AI Training for Application to Industrial Robotics: Trajectory Generation for Neural Network Tuning

Mikel Merino[1(✉)], Javier Ibarrola[1], Jokin Aginaga[1,2], and Mikel Hualde[1]

[1] Departamento de Ingeniería, Universidad Pública de Navarra, Pamplona, Spain
mikel.merino@unavarra.es

[2] Institute of Smart Cities (ISC), Universidad Pública de Navarra, Pamplona, Spain

Abstract. In the present work robot trajectories are generated and kinematically simulated. Different data (joint coordinates, end effector position and orientation, images, etc.) are obtained in order to train a neural network suited for applications in robotics. The neural network has the goal of automatically generating trajectories based on a set of images and coordinates. For this purpose, trajectories are designed in two separate sections which are conveniently connected using Bezier curves, ensuring continuity up to accelerations. In addition, among the possible trajectories that can be carried out due to the different configurations of the robot, the most suitable ones have been selected avoiding collisions and singularities. The designed algorithm can be used in multiple applications by adapting its different parameters.

Keywords: Industrial robotics · Trajectory planning · Artificial Intelligence

1 Introduction

In recent years, the rapid advance of industrial robotics has made it possible to automate a multitude of operations. However, automation is more complicated for certain operations that present some degree of randomness. The development of neural networks (NN) in the field of artificial intelligence (AI) is making it possible to automate this type of task. Specifically, in the project in which this work is included, a NN will be used to generate trajectories from images. The algorithm will be able to identify the object to be picked up and command the trajectory. This work will focus on generating the data sets in the form of images and time series necessary to adjust the parameters of the NN in the initial training phase.

2 Trajectory Generation

In robotics, a trajectory is the desired motion of a manipulator from an initial point to an end point, described by the position and orientation of the robot's end effector. The generation of trajectories seeks to find the functions that describe the position, velocity

A. Vizán Idoipe and J. C. García Prada (Eds.): IACME 2022, *Proceedings of the XV Ibero-American Congress of Mechanical Engineering*, pp. 405–411, 2023.
https://doi.org/10.1007/978-3-031-38563-6_59

and acceleration as a function of time of the joints, knowing the desired initial and final position and orientation for the end effector. In this work a series of trajectories will be generated using the UR10e robot. It is a serial manipulator with 6 degrees of freedom whose kinematics have been described using the Denavit-Hartenberg parameters [1–3]. The robot has the task of gripping a cable whose position will determine the final position of the end effector.

Trajectories can be defined either in the operation space or in the joint space. Both are related by means of direct and inverse kinematics. The direct kinematics has only one solution, while the inverse kinematics has 8 solutions for the UR10e robot. In this work, both the operation space and the cartesian space will be used, since the trajectory has been divided into two distinct sections. The first section is called *approach* and the robot moves from its initial pose (P1) to an intermediate pose (P2) close to the final pose (P3). In this stage, the aim is that the robot travels the distance in the shortest possible time, and it is based on the joint space. An intermediate pose is reached and the necessary orientation for the subsequent manipulation of the object is achieved. The second section is called *slow*, moving the robot from P2 to P3. In this section the manipulator moves to P3 with reduced speed and without changing the orientation of the end effector. This section has been defined in the operating space.

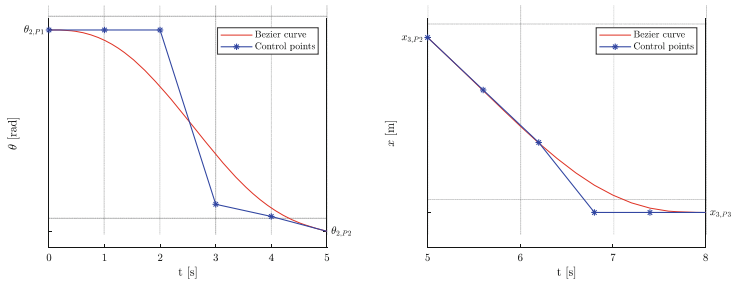
2.1 Trajectory Point Calculation

To generate the trajectory given the initial and final pose of the robot, the first step is calculating the intermediate pose where the two sections of the trajectory meet. This pose has the same orientation as the final pose and x,y,z cartesian coordinates that are calculated with the desired distance (in modulus) between the cartesian coordinates that define P2 and P3. Then, using inverse kinematics [4, 5], the joint coordinates of the initial, intermediate and final poses are obtained. The same configuration of the robot has been used for the three calculations so that it uses the same configuration during the whole trajectory. After this process, the joint coordinates of the robot (vector $\boldsymbol{q}$) and the cartesian coordinates of the end effector (vector $\boldsymbol{x}$) have been obtained for poses P1, P2 and P3.

