Trajectory Optimization for a Collaborative Robot UR5 in a Scenario with Obstacles

Miguel Felipe Nery Vieira [*]
André Gustavo Scolari Conceição. [**]

[*] *Electrical Engineering Program, Federal University of Bahia, BA, (e-mail: miguel.felipe@ufba.br).*
[**] *Electrical Engineering Program, Federal University of Bahia, BA (e-mail: andre.gustavo@ufba.br)*

Abstract: Collaborative robots are becoming more present in various activities, inside and outside the industry. The use of these robots allows greater precision and accuracy in carrying out the tasks. However, it is important to take into account some factors to ensure the safety of the system, such as the ability to avoid obstacles that may be present in the operating environment. In this work, we propose a system for trajectory optimization of a robotic manipulator in complex environments using the algorithms Covariant Hamiltonian Optimization for Motion Planning (CHOMP) and Stochastic Trajectory Optimization for Motion Planning (STOMP), and an RGB+D sensor for obstacle detection. The entire system was implemented based on the open-source framework Robot Operating System (ROS). Performance of the algorithms was analyzed based on their success rate, planning time, and duration of the generated trajectory. Results indicate that the proposed system can generate feasible and collision-free trajectories in static environments.

Keywords: Robots manipulators; Intelligent robotics; Trajectory optimization; ROS; Moveit.

1. INTRODUCTION

Robotic manipulators have been known to the industry since the 1960s. The first one, the Unimate, was developed by George Dovel and it worked on a vehicle production line performing tasks considered dangerous to be performed by human operators (IEEE, 2018). Over the years, several other models of robotic manipulators have emerged, such as the Stanford Arm, developed at Stanford University in 1969 (Stanford InfoLab, 2019), and the ASEA IRB 6, developed by ASEA, today ABB, in 1975 (History Information, 2021).

Nowadays, with the arrival of industry 4.0, a new model of manufacturing it's expected to be created, featuring the collaboration of different systems working connected and autonomously. This is only possible due to the development of technologies, such as additive manufacturing, blockchain, AI (Artificial Intelligence), IoT (Internet of Things), and robotic systems (Olsen and Tomlin, 2019).

Differently of industrial arms, which usually work isolated from humans, in smart factories, arms will be equipped with perception sensors and AI to allow collaboration with humans. The use of collaborative robots, also known as "cobots", allows for more safety, greater reliability, repeatability, and quality in performed tasks (Evjemo et al., 2020). One of the most popular cobots are the UR series robots, Figure 1, from Universal Robots.

⋆ This work was carried out with support from the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Funding Code 001.

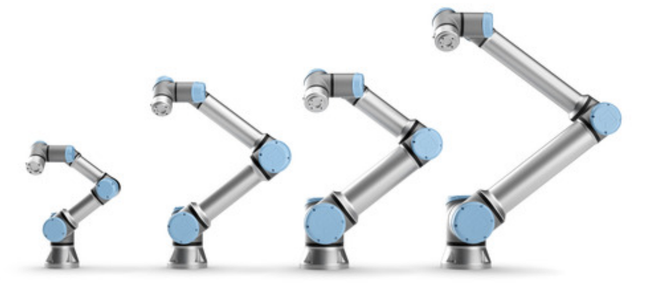


Figure 1. Collaborative UR Series from Universal Robots (from left to right: UR3, UR5, UR10 and UR16) (Universal Robots, 2021).

The manufacturing industry is responsible for most of the applications for cobots, mainly related to pick and place tasks, welding, and other activities that could be dangerous or impossible to be realized by humans. The use of these robots can result in better performance, higher accuracy and reduced operational costs (Sherwani et al., 2020). In these scenarios, it is important to have effective motion planning that guarantees a feasible and smooth path, free from obstacles and that also respects imposed restrictions. This work proposes a system for the optimization of trajectories using the algorithms Stochastic Trajectory Optimization for Motion Planning (STOMP) and Covariant Hamiltonian Optimization for Motion Planning (CHOMP) applied to manipulators inserted in an environment with obstacles, which are detected using an RGB+D sensor.

2. RELATED WORK

Manipulator's motion planning has received increasing attention in the last years, with a large number of works related to this theme being published (Meijer et al., 2017; Pavlichenko and Behnke, 2017; Han et al., 2018; Bormann et al., 2019; Xu and Duguleana, 2019; Yang et al., 2019; Ye and Sun, 2020; Ferrentino et al., 2021) . In this context, planners that use optimization techniques, such as CHOMP and STOMP, can provide more consistent and reliable results, this way gaining popularity among the robotics community (Peng et al., 2021).

Solutions for collision-free operations and optimization of defined cost functions have been developed based on these algorithms, such as in Kaden and Thomas (2019). In this paper, the authors used a combination of STOMP and Gaussian Mixture Models for additional optimization of trajectories generated by the algorithm RRT-Connect. The system was developed in a simulated environment and the results indicated that the proposed method provided a considerable rase of the robot's manipulability. In Pavlichenko and Behnke (2017), an optimization method based on STOMP is presented, the STOMP-NEW. The proposed system considered torque costs, orientation restrictions, obstacles, joint limits, and speed as state costs, been possible the optimization of trajectory duration. Experiments were carried out in a Momaro 7 DoF(Degrees of Freedom) manipulator, and the results indicate the proposed method is applicable in dynamic environments that require frequent replanning .

