Acta Mechanica Slovaca **29 (3): 54 - 60,** September **2025**
https://doi.org/10.21496/ams.2025.023

Acta Mechanica Slovaca
ISSN 1335-2393
www.actamechanica.sk

Path Optimization of a Mobile Robot Platform with a Robotic Arm

Patrik Pilát [1],*, Jozef Varga [1], Ján Semjon [2], Matúš Sabol [2]

[1] Prototyping and Innovation centre, Faculty of Mechanical Engineering, Technical University of Košice, Faculty of Mechanical Engineering, Slovakia
[2] Department of Production Technology and Robotics, Technical University of Košice, Faculty of Mechanical Engineering, Slovakia

Abstract: This paper presents the development and simulation of a path optimization approach for a mobile assistive service robot equipped with a robotic manipulator arm to operate in hot gas chamber. The proposed control architecture integrates Model Predictive Control (MPC) with Jacobian-based Inverse Kinematics (IK) for enabling smooth, adaptive motion planning even under dynamically changing environmental conditions. A Matlab based simulation setup was used to verify the control approach using random disturbances to simulate real-world complications like payload mass variations, centre of gravity shifts, and obstacle interference. Results show that MPC adapts trajectories in real time. However, actuator constraints and very sudden changes in the environment could lead to increased deviations. Future improvements include refining the MPC cost function, introducing adaptive prediction horizons, and employing trajectory filtering to improve robustness. The presented approach forms a basis for future work on physical robots, with plans for implementation in the Webots simulation environment for digital twin creation and validation of semi-autonomous control.

Keywords: inverse kinematic; model predictive control; mobile service robot

1. Introduction

Mobile service robots are frequently employed in unstructured, uncertain, and incontinently continuously varying environments, e.g., homes, public indoor spaces, or warehouses [1],[2]. These are environments with limited knowledge, in which robots need to operate without complete maps while reacting to unpredictable objects and dynamic conditions like human individuals or mobile machinery [3],[4],[5]. To support such movements, robots must localize themselves in real-time and modify their motion according to immediate sensor feedback from LiDARs or cameras [6]. Moreover, moving through environments also requires precision but also strong obstacle avoidance and smooth adjustment of trajectory [1],[4],[7].

In that regard, Model Predictive Control (MPC) has been a powerful and generic framework for autonomous robot navigation in unstructured and dynamic environments. The inherent power of MPC is to predict the future trajectory of the robot's state over a time horizon from a mathematical model and optimize control action accordingly with velocity, acceleration, and workspace constraint [8],[9]. Compared to traditional controllers, MPC is more sensitive and flexible toward environmental variation, and it is therefore suitable for real-time navigation in both indoor and outdoor settings [10].

MPC is currently widely used in robotics, process control, and autonomous vehicles due to its ability to be flexible and employ dynamic models and constraints as a component of the control logic. Its strength comes from its capability to adapt trajectories dynamically based on predictions of the future [11],[12].

* **Corresponding author:** Patrik Pilát, ***E-mail address:*** *patrik.pilat@tuke.sk*

Another section discussed in this paper is the kinematics of the robotic arm. Kinematics is the study of motion without the forces responsible for it. Forward kinematics in robotics is the calculation of the position of the end-effector through known joint variables — rotations or displacements, for example. Conversely, inverse kinematics attempts to come up with the joint settings necessary for a desired position or orientation of the end-effector. While forward kinematics is straightforward algebraic mapping, inverse kinematics usually requires more effort due to nonlinearities, redundant solutions, and limited workspaces [13]. Inverse Kinematics (IK) refers to the mathematical process of solving joint angles that put a robot's end-effector at a desired coordinate [14],[15].

2. Case Study

This MPC form will be used for custom-built assistive service robot (Fig.2) control. The robot is a special-purpose equipment that is designed to assist during the decontamination process of a hot gas chamber (Fig.1). This service robot was developed specifically due to the extreme inaccessibility of the chamber and radiation levels in the environment. Consequently, the decontamination activities must be performed using remotely operated robotic equipment in semi-autonomous mode but with the possibility of operator intervention from remote locations if necessary.

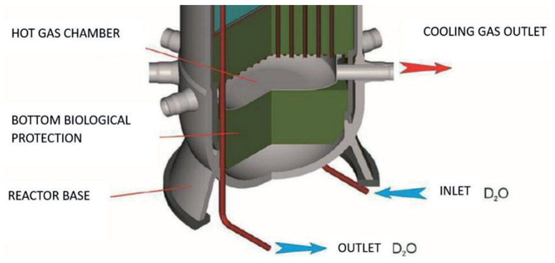

Figure 1: Hot gas chamber [16]

The primary functions of this mobile robot will include:

– *navigating through pipelines with an internal diameter of 380 mm,*

– *inserting and extracting the decontamination robot from the hot gas chamber using a robotic manipulator arm,*

– *retrieving a sediment-filled container from the decontamination robot, performing emptying operations, and reinserting the emptied container back into the decontamination robot.*

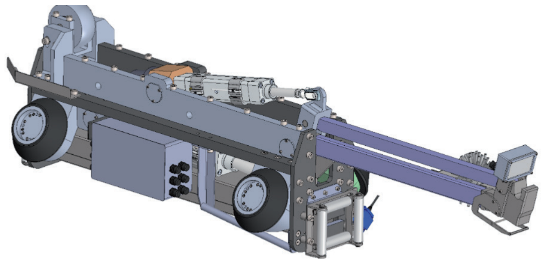

Figure 2: Assistive service robot

The operation of these tasks occurs in highly dynamic and unstructured environments, where rigidly predefined trajectories and control actions can lead to trajectory tracing errors or collisions. Dynamic variability sources of relevance to the issue are:

– *container that is mounted on the mobile base, its location is relative and determined by the position of the cleaning robot,*

– *variations of the mass and centre of gravity of the container as a result of the insertion and unloading of the material,*

– *and the shifting of its own centre of gravity due to manipulation with the decontamination robot.*

These nonlinear and time-varying changes introduce significant uncertainty into the system's dynamics, which needs to be predicted and continuously adapted for the control strategy. To offer compensation for such changes, a sensor fusion architecture combining exteroceptive and proprioceptive information was employed. The sensing subsystem consists of a stereo camera with an inertial measurement unit mounted on the end effector (Fig. 3), motor encoder information, and drive torque estimation. This multi-sensor information provides real-time feedback to the model predictive control algorithm, updating iteratively control inputs to reflect dynamic variations and converge to the destination state under varying operating conditions.

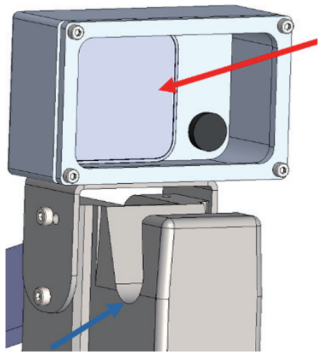

Figure 3: End-effector

3. Experimental Work

In this experiment, we use a basic simulation to demonstrate the possibility of creating MPC logic in MATLAB environment. In the simulation, we introduce random disturbances, which in real life would represent changes in the environment, potential collisions. Based on these disturbances, the MPC logic dynamically adapts the path so that the end-effector still reaches the intended final position.

Since the experiment is conducted within the virtual environment, these external forces are artificially developed. Disturbance is modeled as random variation from the planned path, simulating, for example, a collision that can occur when taking out an insertable container. During experiment simulation we note the reaction of the system to these random changes and quantify the deviation between desired and actual path travelled. This deviation results from the dynamic characteristics of the system, influenced by mass, acceleration, and inertia.

To add more realism to the simulation, we defined maximum accelerations and velocities of actuators and various constraints. They are found from the robot's mechanical structure, such as limits of arm reach, and physical actuator parameters, which influence the dynamics of the system. In the experiment, we defined the following parameters:

- *robot arm lengths, initial positions, and joint angle range,*
- *the maximum velocity and acceleration of each actuator.*

Our final control structure combines the predictability of MPC with the local accuracy of Jacobian-based IK. The MPC block produces a smooth reference path, say a circle, as well as compensates for disturbances or infeasible points. The IK block transforms each reference point into joint angles and base positions via real-time Jacobian calculation. This method uses a Jacobian-based numerical procedure that incorporates both arm joint motion and base translation in the inverse kinematics calculation.

The robot arms' movement is obtained by means of two rotary motors with ball screw mechanisms connected to each arm. First arm is red, and second arm is blue (Fig.4). Figure 4 also shows the base (which is shaded grey) moving in the x-axis on a wheeled platform. The angular movement of the two motors is related to the linear translation of the actuator by the screw pitch. Turning the first motor, that of the red arm, we change angle θ_1, and turning the second motor, that of the blue arm, we change angle θ_2.

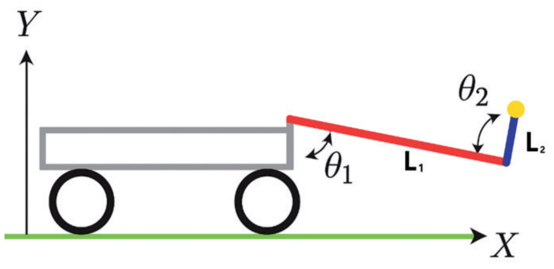

Figure 4: Simplified diagram of the robot model

Position Equations:

$$x = x_0 + L_1(\theta_1)\cos + L_2\cos(\theta_1 + \theta_2) \qquad (1)$$

$$y = 0 + L_1\sin(\theta_1) + L_2\sin(\theta_1 + \theta_2) \qquad (2)$$

Jacobian matrix (2x3):

$$J = \begin{bmatrix} -L_1\sin(\theta_1) - L_2\sin(\theta_1+\theta_2) & -L_2\sin(\theta_1+\theta_2) & 1 \\ L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) & L_2\cos(\theta_1+\theta_2) & 0 \end{bmatrix} \qquad (3)$$

In our setup, the robot follows a reference trajectory, which can be any predefined path such as a circle, a straight line, or a spline-based curve. The MPC continuously recalculates the optimal movement based on possible changes in this target path. To simulate real-world conditions, random deviations are applied during the motion to represent environmental changes or real-time updates from sensors. However, these changes are disabled in the initial and final parts of the motion to avoid large deviations at the start or end of the trajectory.

At each control step, the MPC algorithm takes the robot's current position and computes a short-term movement plan that moves the end-effector closer to the desired trajectory while respecting various system constraints. These constraints include limits on joint angles, maximum speed of the actuators, and maximum acceleration of both the robot's base and its arm joints. Once the optimal short-term plan is found, only the first movement is executed. The process is then repeated at the next time step, considering any new information about the robot's state or the target.

The MATLAB implementation runs a real-time optimization loop using the "fmincon" solver. This solver tries to minimize a single numerical value called the scalar cost [17]. The scalar cost is a number that represents how "good" a particular movement option is. A low cost means the movement is close to the target path, respects the reference trajectory, and avoids obstacles. A high cost means one or more of these goals are not being met.

In our case, this scalar cost is determined by three main factors:

– *tracking accuracy – how close the predicted end-effector position will be to the updated target point,*
– *path adherence – how much the movement deviates from the original reference path,*
– *obstacle avoidance – how far the movement keeps the end-effector away from defined obstacles.*

If a sudden change in the target trajectory occurs, the robot uses an adaptive slowdown mechanism. This means it automatically reduces its maximum speed and acceleration when large corrections are required. This prevents overshooting and helps the robot maintain stability, even during aggressive trajectory changes.

For example, if an obstacle is detected during motion, the MPC increases the penalty for moving near the obstacle, causing the optimizer to choose a detour. Once the obstacle is gone, the path adherence term gradually brings the motion back toward the reference trajectory. This entire process happens in real time, ensuring that the robot adapts smoothly without stopping or recalculating the entire path from scratch.

In our system, the robot follows a reference path, any pre-specified path such as a circle, line, or spline-curves. The Model Predictive Control (MPC) continuously re-computes optimum motion from possible variations in this target path. For real-world imitation purposes, random disturbances are added in most of the motion to reproduce environmental noise or sensor updates in real time.

4. Results and Discussion

The MATLAB simulation produces a series of plots representing the motion of the robot, control performance, and system dynamics. Figure 5 indicates to us that we can plot the paths. The paths were designed using a MATLAB script and illustrate the individual trajectories that were executed by the tip of the end effector. In the figure 5, we can see a blue circle that represents the ideal trajectory we wanted to perform. The orange dashed line shows the optimized path obtained based on the MPC logic. The green line shows the executed path. In the figure 5, we can see that the real system did not necessarily follow the MPC trajectory, due to the ratio of the system's dynamics and speed of random change. The green trajectory has small but repeated oscillations, primarily in the lower-left region of the circle, because of random reference disturbances and mechanical limitations on base and joint accelerations.

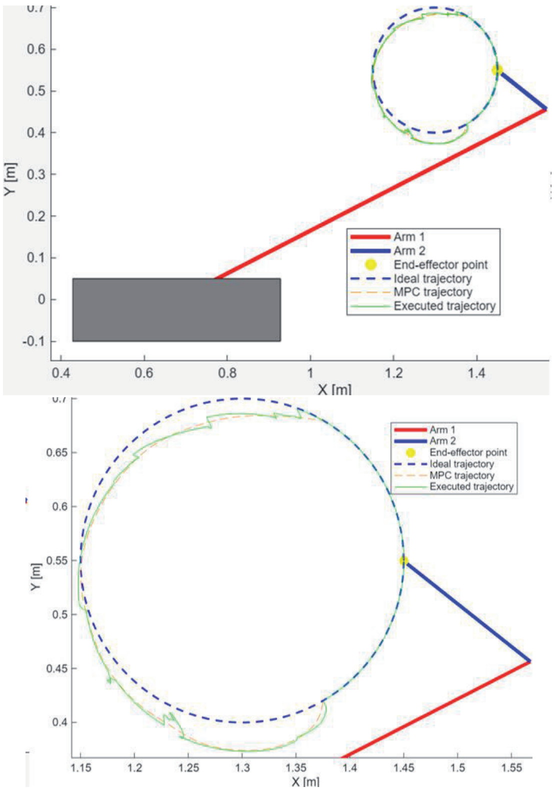

Figure 5: Comparison between different trajectories

The difference between MPC trajectory and executed trajectory plot (Fig.6) shows the magnitude of the deviation between the MPC setpoint and the executed trajectory. Low, constant values indicate precise tracking, and sudden spikes are indicative of rapid path switching or large disturbances. In our experiments, a typical steady-state error was within the range of a few millimetres, but in the case of severe disturbances, the errors would increase significantly if acceleration bounds were saturated.

Angular velocities of the first joint (ω_1, red line) and the second joint (ω_2, blue line) of the robot arm are represented in figure 7. Smooth and low values

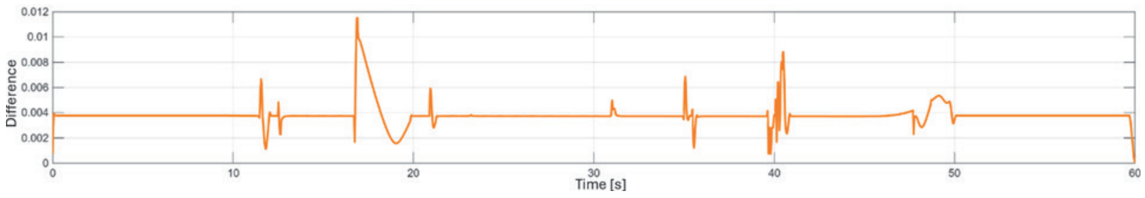

Figure 6: Difference between MPC trajectory and executed trajectory

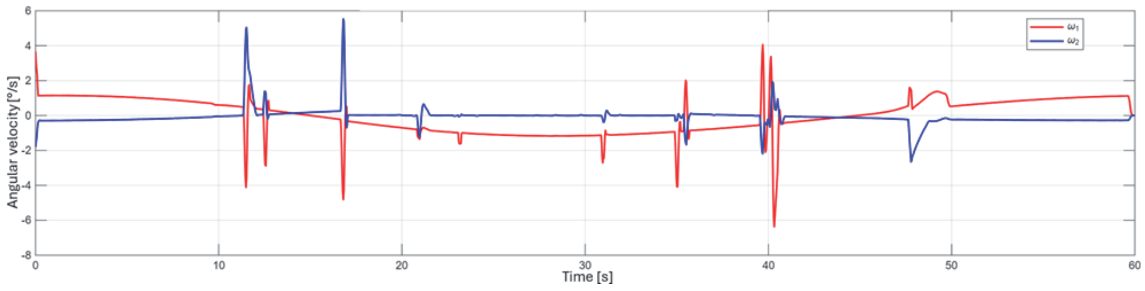

Figure 7: Angular velocities

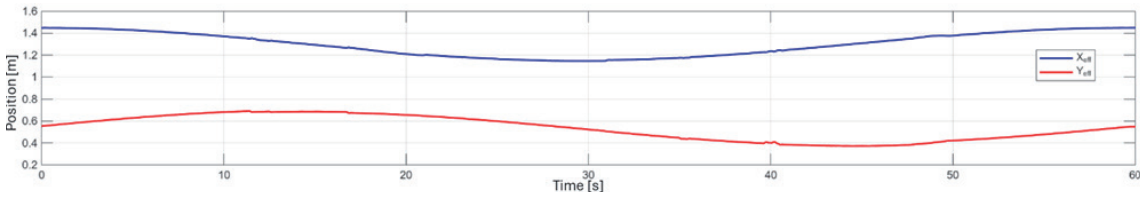

Figure 8: Endpoint position in X and Y Coordinates

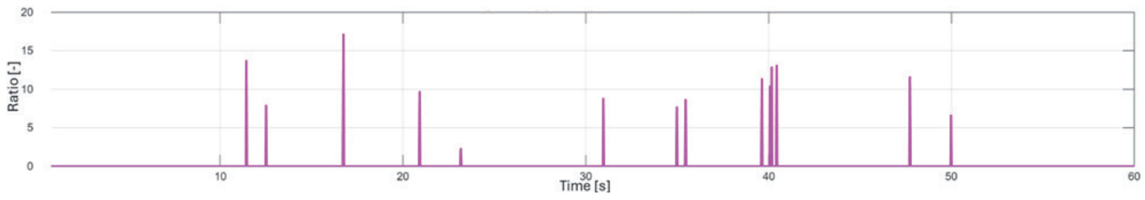

Figure 9: External change / max reaction ratio

indicate that the arm movement are stable and not jerky. Sudden angular velocity peaks indicate rapid direction changes or stiff reaction to large trajectory disturbances that might cause greater mechanical wear or likelihood of breaching actuator acceleration limits. To ensure good quality of control, the curves should be very smooth with no sharp changes occurring too frequently, and their values should not exceed the physical actuators' limits.

End-Effector Position in X and Y Coordinates shows how the end-effector position changes with time along both the X-axis (blue curve Fig.8) and Y-axis (red curve Fig.8). Smooth sinusoidal-like curves are expected when following a circular or other smooth trajectory. Large excursions or oscillations in either axis indicate disturbances, tracking error, or actuator limitations.

Dynamic ratio plot axis (Fig.9) measures the ratio of disturbance magnitude to the system's maximum possible correction in one control step. Less than 1 indicates that the system can easily follow the disturbances. Greater than 1 indicates that the disturbance change occurred faster than the robot's reaction ability.

We can see from the charts that our Matlab optimization model has errors and needs to be adjusted to be use in the assistance robot. The tracking error between the executed trajectory and the mpc-planned path can be solved through several strategies. Some solutions try to improve robot's physical performance, and others try to improve control algorithms and trajectory planning. Improving the dynamic capabilities of the robot. Possible solutions that could reduce the deviation and errors are listed below.

