DIFFERENTIAL DRIVE MOBILE ROBOT (DDMR)

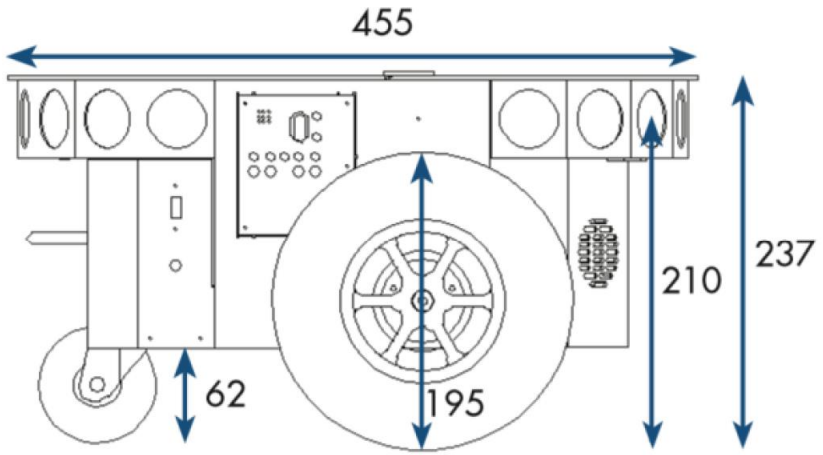
Djoko Purwanto
Dept. of Electrical Engineering, ITS
djoko@ee.its.ac.id

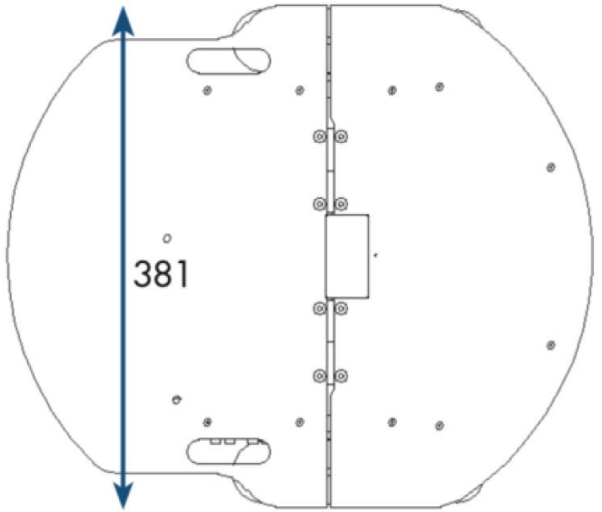
Introduction

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

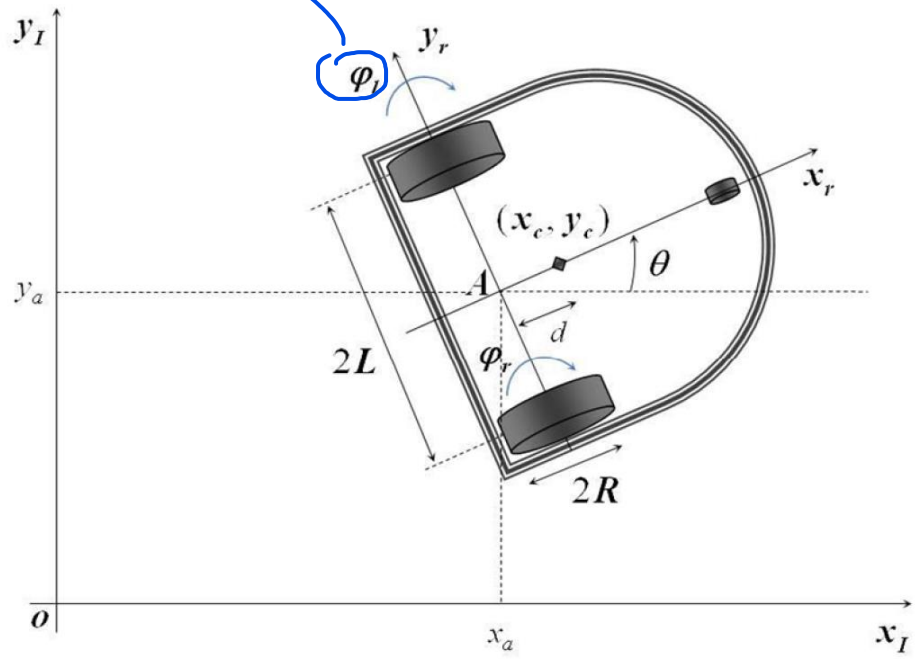
Dimension (in mm)





