Lab Number 07
CoppeliaSim EDU – Computing Inverse Kinematics of UR5

Introduction:

Inverse Kinematics is computed when robot needs to move at a known end effector position and orientation calculating the required joint angles for smooth operation.

Software Used:

CoppeliaSim EDU v4.7

Programming Language:

LUA (SimIk Plugin Used)

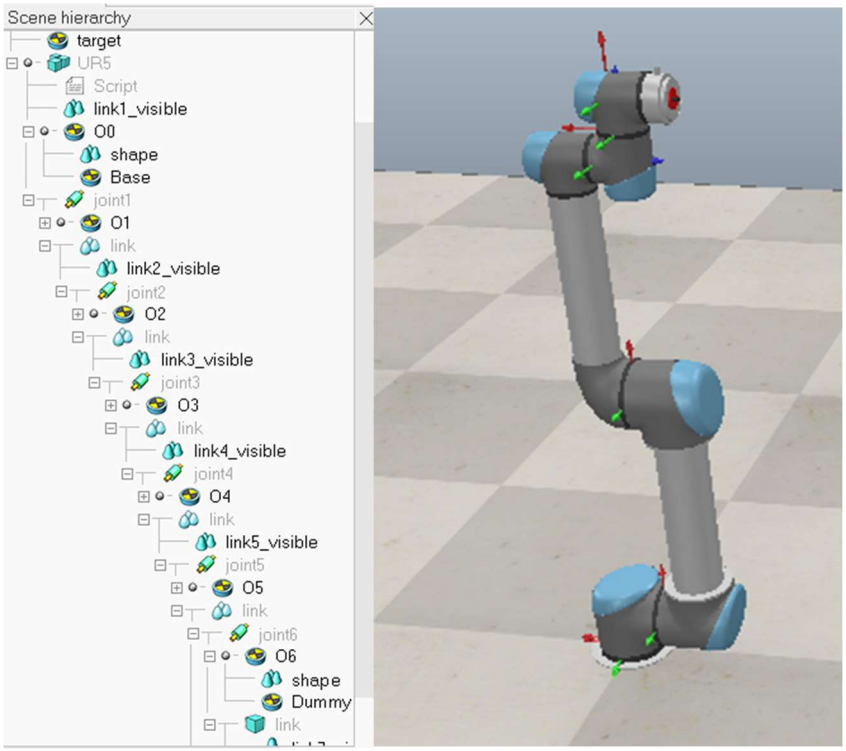
Objectives of the Lab:

- To compute Inverse Kinematics of UR5

INVERSE KINEMATICS:

Add 7 reference frames and name them as O0, O1, …. , O6.

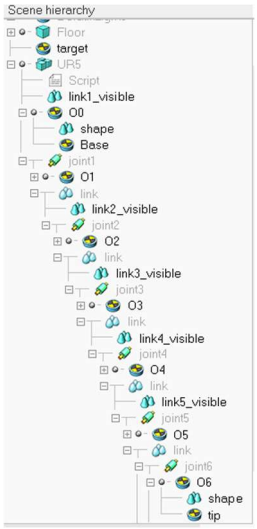
Put O0 as a child of UR5 and O1 for joint 1, O2 for joint 2 and so on…



Now translate and rotate all the frames relative to parent frame and give them values 0,0,0 for both position and orientation. Also make sure that UR5 is at 0,0,0 position for smooth measurement.

BASE, TIP and TARGET:

To compute inverse kinematics three dummy object to be placed. Tip to be placed under joint 6 along with O6. Target dummy object to be placed individually without being a child object of anyone. Base to placed as a child object of UR5. Also make TIP and BASE orientation and position 0,0,0 with respect to the parent frame. Make target near robot for the robot to move in that direction but it should not be outside workspace of the robot.



CODE & EXPLANATION:

➔ ADD modules

```
sim=require'sim'
simIK=require'simIK'   - - specifically for inverse kinematics (simIK plugin)
```

➔ Initialize the robot, frames in sysCall_init( ) function

```
function sysCall_init()
    -- Get the joint handles
   jointHandles = {}
   for i = 1, 6 do
      jointHandles[i] = sim.getObjectHandle('../joint'..i)  -- Replace '/robot_joint_' with your joint
names
   end
```

```
-- Get the frame Handles
O = {}
for i = 1, 6 do
  O[i] = sim.getObjectHandle('../O'..i)  -- Replace '/robot_joint_' with your joint names
end
```

➔ Get object handles of base , tip and target to compute inverse kinematics

```
simBase = sim.getObjectHandle('..')
simTip = sim.getObjectHandle('../tip')
simGoal = sim.getObjectHandle('/target')
```

➔ Create the environment for IK

```
ikEnv = simIK.createEnvironment()
ikGroup_undamped = simIK.createIkGroup(ikEnv)
simIK.setIkGroupCalculation(ikEnv, ikGroup_undamped, simIK.method_pseudo_inverse, 0, 6)
simIK.addIkElementFromScene(ikEnv, ikGroup_undamped, simBase, simTip, simGoal,
simIK.constraint_pose)
end
```

➔ In sysCall_actuation( ) function

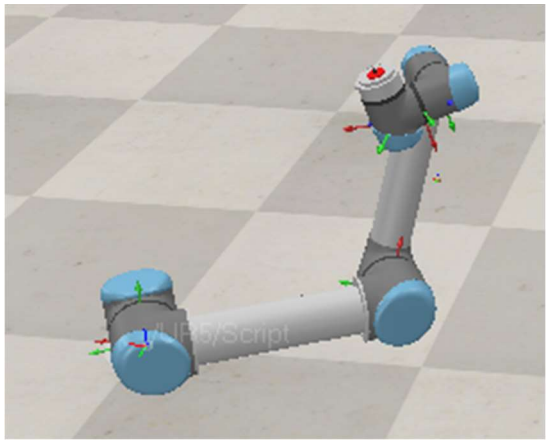
```
function sysCall_actuation( )
simIK.applyIkEnvironmentToScene(ikEnv, ikGroup_undamped, true)
```

➔ Optional (calculation of joint angles)

```
--calculate the angles to validate
  jointAngles = {}
  for i = 1, #jointHandles do
    local rad = sim.getJointPosition(jointHandles[i])
    local deg = rad * (180 / math.pi)  -- Convert radians to degrees
    jointAngles[i] = deg
  end

  -- Print joint angles in degrees
```

```
    for i, angle in ipairs(jointAngles) do
        print('Joint ' .. i .. ' angle: ' .. angle .. ' degrees')
    end
end
```



Now check the joint angles it has moved and verify using joint tool to make sure that the robot had moved correctly.