The use of ROS (Robot Operating System) allows, among a range of benefits, that the simulation of robots can be evaluated with fidelity, reliability, and low cost. In Ye and Sun (2020), the authors used ROS and Moveit for the application of motion planning techniques in a simulated 7 DoF manipulator. A perception sensor was installed on the manipulator, allowing obstacle detection and avoidance. Simulation tests indicated the RRT algorithm is capable of generating obstacle-free trajectories for redundant manipulators (7DoF). A novel system for trajectory planning based on learning by demonstration techniques is proposed by Zhang et al. (2020). The system allows users that are not familiar with programming languages can easily send trajectories to the manipulator. Thanks to the development based on ROS, the system is compatible with different types of robots.

Pick and place of objects is one of the most related application for robotic manipulators. In Jung et al. (2020), a Gazebo simulation of a UR5 manipulator integrated with robotiq's 2F-85 gripper is presented. The authors applied deep learning techniques for object detection and used Moveit for solving the manipulator kinematics and path planning. A system that combines Moveit with the collaborative robot UR5 is also proposed by Kumar et al. (2017). In this work, the authors developed an autonomous system for pick and place tasks in warehouses. Tests were performed with the robot in a real scenario, and the results indicated an average task execution time of 24 seconds and an accuracy of 90% for objects detection.

When talking about collaboration between humans and robots, the robot must have the ability to plan its trajectory free from collisions with obstacles. In Brito et al. (2018), two path planning algorithms, RRT and PRM, were compared in a virtual model of the collaborative manipulator UR5. The proposed system uses a Kinect sensor to perceive the environment. A novel motion planning algorithm based on RRT and Memory-Goal-Biasing is proposed in (Han et al., 2018). The algorithm was implemented on a simulation developed on ROS and Moveit, using the redundant manipulator Baxter robot in different scenarios with obstacles. Results indicated that the proposed method has better optimization performance and lower computation complexity than other RRT-based algorithms.

3. THEORETICAL BASIS

The objective of trajectory optimization is to find a feasible trajectory that minimizes a cost function and respects defined constraints. In this section, optimization algorithms, CHOMP and STOMP will be discussed, as soon as the concepts of ROS framework and Moveit. The computer vision system used in this work and the obstacle collision detection pipeline will also be exhibited, as well as the kinematic model of the collaborative robot UR5.

3.1 CHOMP

CHOMP (Zucker et al., 2013; Ratliff et al., 2009) is a trajectory optimization algorithm that produces *smooth* and *collision-free* trajectories between two specified points q_{init}, q_{goal}. It uses a similar approach to the elastic bands, where the trajectory is repelled from obstacles by forces. However, unlike previous techniques, CHOMP dispenses the requirement that the initial trajectory be collision-free. By iteratively updating the trajectory, the algorithm minimizes the cost functions, which contain obstacles, acceleration and velocity costs. CHOMP uses gradient techniques for optimization, so the cost function must be differentiable.

Given a trajectory $\xi : [0,1] \to \mathcal{C}$ as a function mapping time to robot configuration, the algorithm minimizes an objective functional $\mathcal{U} : \Xi \to \mathbb{R}$ which maps each trajectory ξ in the space of trajectories Ξ to a real number. The objective function is defined as follows

$$\mathcal{U}(\xi) = \mathcal{F}_{obs}(\xi) + \lambda \mathcal{F}_{smooth}(\xi) \tag{1}$$

The term $\mathcal{F}_{smooth}$ penalizes a trajectory based on joint velocities and accelerations and, simultaneously, the term $\mathcal{F}_{obs}$ penalizes proximity to objects in the environment. To improve computational efficacy, the algorithm discretizes the trajectory ξ into a set of n waypoints equally distributed in time q_1,...,q_n, excluding the end points $q_{init} = q_0$ and $q_{goal} = q_{n+1}$, and computes velocities and accelerations via finite differencing.

CHOMP uses a signed distance field as an environment representation, which allows obtaining gradients even for non-collision-free points of the trajectory (Pavlichenko and Behnke, 2017). However, since CHOMP's cost function must be differentiable, as many gradient-based algorithms, it can suffer from local minima problems.

3.2 STOMP

STOMP (Kalakrishnan et al., 2011; Kalakrishnan, 2014) is an algorithm based on CHOMP that also treats the motion planning problem as an optimization problem. It uses a stochastic approach for cost minimization, not requiring that the cost function to be differentiable. The objective of STOMP is to find a smooth, collision-free trajectory that minimizes a predefined cost function that can contain costs related to obstacles and constraints of the robot.

