DIFFERENTIAL DRIVE MOBILE ROBOT (DDMR)

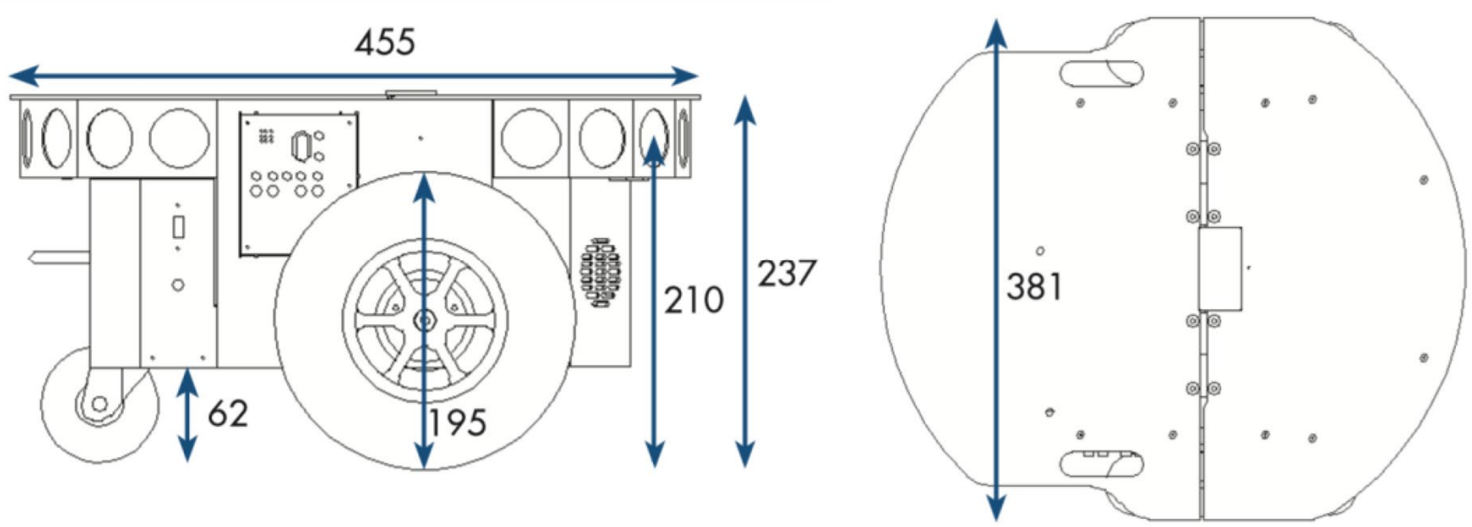
Djoko Purwanto
Dept. of Electrical Engineering, ITS
djoko@its.ac.id

INTRODUCTION

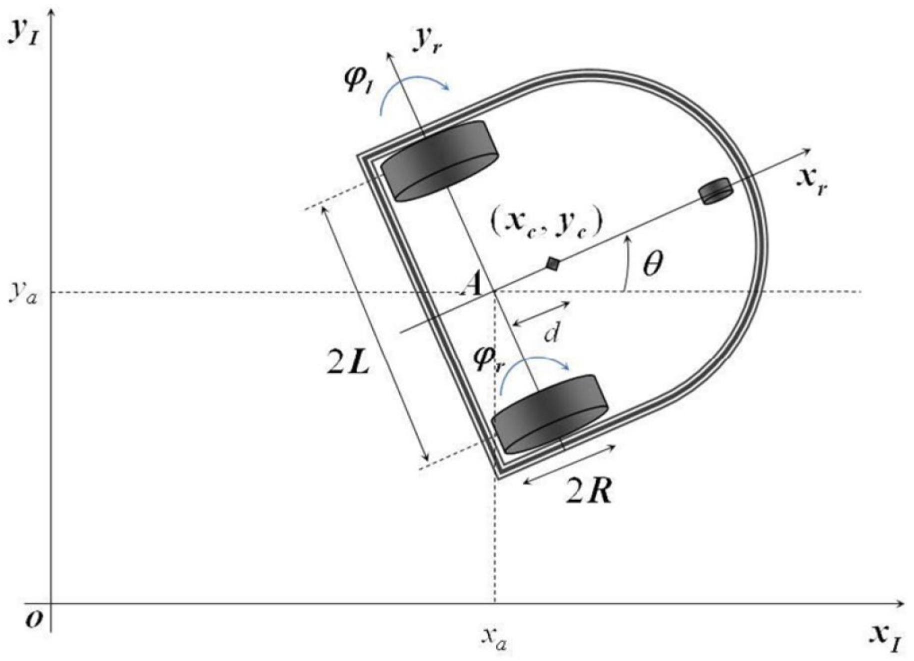
Differential drive is a method of controlling a mobile robot motion with two motorized wheels. The Pioneer P3DX is a Differential Drive Mobile Robot (DDMR) type in CoppeliaSim.