2.2 Trajectory Design Using Bézier Curves

In this work polynomial Bézier curves have been used to design the two sections that will compose the complete trajectory of the robot. For the first section of the trajectory, a total of 6 Bézier curves have been defined since there are 6 joint coordinates. To guarantee continuity up to accelerations, the control points of the vector of control points $\boldsymbol{c} = [c_1, c_2, c_3, c_4, c_5, c_6]^T$ have been defined in such a way that they fulfill certain requirements. In order that the joint velocities ($\dot{\boldsymbol{q}}$) and accelerations ($\ddot{\boldsymbol{q}}$) are zero at the initial instant, it has been imposed that the first three control points have the value of the joint coordinates at P1, so that $c_{1i} = c_{2i} = c_{3i} = \theta_{i,P1}$ where i refers to the i-th joint coordinate. To ensure continuity in positions at P2, it must be satisfied that $c_{6i} = \theta_{i,P2}$, i.e., the last control point must be equal to the joint coordinates of P2. To ensure continuity in velocities and accelerations at P2, the values of c_{5i} and c_{4i} are calculated in such a way that the desired velocities and accelerations are equal to the ones

from the Bézier curves. In the second section, since the orientation of the end effector is constant, only 3 Bézier curves are necessary. An analogous procedure to the first section has been followed and the results have been transferred to the joint. Hence, the complete trajectory is univocally described Fig. 1.



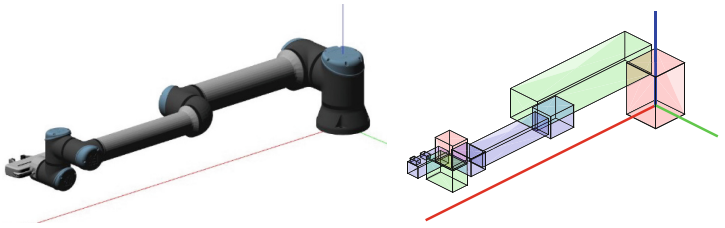
(a) (b)

Fig. 1. (a) Joint coordinate θ_2 evolution in first trajectory section. (b) Cartesian coordinate x_3 evolution in second trajectory section.

3 Trajectory Selection

Under the assumptions of Sect. 2, a total of 8 trajectories have been generated for given initial and final poses, so that each trajectory corresponds to each of the robot configurations. These 8 trajectories represent 8 possible solutions for the robot to move from the same initial pose to the same final pose.

In order to eliminate trajectories in which there is some type of collision (self-collision or with an object in the environment), a collision detection algorithm has been designed. The robot has been modeled using rectangular parallelepipeds in order to simplify and speed up the algorithm, as shown in Fig. 2.



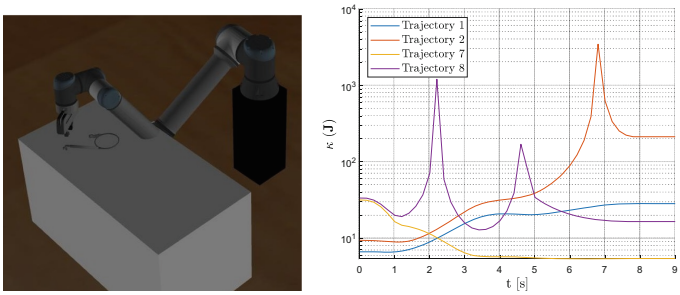
(a) (b)

Fig. 2. (a) Original solid modelled robot in Gazebo. (b) Collision solid modelled robot.

On the other hand, the trajectories in which the robot passes close to a singularity [6, 7] have been eliminated by establishing a threshold value of the condition number

of the Jacobian matrix. When this threshold is exceeded trajectory is rejected. Figure 3 shows an example of trajectory rejection due to collision and singularity.

Finally, three criteria have been defined in order to select among the trajectories not rejected: minimization of the maximum joint velocity, minimization of the distance covered by the end effector and minimization of the maximum value of the condition number. The final trajectory selection algorithm is described in Fig. 4.



(a) (b)

Fig. 3. Trajectory rejection criteria application. (a) Collision based example. (b) Singularity based example (trajectories 2 and 8 rejected).

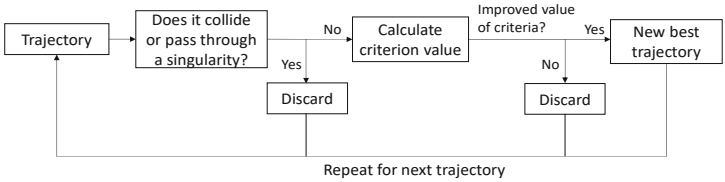


Fig. 4. Trajectory selection algorithm diagram.

4 Generation of Data Based on the Optimal Trajectory

In order to feed the neural network, a single trajectory is not enough, being necessary a large number of trajectories in which some of their parameters are varied. In order to generate this set of trajectories, the following parameters of the cable to be taken by the robot have been randomly varied: position, radius, opening angle, torsion angle of the terminals and length of the cable. Thus, the final pose to be reached by the robot can be varied and a large number of random trajectories can be created. The generated data sets are composed of a series of images and text files with time series (position, joint velocity and acceleration, position and orientation of the end effector, etc.) organized as shown in Fig. 5.

The images of the manipulator carrying out the trajectory at different time instants is the part that provides the most information to the AI since after the training with these images it will be able to generate trajectories autonomously. These images have been

obtained using the free robotics simulation software Gazebo, where a virtual environment as close as possible to the real one has been created. Thus, at each discrete instant of the trajectory, the position of the robot has been varied and the necessary images have been taken.

