Predictive Control of the KINOVA Gen3 Robotic Manipulator Using a Nonlinear Model

Conference Paper · February 2024

DOI: 10.1115/IMECE2023-112777

2 authors:

Amanuel Tereda
North Carolina Agricultural and Technical State University
5 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Sun Yi
North Carolina Agricultural and Technical State University
48 PUBLICATIONS **169** CITATIONS
SEE PROFILE

**Proceedings of of the ASME 2023 &
International Mechanical Engineering Congress and Exposition
IMECE2023
October 29-November 2, 2023, New Orleans, Louisiana, USA**

IMECE2023-112777

PREDICTIVE CONTROL OF THE KINOVA GEN3 ROBOTIC MANIPULATOR USING A NONLINEAR MODEL

Amanuel Abrdo Tereda[1], **Sun Yi**[1]
[1]North Carolina Agricultural and Technical State University, North Carolina, USA

ABSTRACT

In contemporary production settings, reliable but efficient pick-and-place robots are frequently used. The automation and optimization of the pick and place procedures utilizing various path-planning approaches thereby support the expansion of application areas. Yet, the design of a controller faces significant difficulties due to the nonlinearities of robotic manipulators and the unpredictable nature of the ambient factors. In place of the classic model predictive control (MPC), this paper presents the Nonlinear Model Predictive Controller (NLMPC) as an acceptable control mechanism for real-time optimization and robust stability of the Kinova Gen3 robotic arm. The developed NLMPC-based method ensures that the robotic arm does not run into obstacles in the workplace or with itself while reaching, gripping, selecting, and placing the necessary items. To acquire the next control input trajectory, the optimization in NLMPC is solved repeatedly with each newly measured state. When input constraints are available, the modeled system tracks reference trajectories to achieve the aim of recognizing and organizing distinct objects. After the NLMPC is successfully developed, a simulation environment is built and finally brought to life by combining all the processes into one using a MATLAB Stateflow chart.

Keywords: Automation, optimization, path-planning, nonlinear model predictive controller, kinova Gen3, stateflow

1 INTRODUCTION

Understanding the major aspect and application of the manipulators, which is pick and place, is highly beneficial in improving and enhancing robot manipulator operations. In today's manufacturing environments, pick-and-place robots are widespread and their automation shortens the time it takes to pick up parts or things and move them to new locations. This technique can be automated to help boost manufacturing rates. Pick-and-place robots take care of repetitive jobs, allowing humans to focus on more complicated tasks [1]. Therefore, selecting and implementing a path planning approach that corresponds to the defined application of pick and place which is robust in its design and can work immensely and effectively with the robotic arm is quite helpful. The primary goal of path planning is to generate a reasonable and smooth trajectory by extracting waypoints from the start to the destination address point. The waypoints are divided into numerous center coordinate positions between the extracted map linkages [2].

Model Predictive Control (MPC) is a highly established technique today and it is also safe to state that it is the most common method for establishing limited, multivariable control in today's process industries [3]. MPC can also be used to solve control problems where computing off-line control laws is difficult or impossible [4]. One of the important characteristics of this sort of control is its capacity to deal with severe limits on controls and states [5]. In contrast, nonlinear Model Predictive Control puts a focus on performance by solving an online finite horizon optimum control problem and applying the first element of the calculated open-loop input trajectory to the system [6]. To acquire the next control input trajectory, the optimization in NLMPC is solved repeatedly with each newly measured state [7]. As the name implies, model predictive control is heavily reliant on the model of the robotic manipulator employed for the desired task. According to research, the KINOVA Gen3 robotic

arm model is NLMPC technique compatible [8].

Oleinikov et.al., presented a nonlinear model predictive control technique for real-time planning of point-to-point motions of serial robot manipulators who share their workspace with a human. In their paper, the NLMPC law, which is based on a kinematic model, solves a nonlinear program online and ensures safety by restricting the robot speed within the time-varying boundaries established by the speed-and-separation-monitoring (SSM) principle [9].

Elsisi et.al., worked on the effective nonlinear model predictive control which is tuned by an improved neural network (NN) for robotic manipulators. According to their research, the nonlinearities of robotic manipulators as well as the uncertainties of their parameters, provide significant difficulties to controller design. Furthermore, the major goal of the robotic reaction is to track regular and irregular trajectories with fewer overshoots, a quick settling time, and a minimal steady-state error. As a result, instead of using the standard MPC, this work introduces the nonlinear MPC as an appropriate control approach for nonlinear systems [10].

Another work by Nikou et.al., dealt with a nonlinear model predictive control strategy for cooperative manipulation with singularity and collision avoidance. The paper discusses the subject of cooperative transportation of a firmly gripped object by N robotic agents. The study proposes a Nonlinear Model Predictive Control strategy that ensures object navigation to a desired posture in a restricted workspace with obstacles while adhering to specific agent input saturations [11].

Tang et.al., worked on trajectory tracking of robotic manipulators with constraints based on model predictive control. This paper describes a model predictive control scheme for robotic manipulators in trajectory tracking in the presence of input constraints, which provides convergent tracking of reference trajectories and robustness to model mismatch. Finally, the suggested control scheme's convergence is demonstrated using the universal robot (model-UR5) in simulation [12].

The majority of previously published efforts in this field focus on developing a nonlinear model predictive control or other path planning methodologies for robotic arms other than the KINOVA Gen3. As a result, researching the NLMPC path planning approach for the KINOVA Gen3 robotic arm is beneficial, and this study explicitly addresses this benefit by applying the NLMPC path planning method to the Kinova Gen3 robotic arm. It would also be possible to execute the results of this work by gaining access to the controllers of Kinova Gen3's robot drive, which allows for the adjustment of controller gains to obtain improved dynamics and precision [13].

The methodologies utilized to carry out the research are covered in Section 2 of the paper. The procedures taken to implement the research and the results are covered in Section 3. The final section presents a summary of the overall paper and suggests potential directions for further research.

FIGURE 1. KINOVA GEN2 ROBOTIC ARM MOUNTED ON JACKAL GROUND VEHICLE.

2 METHODOLOGY

2.1 Robot Manipulator

Since the robotic arm, KINOVA Gen3 is a crucial part of this study, it is worth taking a closer look at it. Kinova's robot is durable and simple to operate, making it a good foundation for prototyping. In terms of hardware, the new Gen3 appears to have grown in size, delivering an even more powerful thump as a research tool. The Gen3 embedded controller makes it simple and easy to connect our robot in a variety of ways, and it can be customized to fit the needs of a wide range of applications.

It is possible to operate the robot in different ways, including bypassing the controller and regulating each individual actuator directly with closed-loop control at 1KHz control feedback in order to prevent jerky motion [14]. Kinova's Gen3 robot can also be programmed to do simple or sophisticated grasping and manipulation tasks at different levels of skill. High-level and low-level control, intelligent actuators with built-in torque sensors, an optional 2D/3D vision module, a flexible end-effector interface module, and limitless rotation on all joints are among Kinova Gen3's basic characteristics [15]. Figure 1 shows an overview of the Kinova Gen2 robotic arm mounted on the Clearpath Jackal ground vehicle, which is available at North Carolina A&T State University (NCAT) and is one model prior to the Kinova Gen3 robot.

2.2 Implementation Environment

The MATLAB toolboxes Model Predictive Control Toolbox and Optimization Toolbox are utilized to build the nonlinear model predictive control path planning technique employed

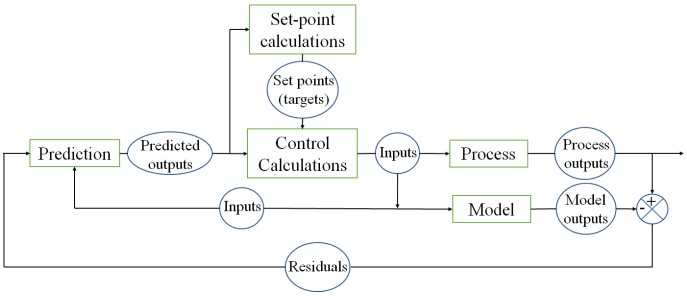

FIGURE 2. BLOCK DIAGRAM OF THE MODEL PREDICTIVE CONTROL SYSTEM.

in this research, which is used to create collision-free, optimum paths for the manipulator to follow. The high-level tasks are scheduled, and they are passed from one to the next, using MATLAB Stateflow. Designing 3D components for the simulation environment is aided by the use of SOLIDWORKS. Finally, the manipulator is modeled, simulated, and shown using the MATLAB Robotics System Toolbox, which also checks for collisions.

2.3 Path Planning Approach

The path followed by the robotic manipulator is planned using model predictive control. Model predictive control is a type of control technique in which the current control action is determined by solving a finite horizon, open-loop, and optimum control problem with the plant's current state as the initial state at each sampling instant. Figure 2 shows a block schematic of the model predictive control system.

Two user-defined functions are used to make up the prediction model for the nonlinear MPC controller: state function and output function. The state function predicts the evolution of plant states across time. The output function calculates plant outputs in terms of state and input variables. The state function in the discrete-time prediction model is the state update function, and it is given by:

$$x(k+1) = f(x(k), u(k)) \tag{1}$$

Where: $f(x(k), u(k))$ is implicitly defined by the originating differential equation that has an equilibrium point at the origin (i.e., $f(0,0) = 0$). $u(k) \in \mathbb{R}^r$ is the vector of controlled variables to be determined by the controller. $x(k) \in \mathbb{R}^r$ is the state vector at time k. $u(k) \in U$ and $x(k) \in X$ must be satisfied by the control and state sequences. U is usually a convex, compact subset of $\mathbb{R}^r$ and X is a convex, closed subset of $\mathbb{R}^n$, with the origin in the interior of each set.

Prior to configuring the nonlinear model predictive controller, it is important to get the number of joints from the Robot

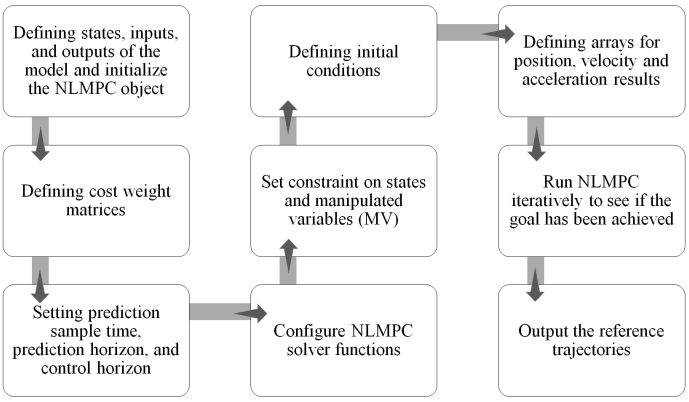

FIGURE 3. STEPS IN SETTING UP THE NLMPC.

as well as the number and kind of obstacles. Because the Kinova Gen3 robotic arm used in this research has 7 degrees of freedom (DOF), 7 joints are taken into account when developing the controller. In order to ensure a precise approximation of the constraint Jacobian in the definition of the nonlinear model predictive control technique, the obstacles are represented as a rectangle and cylinder.

Based on the type of robot and the obstacles the current robot pose and the final end-effector pose are defined next. The nonlinear model predictive controller is created using the MATLAB model predictive control toolbox. During controller development, the maximum number of iterations for the optimization solver is set to 5 in order to save computation time. However, increasing the iteration value is recommended for more accurate and reliable results. Furthermore, the lower and upper boundaries for the joint position, velocity, and acceleration are explicitly set. The processes involved in configuring the nonlinear model Predictive controller are summarized in Fig. 3.

Double integrators are used to characterize the robot joints model. The states of the model are $x = [q, \dot{q}]$, where q represents the 7 joint locations and $\dot{q}$ represents their velocities. The model's inputs are the joint accelerations $u = \ddot{q}$. The dynamics of the model are determined by:

$$\dot{x} = \begin{bmatrix} 0 & I_7 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ I_7 \end{bmatrix} u \tag{2}$$

Where I_7 denotes the $7x7$ identity matrix. The output of the model is defined as:

$$y = \begin{bmatrix} I_7 & 0 \end{bmatrix} x \tag{3}$$

As a result, there are 14 states, 7 outputs, and 7 inputs in the

nonlinear model predictive controller, where:

Inputs ($n_u = \ddot{q}$), joint acceleration = 7 inputs
Outputs ($n_y = q$), joint position = 7 outputs
States ($n_x = [q, \dot{q}]$), joint position and velocity = 14 states

The next stage in the NLMPC setup is to define the cost function. A cost function uses design variable values to analyze our design requirements. The cost function is utilized at the command line after writing and saving it for estimation, optimization, or sensitivity analysis. The cost function employed is a custom nonlinear cost function defined as a quadratic tracking cost plus a terminal cost.

$$J = \int_0^T (p_{ref} - p(q(t)))' Q_r (p_{ref} - p(q(t))) + u'(t) Q_u u(t) dt + (-p(q(T)))' Q_t (p_{ref} - p(q(T))) + \dot{q}'(T) Q_v \dot{q}(T) \quad (4)$$

The matrices Q_r, Q_u, Q_t, and Q_v are constant weight matrices. The importance of each of the control objectives in the cost function is determined by these weights. The cost weights that are employed in the model are: running cost weight (Q_r), terminal cost weight (Q_t), input cost weight (Q_u), and terminal joint velocity cost weight (Q_v). On desired end-effector pose $[x, y, z, phi, theta, psi]$, the running ($Q_r$) and terminal ($Q_t$) cost weights are utilized, whereas the input cost weight (Q_u) is used on joint acceleration ($\ddot{q}$), and the terminal joint velocity cost weight (Q_v) is used on joint velocities ($\ddot{q}$).

After setting the cost weights, the next stage is to define the prediction model sample time (T_s), the prediction horizon (p), and the control horizon (c). For prediction, the controller employs a discrete-time model with sample time T_s. Sample time for the prediction model is provided as a positive finite scalar. The control interval time (T_s) should be chosen first, and then held constant as other controller properties are adjusted. If it becomes clear that the original choice was poor, T_s must be updated. Other settings may need to be retuned as well. Qualitatively, as T_s lowers, rejection of unknown disturbances typically improves before remaining stable. The T_s value at which performance reaches a plateau is determined by the dynamic features of the plant. The computational effort, on the other hand, rises considerably as T_s decreases. As a result, a compromise between performance and computational effort is the best option.

The prediction horizon, p, is also an important factor in Model Predictive Control. The number of steps in the prediction horizon is given as a positive integer. The prediction time is calculated as the product of the prediction Horizon and the prediction model sample time. If the prediction horizon duration (the product $p * T_s$) is kept constant, p must fluctuate in inverse proportion to T_s. The size of many arrays is proportional to p. As p grows, the controller memory requirements and the quadratic programming solution time increase. Quadratic programming is the problem of optimizing a quadratic objective function and is one of the simplest forms of non-linear programming [16]

The following should be kept in mind when selecting T_s, p, and c:

1. Run at least one simulation to check if T_s improves unmeasured disturbance rejection significantly. If this is the case, T_s should be revised.
2. See the result by setting $T_s < 1$.
3. It is recommended to increase the prediction horizon (p) until it has only a tiny effect on performance.
4. The control horizon (c) is located between 1 and p.

The following values are determined by considering the above parameters:

1. Prediction model sample time (T_s) = mpc time step = 0.55
2. Prediction horizon (p) = 2
3. Control horizon (c) = 1

The NLMPC solver functions must now be configured. After that, a closed-loop trajectory optimization is used to build reference trajectories. The NLMPC solver functions that must be configured are the model state function, model output function, and the NLMPC cost function with their Jacobian. Optimization solver options must also be configured. The function tolerance, step tolerance, maximum iteration, and constraint tolerance are the optimization solver options that must be configured.

Following that, a constraint on states and plant-manipulated variables (MV) is specified. When using the MPC command to build a controller object, there are no constraints by default. The relevant controller property must be set to integrate a constraint. Tab. 1 lists the maximum and minimum constraint values for the 14 states while Table 2 lists the maximum and minimum constraints for the seven manipulated variables.

The next stage is to generate the reference trajectories. When using closed-loop trajectory optimization to create reference trajectories, the initial conditions for the current robot setup, as well as the maximum iteration number must be first determined. The maximum iteration is set to 50, which means that it will compute the optimal trajectory for the robotic arm to follow within this iteration. Then arrays to hold the results of position, velocity, and acceleration can be created. After initializing the arrays of position, velocity, and acceleration to store results, the next step is to run the NLMPC iteratively across the specified time horizon until the goal is achieved or the maximum number of iterations is reached. During this operation, the initial state and time must be adjusted for the next iteration. The trajectory points will then be saved, and the target will be checked to see if it was met.

Now that the aim of computing a feasible trajectory has been fulfilled, the reference trajectories for location, velocity, and acceleration can be outputted. Finally, the running cost, terminal

TABLE 1. THE MAXIMUM AND MINIMUM CONSTRAINT VALUES FOR THE 14 STATES.

States (joint position and velocity)		
Fields	Minimum	Maximum
1	-174.5300	174.5300
2	-2.2000	2.2000
3	-174.5300	174.5300
4	-2.5656	2.5656
5	-174.5300	174.5300
6	-2.0500	2.0500
7	-174.5300	174.5300
8	-0.8727	0.8727
9	-0.8727	0.8727
10	-0.8727	0.8727
11	-0.8727	0.8727
12	-0.8727	0.8727
13	-0.8727	0.8727
14	-0.8727	0.8727

cost, and total cost (sum of running and terminal cost) will all be calculated. Jacobians will be initialized as well, including the robot Jacobian, the running cost Jacobian, and the terminal cost Jacobian.

3 IMPLEMENTATION AND RESULTS

The output of the developed NLMPC is evaluated in a simulated scenario that includes the Kinova Gen3 robotic arm, a table, a trashcan, and trash. The developed NLMPC approach assures that the robotic arm does not collide with itself or with barriers in the workplace while reaching, grabbing, picking, and placing the items that are required. The 3D components of the simulation environment are created using the computer-aided design tool SOLIDWORKS and MATLAB built-in toolboxes. The simulation is finally brought to life by combining all the processes into one using a MATLAB Stateflow chart.

The aim is for the Kinova Gen3 robotic arm to approach the trash while avoiding collisions with obstacles, including itself, using the path generated by the developed NLMPC controller. Once it has arrived at the trash, it will pick it up and dispose of it in the trashcan by following the path generated by the NLMPC. Using the black broken lines, we can observe the route formed

TABLE 2. THE MAXIMUM AND MINIMUM CONSTRAINTS FOR THE SEVEN MANIPULATED VARIABLES.

Manipulated variables		
Fields	Minimum	Maximum
1	-1	1
2	-1	1
3	-1	1
4	-1	1
5	-10	10
6	-10	10
7	-10	10

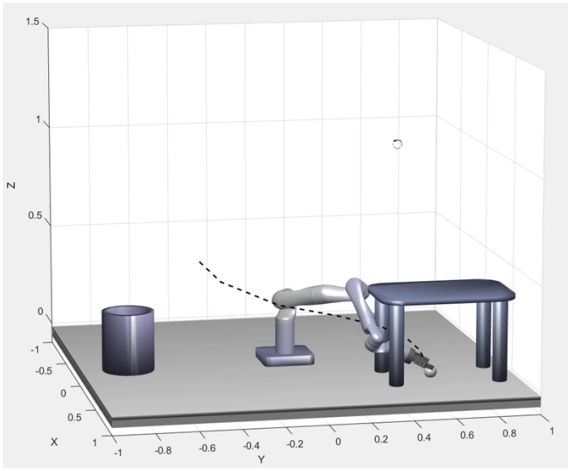

FIGURE 4. PATH FOLLOWED BY THE ROBOT WHILE PICKING UP THE TRASH.

by the NLMPC controller during the activity. Figure 4 and Fig. 5 show the path taken by the robotic arm while picking up trash from its initial position and placing it in a trashcan, respectively.

The simulation demonstrates that the route planning algorithm was developed successfully and that the robot can successfully follow the path defined by the NLMPC controller.

The velocity and acceleration vs time plots, as the robot advances to the home position, picking approach position, and placement approach position, are shown in addition to the simulation findings in Figs. 6-8.

The plot relates the end effector velocity and acceleration to the overall robot velocity and acceleration, which is the average of the 7 joints. The graphs clearly show that the end effector is functioning in harmony with the overall robot movement and will finally asymptote to zero (stops) at the same time as the robot.

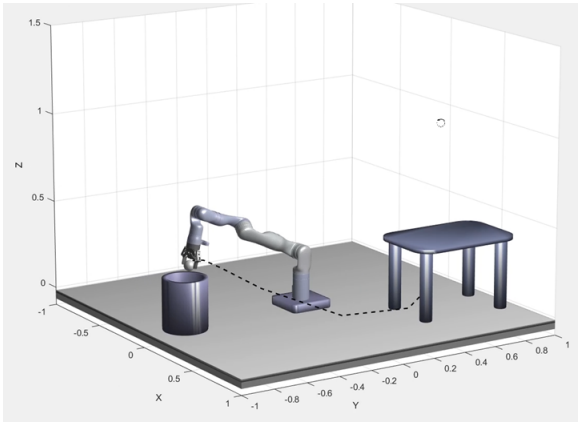

FIGURE 5. PATH FOLLOWED BY THE ROBOTIC ARM WHEN DISPOSING OF THE TRASH.

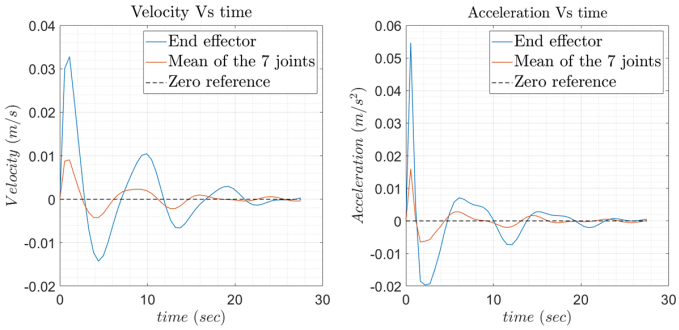

FIGURE 6. VELOCITY AND ACCELERATION VS TIME PLOT WHEN ROBOT MOVES TO THE HOME POSITION.

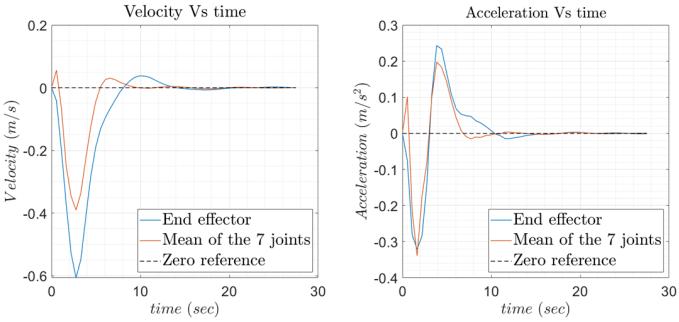

FIGURE 7. VELOCITY AND ACCELERATION VS TIME PLOT WHEN ROBOT MOVES TO THE PICKING APPROACH POSITION.

The deflection of the values in the graph occurs due to obstacles, and as the robot approaches the obstacles, it automatically manages its deceleration and begins moving to reach the target after avoiding them.

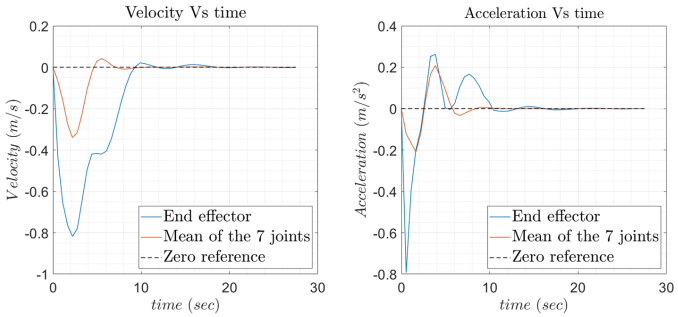

FIGURE 8. VELOCITY AND ACCELERATION VS TIME PLOT WHEN ROBOT MOVES TO PLACEMENT APPROACH POSITION.

4 CONCLUSION

The goal of the research presented is to design a controller for planning and control of the Kinova Gen3 robotic arm based on a Nonlinear Model Predictive Controller, as well as to develop elements of a 3D simulation environment using SOLIDWORKS and MATLAB to show the control technique. The NLMPC is developed and implemented effectively in the research, and as a consequence, it works in conjunction with other MATLAB states produced to achieve the goal of picking and placing objects.

Nonlinear optimization algorithms, on the other hand, have a large computational load, preventing them from being employed as a controller for quick plans or when a fast action, such as trajectory tracking, is required. To tackle the trajectory tracking problem in the future, it is preferred to use the Quasi-Linear Parameter Varying (Quasi-LPV) representation, which combines the Nonlinear Model Predictive Control approach with the presence of input saturation and un-modeled dynamics. As a consequence, typical Quadratic Programming optimization approaches for the online optimization issue may be used, resulting in faster and more efficient convergence to the optimal solution [17].

Finally, when it comes to putting the results into action, job safety is a major problem for many industrial organizations where human eyes or presence are not permitted, such as the boiling place and nuclear power plants. Machine learning and image processing technologies might be used extensively to circumvent these constraints [18]. To achieve the image processing features a depth camera and a 3D Lidar can be utilized with 2D and 3D image processing machine learning models. Additionally, after choosing the required object, a robotic arm mounted on a ground robot as shown in Fig. 1 might be employed in a joint application to solve the issues with distance. [19].

As a result, implementing the results of this work on an actual Kinova Gen3 is planned for the future while taking the aforementioned crucial elements into account.

ACKNOWLEDGEMENTS

The work is supported by the U.S. National Science Foundation (NSF), NSF'S Hybrid Autonomous Manufacturing Moving from Evolution to Revolution Engineering Research Center (HAMMER ERC), and Oak Ridge National Lab. This work is also supported by the Department of Energy Minority Serving Institution Partnership Program (MSIPP) managed by Savannah River National Laboratory under BSRA contract 0000602156.

REFERENCES

[1] M. R. Pedersen et al., "Robot skills for manufacturing: From concept to industrial deployment," Robotics and Computer-Integrated Manufacturing, vol. 37, pp. 282-291, 2016.

[2] S. C. Dekkata, S. Yi, M. Muktadir, S. Garfo, X. Li, and A. A. Tereda, "Improved Model Predictive Control System Design and Implementation for Unmanned Ground Vehicles," 2022.

[3] P. Tatjewski, Advanced control of industrial processes: structures and algorithms. Springer Science Business Media, 2007.

[4] D. Angeli, A. Casavola, G. Franzè, and E. Mosca, "An ellipsoidal off-line MPC scheme for uncertain polytopic discrete-time systems," Automatica, vol. 44, no. 12, pp. 3113-3119, 2008.

[5] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," IEEE Transactions on control systems technology, vol. 18, no. 2, pp. 267-278, 2009.

[6] U. Rosolia, F. Braghin, A. Alleyne, and E. Sabbioni, "Nlmpc for real time path following and collision avoidance," SAE International Journal of Passenger Cars-Electronic and Electrical Systems, vol. 8, no. 2015-01-0313, pp. 401-405, 2015.

[7] E. Ali and E. Zafiriou, "Optimization-based tuning of nonlinear model predictive control with state estimation," Journal of Process Control, vol. 3, no. 2, pp. 97-107, 1993.

[8] A. Mavrommati, C. Osorio, R. G. Valenti, A. Rajhans, and P. J. Mosterman, "An Application of Model Predictive Control to Reactive Motion Planning of Robot Manipulators," in 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), 2021, pp. 915-920: IEEE.

[9] A. Oleinikov, S. Kusdavletov, A. Shintemirov, and M. Rubagotti, "Safety-aware nonlinear model predictive control for physical human-robot interaction," IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 5665-5672, 2021.

[10] M. Elsisi, K. Mahmoud, M. Lehtonen, and M. M. Darwish, "Effective nonlinear model predictive control scheme tuned by improved NN for robotic manipulators," IEEE Access, vol. 9, pp. 64278-64290, 2021.

[11] A. Nikou, C. Verginis, S. Heshmati-Alamdari, and D. V. Dimarogonas, "A nonlinear model predictive control scheme for cooperative manipulation with singularity and collision avoidance," in 2017 25th Mediterranean Conference on Control and Automation (MED), 2017, pp. 707-712: IEEE.

[12] Q. Tang, Z. Chu, Y. Qiang, S. Wu, and Z. Zhou, "Trajectory tracking of robotic manipulators with constraints based on model predictive control," in 2020 17th International Conference on Ubiquitous Robots (UR), 2020, pp. 23-28: IEEE.

[13] R. Szczepanski, K. Erwinski, M. Tejer, A. Bereit, and T. Tarczewski, "Optimal scheduling for palletizing task using robotic arm and artificial bee colony algorithm," Engineering Applications of Artificial Intelligence, vol. 113, p. 104976, 2022.

[14] L. Guzman, V. Morellas, and N. Papanikolopoulos, "Robotic Embodiment of Human-Like Motor Skills via Reinforcement Learning," IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 3711-3717, 2022.

[15] KINOVA. Gen3 robots. Available: https://www.kinovarobotics.com/product/gen3-robots.

[16] M. Frank and P. Wolfe, "An algorithm for quadratic programming," Naval research logistics quarterly, vol. 3, no. 1-2, pp. 95-110, 1956.

[17] M. Esfandiari, S. Chan, G. Sutherland, and D. Westwick, "Nonlinear model predictive control of robot manipulators using quasi-LPV representation," in 2019 7th International Conference on Control, Mechatronics and Automation (ICCMA), 2019, pp. 116-120: IEEE.

[18] S. Garfo, M. Muktadir, and S. Yi, "Defect detection on 3D print products and in concrete structures using image processing and convolution neural network," Journal of Mechatronics and Robotics, vol. 4, no. 1, pp. 74-84, 2020.

[19] M. Muktadir and S. Yi, "Machine Vision-Based Detection of Surface Defects of 3D-Printed Objects," in 2021 ASEE Virtual Annual Conference Content Access, 2021.