Pioneer P3DX

Dimension (in mm)



KINEMATICS



$$\dot{q}^I = \begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos\theta & \frac{R}{2}\cos\theta \\ \frac{R}{2}\sin\theta & \frac{R}{2}\sin\theta \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \quad (1)$$

$$\dot{q}^I = \begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (2)$$

Forward Kinematics

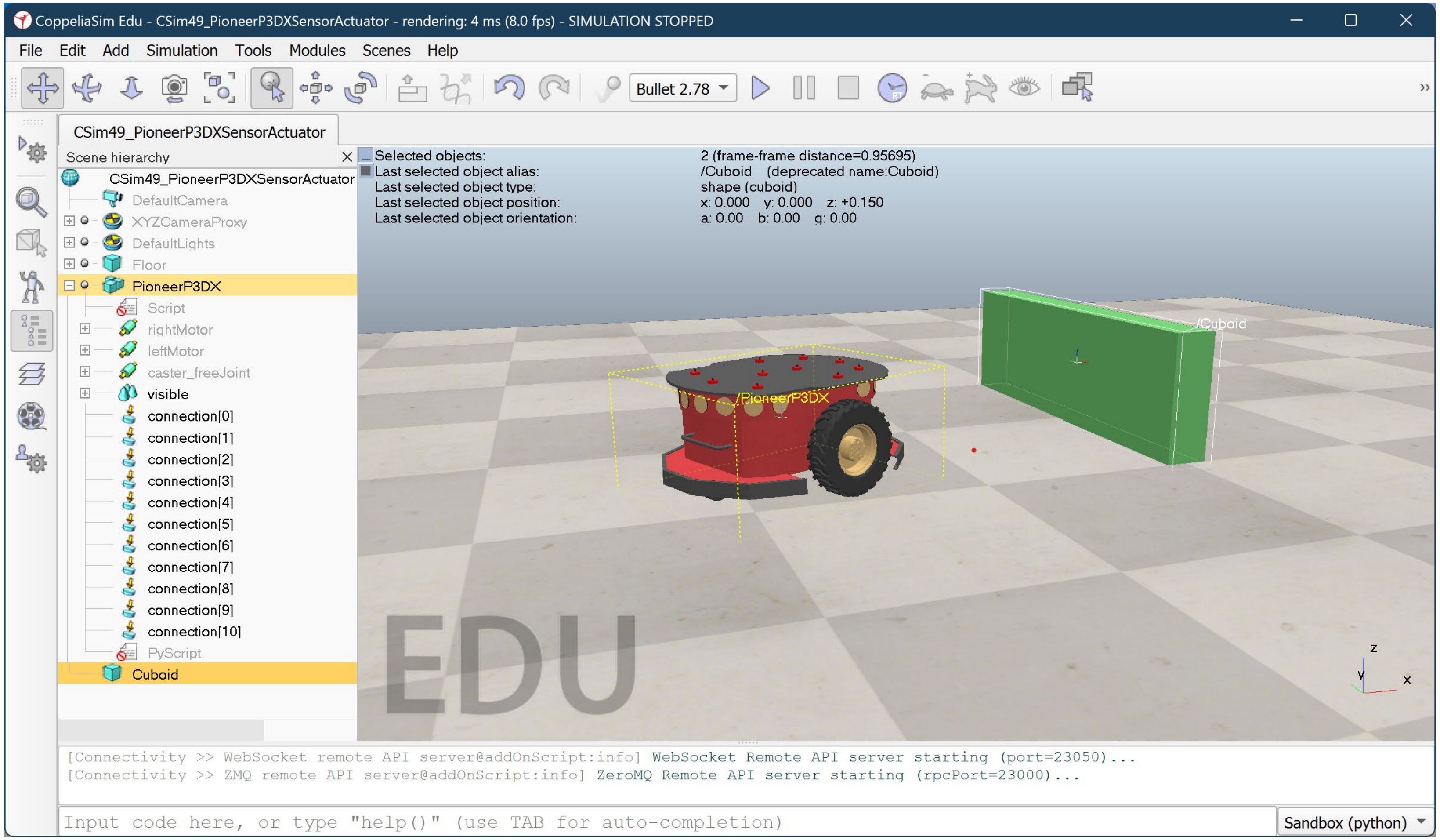
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \quad (3)$$

