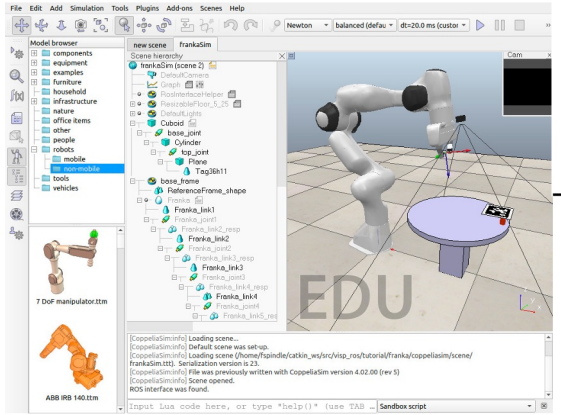
Contents

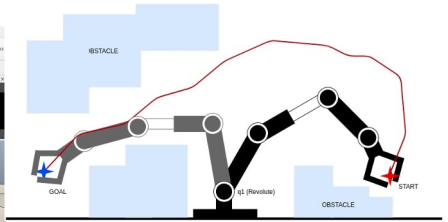
Task 1 - Self-Collision Avoidance

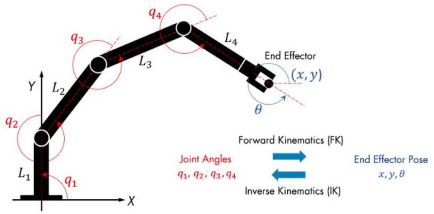
Methods and Tools

1. CoppeliaSim with Lua

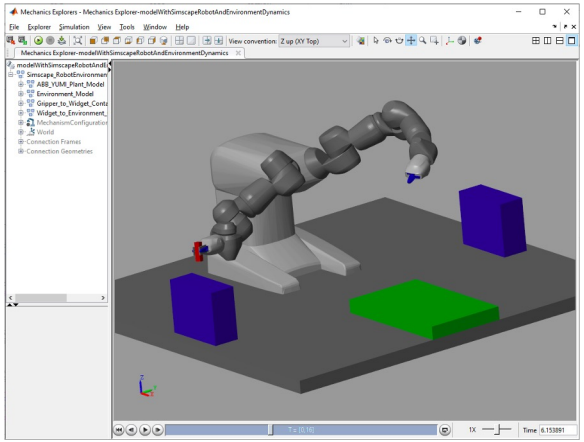
- OMPL (Open Motion Planning Library)
- IK (Inverse kinematics)







2. Matlab



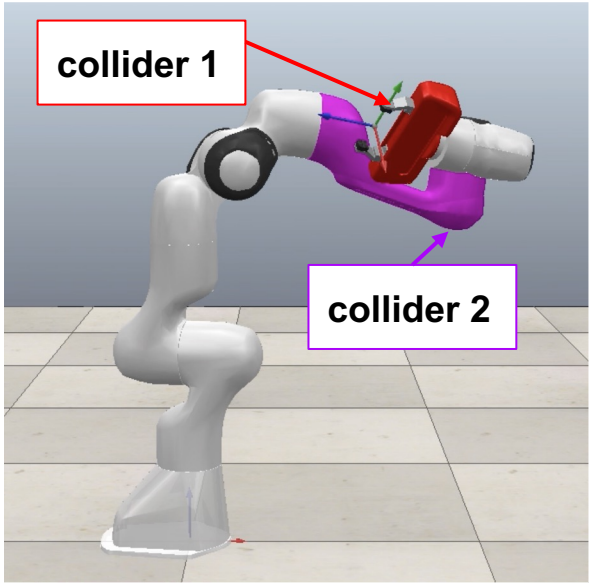
Task 1.1 Check self-collision from specific joint configuration

Implementation process

- Input 7 joint poses (**self_collision_specific_joint_poses.data**)

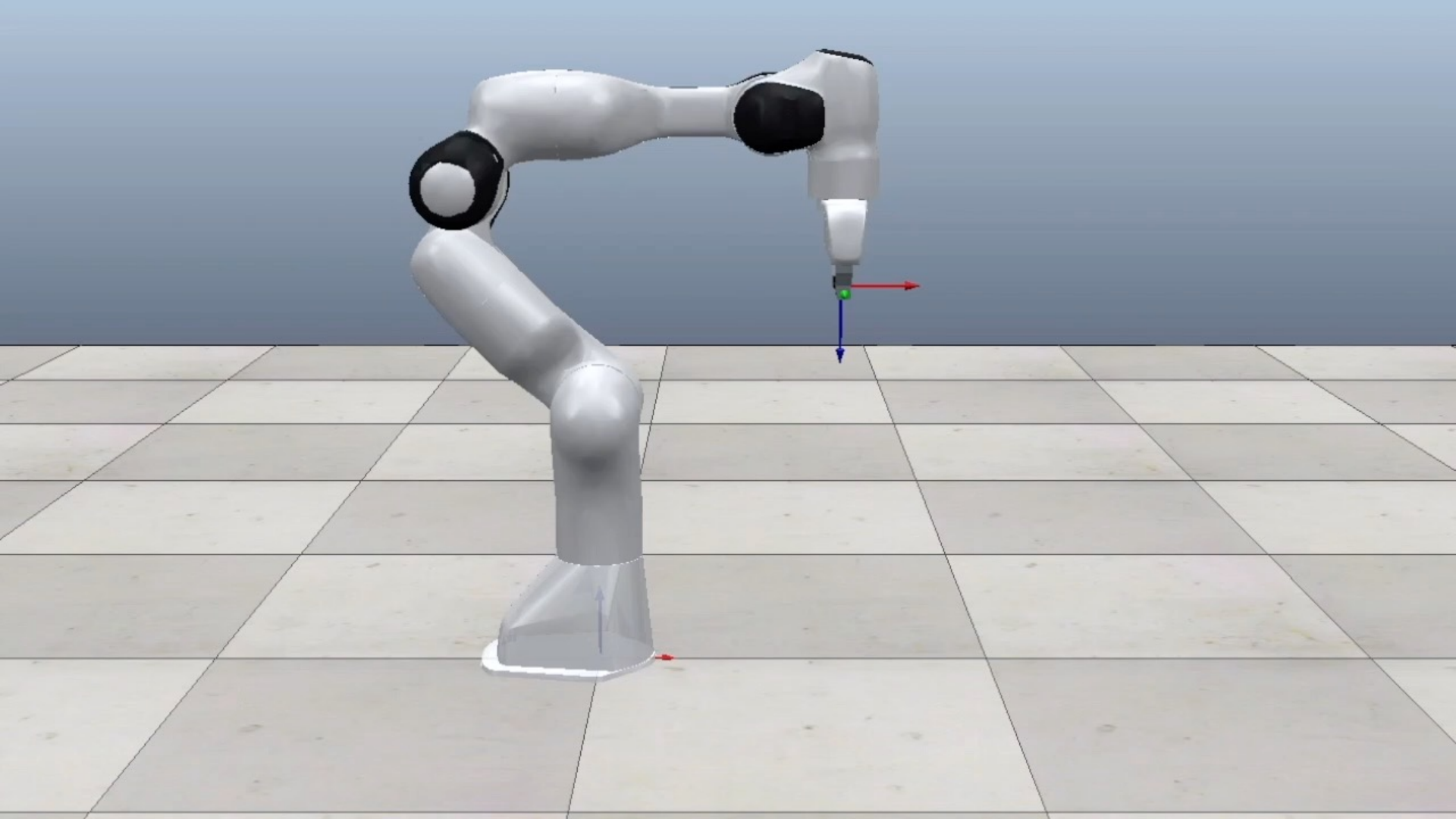
  {"q_example": [-0.015896176064967, 0.0064190366448024, -0.0075707499149116, -0.035205070178435, 0.010341694431194, 0.0093508241818754, 0.043023672042527]}

- Definition Collison Pairs (**collider 1 and collider 2**)
- Chek for Collison between Collidon Pairs (**sim.chekCollison**)
- Highlight collisions by coloring the affected Collion Pairs
- Save results in `results_self_collision.csv` file




```
function sysCall_sensing()
    local result,pairHandles=sim.checkCollision(robotCollection,robotCollection)
    restorePairColor()
    r=sim.setInt32Signal("co_self",result)
    if result>0 then
        -- Change color of the collection and the collidee:
        -- changePairColor({robotCollection,pairHandles[2]},collisionColors)
        -- Change color of the collider and collidee objects:
        changePairColor({pairHandles[1],pairHandles[2]},collisionColors)
        objectNamePair={sim.getObjectName(pairHandles[1]),sim.getObjectName(pairHandles[2])}
        r=sim.setStringSignal("collection1",objectNamePair[1])-- collection 1
        r=sim.setStringSignal("collection2",objectNamePair[2])-- collection 2
       -- print results
        print('"collision": true ')
        print('"collision groups" : '..getAsString(objectNamePair))
    else
        print('"collision": false ')
    end
end
```

```
%% Save Result
[r, state_self] = sim.simxGetIntegerSignal(clientID,'co_self',sim.simx_opmode_blocking);
if state_self
    [r, collisionP1] = sim.simxGetStringSignal(clientID,'collection1',sim.simx_opmode_blocking);
    [r, collisionP2] = sim.simxGetStringSignal(clientID,'collection2',sim.simx_opmode_blocking);
    X1 = "collision: True";
    a = string(collisionP1);
    b = string(collisionP2);
    X2 = "collision Group:" + string(i) +' '+ a +' '+ b;
    results(i) = X2;
else
    results(i) = "collision: False";
end
```