The algorithm takes as input the start and the goal pose of the end effector (x_s, x_g) and outputs a path vector $\theta \in \mathbb{R}^N$ for each joint. It starts the optimization with a fixed duration trajectory, not necessarily feasible, and discretized in n points equally distributed in time. The STOMP trajectory cost function $J(\theta)$ is defined as:

$$J(\theta) = J_x(\theta) + J_u(\theta) \tag{2}$$

The first term in 2, J_x, represents the state-dependent costs. It can contain obstacle costs, constraint violations, and other objectives related to the task accomplishment. Being $q(\theta_t)$ an arbitrary state-dependent cost function at time t, the state costs can be defined as:

$$J_x(\theta) = \sum_{t=1}^{N} q(\theta_t) \tag{3}$$

The second term, $J_u(\theta)$, represents the control costs of the robot and it is quadratic in parameters θ. Being R a positive semi-definite matrix that represents the control costs, chosen in a way that $J_u(\theta)$ represents the sum of squared accelerations along the trajectory, it is defined as:

$$J_u(\theta) = \frac{1}{2}\theta^T R\theta \tag{4}$$

STOMP allows that arbitrary costs can be optimized, even those that are non-differentiable or non-smooth. Being $\tilde{\theta}$ a noisy parameter vector with mean θ and covariance Σ, the algorithm attempts to solve the following optimization problem:

$$\min_{\theta} \mathbb{E}\left[J(\theta)\right] = \min_{\theta} \mathbb{E}\left[\sum_{t=1}^{N} q(\tilde{\theta}_t) + \frac{1}{2}\tilde{\theta}^T R\tilde{\theta}\right] \tag{5}$$

Based on probability matching and path integral reinforcement learning, STOMP estimates the gradient of 5 as the expectation of a noise ϵ in the vector θ under the probability metric $P \propto \exp\left(-\frac{1}{\lambda}J(\tilde{\theta})\right)$. So, the stochastic gradient can be formulated as:

$$\delta\hat{\theta}_G = \int \epsilon \, dP = \int \exp\left(-\frac{1}{\lambda}J\left(\theta + \epsilon\right)\right)\epsilon \, d\epsilon \tag{6}$$

In practice, the gradient is estimated by sampling a finite number of trajectories:

$$\delta\hat{\theta}_G = \sum_{k=1}^{K} P(\theta + \epsilon_k)\epsilon \tag{7}$$

$$P(\theta + \epsilon_k) = \frac{\exp\left(-\frac{1}{\lambda}J(\theta + \epsilon_k)\right)}{\sum_{l=1}^{K}\exp\left(-\frac{1}{\lambda}J(\theta + \epsilon_l)\right)} \tag{8}$$

At every iteration of the algorithm, the gradient update shown in 7 is applied to the original trajectory. The probabilities of each noisy parameter are computed per time-step, as shown in 8. Lastly, the parameter λ is calculated to regulate the sensitivity of the exponential cost.

3.3 Kinematic model of UR5 collaborative robot

A robot manipulator can be described as a series of rigid elements (links) connected by joints, e.g. prismatic joints or revolute joints. These joints allow that links to move with each other and the number of joints determines the manipulator's degrees of freedom.

The kinematic analysis of an n-link robot manipulator in not an easy task and the use of conventions simplifies the analysis. A commonly used convention in robotics applications is the DH (Denavit-Hartenberg) convention. DH convention represents each homogeneous transformation T_i^{i-1} as a product of four basic transformations:

$$T_i^{i-1} = Rot_{z,\theta_i}Trans_{z,d_i}Trans_{x,a_i}Rot_{x,\alpha_i} \tag{9}$$

In 9, θ_i, α_i, a_i and d_i are parameters associated with link i and joint i, also known as DH parameters. A common representation of UR5 robot kinematic structure, with all joint variables (θ_i) at 0, is shown in Figure 2.

The DH parameters for UR5 manipulator are shown in Table 1. These parameters can be used to write 6 transformation matrixes, one for each link, with their formats following 9. The complete equation, from axis 6 to base, can be obtained by multiplication of the 6 transformation matrixes, generating The matrix T_6^0, which represents the homogeneous transformation from the end effector to base.

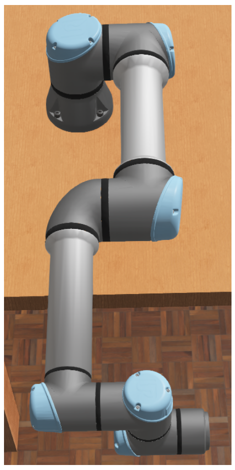
Table 1. DH parameters for UR5 manipulator.

i	θ	α	d	a
1	θ_1^*	$\pi/2$	d_1	0
2	θ_2^*	0	0	a_2
3	θ_3^*	0	0	a_3
4	θ_4^*	$\pi/2$	d_4	0
5	θ_5^*	$-\pi/2$	d_5	0
5	θ_6^*	0	d_6	0

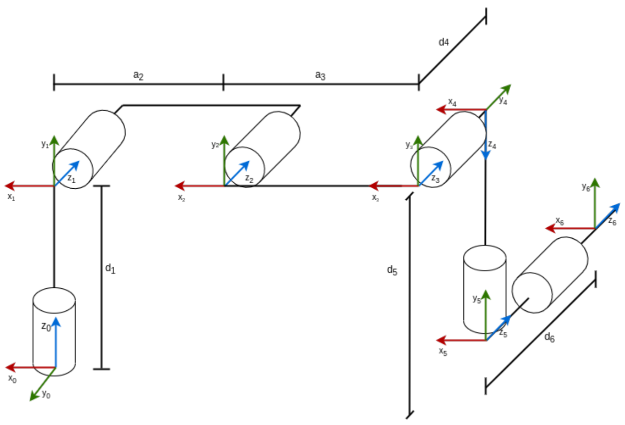
* joint variable

3.4 ROS

ROS is an open-source framework that works between multiple platforms and provides a series of tools and databases for robotics development. It has high compatibility, being used with a wide number of robots, and we can say ROS is one of the most popular robot development platform nowadays (Xu and Duguleana, 2019).