Inverse Kinematics

$$\begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix}^{-1} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (4)$$

ROBOT SENSORS AND MOTORS

Simulation Environment



Writing Code

	Embedded script	Add-on / sandbox script	Plugin	Client application	Remote API client	ROS / ROS2 node	ZeroMQ node
Control entity is external (i.e. can be located on a robot, different machine, etc.)	No	No	No	No	Yes	Yes	Yes
Supported programming language	Lua, Python	Lua, Python	C/C++	C/C++, Python	C/C++, Python, Java, JavaScript, Matlab, Octave	Any [1]	Any
Code execution speed	Relatively fast [2]	Relatively fast [2]	Fast	Fast	Depends on programming language	Depends on programming language	Depends on programming language
Communication lag	None [3]	None [3]	None	None	Yes	Yes	Yes
Communication channel	Python: ZeroMQ [3]	Python: ZeroMQ [3]	None	None	ZeroMQ or WebSockets	ROS / ROS2	ZeroMQ
Control entity can be fully contained in a scene or model, and is highly portable	Yes	No	No	No	No	No	No
Stepped operation [4]	Yes, inherent	Yes, inherent	Yes, inherent	Yes, inherent	Yes	Yes	Yes
Non-stepped operation [4]	Yes, via threads	Yes, via threads	No (threads available, but API access forbidden)	No (threads available, but API access forbidden)	Yes	Yes	Yes

[1] Depends on ROS / ROS2 bindings

[2] Depends on the programming language, but the execution of API functions is very fast

[3] Lua scripts are executed in CoppeliaSim's main thread, Python scripts are executed in separate processes

