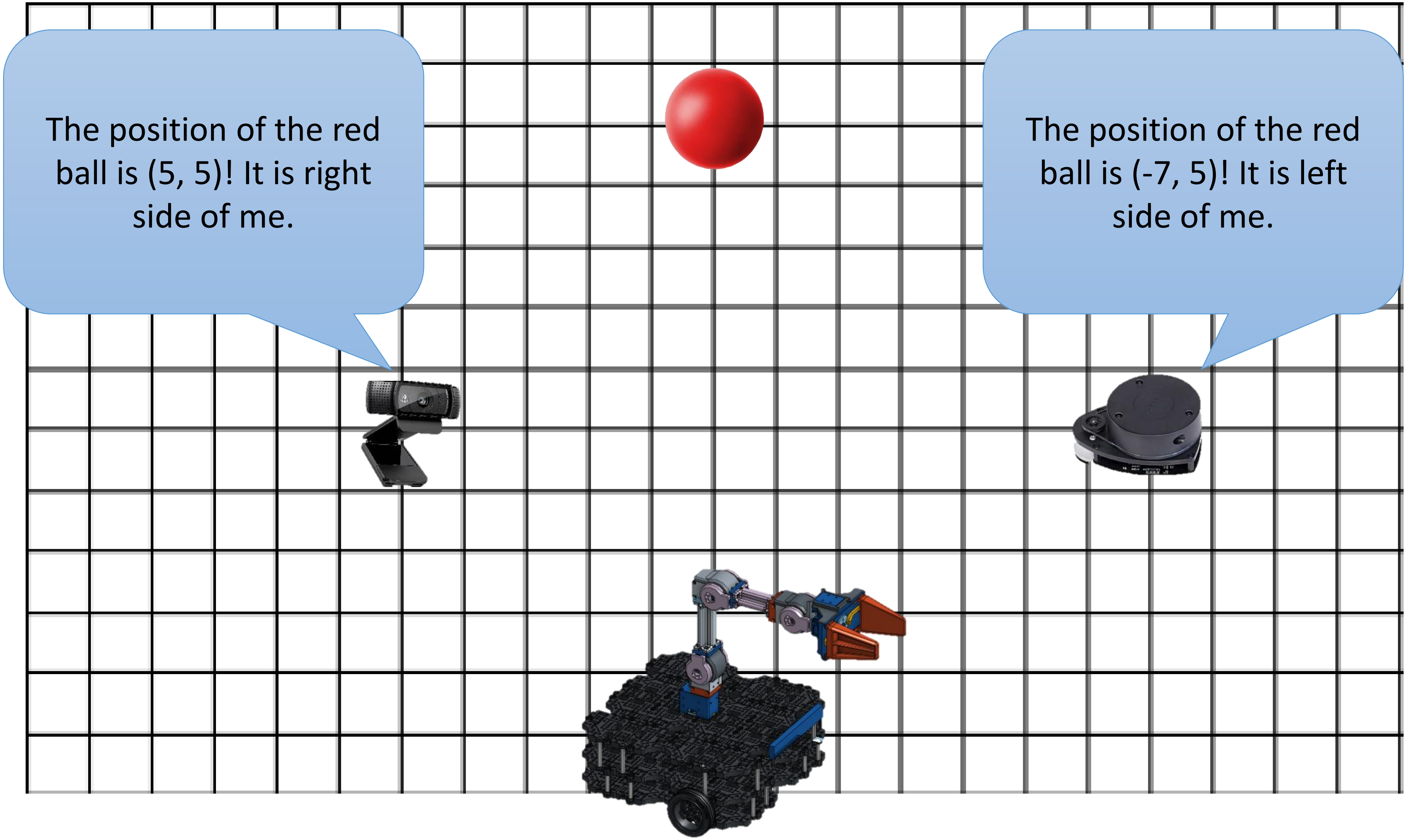
Why we need to understand transformation?

Imagine a mobile platform with webcam & lidar sensor (it is the same with your project)



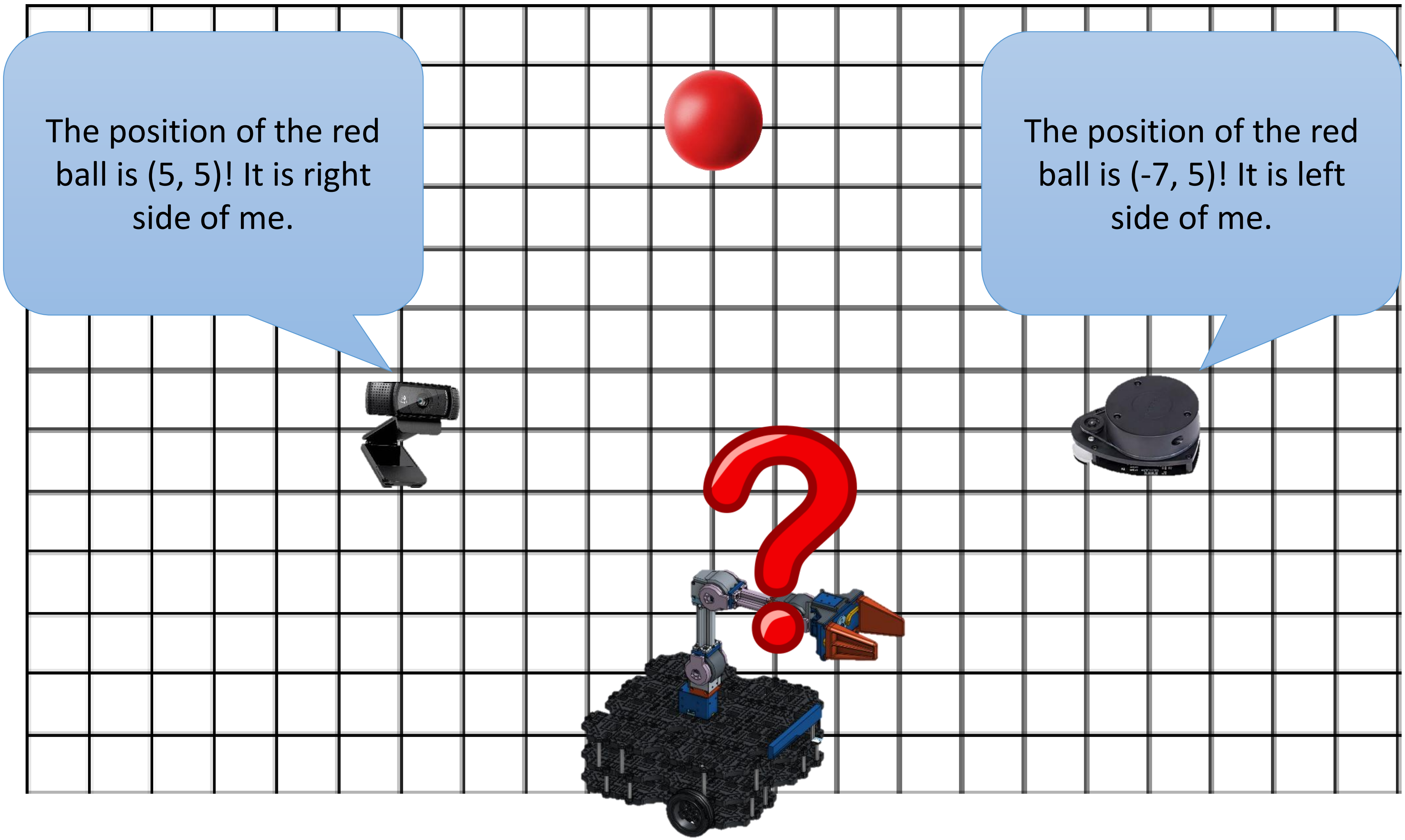
Why we need to understand transformation?

Each sensor will output the location of the ball in their coordinate.