(a) Virtual model



(b) Kinematic chain

Figure 2. UR5 collaborative robot kinematic.

The main goal of ROS is to make the components of a robotic system easier to develop and share, so they can be used on other robots with minimal changes, allowing code reuse and improving code's quality (Mahtani et al., 2016). ROS provides essential functions for robots programming, such as communication among heterogeneous hardware and error treatment and it has been forming an ecosystem that distributes packages made by users (Pyo et al., 2017).

A lot of research institutions and companies have been developing projects in ROS by adding hardware drivers and sharing code samples. In this work, we use ROS packages to calculate UR5 forward and inverse kinematics, get RBG+D sensor data and integrate it into the system, and control the movements of the arm. The simulation of the system was also developed based on ROS packages and using the open source simulator Webots. The figure 3 shows the simulation containing the cobot, a 3D printer, manufactured parts, and obstacles. ROS Melodic Morenia distribution is used.

3.5 Moveit

Moveit is a framework that integrates a set of tools for motion planning and control of robot arms. It supports popular solutions for inverse kinematics, such as KDL, IKFast and TRAC-IK (Beeson and Ames, 2015). It also integrates advanced motion planning algorithms, including

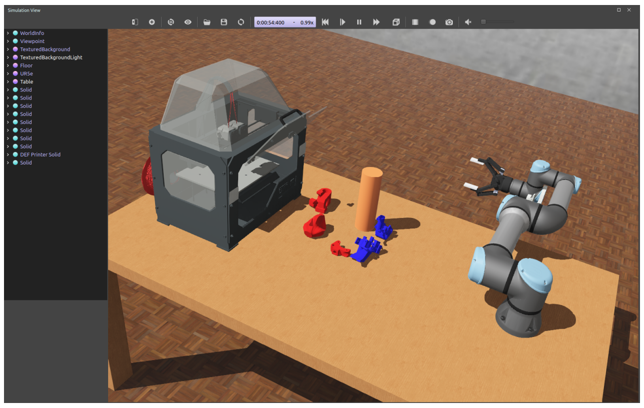


Figure 3. Simulation environment.

OMPL (Sucan et al., 2012), CHOMP and STOMP. The framework combines state-of-the-art algorithms for motion planning, kinematics, control, perception and navigation. Moveit offers a friendly interface for the development of advanced applications, and it has been extensively used with a wide range of robotic manipulators.

The figure 4 shows the `move_group node`, center of Moveit architecture. This node integrates all the individual components to provide ROS actions and services for users (Moveit, 2018). From ROS Param Server, it collects the robot kinematics data, such as URDF, SDRF and configuration files. The SDRF and the configuration files contain the parameters of the manipulator, such as joint limits, kinematics and end effector. The move_group node provides the state and control of the robot through ROS topics and actions, e.g. the `/joint_states` topic and the `JointTrajectoryAction` interface. It also enables an interface to different motion planners that can generate trajectories for desired locations of the end effector respecting constraints such as position, orientation or joint constraints.

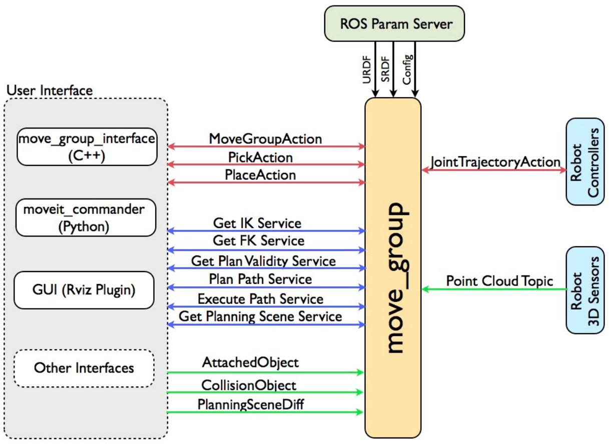


Figure 4. The move_group node.

In this work, Moveit is used to integrate TRAC-IK, for inverse kinematics, with STOMP and CHOMP, for motion planning, allowing it to control a UR5 collaborative manipulator and the Robotiq gripper attached to its end effector, in a pick and place application at an environment with obstacles. Additionally, point Cloud data obtained by an RGB+D sensor is integrated into the framework for obstacle and collision detection.

3.6 Computer Vision

Computer vision is a common topic in robotic research nowadays since vision sensors have become more accessible, and computers are getting smaller and more powerful. In this work, we used a depth camera Intel Realsense D435, mounted on the robot's end effector, to provide point cloud data for obstacle detection. Based on Moveit perception tools, a schematic of the obstacle detection is shown in Figure 5 as it contains the following stages:

(1) The initial state S_i of the robot is stored.
(2) Point cloud data P_i is obtained by RGB+D sensor.
(3) Based on acquired information, a Planning Scene is generated with geometrical representation of the objects in the environment.
(4) If some link of the robot is assumed to be in contact with any object, the mesh in contact is represented in the color red, and the movement is not allowed to be executed. Examples of collision detection with objects (4a) and printer (4b) are shown.