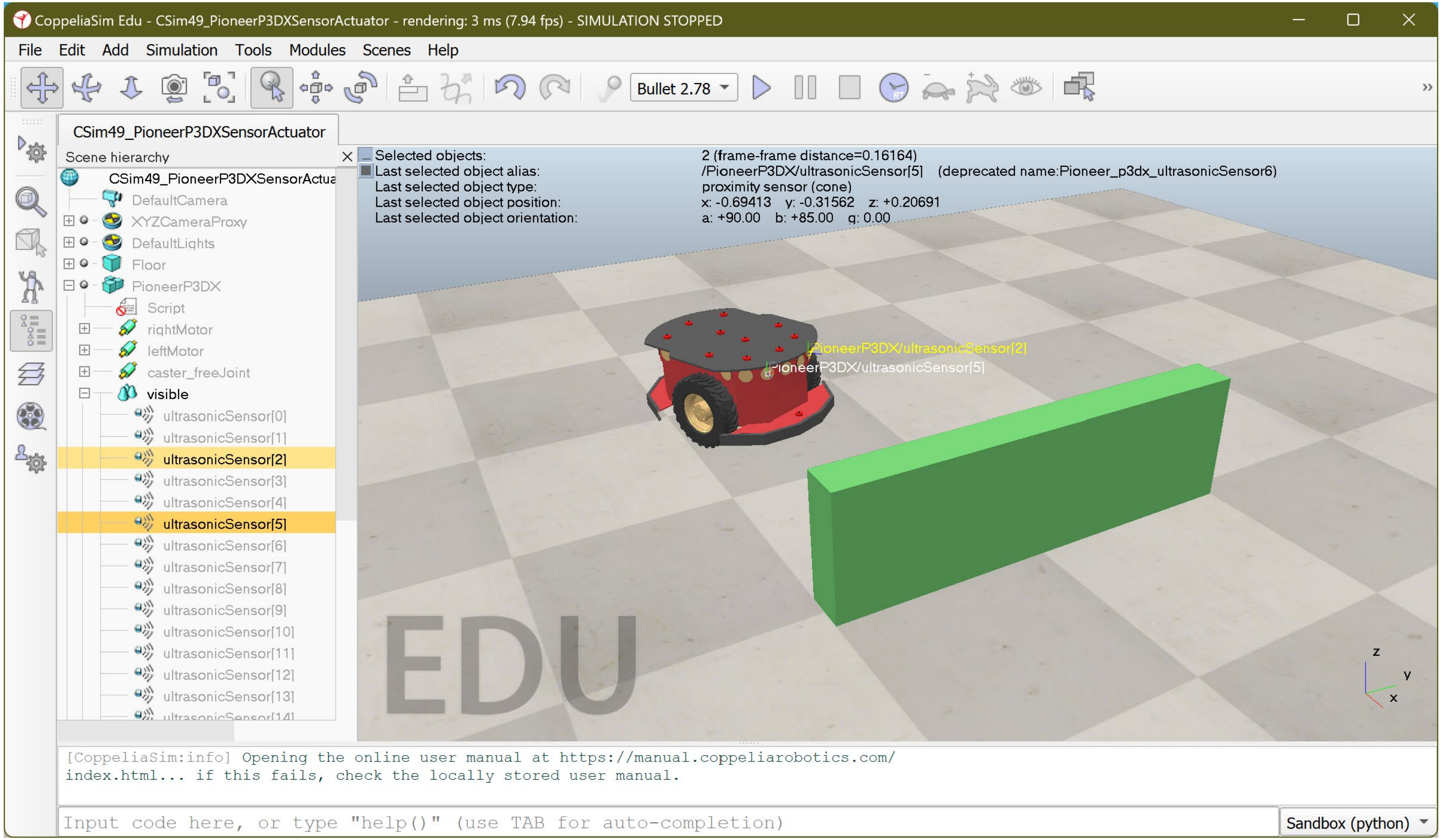
[4] *Stepped* as in *synchronized* with each simulation step