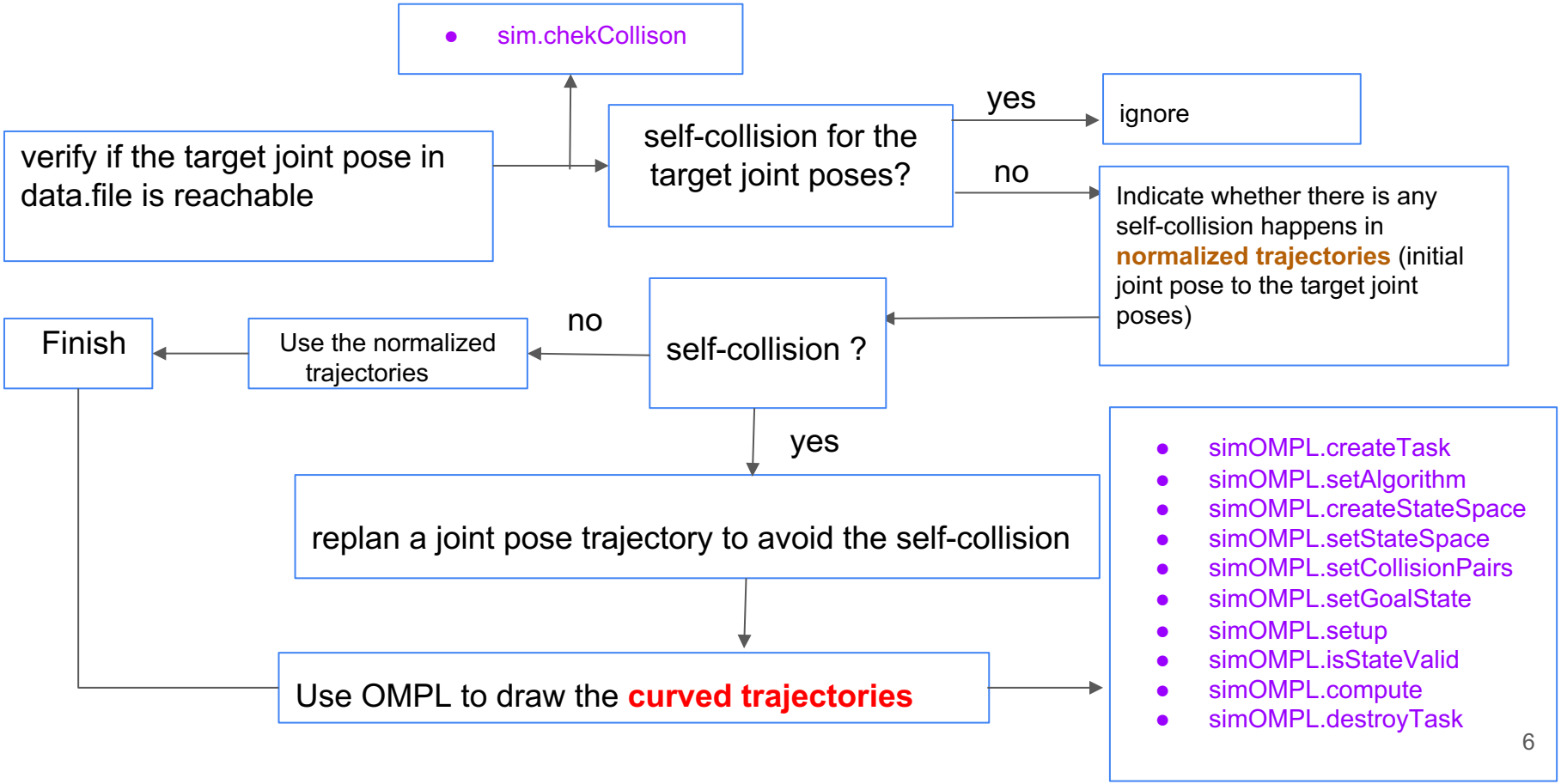
Result

1	collision: False
2	collision: False
3	collision: False
4	collision Group:4 FrankaGripper_visible Franka_link6
5	collision: False
6	collision: False
7	collision: False
8	collision: False
9	collision Group:9 FrankaGripper_visible Franka_link6
10	collision: False
11	collision: False
12	collision: False
13	collision: False
14	collision: False
15	collision: False
16	collision: False
17	collision: False
18	collision: False
19	collision Group:19 Franka_link3 FrankaGripper_leftFinger_visible

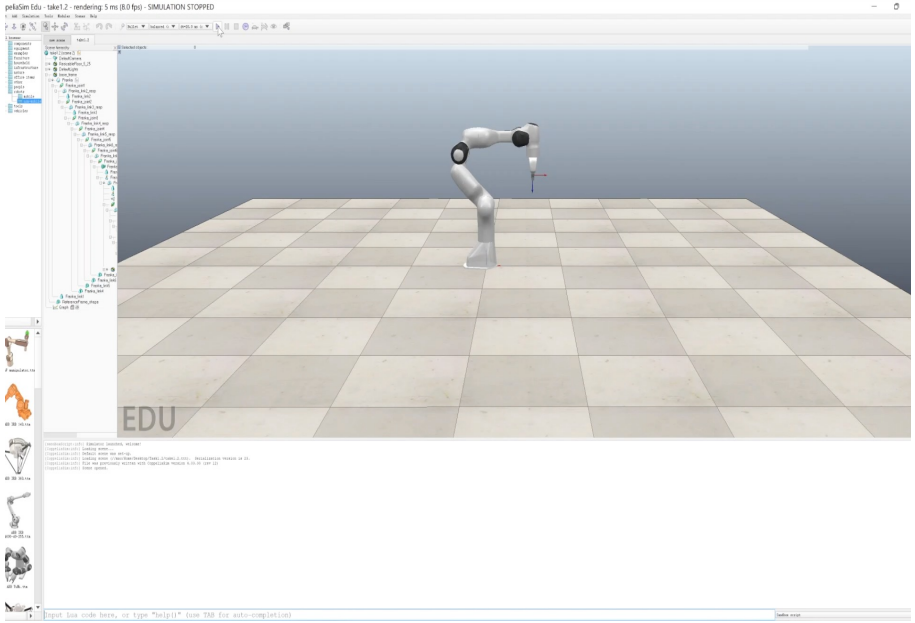
collision: False	155
collision Group:4 FrankaGripper_visible / Franka_link6	1
collision Group:9 FrankaGripper_visible / Franka_link6	1
collision Group:19 Franka_link3 / FrankaGripper_leftFinger_visible	1
collision Group:36 Franka_link2 / FrankaGripper_leftFinger_visible	1
collision Group:40 Franka_link2 / FrankaGripper_visible	1
collision Group:54 Franka / Franka_link7	1
collision Group:72 Franka / FrankaGripper_leftPad_visible	1
collision Group:80 Franka_link2 / FrankaGripper_visible	1
collision Group:84 Franka / Franka_link6	1
collision Group:86 Franka / Franka_link8	1
collision Group:127 FrankaGripper_visible / Franka_link6	1
collision Group:133 Franka / Franka_link7	1
collision Group:151 Franka / Franka_link6	1
collision Group:163 Franka / FrankaGripper_visible	1
collision Group:170 Franka / Franka_link8	1

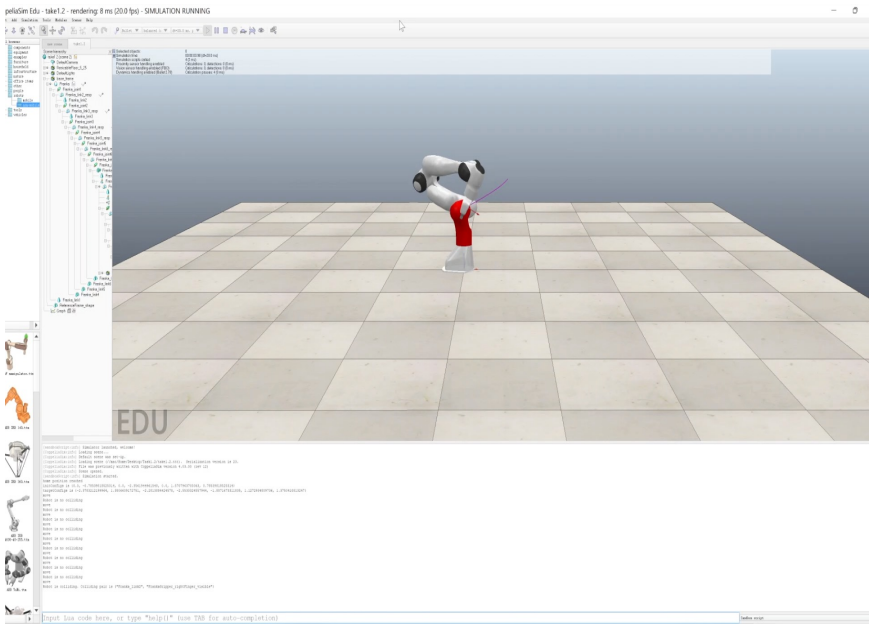
Task 1.2 dentify and avoid self-collision from given joint

Implementation process



Normalized trajectories (purple lines)





Detect Self-collision

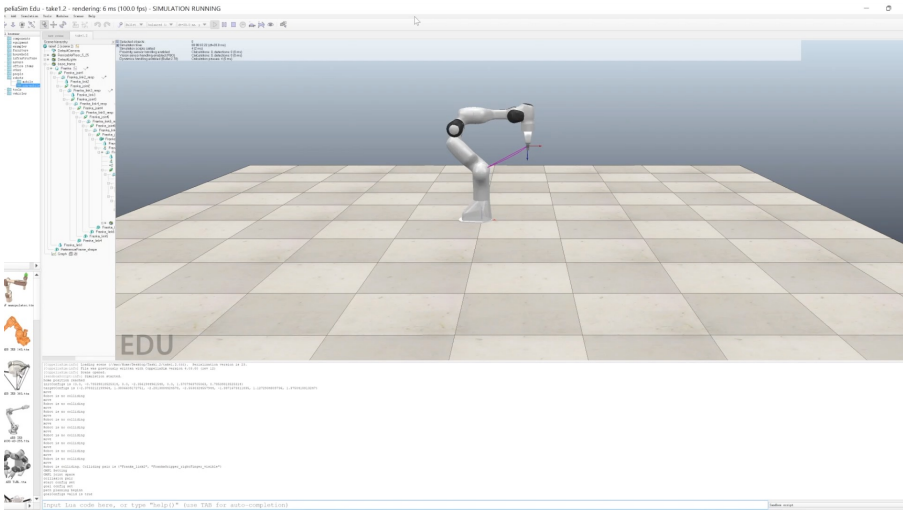
Joint poses {-2.3783212199964, 1.3806609172751, -2.2813884424578, -2.5530324557944, -1.8871673311835, 1.1272934809736, 1.3750410813247}

Collidon Pairs is {"Franka_link2", "FrankaGripper_rightFinger_visible"}

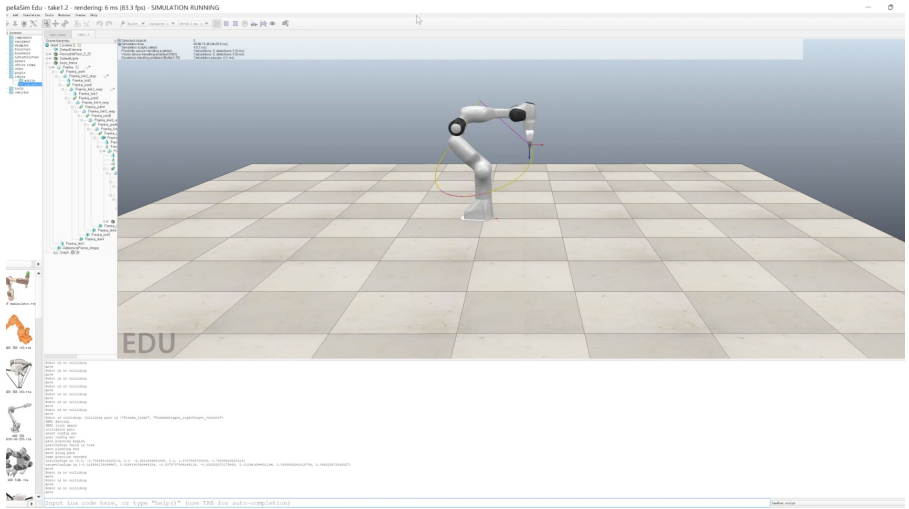
Return to start position

q_start = {{0, -M_PI_4, 0, -3 * M_PI_4, 0, M_PI_2, M_PI_4}}

Curved trajectories (yellow line)