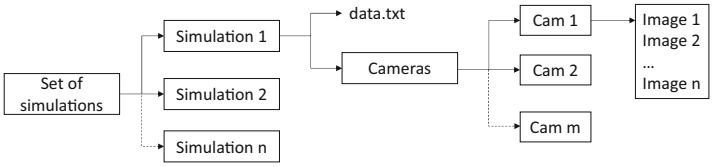


Fig. 5. Organization of a data-set.

On the other hand, the time series data is stored in a text file containing the header in the first row with all the names of the variables to be stored: time, joint coordinates, velocities and accelerations, and position and orientation of the end effector in cartesian coordinates. The remaining rows represent the value of each of these variables at a given time instant.

5 Results

In this Section the results obtained after generating several datasets that can be used to feed the AI will be analyzed. First, a single trajectory will be analyzed as an example and then some general results of the data sets obtained will be discussed.

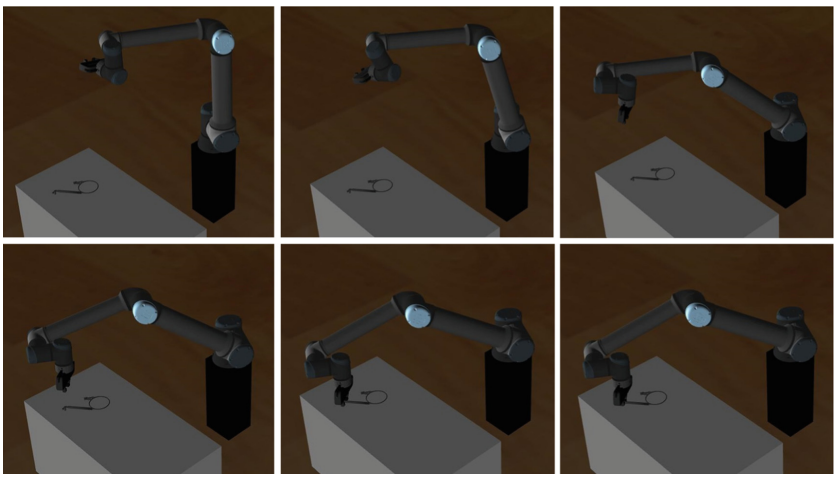


Fig. 6. Example trajectory evolution.

In this example, 4 of the 8 trajectories have been discarded (trajectories 3–6) due to collisions with the table on which the cable is located. These configurations are called

"elbow down" configurations. Configurations 2 and 8 have also been discarded since they exceed the threshold established for the condition number of the jacobian matrix. Among the remaining valid trajectories number 1 has been chosen based on the criterion of minimizing the maximum number of condition. Images of different instants of the selected trajectory can be seen in Fig. 6.

Regarding the results of the complete data set composed of 1000 trajectories, it has been found that trajectories 1 and 7 are the most used with usage percentages of 42.84% and 26.35% respectively. Configurations 3 to 6 have not been used in any occasion because they always collide with the table as they are "elbow down". Trajectories 2 and 8 have a very low percentage of use of 3.2% and 0.46%, the rest (27.15%) are erroneous trajectories where all the trajectories have been rejected due to collisions and/or singularities.

6 Conclusions

This work presents a methodology to create robotic trajectories and select between them in order to obtain a data-set to train a neural network. The advantages of generating trajectories in a kinematic way in terms of simplicity of trajectory generation and ideality of the trajectories have been tested. An algorithm has been created to avoid collisions and also singularities and their neighborhood. The generated trajectories have been used to export a series of data (images and time series) that will be used for the first training phase of the neural network. In addition, the algorithm has been designed parametrically so that it could be used for other similar applications.

Acknowledgements. This work was funded by the "Convocatoria de ayudas a proyectos de I+D del Gobierno de Navarra" under the projects with Ref. 0011–1365-2021–000080 and Ref. 0011–1411-2021–000023.

References

1. Denavit, J., Hartenberg, R.S.: A kinematic notation for lower-pair mechanisms based on matrices. J. Appl. Mech. **22**, 215–221 (1965)
2. Siciliano, B., et al.: Robotics: Modelling. Planning and Control. Springer, London (2010)
3. Angeles, J.: Fundamentals of Robotic Mechanical Systems. Springer, New York (2007). https://doi.org/10.1007/978-0-387-22458-9_4
4. Zaplana, I., Claret J.A., Basanez, L.: Análisis cinemático de robots manipuladores redundantes: Aplicación a los Robots Kuka LWR 4+y ABB Yumi. In: Revista Iberoamericana de Automática e informática Ind. **15**, 192–202 (2018)
5. Qiang, L., et al., W.: Research on kinematic modeling and analysis methods of UR robot. In: 2018 IEE 4th Information and mechatronics Engineering Conference, pp. 159–164 (2018)
6. Craig, J.J.: Introduction to Robotics: Mechanics and Control. Pearson Education, New Jersey (2005)
7. Tsai, L.-W.: Robot Analysis. Wiley, New York (1999)