Lab Number 06
CoppeliaSim EDU – Computing Forward Kinematics of UR5

Introduction:

Forward Kinematics is computed when desired end effector position and orientation is required after a robot is moved by angles.

Software Used:

CoppeliaSim EDU v4.7

Programming Language:

LUA

Objectives of the Lab:

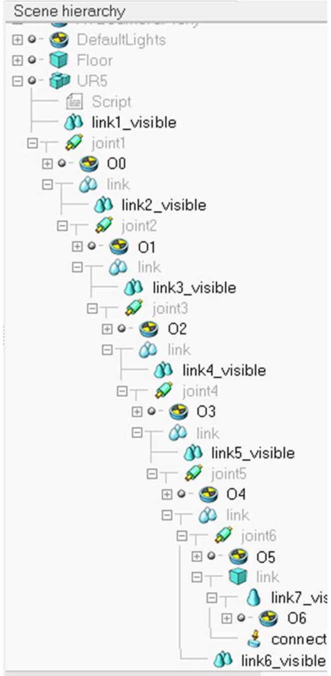
- To Extract DH Parameters of UR5
- To compute Forward Kinematics of UR5

Procedure:

First of all, add the UR5 robot in the scene and set its position and orientation to

Add 7 reference frames and name them O0, O1, O2, ….

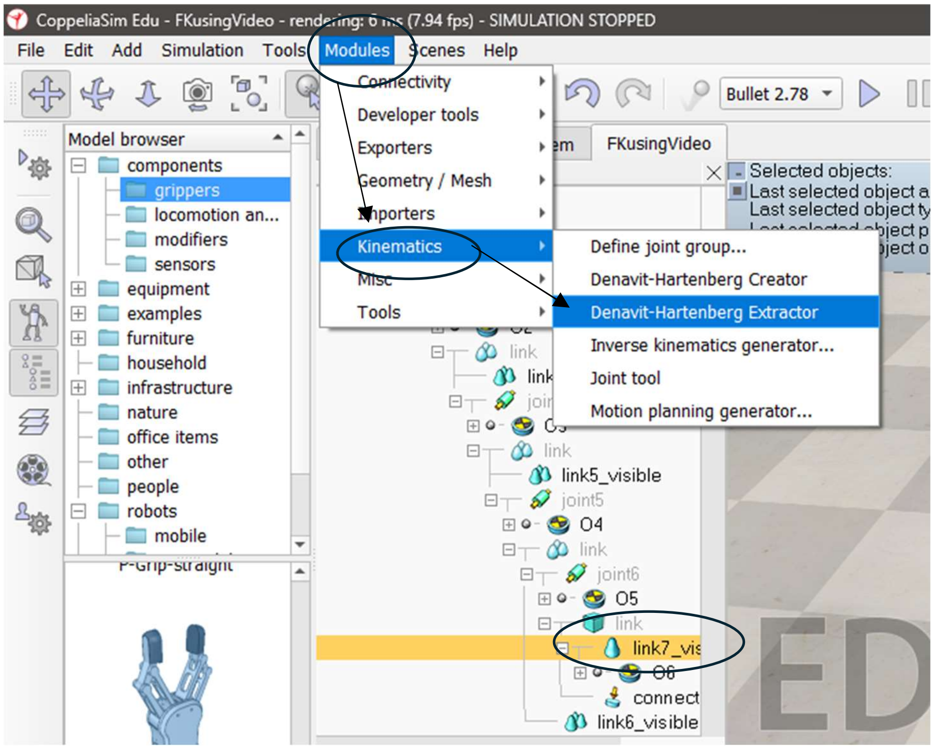
Set position and orientation of all reference frames 0,0,0

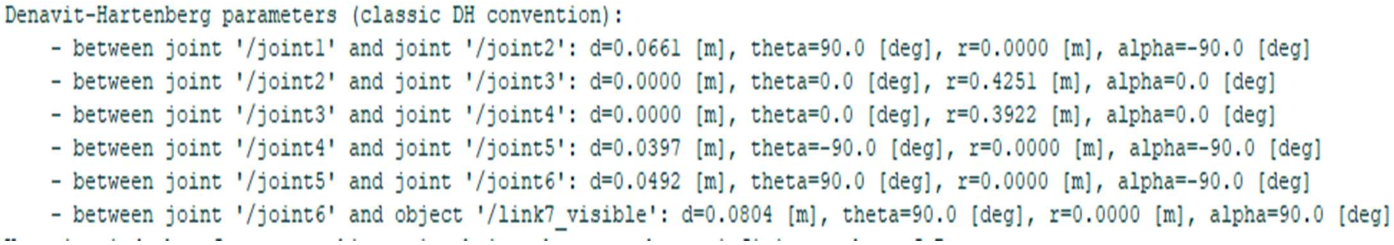


```
Scene hierarchy
DefaultLights
Floor
UR5
  Script
  link1_visible
  joint1
    O0
    link
      link2_visible
      joint2
        O1
        link
          link3_visible
          joint3
            O2
            link
              link4_visible
              joint4
                O3
                link
                  link5_visible
                  joint5
                    O4
                    link
                      joint6
                        O5
                        link
                          link7_vis
                            O6
                            connect
                      link6_visible
```

EXTRACT DH-PARAMETERS:

(DH – Parameters are conventional ones in CoppeliaSim)





```
Denavit-Hartenberg parameters (classic DH convention):
    - between joint '/joint1' and joint '/joint2': d=0.0661 [m], theta=90.0 [deg], r=0.0000 [m], alpha=-90.0 [deg]
    - between joint '/joint2' and joint '/joint3': d=0.0000 [m], theta=0.0 [deg], r=0.4251 [m], alpha=0.0 [deg]
    - between joint '/joint3' and joint '/joint4': d=0.0000 [m], theta=0.0 [deg], r=0.3922 [m], alpha=0.0 [deg]
    - between joint '/joint4' and joint '/joint5': d=0.0397 [m], theta=-90.0 [deg], r=0.0000 [m], alpha=-90.0 [deg]
    - between joint '/joint5' and joint '/joint6': d=0.0492 [m], theta=90.0 [deg], r=0.0000 [m], alpha=-90.0 [deg]
    - between joint '/joint6' and object '/link7_visible': d=0.0804 [m], theta=90.0 [deg], r=0.0000 [m], alpha=90.0 [deg]
```

CODE & EXPLANATION:

Add a script and do the following code

```
sim = require'sim'                    - -module
```

➔ initialize the Robot and Frames

```
function sysCall_init()
n = 6                                 - - number of joints
joints = {}                           - - Initialize joints of UR5
for i = 1,n,1 do
joints [i] = sim.getObjectHandle('../joint'..i)
```

```
end
O = {}                              - - initialize Reference Frames associated with UR5
for i =1,n+1,1 do
O[i] = sim.getObjectHandle('../O'..(i-1))
```

➔ Add target angles and DH parameters extracted

```
t = {math.pi/6,math.pi/3,0,0,0,0}  - - angles to move each joint
DH_params = {
   {t[1]+math.pi/2, 0.0661, 0, -math.pi / 2},       -- Joint 1
   {t[2]-math.pi/2, 0, 0.425, 0},                   -- Joint 2
   {t[3], 0, 0.39225, 0},                 -- Joint 3
   {t[4]-math.pi/2, 0.0397, 0, -math.pi / 2},       -- Joint 4
   {t[5]+math.pi/2, 0.0492, 0, -math.pi / 2},       -- Joint 5
   {t[6]+math.pi/2, 0.0804, 0, math.pi/2}                   -- Joint 6
}
end    - - End of sysCall_init function
```

➔ Add a function DH( ) that return a matrix with solving theta, d, a, alpha.

```
function DH(joint,params)
local th,d,a,alpha
local cth,sth,calpha,salpha
local A,q
q = sim.getJointPosition(joint)
if(sim.getJointType(joint)==sim.joint_revolute_subtype) then
th = q+params[1]
d = params[2]
else
th = params[1]
d=q+params[2]
end
a = params[3]
```

```
alpha = params[4]
cth = math.cos(th)
sth = math.sin(th)
calpha = math.cos(alpha)
salpha = math.sin(alpha)
```

➔ This is the transformation matrix for conventional DH params

```
A= Matrix (4,4,{cth,-sth*calpha,sth*salpha,a*cth,
            sth,cth*calpha,-cth*salpha,a*sth,
            0, salpha,calpha,d,
    0,0,0,1})
        return A
end
```

➔ Get the each transformation matrix for that purpose make a function

```
function getAi(joints,DH_params)
      local Ai={}
      for i=1,n,1 do
            Ai[i] = DH(joints[i],DH_params[i])
      end
      return Ai
end
```

➔ Update the frames relative to the previous frame

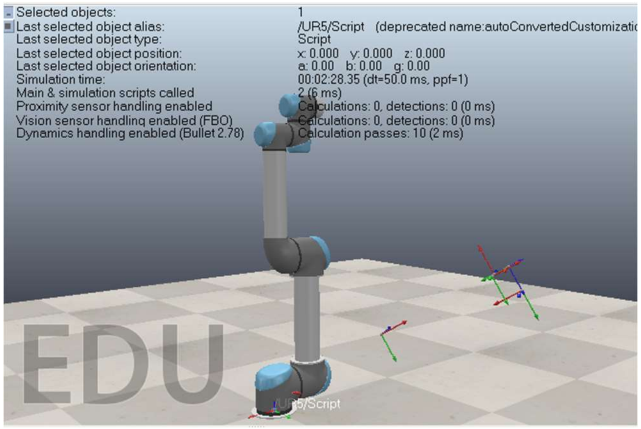
```
function updateFrames(joints,DH_params)
Ai = getAi(joints,DH_params)
for i = 1,n,1 do
sim.setObjectMatrix(O[i+1],O[i],Ai[i]:data())
~~--sim.setJointTargetPosition(joints[i],t[i])~~
end
end
```

➔ Call updateFrames function in actuation

```
function sysCall_actuation()
      updateFrames(joints,DH_params)
    sim.step()


end
```

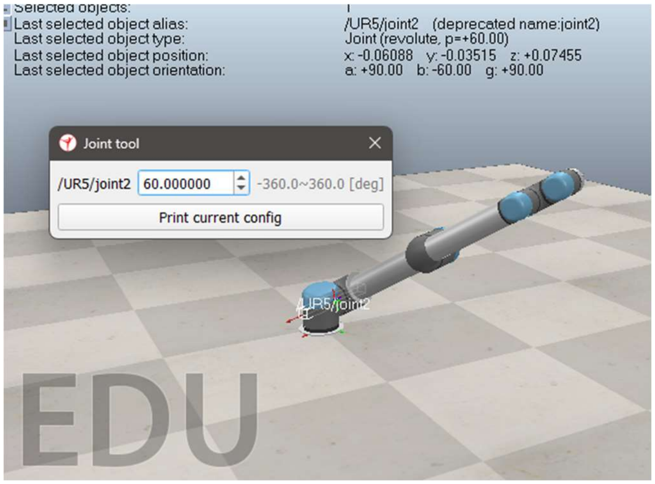
Now you can see that frames are updated on the position by angles t.



```
Selected objects:                        1
Last selected object alias:              /UR5/Script   (deprecated name:autoConvertedCustomizatic
Last selected object type:               Script
Last selected object position:           x: 0.000   y: 0.000   z: 0.000
Last selected object orientation:        a: 0.00   b: 0.00   g: 0.00
Simulation time:                         00:02:28.35 (dt=50.0 ms, ppf=1)
Main & simulation scripts called         2 (6 ms)
Proximity sensor handling enabled        Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)     Calculations: 0, detections: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78)  Calculation passes: 10 (2 ms)
```

This is done by moving joint angles of first two joints only 30 and 60 degrees.

Now go to Modules → kinematics → joint tool and verify whether robot is moved to the same direction or not by giving it angles 30 and 60.



```
Selected objects:                        1
Last selected object alias:              /UR5/joint2   (deprecated name:joint2)
Last selected object type:               Joint (revolute, p=+60.00)
Last selected object position:           x: -0.06088   y: -0.03515   z: +0.07455
Last selected object orientation:        a: +90.00   b: -60.00   g: +90.00
```

Joint tool

/UR5/joint2 60.000000 -360.0~360.0 [deg]