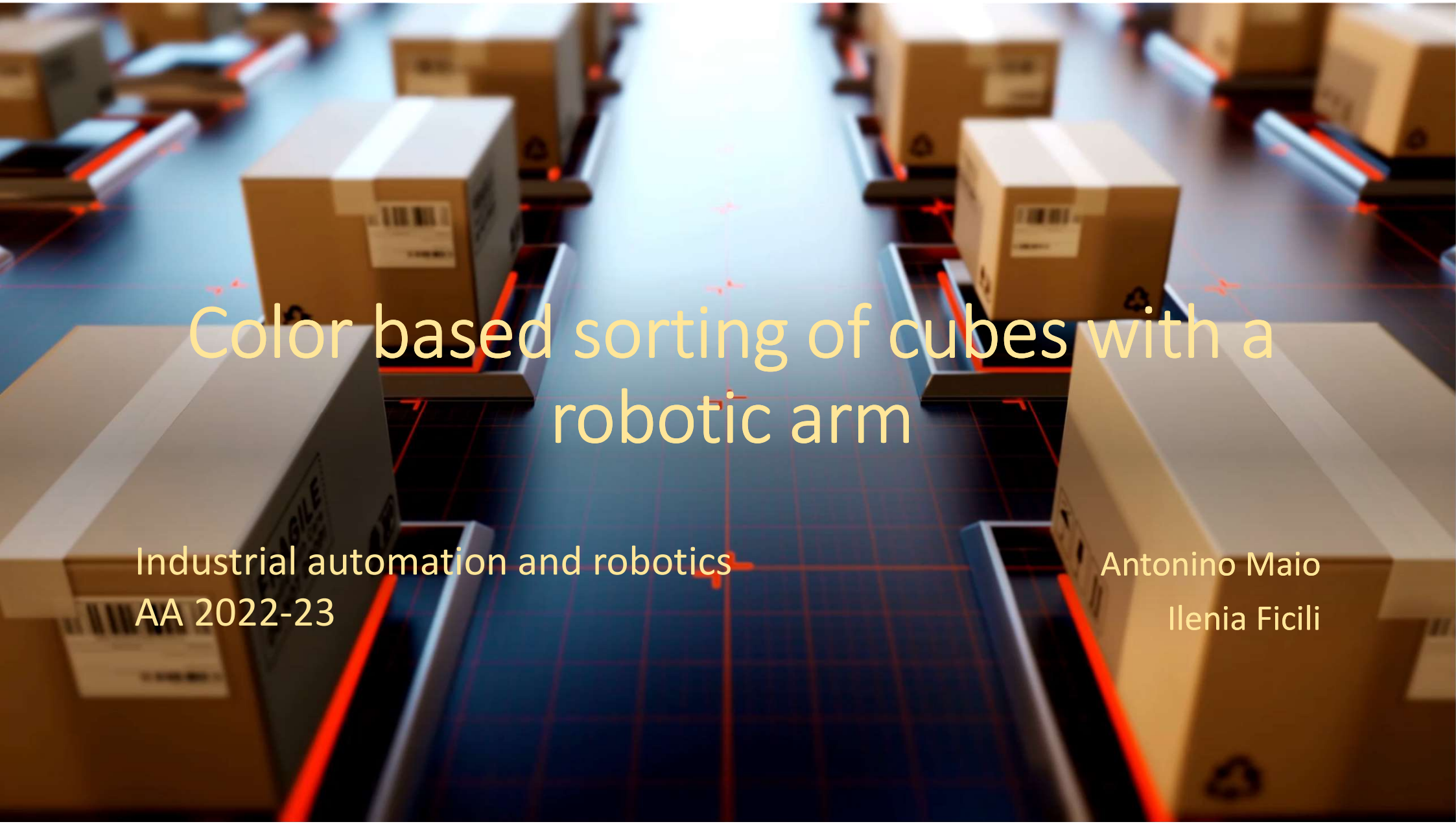


Color based sorting of cubes with a robotic arm

Industrial automation and robotics
AA 2022-23

Antonino Maio
Ilenia Ficili

IRB 140

- The industrial robot has 6 axis and it’s compact and powerful. The load capacity is 6 kg, and the reach reaches 810 mm. The robot can be mounted on the floor, wall or suspended and inclined at any angle. IRB 140 is easy to integrate and adapts to any environment
- Outstanding motion technology and excellent path accuracy, as well as being highly robust
- Multi purpose robot
- The end-effector used is the Robotiq 85 Gripper.

