Ph.D in ECE @ FEUP: Robotic Manipulators 2021/2022
Summarized Report of the Final Project: Bowling Robot Arm*

Ricardo B. Sousa Rui M. Coutinho

January 2022

1 Introduction

This work represents the final project developed in the scope of the course Robotic Manipulators (RM) of the doctoral program in Electrical and Computers Engineering (ECE) at the Faculty of Engineering of the University of Porto (FEUP). The project consists in developing an automated system based on a robot manipulator for playing bowling. The bowling game and the robot manipulator are simulated in the SimTwo [1, 2]. As for the application to interact with the user, it is developed in Lazarus, a cross-platform IDE compatible with Delphi. All the code is available in a GitHub repository[1].

The rest of the report is organized as follows. Section 2 presents the simulated robot manipulator. Section 3 presents the bowling scene developed in SimTwo. Section 4 formulates the forward and inverse kinematics of the robot. Section 5 explains the application developed in Lazarus. Section 6 presents the experimental results obtained in this work. Lastly, Section 7 presents the conclusions.

2 Robot Manipulator

A robot manipulator is required in this work to be able to pick the ball and throw it in the direction of the bowling pins. The manipulator used in SimTwo is illustrated in figure 1, in which the blue axes help identify the 7 Degrees of Freedom (DoF) of the robot. Six of them can be viewed as an independent articulated/anthropomorphic manipulator represented by a RRR (the first three revolute joints) kinematic arrangement followed by a spherical wrist (the other 3 revolute joints represented in figure 1) [3]. The other DoF simulates the anthropomorphic manipulator attached to a prismatic joint for adding kinematic redundancy.

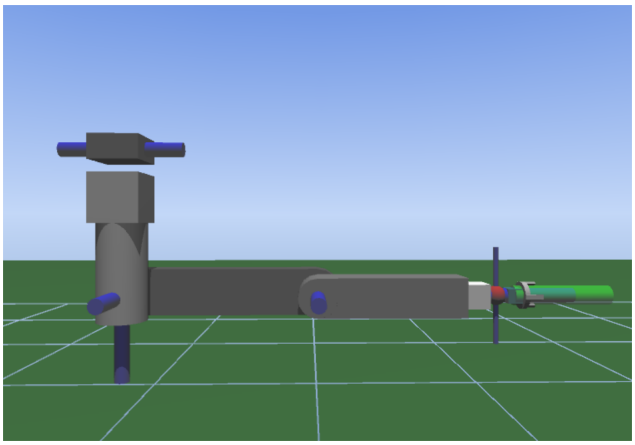


Figure 1: Robot manipulator used in SimTwo with 7 Degrees of Freedom (DoF)

The main reason to choose the anthropomorphic kinematic arrangement is due to its similar behavior to a human arm, given the goal of simulating a bowling game. Furthermore, the use of the spherical wrist allows an independent formulation of the robot's inverse kinematics in terms of position and orientation, as presented in Section 4 [3].

As for picking the bowling ball, the simulated robot has a magnetic actuator in its tool. The ball is modified to have ferromagnetic properties. These properties allow the magnetic actuator to grab the bowling ball. Although the goal would be attracting all magnetic objects around the actuator, SimTwo simulates the actuator only within a circumscribed cylinder.

In terms of our contribution for developing this arm in SimTwo, the original version only had 6 DoF equivalent to an anthropomorphic manipulator. We added the prismatic joint for having more flexibility to pick the ball and throw it. Furthermore, we adapted the wrist size to be compatible with the bowling ball dimensions. Although we tried to reduce the overall size of the robot for scale purposes, the simulation became unstable leading to not changing the original scale. So, we only changed the wrist dimensions. Another modification is the friction added to the three anthropomorphic joints of the robot. However, the current version is not fully stable and needs further improvements in future work.