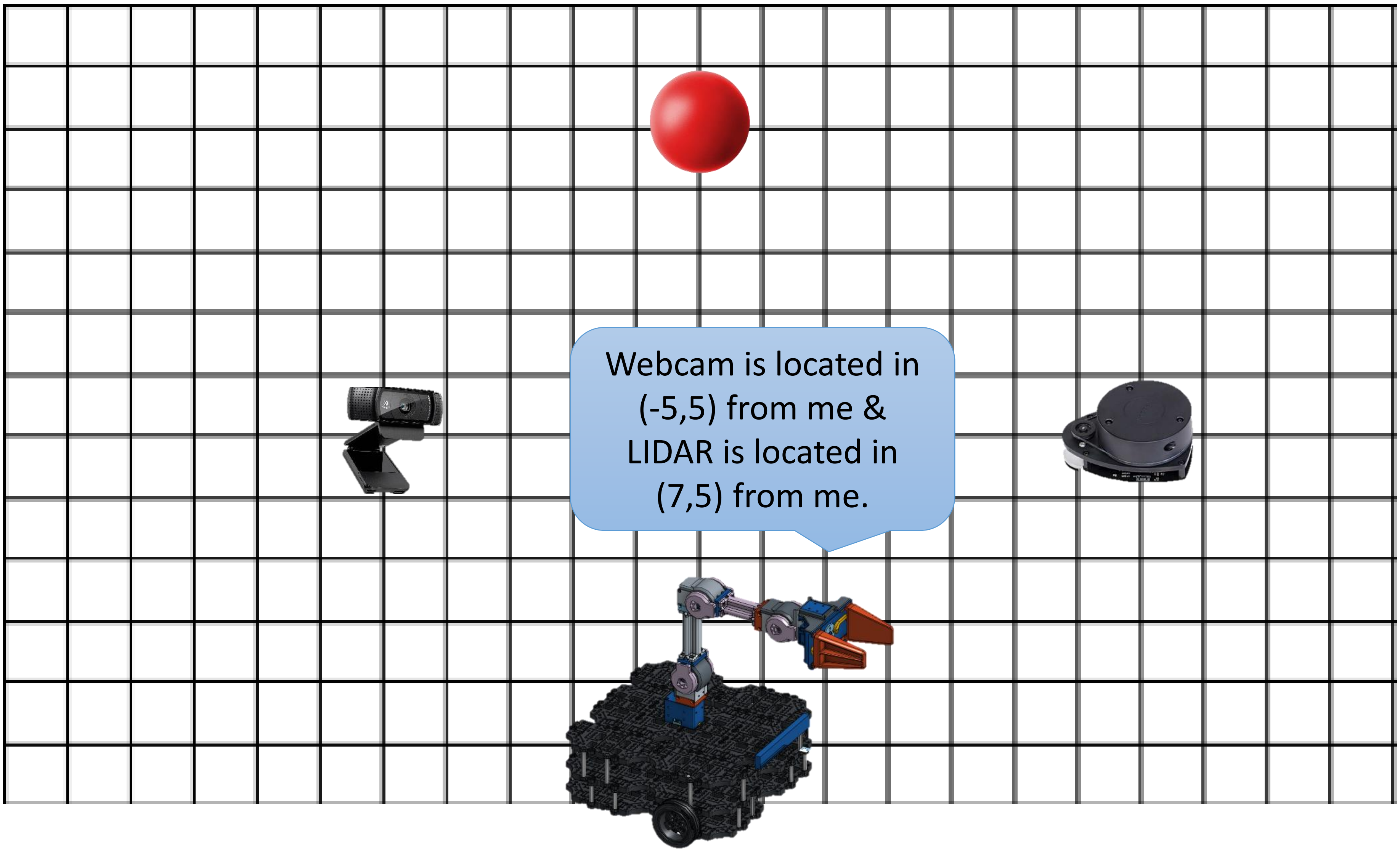
Why we need to understand transformation?

Then, which instruction the mobile platform should follow?? Go left or right??



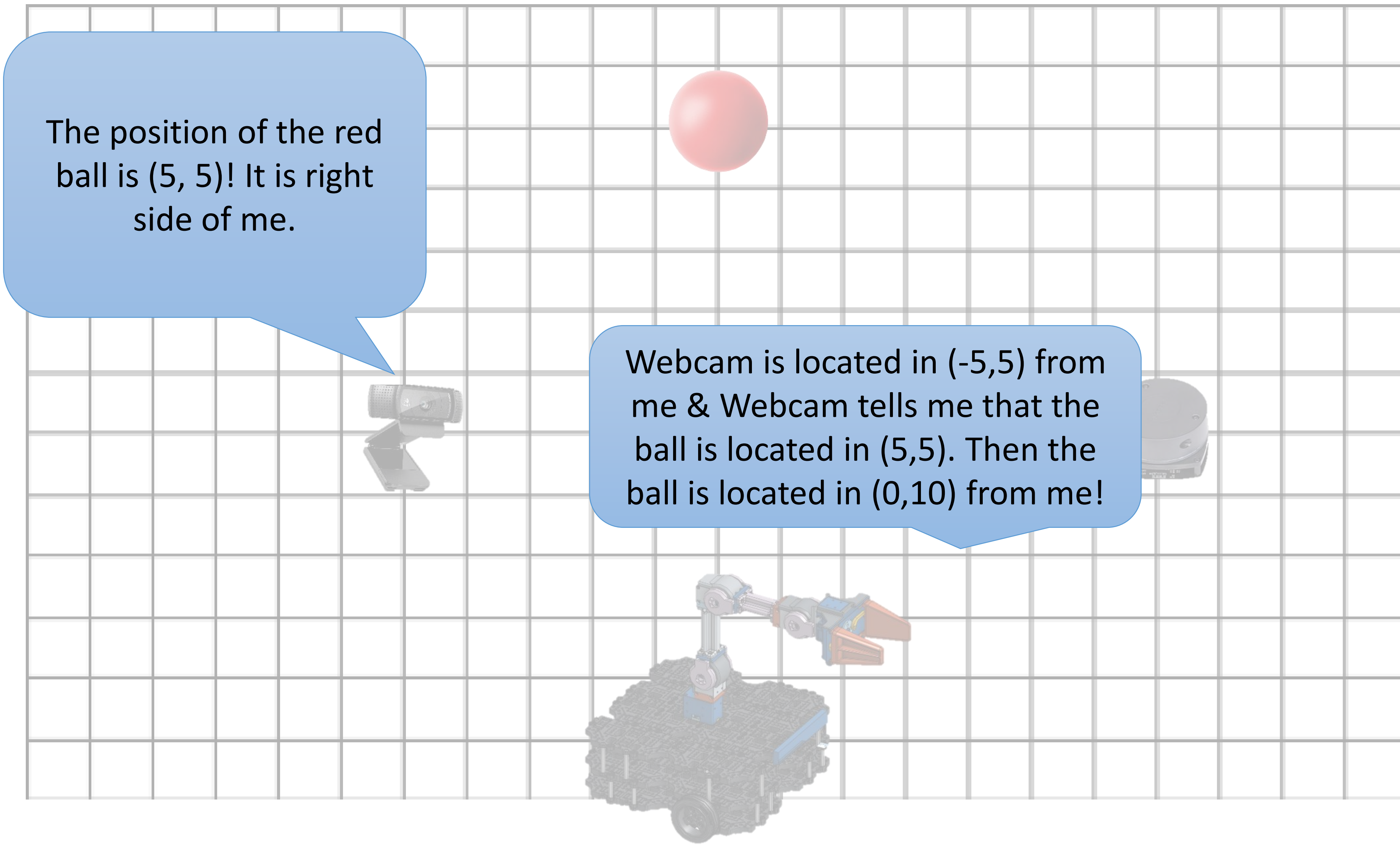
Why we need to understand transformation?

To resolve this, the mobile platform should know where the sensors are located.



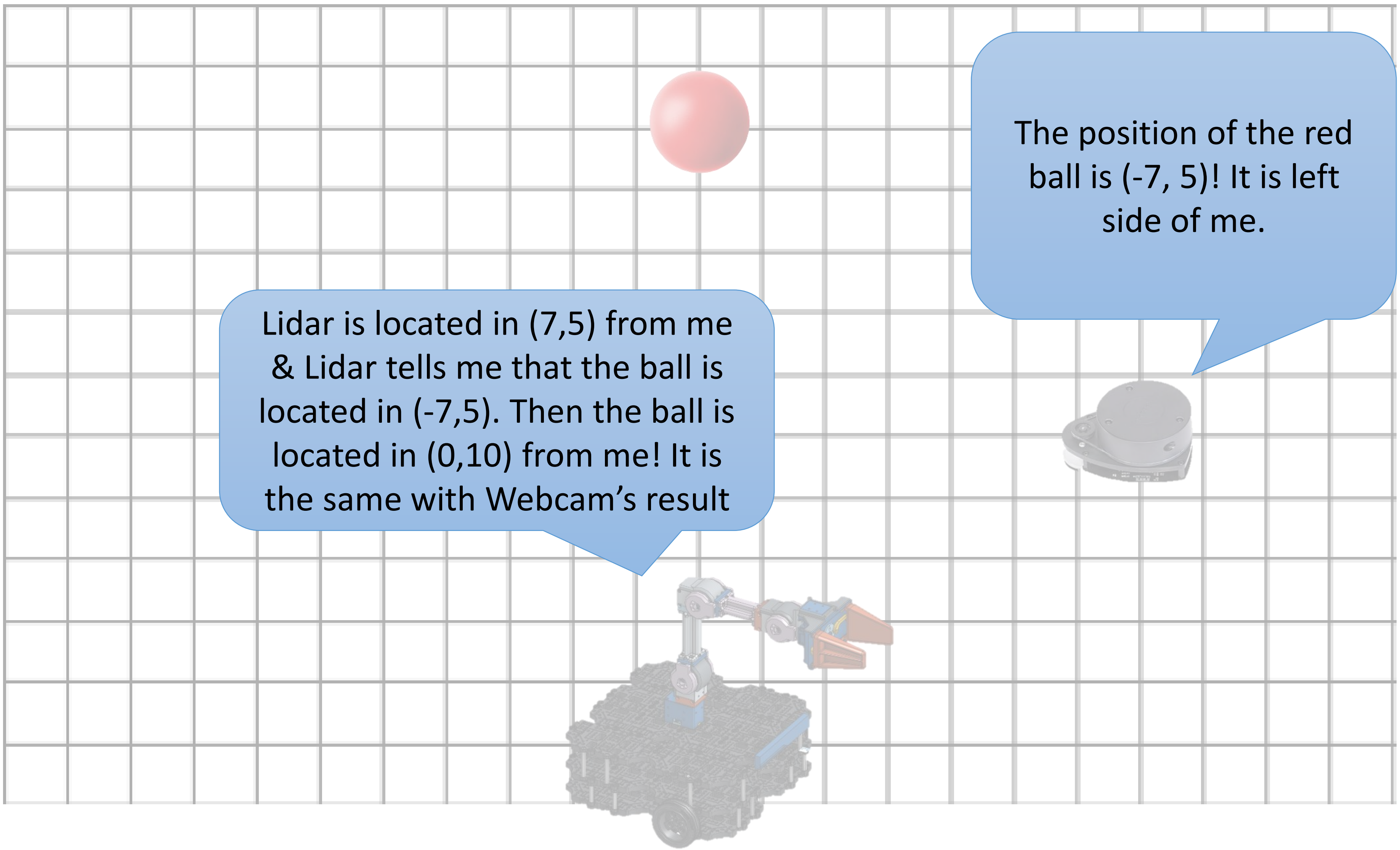
Why we need to understand transformation?