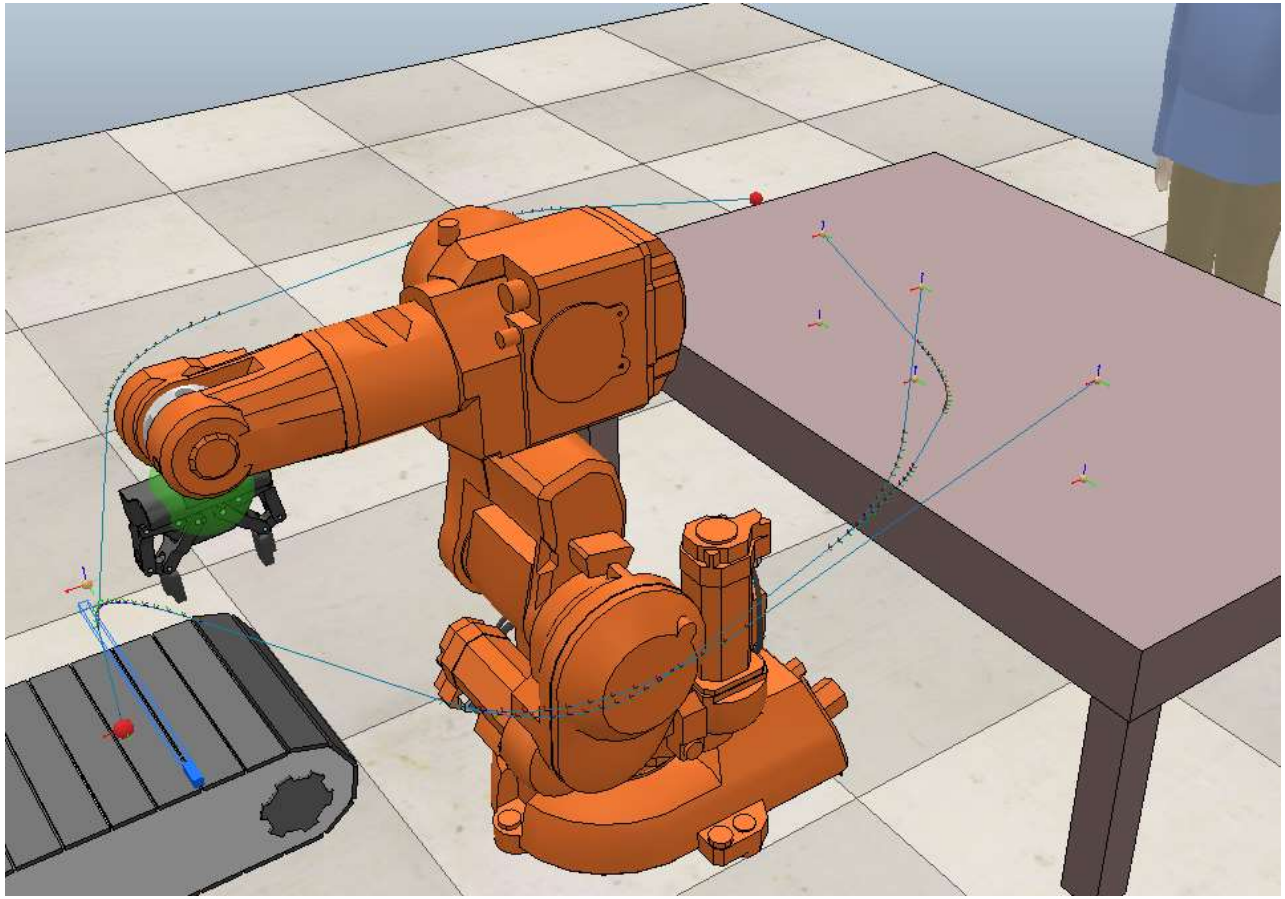
Goal of the project:

- In this project we want to use the robotic arm to pick up the boxes arriving in the conveyor belt and sort them by color (blue, red and green).
- Once done that, each time a box is in place, the human operator will be there to collect it and move on for further processing.
- The arm is being moved using inverse kinematis.
- Three paths are defined, one for each color, and thanks to the tecnique used all the joints are moved depending on the position of the end-effector.