*Ricardo B. Sousa[1, 2]
up201503004@edu.fe.up.pt

Rui M. Coutinho[1]
up201503006@edu.fe.up.pt

[1] Faculty of Engineering of the University of Porto, Electrical Engineering Department, Porto, 4200-465 Porto, Portugal
[2] INESC TEC – Institute for Systems and Computer Engineering, Technology and Science, CRIIS – Centre for Robotics in Industry and Intelligent Systems, Porto, 4200-465 Porto, Portugal

[1] https://github.com/sousarbarb/pdeec-rm

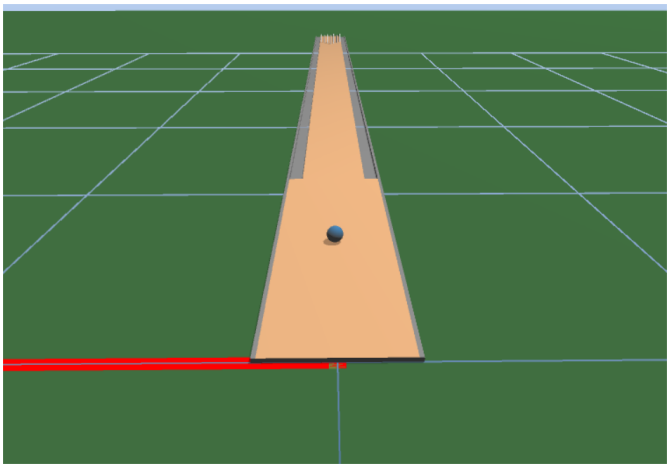


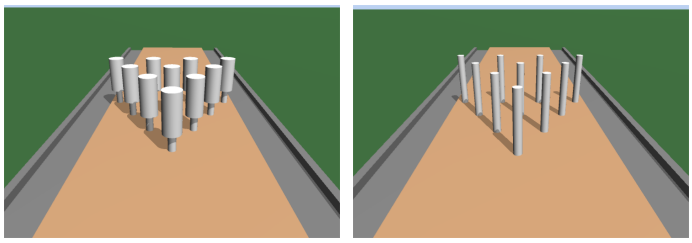
Figure 2: Bowling simulation scene developed in SimTwo

3 Bowling Scene

All the dimensions of the bowling scene developed in SimTwo are the same ones available in Dimensions – Database of Dimensioned Drawings[2] [3]. These dimensions follow the official equipment specifications and certifications manual[4] from the United States Bowling Congress (USBC). The SimTwo bowling scene developed by us is illustrated in figure 2. Relative to the USBC bowling official dimensions, we apply a scale reduction factor for avoiding the bowling lane to exceed the available space in SimTwo's scene.

Another detail is the simulation of the 10-pin bowling deck. Two different versions are illustrated in Figure 3. Given that SimTwo only has available simple collision models (e.g., cuboid, cylinder, or sphere), these versions are only composed of cylinders to try to model a 10-pin deck bowling pin. The first version, Figure 3a, is the most realistic one in which the radius of the two cylinders corresponds to the base and maximum outer diameters of a 10-pin deck bowling pin. The other version, Figure 3b, only models the base diameter of the bowling pin. Although the version in Figure 3a is more realistic, the simulation runs on approximately 12.9fps with no motion. In contrast, but using the same conditions, Figure 3b's version achieves 20.8fps. The main reason is that the first version requires the definition of a `robot` tag for each pin, while in the second version all pins are treated with the `things` tag being more light for the simulation. Note that the fps values should be interpreted relative to each other and not in terms of the absolute values due to being dependent on the computer performance. Even so, we chose the lighter version in terms of fps for having a fluid simulation and interaction with the user.

[2] `https://www.dimensions.com/element/bowling-lane`
[3] `https://www.dimensions.com/element/ten-pin-pin-deck`
[4] `http://usbcongress.http.internapcdn.net/usbcongress/bowl/equipandspecs/pdfs/ESManual.pdf`



(a) (b)

Figure 3: Simulated 10-pin bowling pin deck: (a) more realistic version; (b) simple and light version

Table 1: Denavit-Hartenberg (DH) parameters of the anthropomorphic robot arm

i	**Rot.z** (θ_i)	**Trans.z** (d_i)	**Trans.x** (a_i)	**Rot.x** (α_i)
1	θ_1^*	$-l_1$	0 m	90^o
2	θ_2^*	0 m	l_2	0^o
3	$\theta_3^* + 90^o$	0 m	0 m	90^o
4	$\theta_4^* + 90^o$	l_3	0 m	90^o
5	$\theta_5^* + 180^o$	0 m	0 m	90^o
6	θ_6^*	l_T	0 m	0^o

4 Kinematics

In this section, the forward and inverse kinematics are formulated for the robot manipulator used in this work. These kinematics are based on the ones presented in [3]. First, the kinematics are presented for the anthropomorphic arm. Then, the transformations between the world and the robot's base coordinate frame ($\{X^W, Y^W, Z^W\}$ and $\{X^0, Y^0, Z^0\}$, respectively) are explained in the context of setting the desired pose and orientation for the robot's tool.