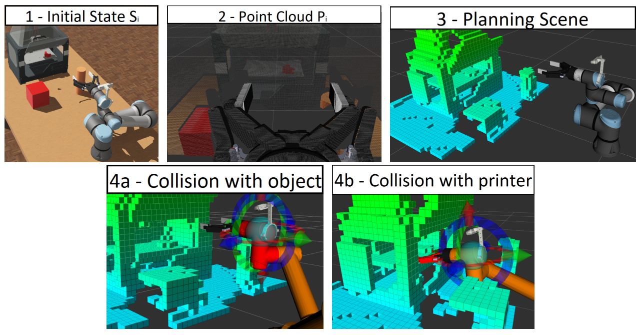


Figure 5. Obstacle collision detection schematic.

4. RESULTS AND DISCUSSION

All the experiments were conducted in an additive manufacturing unit composed of an UR5 robotic manipulator, from Universal Robots, controlled by ROS/Moveit and a 3D printer. The CPU platform used is an Intel® Core™ i7-7050H CPU @ 2.60GHz, 8GB memory, and the GPU platform is NVIDIA® GeForce® GTX 1650 4GB.

In this work, the experiments were performed with the objective of achieving a pick and place task, avoiding collision with static obstacles in the scene. After the execution, the cobot must return to its initial position. First, the experiments were conducted in different simulation environments, varying the amount and positioning of obstacles within the scenes. After the simulated experiments, the system was validated in a real additive manufacturing cell.

4.1 Simulated Experiments

We developed four simulation scenes, each one with different complexity of obstacles, as shown in Figure 6. The execution routine on each scene is similar, in which the cobot must pick a manufactured piece of known location, and place it on the table, while avoiding collisions with obstacles. The execution routine for the simulated scene III is exhibited in Figure 7.

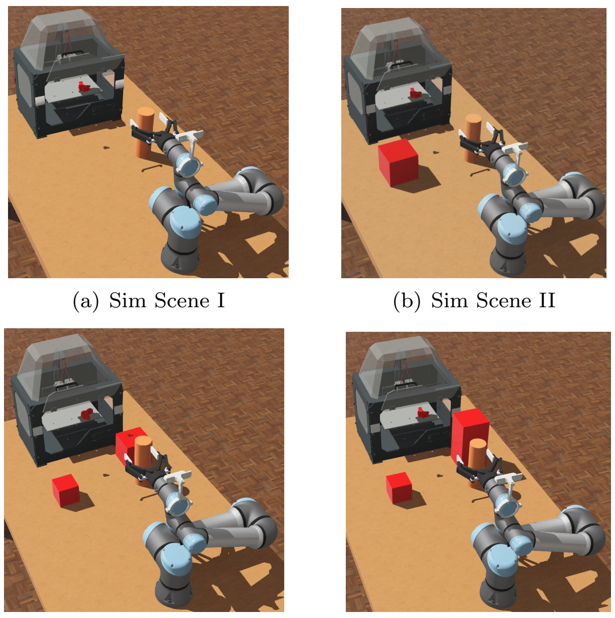

(a) Sim Scene I (b) Sim Scene II
(c) Sim Scene III (d) Sim Scene IV

Figure 6. Simulation scenes in order of obstacles complexity.

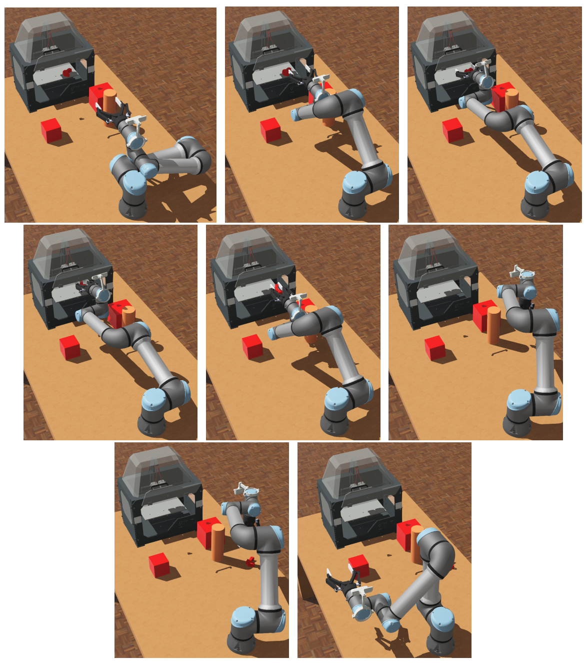


Figure 7. Execution routine for simulated scene III.

We made 10 executions using each algorithm, CHOMP and STOMP, by scene. Additionally, for comparison, we have also executed the routine using the algorithm RRT-Connect, from OMPL, the default planner on Moveit. The metrics used to compare the performance of the algorithms were: Success Rate, Planning time(s), and Trajectory Duration(s). The results for each metric can be seen in Table 2, and Figures 8 and 9.

Table 2. Success rate on simulated scenarios.