The components used to create this scenario are:

- Conveyor belt
- Vision sensor
- IRB140

- In this image is possible to see a close up of the paths described to sort the object in 3 different position in the table



Generator Script

```
function sysCall_threadmain()

    gen = sim.getObjectHandle("Generazione")
    Position = sim.getObjectPosition(gen,-1)

    colors = { {0.9, 0.5, 0.5}, {0.5, 0.9, 0.5}, {0.5, 0.5, 0.9} }
    currentColor = colors[1]
    while(true) do
        currentColor = changeColor()
        cuboid = sim.createPureShape(0, 15, {0.05,0.05,0.05}, 0.2, nil)

        sim.setObjectInt32Parameter(cuboid, 3003, 0)
        sim.setObjectInt32Parameter(cuboid, 3004, 1)
        sim.setObjectSpecialProperty(cuboid, sim.objectspecialproperty_renderable)
        sim.setShapeColor(cuboid, nil, sim.colorcomponent_ambient, currentColor)
        sim.setObjectParent(cuboid, -1, true)
        sim.setObjectPosition(cuboid, -1, Position)
        sim.wait(10)
    end
end

function sysCall_cleanup()
end

function changeColor()
    randomNumber = sim.getRandom()
    if (randomNumber < 0.33) then
        return colors[1]
    elseif (randomNumber >= 0.33 and randomNumber <0.67) then
        return colors[2]
    else
    return colors[3]
    end
end
```

First of all, it is created the handle for the ‘dummy_generator’, which represent the present the position for the creation of the cuboids.
Then after defining the colors table, the main loop is initializated.
Here, a color between red blue and green is randonmly chosen by the ‘changeColor’ function, and then a cuboid shape-like with dimension (0.05, 0.05, 0.05) and mass 0.2 it’s created.

After this, all the parameters of this shape are set:

- Parameter 3003 corresponds to the property ‘static’ and it is set to false (0) which means that the shape is dynamic
- Parameter 3004 corresponds to the property ‘respondable’ and it is set to true (1)
- The special property ‘renderable’ is set to true in order to make the cuboid visible to the vision sensor.

Moreover, the color chosen at the beginning of the loop it’s assigned to the cuboid which is also made parentless and moved to the position of the dummy generator.

Conveyor Script

```
function sysCall_init()
    pathHandle=sim.getObjectHandle("ConveyorBeltPath")
    forwarder=sim.getObjectHandle('ConveyorBelt_forwarder')
    sim.setPathTargetNominalVelocity(pathHandle,0)
    visionSensor = sim.getObjectHandle("Vision_sensor")
end

function sysCall_actuation()
    imageBuffer = sim.getVisionSensorCharImage(visionSensor, 0, 0, 1, 1)
    red = tonumber(string.byte(imageBuffer, 1))
    green = tonumber(string.byte(imageBuffer, 2))
    blue = tonumber(string.byte(imageBuffer, 3))
    local beltVelocity = nil
    if (red ~= 0 and green ~=0 and blue ~= 0) then
        beltVelocity = 0
    else
        beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")
    end

    local dt=sim.getSimulationTimeStep()
    local pos=sim.getPathPosition(pathHandle)
    pos=pos+beltVelocity*dt
    sim.setPathPosition(pathHandle,pos)

    local relativeLinearVelocity={beltVelocity,0,0}
    sim.resetDynamicObject(forwarder)
    local m=sim.getObjectMatrix(forwarder,-1)
    m[4]=0
    m[8]=0
    m[12]=0
    local absoluteLinearVelocity=sim.multiplyVector(m,relativeLinearVelocity)
    sim.setObjectFloatParameter(forwarder,sim.shapefloatparam_init_velocity_x,absoluteLinearVelocity[1])
    sim.setObjectFloatParameter(forwarder,sim.shapefloatparam_init_velocity_y,absoluteLinearVelocity[2])
    sim.setObjectFloatParameter(forwarder,sim.shapefloatparam_init_velocity_z,absoluteLinearVelocity[3])
end
```

In the sysCall_actuation function the image that was read from the vision sensor is saved in the imageBuffer variable and the RGB components are separated and processed.

With the if – else construct the conveyor is stopped if the vision sensor detects an object, otherwise its velocity is steady.