Kinematics

Percepatan Sudut Roda/ Angular Velocity



Ini mempresentasikan apa ya? Posisi?

vector kecepatan robot

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

misal $\theta = 0°$

$$\dot{q}^I = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ 0 & 0 \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix}$$

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

$\dot{x} = \frac{R}{2}\dot{\varphi}_R + \frac{R}{2}\dot{\varphi}_L$

$= R\dot{\varphi}_R$

$= v$

Forward Kinematics

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

Inverse Kinematics

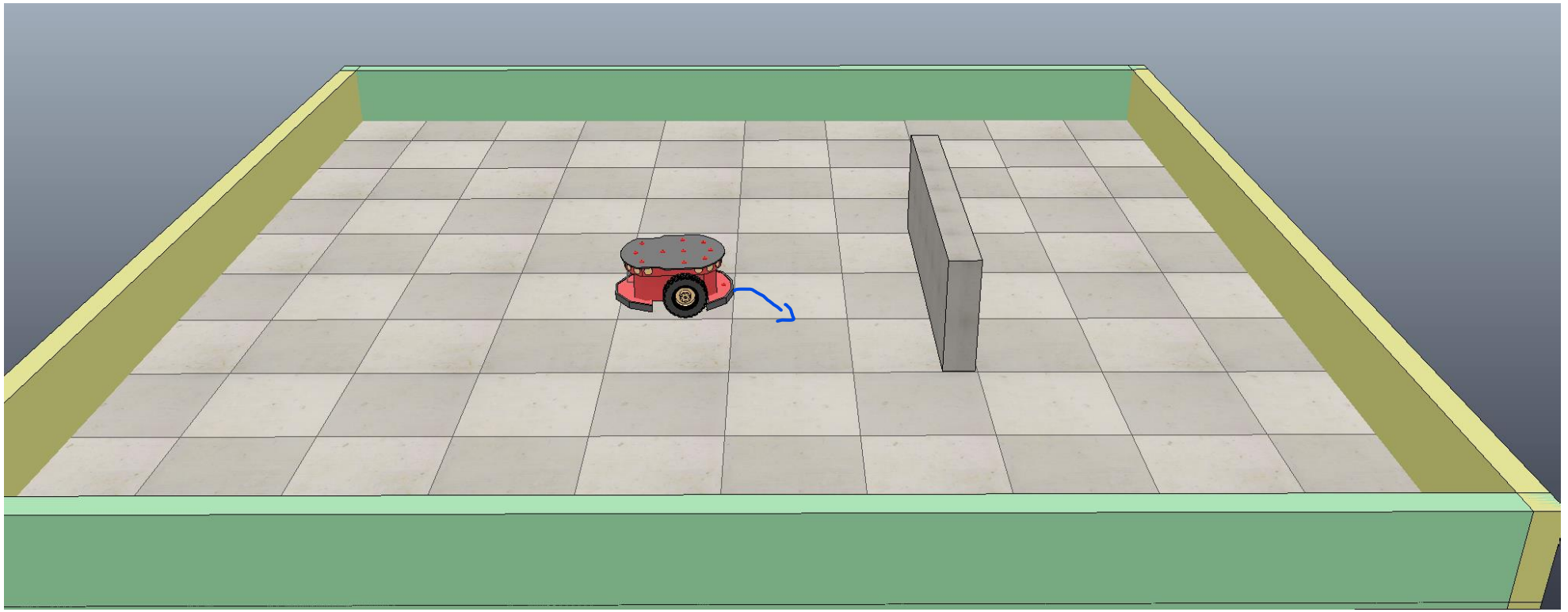
$$\begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix}^{-1} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

