Forward Kinematics

Ghanshyam Chandra

Ph.D. Candidate (Computational Mathematics Group)
Department of Computational and Data Sciences
Indian Institute of Science

ghanshyamc@iisc.ac.in

February 24, 2021

Agenda

- Quick Revision

Agenda

- Quick Revision
- Implementation

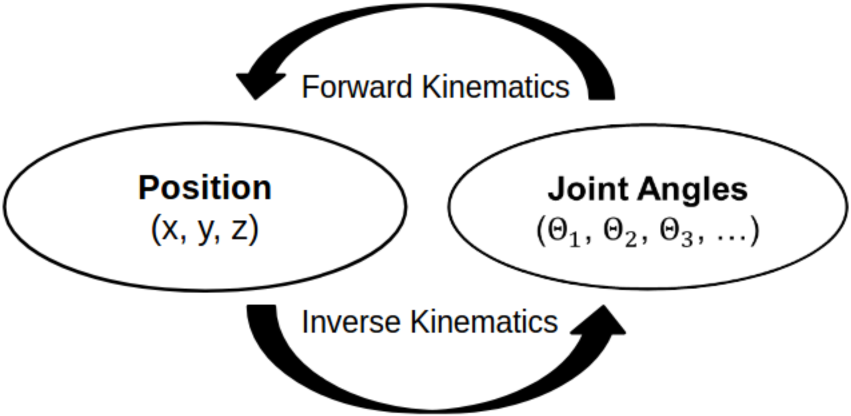
Agenda

- Quick Revision
- Implementation
- Live Simulation

Agenda

- Quick Revision
- Implementation
- Live Simulation
- Deep Reinforcement Learning

What is?


Forward Kinematics
Position
(x, y, z)
Joint Angles
$(\Theta_1, \Theta_2, \Theta_3, \ldots)$
Inverse Kinematics

Implementation

Implementation

- DH Parameters.

Implementation

- DH Parameters.
- Typical Form (4 DoF).

Link	d_i	θ_i	a_i	α_i
1	d_1	θ_1	a_1	α_1
2	d_2	θ_2	a_2	α_2
3	d_3	θ_3	a_3	α_3
4	d_4	θ_4	a_3	α_4

Implementation

- Transformation Matrix

Implementation

- Transformation Matrix

$$[Z_i] = \mathrm{Trans}_{Z_i}(d_i)\,\mathrm{Rot}_{Z_i}(\theta_i)$$

Implementation

- Transformation Matrix

$$[Z_i] = \mathrm{Trans}_{Z_i}(d_i)\,\mathrm{Rot}_{Z_i}(\theta_i)$$

$$[X_i] = \mathrm{Trans}_{X_i}(a_{i,i+1})\,\mathrm{Rot}_{X_i}(\alpha_{i,i+1})$$

Implementation

- Transformation Matrix

$$[Z_i] = \text{Trans}_{Z_i}(d_i)\,\text{Rot}_{Z_i}(\theta_i)$$

$$[X_i] = \text{Trans}_{X_i}(a_{i,i+1})\,\text{Rot}_{X_i}(\alpha_{i,i+1})$$

$$^{n-1}T_n = \text{Trans}_{z_{n-1}}(d_n) \cdot \text{Rot}_{z_{n-1}}(\theta_n) \cdot \text{Trans}_{x_n}(a_n) \cdot \text{Rot}_{x_n}(\alpha_n)$$

Implementation

$$
{}^{n-1}T_n = \left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & a_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & a_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

$$
= \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

Implemenatation

- Get Transformation Matrix.

Implemenatation

- Get Transformation Matrix.

```
function [T] = getTransformMatrix(theta, d, a, alpha)
T = [cos(theta) -sin(theta) * cos(alpha) sin(theta) * sin(alpha) a * cos(theta);
     sin(theta) cos(theta) * cos(alpha) -cos(theta) * sin(alpha) a * sin(theta);
     0,sin(alpha),cos(alpha),d;
     0,0,0,1];
end
```

Implemenatation

- Forward Kinematics.