IRB 140 Script

```
changeTarget(idleTarget, idleOrientation)
    while (true) do
    imageBuffer = sim.getVisionSensorCharImage(visionSensor, 0, 0, 1, 1)
    red = tonumber(string.byte(imageBuffer, 1))
    green = tonumber(string.byte(imageBuffer, 2))
    blue = tonumber(string.byte(imageBuffer, 3))
    if (red~= 0 and green~=0 and blue~=0) then
        targetPosition = {}
        targetOrientation = {}
        targetPathPosition = {}
        targetPathOrientation = {}
        if ( red > green and red> blue) then
            Andata = Rosso
            i = 1
            updatePosition(i)
            targetPosition[1] = targetPosition[1] - (0.1* getLength(redProducts))
            targetPathPosition[1] = targetPathPosition[1] - (0.1 * getLength(redProducts))
            cuboidColor = "red"
        elseif ( green > red and green > blue) then
            Andata = Verde
            i = 2
            updatePosition(i)
            targetPosition[1] = targetPosition[1] - (0.1* getLength(greenProducts))
            targetPathPosition[1] = targetPathPosition[1] - (0.1 * getLength(greenProducts))
            cuboidColor = "green"
        elseif ( blue > red and blue > green ) then
            Andata = Blu
            i = 3
            updatePosition(i)
            targetPosition[1] = targetPosition[1] - (0.1* getLength(blueProducts))
            targetPathPosition[1] = targetPathPosition[1] - (0.1 * getLength(blueProducts))
            cuboidColor = "blue"
        end
```

```
            changeTarget(grabTarget, grabOrientation)
            sim.wait(1)
            grabCuboid(cuboidColor)
            sim.wait(0.5)
            sim.followPath(target, Andata, changePositionOnly, 0, 0.7,15)
            sim.wait(1)
            changeTarget(targetPosition, targetOrientation)
            sim.wait(1)
            dropCuboid()
            sim.wait(1)
            changeTarget(targetPathPosition, targetPathOrientation)
            sim.wait(0.5)
            sim.followPath(target, Ritorno, changePositionOnly, 0, 0.7,15)
            sim.wait(1)
            clearTables()
        end
    end
end
```

The image from the vision sensor is acquired again, and processed. If an object is detected the if-else construct will check the color and update the path with corresponding one, and the finals positions for the cuboid.

At the end a simple sequence of istruction is performed in order to make the manipulator follow the round trip path and grab and drop the cuboid at the right time and position.

```
function grabCuboid (color)

    index = 0
    while (true) do
        objectInScene = sim.getObjects(index, sim.object_shape_type)
        if (objectInScene == -1) then break end
        objectName = sim.getObjectName(objectInScene)
        isCuboid = "Cuboid" == string.sub(objectName, 1, 6)
        if ((isCuboid) and
                (sim.getObjectInt32Parameter(objectInScene, sim.shapeintparam_respondable) ~= 0) and
                (sim.checkProximitySensor(gripperSensor, objectInScene) == 1 )  ) then
                attachedObject = objectInScene
                sim.setObjectParent(objectInScene, connector, true)
                if ( color == "red" ) then
                    table.insert(redProducts, objectInScene)
                elseif ( color == "green" ) then
                    table.insert(greenProducts, objectInScene)
                else
                    table.insert(blueProducts, objectInScene)
                end
                break
        end
    index = index + 1
    end
end

function dropCuboid()
    sim.setObjectParent(objectInScene, -1, true)
end
```

The grabcuboid function gets the handle of the object that will be grabbed by the manipulator, and then checks if the frist 6 characters of the name are equal to ‘cuboid’.

If this condition is true and the object is respondable and the proximity sensor of the gripper detects it, the object is attached to the gripper making it its parent.