	CHOMP	STOMP	RRT-Connect
Success	96.66%	96.66%	93.33%

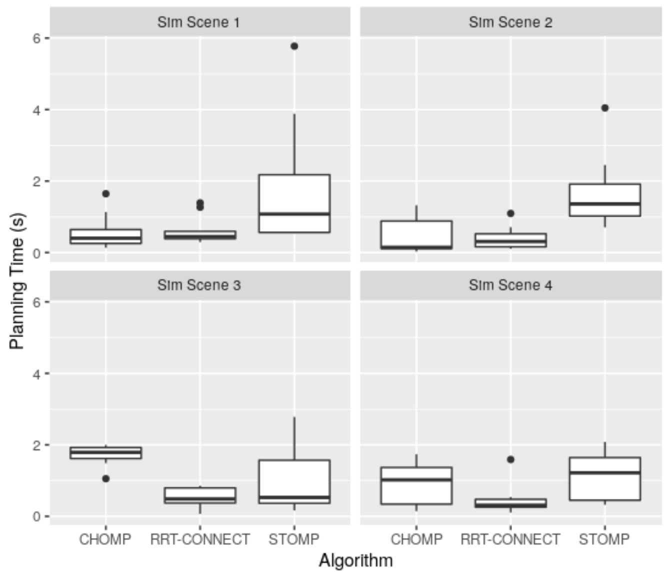


Figure 8. Planning time by scene and algorithm on simulated scenarios.

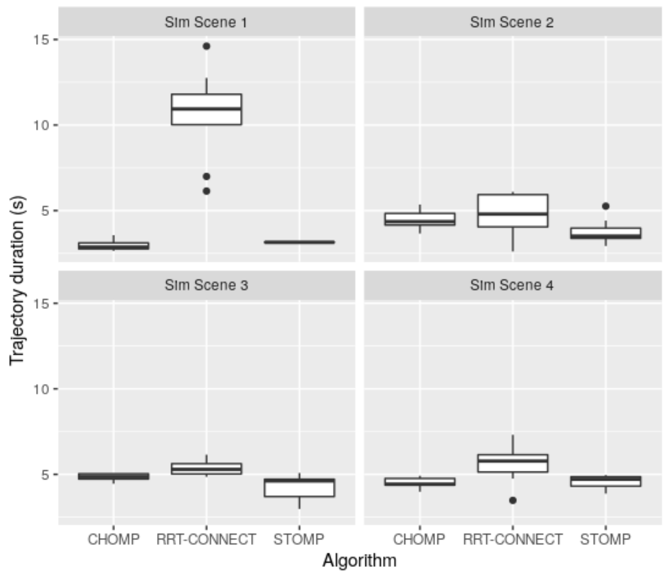


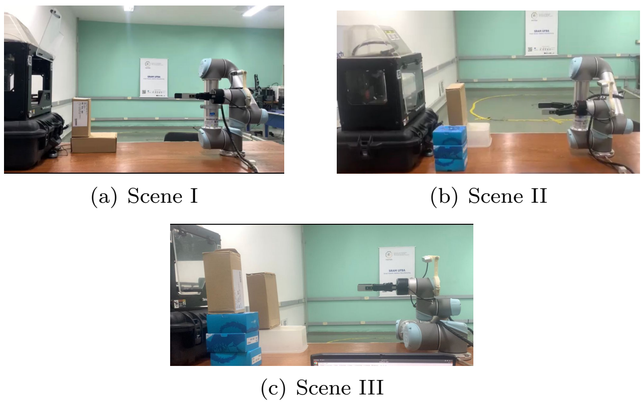
Figure 9. Trajectory duration by scene and algorithm on simulated scenarios.

Results indicate that CHOMP, STOMP, and RRT-Connect have similar success rates in the routine execution, generating feasible trajectories for the collaborative robot. All the algorithms also have similar average planning time, with STOMP presenting some variation in the obtained data, which can be a reflex of its stochastic approach.

When analyzing the duration of the trajectory obtained by each algorithm, we can see that STOMP produces a trajectory of shorter duration than CHOMP and RRT-Connect. It is also notable that, in all simulated scenarios, RRT-Connect generated trajectories of longer duration than the others, consequently increasing the risk of executing trajectories that could cause some type of damage to the manipulator.

4.2 Real Experiments

After the execution and validation of the system in the simulation scenes, tests were carried out in a real environment, on three different scenes, each one with different complexity of obstacles, as shown in Figure 10. The routine execution for the real scenes are similar, in an approach task for a 3D printed part, while avoiding collisions with obstacles in the workspace. Figure 11 shows the execution routine for the real scene III.



(a) Scene I (b) Scene II (c) Scene III

Figure 10. Real scenes in order of obstacles complexity.

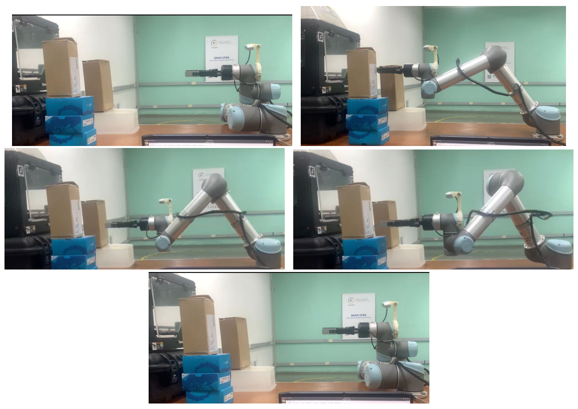


Figure 11. Execution routine for real scene III.

Similarly to simulated scenarios, we made a series of executions using each algorithm, CHOMP and STOMP, by scene. On simulation tests, we observed that RRT-Connect generated trajectories of longer duration, which often led the cobot to some kind of self-collision during the execution. This way, for security, we decided not to use the algorithm on real executions.