– *Improvement in the MPC Controller, higher position error penalty in the cost function, with greater priority on precise tracking versus low control effort. Jerk penalty or acceleration change penalty to avoid abrupt change in velocity and give smoother control action. Adaptive prediction horizon — longer horizon for steady motion and shorter horizon for high-speed disturbances, balancing responsiveness and stability.*

– *MPC trajectory filtering, applying a low-pass filter to remove high-frequency noise before the trajectory is fed into the MPC. Applying a moving average filter or Bézier spline smoothing to allow the path to transition smoothly.*

Because we test MPC only in a virtual environment, in my future work I will focus on integrating Matlab with Webots. We can use the Webots platform to simulate and send signals to a real robot and also to create a digital twin. by this achieve reliable semi-autonomous control of an assistance service robot.

5. Conclusions

The combination of Model Predictive Control with Jacobian-based Inverse Kinematics was effective for smooth, adaptive trajectory generation of a mobile robotic platform in dynamically changing environments. Simulation results verify that the control approach can ensure tracking precision under moderate disturbances while adhering to actuator limits. The deviations experienced under severe disturbances point towards the necessity of further optimizing the MPC cost function through higher position error penalties, jerk minimization, and adaptive prediction horizons. From the results, we found that the MPC model still needs to be tuned before it can be used to control an assistance robot. The main problem is oscillations which could also be alleviated and control stability improved through trajectory smoothing techniques such as low-pass filtering or spline fitting. Although the current study was restricted to MATLAB simulation, the methodology has prospects for real-world application in hazardous and inaccessible environments, including decontamination of hot gas chambers. Future research efforts will involve the incorporation of the control algorithm within the Webots environment for enabling digital twin development, semi-autonomous operation, and validation of the methodology on the physical robot system.

Acknowledgments

This research was supported by project VEGA: 1/0215/23 Research and development of robotic workplaces equipped with industrial and collaborative robots.

References

1. Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to reactive collision avoidance for mobile robots with synchro-drives (Tech. Rep.). Department of Computer Science, University of Bonn; Robotics Institute, Carnegie Mellon University.
2. Brock, O., & Khatib, O. (1999, May). High-speed navigation using the global dynamic window approach. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (p. 341). IEEE.
3. T. Qin, P. Li and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004-1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.
4. Wang et al., Navigating Mobile Robots in Unknown and Dynamic Environments Using Deep Reinforcement Learning, Scientific Reports (Nature), 2024
5. Khouili et al., A Perception-Aware MPC for Real-Time Mobile Robot Navigation, MDPI Applied Sciences.
6. Lin et al., Real-Time Mapping and Path Planning for Autonomous Robots, MDPI Sensors, 2023
7. Alatise & Hanheide, Challenges in Robot Deployment in Real-World Environments, Robotics and Autonomous Systems, 2021
8. Wu, X., Chen, S., Sreenath, K. S., & Mueller, M. W. (2022). Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry (IEEE Access, 10, 65 507–65 519).
9. Hsieh, C.-H., & Liu, J.-S. (2012, July 11–14). Nonlinear model predictive control for wheeled mobile robot in dynamic environment. In Proceedings of the 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (pp. 363–368). IEEE
10. Mohamed, I. S., Ali, M., & Liu, L. (2025). Chance-Constrained Sampling-Based MPC for Collision Avoidance in Uncertain Dynamic Environments (arXiv:2501.08520).
11. E. F. Camacho and C. Bordons, Model Predictive Control, Springer, 2004.
12. H. Gao et al., "Distributed MPC for Cooperative Multi-Robot Navigation,"
13. de Cos, C. R., Acosta, J. Á., & Ollero, A. (2020, March). Adaptive

Integral Inverse Kinematics Control for Lightweight Compliant Manipulators. IEEE Robotics and Automation Letters, 5(2), Article PP(99):1-1.

14. A. Lunia, Inverse Kinematics – Modeling, Motion Planning, and Control, Clemson University Open Textbook, Chapter 3.
15. "Mastering Inverse Kinematics in Robotics," NumberAnalytics blog, Jun. 23, 2025
16. Stubna m.; Pekar, A.; Moravek, J.; Spirko, M. "Decommissioning Project of A1 Bohunice NPP," VUJE Trnava Inc., February 2002.
17. MathWorks, Optimization Toolbox – fmincon, The MathWorks, Inc., 2024. [Online]. Available: https://www.mathworks.com/help/optim/ug/fmincon.html