Kinematics of Robot Manipulators

Class at Randolph College

Yue Cao
Purdue University, ECE
yuecao@purdue.edu

Terminology – Manipulator

Link

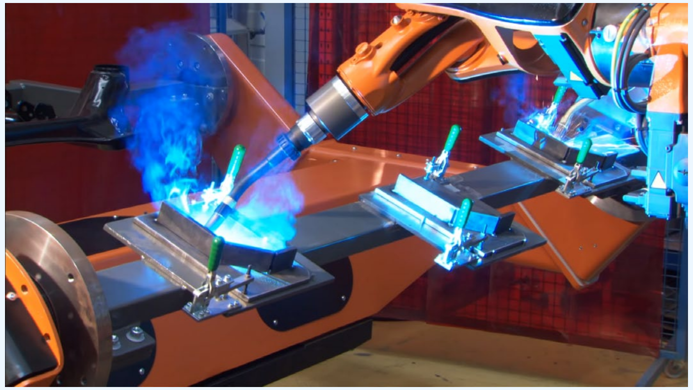
Joint

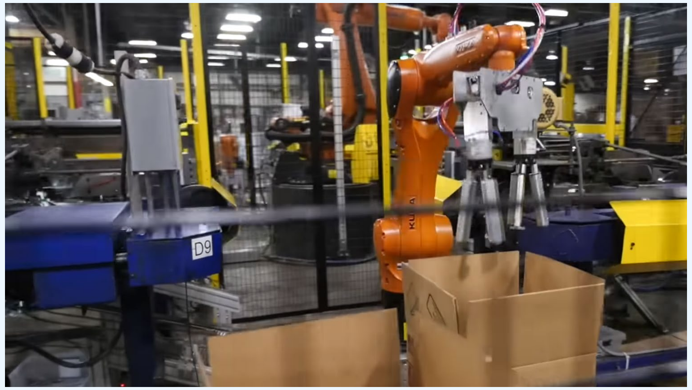
End-effector

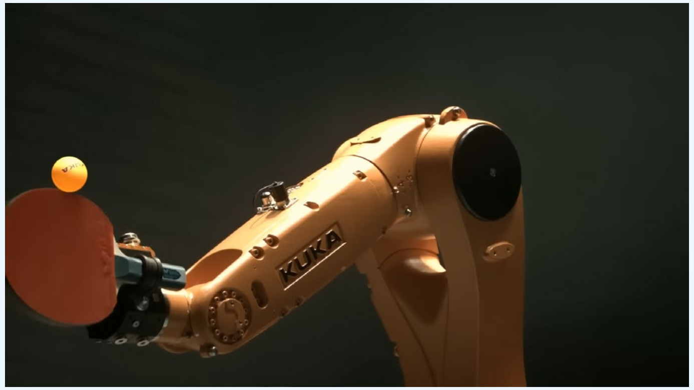
Base

Terminology – Manipulator

A variety of end-effectors







Definition – Kinematics & Dynamics

Kinematics

The study of motion without regard for the force

Dynamics

The study of how force affects the motion

Definition – Kinematics & Dynamics

Kinematics

The study of motion without regard for the force.

For manipulators:

Joint angles ↔ End-effector position & orientation

Dynamics

The study of how force affects the motion.

For manipulators:

Joint torques ↔ End-effector position, velocity, acceleration

Problem Formulation – Kinematics

Two sub-topics in kinematics:

1. Forward Kinematics

Joint angles → End-effector position & orientation

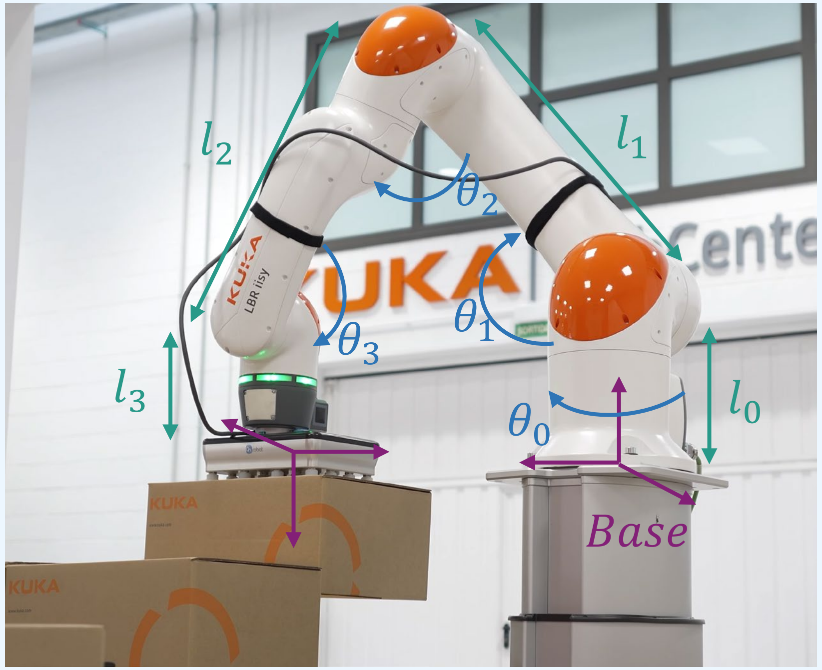
Given: joint angle $\theta_0, \theta_1, \theta_2, \ldots$, link length $l_0, l_1, l_2, \ldots$

Aim for: end-effector $[x\ y\ z\ \alpha\ \beta\ \gamma]$

$x\ y\ z$: position in the Cartesian coordinate.

$\alpha\ \beta\ \gamma$: orientation in Euler angles (roll, pitch, yaw).

Base is the reference.



Problem Formulation – Kinematics

Two sub-topics in kinematics:

2. Backward Kinematics

Joint angles ← End-effector position & orientation

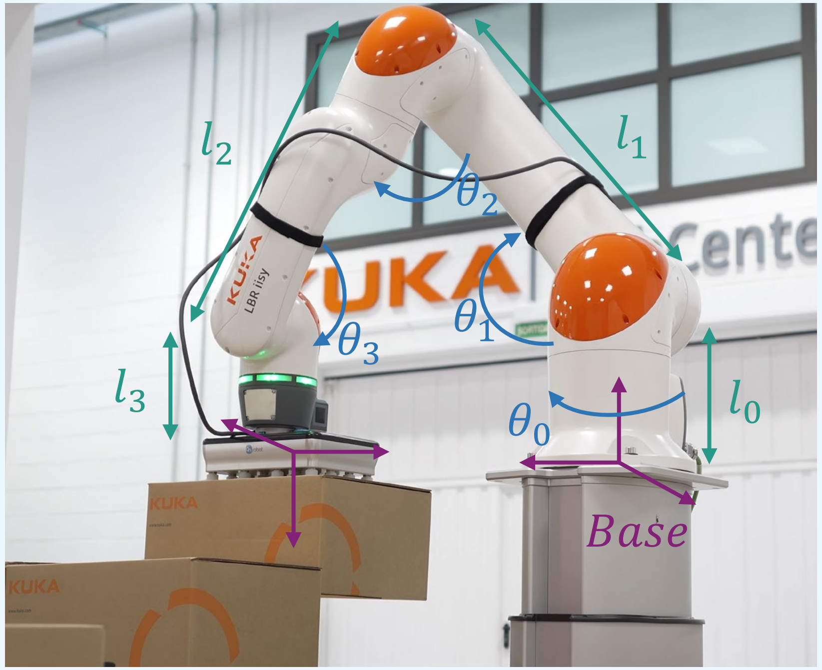
Given: end-effector $[x\ y\ z\ \alpha\ \beta\ \gamma]$, link length $l_0, l_1, l_2, \ldots$

Aim for: joint angle $\theta_0, \theta_1, \theta_2, \ldots$

$x\ y\ z$: position in the Cartesian coordinate.

$\alpha\ \beta\ \gamma$: orientation in Euler angles (roll, pitch, yaw).

Base is the reference.