Implemenatation

- Forward Kinematics.

```
function [T00,T01,T12,T23,T34,T45,T56,Etip]=
    forwardKinematics(theta1,d1,a1,alpha1,theta2,
    d2,a2,alpha2,theta3,d3,a3,alpha3,theta4,d4,a4
    ,alpha4,theta5,d5,a5,alpha5,theta6,d6,a6,
    alpha6)

T00 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
T01 = getTransformMatrix(theta1,d1,a1,alpha1);
T12 = getTransformMatrix(theta2,d2,a2,alpha2);
T23 = getTransformMatrix(theta3,d3,a3,alpha3);
T34 = getTransformMatrix(theta4,d4,a4,alpha4);
T45 = getTransformMatrix(theta5,d5,a5,alpha5);
T56 = getTransformMatrix(theta6,d6,a6,alpha6);

Etip = T00 * T01 * T12 * T23 * T34 * T45 * T56;
```

Implemenatation

- Final Transformation Matrix.

Implemenatation

- Final Transformation Matrix.

$$
Etip = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
$$

Implemenatation

- Plotting the Robot Link.

- Plotting the Robot Link.

```
function [FK_plot3D] = FK_plot3D(Th_1, Th_2, Th_3, Th_4, Th_5, Th_6, a_1, a_2, a_3, a_4, a_5, a_6, d_1, d_2, d_3, d_4, d_5, d_6, al_1, al_2, al_3, al_4, al_5, al_6)
%Plotting The workspace
L (1) = Link( [Th_1 d_1 a_1 al_1] );
L (2) = Link( [Th_2 d_2 a_2 al_2] );
L (3) = Link( [Th_3 d_3 a_3 al_3] );
L (4) = Link( [Th_4 d_4 a_4 al_4] );
L (5) = Link( [Th_5 d_5 a_5 al_5] );
L (6) = Link( [Th_6 d_6 a_6 al_6] );
Robot = SerialLink(L);
Robot.name = '6 - DoF forward Kinematics';
Robot.plot([Th_1 Th_2 Th_3 Th_4 Th_5 Th_6]);
```

Implemenatation

- Computing Each Transformation Matrix.

Implemenatation

- Computing Each Transformation Matrix.

```
[T00,T01,T12,T23,T34,T45,T56,Etip]=forwardKinematics(Th_1,d_1,a_1,al_1 ,Th_2,d_2,a_2,al_2 ,Th_3,d_3,a_3,al_3 ,Th_4,d_4,a_4,al_4 ,Th_5,d_5,a_5,al_5 ,Th_6,d_6,a_6,al_6);

T2 = T01*T12; T3 =T2*T23 ; T4 = T3*T34 ; T5 = T4*T45; T6 = T5*T56;
```

3D WorkSpace Plot

- **For 6 DoF, we need at most n iteration for 24 variables.**

3D WorkSpace Plot

- **For 6 DoF, we need at most n iteration for 24 variables.**
- **But how to implement it for generalised case?**

3D WorkSpace Plot

- **For 6 DoF, we need at most n iteration for 24 variables.**
- **But how to implement it for generalised case?**
- **I tried to implement for 12 variables and it took around 48,000 live lines of code.**

3D WorkSpace Plot

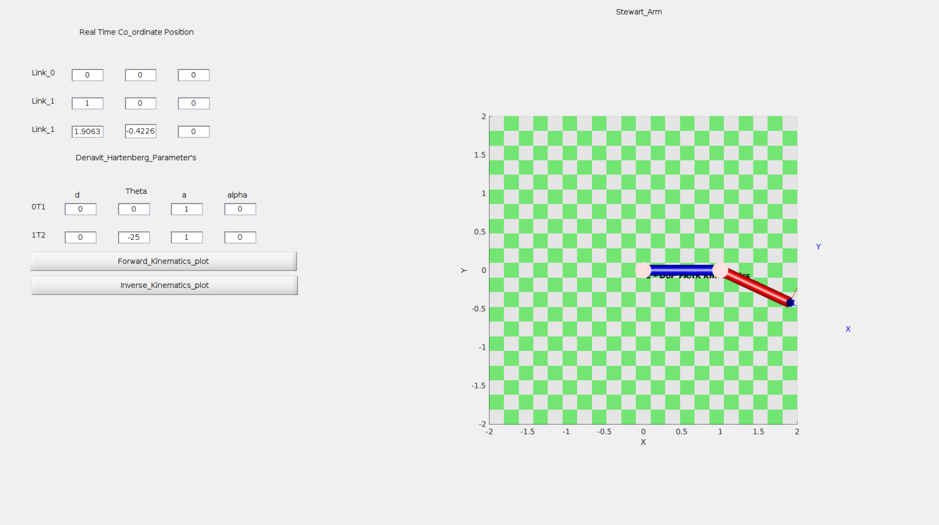
- **For 6 DoF, we need at most n iteration for 24 variables.**
- **But how to implement it for generalised case?**
- **I tried to implement for 12 variables and it took around 48,000 live lines of code.**
- **Have you dared to write a generalised Workspace code for 24 variable?**

