Offline approach towards obstacle avoidance in collaborative robots

Carlos Wilfrido Ponce Quiroga, T Raharijaona, Gabriel Abba

Offline approach towards obstacle avoidance in collaborative robots

C.W. PONCE-QUIROGA[a], T. RAHARIJAONA[a] and G. ABBA[a]

a. Université de Lorraine, Arts et Métiers, HESAM Université, F-57070 Metz, France
e-mail: {carlos-wilfrido.ponce-quiroga, thibaut.raharijaona, gabriel.abba}@univ-lorraine.fr

Abstract:

Performance improvement in Human-Robot Collaboration is one of the objectives in collaborative robots. This improvement can be achieved through the correct obstacle avoidance planification of collision-free paths, focusing on terms of path continuity, final motion times and average speed of the end-effector. We propose an offline path planning approach considering an obstacle given an initial linear trajectory from pose A to pose B. By using ROS, MoveIt! and Rviz we generate three trajectories varying the sagitta of our equations for evaluation and determining the best performing trajectory. Results in simulations are compared with the actual motions using the UR10e cobot.

Keywords: obstacle avoidance, collaborative robots, collision-free path planning, ROS, MoveIt!, Dubins' curves variation

1 Introduction

The combination of the human capabilities and dexterity in conjunction with the machine efficiency has been a dream for the past several decades. Conventional automation technology has arrived at a bottle-neck, where this collaboration is the potential solution for the industrial dilemma which improvement can be achieved through the correct obstacle avoidance planification of collision-free paths, focusing on terms of path continuity, final motion times and average speed of the end-effector.

The general collaborative robot obstacle avoidance path planning refers to plan a feasible path of the manipulator end-effector, from a starting pose towards a target pose, while avoiding collision with any obstacle during this operation.

Human decision-making capabilities combined with the robots' precision and efficiency, may indeed enable a remarkable level of performance, however, this collaboration must into consideration the security for human beings [1], [2].

In the past decade, collaborative robots are used for many industrial automation applications. Because of that, the standard ISO/TS15066:2016 was submitted into revision and confirmed its validity in 2019 [3]: it announces the specifications for the safety requirements of collaborative robots and the workspace environment.

Different collision avoidance strategies to assure the safety of the operators. In [4], an emergency stop is activated when an operator entered a hazard zone. However, productivity can be severely reduced due to potential frequent stops.

In [5], four control solutions are proposed for collision avoidance: i) speed reduction and warning, ii) calling a stop interruption, iii) moving away from the oncoming operator, and iiii) changing the path to the target at run-time.

The level of difficulty to resolve the obstacle avoidance problem depends on different factors such as the environment, the type of robot, the task to execute, in order for the robot to move safely around the obstacles [6]. This planning is performed in two different cases: i) off-line planning, where a complete description of the workspace is known in advance (this is a global approach, finding the solutions before implementing them); and ii) on-line planning, where the robot makes decisions given the current state of the environment and moves toward the target, however, the path found may not correspond to the most efficient or optimal one.
The problematic addressed here is to find suitable trajectories for obstacle avoidance in collaborative robots, such that have good performances in terms of overall speed along the trajectory and total time for them to accomplish the task while assuring path smoothness.

2 Path planning

Path planning is a subproblem of the general motion planning problem that consists in finding collision-free paths between an initial and final configuration. While trajectory planning is an additional step where the series of robot configurations are specified as a function of time. The trajectory planning should as well take into consideration constraints such as limits on joint velocities, accelerations or torques [7].

2.1 Path continuity

Continuity notion is fundamental in the definition of a path. The continuity problem refers to the geometric continuity of a path in terms of curvature continuity and tangential continuity. There are two types of continuities [8]: 1) Geometric continuity and 2) Parametric continuity.
Geometric continuity (G^i) guarantees that the endpoints of n segments of the path meet, and tangent vector's directions are equal. Parametric continuity (C^i) guarantees that the endpoints of n segments of the path meet, and that both, the tangent vector's direction and magnitudes are equal.

As cited by Ravankar [8], parametric continuity mean smoothness both of the curve and of its parameterization. Geometric continuity simply means the smoothness of the track that the robot traverses.