Start with 2D Plane

Forward Kinematics

3D Space

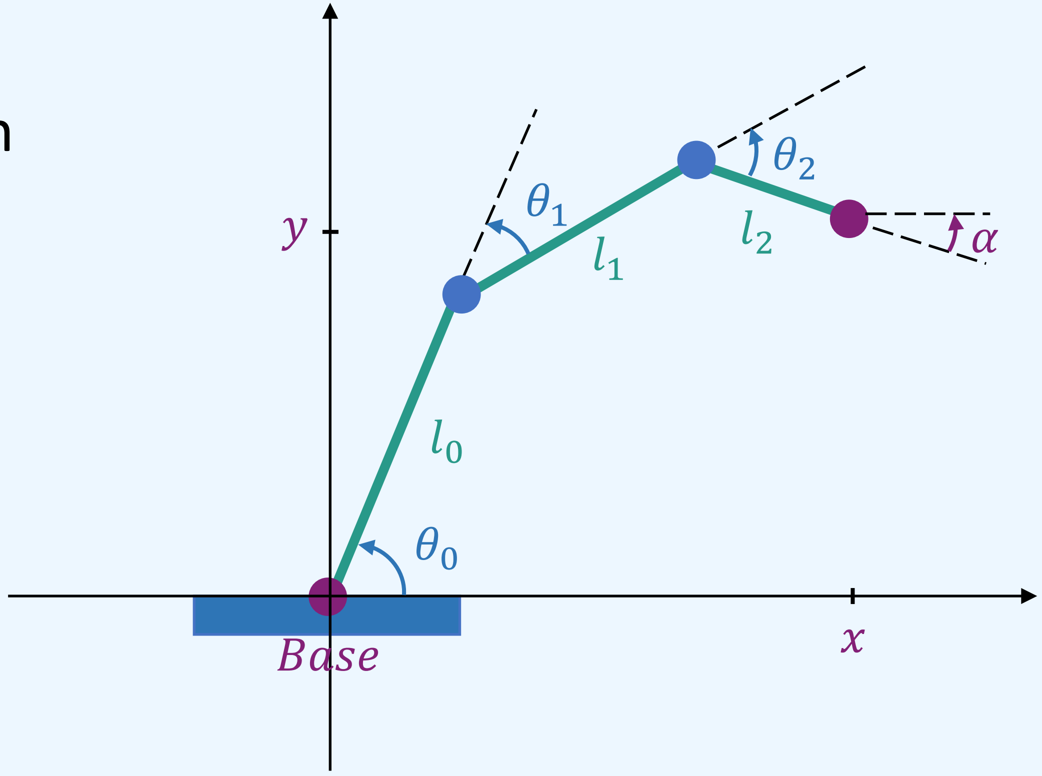
Given: joint angle $\theta_0, \theta_1, \theta_2, \dots$, link length $l_0, l_1, l_2, \dots$

Aim for: end-effector $[x\ y\ z\ \alpha\ \beta\ \gamma]$

2D Plane

Given: joint angle $\theta_0, \theta_1, \theta_2, \dots$, link length $l_0, l_1, l_2, \dots$

Aim for: end-effector $[x\ y\ \alpha]$

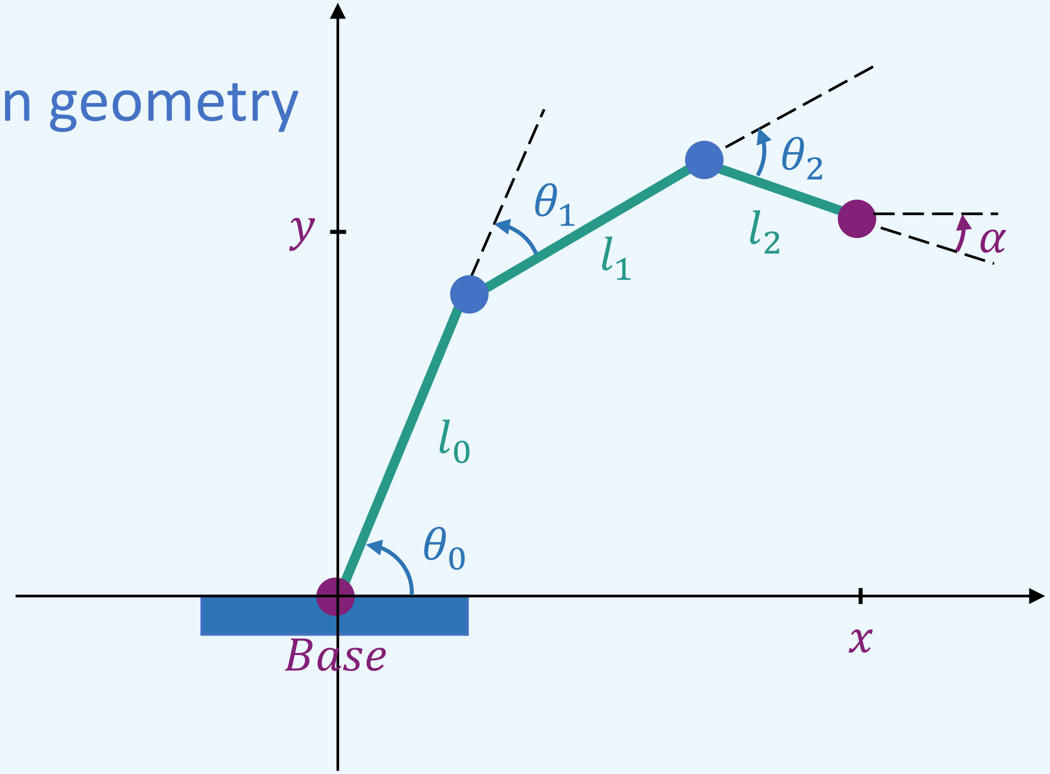


$[x\ y\ \alpha]$ is with respect to the base coordinate frame.

2D Forward Kinematics

How to obtain $[x\ y\ \alpha]$?

Approach 1: Directly analyze the Euclidean geometry



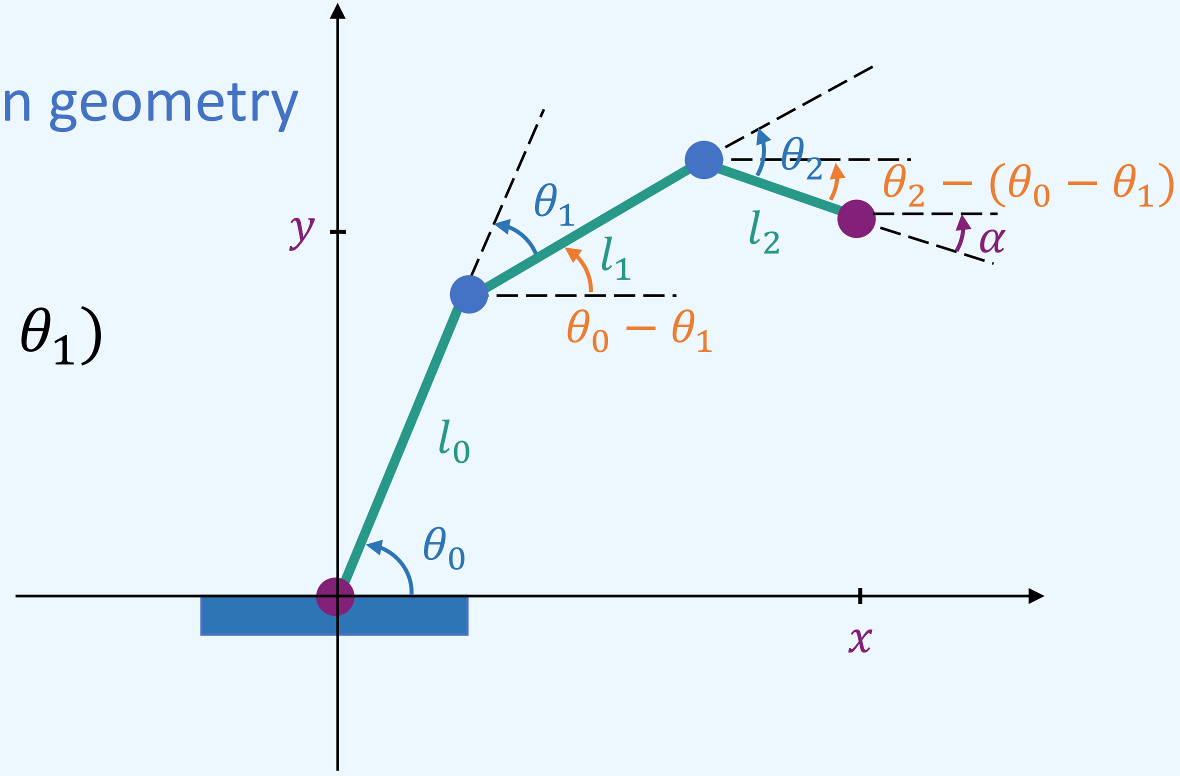
2D Forward Kinematics

How to obtain $[x\ y\ \alpha]$?