DH-Table

- Planner Robot (2 DoF).

Link	d_i	θ_i	a_i	α_i
1	0	10	0	0
2	0	20	0	0

2 DoF (Open_Robo_Simulator)

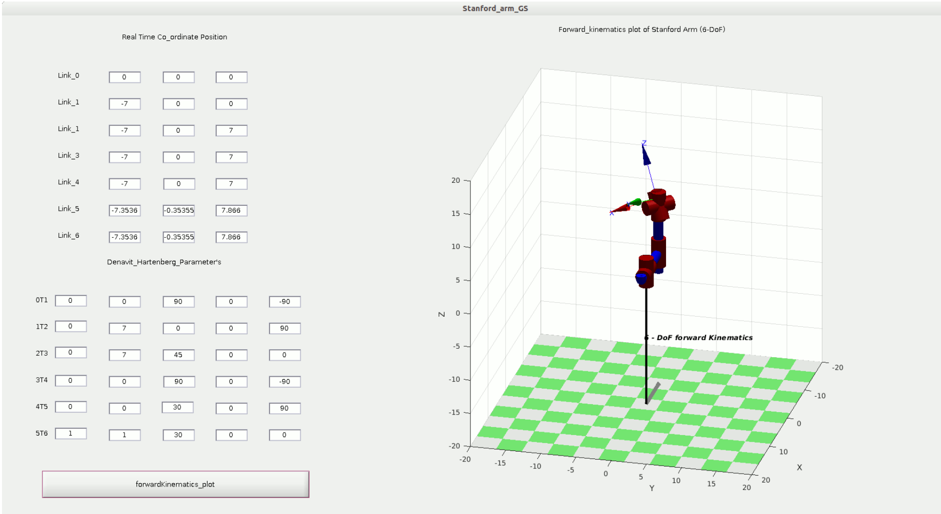


DH-Table

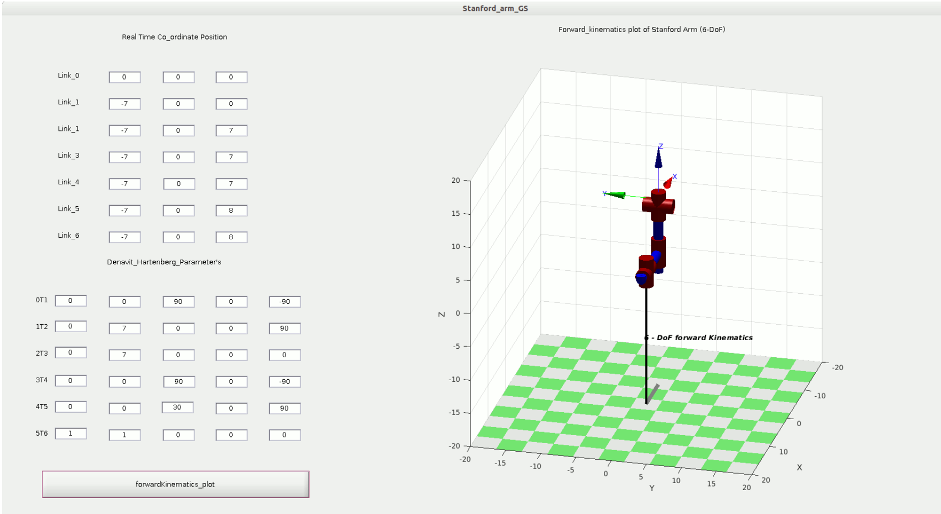
- Stanford Arm (6 DoF).