Python Programming for Accessing Robot Sensors and Motors

Initialization

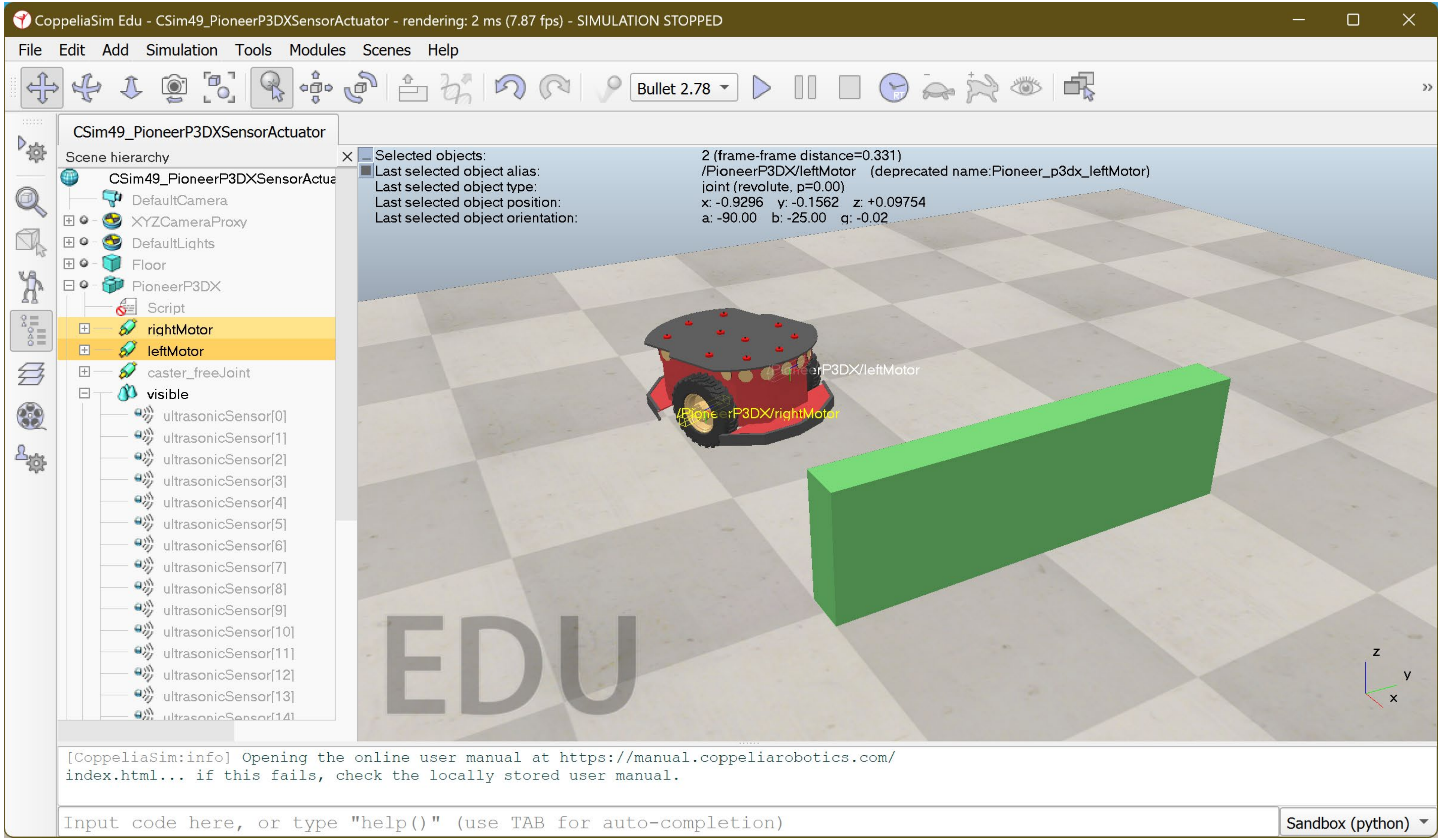
```python
############################################################
# Pioneer P3DX - Mobile Robot Sensors and Actuators
# (c)2025 Djoko Purwanto, djoko@its.ac.id
############################################################

from coppeliasim_zmqremoteapi_client import RemoteAPIClient
import keyboard
import numpy as np

# === Function Definition ===
# ---------------------------