4.1 Anthropomorphic robot manipulator

4.1.1 Forward kinematics

Following the Denavit-Hartenberg (DH) convention, the obtained coordinates frames for the anthropomorphic robot arm are the ones illustrated in Figure 4. These frames originate the DH parameters presented in Table 1.

Equation 1 formulates the homogeneous transformation dependent on the DH parameters of the i coordinate frame: joint angle (θ_i, rotation around z-axis), link offset (d_i, translation on the z-axis), link length (a_i, translation on the x-axis), and link twist (α_i, rotation around x-axis). Multiplying the 6 homogeneous transformation matrices equivalent to each entry i in Table 1 and as shown in the Equation 2, we obtain the forward kinematics for the anthropomorphic arm that we implemented.

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

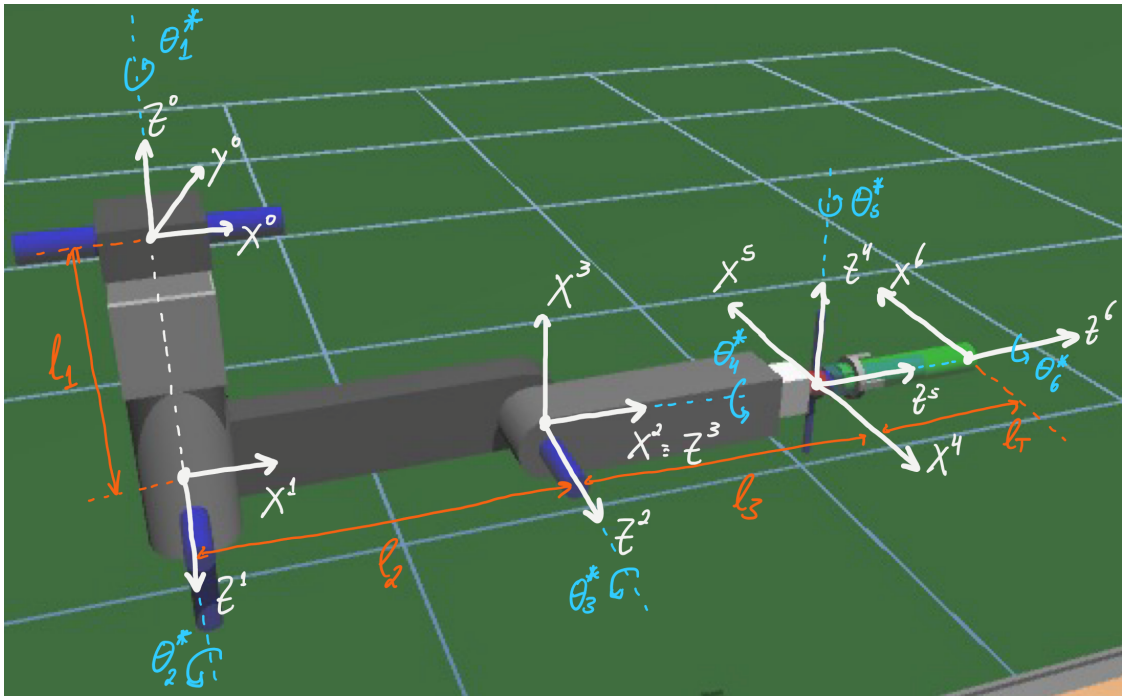


Figure 4: Coordinate frames of the anthropomorphic robot arm accordingly to the Denavit-Hartenberg (DH) convention

$$H_6^0 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \tag{2}$$

4.1.2 Inverse kinematics

In terms of inverse kinematics, as already stated in Section 2, one advantage of using a spherical wrist is the decoupling of position and orientation inverse kinematics. Consequently, we can formulate the inverse kinematics for the wrist position (o_c^0) dependent on the desired position and orientation for the arm's tool (o and R, respectively) while only using the first 3 revolute joints (θ_1, θ_2, and θ_3). This formulation is equivalent to Equation 3. Then, the tool's orientation is achieved by computing the orientation R_6^3 only dependent on the last three revolute joints (θ_4, θ_5, and θ_6) equivalent to the ones of the spherical wrist, as presented in Equation 4.