Approach 1: Directly analyze the Euclidean geometry

Link 1 angle w.r.t to the base: $\theta_0 - \theta_1$
Link 2 angle w.r.t to the base: $\theta_2 - (\theta_0 - \theta_1)$

We now have, $\alpha = \theta_2 - (\theta_0 - \theta_1)$



w.r.t: with respect to, means use the specified coordinate frame as a reference

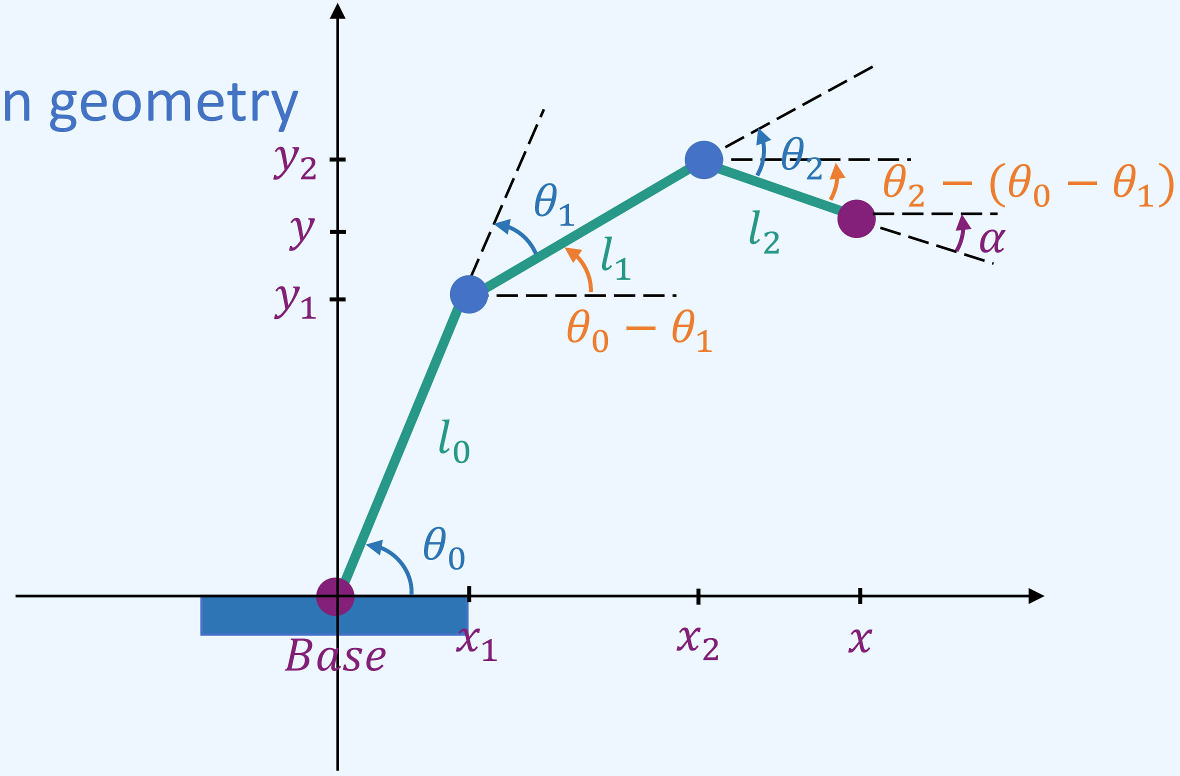
2D Forward Kinematics

How to obtain $[x\ y\ \alpha]$?

Approach 1: Directly analyze the Euclidean geometry

Link 1 left-end position:

$x_1 = l_0 C(\theta_0), y_1 = l_0 S(\theta_0)$



$S(\theta)$ is short for $sin\ (\theta)$, $C(\theta)$ is short for $cos\ (\theta)$

2D Forward Kinematics

How to obtain $[x\ y\ \alpha]$?

Approach 1: Directly analyze the Euclidean geometry

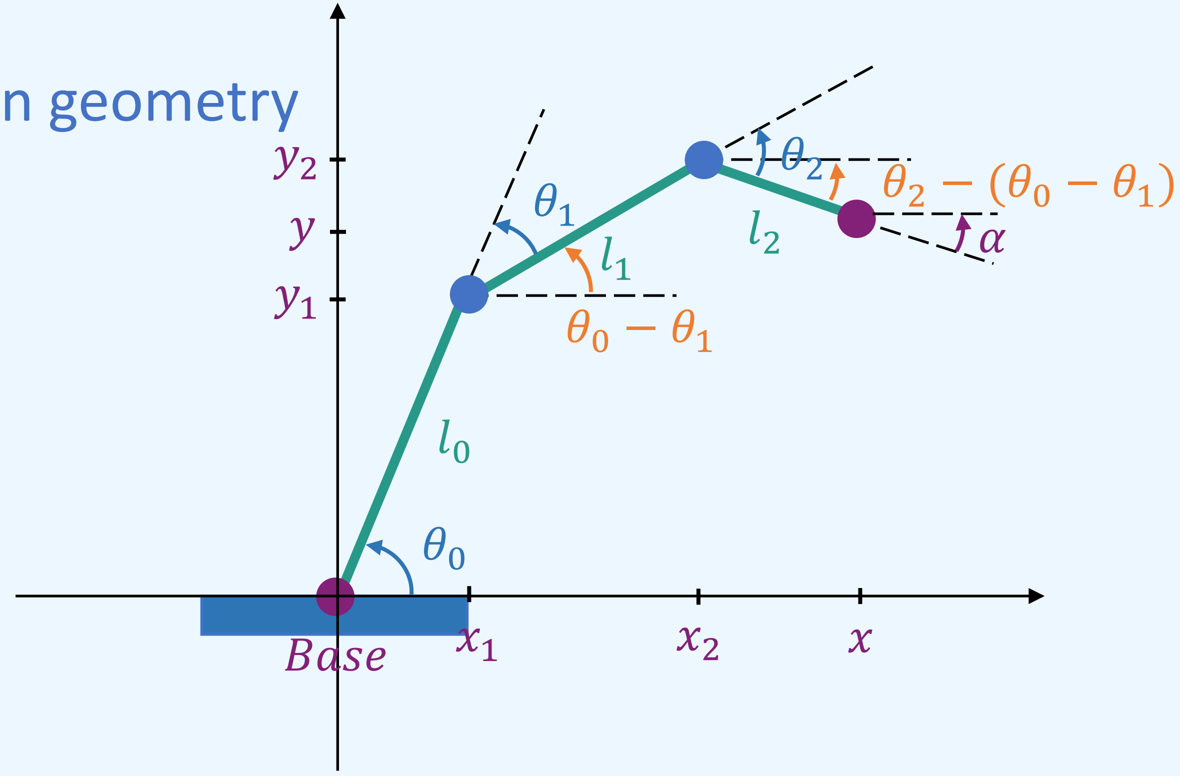
Link 1 left-end position:

$x_1 = l_0 C(\theta_0), y_1 = l_0 S(\theta_0)$

Link 2 left-end position:

$x_2 = l_0 C(\theta_0) + l_1 C(\theta_0 - \theta_1),$

$y_1 = l_0 S(\theta_0) + l_1 S(\theta_0 - \theta_1)$

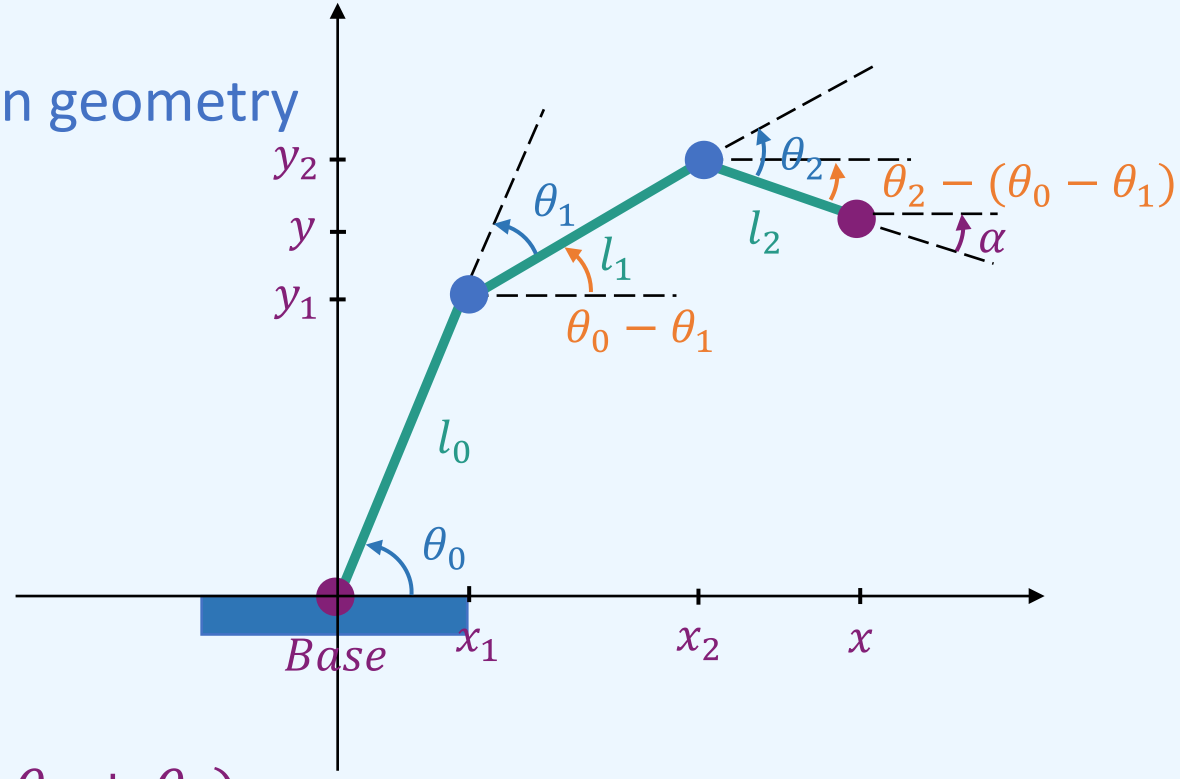


$S(\theta)$ is short for $sin\ (\theta)$, $C(\theta)$ is short for $cos\ (\theta)$

2D Forward Kinematics

How to obtain $[x\ y\ \alpha]$?

Approach 1: Directly analyze the Euclidean geometry



Link 1 left-end position:

$x_1 = l_0C(\theta_0), y_1 = l_0S(\theta_0)$

Link 2 left-end position:

$x_2 = l_0C(\theta_0) + l_1C(\theta_0 - \theta_1),$

$y_1 = l_0S(\theta_0) + l_1S(\theta_0 - \theta_1)$

We now have,

$x = l_0C(\theta_0) + l_1C(\theta_0 - \theta_1) + l_2C(\theta_2 - \theta_0 + \theta_1),$

$y = l_0S(\theta_0) + l_1S(\theta_0 - \theta_1) - l_2S(\theta_2 - \theta_0 + \theta_1)$

$S(\theta)$ is short for $sin\ (\theta)$, $C(\theta)$ is short for $cos\ (\theta)$

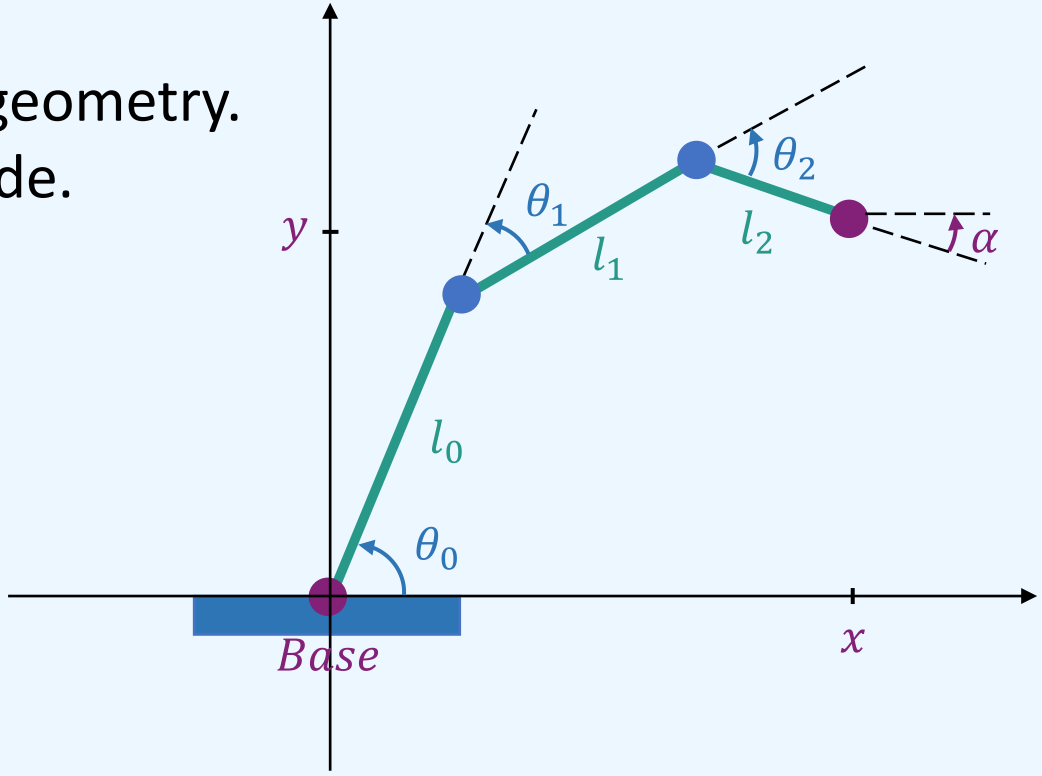
2D Forward Kinematics

How to obtain $[x\ y\ \alpha]$?

But robots cannot analyze the Euclidean geometry.
Robots are essentially programmed by code.

Can we make something like this code?

```
link_length_0 = float( )
link_length_1 = float( )
...
def my_kinematics(joint_angle_0, joint_angle_1, ...):
    ............
    ............
    end_effector_pos=...
    end_effector_ori=...
```



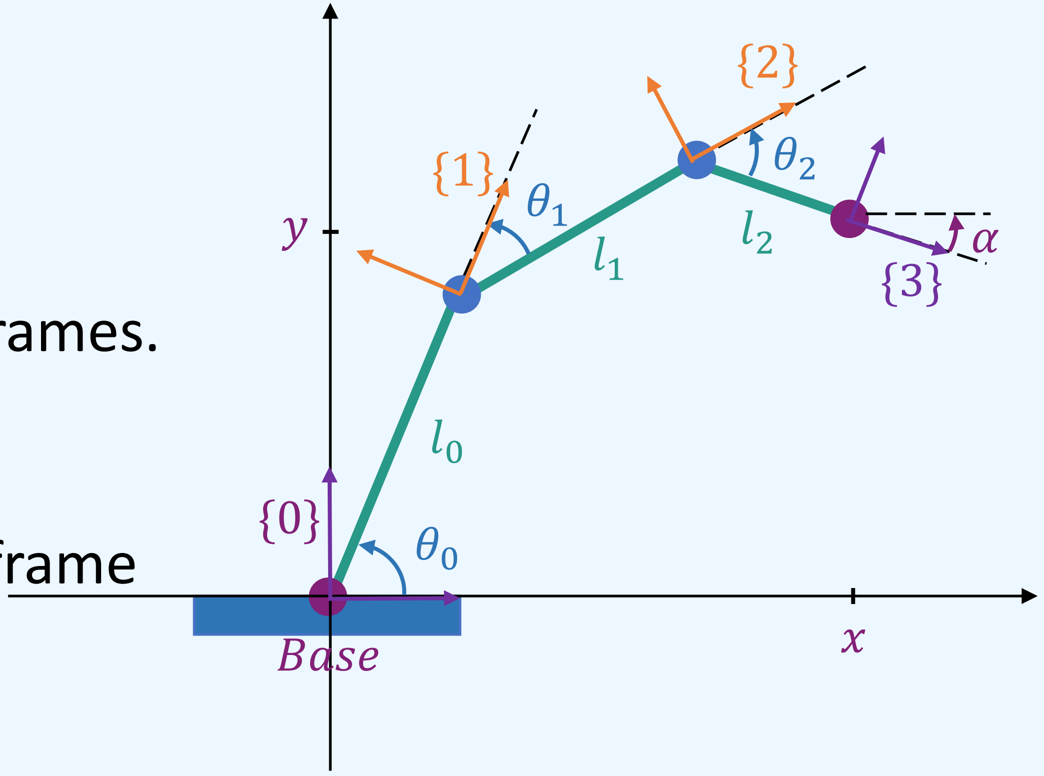
2D Forward Kinematics

How to obtain $[x\ y\ \alpha]$?

Approach 2: Transformation Matrix

Set a coordinate frame { } for each joint,
then build transformation between two frames.

Here, we eventually want to obtain ${}^{0}_{3}T$,
that is, how to describe the end-effector frame
w.r.t the base frame.



${}^{A}_{B}T$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

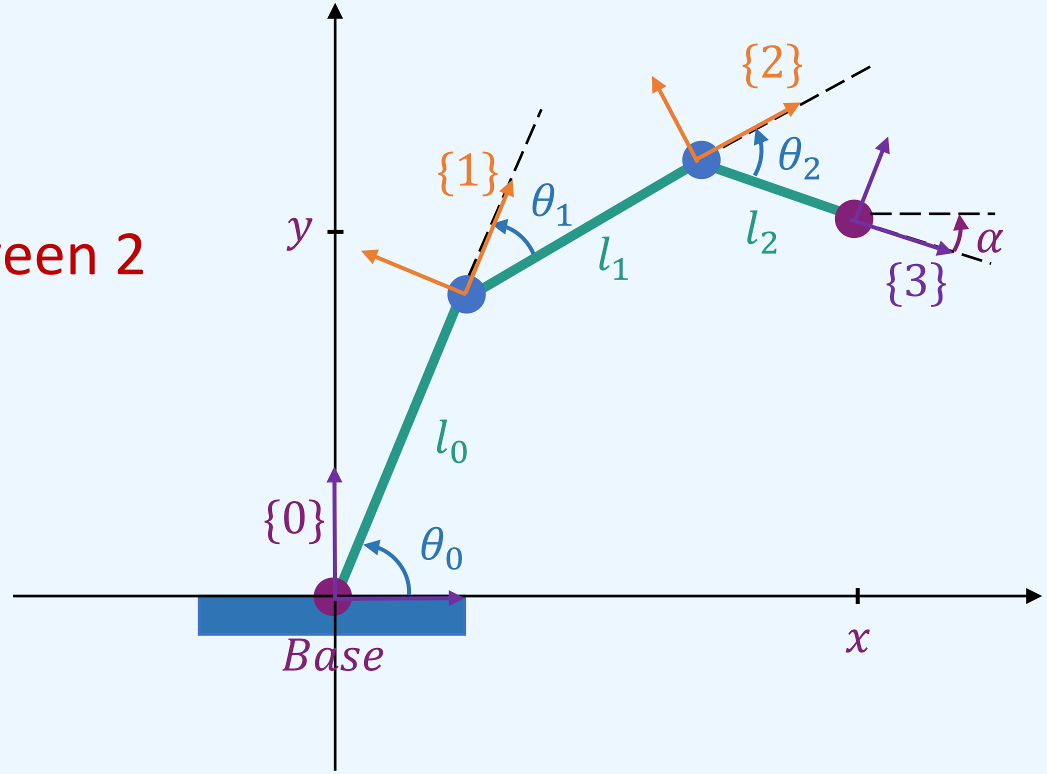
How to obtain $[x\ y\ \alpha]$?

Approach 2: Transformation Matrix

How to describe the transformation between 2 frames?

1. Translation transform

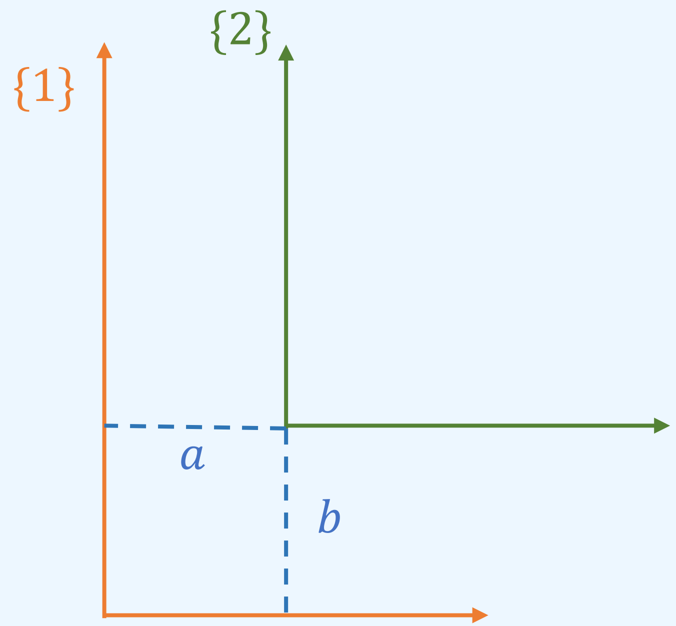
2. Rotation transform



2D Forward Kinematics

Translation transform

What will be ${}^{1}_{2}T$?



${}^{A}_{B}T$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

Translation transform

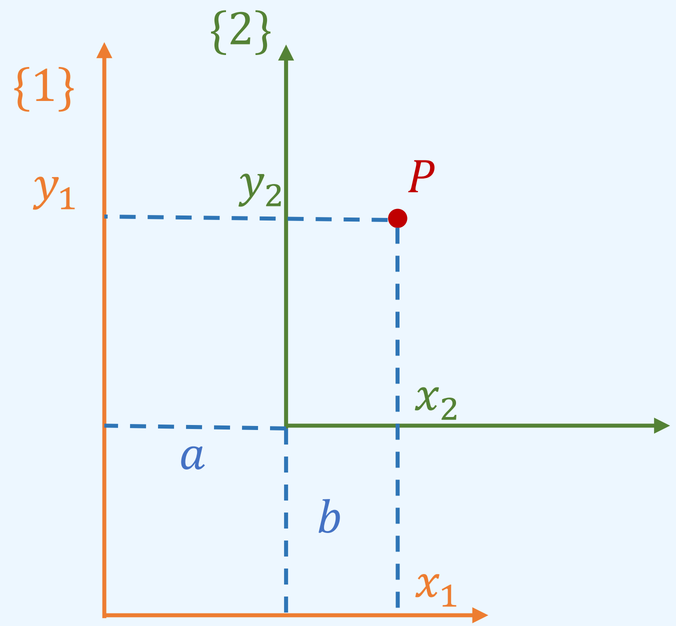
What will be ${}^{1}_{2}T$?

Consider an arbitrary point P,
its position at $\{1\}$ is (x_1, y_1)
its position at $\{2\}$ is (x_2, y_2)
Then,

$$x_1 = x_2 + a, y_1 = y_2 + b$$

Write in vector,

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$$



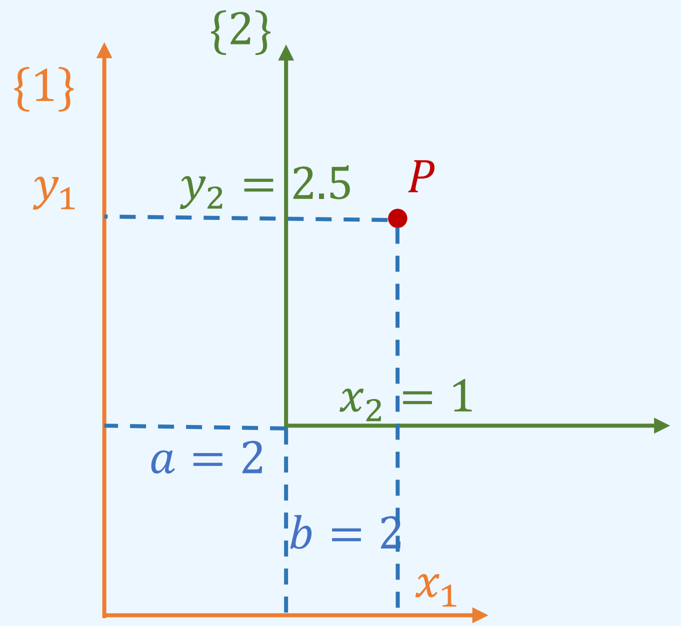
${}^{A}_{B}T$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

Translation transform

Example

Given P at $\{2\}$ is $(1, 2.5\ )$, $a = b = 2$



Then,

$$x_1 = 1 + a, y_1 = 2.5 + b$$

Write in vector,

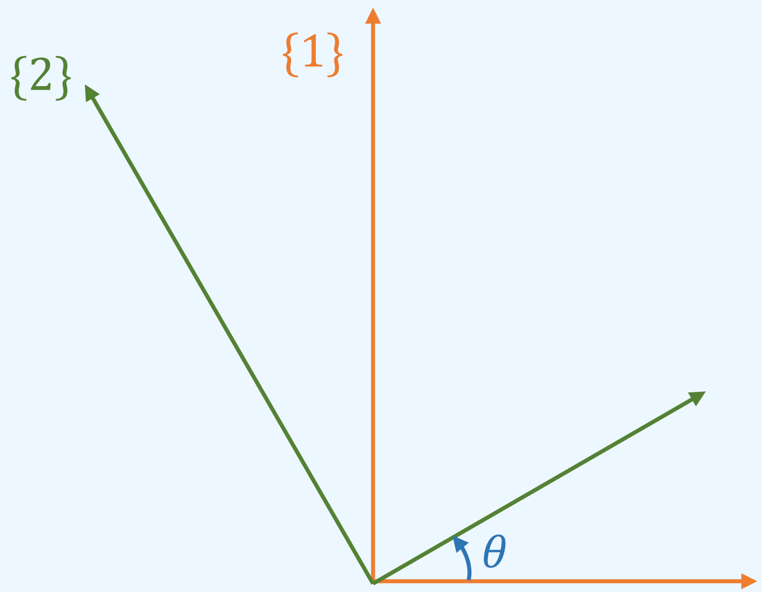
$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2.5 \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = = \begin{bmatrix} 1 \\ 2.5 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4.5 \end{bmatrix}$$

${}^{A}_{B}T$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

Rotation transform

What will be ${}^{1}_{2}T$?



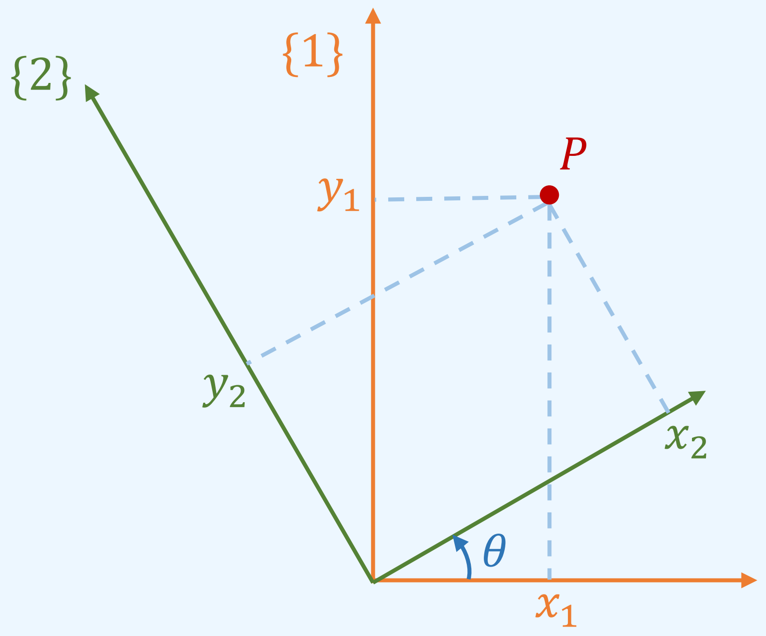
${}^{A}_{B}T$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

Rotation transform

What will be ${}^{1}_{2}T$?

Consider an arbitrary point P,
its position at $\{1\}$ is (x_1, y_1)
its position at $\{2\}$ is (x_2, y_2)



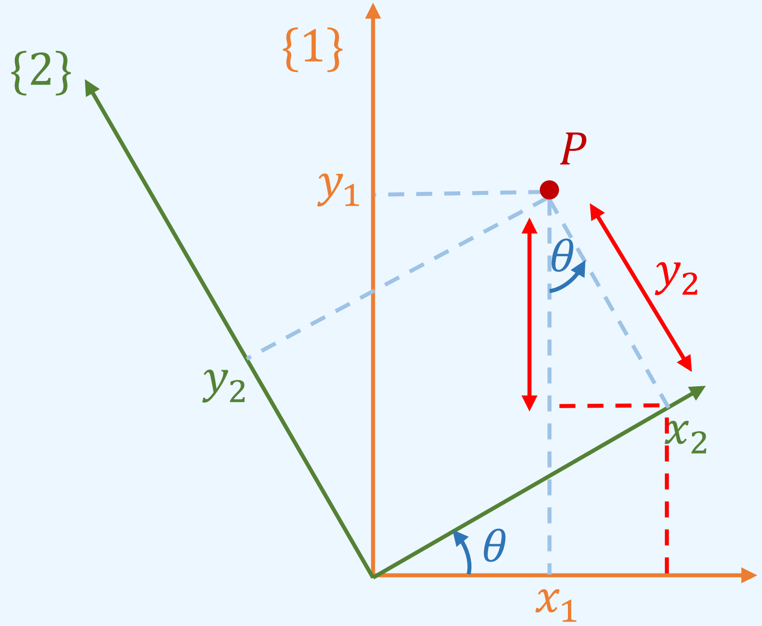
${}^{A}_{B}T$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

Rotation transform

What will be ${}^1_2T$?

Consider an arbitrary point P,
its position at $\{1\}$ is (x_1, y_1)
its position at $\{2\}$ is (x_2, y_2)



$$x_1 = x_2C(\theta) - y_2S(\theta), \;\; y_1 = x_2S(\theta) + y_2C(\theta)$$

Write in vector,

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) \\ S(\theta) & C(\theta) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

$S(\theta)$ is short for $sin\ (\theta)$, $C(\theta)$ is short for $cos\ (\theta)$

2D Forward Kinematics

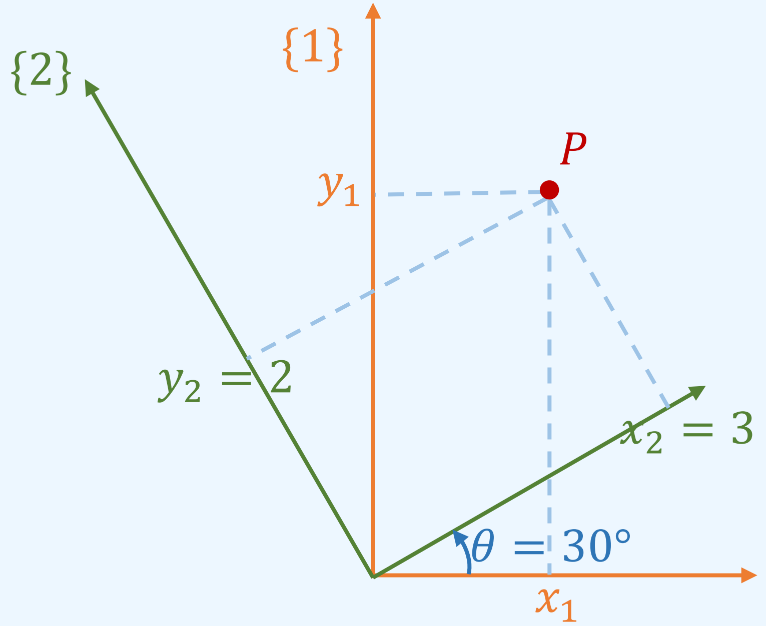
Rotation transform

Example

Given P at $\{2\}$ is $(3, 2)$, θ=30°

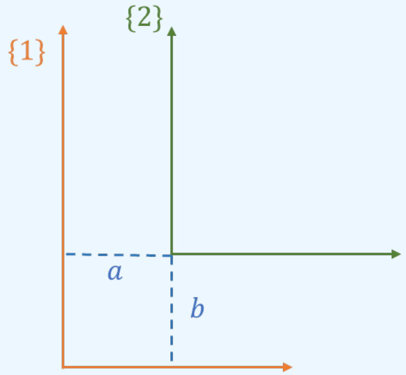
$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) \\ S(\theta) & C(\theta) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} C(30°) & -S(30°) \\ S(30°) & C(30°) \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1.60 \\ 3.23 \end{bmatrix}$$



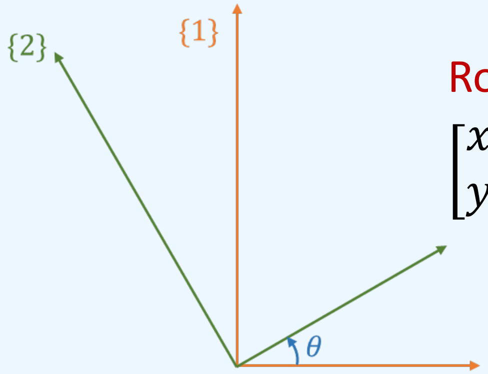
$S(\theta)$ is short for $sin\ (\theta)$, $C(\theta)$ is short for $cos\ (\theta)$

2D Forward Kinematics



Translation transform

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$$



Rotation transform

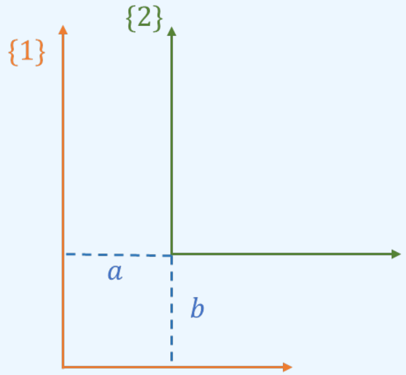
$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) \\ S(\theta) & C(\theta) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

Unified Expression for ${}^1_2T$!