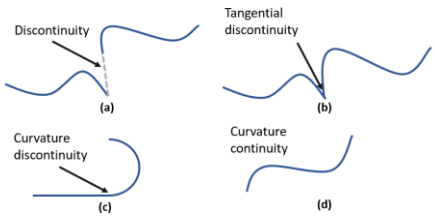

Figure 2 Parametric continuity paths. (a) Discontinuous segments. (b) C^0 continuity. (c) C^1 continuity. (d) C^2 continuity.

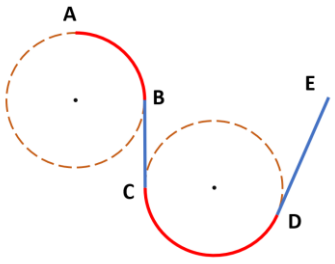

Figure 2 Sample path using principle of Dubins curves

For example, C^1 continuity means continuity of the tangent vector, while G^1 continuity means continuity of slope; C^2 continuity means continuity of the acceleration vector, while G^2 continuity means continuity of the curvature. In terms of robot motion, the C^1 continuous motion preserves velocity, whereas C^2 continuous path preserves acceleration (see Figure 2.1). For robot path planning, what matters is mainly the path's C^1 or C^2 continuity.

Dubins curves [9] were originally proposed in 1957 to describe a continuously differentiable path (C^1continuity type). The initial interest was to obtain a path of minimal length for a given particle. Since then, these types of curves have been used in applications in mobile robotics and UAV's for obstacle avoidance purposes [10]–[12].

Figure 2.2 depicts a sample smoothing path using Dubins curves, which is composed by the combination of circular arcs and straight segments. Points B, C and D are points that represent C^1 continuous motion, by having the same tangent vector of motion (magnitude and direction).

3 Arcs path planning

The approach proposed in this work consists in the use of a variation of the Dubins curve. Dubins curves applications mainly consist of the computation of the smallest path length, constrained by the maximum curvature possible by the mobile system (which can be a negligible restriction for manipulators robots). For manipulators there is not restriction of specific maximum curvature of a path, therefore the final path is composed by several arc with different arc radiuses.

To simplify the calculation, we will consider a single particle obstacle as first approach.

Considering a plane XZ with origin in O as in Figure 3.1, we can compute these equations given four main parameters: h, r, R2, R3 (equations are shown in Table 3.1).

h: represents the sagitta of the main arc at the center with limits [0, r]