$$o_c^0 = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = o - R \cdot \begin{bmatrix} 0 \\ 0 \\ l_T \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - R \cdot \begin{bmatrix} 0 \\ 0 \\ l_T \end{bmatrix} \tag{3}$$

$$R_6^3 = \left(R_3^0\right)^T R \tag{4}$$

Furthermore, first, we need to compute the reference values required for achieving o_c^0. Joint 1 (θ_1) is computed by Equation 5. As shown in the equation, there are 2 possible solutions for this joint. We opted for the first one. A special case occurs when both x_c and y_c are 0 equivalent to the wrist being on the rotation axis of the joint 1. Given the infinite number of solutions for this case, we maintain the last reference value for θ_1.

$$\theta_1 = \text{atan2}\left(y_c, x_c\right) \text{ or } \pi + \text{atan2}\left(y_c, x_c\right) \tag{5}$$

Next, Equations 6, 7, and 8 compute auxiliary variables (s, r, and D). These variables are used to compute the value of joint 3 (θ_3), as formulated in Equation 9. Two possible solutions exist for the possibility of having the elbow up or elbow down (minus or plus in the equation, respectively). Our implementation considers both solutions depending on the user setting elbow up or down.

$$s = z_c - l_1 \tag{6}$$

$$r = \sqrt{x_c^2 + y_c^2} \tag{7}$$

$$\text{c}_{\theta_3} = \frac{s^2 + r^2 - l_2^2 - l_3^2}{2 l_2 l_3} := D \tag{8}$$

$$\theta_3 = \text{atan2}\left(\pm\sqrt{1 - D^2}, D\right) \tag{9}$$

Joint 2 (θ_2) is computed depending on the auxiliary variables (s, r, and D), links' lengths (l_2 and l_3, also illustrated in Figure 4), and the value of the joint 2 itself. This computation is formulated in Equation 10.

$$\theta_2 = \text{atan2}\left(s, r\right) - \text{atan2}\left(l_3\,\text{s}_{\theta_3}, l_2 + l_3\,\text{c}_{\theta_3}\right) \tag{10}$$

As for the inverse kinematics for the wrist's orientation (R_6^3), Equation 11 formulates the matrix R_6^3 dependent on the values of the joints 4, 5, and 6 (θ_4, θ_5, and θ_6, respectively). This equation was obtained using symbolic math in Matlab (scripts also available in our GitHub repository referred in Section 1). Then, the inverse kinematics for R_6^3 are formulated as in the Algorithm 1.

Algorithm 1: Inverse kinematics for the orientation of the wrist

input : R_6^3
output: θ_4,θ_5,θ_6

$\theta_5 = \text{atan2}\left(\pm\sqrt{1-r_{33}^2}, r_{33}\right)$
if $(r_{13} \neq 0)$ *AND* $(r_{23} \neq 0) \Leftrightarrow \text{s}_{\theta_5} \neq 0$, $\text{c}_{\theta_5} = 0$ **then**
  $\theta_4 = \text{atan2}\left(r_{13}, -r_{23}\right)$
  $\theta_6 = \text{atan2}\left(r_{32}, -r_{31}\right)$
else
  $\theta_4 = 0$
  if $r_{33} \Leftrightarrow \text{c}_{\theta_5} = 1$ **then**
    /* $\theta_4 + \theta_6 = \text{atan2}\left(r_{11}, r_{12}\right)$ */
    $\theta_6 = \text{atan2}\left(r_{11}, r_{12}\right)$
  else
    /* $\theta_4 - \theta_6 = \text{atan2}\left(-r_{11}, r_{12}\right)$ */
    $\theta_6 = -\text{atan2}\left(-r_{11}, r_{12}\right)$
  end
end