Name: Homogeneous Transformation Matrix

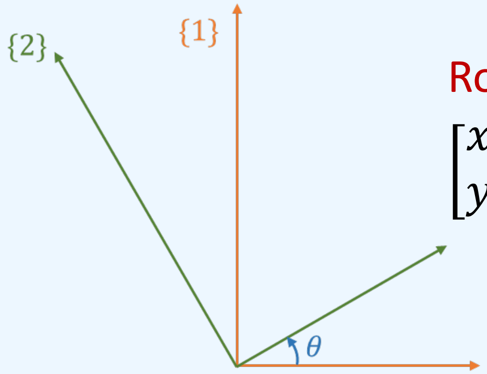
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = {}^1_2T \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) & a \\ S(\theta) & C(\theta) & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

2D Forward Kinematics



Translation transform

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$$



Rotation transform

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) \\ S(\theta) & C(\theta) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

Unified Expression for ${}^1_2T$!

Name: Homogeneous Transformation Matrix

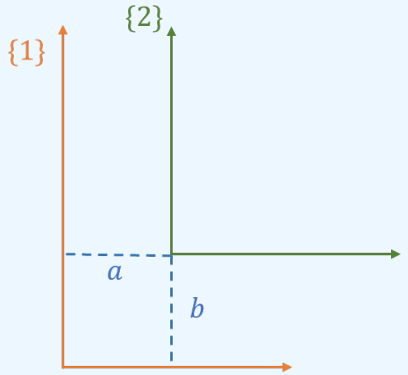
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = {}^1_2T \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) & a \\ S(\theta) & C(\theta) & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Imagine a case with translation only,

$\theta = 0$,

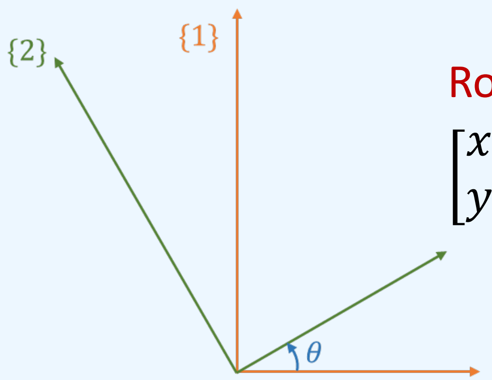
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} x_2 + a \\ y_2 + b \\ 1 \end{bmatrix}$$

2D Forward Kinematics



Translation transform

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$$



Rotation transform

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) \\ S(\theta) & C(\theta) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

Unified Expression for ${}^1_2T$!

Name: Homogeneous Transformation Matrix

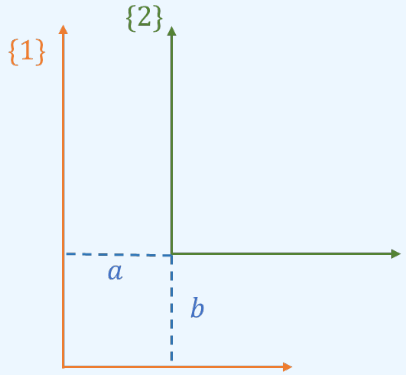
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = {}^1_2T \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) & a \\ S(\theta) & C(\theta) & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Imagine a case with rotation only,

$a = b = 0$,

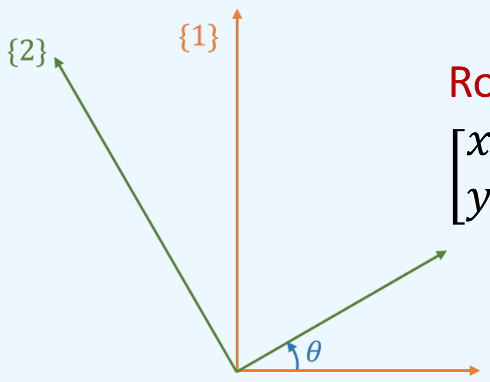
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) & 0 \\ S(\theta) & C(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

2D Forward Kinematics



Translation transform

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$$



Rotation transform

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) \\ S(\theta) & C(\theta) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

Unified Expression for ${}^1_2T$!

Name: Homogeneous Transformation Matrix

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = {}^1_2T \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} C(\theta) & -S(\theta) & a \\ S(\theta) & C(\theta) & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Infer the frame relationship just from the matrix?

Frame {2} orientation rotates θ w.r.t {1},
Also, the origin of {2} is $[a\ b]$ far way from {1}.

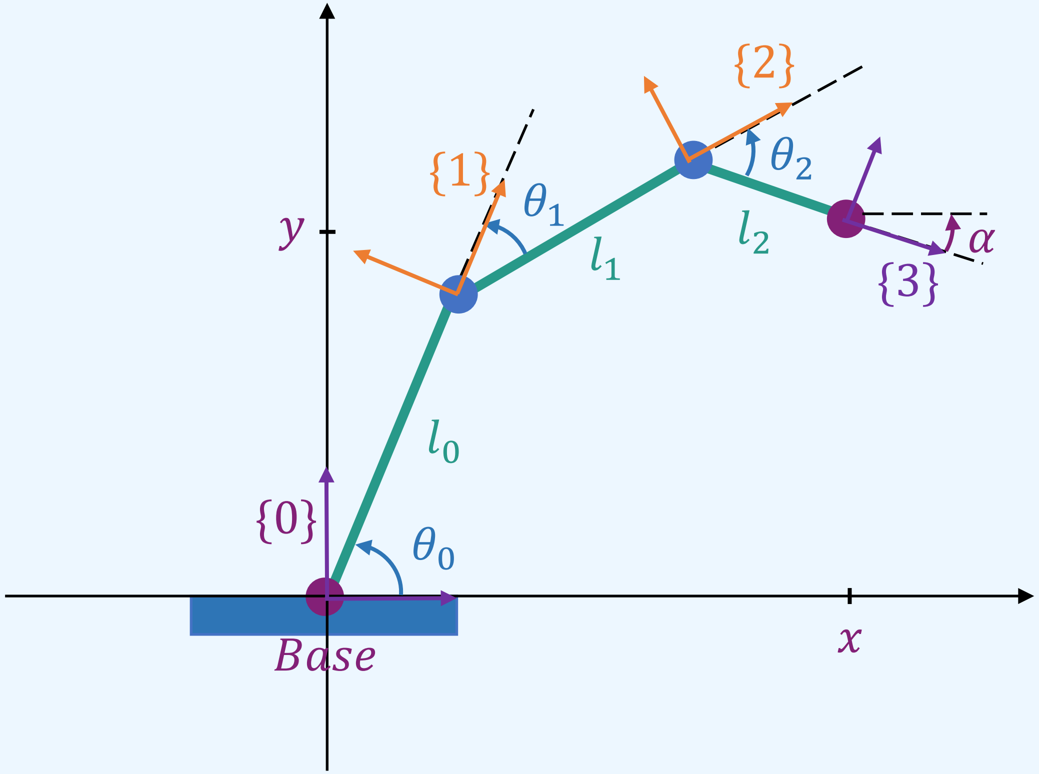
2D Forward Kinematics

Back to our problem: How to obtain $[x\ y\ \alpha]$?

Approach 2: Transformation Matrix

If we can get ${}^0_3T = \begin{bmatrix} C(\theta) & -S(\theta) & a \\ S(\theta) & C(\theta) & b \\ 0 & 0 & 1 \end{bmatrix}$,

then $[x\ y\ \alpha] = [a\ b\ \theta]$ in this transformation matrix



Then, how to get ${}^0_3T$?