Then, the position of the ball in platform's coordinate can be computed.



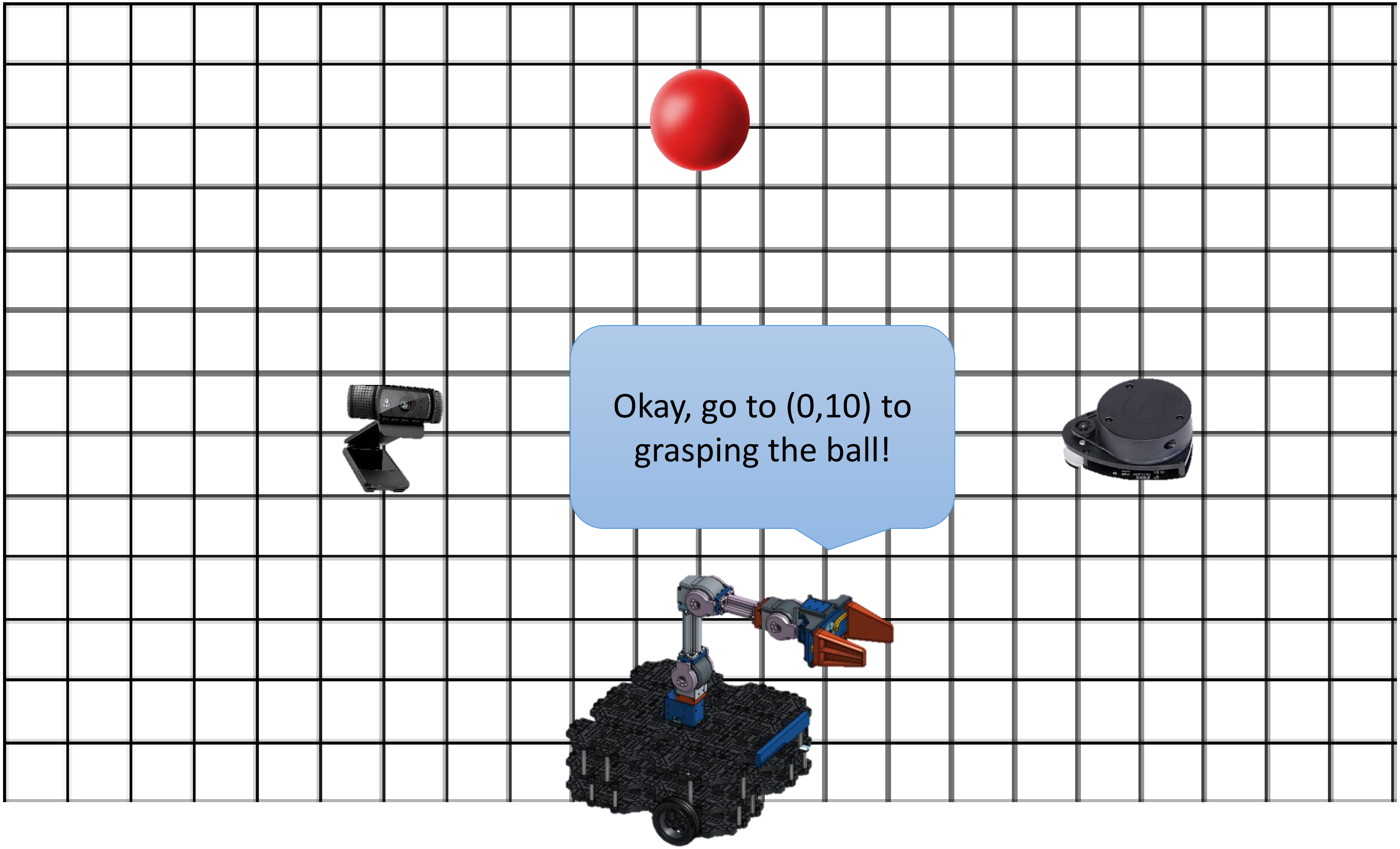
Why we need to understand transformation?

Then, the position of the ball in platform's coordinate can be computed.



Why we need to understand transformation?

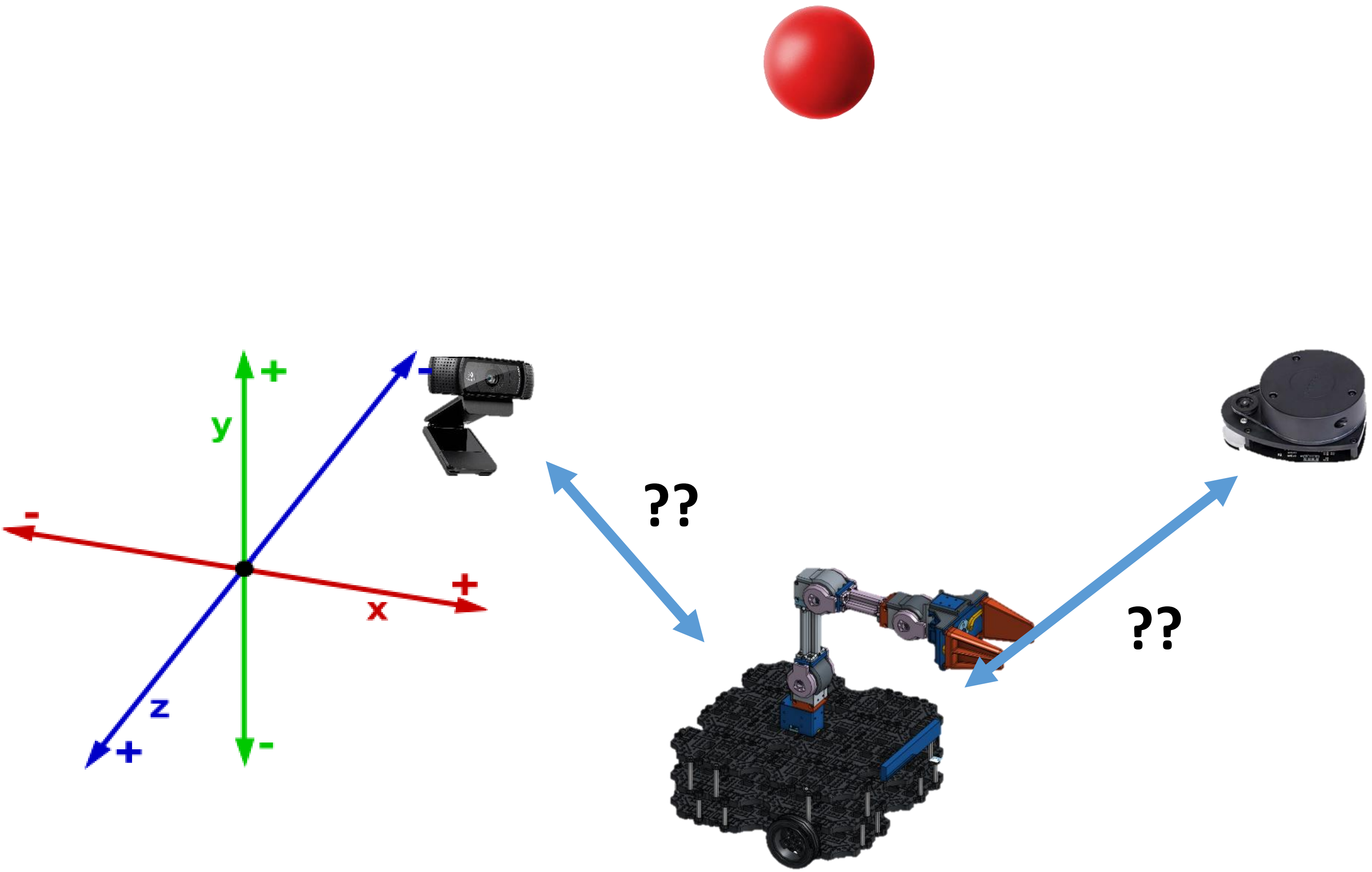
Finally, the decision can be made.