We also used the same metrics to compare the performance of the algorithms: Success Rate, Planning time(s), and Trajectory Duration(s). The obtained results for success rate can be seen in Table 3, and we can note they were satisfactory for both algorithms, indicating that they are able to generate feasible trajectories for all the scenarios.

Table 3. Success rate on real scenarios.

	CHOMP	STOMP
Success	93.33%	93.33%

Results on Figures 12 and 13 indicate that STOMP can generate trajectories of shorter duration than CHOMP, in

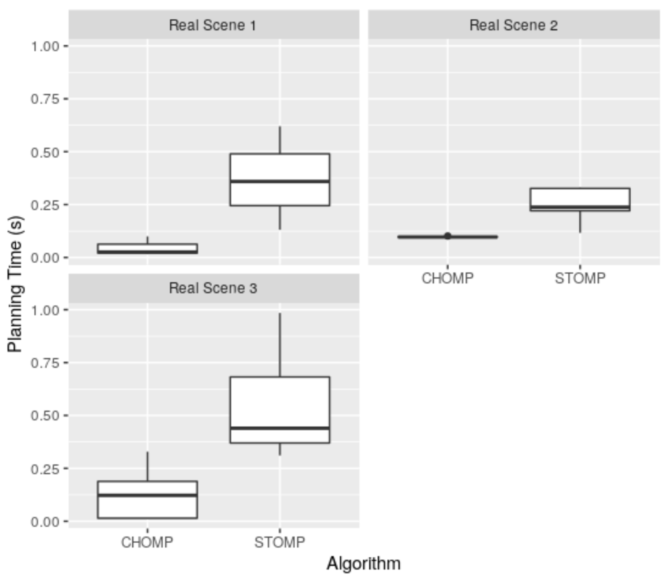


Figure 12. Planning time by scene and algorithm on real scenarios.

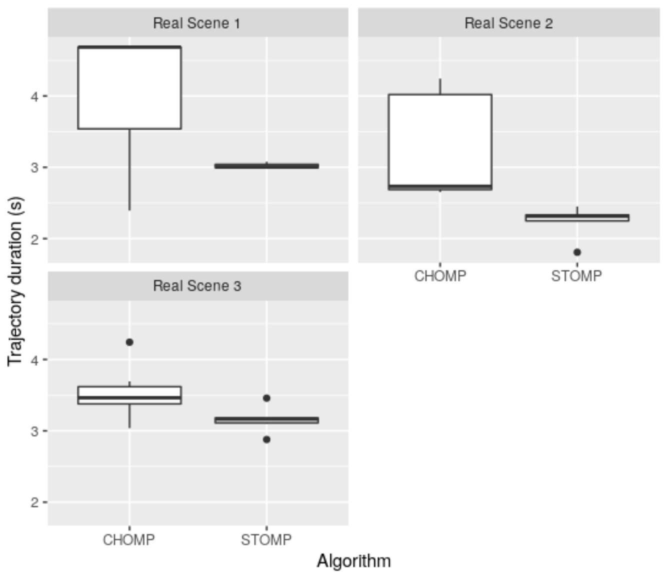


Figure 13. Trajectory duration by scene and algorithm on real scenarios.

a scenario with obstacles, despite needing more planning time. In this way, the trajectories generated by STOMP prevent the cobot from performing unnecessary movements, reducing risks and efforts on its joints. Therefore, the collected data allow us to conclude that STOMP delivers smoother trajectories than the others analyzed algorithms analyzed in this work.

5. CONCLUSION

In this work, we proposed a system for trajectory optimization of a collaborative robot UR5 in a scenario with obstacles, using the algorithms CHOMP and STOMP, and an RGB+D sensor for obstacle detection and collision avoidance.

According to the results, all studied algorithms presented similar and satisfactory performances for planning time and success rate. However, analyzing the duration of the trajectory, the STOMP algorithm showed better results, indicating that it has a greater capacity to deliver smoother trajectories to the robot, avoiding possible collisions with the environment, thus guaranteeing safety to the device and the operator.

Furthermore, due to STOMP stochastic optimization method, several other costs can be optimized, whether or not they are differentiable. Finally, it is important to note that the use of a trajectory planner to generate the initial trajectory to be optimized by STOMP can be an alternative that reduces the algorithm planning time, contributing to better performance.

REFERENCES

Beeson, P. and Ames, B. (2015). Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 928–935. IEEE.

Bormann, R., de Brito, B.F., Lindermayr, J., Omainska, M., and Patel, M. (2019). Towards Automated Order Picking Robots for Warehouses and Retail. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11754 LNCS(November 2020), 185–198. URL `https://dx.doi.org/10.1007/978-3-030-34995-0_18`.

Brito, T., Lima, J., Costa, P., and Piardi, L. (2018). Dynamic Collision Avoidance System for a Manipulator Based on RGB-D Data. *Advances in Intelligent Systems and Computing*, 694, 643–654. doi:10.1007/978-3-319-70836-2_53.

Evjemo, L.D., Gjerstad, T., Grøtli, E.I., and Sziebig, G. (2020). Trends in smart manufacturing: Role of humans and industrial robots in smart factories. *Current Robotics Reports*, 1(2), 35–41.

Ferrentino, E., Salvioli, F., and Chiacchio, P. (2021). Globally optimal redundancy resolution with dynamic programming for robot planning: A ros implementation. *Robotics*, 10(1), 1–23. doi:10.3390/robotics10010042. URL `https://dx.doi.org/10.3390/robotics10010042`.