Frame by frame transform: ${}^0_3T = {}^0_1T\,{}^1_2T\,{}^2_3T$

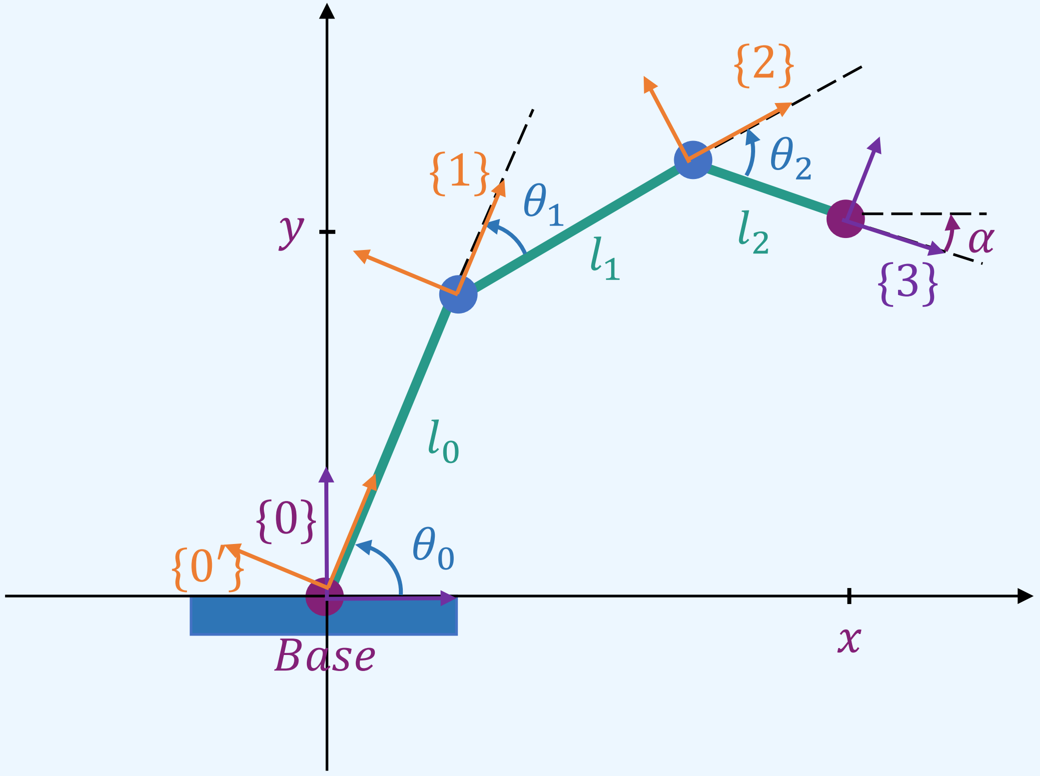
${}^A_BT$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

Example: obtain ${}^0_1T$, $\{1\}$ w.r.t $\{0\}$

$\{0\}$ first rotates θ_0 to $\{0'\}$,
then $\{0'\}$ translates $[l_0, 0]$ to $\{1\}$

$$
{}^0_1T = {}^0_{0'}T\,{}^{0'}_{1}T = \begin{bmatrix} C(\theta_0) & -S(\theta_0) & 0 \\ S(\theta_0) & C(\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & l_0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
= \begin{bmatrix} C(\theta_0) & -S(\theta_0) & l_0C(\theta_0) \\ S(\theta_0) & C(\theta_0) & l_0S(\theta_0) \\ 0 & 0 & 1 \end{bmatrix}
$$



$\{0'\}$ is an auxiliary frame we add

${}^A_BT$: describe frame $\{B\}$ w.r.t frame $\{A\}$

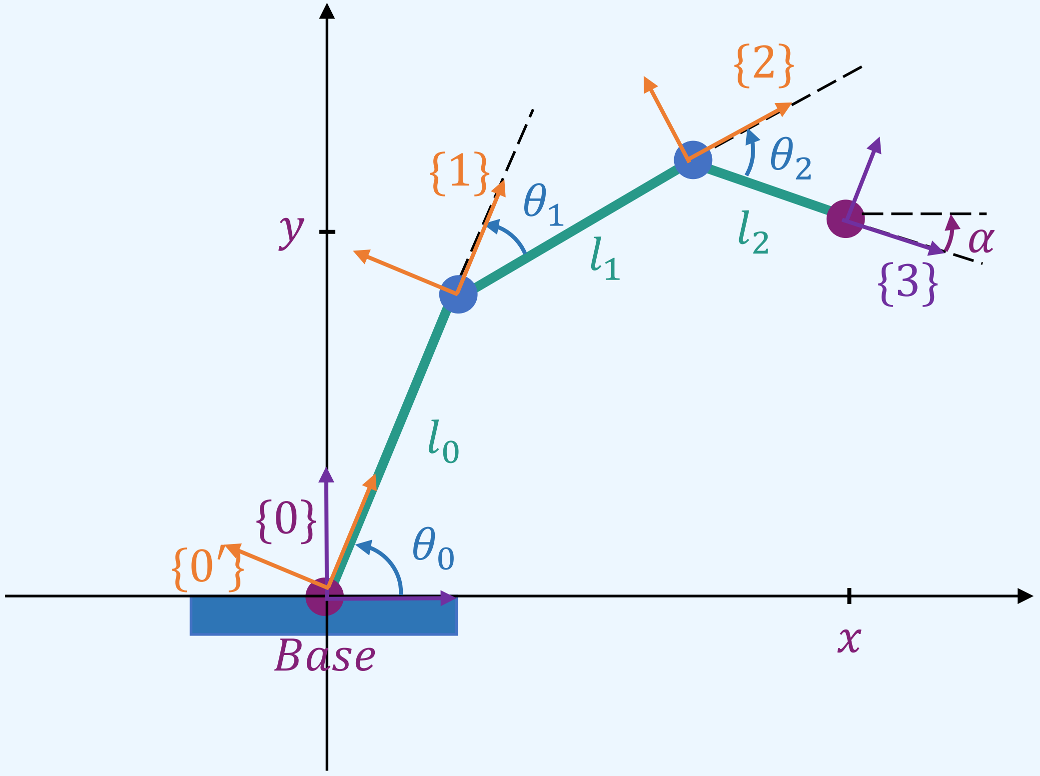
2D Forward Kinematics

Example: obtain ${}^0_1T$, $\{1\}$ w.r.t $\{0\}$

$\{0\}$ first rotates θ_0 to $\{0'\}$,
then $\{0'\}$ translates $[l_0, 0]$ to $\{1\}$

$$
{}^0_1T = {}^0_{0'}T\,{}^{0'}_1T = \begin{bmatrix} C(\theta_0) & -S(\theta_0) & 0 \\ S(\theta_0) & C(\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & l_0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} C(\theta_0) & -S(\theta_0) & l_0C(\theta_0) \\ S(\theta_0) & C(\theta_0) & l_0S(\theta_0) \\ 0 & 0 & 1 \end{bmatrix}
$$



$\{0'\}$ is an auxiliary frame we add

Infer the frame relationship just from the matrix?

Frame {1} orientation rotates θ w.r.t {0},
Also, the origin of {2} is $[l_0C(\theta_0)\ l_0S(\theta_0)]$ far way from {1}.

${}^A_BT$: describe frame $\{B\}$ w.r.t frame $\{A\}$

2D Forward Kinematics

Example: obtain ${}^{0}_{1}T$, $\{1\}$ w.r.t $\{0\}$

$\{0\}$ first rotates θ_0 to $\{0'\}$,
then $\{0'\}$ translates $[l_0, 0]$ to $\{1\}$

$$
{}^{0}_{1}T = {}^{0}_{0'}T\,{}^{0'}_{1}T = \begin{bmatrix} C(\theta_0) & -S(\theta_0) & 0 \\ S(\theta_0) & C(\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & l_0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} C(\theta_0) & -S(\theta_0) & l_0 C(\theta_0) \\ S(\theta_0) & C(\theta_0) & l_0 S(\theta_0) \\ 0 & 0 & 1 \end{bmatrix}
$$

Infer the frame relationship just from the matrix?
Frame {1} orientation rotates θ w.r.t {0},
Also, the origin of {2} is $[l_0 C(\theta_0)\ l_0 S(\theta_0)]$ far way from {1}.

Now, can we build a function for kinematics?

```
link_length_0 = float( )
link_length_1 = float( )
...
def my_kinematics(joint_angle_0, joint_angle_1, ...):
    ............
    ............
    end_effector_pos=...
    end_effector_ori=...
```

${}^{A}_{B}T$: describe frame $\{B\}$ w.r.t frame $\{A\}$

Wrap up

- Kinematics Definition
- 2D Kinematics
  - Geometry based approach (naïve way)
  - Transformation matrix based approach (standard way)