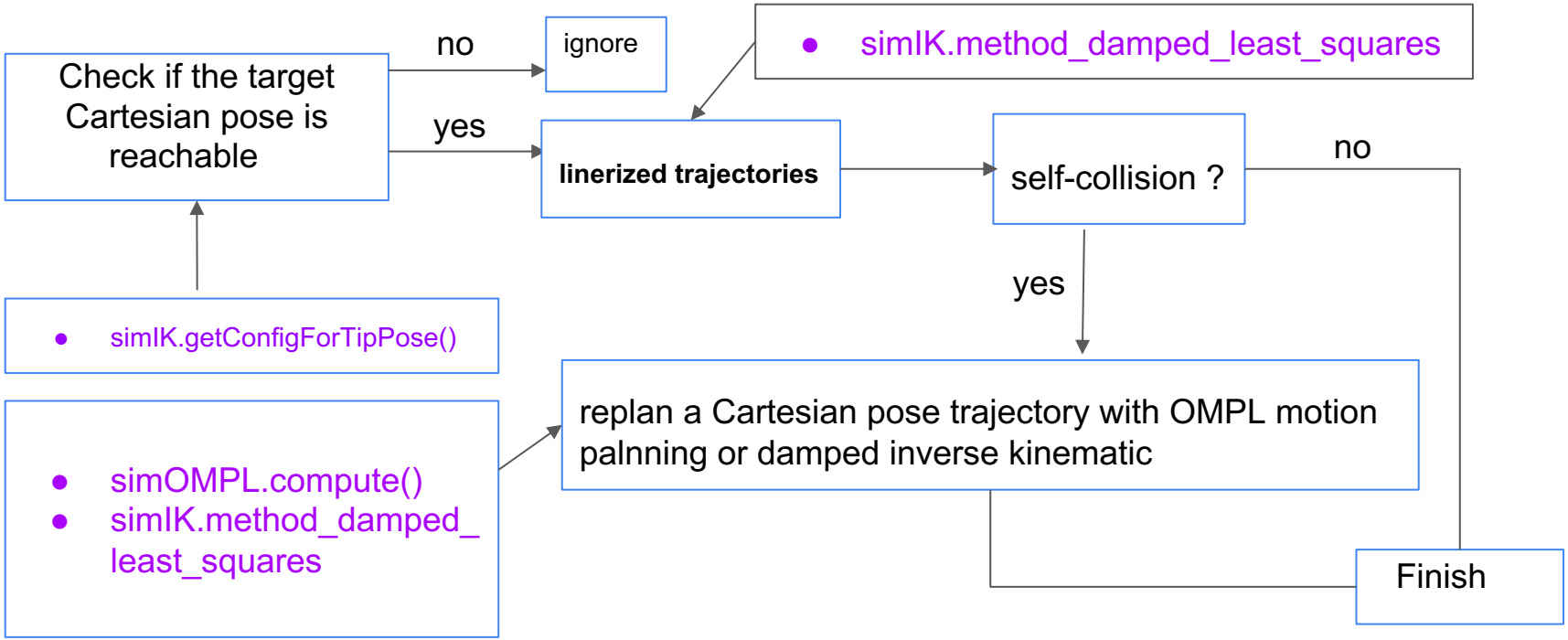


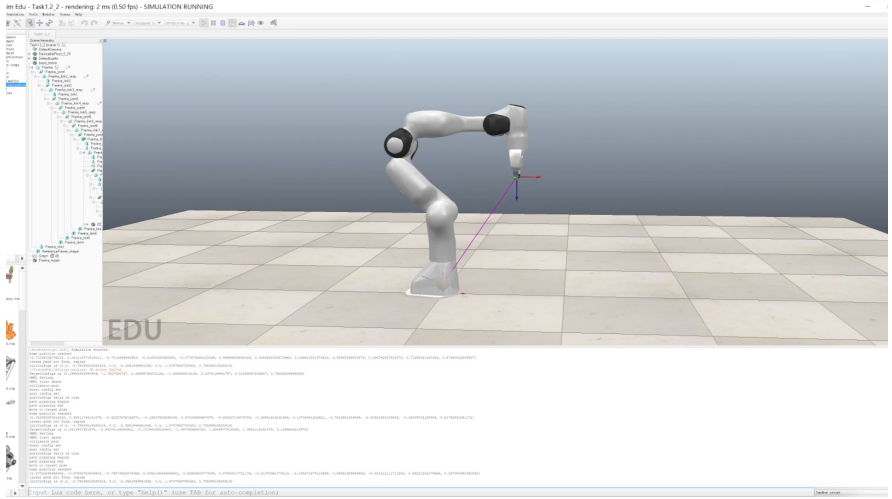
curved trajectories (OMPL)



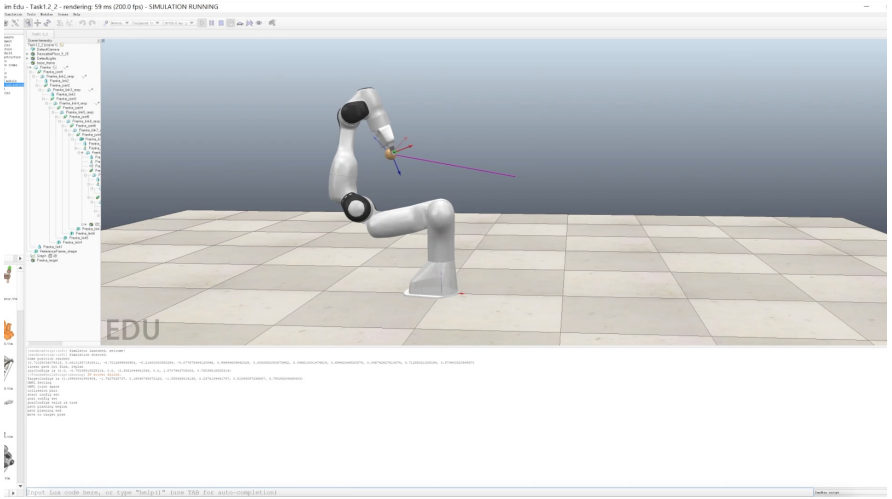
continue

Task 1.2 dentify and avoid self-collision from Cartesian trajectories

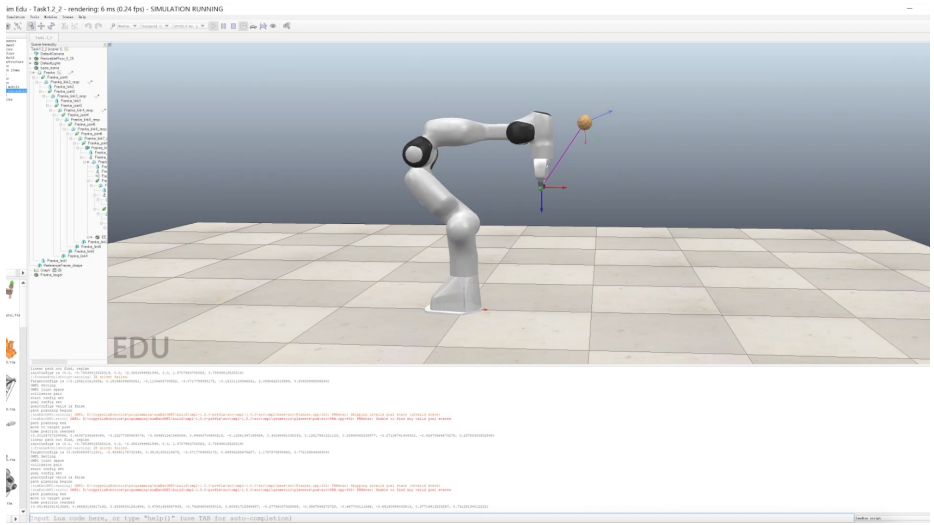




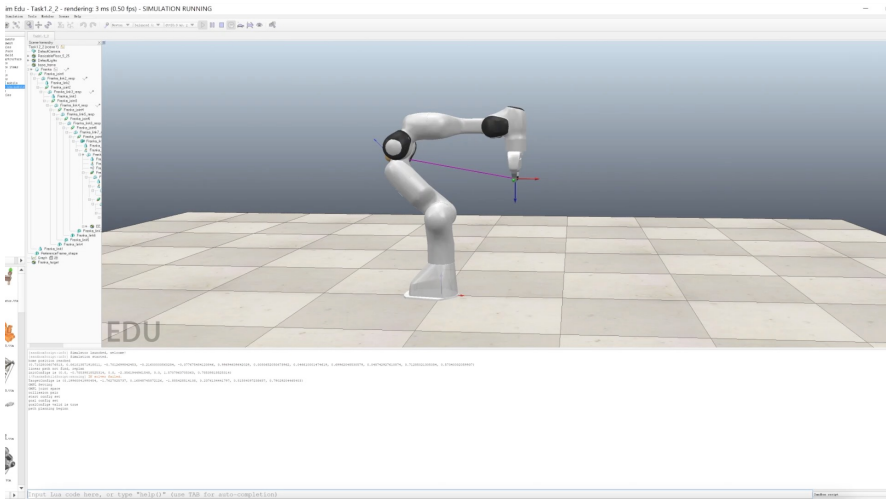
unreachable target Cartesian pose



reachable target Cartesian pose



linear trajectory



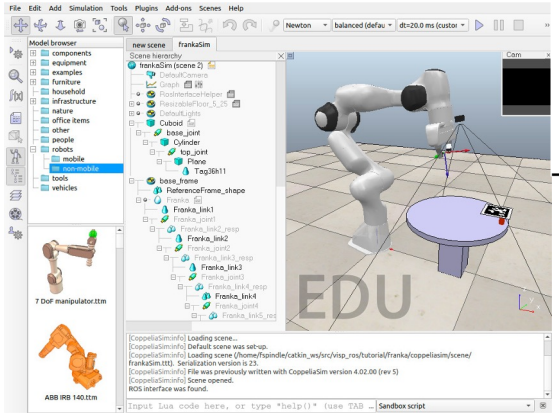
curved trajectories

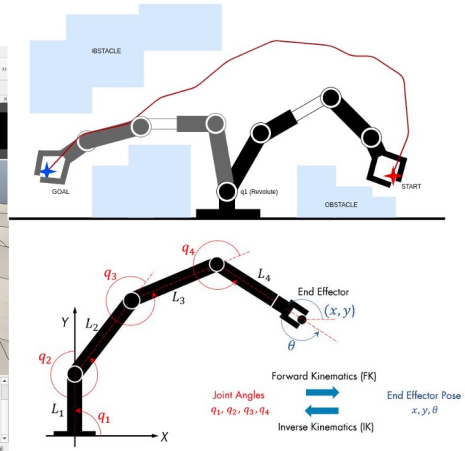
Task 2 - Obstacle Avoidance

Methods and Tools

1.CoppeliaSim with Lua

- OMPL (Open Motion Planning Library)
- IK (Inverse kinematics)

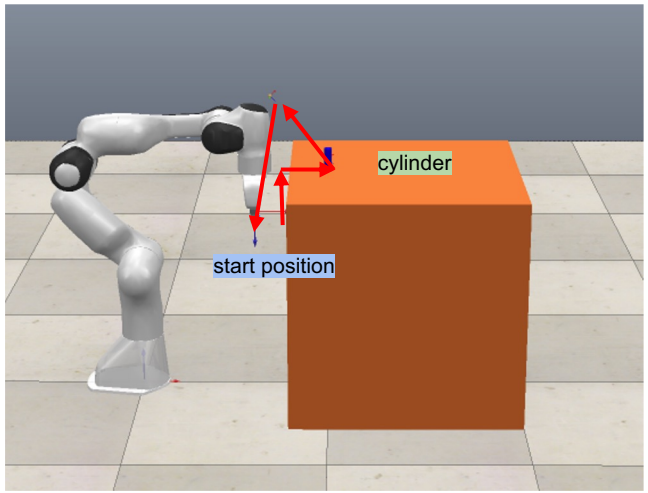


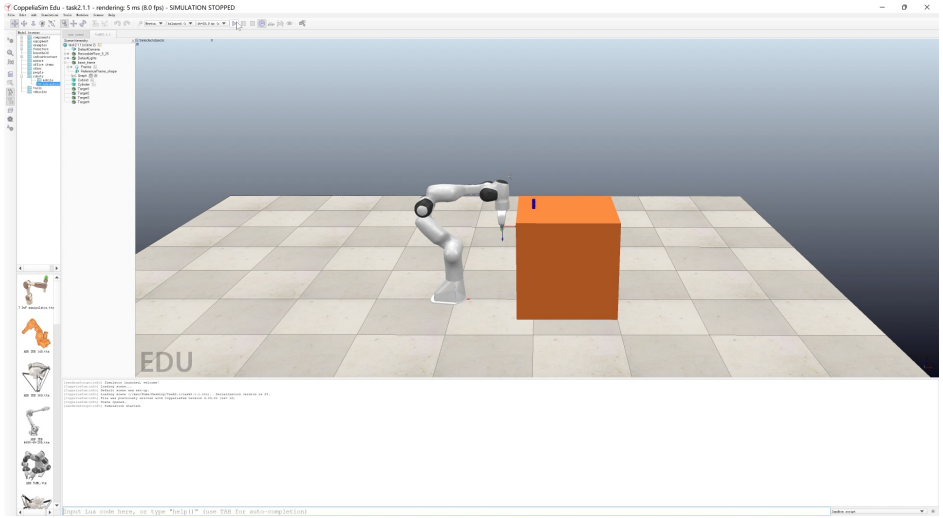


2. Anaconda with Pycharm

Task 2.1 End-effector trajectory planning (scenario 1)