Han, D., Nie, H., Chen, J., and Chen, M. (2018). Optimal randomized path planning for redundant manipulators based on Memory-Goal-Biasing. *International Journal of Advanced Robotic Systems*, 15(4), 1–11. URL `https://dx.doi.org/10.1177/1729881418787049`.

History Information (2021). Asea produces the irb 6, the first microcomputer controlled electric industrial robot. URL `https://www.historyofinformation.com/detail.php?entryid=4352`.

IEEE (2018). Unimate. URL `http://robots.ieee.org/robots/unimate/`.

Jung, H., Kim, M., Chen, Y., Min, H.G., and Park, T. (2020). Implementation of a unified simulation for robot arm control with object detection based on ROS and Gazebo. *2020 17th International Conference on Ubiquitous Robots, UR 2020*, 368–372. doi:10.1109/UR49135.2020.9144984. URL `https://doi.org/10.1109/UR49135.2020.9144984`.

Kaden, S. and Thomas, U. (2019). Maximizing Robot Manipulability along Paths in Collision-free Motion Planning. In *2019 19th International Conference on Advanced Robotics (ICAR)*, 105–110. IEEE. URL `https://dx.doi.org/10.1109/ICAR46387.2019.8981591`.

Kalakrishnan, M. (2014). *Learning objective functions for autonomous motion generation*. University of Southern California.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. 4569–4574. URL `https://doi.org/10.1109/ICRA.2011.5980280`.

Kumar, S., Majumder, A., Dutta, S., Raja, R., Jotawar, S., Kumar, A., Soni, M., Raju, V., Kundu, O., Behera, E.H.L., Venkatesh, K.S., and Sinha, R. (2017). Design and Development of an automated Robotic Pick & Stow System for an e-Commerce Warehouse. 1–15. URL `http://arxiv.org/abs/1703.02340`.

Mahtani, A., Sanchez, L., Fernández, E., and Martinez, A. (2016). *Effective robotics programming with ROS*. Packt Publishing Ltd.

Meijer, J., Lei, Q., and Wisse, M. (2017). An empirical study of single-query motion planning for grasp execution. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, 1234–1241. doi: 10.1109/AIM.2017.8014187. URL `https://dx.doi.org/10.1109/AIM.2017.8014187`.

Moveit (2018). Concepts. URL `http://moveit.ros.org/documentation/concepts/`.

Olsen, T.L. and Tomlin, B. (2019). Industry 4.0: Opportunities and challenges for operations management. *Available at SSRN: https://ssrn.com/abstract=3365733*. doi: http://dx.doi.org/10.2139/ssrn.3365733.

Pavlichenko, D. and Behnke, S. (2017). Efficient stochastic multicriteria arm trajectory optimization. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe(September), 4018–4025. URL `https://dx.doi.org/10.1109/IROS.2017.8206256`.

Peng, Y.C., Chen, S., Jivani, D., Wason, J., Lawler, W., Saunders, G., Radke, R.J., Trinkle, J., Nath, S., and Wen, J.T. (2021). Sensor-guided assembly of segmented structures with industrial robots. *Applied Sciences (Switzerland)*, 11(6). doi:10.3390/app11062669.

Pyo, Y., Cho, H., Jung, R., and Lim, T. (2017). *ROS Robot Programming*. Robotis.

Ratliff, N., Zucker, M., Bagnell, J.A., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, 489–494. IEEE. URL `https://doi.org/10.1109/ROBOT.2009.5152817`.

Sherwani, F., Asad, M.M., and Ibrahim, B. (2020). Collaborative robots and industrial revolution 4.0 (ir 4.0). In *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, 1–5. IEEE.

Stanford InfoLab (2019). Robots and their arms. URL `http://infolab.stanford.edu/pub/voy/museum/pictures/display/1-Robot.htm`.

Sucan, I.A., Moll, M., and Kavraki, L.E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4), 72–82.

Universal Robots (2021). Collaborative robots from universal robots. URL `https://www.universal-robots.com/products/`.

Xu, X. and Duguleana, M. (2019). Trajectory Planning of 7-Degree-of-Freedom Manipulator Based on ROS. *IOP Conference Series: Materials Science and Engineering*, 677(5). URL `https://dx.doi.org/10.1088/1757-899X/677/5/052072`.

Yang, Y., Merkt, W., Ivan, V., and Vijayakumar, S. (2019). Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints. *IEEE-RAS International Conference on Humanoid Robots*, 2018-Novem, 893–900. URL `https://dx.doi.org/10.1109/HUMANOIDS.2018.8624989`.

Ye, L. and Sun, C. (2020). Trajectory planning of 7-DOF redundant manipulator based on ROS platform. *Proceedings of 2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence, ICIBA 2020*, (Iciba), 733–736. doi: 10.1109/ICIBA50161.2020.9277001. URL `https://dx.doi.org/10.1109/ICIBA50161.2020.9277001`.

Zhang, H.D., Liu, S.B., Lei, Q.J., He, Y., Yang, Y., and Bai, Y. (2020). Robot programming by demonstration: a novel system for robot trajectory programming based on robot operating system. *Advances in Manufacturing*, 8(2), 216–229. URL `https://doi.org/10.1007/s40436-020-00303-4`.

Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., and Srinivasa, S.S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10), 1164–1193.