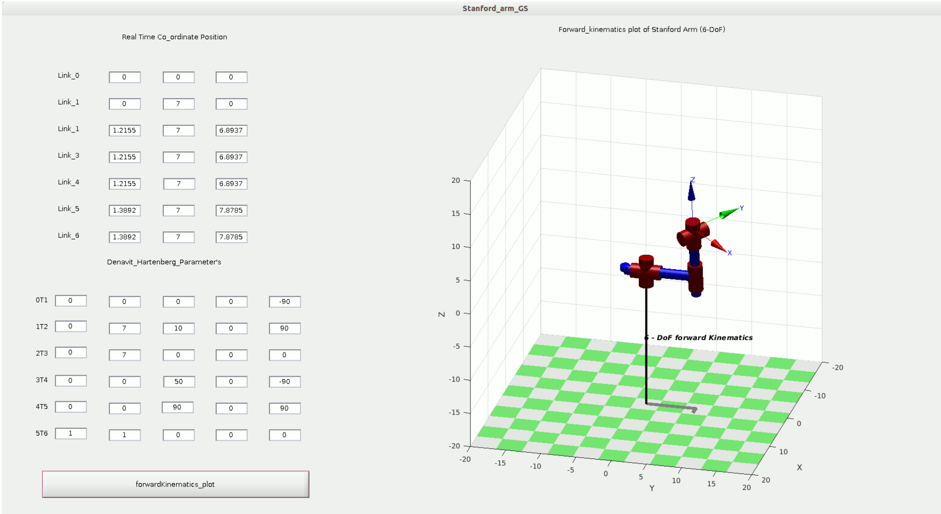
Link	d_i	θ_i	a_i	α_i
1	0	-90	0	-90
2	7	90	0	90
3	7	0	0	0
4	0	90	0	-90
5	0	45	0	90
6	1	45	0	0