Implementation process





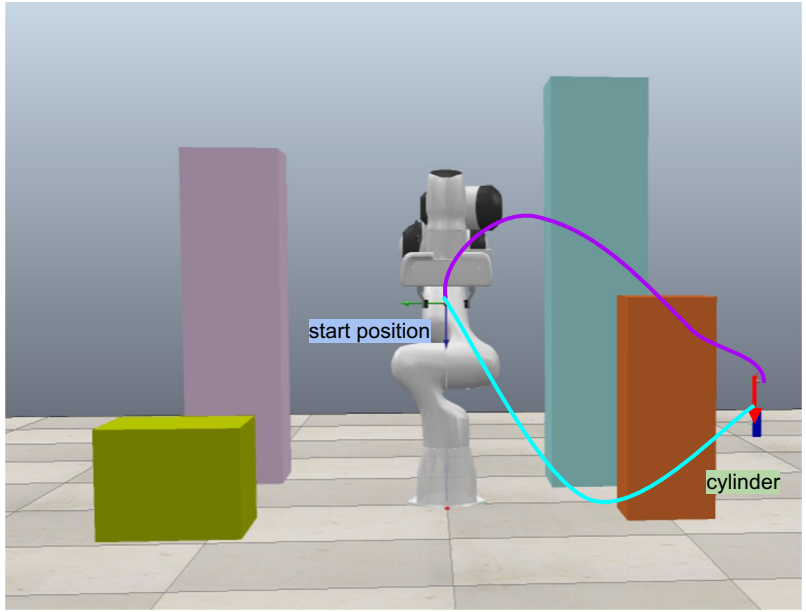
Motion path	
Motion 1	Move towards cylinder **(IK path)**
Motion 2	Grasp cylinder **(IK path)**
Motion 3	Lift cylinder **(IK path)**
Motion 4	Return to start position with cylinder**(IK path)**

Result

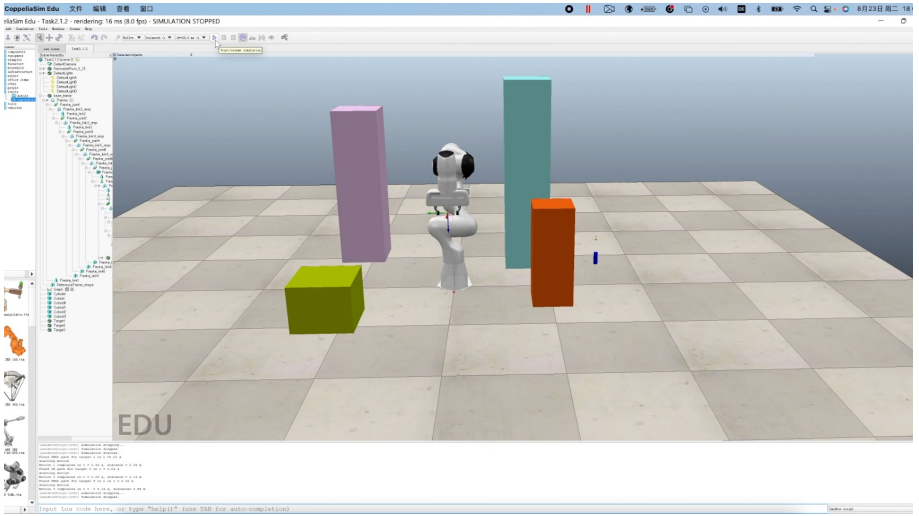
Example	
Motion 1	Found IK path for target 1 in **t = 0.02 s** Motion 1 completed in **t = 0.42 s** distance = **0.17 m**
Motion 2	Found IK path for target 2 in **t = 0.04 s** Motion 2 completed in **t = 0.66 s** distance = **0.15 m**
Motion 3	Found IK path for target 3 in **t = 0.02 s** Motion 3 completed in **t = 0.6 s** distance = **0.19 m**
Motion 4	Found IK path for target 4 in **t = 0.02 s**,Motion 4 completed in **t = 0.74 s** distance = **0.59 m**
Total Motion	Total time: t = **3.52 s** **(grasp the cylinder= 1s)** dlistance = **1.11m**

```
-- Print all path
local len1 = 0
local t1 = sim.getSimulationTime()
path,lengths=generateIkPath(getConfig(),sim.getObjectMatrix(target1,-1),ikSteps,true)
local t2 = sim.getSimulationTime()
print("Found IK path for target 1 in t = " .. round(t2 - t1,2) .. " s")
print("starting Motion 1")
visualizePath(path)
executeMotion(path,lengths,maxVel,maxAccel,maxJerk)
local t3 = sim.getSimulationTime()
print("Motion 1 completed in t = " .. round(t3 - t2,2) .. " s, distance = " .. round(lengths[ikSteps-1],2) .. " m")----target1
-----------------------------------------------------------------------------------------------------
len1 = len1 + lengths[ikSteps-1]
local t4 = sim.getSimulationTime()
path,lengths=generateIkPath(getConfig(),sim.getObjectMatrix(target2,-1),ikSteps,true)
local t5 = sim.getSimulationTime()
print("Found IK path for target 2 in t = " .. round(t5 - t4,2) .. " s")
print("starting Motion 2")
visualizePath(path)
executeMotion(path,lengths,maxVel,maxAccel,maxJerk)
local t6 = sim.getSimulationTime()
print("Motion 2 completed in t = " .. round(t6 - t5,2) .. " s, distance = " .. round(lengths[ikSteps-1],2) .. " m")----target2
-----------------------------------------------------------------------------------------------------
len1 = len1 + lengths[ikSteps-1]
 -- grasp the cylinder
print("grasp the cylinder")
sim.setInt32Signal("hand",1)
sim.wait(1)
-----------------------------------------------------------------------------------------------------
local t7 = sim.getSimulationTime()
path,lengths=generateIkPath(getConfig(),sim.getObjectMatrix(target3,-1),ikSteps,true)
local t8 = sim.getSimulationTime()
print("Found IK path for target 3 in t = " .. round(t8 - t7,2) .. " s")
print("starting Motion 3")
visualizePath(path)
executeMotion(path,lengths,maxVel,maxAccel,maxJerk)
local t9 = sim.getSimulationTime()
print("Motion 3 completed in t = " .. round(t9 - t8,2) .. " s, distance = " .. round(lengths[ikSteps-1],2) .. " m")----target3
-----------------------------------------------------------------------------------------------------
len1 = len1 + lengths[ikSteps-1]
local t10 = sim.getSimulationTime()
path,lengths=generateIkPath(getConfig(),sim.getObjectMatrix(target4,-1),ikSteps,true)
local t11 = sim.getSimulationTime()
print("Found IK path for target 4 in t = " .. round(t11 - t10,2) .. " s")
print("starting Motion 4")
visualizePath(path)
executeMotion(path,lengths,maxVel,maxAccel,maxJerk)
local t12 = sim.getSimulationTime()
print("Motion 4 completed in t = " .. round(t12 - t11,2) .. " s, distance = " .. round(lengths[ikSteps-1],2) .. " m")----target4
-----------------------------------------------------------------------------------------------------
len1 = len1 + lengths[ikSteps-1]
print("total time: t = " .. round(t12 - t1,2) .. " s (grasp the cylinder=1 s), distance = " .. round(len1,2) .. " m")
-----------------------------------------------------------------------------------------------------
end
```

Task 2.1 End-effector trajectory planning (scenes 2)



Implementation process



Motion path	
Motion 1	Move towards cylinder (**OMPL path**)
Motion 2	Grasp cylinder (**IK path**)
Motion 3	Return to start position with cylinder(**OMPL path**)

Result

Example	
Motion 1	Found OMPL path for target 1 in **t = 0.04 s** Motion 1 completed in **t = 1.75 s** distance = **1.58 m**
Motion 2	Found IK path for target 2 in **t = 0.04 s** Motion 2 completed in **t = 0.20 s** distance = **0.13 m**
Motion 3	Found OMPL path for target 3 in **t = 0.04 s** Motion 3 completed in **t = 1.82 s** distance = **1.92 m**
Total Motion	Total time: t = **3.89 s** **(grasp the cylinder= 1s)** dlistance = **3.63m**

```
for i=1,#jh,1 do
    ikJoints[i]=simToIkMap[jh[i]]
end
ikTip=simToIkMap[simTip]
local collection=sim.createCollection(0)
sim.addItemToCollection(collection,sim.handle_tree,simBase,0)
sim.setInt32Signal("robot_collection",collection)
-- 4 collision pairs: Robot and each of the four cuboids
obstacle1 = sim.getObjectHandle('/Cuboid4')
obstacle2 = sim.getObjectHandle('/Cuboid44')
obstacle3 = sim.getObjectHandle('/Cuboid43')
obstacle4 = sim.getObjectHandle('/Cuboid42')
obstacle5 = sim.getObjectHandle('/Cuboid45')
collisionPairs={collection,obstacle1,collection,obstacle2,collection,obstacle3,collection,obstacle4,collection,obstacle5}
maxVel=1
maxAccel=10
maxJerk=8000
forbidLevel=0
metric={0.2,1,0.8,0.1,0.1,0.1,0.1}
ikSteps=20
maxOMPLCalculationTime=2 -- for one calculation
OMPLAlgo=simOMPL.Algorithm.PRMstar-- the OMPL algorithm to use
```

```
all_paths = {}
all_lengths = {}
mean_length = 0
lengths_summed = 0
n_paths = 0
for ii = 1,0,1 do
    configs=findSeveralCollisionFreeConfigs(sim.getObjectMatrix(target3, -1),300,5)
    path,lengths=findPath(getConfig(),configs)
    if lengths[200] and path then
        lengths_summed = lengths_summed + lengths[200]
        n_paths = n_paths + 1
    end
end
if n_paths then
end

local t8 = sim.getSimulationTime()
local configs=findSeveralCollisionFreeConfigs(sim.getObjectMatrix(target3, -1),300,5)
local t9 = sim.getSimulationTime()
 --print("Found collision-free target configuration in t = " .. round(t2 - t1,2) .. " s")
path,lengths=findPath(getConfig(),configs)
local t10 = sim.getSimulationTime()
print("Found OMPL path for target 3 in t in t = " .. round(t10 - t9,2) .. " s")
print("entering motion")
if path then
```

