Offloading of Time-Optimal Motion Planning with Jerk Constraints

Ahmed Al Bayati*, Philip Olhager†, Björn Olofsson* Karl-Erik Årzen*
*Dept. of Automatic Control, Lund University, Sweden
†Cognibotics AB, Lund, Sweden

***Abstract*—In this paper offloading an advanced motion planning pipeline consisting of a path planner and a trajectory planner is investigated. The path planner generates an obstacle-free geometric path, while the trajectory planner computes time-optimal trajectories which respect velocity, acceleration, and jerk constraints for path tracking. The complete motion planner is integrated into an offloading framework, deployed on an edge cluster, and communicates with a robot via 5G to evaluate its performance on a physical system.**

I. Introduction

Robots are becoming more popular in today's society including anything from robotic vacuum cleaners to welding robots on manufacturing floors. The robots usually execute algorithms deployed on an industrial PC (IPC) which they are connected to. These IPCs often have limited computational power, restricting the complexity of algorithms they can run onboard [1]. One solution for enabling the robot to utilize complex algorithms is to offload the computationally demanding algorithms to the cloud or nearby edge devices. This enables the robot to perform tasks beyond its local processing capabilities by leveraging compute somewhere else on the network. The meaning of edge used here is a server or cluster of servers located in close vicinity to a base station [2], [3]. This is sometimes referred to as a cloudlet.

Motion planning could be used to demonstrate how leveraging more computational power can results in a performance improvement. A motion planner could either be made very simplistic and require little computational power to run at the cost of performance and/or robustness. Alternatively, it could be made more advanced allowing it to handle more constraints and/or to be more robust in a more dynamic workspace.

Collision free, time-optimal motion planning with jerk constraints is an example of a planner that requires computational power to run but in turn improves the robot's speed, durability and work range. In this paper, we have implemented an advanced motion planner capable of computing collision free, time-optimal, jerk-constrained trajectories to transport an object between two points as fast as possible. This algorithm was then used as an offloaded motion planning algorithm for the robot to use without straining its industrial PC (IPC).

Offloading motion-planning algorithms introduces two key challenges: the delay in executing control tasks and the risk of not receiving a response in time. Several factors contribute to this risk. Optimization-based algorithms may fail to find a solution, communication links can drop packets, or the offloading target may reject requests due to overload. Deploying the algorithm closer to the robot at an edge server mitigates some of these challenges. The transmission time gets reduced and packet delivery success increases since the number of hops between the robot and the edge are fewer compared to cloud-based offloading, [3]. However, the risk of missing responses remains, necessitating a fallback algorithm on the robot to maintain workflow or to function as an emergency fallback to gracefully move the robot to a safe location until the advanced motion planner is operational again.

For mobile robots offloading, wireless communication between the robot and the cloud/edge is crucial. Making cellular 5G a promising enabler thanks to its high data rates, low latency, and reliable connectivity [4], [5].

This work was supported by: The Vinnova AORTA project (Dnr 2022-03039), the ELLIIT Strategic Research Area, and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

II. Motion Planner

An advanced motion planner capable of computing collision free, time-optimal, jerk-constrained trajectories to transport an object between two points as fast as possible is implemented.

The advanced motion planner comprises a path planner and a trajectory planner. The path planner generates a collision-free geometric path, and the trajectory planner computes time-optimal trajectories that adhere to velocity, acceleration, and jerk constraints.

The path planning is performed using the Open Motion Planning Library (OMPL) [6]. In this study, the C-Forest planner [7], a parallel framework designed to accelerate single-query, sampling-based shortest path planning, is utilized. The C-Forest planner operates by simultaneously growing multiple RRT* trees, [8], to efficiently find a shorter path. The output of the path planner is a set of points that are interpolated to create a path as illustrated in Fig. 1

The trajectory planner generates time-optimal trajectories for the robot to follow the geometric path. The robot's dynamics are approximated as a kinematic double integrator model, and the trajectory planning problem is formulated as an optimization problem that minimizes the total traversal time while satisfying constraints on joint velocity, acceleration, and jerk. This optimization problem is solved using CasADi, [9], and IPOPT, [10]. An example of the generated trajectories alongside their respective limits can be seen in Fig. 2.

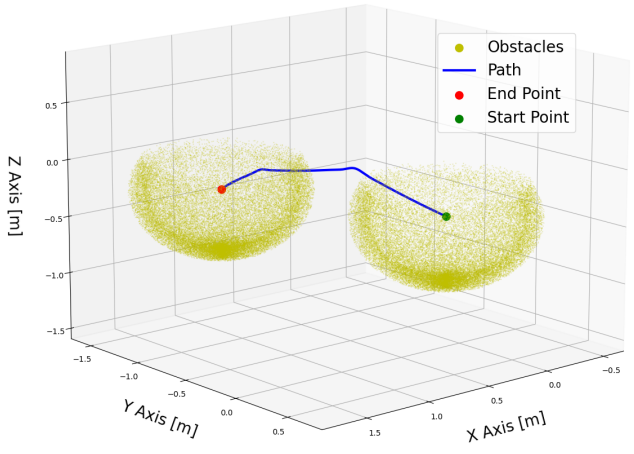

Fig. 1. Given a start and end configuration and obstacles, the advanced path planner computes a geometric path that connects the start point to the end point while avoiding the obstacles. In this figure the obstacles are represented as a point cloud to easily show the path, start and end points.