```

Retrieve the Data from the Sensors



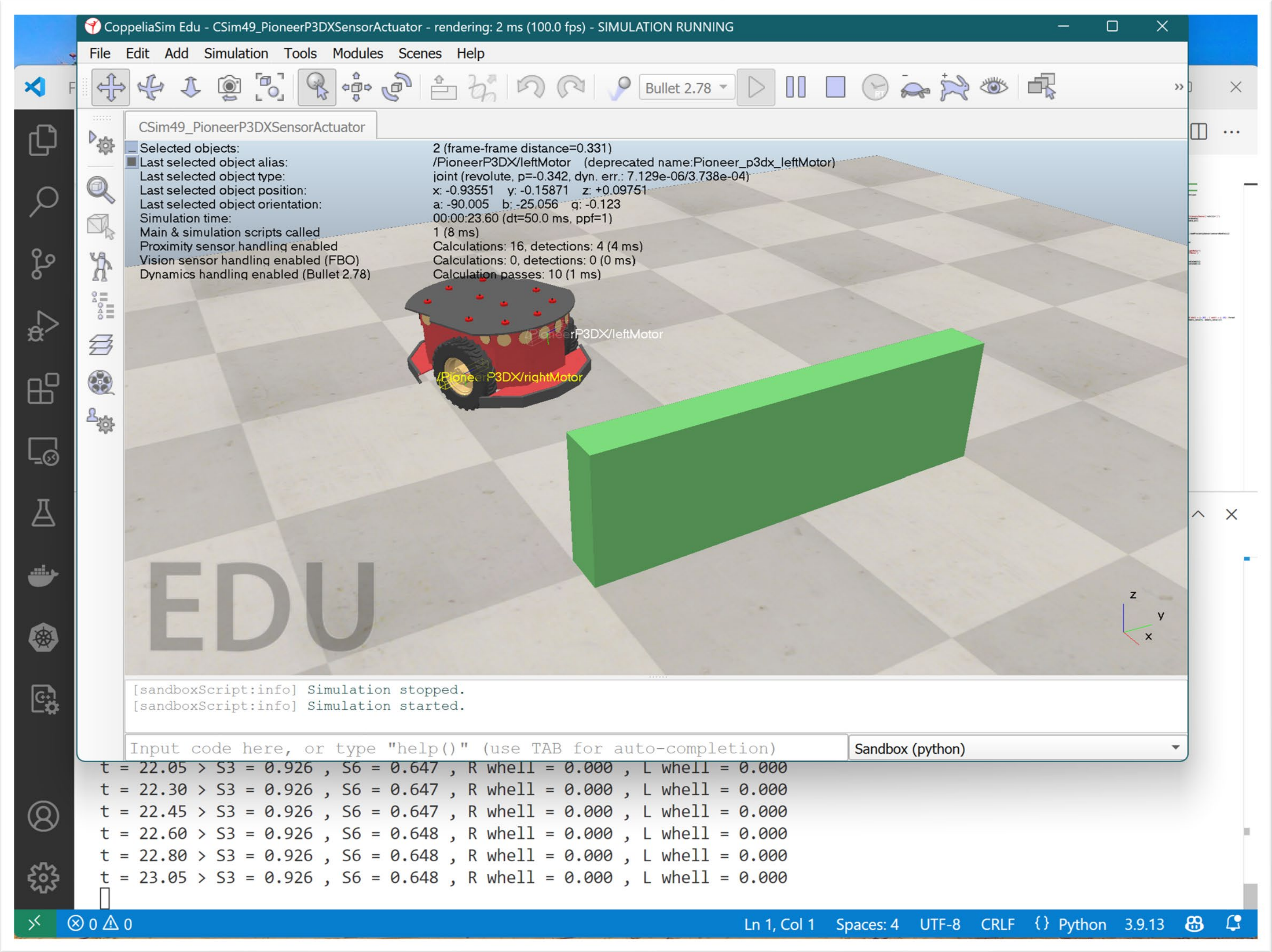
```python
def getSensorsHandle():
    sensorsHandle = np.array([])
    for i in range(16):
        sensorHandle = sim.getObject('/PioneerP3DX/ultrasonicSensor['+str(i)+']')
        sensorsHandle = np.append(sensorsHandle,sensorHandle)
    _, _, _, _, _ = sim.handleProximitySensor(sim.handle_all)
    return sensorsHandle

def getDistances(sensorsHandle):
    Distances = np.array([])
    for i in range(16):
        detectionState, _, detectedPoint, _, _ = sim.readProximitySensor(sensorsHandle[i])
        distanceValue = detectedPoint[2]
        if detectionState == False:
            distanceValue = 2.0
        Distances = np.append(Distances,distanceValue)
    return Distances

```

Set the Motors Velocities



```python
def getMotorsHandle():
    motorRightHandle = sim.getObject('/PioneerP3DX/rightMotor')
    motorLeftHandle = sim.getObject('/PioneerP3DX/leftMotor')
    return (motorRightHandle, motorLeftHandle)

def setRobotMotion( motorsHandle, veloCmd):
    _ = sim.setJointTargetVelocity(motorsHandle[0], veloCmd[0])
    _ = sim.setJointTargetVelocity(motorsHandle[1], veloCmd[1])
    return