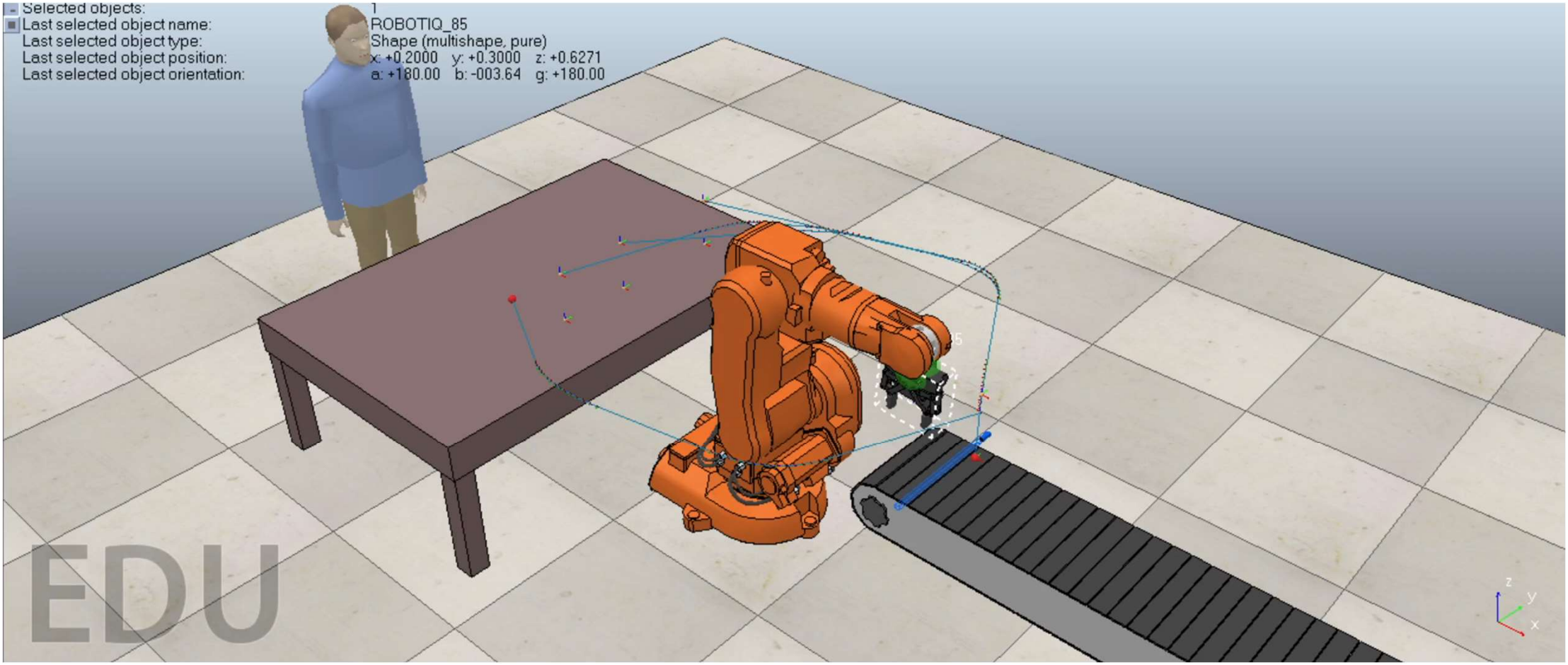
Then the color of the object is checked and it is insert in the relative table.

The dropCuboid function simply makes the object parentless, which means that it is not connected anymore to the gripper.

```
function getLength(t)
    count = 0
    for index,values in pairs(t) do
        count = count + 1
    end
    return count
end

function clearTables()

    redCount = getLength(redProducts)
    greenCount = getLength(greenProducts)
    blueCount = getLength(blueProducts)
    if (redCount >= 2) then
        for index, value in pairs(redProducts) do
            sim.removeObject(redProducts[index])
        end
        redProducts = {}
    end
    if (greenCount >= 2) then
        for index, value in pairs(greenProducts) do
            sim.removeObject(greenProducts[index])
        end
        greenProducts = {}
    end

    if (blueCount >= 2) then
        for index, value in pairs(blueProducts) do
            sim.removeObject(blueProducts[index])
        end
        blueProducts = {}
    end
end
```

The function getLength counts how many objects of a determinated color are present in the scene (on the table).

The function clearTables get the length of each color and analyzing each row of colors, it checks if the number of cuboids is equal to 2, if so the cuboids are deleted because collected from the human operator.



In this video it is possible to see the whole system functioning.

Bibliografia:

- ABB IRB 140 datasheet: https://library.e.abb.com/public/a7121292272d40a9992a50745fdaa3b2/3HAC041346%20PS%20IRB%20140-en.pdf
- CoppeliaSim Robotics User Manual https://www.coppeliarobotics.com/helpFiles/