$$R_6^3 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \text{c}_{\theta_4}\,\text{s}_{\theta_6} + \text{s}_{\theta_4}\,\text{c}_{\theta_5}\,\text{c}_{\theta_6} & \text{c}_{\theta_4}\,\text{c}_{\theta_6} - \text{s}_{\theta_4}\,\text{c}_{\theta_5}\,\text{s}_{\theta_6} & \text{s}_{\theta_4}\,\text{s}_{\theta_5} \\ \text{s}_{\theta_4}\,\text{s}_{\theta_6} - \text{c}_{\theta_4}\,\text{c}_{\theta_5}\,\text{c}_{\theta_6} & \text{s}_{\theta_4}\,\text{c}_{\theta_6} + \text{c}_{\theta_4}\,\text{c}_{\theta_5}\,\text{s}_{\theta_6} & -\text{c}_{\theta_4}\,\text{s}_{\theta_5} \\ -\text{s}_{\theta_6}\,\text{c}_{\theta_6} & \text{s}_{\theta_6}\,\text{s}_{\theta_6} & \text{c}_{\theta_5} \end{bmatrix} \tag{11}$$

In the implementation, there is still a bug to resolve in the computation of θ_1 depending on which quadrant the desired pose is found. Even though we could not solve it on time of delivery, our implementation is still stable to play the simulated bowling game, as shown in Section 6.

4.2 World – robot base coordinate frames

The coordinates of the bowling ball are sent to our application in the world coordinate frame. For having consistency between the kinematics formulated in the previous subsection and the ball position, we convert the ball position from the world to the anthropomorphic arm's base coordinate frame ($\{X^W, Y^W, Z^W\}$ and $\{X^0, Y^0, Z^0\}$, respectively). The transformation between these two frames is formulated in the following two equations:

$$H_0^W = \begin{bmatrix} R_0^W & T_0^W \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0.57125 + q_0 \\ 0 & 1 & 0 & 0.00 \\ 0 & 0 & 1 & 1.15 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$H_W^0 = \begin{bmatrix} {R_0^W}^T & -{R_0^W}^T \cdot T_0^W \\ 0 & 1 \end{bmatrix} \tag{13}$$

in which q_0 is the current value of the prismatic joint. Thus, the homogeneous matrix H_W^0 is used to convert the ball coordinates to the robot's base coordinate frame.

5 Application

As for the application to communicate with SimTwo, it was developed in Lazarus that is a cross-platform IDE compatible with Delphi. A preview of the main window is shown in Figure 5 on the application's simulation tab, with which the user must interact for throwing the bowling ball.

The application communicates with SimTwo over a UDP connection. The simulator sends the current joints' position and velocity and the ball position. As for the application, the position reference for the joints and the desired state for the magnetic actuator is sent over the UDP connection. We focus on playing the simulated bowling game instead of improving the original control scheme (PID-based controllers for the joints' reference position). This approach was chosen for having the assurance of having a first stable version. Although the several improvements can be made to our application and specially on the control side, it is at least stable for the purpose of playing the bowling game with the simulated robot manipulator.

For playing the bowling game, you must choose the Ball control mode in the application. The ball position can be reset in the SimTwo sheet. Then, the following steps are required for playing (wait for the robot to reach its steady-state between steps):

1. Hover ball
2. Go to ball
3. Grab ball
4. Adjust the y-axis position by clicking on `+Y` or `-Y`
5. Throw ball

6 Results

The experimental results from this work are presented in a YouTube video[5]. This video shows how to execute the Lazarus-based application and the simulation environment for you to experiment later.

In terms of throwing the bowling ball, it is clear that our implementation is capable of doing that. However, the video also shows the main disadvantage of our implementation: the control scheme of the joints. Even though we only actuate the prismatic joint when throwing the ball, the other joints are affected by that actuation. The reason is due to the original PID-based controlling scheme assuming independent joint control. The inertia from throwing the ball grasped by the arm's tool leads to disturbances on all the joints. A possible solution would be multivariable control using a state-space control scheme. This control scheme could consider the problem of inverse dynamics for diminishing the influence of the arm's mass and inertia on motion, even though the ball would still be treated as a disturbance. However, this approach was not implemented and was left out for future work.