Task 2.2. Joint obstacle avoidance - exploiting null space

nullspace theory

Nullspace Base Matrix

$$J(q)Z(q)^T = 0$$

General Case

$$J(q) = [J_m(q), J_r(q)]$$
$$Z(q) = [Z_m(q), Z_r(q)]$$

$$Z_m(q)^T = -J_m(q)^{-1}J_r(q)$$
$$Z_r(q)^T = \mathrm{I}$$

One-Dimensional Nullspace

$$J_Z(q) = \begin{pmatrix} J(q) \\ Z(q) \end{pmatrix}$$

$$z_i(q) = (-1)^{n+i}\det(J_i(q)) \qquad J_i(q) \in \mathbb{R}^{m\times m}$$

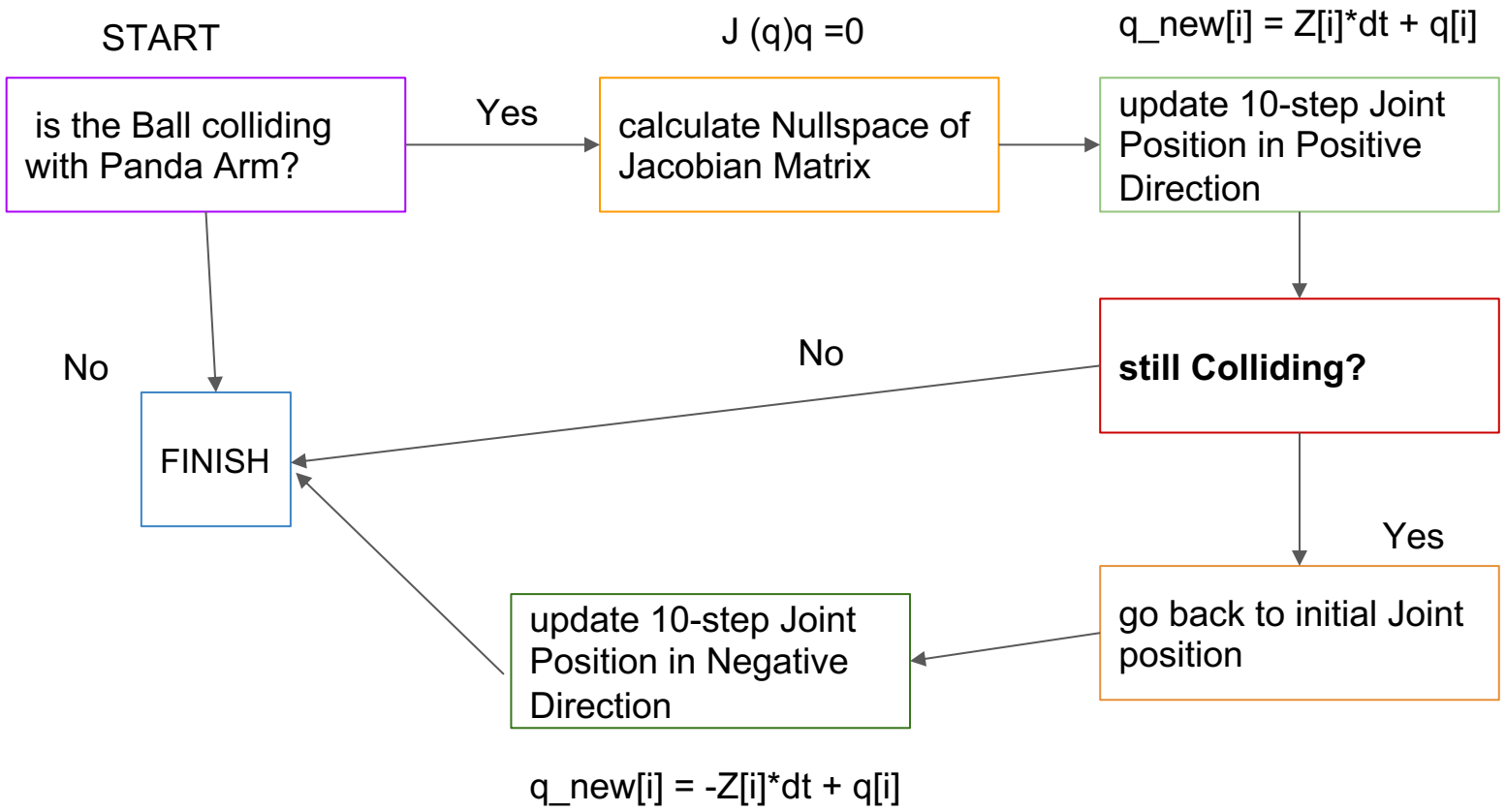
```
function getJacob(ikEnvd,ikGroupd,ikGroupName)  --calculate  Jacob Matrix
    jointsu =simIK.computeJacobian(ikEnvd,ikGroupd,1)
    if jointsu then
        ikGroupHandle = simIK.getIkGroupHandle(ikEnvd,ikGroupName)
        jacobian,matrixSize=simIK.getJacobian(ikEnvd,ikGroupHandle)
        local n = matrixSize[2] -- dimension 6

        local m = matrixSize[1] -- num of joint 7

        local J = Matrix(n,m)
        for i = 1,m,1 do
            for j = 1,n,1 do
                J[j][m-i+1] = jacobian[(j-1)*m+i]
            end
        end

        return J

    else
        return nil
    end

end
```

```
function nullSpaceBasic(J)   --calculate  NullSpace
    local Jr,Jm
    local Zrt,Zmt
    Z = Matrix(7,1)
    Jm = J:slice(1,1,6,6) -- 6x6
    Jr = J:slice(1,7,6,7) -- 6x1
    invJm = Matrix.inv(Jm) -- 6x6
    Zmt = -invJm*Jr -- 6x6x6x1 = 6x1
    Zrt = Matrix:eye(1) -- 1x1
    Zt = Zmt:vertcat(Zrt) --7x1
    Z = Zt:t() --1x7
    return Z
end
```

```
function detOmmitted(J,i)  --General Case
    local Jo = Matrix(6,6)
    if i == 1 then
        Jo = J:slice(1,2,6,7)
    elseif i == 7 then
        Jo = J:slice(1,1,6,6)
    else
        J_l = J:slice(1,1,6,i-1)
        J_r = J:slice(1,i+1,6,7)
        Jo = Matrix.horzcat(J_l, J_r)
    end
    return Matrix.det(Jo)
end
```

```
function nullSpaceBasic_1(J)
    local Z = Matrix(7,1)
    for i = 1,7,1 do
        jointdetJi = detOmmitted(J,i)
        Z[i] = (-1)^(7+i)*jointdetJi  --One-Dimensional Nullspace
    end
    return Z
end
```

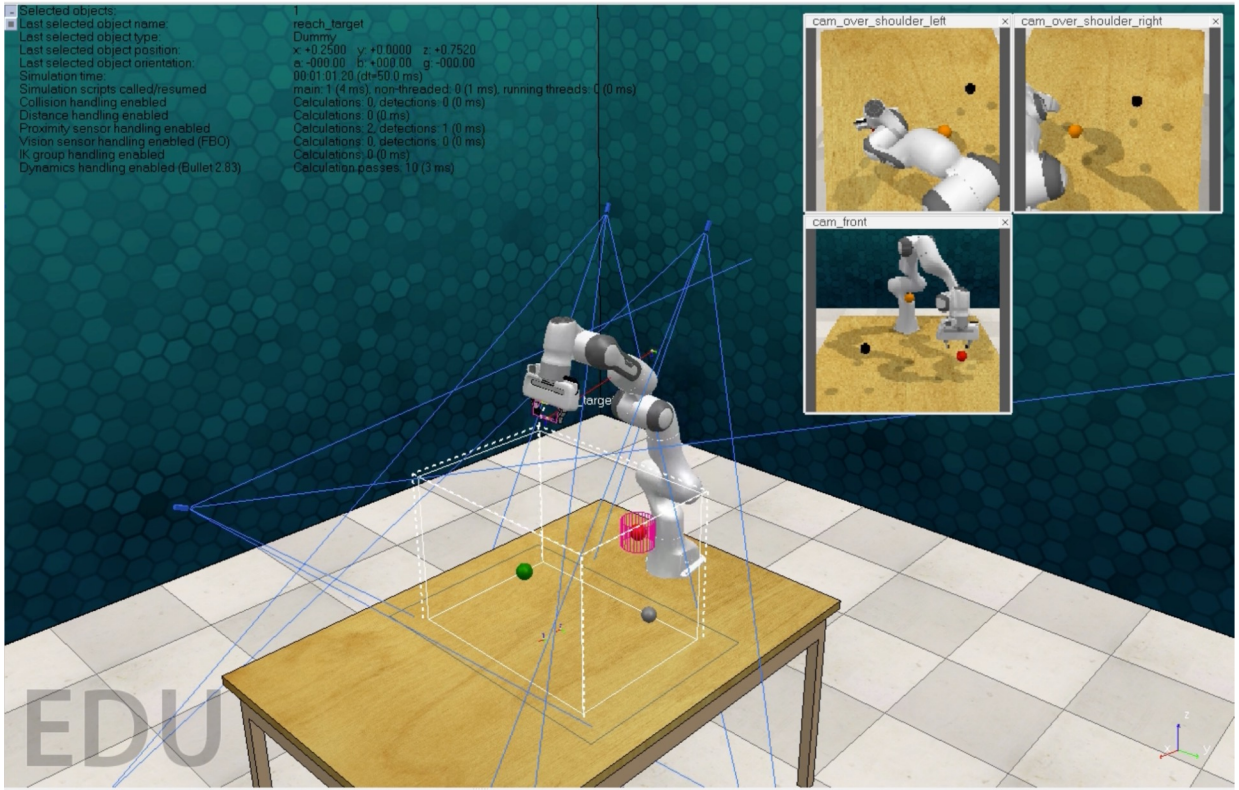
Null space obstacle avoidance flow chart



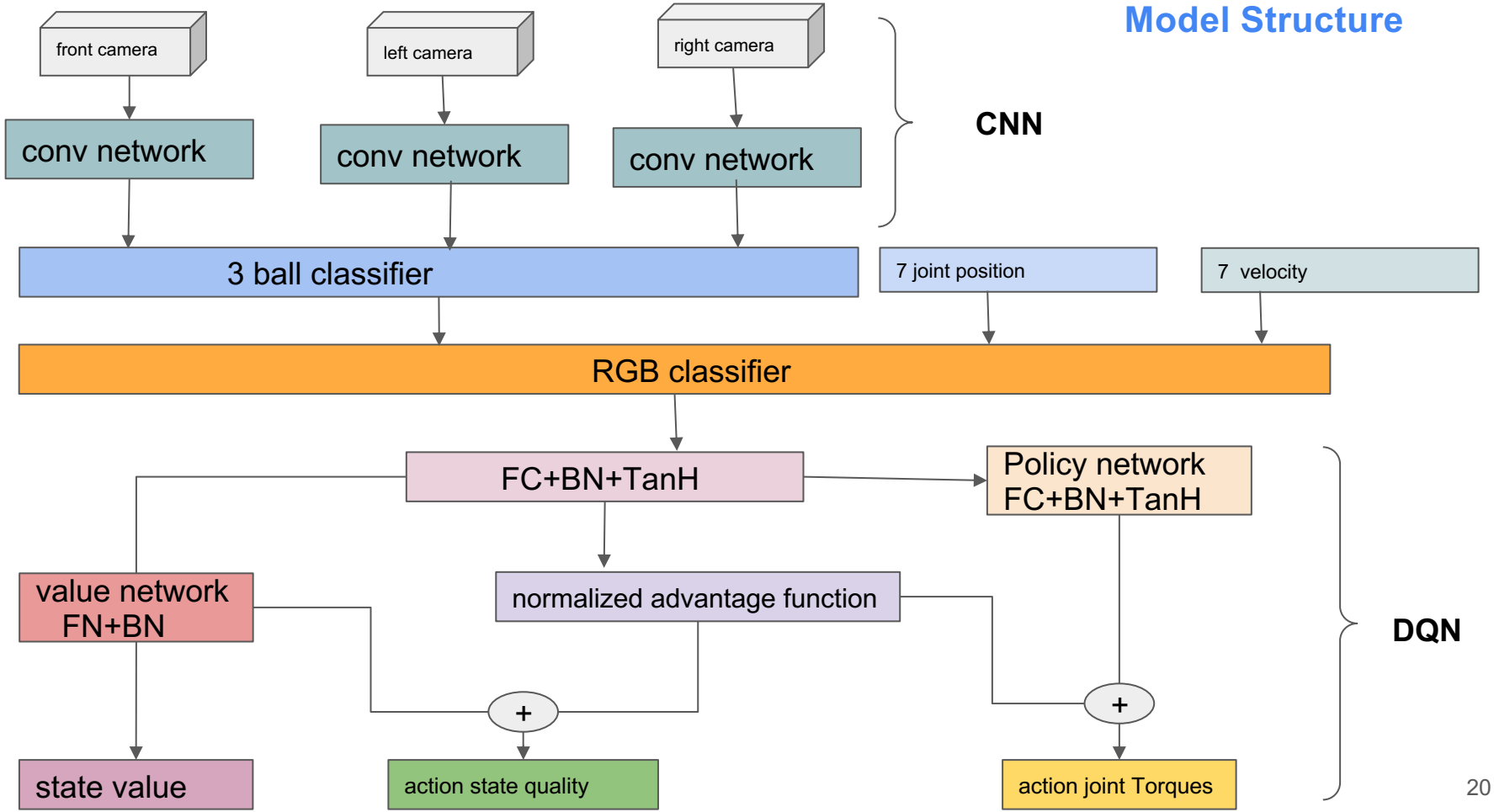
Task 2.3. Learning - vision-based obstacle avoidance (RLBench)