Why we need to understand transformation?

Back to the issue of sensor's locations, how can we represent relations between sensors & mobile platform mathematically?

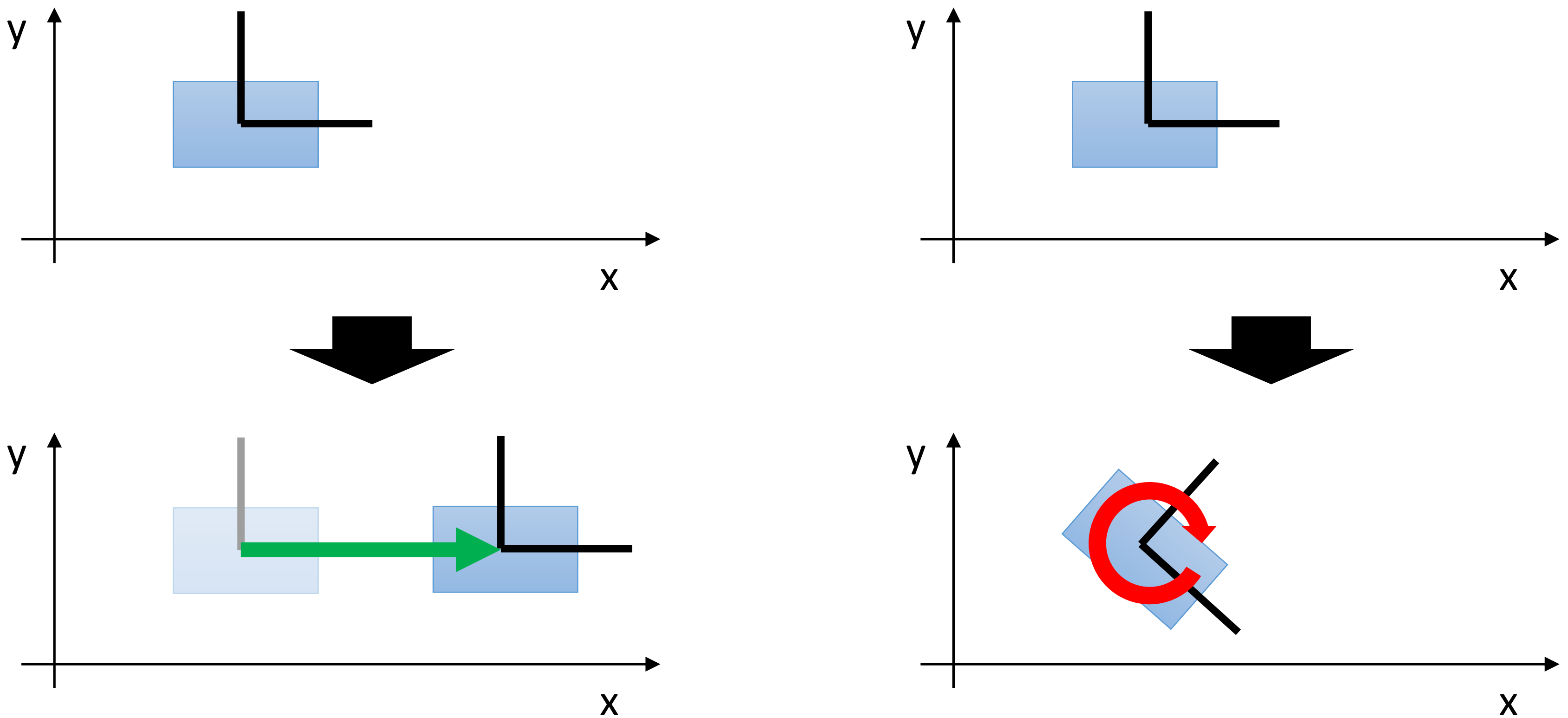
➔ Transformation Matrix



Transformation?

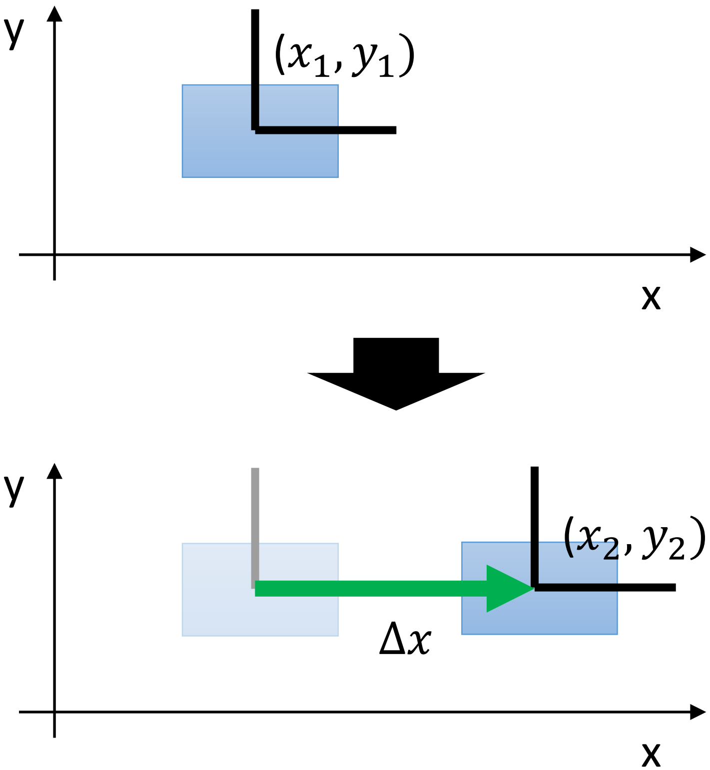
As learnt in Dynamics Lecture,

$$Transformation = \underbrace{Translation}_{\text{Position}} + \underbrace{Rotation}_{\text{Orientation}}$$



Transformation?

We already know how to represent translation & rotation in matrix forms.
In case of translation,



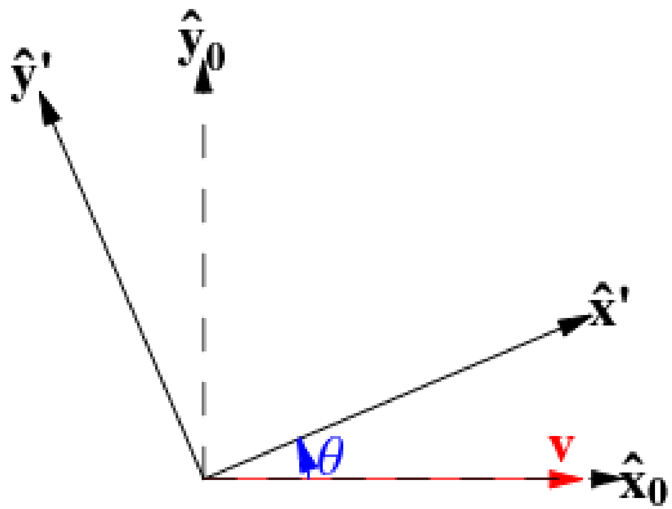
$$x_2 = x_1 + \Delta x$$

$$y_2 = y_1 + \Delta y$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Transformation?

We already know how to represent translation & rotation in matrix forms.
In case of rotation,



$$x_2 = \cos\theta \cdot x_1 - \sin\theta \cdot y_1$$

$$y_2 = \sin\theta \cdot x_1 + \cos\theta \cdot y_1$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

We already know how to represent translation & rotation in matrix forms.
Finally, the transformation in 2D is,

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Position

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Orientation

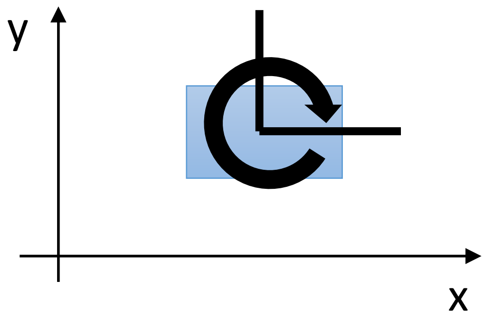
$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$= \boxed{\begin{bmatrix} \cos\theta & -\sin\theta & \Delta x \\ \sin\theta & \cos\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

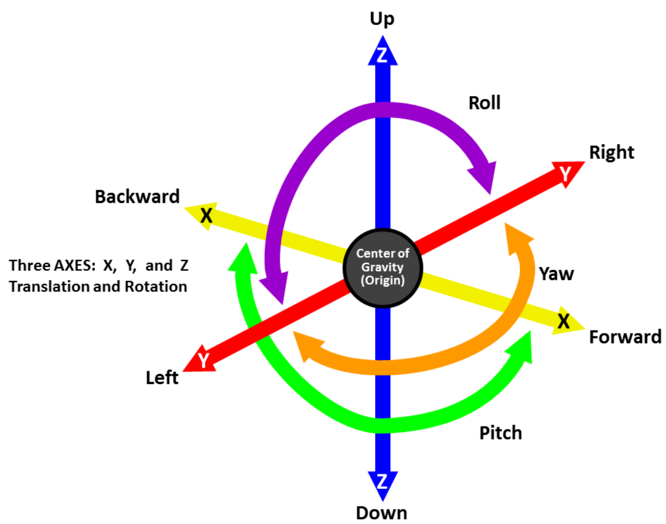
Transformation matrix in 2D

Transformation?