```

Do the Simulation



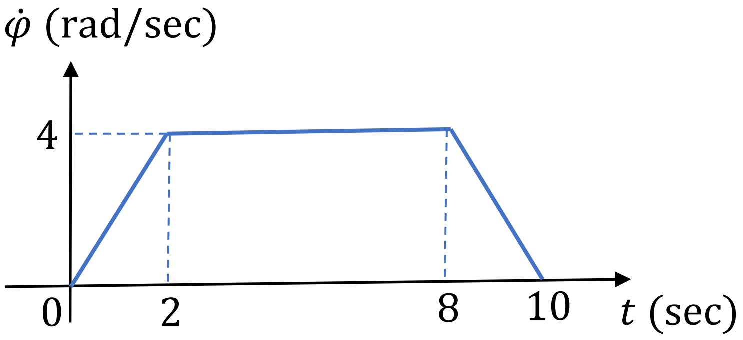
```python
# === Main Program ===
# --------------------

print('Program started')
# --- Connection to Coppelia Simulator
client = RemoteAPIClient()
sim = client.require('sim')
sim.setStepping(False)
sim.startSimulation()
# --- Object Handle
sensors_handle = getSensorsHandle()
motors_handle = getMotorsHandle()
# --- Simulation
wheels_velo = [0.0, 0.0]
time_start = sim.getSimulationTime()
```

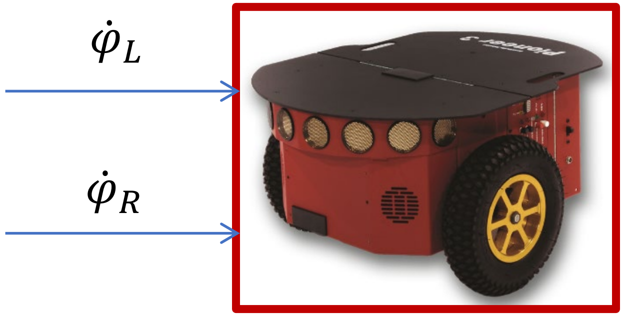
```python
while (True):
    t_now = sim.getSimulationTime()-time_start
    # Get sensors data
    obj_distances = getDistances(sensors_handle)
    # Set velocity command for robot motion
    setRobotMotion(motors_handle,wheels_velo)
    # Print the information (if necessary)
    print('t = {:.2f} > S3 = {:.3f} , S6 = {:.3f} , R whell = {:.3f} , L whell = {:.3f}'.format
        (t_now,obj_distances[2], obj_distances[5], wheels_velo[0], wheels_velo[1]))
    if keyboard.is_pressed('esc'): break
# --- Simulation Finished
sim.stopSimulation()
print('\nProgram ended\n')
```

BASIC MOTION

Trapezoidal Motion

Trapezoidal motion is a motion profile for the angular velocity command applied to the DDMR wheels to generate the smooth motion. The motion command includes acceleration, constant velocity, and deceleration.