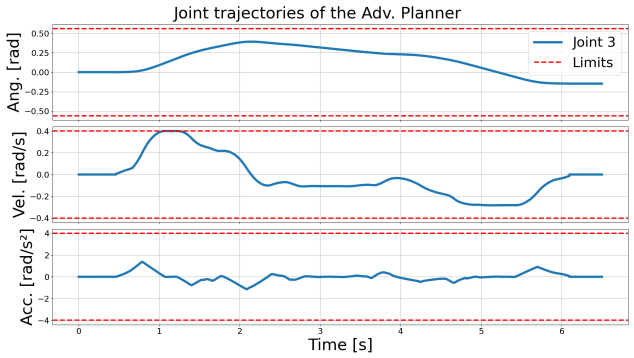

Fig. 2. An example trajectory for a robot joint, joint 3 is shown, where the first plot illustrates the joint angle over time, the second plot shows the velocity, and the third one showcases the acceleration. Along with the trajectories, the limits of joint angle, velocity, and acceleration are plotted.

III. Offloading

The advanced motion planning algorithm requires resources such as a fast processor, sufficient RAM, and efficient multi-threaded execution to run efficiently. Since the robot has limited computational power, the advanced motion planner is deployed on a platform with sufficient resources to support it. Because the robot depends on the motion planner to perform its movements, it is desirable to have computational resources close to the robot to minimize delays and packet loss. This makes edge computing an attractive deployment option instead of the cloud due to its proximity to the robot.

In this setup, the edge server is positioned at the edge of a network, specifically a 5G network, acting as a local computational resource for offloading [2], [3]. To enhance mobility and demonstrate feasibility, communication between the edge and the robot is established using a private 5G network provided by Ericsson.

However, offloading comes with risks, such as network failures, optimization issues, or edge server unavailability. To mitigate these risks, the robot has a local offloading agent, a logical software unit, that determines whether the motion planning should be offloaded or not based on the task at hand. If no response is received from the advanced motion planner, the agent also deploys the fallback local heuristic motion planner. The overall architecture is presented in Fig. 3 to illustrates the idea.

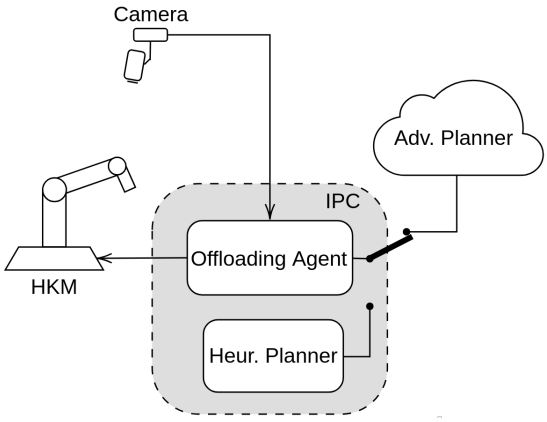

Fig. 3. When the IPC receives a pick-and-place command along with a list of obstacles, the offloading agent evaluates the pick-and-place task and delegates the motion planning to either the local heuristic or remote advanced motion planner. The local planner also serves as a fallback if the remote planner fails to provide trajectories on time.

References

[1] K. E. Chen, Y. Liang, N. Jha, J. Ichnowski, M. Danielczuk, J. Gonzalez, J. Kubiatowicz, and K. Goldberg, "Fogros: An adaptive framework for automating fog robotics deployment," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 2035–2042, 2021.

[2] Akamai Technologies, "What is edge computing?" 2025, accessed: 2025-03-30. [Online]. Available: https://www.akamai.com/glossary/what-is-edge-computing

[3] Ericsson, "Ericsson - edge computing," 2023, accessed: 2025-03-31. [Online]. Available: https://www.ericsson.com/en/edge-computing

[4] Ericsson, "Ericsson - 5g," 2023, accessed: 2025-03-31. [Online]. Available: https://www.ericsson.com/en/5g

[5] Inseego, "What is urllc?" 2023, accessed on March 31, 2025. [Online]. Available: https://inseego.com/resources/5g-glossary/what-is-urllc/

[6] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[7] M. Otte and N. Correll, "C-forest: Parallel shortest path planning with superlinear speedup," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 798–806, 2013.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, DOI 10.1177/0278364911406761, no. 7, pp. 846–894, 2011.

[9] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[10] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.