Generation of Smooth Cartesian Paths Using Radial Basis Functions*

Leon Žlajpah[1][0000−0002−2820−2697] and Tadej Petrič[1][0000−0002−3407−4206]

Dept. of Automation, Biocybernetics and Robotics, Jožef Stefan Institute
Ljubljana, Slovenia
{leon.zlajpah,tadej.petric}@ijs.si

Abstract. In this paper, we consider the problem of generating smooth Cartesian paths for robots passing through a sequence of waypoints. For interpolation between waypoints we propose to use radial basis functions (RBF). First, we describe RBF based on Gaussian kernel functions and how the weights are calculated. The path generation considers also boundary conditions for velocity and accelerations. Then we present how RBF parameters influence the shape of the generated path. The proposed RBF method is compared with paths generated by a spline and linear interpolation. The results demonstrate the advantages of the proposed method, which is offering a good alternative to generate smooth Cartesian paths.

Keywords: Robot Motion Generation, Gaussian RBF, Path Interpolation

1 Introduction

One of the fundamental tasks in robotics is to provide the capability to move the manipulator arm and its end effector from the initial pose to the final pose. To plan the motion of the robot different planning algorithms are used that generate appropriate trajectories to perform the task. First, it is necessary to define a sequence of points that the robot has to follow. This pure geometric description of the motion is termed as *path*. Next, a timing law has to be applied to the path. i.e. the velocities (and accelerations) in each point along the path. The path together with velocities (timing) defines the *trajectory*.

The trajectories can be generated in a joint or task (Cartesian) space. The joint space trajectories fully specify the position and orientation of the robot's end-effector and are in general easier to generate as the trajectories in the task space [8]. However, using joint trajectories it is not easy to predict the resulting task-space motion of the end-effector due to the nonlinear effects introduced by the direct kinematics. The geometric path can be more natural specified

* This work was supported by EU Horizon 2020 Programme grant 820767, CoLLaboratE and by Slovenian Research Agency grant P2-0076, and IJS Director's found grant CoBoTaT.

in the Cartesian (task) space. Therefore, it is preferred to do the path and trajectory planning in the task space. The task space trajectories are usually based on geometric paths that pass through several given waypoints and the trajectory planning consists to generate a time sequence of values (considering the constraints) that specify the evolution of positions and orientations of the end-effector.

When the task does not specify the motion exactly, it can be specified in an approximated way by using waypoints as the key path points and for the motion between waypoints different interpolation methods can be used [8, 9, 12]. Selection of the suitable interpolation methods and the timing law to represent the motion between waypoints depends on task requirements like continuity, velocity profile, limits, etc. Regardless of the particular method used, it is necessary that the planning algorithm outputs smooth trajectories, which express a high order of continuity. In practice, it is preferable that generated trajectories ensure continuity of accelerations to obtain trajectories with bounded jerks. To obtain smooth trajectories different blending algorithms have been proposed. The simplest one are linear interpolation functions which result in straight line segments between consecutive pairs of waypoints. However, such trajectories ensure only $\mathcal{C}^0$ continuity, meaning that the velocities at waypoints are not smooth (infinite accelerations at waypoints). To obtain higher-order continuity, parabolic blends [7, 8] or circular blends [6] have been proposed, which ensure continuity but the trajectory does not pass through the waypoints except at the beginning and at the end of the path. Generating trajectories passing through the waypoints is possible when higher-order polynomial functions are used where velocity (and acceleration) boundary constraints ensure the desired continuity of the trajectory at waypoints [3, 10, 8, 9].

For complex robot motion when the path cannot be defined analytically, a common approach is to use a suitable parametrization. In this context, radial basis functions (RBF) have been successfully used to approximate the path. They can form a basis for dynamic motion primitives [5] or can be used directly to parametrize a function [11]. The use of RBFs for trajectory generation is not very common. Lately, in [1] RBFs have been proposed to be used to plan smooth joint PTP trajectories.

In contrast to joint-space trajectories, where motion in all joints can be generated using the same algorithm, the task-space trajectory generation requires to consider distinctive features of positions and orientations of the end-effector. For positions, any Euclidean interpolation method can be used. On the other hand, rotations are more complex and rotation interpolation methods depend on the representation and implementation of rotation [2]. A good survey of interpolation methods considering positions and orientations is given in [4]. They conclude that it is preferable for most interpolations in Cartesian space to use split interpolation of positions and rotations.

In the following, we present a new approach to generate smooth Cartesian paths. In Section 2 a method for path generation based on RBF interpolation is described, and we discuss how to select RBF parameters to obtain the desired

shape of the path. Then, in Section 3 we apply the RBF method to generate a Cartesian path. Next, in Section 4 we compare the proposed method with the common methods using spline or linear interpolation. Finally, we give some concluding remarks.

2 Path generation problem

The path generation problem can be defined as follows. Given the data set of points $(x_i, y_i), i = 1, \ldots, N$, which represent the waypoints of a trajectory, find a $\mathcal{C}^k$ continuous functions such that

$$f(x_i) = y_i, \text{ for } i = 1, \ldots, N \text{ and } x_i \neq x_j, i \neq j \tag{1}$$

We assume that (x_1, y_1) is the start and (x_N, y_N) the end of the path.

2.1 Radial basis functions

To solve the problem we propose to use radial basis functions (RBF). Among many possible radial basis functions we have selected ones with Gaussian kernels defined as

$$\Psi(x) = \exp(-\frac{(x-c)^2}{2\sigma}) \,, \tag{2}$$

which is centered at c and h is defining the width of the kernel function. An important feature of this kernel function is that it expresses $\mathcal{C}^\infty$ continuity.

The function f can be defined as a linear combination of wighted radial basis functions. Instead of using directly the function Ψ as defined in (2), we use a normalized version of this function, which yields

$$f(x) = \frac{\sum_{j=1}^{m} w_j \Psi_j(x)}{\sum_{j=1}^{m} \Psi_j(x)} \tag{3}$$

where m is the number of kernel functions. Applying this to the given data set, it makes sense that a kernel function is assigned to each point in the data set. This means that $m = N$ and kernel functions centered at the data points, $c_i = x_i$. Using $\mathbf{\Phi}(s) \in \mathbb{R}^m$ defined as a row vector with components

$$\mathbf{\Phi}_k(x) = \frac{\Psi_k(x)}{\sum_{j=1}^{m} \Psi_j(s)} \tag{4}$$

in (3) yields

$$f(x) = \mathbf{\Phi}(x)\boldsymbol{w} \,. \tag{5}$$

where $\boldsymbol{w} \in \mathbb{R}^N$ is a vector with elements w_j. Applying (3) to the given data set of points (1) we obtain a set of linear independent equations

$$\mathbf{A}\boldsymbol{w} = \boldsymbol{y} \tag{6}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ is a vector with elements y_i, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a matrix with rows $\boldsymbol{\Phi}(x_i)$. The corresponding wights $\boldsymbol{w}$ can be found by solving (6)

$$\boldsymbol{w} = \mathbf{A}^{-1}\boldsymbol{y} \; . \tag{7}$$

Having in mind that we are generating a path for a robot, it is necessary to assign the initial and the final state. In most cases this means to define the velocity and acceleration for the first and the last point. Therefore, we add two auxiliary kernel functions near x_1 and two near x_N. Hence, $\hat{\boldsymbol{\Phi}} \in \mathbb{R}^{N+4}$. Next, the system of equations (6) is augmented with the equations for boundary conditions

$$\left.\frac{\partial \hat{\boldsymbol{\Phi}}(x)}{\partial x}\right|_{x_i} \hat{\boldsymbol{w}} = \dot{y}_i, \; i = 1, N \tag{8}$$

$$\left.\frac{\partial^2 \hat{\boldsymbol{\Phi}}(x)}{\partial^2 x}\right|_{x_i} \hat{\boldsymbol{w}} = \ddot{y}_i, \; i = 1, N \tag{9}$$

and new weights $\hat{\boldsymbol{w}} \in \mathbb{R}^{N+4}$ are calculated by solving the augmented set of equations.

2.2 Selection of kernel function parameters

When RBFs are used for approximation there are many more data points than kernel functions, $m \gg N$, and the RBFs are centered equidistantly. Here, one kernel function is centered at each data point. When the distance between the data points is not equal, the influence of the kernel functions is not the same on both sides. Therefore, we propose to map the data points so, that they are spread equidistantly, calculate the weights for mapped data points and finally, remap them back. Fig. 1 shows interpolation for an example data set. As we can see, interpolation with equidistant data points gives better results, i.e. lower oscillations.

The interpolation depends significantly also on the parameter σ as shown in Fig. 2. We can se that the selection of σ is trade-off between oscillatory interpolation and higher values of velocities and accelerations.

Although the auxiliary kernel function for boundary conditions can be placed anywhere, we propose to center relatively close to the first and the last point, $c_a = x_1 + k(x_2 - x_1)$ and $c_a = x_N - k_c(x_N - x_{N-1})$ with $k_c = (0.05, 0.1)$, respectively. Additionally, we use for them larger σ_a, $\sigma_a = 3\sigma$.

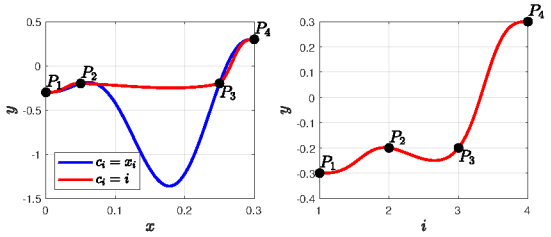

Fig. 1. RBF interpolation using original data set (left figure,blue) and with equidistantly mapped data points (right figure,red). P_i represent the data set ($\boldsymbol{x} = [0, 0.05, 0.25, 0.33]^T, \boldsymbol{y} = [-0.3, -0.2, -0.2, 0.3]^T$).

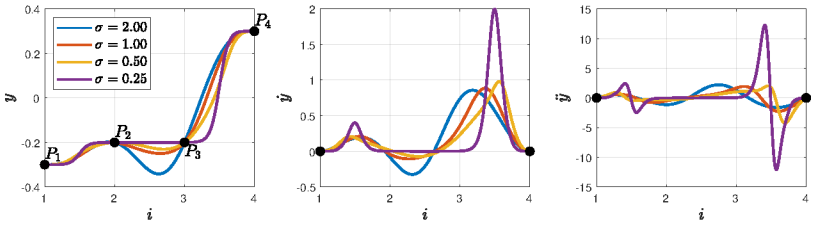

Fig. 2. RBF interpolation versus kernel width σ. P_i represent the data set ($\boldsymbol{x} = [0, 0.05, 0.25, 0.33]^T, \boldsymbol{y} = [-0.3, -0.2, -0.2, 0.3]^T$); dots on velocity and acceleration plot show the conditions for calculations of weights $\boldsymbol{w}$.

3 Cartesian space paths

Let the path $\mathcal{P}$ be given as a curve $\boldsymbol{f}$ in Cartesian (task) space. Three dimensions of Cartesian space represent the position $\boldsymbol{p} \in \mathbb{R}^3$ of the curve which the robot end-effector has to follow, and the remaining three the orientation of the end-effector. Selecting quaternions $\mathcal{Q} \in \mathbb{SU}(2) \cong \mathbb{S}^3$ to represent the rotations, $\mathcal{P}$ can be represented as

$$\boldsymbol{x} = \{\boldsymbol{p}, \mathcal{Q}\} = \boldsymbol{f}(s) \,, \tag{10}$$

where $\boldsymbol{x} \in \mathbb{R}^3 \times \mathbb{SU}(2)$ and s is a strictly increasing scalar path parameter varying in $[s_{\text{start}}, s_{\text{end}}]$. As s increases for s_{start} to s_{end}, point $\boldsymbol{x}$ moves along the path. If the path parameter s is the time, then $\boldsymbol{f}(s)$ is a *trajectory*. Otherwise, s is an arbitrary parameter and $\boldsymbol{f}(s)$ represents the geometric path.

In many situations, especially when not accurate tracking is required, the motion is specified by a sequence of waypoints that the trajectory has to pass through. The motion between waypoints is then interpolated using different blending algorithms. One possibility is to use RBF based interpolation methods that will be presented next.

3.1 Path generation using waypoints

Compared to the trajectory generation in the joint space, the problem with Cartesian space methods is that for orientations we can not use methods developed for Euclidean space interpolation directly. In practice, a good approximation is to interpolate orientations using methods for Euclidean space followed by a renormalization or re-orthogonalization.

Following this strategy, we apply the previously defined RBF interpolation method to a set of Cartesian points representing the waypoints defined as $P_i = (\boldsymbol{p}_i, \mathcal{Q}_i) \in \mathbb{R}^7$, where $\mathcal{Q}_i$ are quaternions representing the orientation. Using (5) to represent Cartesian path $\boldsymbol{f}(\hat{s})$ with same RBF kernels (4) for all dimensions yields

$$\boldsymbol{x}_i = \boldsymbol{\Phi}(s(\boldsymbol{x}_i))\boldsymbol{w} \ , \tag{11}$$

where $s(\boldsymbol{x}_i)$ is the path parameter value for the point $\boldsymbol{x}_i$, and $\boldsymbol{w} \in \mathbb{R}^{N\times 7}$ is a matrix representing wights for all components of $\boldsymbol{x}$. Regardless of how the path parameter s is defined, we calculate the wights using equidistant kernel functions by selecting path parameter $\hat{s}(\boldsymbol{x}_i) = i - 1$. To use another parametrization of the path s it is necessary to define the mapping between the path parameter $\hat{s}$ and the parameter s and then use the interpolated points with this path parameter.

Note that the proposed method gives the correct (unit) quaternion only at the waypoints. To get the correct quaternion component values for the path points between the waypoints it is necessary to renormalize the quaternion part of the interpolated point $\boldsymbol{x}$ from the (11) so that the quaternion becomes a unit quaternion.

4 Illustrative example of Cartesian path generation

To test the proposed method and compare it with the cubic spline interpolation method we have selected a task where the robot has to move along a path passing through a sequence of waypoints defined in Table 1.

Table 1. Waypoints defining a Cartesian path

$p_i = (x, y, z)$	Q_i
(-0.2, -0.2, -0.3)	(1.0, 0.0, 0.0, 0.0)
(-0.2, -0.2, 0.1)	($\sqrt{2}$/2, -$\sqrt{2}$/2, 0.0, 0.0)
(-0.2, 0.1, 0.1)	(0.5, -0.5, 0.5, -0.5)
(0.3, 0.1, 0.1)	($\sqrt{2}$/2, 0.0, 0.0, -$\sqrt{2}$/2)

For spline interpolation, we have used for the positions the classic Euclidean space cubic spline interpolation and for the orientations the spherical spline quaternion interpolation (Squad) [2]. Figs. 3 and 4 show the generated curves using RBFs and splines. As we can see, tuning the σ parameter of RBF kernels allows obtaining with RBF interpolation method a curve, which is very similar to

the curve obtained by spline interpolation. As RBF kernels are $\mathcal{C}^\infty$ the velocities and accelerations are smoother than cubic splines that are $\mathcal{C}^2$. Of course, it is possible to use higher-order splines, but this makes the path generation more complex. The benefit of using RBF interpolation is also the calculation time, which is lower for RBF compared to spline interpolation.

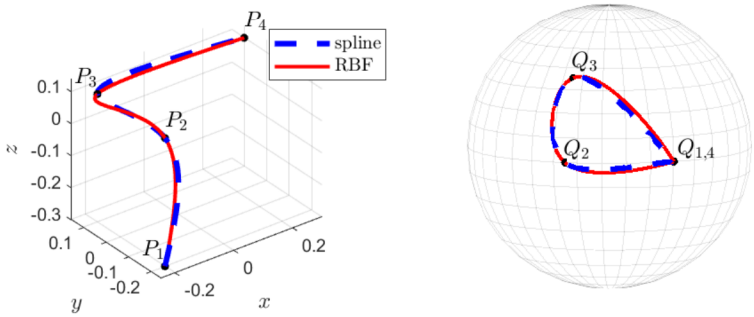

Fig. 3. Comparison of RBF and spline interpolation using waypoints: position in 3D (left), orientation on unit sphere (right) (To show smoothness of orientation interpolation curves, one component of quaternion is fixed and the other three are normalized and shown on unit sphere). RBF parameters: $\sigma = 0.6$.

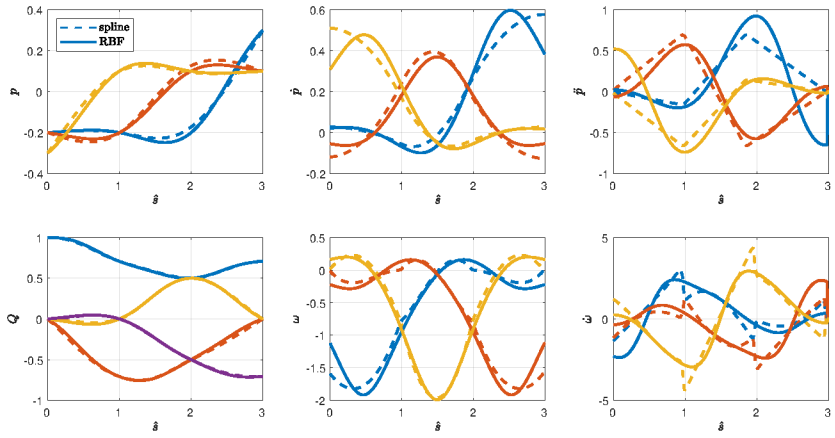

Fig. 4. Comparison of RBF (line) and spline (dashed line) interpolation using waypoints: positions and quaternion components (left), linear and angular velocities (middle), linear and angular accelerations (right). RBF parameters: $\sigma = 0.6$.

Using RBF interpolation we can easily add initial and final conditions for velocity and accelerations. Fig. 5 shows the generated curve with initial and final velocities/accelerations set to 0. Due to these conditions, velocities and accelerations near initial and final point all higher (for the same path parameterization).

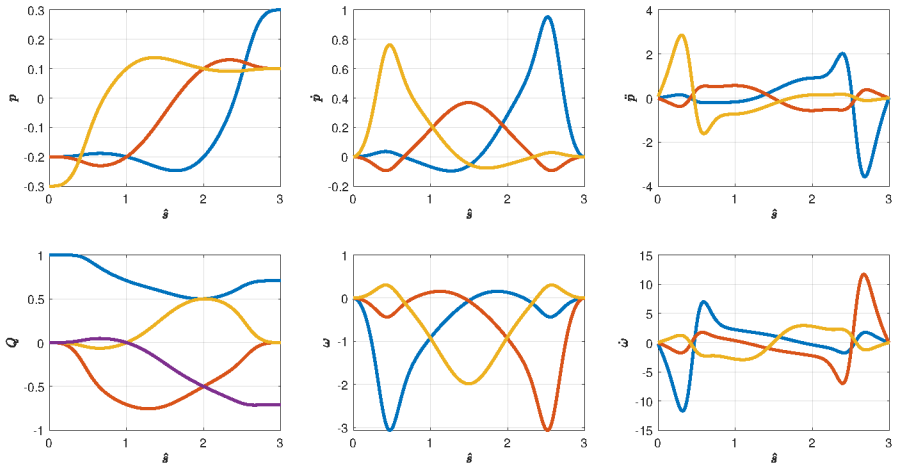

Fig. 5. RBF interpolated curve with initial and final boundary conditions for velocity and acceleration: positions and quaternion components (left), linear and angular velocities (middle), linear and angular accelerations (right). RBF parameters: $\sigma = 0.6$, $k_c = (0.05, 0.1)$ and $\sigma_a = 3\sigma$.

The RBF parameters allow to shape the path. For example, with low σ values the generated path the "corners" become sharper. Fig. 6 shows a path where segments between the waypoints are almost linear as when the path is generated by linear interpolation (SLERP for rotations). The path is still $\mathcal{C}^\infty$ in contrast to linear interpolated path. As it can bee seen in Fig 7 the velocities for linear interpolation are constant between waypoints, but for RBF interpolation velocity and acceleration profile are smooth with near zero values at waypoints.

5 Conclusion

In the paper, we propose to use radial basis functions with Gaussian kernels for the generation of smooth Cartesian paths for robot where the motion is defined by a sequence of waypoints through which the robot has to move. We explain how RBF parameters influence the shape of the interpolated path, For additional boundary conditions we augment the set of RBF with additional functions. The performance of the proposed approach is demonstrated by comparison with spline and linear interpolation based methods. The results show that RBF

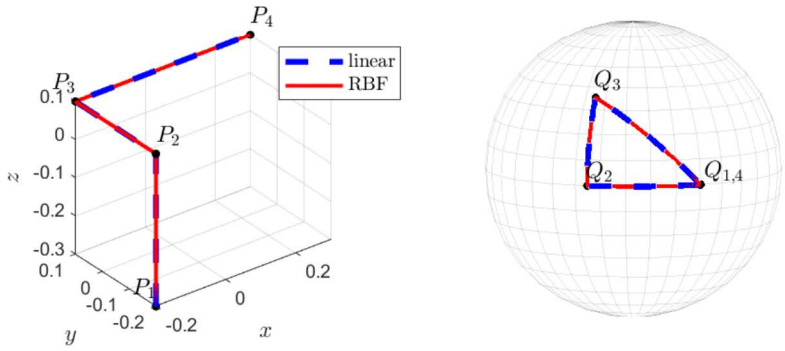

Fig. 6. Comparison of RBF and linear interpolation using waypoints: position in 3D (left), orientation on unit sphere (right). RBF parameters: $\sigma = 0.25$, $k_c = (0.05, 0.1)$ and $\sigma_a = 3\sigma$.

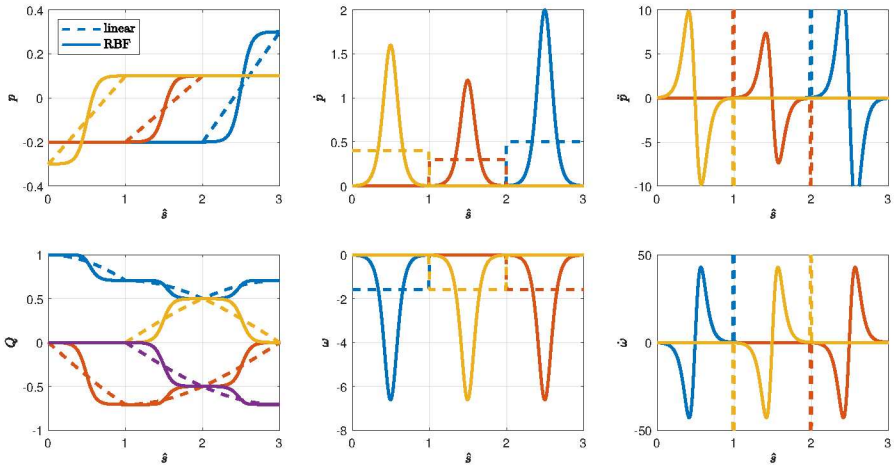

Fig. 7. Comparison of RBF (line) and linear (dashed line) interpolation using waypoints: positions and quaternion components (left), linear and angular velocities (middle), linear and angular accelerations (right). RBF parameters: $\sigma = 0.25$, $k_c = (0.05, 0.1)$ and $\sigma_a = 3\sigma$.

interpolation methods can generate paths similar to spline or linear interpolation with the benefit of $\mathcal{C}^\infty$ continuity and lower computational complexity.

We think that RBF based interpolation can be successfully used in a variety of robot applications. This approach is interesting when the path has to be modified, e.g by kinesthetic guidance. Namely, the RBF parametrization allows easy modifications of the path.

References

1. Chettibi, T.: Smooth point-to-point trajectory planning for robot manipulators by using radial basis functions. Robotica **37**(3), 539–559 (2019)
2. Dam, E.B., Koch, M., Lillholm, M.: Quaternions, Interpolation and Animation. Tech. rep., Department of Computer Science, University of Copenhagen, Technical Report DIKU-TR-98/5 (1998)
3. Froissart, C., Mechler, P.: On-line polynomial path planning in Cartesian space for robot manipulators. Robotica **11**(3), 245–251 (1993)
4. Haarbach, A., Birdal, T., Ilic, S.: Survey of higher order rigid body motion interpolation methods for keyframe animation and continuous-time trajectory estimation. Proceedings - 2018 International Conference on 3D Vision, 3DV 2018 (September), 381–389 (2018)
5. Ijspeert, A., Nakanishi, J., Shibata, T., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots. pp. 219–226. IEEE (2001)
6. Kwon, H., Ahn, K.H., Song, J.B.: Circular Path Based Trajectory Blending Algorithm Considering Time Synchronization of Position and Orientation Trajectories. 2018 15th International Conference on Ubiquitous Robots, UR 2018 pp. 847–851 (2018)
7. Paul, R.: Manipulator Cartesian Path Control. IEEE Transactions on Systems, Man and Cybernetics **9**(11), 702–711 (1979)
8. Sciavicco, L., Siciliano, B.: Modelling and Control of Robot Manipulators. Advanced textbooks in control and signal processing, Springer, London, 2nd edn. (2000)
9. Visioli, A.: Trajectory planning of robot manipulators by using algebraic and trigonometric splines. Robotica **18**(6), 611–631 (2000)
10. Volpe, R.: Task space velocity blending for real-time trajectory generation. In: [1993] Proceedings IEEE International Conference on Robotics and Automation. pp. 680–687. No. 2, IEEE Comput. Soc. Press (1993)
11. Žlajpah, L., Petrič, T.: Unified Virtual Guides Framework for Path Tracking Tasks. Robotica pp. 1–17 (2019)
12. Williams, R.L.: Improved robotics joint-space trajectory generation with via point. Proceedings of the ASME Design Engineering Technical Conference **6**(PARTS A AND B), 669–676 (2011)