6 DoF (Open_Robo_Simulator)

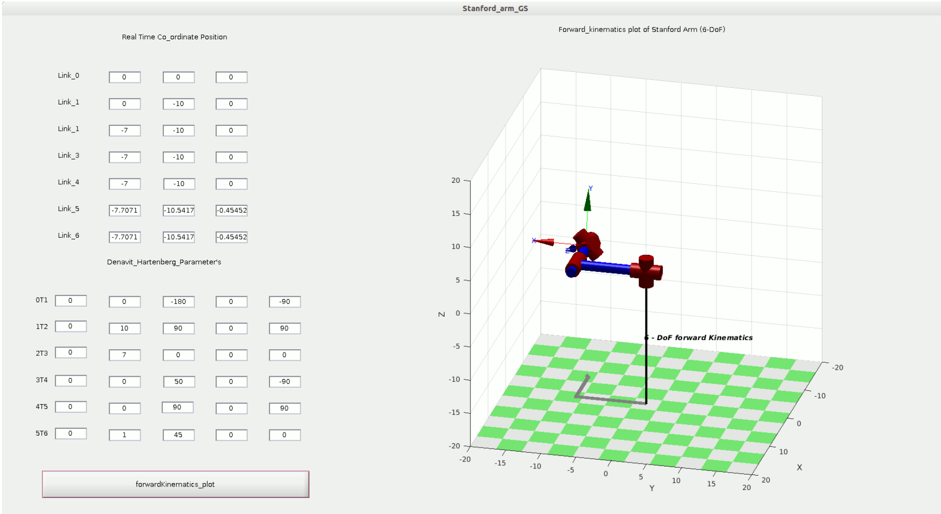


6 DoF (Open_Robo_Simulator)

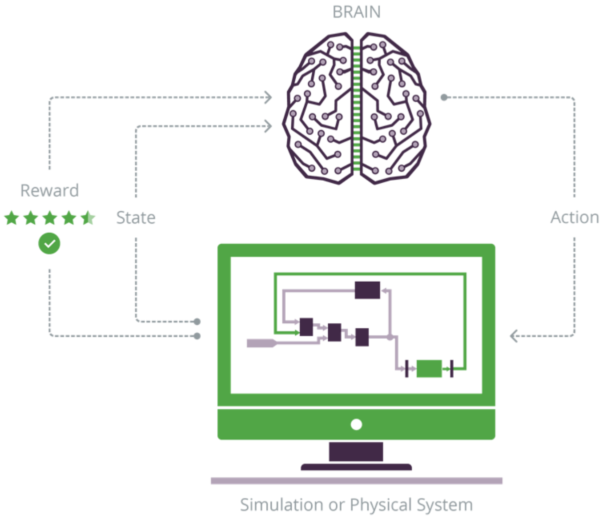



Stanford_arm_GS
Real Time Co_ordinate Position
Link_0 0 0 0
Link_1 0 7 0
Link_1 1.2155 7 6.8937
Link_3 1.2155 7 6.8937
Link_4 1.2155 7 6.8937
Link_5 1.3892 7 7.8785
Link_6 1.3892 7 7.8785
Denavit_Hartenberg_Parameter's
0T1 0 0 0 0 -90
1T2 0 7 10 0 90
2T3 0 7 0 0 0
3T4 0 0 50 0 -90
4T5 0 0 90 0 90
5T6 1 1 0 0 0
forwardKinematics_plot
Forward_kinematics plot of Stanford Arm (6-DoF)
6 - DoF forward Kinematics
Z
X
Y

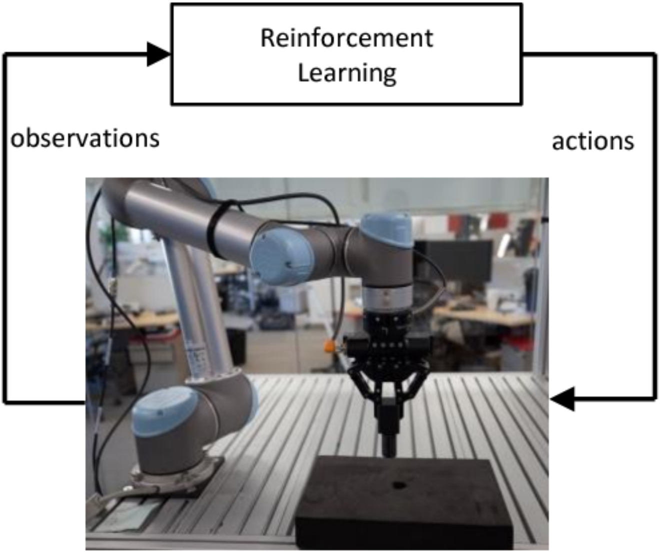
6 DoF (Open_Robo_Simulator)



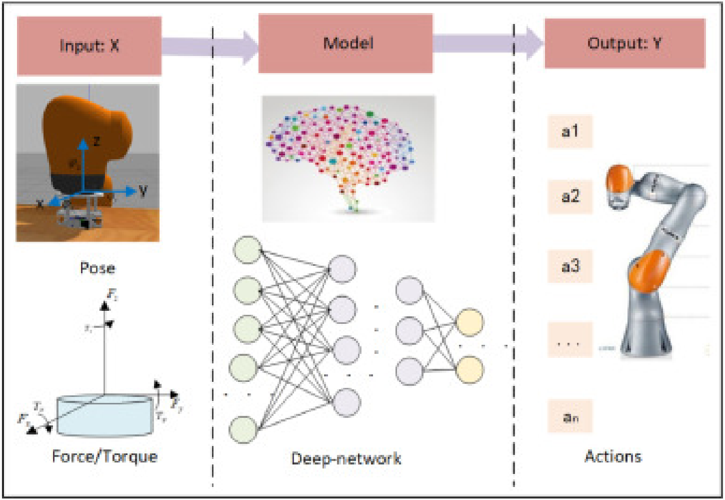
Deep Reinforcement Learning



Deep Reinforcement Learning



Deep Learning



Referances

1)ROBOTICS: FUNDAMENTAL CONCEPTS AND ANALYSIS
2)Introduction to Robotics: Analysis, Control, Applications, 3rd Edition(Saeed B. Niku)
3)Denavit–Hartenberg parameters(Wikipedia)