r: represents the minimal distance from O when the avoidance starts or ends

R_2: represents the radius of arc_2, first arc most proximate to point A

R_3: represents the radius of arc_3, first arc most proximate to point B

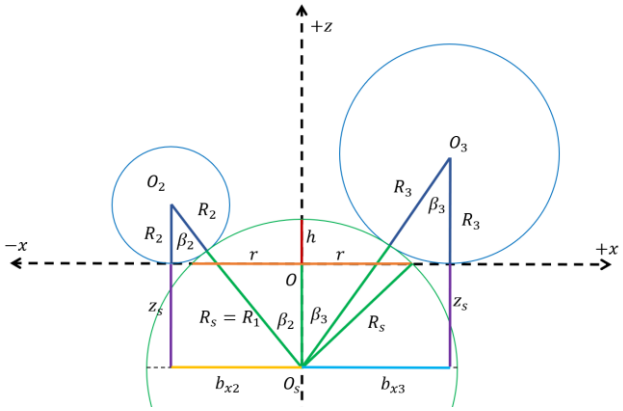

Figure 3.1 Geometric diagram for equations computation

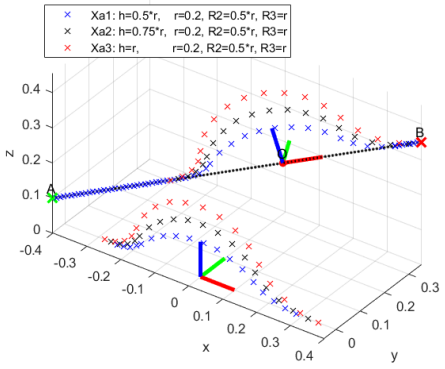

Figure 3.2 Three avoiding trajectories in 2D and in 3D

Table 3.1 Equations used to compute the arcs path

Arc 2	Arc 1 (green)	Arc3
$b_{x2} = \sqrt{(R_2 + R_s)^2 - (R_2 + z_s)^2}$	$R_s = \frac{1}{2h}(h^2 + r^2)$	$b_{x3} = \sqrt{(R_3 + R_s)^2 - (R_3 + z_s)^2}$
$\beta_2 = \cos^{-}(\frac{R_2 + z_s}{R_2 + R_s})$		$\beta_3 = \cos^{-}(\frac{R_3 + z_s}{R_3 + R_s})$
$arc_2: [-\frac{\pi}{2}, -\frac{\pi}{2} + \beta_2]$	$arc_1: [\frac{\pi}{2} + \beta_2, \frac{\pi}{2} - \beta_3]$	$arc_3: [-\frac{\pi}{2} - \beta_3, -\frac{\pi}{2}]$

To pass from the 2D case to a 3D case, we take a finite n series of waypoints from the 2D and then each one of the points is rotated towards the new coordinate system orientation, finally the waypoints are translated to the 3D case scenario (see Figure 3.2)
To achieve these, a series of equations and functions was developed in MATLAB, by computing quaternions, cross products, dot products, etc.

4 Cartesian trajectory generation

4.1 ROS, MoveIt! and Rviz

ROS is an open source robot operating system [13]. It provides a structured communication layer that allows to send and to receive messages above the host operating systems. Many researchers all around the world have used this and it is up today one of the most popular frameworks in robotics.
MoveIt! is a motion planning framework that works alongside ROS [14]. It has been integrated successfully with many robots, including the UR family robots. It uses OMPL motion planning plugins, OROCOS Kinematics and Dynamics Library (KDL) for inverse and forward kinematics, Fast Collision Library (FCL) for collision detection.
Rviz is a three-dimensional visualizer integrated in the core of ROS, used to visualize robots, the environment and sensor data. It can be used for simulations and real-time visualization when on-line with the robot. It is configurable and may integrate plugins: MoveIt! for example.
Universal Robots ROS Driver is open-source set of ROS libraries that provides a stable interface between the UR robots and ROS [15]. It uses the Real-Time Data Exchange (RTDE) and it allows factory calibration of the robot inside ROS. It provides all the necessary configuration files for a transparent interaction and to integrate the use of the MoveIt! framework.

4.2 Nodes and modules interaction

The main node of MoveIt is called *move_group*. This node uses all the individual elements together to provide the interface of a set of ROS services and actions. This node can be configured through the ROS Param Server, getting as well the URDF and SRDF configuration files that describe the environment and the robot itself.
Figure 4.1 depicts the basic architecture and interaction between the principal nodes when using MoveIt with ROS, in addition to our Obstacle Avoidance Planner.

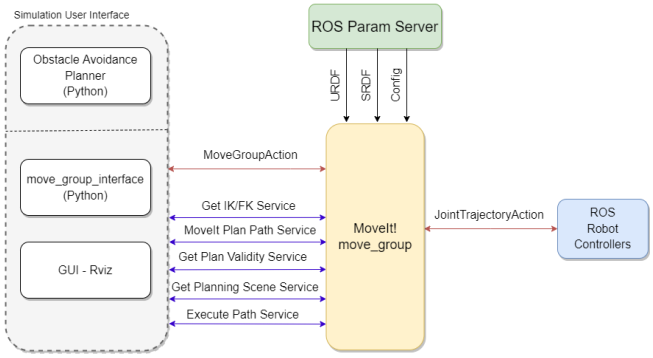

Figure 4.1 Basic architecture with ROS-MoveIt! + Obstacle Avoidance Planner

The *move_group* node interacts with the robot to get current state information (joint positions, etc.) and to talk to the robot controllers through the *JointTrajectoryAction* interface.
The Obstacle Avoidance Planner with its configurable parameters its done initially. The waypoints information of our precomputed path is given to the *move_group_interface* in order to validate the

proposed path. This is achieved through the execution of several services (Inverse Kinematics, Planning Scene, Plan Validity, etc.) to finally determine the corresponding trajectory, considering all the environment and robot constraints present in the configuration files.
The *move_group* can authorize or not the execution of a trajectory. In the real robot is through the *JointTrajectoryAction* services, taking the state information of the robot and communicating through the ROS Robot Controllers; in simulation environment this is performed through the GUI – Rviz interface, by using *fake* controllers, *fake* robot actuators and state positions.