[5] `https://www.youtube.com/watch?v=lY1BQs15W1s`

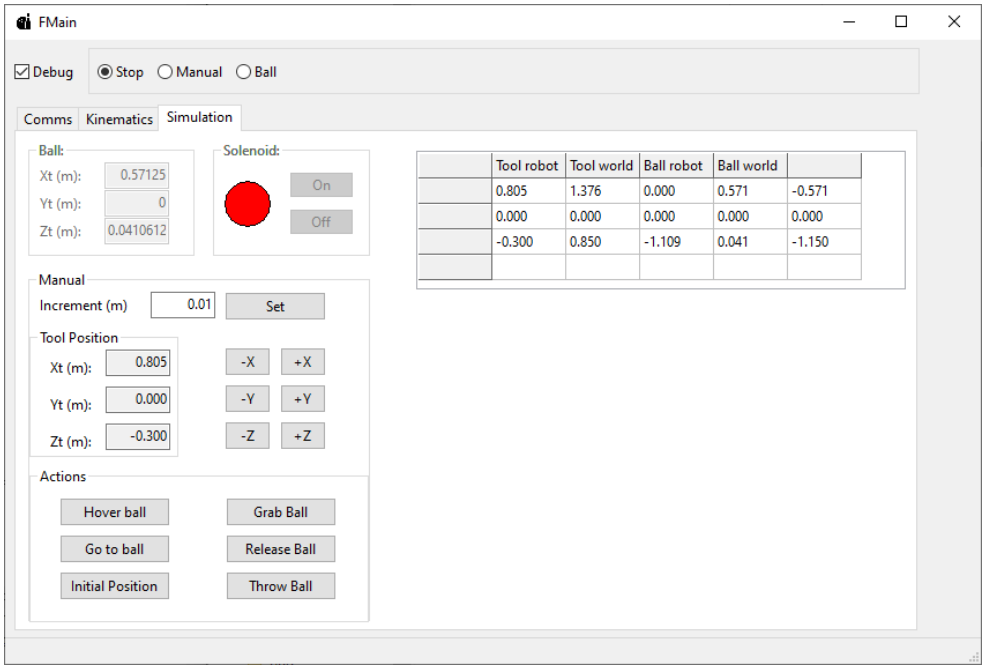


Figure 5: Lazarus-based application for playing the simulated bowling game

Finally, the possibility of adjusting in `+Y` or `-Y` for throwing the ball resulted in the expected result of changing the final trajectory of the ball.

7 Conclusions and Future Work

In conclusion, this work can simulate a bowling game, even though we had problems in terms of stabilizing the simulation and related to the joint's control scheme. All the dimensions of a bowling lane can be easily scaled for modeling a real implementation of a downsized version. Moreover, the application developed in Lazarus is modular in the terms of separating the GUI from the kinematics implementation of the robot facilitating further improvements to this work.

As for future work, the simulation must be modified to be more stable. The robot arm could also be downsized to allow a real implementation of the whole system. In terms of control, the control scheme can be changed to velocity-based control to allow the definition of the throwing velocity. Lastly, the kinematic redundancy of the 7 DoF robot manipulator can be useful to leverage. One possibility would be the computation of the singular valuables of the anthropomorphic's Jacobian matrix and evaluating the inverse kinematics solution that leads to the highest of the smallest singular values. Another possibility would be implementing an optimization-based approach to evaluate the joint space solutions that best suit a trajectory given by the user considering different cost functions.

References

[1] P. Costa, J. Gonçalves, J. Lima, and P. Malheiros. SimTwo realistic simulator: a tool for the development and validation of robot software. *Theory and Applications of Mathematics & Computer Science*, 1(1):11–16, Apr. 2011. `https://hdl.handle.net/10198/4117`.

[2] P. Costa. Simtwo – a realistic simulator for robotics. Accessed on Nov. 22, 2021. `https://github.com/P33a/SimTwo`.

[3] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot modeling and control*. John Wiley & Sons, inc., 1 edition, 2005.