In case of 2D, there are 3 DOF (x, y, theta)



In case of 3D, there are 6 DOF (x, y, z, roll, pitch, yaw)



Transformation?

Again, for translation in 3D

In 2D,

$$x_2 = x_1 + \Delta x$$

$$y_2 = y_1 + \Delta y$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

In 3D,

$$x_2 = x_1 + \Delta x$$

$$y_2 = y_1 + \Delta y$$

$$z_2 = z_1 + \Delta z$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

Transformation?

Again, for rotation in 3D

In 2D,

$$x_2 = \cos\theta \cdot x_1 - \sin\theta \cdot y_1$$

$$y_2 = \sin\theta \cdot x_1 + \cos\theta \cdot y_1$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

In 3D(roll only; that is rotation on x-axis),

$$x_2 = x_2$$

$$y_2 = \cos\theta \cdot y_1 - \sin\theta \cdot z_1$$

$$z_2 = \sin\theta \cdot y_1 + \cos\theta \cdot z_1$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{R_x}$$

Transformation?

Again, for rotation in 3D

In 2D,

$$x_2 = \cos\theta \cdot x_1 - \sin\theta \cdot y_1$$

$$y_2 = \sin\theta \cdot x_1 + \cos\theta \cdot y_1$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

In 3D(pitch only; that is rotation on y-axis),

$$x_2 = \cos\theta \cdot x_1 + \sin\theta \cdot z_1$$

$$y_2 = y_2$$

$$z_2 = -\sin\theta \cdot x_1 + \cos\theta \cdot z_1$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{R_y}} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

Transformation?

Again, for rotation in 3D

In 2D,

$$x_2 = \cos\theta \cdot x_1 - \sin\theta \cdot y_1$$

$$y_2 = \sin\theta \cdot x_1 + \cos\theta \cdot y_1$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

In 3D(yaw only; that is rotation on z-axis),

$$x_2 = \cos\theta \cdot x_1 - \sin\theta \cdot y_1$$

$$y_2 = \sin\theta \cdot x_1 + \cos\theta \cdot y_1$$

$$z_2 = z_2$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{R_z}$$

Transformation?

Finally, the transformation in 3D is,

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

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

$\boldsymbol{R_x}$ $\boldsymbol{R_y}$ $\boldsymbol{R_z}$

Position

Orientation

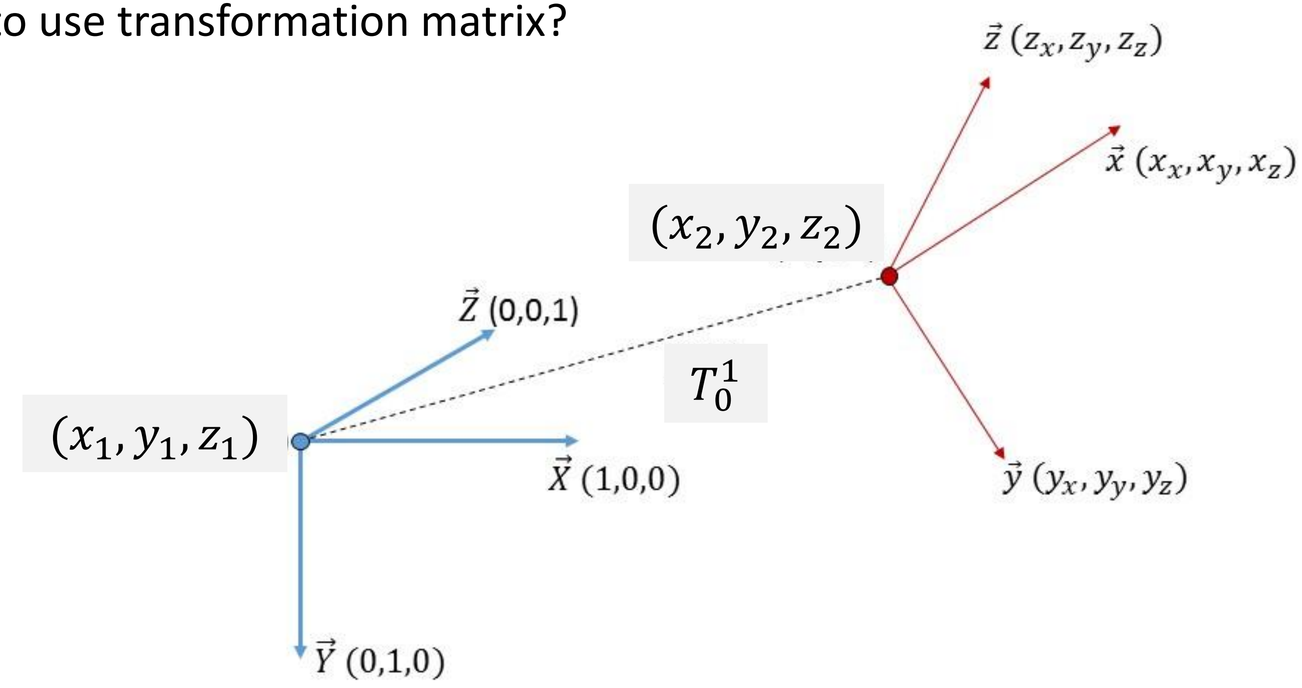
$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot R_x \cdot R_y \cdot R_z \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} =$$

I am too lazy to type this…
Compute by yourself!

The result of this matrix computation is called
“Transformation matrix”

Transformation?

How to use transformation matrix?

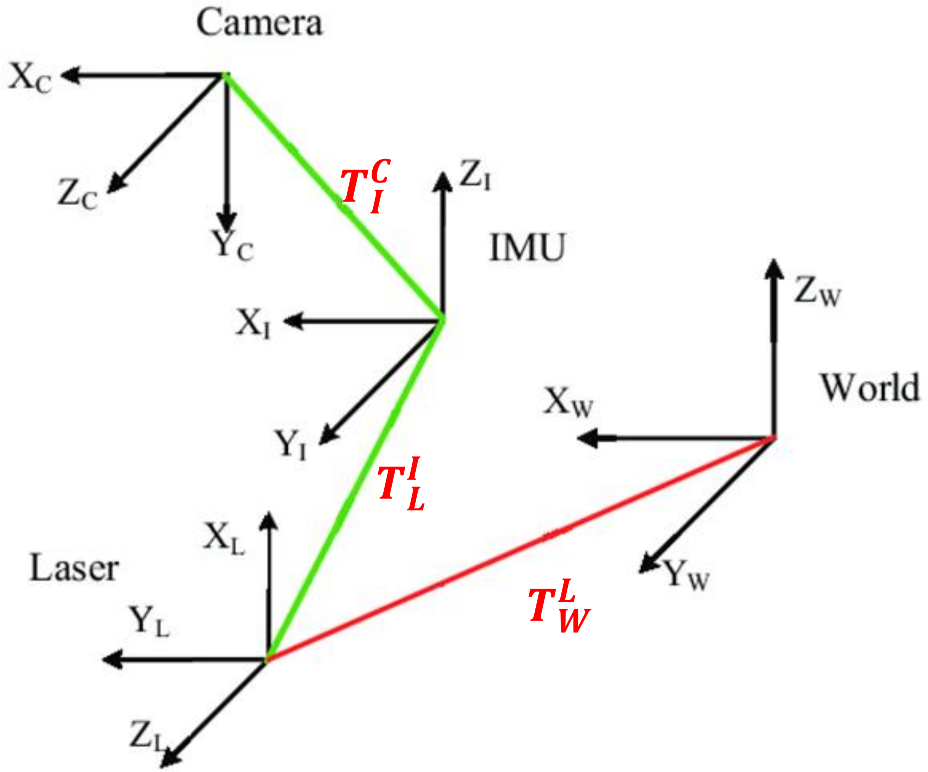


$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = T_1^2 \cdot \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

$$X_2 = T_1^2 \cdot X_1$$

Transformation?

How to use transformation matrix?