5 Simulations and experimental results

The end-effector is constrained to maintain the same orientation along the different trajectories. The obstacle avoidance motion is to be done above the original linear trajectory (**z**-axis). The proposed poses for validation are the following (position in meters and roll-pitch-yaw in radians):

$$A = \left[0.4, \quad -0.4, \quad 0.25, \quad 0, \quad -\pi, \quad \frac{\pi}{2}\right]^T$$

$$B = \left[0.5, \quad +0.4, \quad 0.46, \quad 0, \quad -\pi, \quad \frac{\pi}{2}\right]^T$$

The main interest is to reduce the final time (total task duration) of the trajectory, while safely avoiding any given obstacle. Three different avoiding trajectories are generated and simulated to compare the final times results and their overall performance in the real robot.
For the path planner algorithm, the $\boldsymbol{y}$ coordinate of the obstacle is introduced as input: $\boldsymbol{O_y}$=$-0.1m$.

We vary the sagitta $\boldsymbol{h}$ by a factor of $\boldsymbol{r}$, to obtain three different obstacle avoidance trajectories:

r = $0.20\ m$, $h = \{1.00r, 0.75r, 0.50r\}\ m$

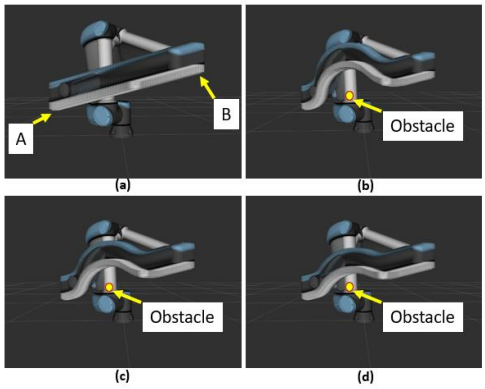

Figure 5.1 UR10e trail trajectories: (a) linear trajectory; obstacle avoidance trajectories (b), (c) and (d)

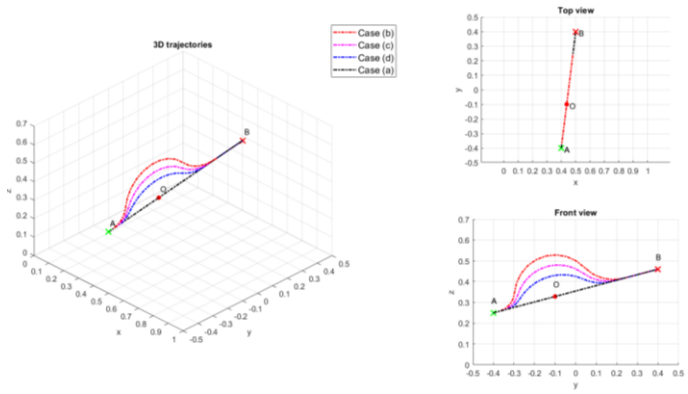

Figure 5.2 Resulting end-effector positions in the Cartesian space

In Figure 5.1 is shown the resulting trail trajectories simulations with the UR10e robot, describing 4 different cases:

(a) Linear trajectory from $\boldsymbol{A}$ to $\boldsymbol{B}$

(b) Obstacle avoidance trajectory with $r = 0.20\ m$, $h = 1.00r$ and $\boldsymbol{O_y} = -0.1\ m$.

(c) Obstacle avoidance trajectory with $r = 0.20\ m$, $h = 0.75r$ and $\boldsymbol{O_y} = -0.1\ m$.

(d) Obstacle avoidance trajectory with $r = 0.20\ m$, $h = 0.50r$ and $\boldsymbol{O_y} = -0.1\ m$.