Train scenario

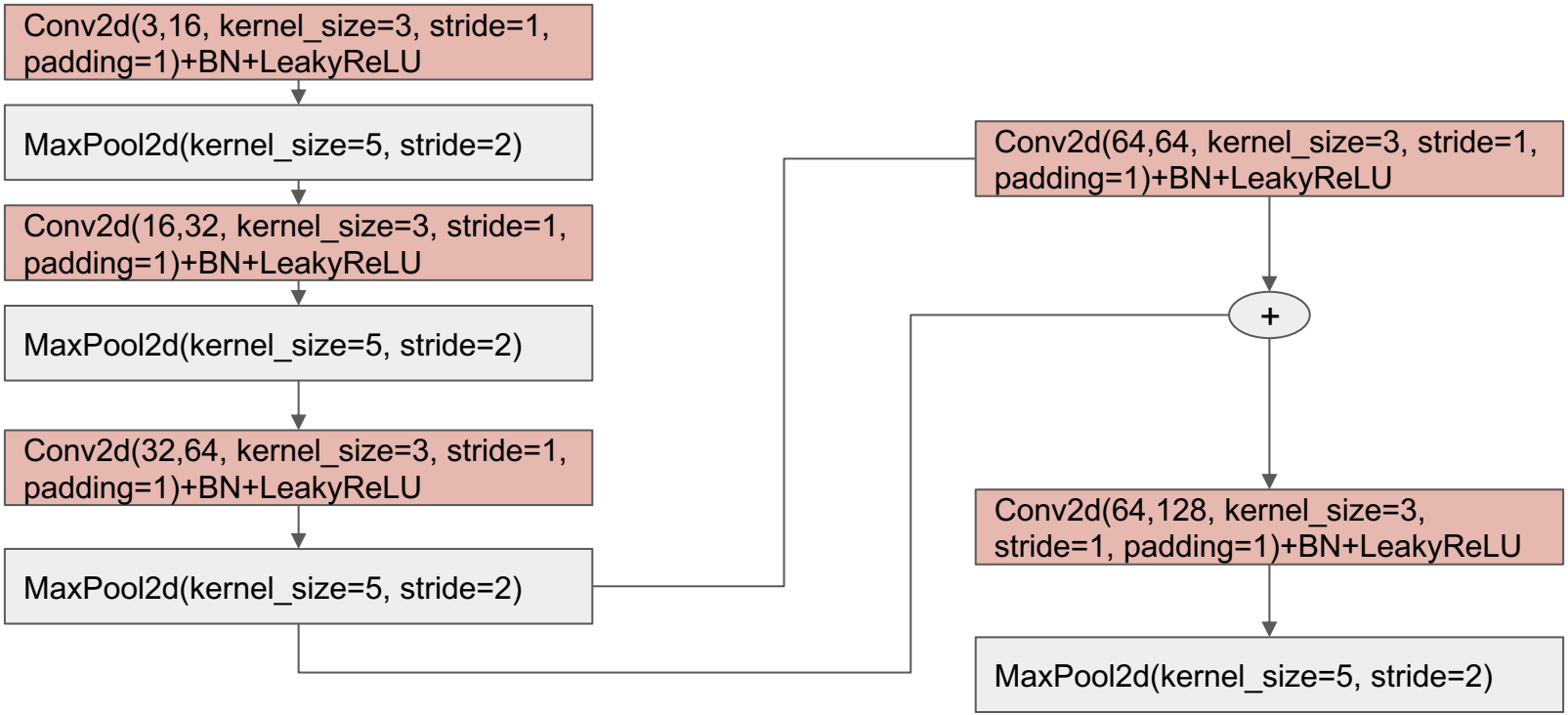
2 obstacle balls with 3 cameras



Model Structure

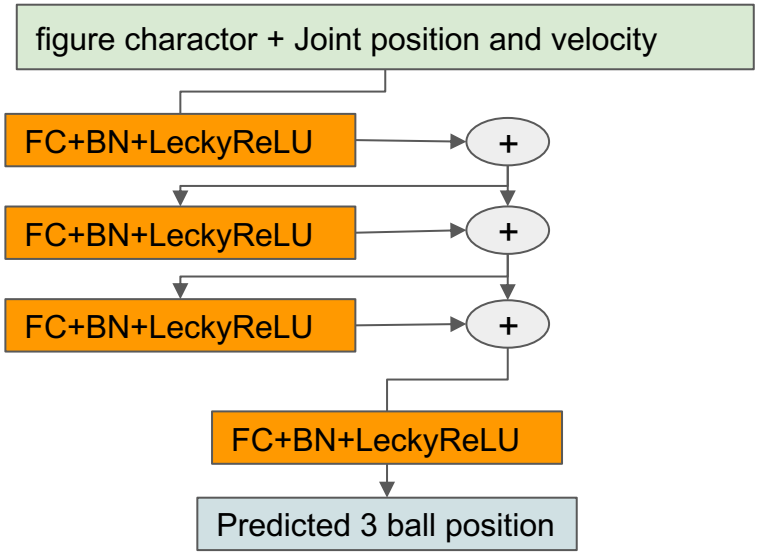


Detailed network structure: **Convolutional neural network**

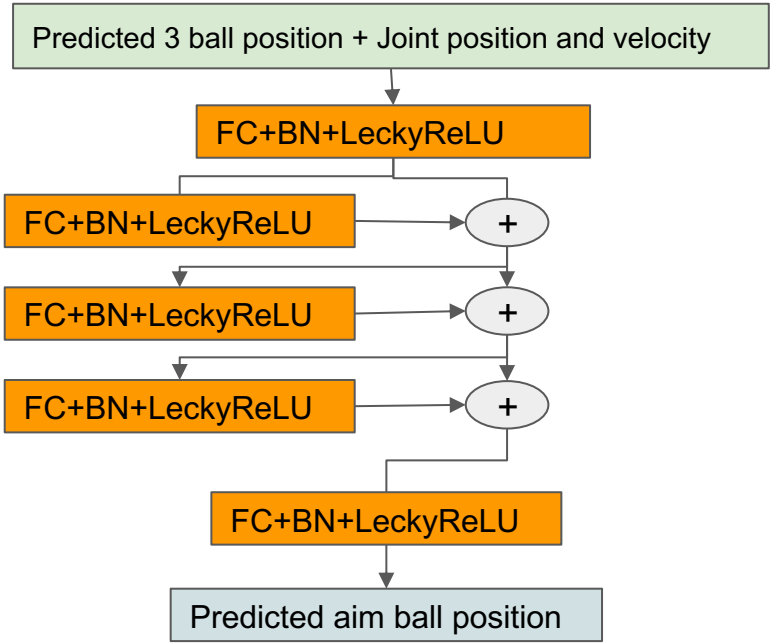


Detailed network structure: **Fully connection network**

3 ball classifier



RGB classifier



- **optimizer: RMSProp**
- **Loss：LSE**

Task23 – robustness.py

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Task23 robustness.py reaching_task_simulation

Project

trained_model6.pt
trained_model7.pt
trained_model8.pt
trained_model9.pt
trained_model100.pt
qLearner
networks.py
qAgent.py
replayMemory.py
training_tracker.py
utils.py
team_h
bin
compiler_compat
conda-meta
CoppeliaSim_Edu_V4_1_0_Ubuntu20_04
etc

robustness.py

```
obs_config.overh
obs_config.wrist
obs_config.front
obs_config.joint
obs_config.joint
obs_config.joint
obs_config.gripp
obs_config.gripp
obs_config.gripp
obs_config.gripp
obs_config.gripp
obs_config.wrist
obs_config.task_

if planning:
    arm_action_m
else:
    arm action m
```

setup_env()

Run: reaching_task_simulation

```
---------- Reset Episode --------------
Episode number:  76
---------- Reset Episode --------------
Episode number:  77
---------- Reset Episode --------------
Episode number:  78
---------- Reset Episode --------------
Episode number:  79
---------- Reset Episode --------------
Episode number:  80
---------- Reset Episode --------------
Episode number:  81
---------- Reset Episode --------------
Episode number:  82
---------- Reset Episode --------------
Episode number:  83
```

Version Control | Run | Python Packages | TODO | Python Console | Problems | Terminal

CoppeliaSim

File Edit Add Simulation Tools Plugins Add-ons Scenes Help

Model browser

components
equipment
examples
furniture
household
infrastructure
nature
office items
other
people
robots
mobile
non-mobile
tools
vehicles

7 DoF manipulator.ttm

ABB IRB 140.ttm

task_design

Scene hierarchy

DefaultCamera
Dummy
remoteApi
workspace
cam_cinematic_base
cam_front
cam_over_shoulder_left
cam_over_shoulder_left_mask
cam_over_shoulder_right
cam_over_shoulder_right_mask
diningTable
ResizableFloor_5_25
DefaultLights
reach_target
boundary
distractor0
distractor1
target
success
waypoint0
Wall1
XYZCameraProxy
Panda
Panda_link0_visual
Panda_target
Panda_joint1
Panda_link1_respondable
Panda_link1_visual
Panda_joint2
Panda_link2_respondable
Panda_link2_visual
Panda_joint3
Panda_link3_respondable
Panda_link3_visual
Panda_joint4
Panda_link4_visual
Panda_joint5
Panda_link5_visual
Panda_joint6
Panda_link

Selected objects: 1
Last selected object name: reach_target
Last selected object type: Dummy
Last selected object position: x: +0.2500 y: +0.0000 z: +0.7520
Last selected object orientation: a: -000.00 b: +000.00 g: -000.00
Simulation time: 00:05:01.35 (dt=50.0 ms)
Simulation scripts called/resumed: main: 1 (4 ms), non-threaded: 0 (0 ms), running: 0 (0 ms)
Collision handling enabled: Calculations: 0, detections: 0 (0 ms)
Distance handling enabled: Calculations: 0 (0 ms)
Proximity sensor handling enabled: Calculations: 2, detections: 1 (0 ms)
Vision sensor handling enabled (FBO): Calculations: 0, detections: 0 (0 ms)
IK group handling enabled: Calculations: 0 (0 ms)
Dynamics handling enabled (Bullet 2.83): Calculation passes: 10 (4 ms)

cam_over_sl
cam_over_sl
cam_front

EDU

[sandboxScript:info] Simulator launched, welcome!
[sandboxScript:info] Simulation started.

Input Lua code here, or type "help()" (use TAB for auto-completion)

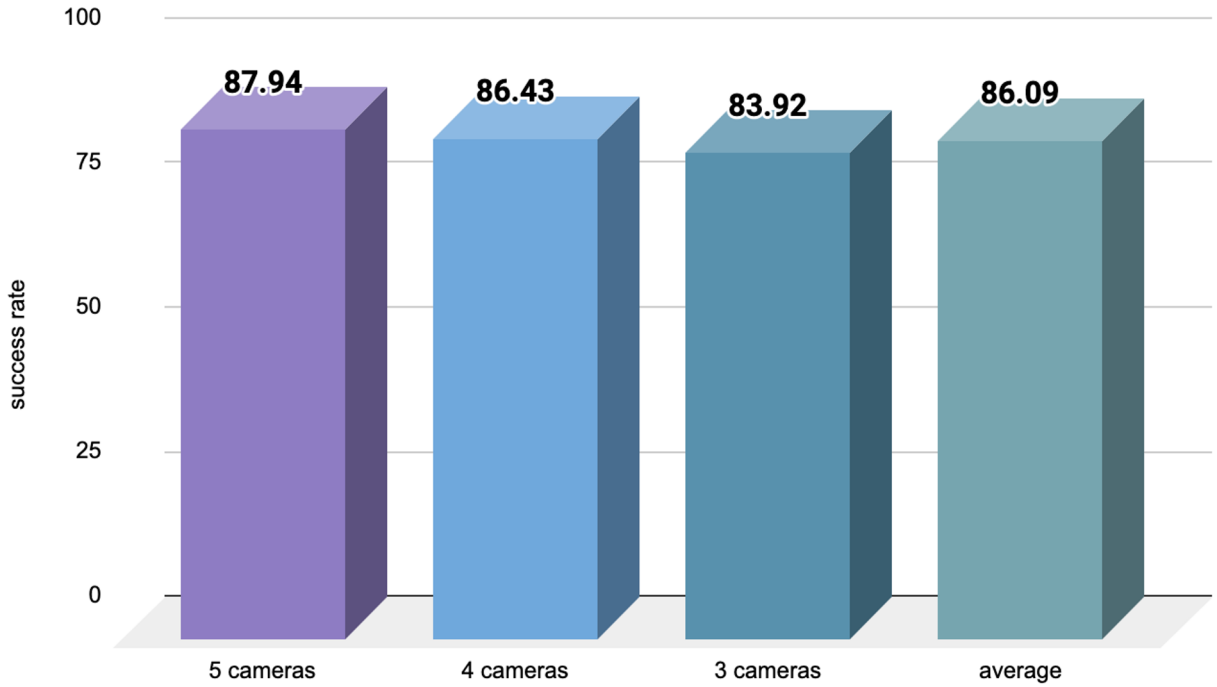
Sandbox script

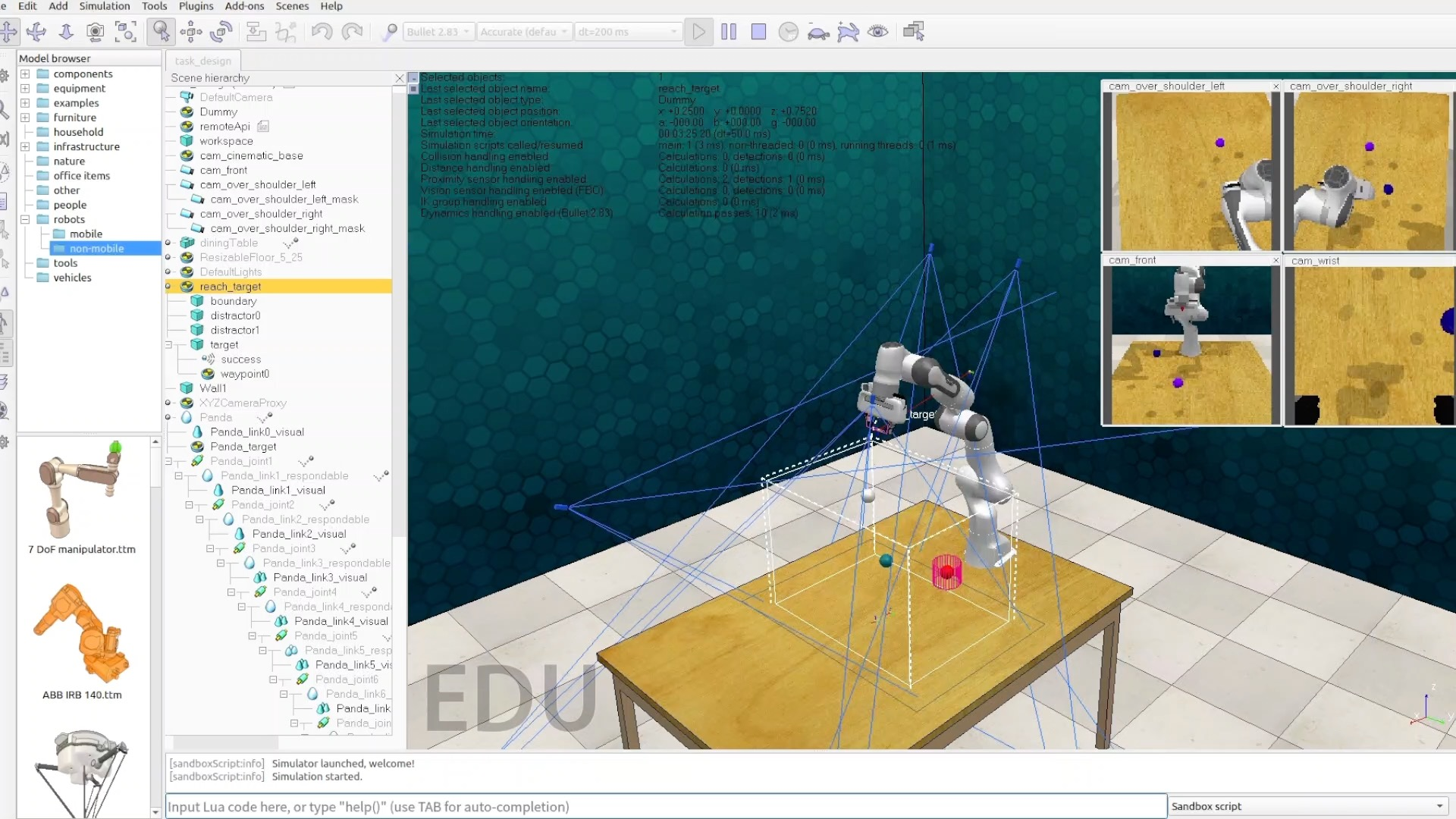
Result

Trained model 8 (best model)

Test scenarios

1. 2 obstacle balls with 5 cameras
2. 2 obstacle balls with 4 cameras
3. 2 obstacle balls with 3 cameras



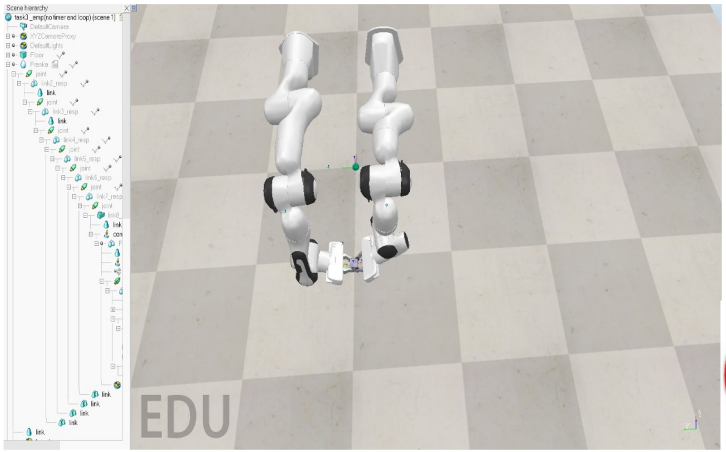

Edit Add Simulation Tools Plugins Add-ons Scenes Help
Model browser
task_design
Scene hierarchy
components
equipment
examples
furniture
household
infrastructure
nature
office items
other
people
robots
mobile
non-mobile
tools
vehicles
DefaultCamera
Dummy
remoteApi
workspace
cam_cinematic_base
cam_front
cam_over_shoulder_left
cam_over_shoulder_left_mask
cam_over_shoulder_right
cam_over_shoulder_right_mask
diningTable
ResizableFloor_5_25
DefaultLights
reach_target
boundary
distractor0
distractor1
target
success
waypoint0
Wall1
XYZCameraProxy
Panda
Panda_link0_visual
Panda_target
Panda_joint1
Panda_link1_respondable
Panda_link1_visual
Panda_joint2
Panda_link2_respondable
Panda_link2_visual
Panda_joint3
Panda_link3_respondable
Panda_link3_visual
Panda_joint4
Panda_link4_visual
Panda_joint5
Panda_joint6
Selected objects: 1
Last selected object name: reach_target
Last selected object type: Dummy
Last selected object position: x +0.2500 y +0.0000 z: +0.7520
Last selected object orientation: a: -000.00 b: +000.00 g: -000.00
Simulation time: 00:03:25.20 (dt=50.0 ms)
Simulation scripts called/resumed: main: 1 (3 ms), non-threaded: 0 (0 ms), running threads: 0 (1 ms)
Collision handling enabled: Calculations: 0, detections: 0 (0 ms)
Distance handling enabled: Calculations: 0 (0 ms)
Proximity sensor handling enabled: Calculations: 2, detections: 1 (0 ms)
Vision sensor handling enabled (FBO): Calculations: 0, detections: 0 (0 ms)
IK group handling enabled: Calculations: 0 (0 ms)
Dynamics handling enabled (Bullet 2.83): Calculation passes: 10 (2 ms)
cam_over_shoulder_left
cam_over_shoulder_right
cam_front
cam_wrist
7 DoF manipulator.ttm
ABB IRB 140.ttm
[sandboxScript:info] Simulator launched, welcome!
[sandboxScript:info] Simulation started.
Input Lua code here, or type "help()" (use TAB for auto-completion)
Sandbox script