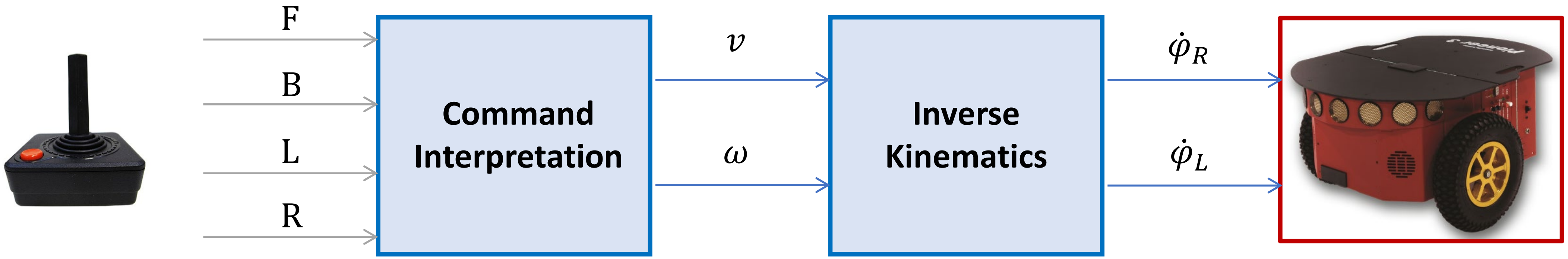




MOTION CONTROL

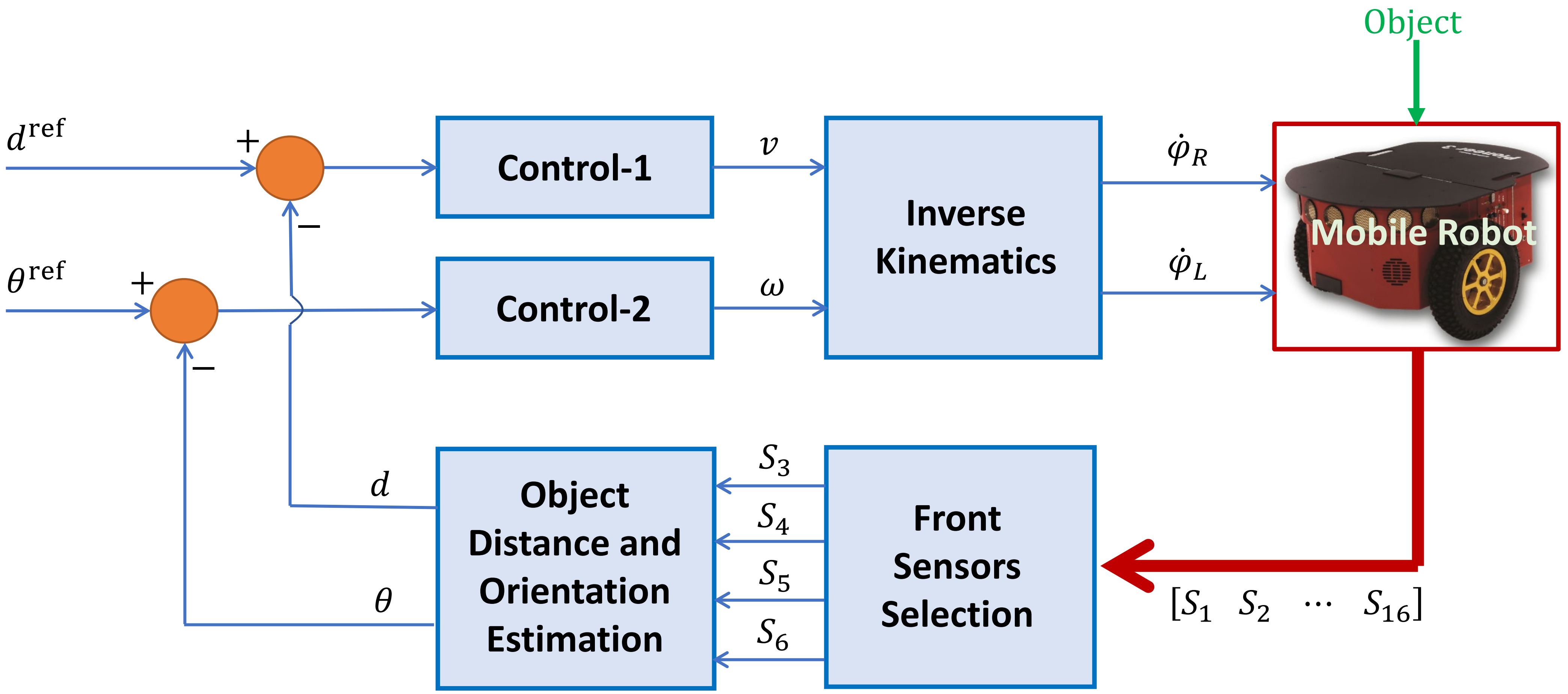
Manual Control

The DDMR motion can be controlled manually using the keyboard or joystick. The robot's motion commands from the keyboard or joystick include Forward(F), Backward (B), Left (L), and Right (R).



Automatic Control

The automatic control principle can be applied to the DDMR to perform specific task. Below is the block diagram of DDMR system to perform the object follower task.



ASSIGNMENT 2

1. Open the CoppeliaSim, pick and place the Pioneer P3DX mobile robot, place a simple object at the front of robot (see the figure). Rewrite and run the program to access the robot sensors and motors (see page 6-13)
2. Create the python programming to control the robot motion manually using keyboard for soccer robot (see page 15 for reference)

THANK YOU