TECHNICAL UNIVERSITY OF CLUJ NAPOCA
AUTOMATION AND COMPUTER SCIENCE

Human-Interactive Doosan Robot Simulation ROS and Gazebo Keyboard Control Implementation

Student name: *Nagy Timea*

Course: *Robot Control Systems* – Professor: *Anastasios Natsakis*
Due date: *January 14th, 2024*

Section 1

General Description of ROS - Robotic Operating System

This project focuses on the integration of the Robot Operating System (ROS) and the simulation environment Gazebo to create a human-interactive Doosan robot simulation. ROS is an open-source middleware framework designed to develop, control, and simulate robotic systems. ROS Noetic is one of the latest versions of ROS, introduced to support the development and deployment of robotic systems. For simulating the robot, Gazeboo simulation environment can be used.

ROS enables users to choose a robotic platform based on their preferences, and in this case, the selected platform is the Doosan robot. The installation process can be found in document Project description.

Section 2

General Description of Doosan Robotic Platform

Doosan robots represent a line of industrial robotic arms designed for various applications, including manufacturing, assembly, and automation. Developed by Doosan Robotics, these robots are known for their precision, speed, and versatility. They offer a range of models suitable for different payloads and working environments.

The Doosan Robotic Platform can be easily installed by following the guidelines provided in the official repository: Doosan Robotics GitHub.

Steps for running the simulation after installation is complete:

1. Open ubuntu console and type

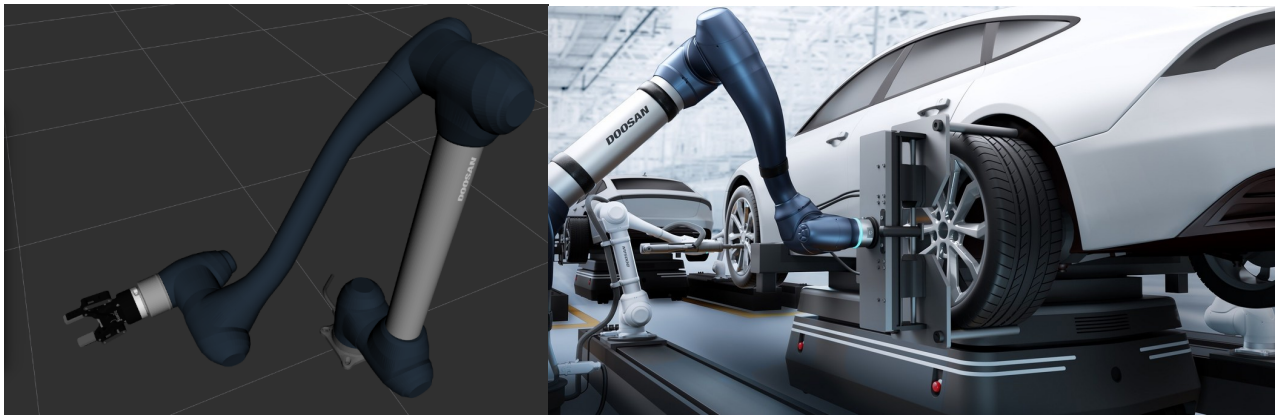
```
$ cd ~/catkin_ws
$ source ./devel/setup.bash
```

2. Start Gazeboo simulator in virtual mode

```
$ roslaunch dsr_launcher single_robot_gazebo.launch
mode:=virtual
```

3. Execute your code

```
$ rosrun dsr_example_py myProject.py
```



(a) ROS environment simulation (b) Doosan Robot

Figure 1: Illustration of ROS environment and the Doosan Robot

Section 3

Description of the task that the robot has to complete

The task involves controlling the robot's end-effector using keyboard inputs. The end-effector should be capable of moving along each axis, and specific keys are assigned for controlling each coordinate:

- Control X (red) coordinate: Q (move left), W (move right)
- Control Y (green) coordinate: A (move backward), S (move forward)
- Control Z (blue) coordinate: Z (move down), X (move up)

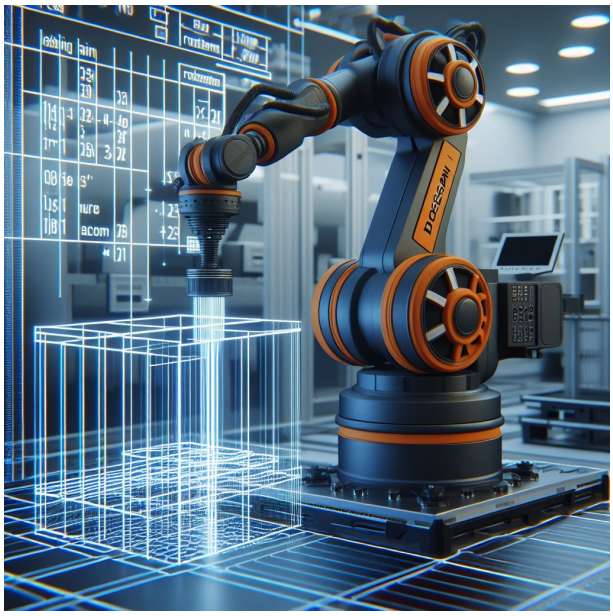


Figure 2: Doosan Robot on 3D Coordinate space

Section 4

Task implementation

We set the initial position in Cartesian space, currentPosition variable, which consistently represents Cartesian coordinates. Inverse kinematics is employed to determine the position in Joint space. The initial position in joint space is obtained through forward kinematics. When pressing a key, the position is changed in cartesian coordinates, then, converted into joint coordinates using inverse kinematics, the robot is moved with theese coordinates.

Listing 1: Keyboard Control

```
def keyboard_control(stdscr):
    stdscr.clear()
    stdscr.addstr(0, 0, "Control X(red)   coordinate: Q, W")
    stdscr.addstr(1, 0, "Control Y(green) coordinate: A, S")
    stdscr.addstr(2, 0, "Control Z(blue)  coordinate: Z, X")
    stdscr.addstr(3, 0, "Close: C")

    current_position = [0.0, -12.6, 101.1, 0.0, 91.5, -0.0]
    initial_position = [0.0, -12.6, 101.1, 0.0, 91.5, -0.0]

    velx = 1
    accx = 1
    t = 2
    movej(initial_position, velx, accx, t=t)
    current_position = fkin(initial_position, DR_WORLD)

    while True:
```

```
        key = stdscr.getch()
        if key == ord('z'):
            print("Z")
            current_position = [current_position[0],
                                current_position[1],
                                current_position[2] - 100,
                                current_position[3],
                                current_position[4],
                                current_position[5]]
            joint_positions = list(ikin(current_position,
                                        sol_space=2))
            movej(joint_positions, velx, accx, t=t)
        elif key == ord('x'):
            .
            .
            .
        elif key == ord('r'):
            print("Reset position")
            movel(initial_position, velx, accx, t=t)
        elif key == ord('c'):
            curses.endwin()
            break
```

Section 5

Video demonstration

The video demonstration can be found here: Doosan Robot YouTube.