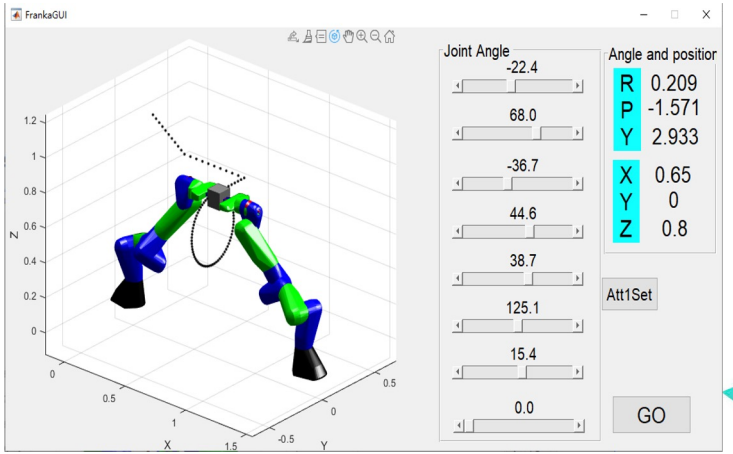
Task 3 Bimanual Manipulation

Methods and Tools

1. CoppeliaSim with Lua

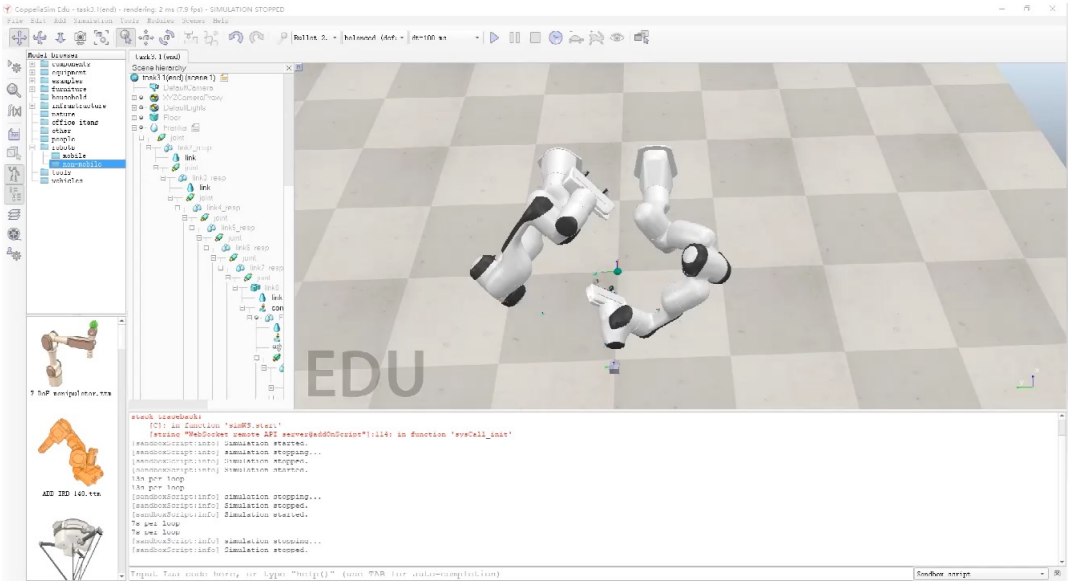
2. Matlab GUI





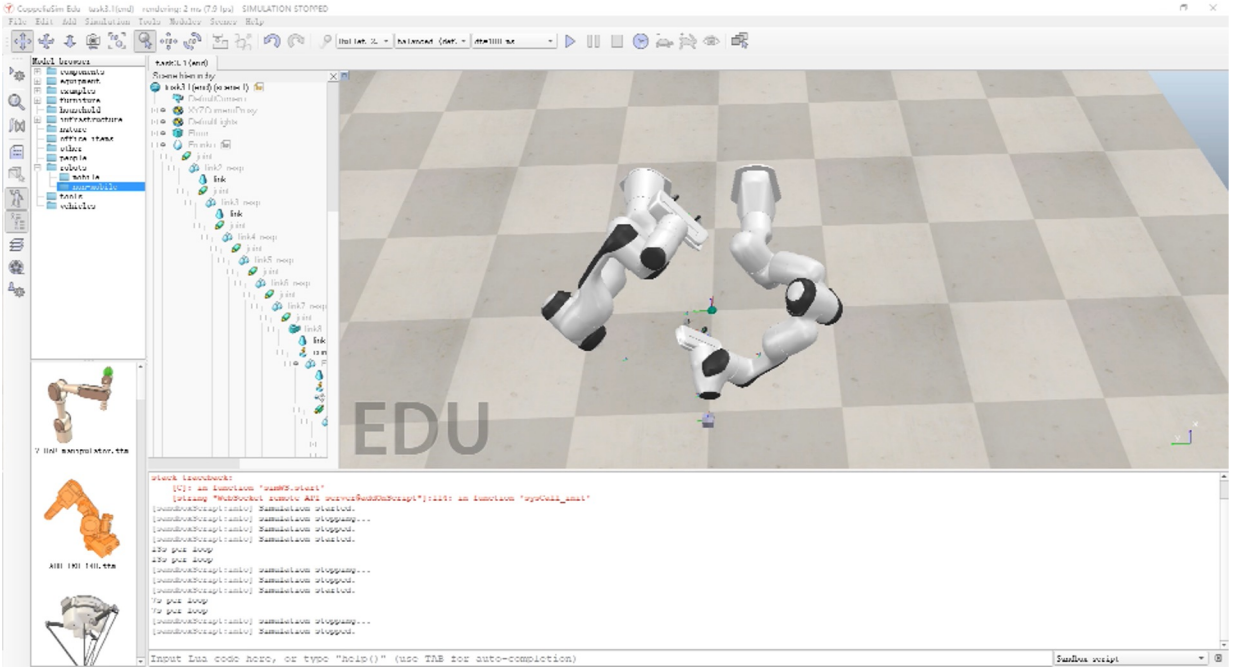
Task 3.1 End-effector trajectory planning

- Manual setting of the start pose of the arm and the final pose when gripping the object
- The robotic arm uses the coppeliasim's own **physics engine** to find the most suitable path.
- The ends of the two robotic arms are alternately switched on and off by calling the **pause command** and **cycling** the process.



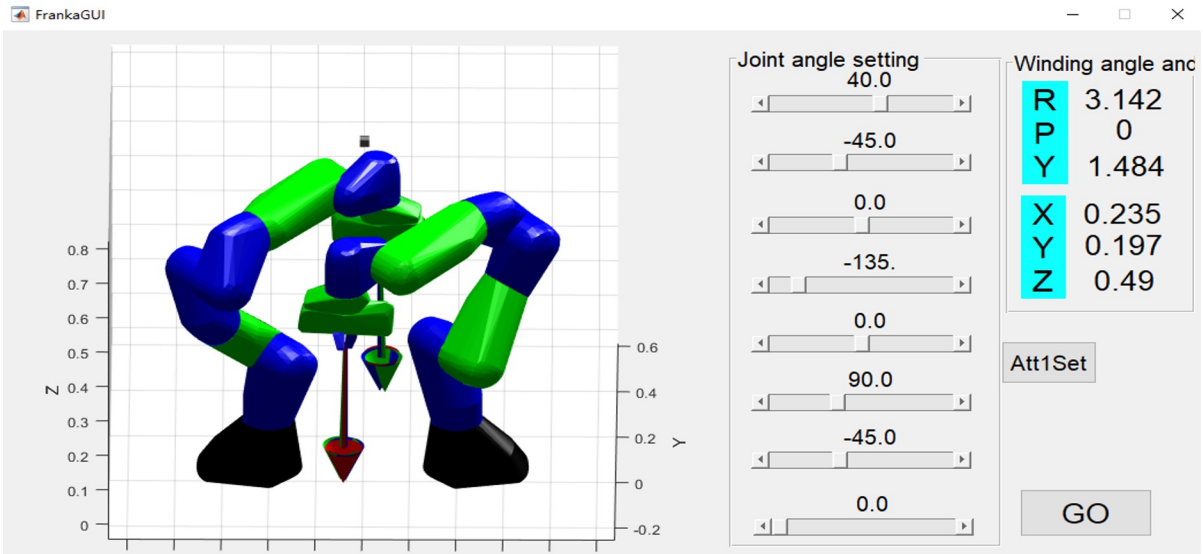
Problems of 3.1

- The actual pinch time should not be an integer
- Lack of knowledge to set the exact pinch state
- print the already set pause

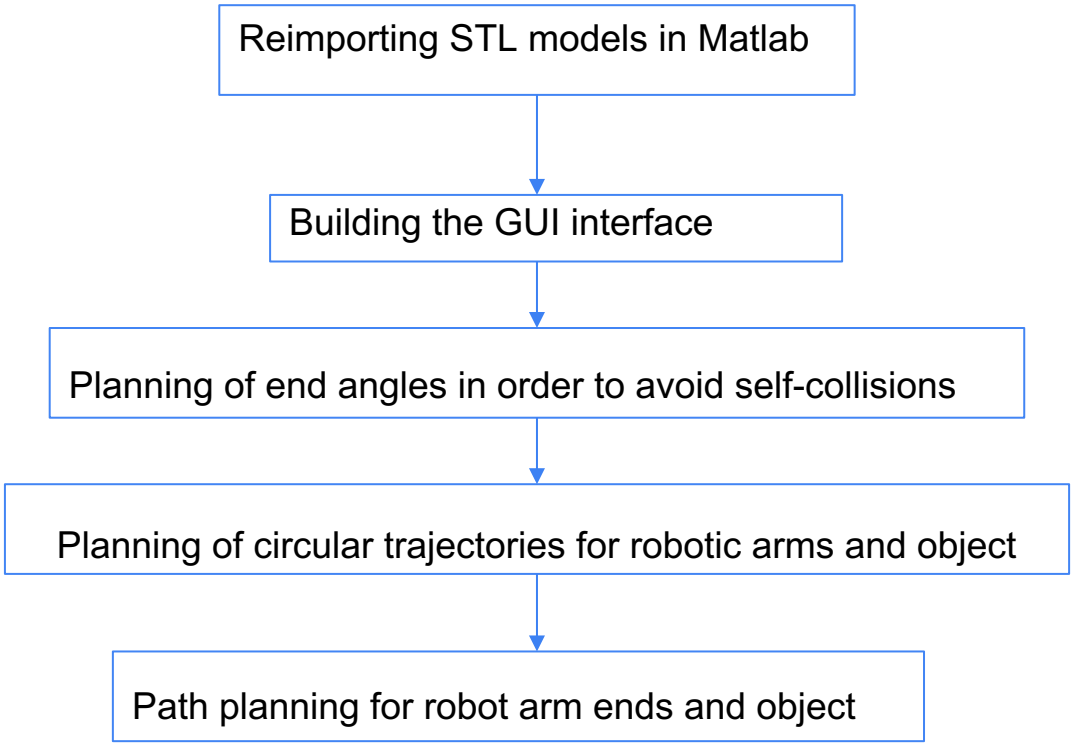


Implementation process

- Build GUI interface
- Import the joint model of the robot arm and build the object model
- Set the transport path and check for self-collision
- The carrying path is represented by interpolation with the insertion of black dots.
- Save results in `FrankaGUI.m`

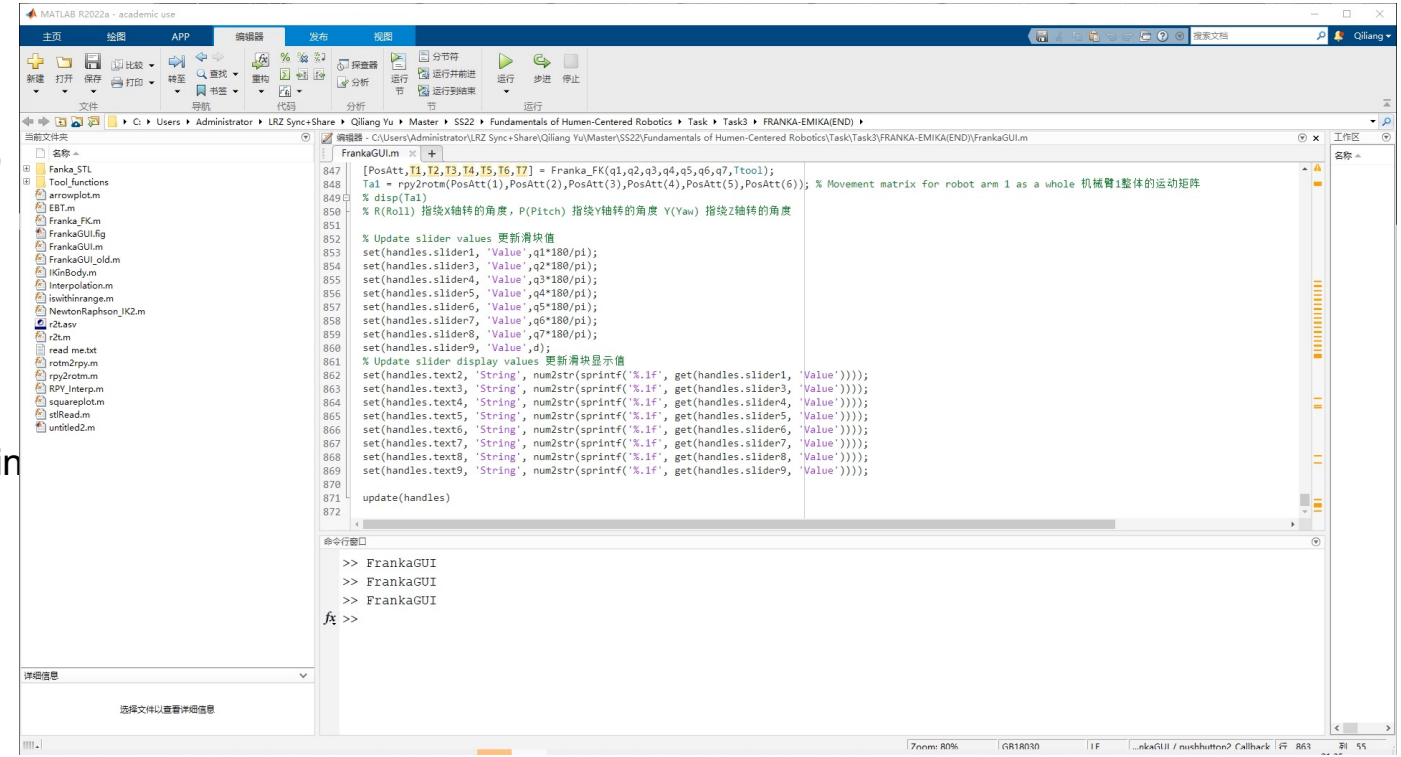


Task 3.2 End-effector trajectory planning



Result

- The left robot arm carries the object to the starting point
- The end of the right robot arm moves to the starting point
- Collaborative handlin and return to the starting point



Problems and ideas

- Adjust the starting pose and distance between 2 arms, a more perfect trajectory can be achieved.
- No suitable physics engine was found for Matlab, so complex trajectory planning was required for the end of the robot arm.**(better with Coppliasim)**
- Try to use collision avoidance algorithem to obtain more possible paths