Model Predictive Control approaches for collision free manipulator arm control

Archit Hardikar
University of Pennsylvania
Philadelphia, USA
architnh@seas.upenn.edu

Joshua Ernest P
University of Pennsylvania
Philadelphia, USA
josherne@seas.upenn.edu

Rohit Bhikule
University of Pennsylvania
Philadelphia, USA
rbhikule@seas.upenn.edu

***Abstract*—Operation of robotic arms in most industrial settings require predefined trajectories for control and optimization. Most applications require obstacle avoidance and interaction at a higher level of abstraction. In this work, we present an approach to implement an online planning and control mechanism for safe control of a 7 degree of freedom (DoF) robotic arm. We discuss two approaches to implement static and dynamic obstacle avoidance. We validate our algorithm on multiple maps with objects of increasing complexity in the simulation environment Gazebo. The robots are controlled using the Robot Operating System (ROS). We demonstrate, that our approach is real-time capable and, quite possible to execute despite having 21 variables in the state vector and numerous constraints which significantly increase the system complexity. Code has been made available at: https://github.com/JoshuaErne/manipulator_mpc**

I. Introduction

Modern industrial processes are increasingly dominated by shorter innovation and product life cycles, reflecting a growing demand for customized products [1]. To match this increasing demand, factories have started employing robotic manipulators. Traditionally, the collision free trajectories of all involved robotic manipulators in industrial applications are planned for a specific task involving an unchanging environment. In case certain parts of the production process are changed, a re-planning of collision free trajectories and re-programming of manipulators is necessary. For this reason, it is imperative to develop efficient and safe online motion control strategies for generating collision free trajectories.

Model Predictive Control is one of the most efficient point-to-point trajectory generation method used in situations that necessitate shortest execution time. It is a well-grounded option for trajectory generation since its formulation includes a final-state constraint to be satisfied at the end of horizon while respecting states and inputs' limits [2]. In the presence of an internal controller with a short time constant considering the robot dynamics, position or velocity references can be used directly making a complex dynamic model not necessary in the MPC. Therefore, a purely kinematic model can be used [2].

II. Related Work

In industries, trajectory generation of manipulators is usually required, along with minimum travelling time and maximum efficiency [1]. In general, applied methods for trajectory generation in robotics applications can be divided into two methods: Sampling-based (like RRT, PRMs) and control based (like potential fields). However, both methods have limitations. Sampling-based methods have difficulties in planning trajectories for narrow passages as they produce jerky motions whereas control-based planners are limited to low-dimensional configuration space. Due to these limitations, there is a need for optimization-based methods which can be formulated as constrained-optimization problem.

Previous work has been done on trajectory generation with MPC for robot manipulator without collision avoidance using mainly two approaches. The first approach is extended by solving the optimization problem not over the whole state space of the considered system, but only over a subset of the state space. This subset excludes all states where a collision might occur and needs to be determined a priori [3]. The second approach is to integrate the collision avoidance constraints to the optimization problem. There exist several approaches to formulate these constraints. One approach is to restrict the distance of all collision-prone object pairs, where the corresponding objects are approximated by convex bodies. Thus, for a kinematic model of a manipulator this results in a connected chain of convex bodies [4]. In another paper on MPC strategies for Robotic Manipulator Arms [5], the authors propose an approach for obstacle avoidance using robust model predictive control (RMPC) and function approximation via supervised learning with (deep) neural networks (NNs). Existing approaches of MPC, still, cannot guarantee a collision free trajectory generation in dynamically changing environments.

III. Methods

A. Dynamics of the model

As in previous research [6], the continuous-time model chosen is constructed by multiple decoupled chains of integrators. Thus, the state vector, $x_c \in \mathbb{R}^{21}$ is composed by the joint angle q_i, velocity $\dot{q}_i$ and acceleration $\ddot{q}_i$ for each robot joint $i = 1, \cdots, 7$.

$$x_c = \begin{bmatrix} q_1 & \dot{q}_1 & \ddot{q}_1 & \cdots & q_7 & \dot{q}_7 & \ddot{q}_7 \end{bmatrix} \in \mathbb{R}^{21}$$

The continuous-time linear model can be written as -

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t)$$

with

$$A_c = \text{blkdiag}([\tilde{A}_c, \cdots, \tilde{A}_c]), B_c = \text{blkdiag}([\tilde{B}_c, \cdots, \tilde{A}_c])$$

blkdiag(.) forms a block diagonal matrix from the given list of matrices, $A_c \in \mathbb{R}^{21x21}, B_c \in \mathbb{R}^{21x7}$, and

$$\tilde{A}_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \tilde{B}_c = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

The continuous-time input is the angular jerk of the joints, $u_c = \dddot{q} \in \mathbb{R}^7$.

While choosing the sampling period, h, to discretize the continuous-time linear system, a sampling period different from the one of the controlled system was chosen for discretization. Then, a linear interpolation of the calculated input sequence is used to provide references at sampling rate of robot.

$$x_{k+} = \Phi x_k + \frac{1}{h}\Gamma_1 u_{k+1} + \left(\Gamma - \frac{1}{h}\Gamma_1\right) u_k$$

with

$$\begin{aligned} \Phi &= \text{blkdiag}([\tilde{\Phi}, \cdots, \tilde{\Phi}]) \\ \Gamma_1 &= \text{blkdiag}([\tilde{\Gamma_1}, \cdots, \tilde{\Gamma_1}]), \\ \Gamma &= \text{blkdiag}([\tilde{\Gamma}, \cdots, \tilde{\Gamma}]) \end{aligned} \tag{1}$$

where $\Phi \in \mathbb{R}^{21x21} \Gamma_1, \Gamma \in \mathbb{R}^{21x7}$, and:

$$\begin{aligned} \tilde{\Phi} &= \begin{bmatrix} 1 & h & h^2/2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix} \\ \tilde{\Gamma} &= \begin{bmatrix} h^3/6 & h^2/2 & h \end{bmatrix}^T \\ \tilde{\Gamma_1} &= \begin{bmatrix} h^4/24 & h^3/6 & h^2/2 \end{bmatrix}^T \end{aligned}$$

Morever, the quadratic cost function chosen for solving this problem at time step k is:

$$V_k(U_k) = \sum_{j=k+1}^{k+N} x_j^T Q x_j + \sum_{j=k}^{k+N-1} u_j^T R u_j$$

where $U_k = [u_k, \cdots, u_{k+N-1}] \in \mathbb{R}^{7xN}$ is the input signal sequence over the control horizon of N steps that minimizes the cost function over the MPC prediction horizon of N steps, and $Q \in \mathbb{R}^{21x21}$ and $R \in \mathbb{R}^{7x7}$ are positive semi-definite weight matrices that penalize the controlled variables and inputs respectively.

This optimization problem is subject to discrete-time model of the system. Additionally, hard constraints are added to the model to ensure the robot reaches the desired configuration at the end of the trajectory. In addition to the hard constraint on final configuration, a set of linear constraint must be included to bound the admissible range of inputs and controlled variables. These were defined by referring to the Fanka Control Interface documentation.

$$\begin{aligned} x_{k+N} &= x_{goal} \\ x_{i,min} &\leq x_i \leq x_{i,max} \\ u_{i,min} &\leq u_i \leq u_{i,max} \end{aligned}$$

The choice of the cost function as convex, as well as the linear model and the convex constraints set makes the optimization problem convex which is beneficial for computation of the problem since if a solution exists, it is globally optimal. Finally, this convex problem is solved at every trajectory recalculation period. The sampling period, h, used in discretization is equal to:

$$h = \frac{T_F - t_k}{N}$$

where N is the number of discrete steps in prediction horizon, T_F is the final time where the goal state must be reached, and t_k is the time when the robot starts using the newly recalculated trajectory reference. The continuous-time prediction horizon of the problem will shrink since, as time goes by, t_k will increase while the T_F and N are constant, thus increasing the resolution of the computed trajectory as the goal state, x_{goal} is approached.

B. Obstacle avoidance using RRT and Potential Fields

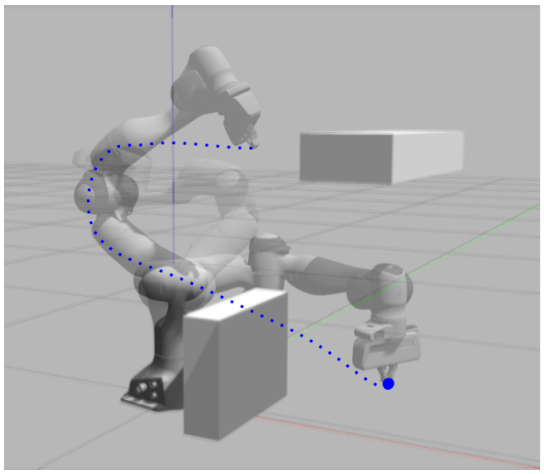

Fig. 1: *Trajectory optimization using RRT and MPC*

After implementing a controller for our robot motion, we implemented dynamic obstacle interaction using Rapidly Exploring Random Trees (RRT). The RRT algorithm samples a random point in the reachable workspace. Then, it finds a point in the tree that is closest to the sampled random point. It then adds to the tree, a point in the direction of the sampled point at a fixed distance away. The original node on the tree is added as its parent. This process is repeated while ensuring that the nodes and the line joining them is obstacle free. This process is repeated until the newest node added to the tree is close to the goal. At this point, the nodes on the tree are back-tracked until the parent-child chain leads to the start position. This algorithm is very fast, and accurate but does not guarantee a path each time, and doesn't guarantee the most optimized path each time.

C. Hierarchical MPC

As the nodes can be sampled differently each time, the path generated can be different for RRT for the same initial and

final goal point. This can be useful for some industry applications, but most industry applications like automation lines and production workshops require an optimized trajectory along with an efficient control policy.

To tackle this issue, we propose an RRT+MPC pipeline for obstacle interaction/ avoidance. Initially, the planner uses RRT to generate multiple paths for the given start and goal point. Our MPC node that has an OSQP solver returns the optimal cost for each path using pydrake (drake.mit.edu). This allows us to optimize the RRT path and return the path with lowest cost value i.e most optimal cost path. This is not guaranteed to give the global optimal path, but provides a fast way to get an efficient path.

The second stage of our algorithm involves controling the robotic arm using the control outputs from MPC node. This makes the robot follow the efficient trajectory selected earlier in the stage. This hierarchy allows our algorithm to plan a path and get efficient control inputs for robotic manipulation. The results and metrics are provided in Section IV.

D. MPC based on Convex sub-regions

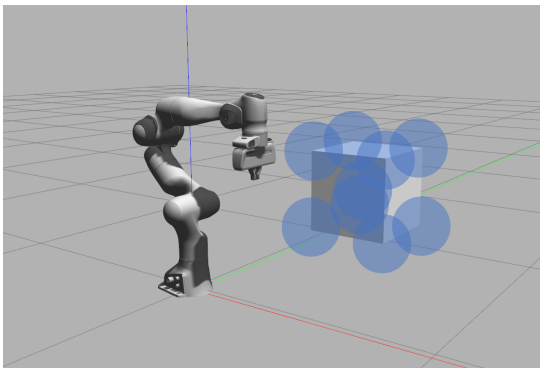

Fig. 2: *Defining obstacles as convex constraint*

To further expand upon our project, we tried to build an end-to-end MPC without the use of a planning algorithm but finding a collision free trajectory is a non-convex problem. Initially, we transformed the obstacles defined in 3D-XYZ coordinates to a 7 dimensional joint space vector. Later, inverse kinematics was used to generate around random robot arm poses that could possibly reach multiple key features (as seen in figure 2) on the obstacle and we randomly sampled 50 poses for each feature, with the assumption that our obstacles are convex. Furthermore, linear inequality constraints were imposed on the robot state configuration to help avoid these obstacles. But it became increasingly harder to solve given that the number of constraints would increase exponentially based on the nature of the obstacles and translating poses to join space would not be justifiable as there are unfathomable number of configurations in which you can hit an obstacle.

To overcome this issue, we reformulated our MPC to move to convex regions around the obstacle. Rather than working with an entire non-convex world, the areas surrounding the obstacle can be broken up into smaller convex areas. We then traverse between different convex regions to eventually reach our goal position.

IV. Results

We tested our MPC algorithm by iterating over 200 valid start and end joint configurations within the joint limits. It successfully converged for 197/200 configurations of which the convergence for first 100 examples are shown in figure 3 below.

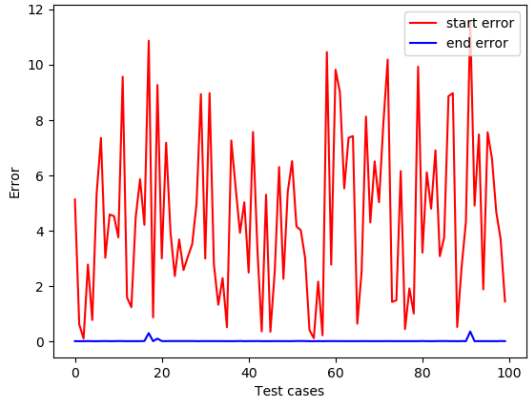

Fig. 3: *Initial vs final errors vs 100 initial and final seeds*

An example of one successful convergence is also shown in figure 4 where the error rates drops rapidly over time, given an initial q_0 and q_f.

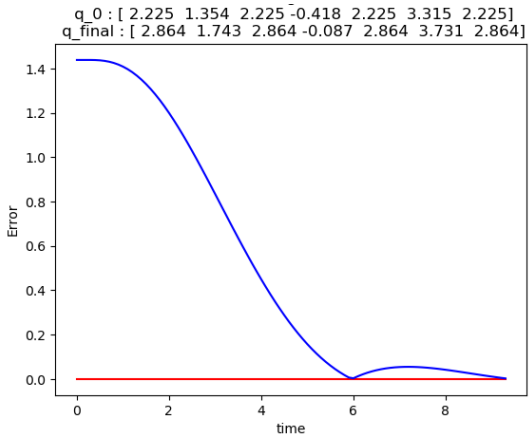

Fig. 4: *MPC Convergence Curve*

The control inputs for a given trajectory: joint angles in figure 5, velocity in figure 6 and acceleration in figure7 are further plotted over the next three graphs. The graphs depict a trajectory defined over multiple waypoints produced by the planning algorithm from a given start node to an end node.
We further evaluated the efficiency of the control policy determined by the MPC and RRT pipeline. We used 3 methods for sending ROS message commands to the robot using a custom function we developed; namely, position control, position and velocity control, position and velocity and acceleration control. Figure 8 shows the comparison between the three. While all

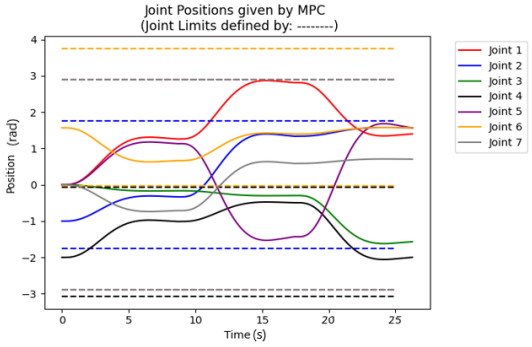

Fig. 5: *Joint Positions computed by MPC*

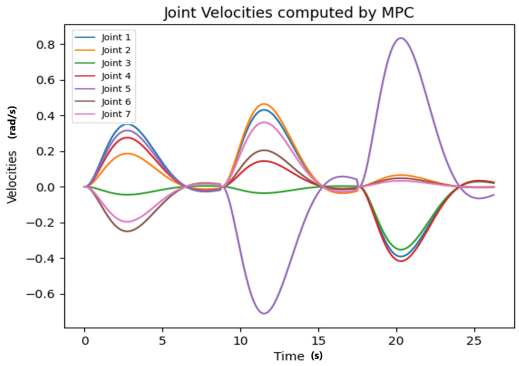

Fig. 6: *Joint velocities computed by MPC*

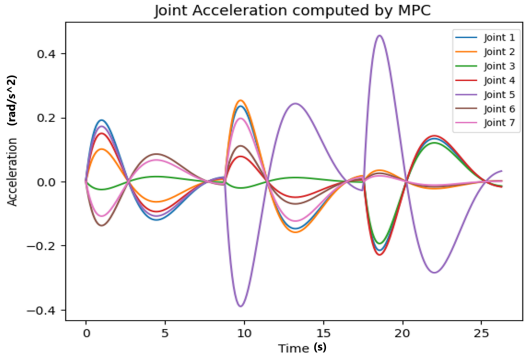

Fig. 7: *Joint Accelerations computed by MPC*

3 controls give comparable results in the path having simple obstacles, the difference becomes cogent in the complex map number 3. Position control takes approximately 90 seconds, whereas, the position, velocity and acceleration control takes approximately 20 seconds.

V. Challenges and Future Work

We tried to impose constraints directly in the MPC formulation to help the algorithm avoid obstacles without the need of RRT but where unsuccessful owing to the non-convexity of our problem. Redefining the state space vectors to include the joint positions defined in the Cartesian plane was not only difficult but also increased the complexity of the MPC problem.

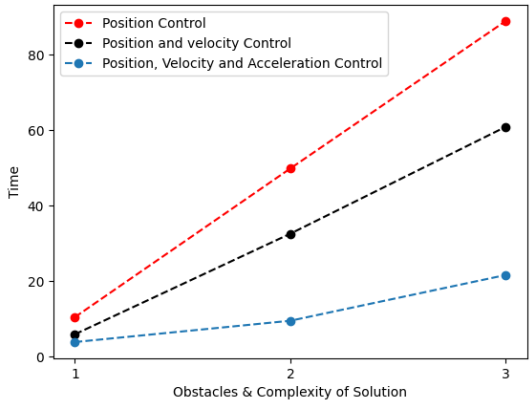

Fig. 8: *Time taken by different control algorithms to manipulate robot (in seconds)*

For future work, rather than using a Sequential Quadratic Programming methods, we can try using multiple shooting along with a recently proposed first order solver - Proximal averaged Newton-type method for optimal control (PANOC) [7]. PANOC is a matrix-free method for non-convex problems that only requires vector-vector operations and decoupling our single-shoot approach to a multiple-shooting approach can possibly exhibit fast convergence.

We currently have not used the jerk control inputs obtained from the MPC on the Franka Panda Arm and would like to implement that to obtain smoother trajectories.

VI. Acknowledgement

We thank Prof. Dr. Michael Posa and the teaching staff of MEAM 517 for providing us with this opportunity.

References

[1] N. Gafur, G. Kanagalingam, and M. Ruskowski, "Dynamic collision avoidance for multiple robotic manipulators based on a non-cooperative multi-agent game," 2021. [Online]. Available: https://arxiv.org/abs/2103.00583

[2] J. M. Salt Ducaju, B. Olofsson, A. Robertsson, and R. Johansson, "Joint stiction avoidance with null-space motion in real-time model predictive control for redundant collaborative robots," in *2021 30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, 2021, pp. 307–314.

[3] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning." [Online]. Available: https://arxiv.org/abs/1709.00627

[4] P. Bosscher and D. Hedman, "Real-time collision avoidance algorithm for robotic manipulators," in *2009 IEEE International Conference on Technologies for Practical Robot Applications*, 2009, pp. 113–122.

[5] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust mpc and neural network control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[6] M. M. Ghazaei Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Model predictive control for real-time point-to-point trajectory generation," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 972–983, 2019.

[7] A. S. Sathya, J. Gillis, G. Pipeleers, and J. Swevers, "Real-time robot arm motion planning and control with nonlinear model predictive control using augmented lagrangian on a first-order solver," in *2020 European Control Conference (ECC)*, 2020, pp. 507–512.