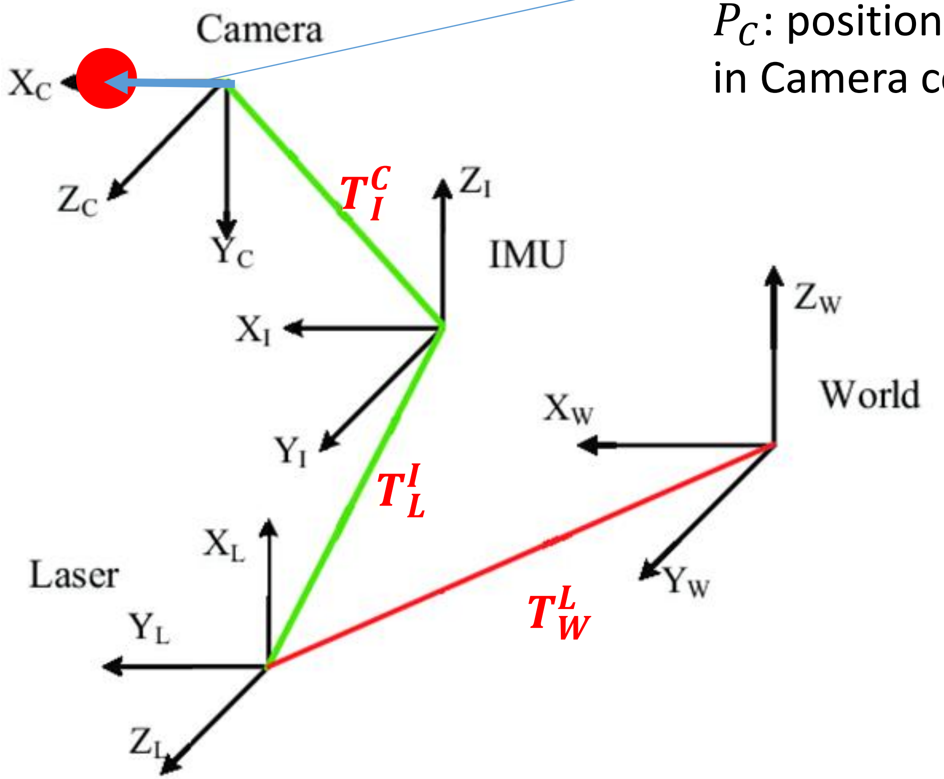
T_I^C : transformation from IMU to Camera

T_L^I : transformation from Laser to IMU

T_W^L : transformation from World to Laser

Transformation?

How to use transformation matrix?

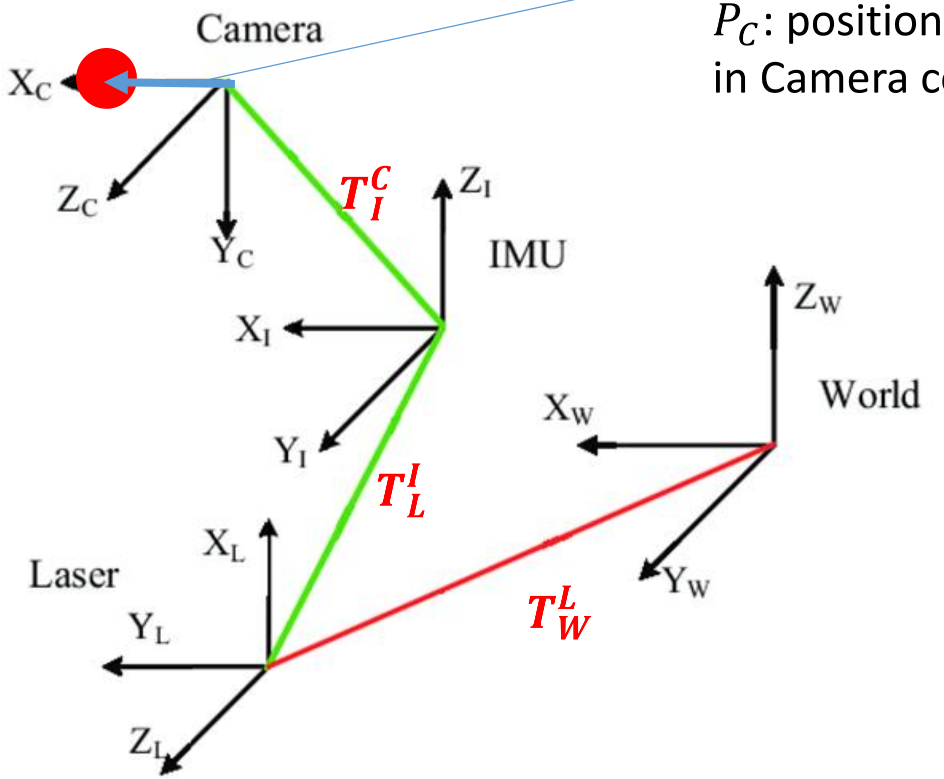


P_C: position of the ball in Camera coordinate

Q. What is the position & transformation of the red ball in "laser coordinate"?

Transformation?

How to use transformation matrix?



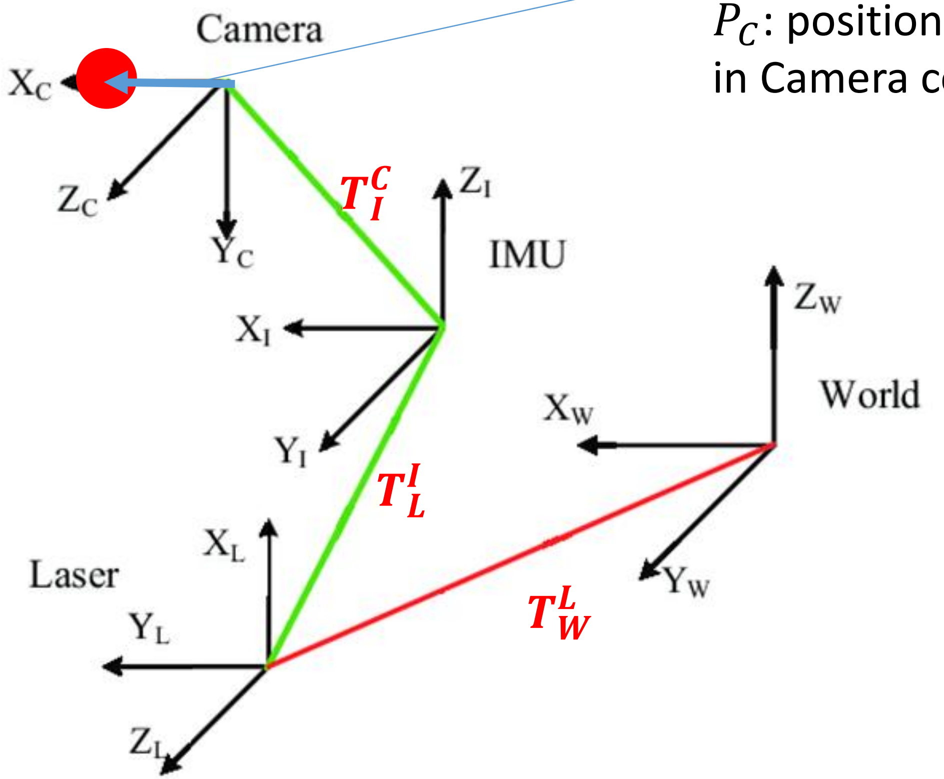
P_C: position of the ball in Camera coordinate

Q. What is the position & transformation of the red ball in “laser coordinate”?

$$P_L = T_L^I \cdot T_I^C \cdot P_C$$

Transformation?

How to use transformation matrix?

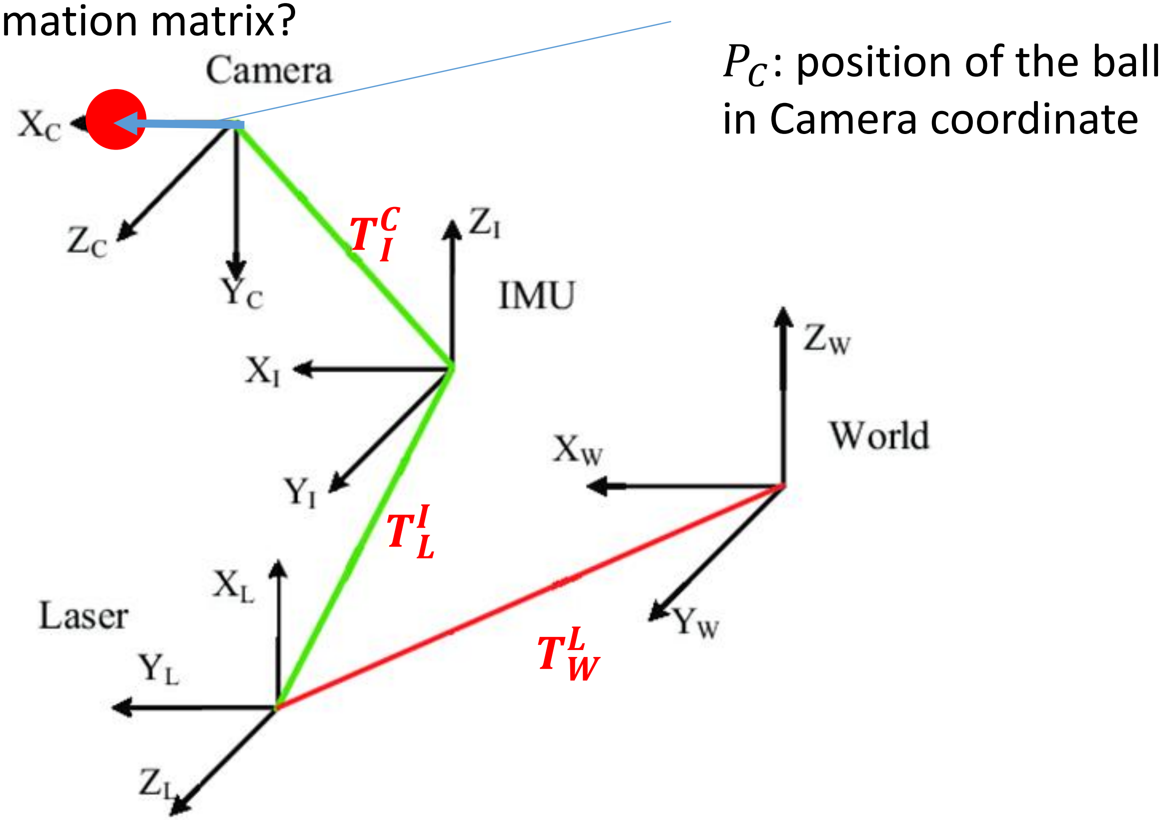


P_C: position of the ball in Camera coordinate

Q. The position of red ball is P_W in "World coordinate" while the position in "Camera coordinate" is P_C. How you can compute positional error of camera-based ball tracking?