DDMR Sensor and Actuator

$v = 5$

$\omega = 0,5$

Simulation Environment



Python Programming for Accessing DDMR Sensor and Actuator

```
1  ############################################################
2  # Pioneer P3DX - Robot Sensors and Motors Test
3  # (c)2022 Djoko Purwanto, djoko@ee.its.ac.id
4  ############################################################
5  import sim
6  import sys
7  import time
8  import keyboard
9  import numpy as np
10
11 # === Function Definition ===
12 # --------------------------
13 def connectSimulator():
14     # Close all opened connections
15     sim.simxFinish(-1)
16     # Connect to CoppeliaSim
17     clientID = sim.simxStart('127.0.0.1', 19997, True, True, 5000, 5)
18     if clientID != -1: print('Connected to remote API server.')
19     else:
20         print('Connection unsuccesful, program ended.')
21         sys.exit()
22     return clientID
23
```

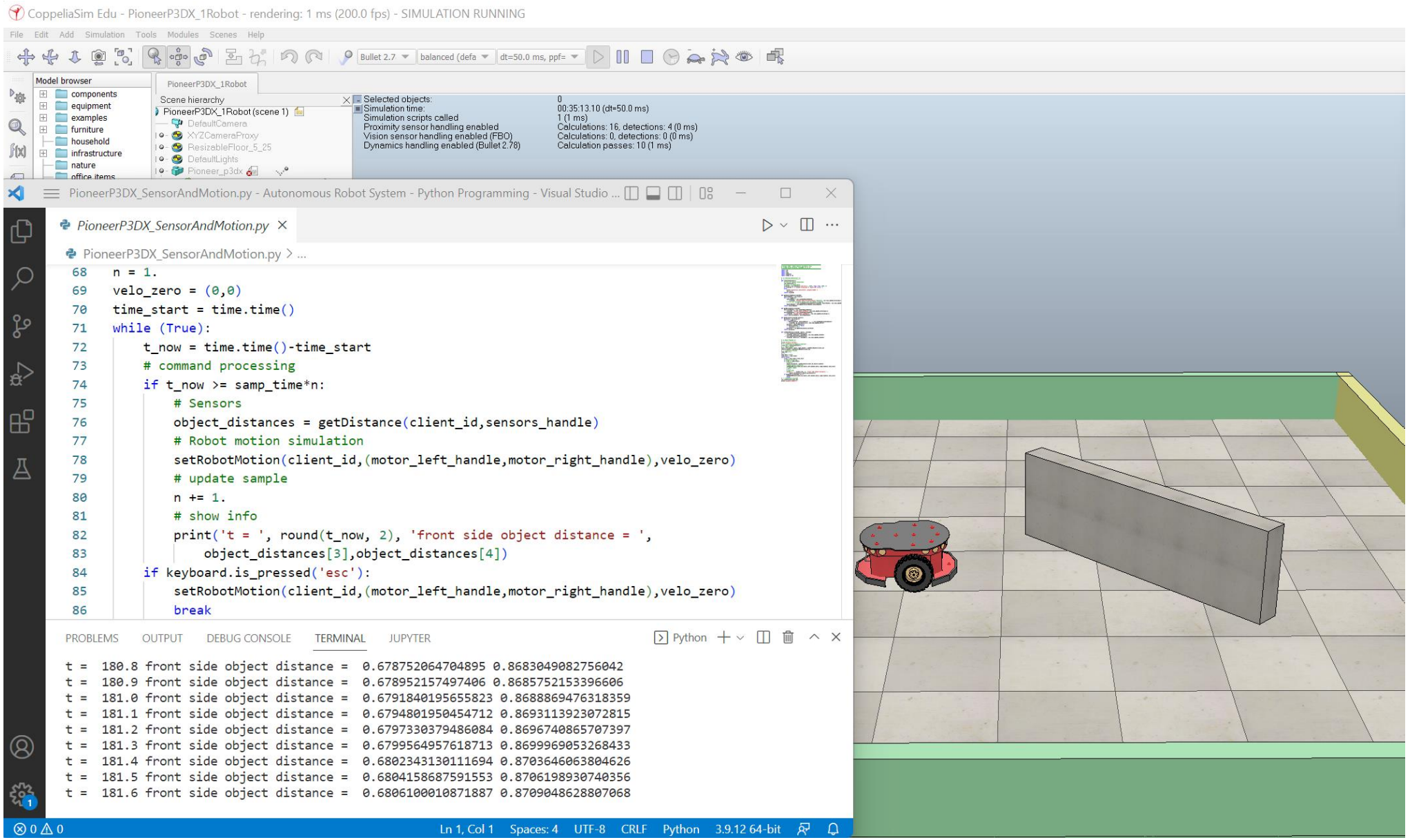
```python
def getSensorsHandle(clientID):
    sensorsHandle = np.array([])
    for i in range(16):
        sensorHandle = sim.simxGetObjectHandle(
            clientID, 'Pioneer_p3dx_ultrasonicSensor'+str(i+1), sim.simx_opmode_blocking)[1]
        # First call proximity sensor must use opmode_streaming
        _, _, _, _, _ = sim.simxReadProximitySensor(clientID, sensorHandle, sim.simx_opmode_streaming)
        sensorsHandle = np.append(sensorsHandle,sensorHandle)
    return sensorsHandle

def getMotorHandle(clientID):
    motorLeftHandle = sim.simxGetObjectHandle(
        clientID, 'Pioneer_p3dx_leftMotor',sim.simx_opmode_blocking)[1]
    motorRightHandle = sim.simxGetObjectHandle(
        clientID, 'Pioneer_p3dx_rightMotor', sim.simx_opmode_blocking)[1]
    return motorLeftHandle, motorRightHandle

def getDistance(clientID,sensors):
    distances = np.array([])
    for i in range(16):
        _, detectionState, detectedPoint, _, _ = sim.simxReadProximitySensor(
            clientID, np.int(sensors[i]), sim.simx_opmode_buffer)
        distance = detectedPoint[2]
        if detectionState == False:
            distance = 10.0
        distances = np.append(distances,distance)
    return distances

def setRobotMotion(clientID, motors, veloCmd):
    _ = sim.simxSetJointTargetVelocity(
        clientID, motors[0], veloCmd[0], sim.simx_opmode_oneshot)
    _ = sim.simxSetJointTargetVelocity(
        clientID, motors[1], veloCmd[1], sim.simx_opmode_oneshot)
```

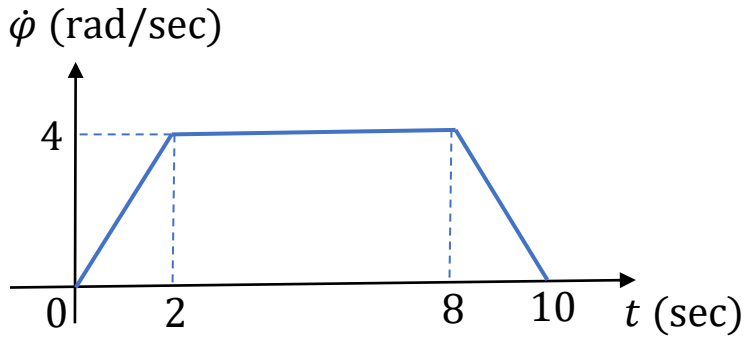
```
57
58  # === Main Program ===
59  # --------------------
60  print('Program started')
61  # --- Connection to Coppelia Simulator
62  client_id = connectSimulator()
63  # --- Object Handle
64  motor_left_handle, motor_right_handle = getMotorHandle(client_id)
65  sensors_handle = getSensorsHandle(client_id)
66  # --- Simulation Process
67  samp_time = 0.1
68  n = 1.
69  velo_zero = (0,0)
70  time_start = time.time()
71  while (True):
72      t_now = time.time()-time_start
73      # command processing
74      if t_now >= samp_time*n:
75          # Sensors
76          object_distances = getDistance(client_id,sensors_handle)
77          # Robot motion simulation
78          setRobotMotion(client_id,(motor_left_handle,motor_right_handle),velo_zero)
79          # update sample
80          n += 1.
81          # show info
82          print('t = ', round(t_now, 2), 'front side object distance = ',
83              object_distances[3],object_distances[4])
84      if keyboard.is_pressed('esc'):
85          setRobotMotion(client_id,(motor_left_handle,motor_right_handle),velo_zero)
86          break
87  # --- Simulation Finished
88  sim.simxFinish(client_id)
89  print('program ended\n')
```

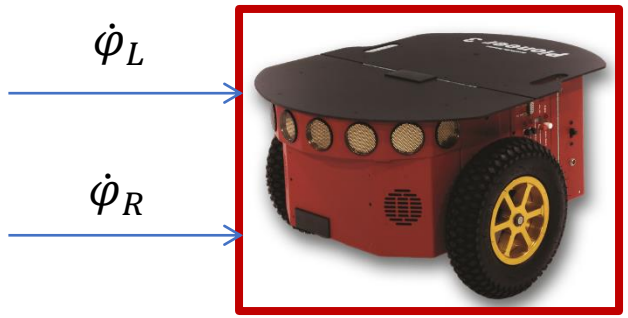

CoppeliaSim Edu - PioneerP3DX_1Robot - rendering: 1 ms (200.0 fps) - SIMULATION RUNNING
File Edit Add Simulation Tools Modules Scenes Help
Model browser
components
equipment
examples
furniture
household
infrastructure
nature
office items
PioneerP3DX_1Robot
Scene hierarchy
PioneerP3DX_1Robot (scene 1)
DefaultCamera
XYZCameraProxy
ResizableFloor_5_25
DefaultLights
Pioneer_p3dx
Selected objects: 0
Simulation time: 00:35:13.10 (dt=50.0 ms)
Simulation scripts called 1 (1 ms)
Proximity sensor handling enabled Calculations: 16, detections: 4 (0 ms)
Vision sensor handling enabled (FBO) Calculations: 0, detections: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78) Calculation passes: 10 (1 ms)
PioneerP3DX_SensorAndMotion.py - Autonomous Robot System - Python Programming - Visual Studio ...
PioneerP3DX_SensorAndMotion.py
PioneerP3DX_SensorAndMotion.py > ...
68 n = 1.
69 velo_zero = (0,0)
70 time_start = time.time()
71 while (True):
72 t_now = time.time()-time_start
73 # command processing
74 if t_now >= samp_time*n:
75 # Sensors
76 object_distances = getDistance(client_id,sensors_handle)
77 # Robot motion simulation
78 setRobotMotion(client_id,(motor_left_handle,motor_right_handle),velo_zero)
79 # update sample
80 n += 1.
81 # show info
82 print('t = ', round(t_now, 2), 'front side object distance = ',
83 object_distances[3],object_distances[4])
84 if keyboard.is_pressed('esc'):
85 setRobotMotion(client_id,(motor_left_handle,motor_right_handle),velo_zero)
86 break
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Python
t = 180.8 front side object distance = 0.678752064704895 0.8683049082756042
t = 180.9 front side object distance = 0.678952157497406 0.8685752153396606
t = 181.0 front side object distance = 0.6791840195655823 0.8688869476318359
t = 181.1 front side object distance = 0.6794801950454712 0.8693113923072815
t = 181.2 front side object distance = 0.6797330379486084 0.8696740865707397
t = 181.3 front side object distance = 0.6799564957618713 0.8699969053268433
t = 181.4 front side object distance = 0.6802343130111694 0.8703646063804626
t = 181.5 front side object distance = 0.6804158687591553 0.8706198930740356
t = 181.6 front side object distance = 0.6806100010871887 0.8709048628807068
0 0
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.12 64-bit

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.

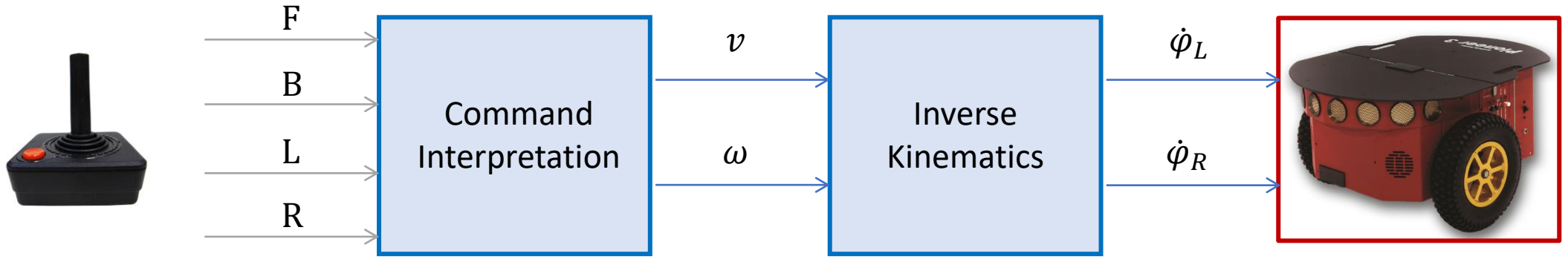




Motion Control

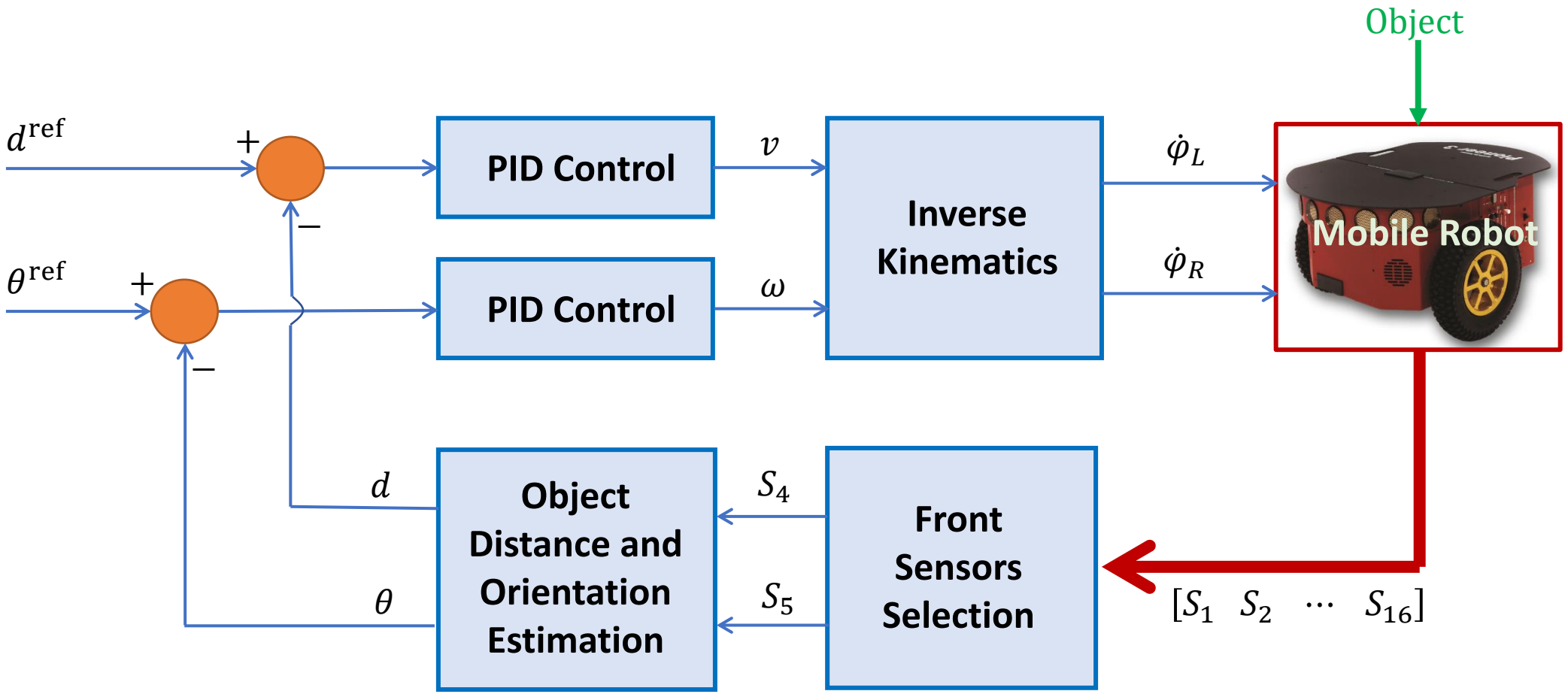
Manual Control

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

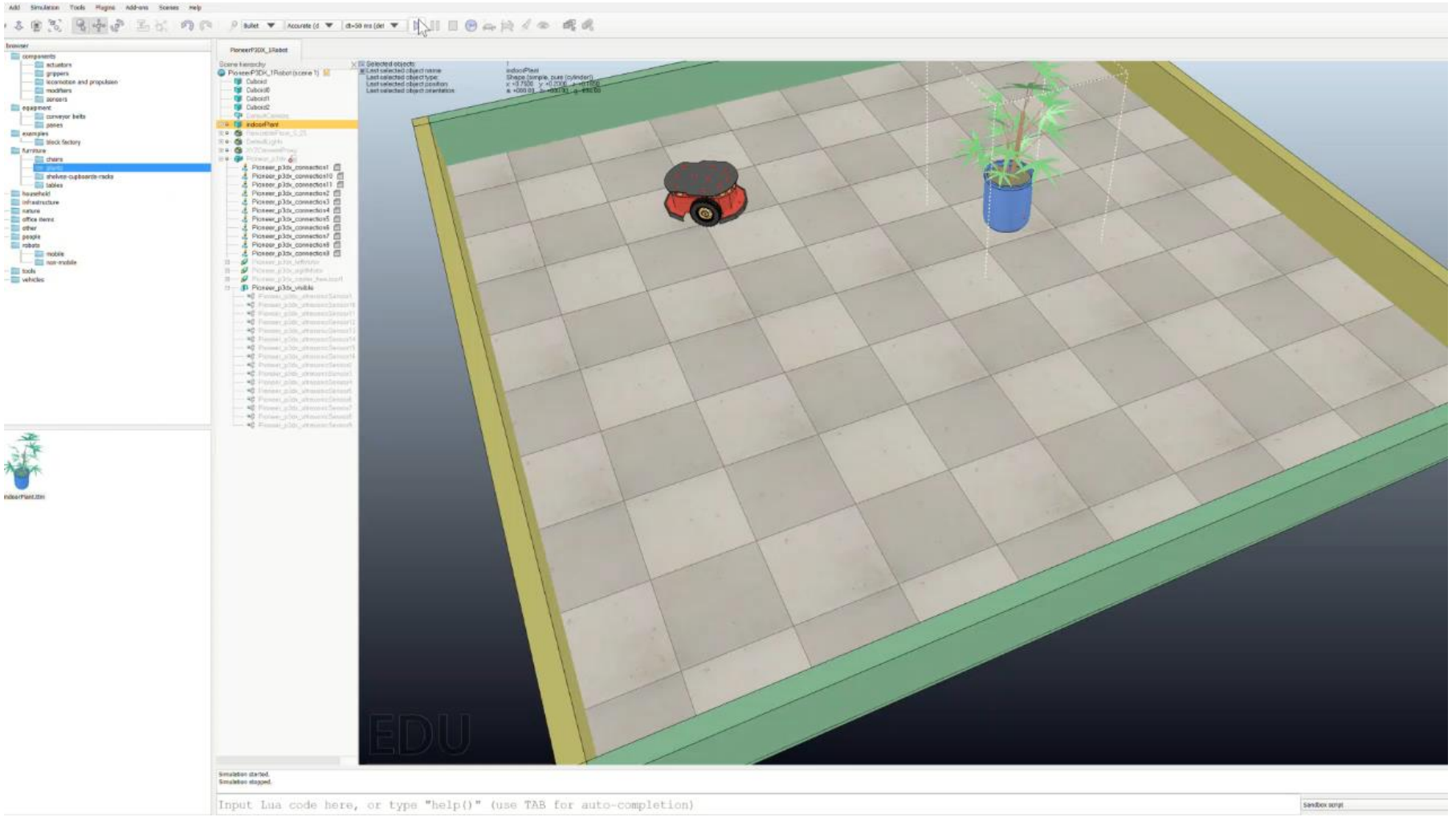


Automatic Control

The automatic control principle (PID control, for example) can be applied to the DDMR. Below is the block diagram of PID control system in DDMR to perform the object tracking task.

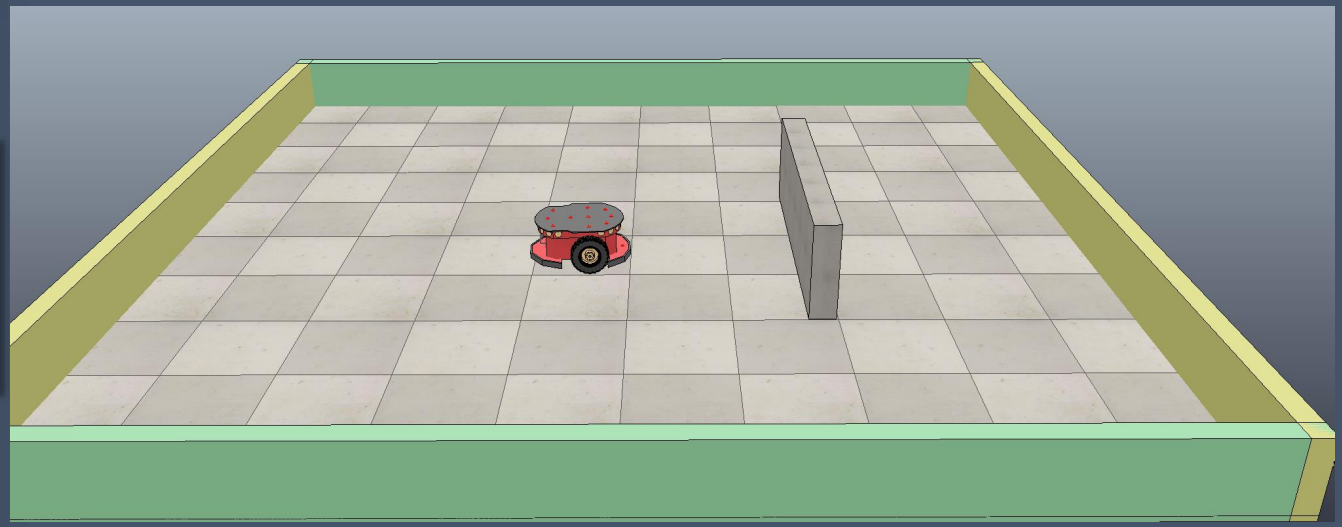


The simulation of the control principle applied to DDMR to perform the object tracking task.



TASK 2

1. Open the CoppeliaSim, pick and place the pioneer p3dx mobile robot, place a simple object at the front of robot, and create the wall around the floor of simulation area (see the figure).
2. Study the method to control the robot from external program, especially using Python programming.
3. Write and run the program to access the sensor and actuator of the robot (see page 5-7)
4. Create the python programming to control the robot motion manually using keyboard (see page 10 for reference)



THANK YOU