Figure 5.2 shows the resulting position trajectories in the Cartesian space for each one of the cases previously presented: linear case in black (a), avoidance trajectory cases (b) in red, (c) in magenta and (d) in blue.

Similarly, Table 5.1 presents the total distance traveled by the end-effector, the total time taken to complete the trajectory, and its average speed for each one of the four trajectory cases. Comparing the average speed among the trajectories gives a first insight of the overall performance of the different trajectories. From these values, we can conclude that the case (d) has the best performance between the obstacle avoidance trajectories, in terms of time and average speed; however, case (d) decreased 44.7% (0.158 m/s) in speed with respect from the original linear trajectory (a) (0.285 m/s). In terms of time, case (d) took 91% (5.590 s) longer to finish with respect to the case (a) (2.921 s).
The MoveIt! real results are shown in the right of Table 5.1. It can be seen that the results are quite similar between simulations and real case scenarios. There are just some minor improvements: in terms of time the real obstacle avoidance trajectories perform between 0.01s and 0.05s faster which can be negligible (it represents less than 1% of difference).

Table 5.1 Quantitative assessment of simulation trajectories and robot actual trajectories execution

Trajectory	Total distance (m)	SIM Total time (s)	SIM Average speed (m/s)	MoveIt! real Total time (s)	MoveIt! real Average speed (m/s)
Case (a) Linear	0.833	2.921 [100%]	0.285 [100%]	3.0 [100%]	0.278 [100%]
Case (b) with $h = 1.00r$	0.989	6.806 (+133%)	0.145 (-49%)	6.76 (+125%)	0.146 (-47%)
Case (c) with $h = 0.75r$	0.930	6.223 (+113%)	0.149 (-48%)	6.19 (+106%)	0.150 (-46%)
Case (d) with $h = 0.50r$	0.882	5.590 (+91%)	0.158 (-45%)	5.58 (+86%)	0.158 (-43%)

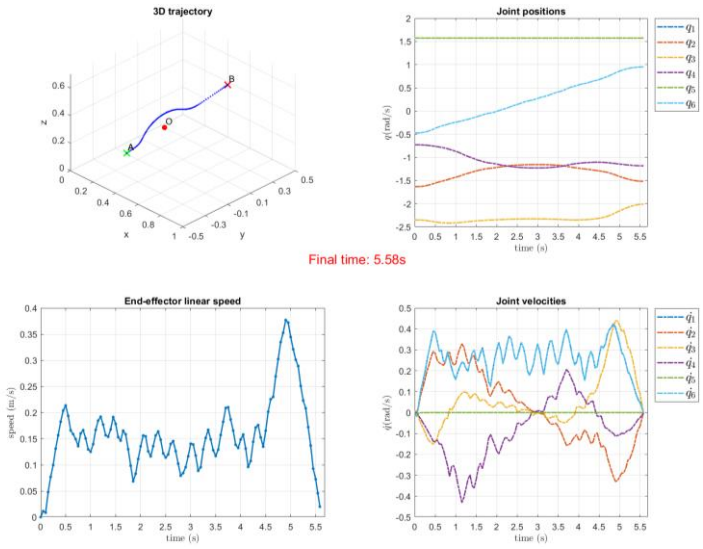

Figure 5.3 MoveIt! real case (d): 3D position trajectory, joint positions, joint velocities, and end-effector linear speed

The best resulting obstacle avoidance trajectory results to be for the case (d), both in simulation and real scenario. The motion performances of all the trajectories are quite similar. Figure 5.3 shows the resulting joint positions, joint velocities and the reflected end-effector linear speed. We can remark that the joint positions present undesired oscillations, which are reflected in the end-effector linear speed. This comes from the Cartesian trajectory generation by using MoveIt!, which lacks from the possibility to directly adjust a desired end-effector speed, which is essential to meet the security requirements of the ISO/TS as it can be seen that it surpasses the 0.25m/s authorized collaboration mode speed.