Transformation?

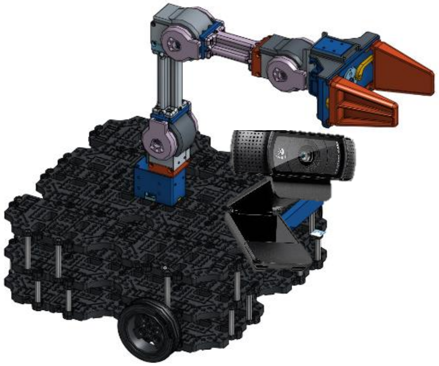
How to use transformation matrix?



Q. The position of red ball is P_W in "World coordinate" while the position in "Camera coordinate" is P_C. How you can compute positional error of camera-based ball tracking?

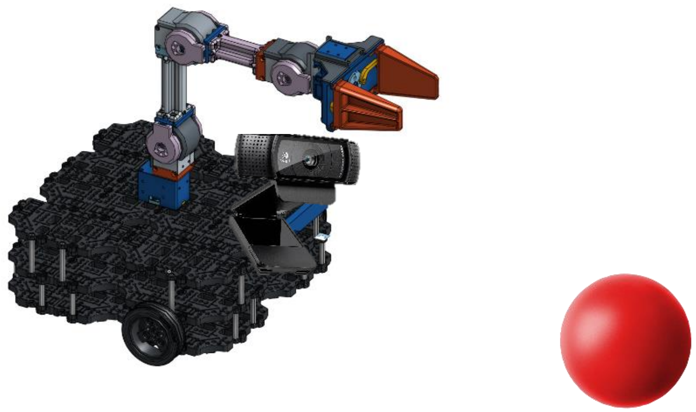
$$Error = P_W - T_W^L \cdot T_L^I \cdot T_I^C \cdot P_C$$

Transformation?



Q. How many coordinate(or frame) does your platform involve?

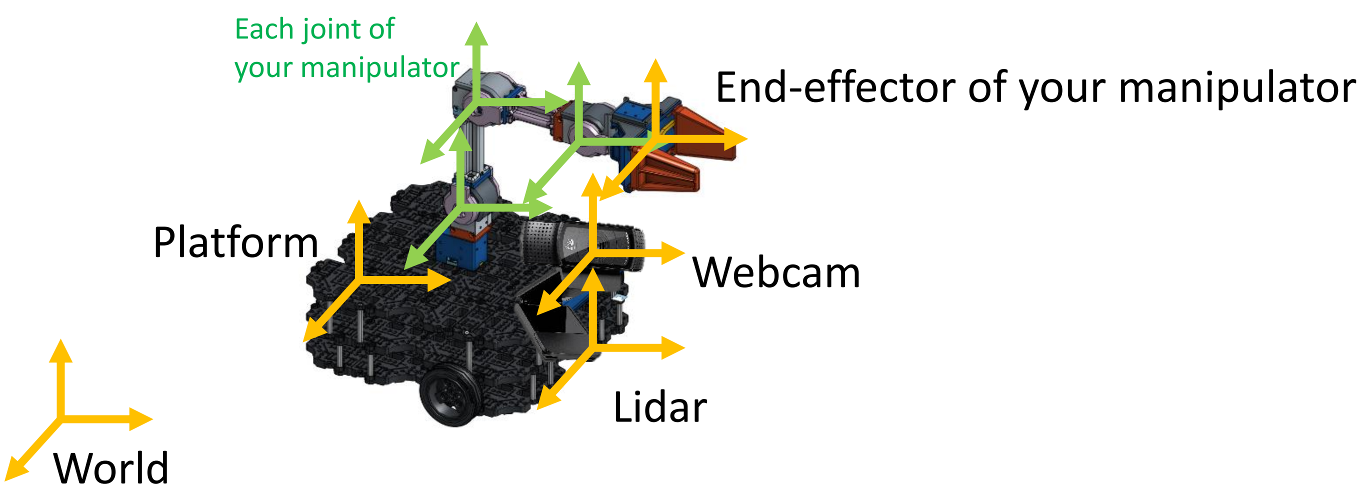
Transformation?



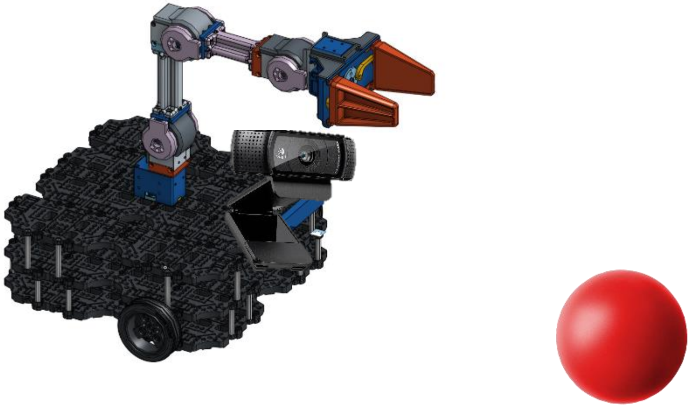
Q. How many coordinate(or frame) does your platform involve?

A. It depends on your configuration.

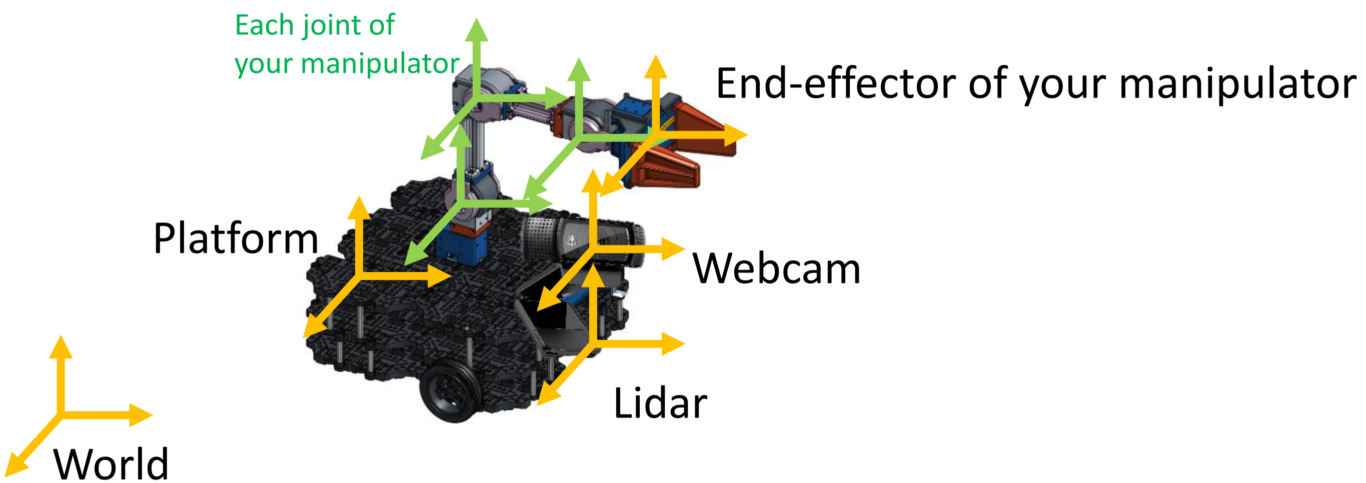
As an example,



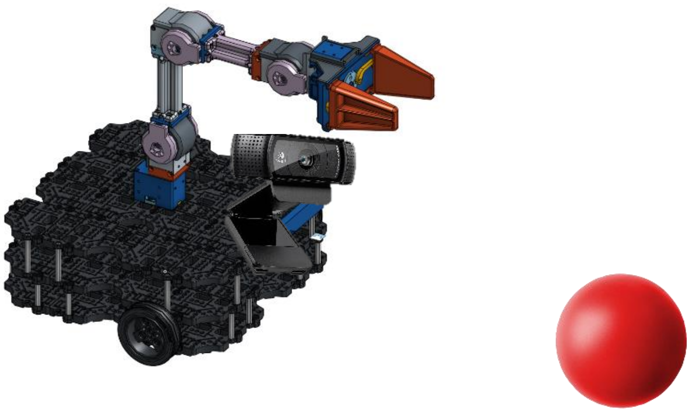
Transformation?



Q. What is the difference between “world-platform” and “platform-webcam”?

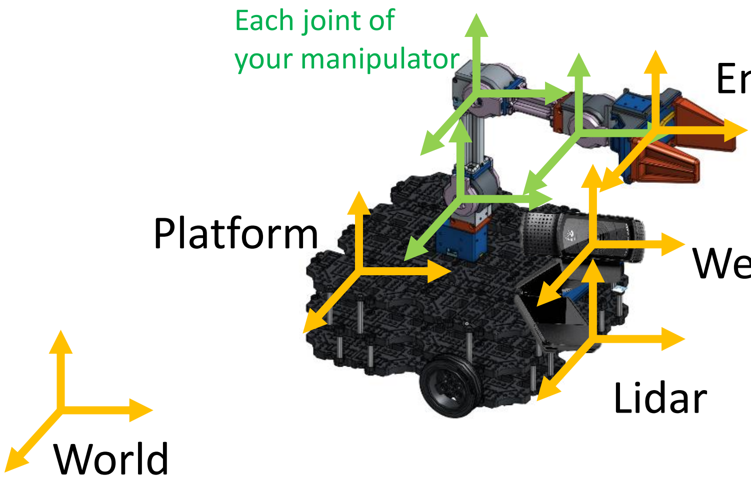


Transformation?

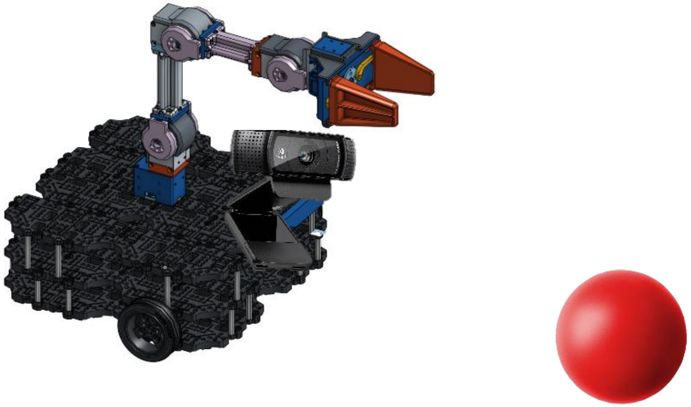


Q. What is the difference between "world-platform" and "platform-webcam"?

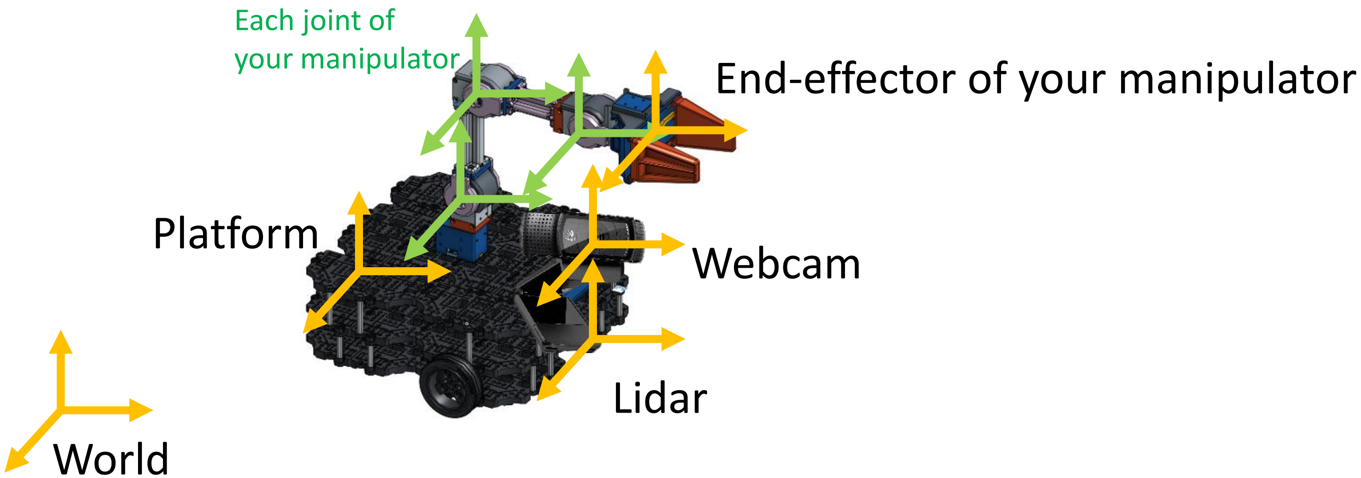
A. Transformation between platform-webcam is **static**(it doesn't change as time goes)
Transformation between world-platform is **dynamic**(it will be changed as time goes when moving)



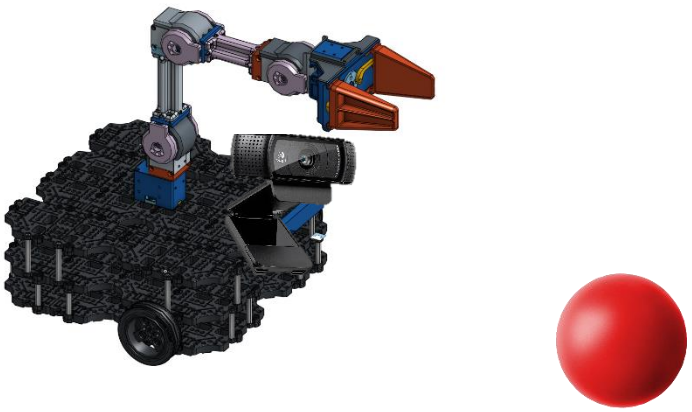
Transformation?



Q. Decide whether frames in the image are static or dynamic.

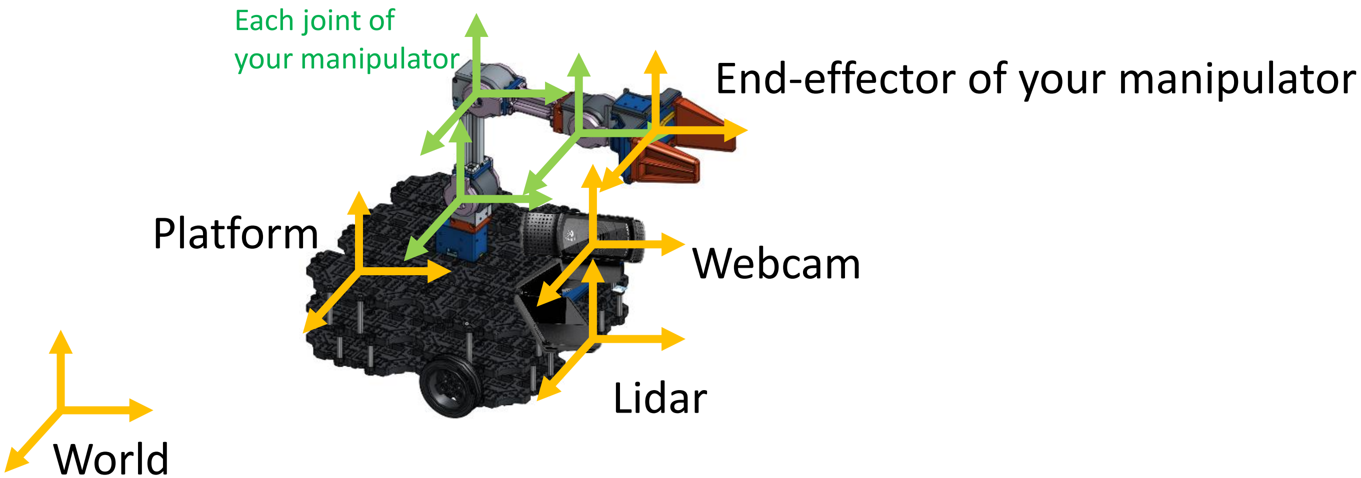


Transformation?



Q. Decide whether frames in the image are static or dynamic.

A. Dynamic – “World-Platform”, “Motor1-Motor2”, “Motor2-Motor3”, …
Static – “Platform-Lidar”, “Platform-Webcam”, “Motor_last-Endeffector”, …



How to apply transformation in ROS?

-Download a source code via github
  - git clone https://githubcom/kaistmecd

-In the folder named “tf_example”, there are 4 nodes.

-Read a file “README.md” and follow it.

-You can check a attached video.

-Read and study codes in this order(*descriptions are included in *.cpp files)
1. fake_bal_in_rviz
2. static_tf_example
3. compute_position_in_other_frame

And then,

4. dynamic_tf_example (*actually, it is very similar to the ‘static_tf_example’)