6 Conclusion and discussion

We successfully generated feasible trajectories with a proposed offline approach that focus on terms of path continuity, final time and average speed of the end-effector.
The collision-free path based on Dubins curve variation, has the advantage of low computation cost as it is uses radius, sagittas, tangents with good continuity in terms of curvature and tangential vectors, resulting in a C^1 continuous path that effectively avoids a given obstacle.
The obstacle particle model allows to examine in detail the performance of the trajectory, the advantage lays in the simplicity, yet scalability, that this approach can allow.
The simulation framework is easily transferred to a real case scenario. The recent development of the ROS framework in conjunction with MoveIt! is of great help to push forward into the robotics research problems, however this may come with some disadvantages.
MoveIt! with ROS and the UR10e, allows to seemingly setup all the necessary tools without much problem for simulation and real implementation, thanks to all the documentation and the robotics community. The main setback of this proposition (when using MoveIt!) is that we can-not control the Cartesian speed of the end-effector trajectory, only the path without any time parametrization.
The results validate the viability of the obstacle avoidance planner, and some key conclusions are extracted, in terms of the expected performances in the real case scenario.
These results were verified and performed under real condition with the UR10e collaborative robot. Further analysis and computations are to be proposed, to address the problematics or disadvantages encountered so far in this scope.

Références (16 gras)

[1] J. Krüger, T. K. Lien, and A. Verl, “Cooperation of human and machines in assembly lines,” *CIRP Annals*, vol. 58, no. 2, pp. 628–646, 2009, doi: 10.1016/j.cirp.2009.09.009.
[2] J. T. C. Tan, F. Duan, Y. Zhang, K. Watanabe, R. Kato, and T. Arai, “Human-robot collaboration in cellular manufacturing: Design and development,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, Oct. 2009, pp. 29–34. doi: 10.1109/IROS.2009.5354155.
[3] ISO/TS, “ISO/TS 15066:2016,” 2016. Accessed: Jan. 07, 2022. [Online]. Available: https://www.iso.org/standard/62996.html
[4] D. Ebert, T. Komuro, A. Namiki, and M. Ishikawa, “Safe human-robot-coexistence: emergency-stop using a high-speed vision-chip,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug. 2005, pp. 2923–2928. doi: 10.1109/IROS.2005.1545242.
[5] B. Schmidt and L. Wang, “Depth camera based collision avoidance via active robot control,” *Journal of Manufacturing Systems*, vol. 33, no. 4, pp. 711–718, Oct. 2014, doi: 10.1016/j.jmsy.2014.04.004.
[6] M. A. C. Gill and A. Y. Zomaya, “A parallel collision-avoidance algorithm for robot manipulators,” *IEEE Concurrency*, vol. 6, no. 1, pp. 68–78, Jan. 1998, doi: 10.1109/4434.656781.
[7] K. M. Lynch and F. C. Park, *Modern robotics: mechanics, planning, and control*. Cambridge, UK: Cambridge University Press, 2017.
[8] A. Ravankar, A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, “Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges,” *Sensors*, vol. 18, no. 9, p. 3170, Sep. 2018, doi: 10.3390/s18093170.
[9] L. E. Dubins, “On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents,” *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957, doi: 10.2307/2372560.

[10] Y. Lin and S. Saripalli, "Path planning using 3D Dubins Curve for Unmanned Aerial Vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 296–304. doi: 10.1109/ICUAS.2014.6842268.
[11] J. Reeds and L. Shepp, "OPTIMAL PATHS FOR A CAR THAT GOES BOTH FORWARDS AND BACKWARDS," 1990, doi: 10.2140/PJM.1990.145.367.
[12] D. Yang, D. Li, and H. Sun, "2D Dubins Path in Environments with Obstacle," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–6, Jan. 2013, doi: 10.1155/2013/291372.
[13] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," Jan. 2009, vol. 3.
[14] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study," *arXiv:1404.3785 [cs]*, Apr. 2014, Accessed: Jan. 08, 2022. [Online]. Available: http://arxiv.org/abs/1404.3785
[15] fmauch, *Universal_Robots_ROS_Driver*. Universal Robots A/S, 2019. Accessed: Jan. 11, 2022. [Online]. Available: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver