S.T.O.R.M: An Integrated Framework for Fast Joint Space Model-Predictive Control for Reactive Manipulation (Supplementary Material)

Mohak Bhardwaj[1,2], Balakumar Sundaralingam[1], Arsalan Mousavian[1], Nathan Ratliff[1], Dieter Fox[1,2], Fabio Ramos[1,3], Byron Boots[1,2]

[1]NVIDIA [2]University of Washington [3] University of Sydney

1 Cost Function Design

1.0.1 Reaching Goal Poses

Given the desired and current Cartesian poses for the end-effector ${}^{w}X_g, {}^{w}X_{ee} \in \mathbb{SE}(3)$ respectively in the world frame w, we compute a task-space cost term that penalizes their distance

$$\hat{c}_{\text{pose}}({}^{w}X_{ee}, {}^{w}X_g) = ||\alpha_1(I - {}^{w}R_g^{\top}\,{}^{w}R_{ee})||_2 + ||\alpha_2({}^{w}R_g^{\top}\,{}^{w}d_{ee} - {}^{w}R_g^{\top}\,{}^{w}d_g)||_2 \tag{1}$$

where ${}^{w}R_t, {}^{w}R_g$ are the rotation and translation part of the goal pose X_g, and ${}^{w}d_t, {}^{w}d_g$ are the rotation and translation part of the current end-effector pose. The weight vectors $\alpha_1, \alpha_2 \in \mathbb{R}^3$ allow us to weigh errors in different directions and orientations with respect to each other and can be set to different values to obtain qualitatively different behavior such as partial pose reaching or enforcing partial pose constraints. For example, we can set a high weight in α_2 along a desired axes to maintain the goal orientation throughout the motion of the robot as we demonstrate in our experiments.

1.0.2 Stopping for Contingencies

The finite horizon of MPC makes it myopic to events that can occur far out in the future, especially in dynamic environments. Thus, it is desirable to ensure that the robot can safely stop within the horizon in reaction to events that might be observed at timestep $H-1$. We encode this behavior by computing a time varying velocity limit $\dot{\theta}_{max} \in \mathbb{R}^H$ for every timestep in the horizon based on the maximum acceleration of the robot $\ddot{\theta}_{max}$ (or a user-specifed maximum acceleration) and the time available until $H-1$. This means the joint velocity of the robot must allow it to come to a stop at the end of the horizon by applying the max acceleration. Any state that exceeds this velocity is penalized by a cost which is expressed as

$$\dot{\theta}_{max} = S_u(1)\ddot{\theta}_{max}dt \qquad \hat{c}_{\text{stop}}(\dot{\theta}_t) = \begin{cases} ||\dot{\theta}_{max,t} - |\dot{\theta}|||_2 & \text{if } \dot{\theta}_{max,t} - |\dot{\theta}| > 0.0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $S_u(1)$ is an upper triangular matrix filled with 1.

1.0.3 Joint Limit Avoidance

Given minimum and maximum limits on joint state $\theta_{min}, \theta_{max}$ respectively, we penalize the joint state θ_t only if it exceeds a safety threshold defined by a k_{jl} ratio of its full range.

$$\hat{\theta}_{min} = \theta_{min} + k_{jl}(\theta_{max} - \theta_{min}) \qquad \hat{\theta}_{max} = \theta_{max} - k_{jl}(\theta_{max} - \theta_{min})$$

$$\hat{c}_{\text{joint}}(\theta_t) = \begin{cases} ||\theta_t - \hat{\theta}_{min}||_2 & \text{if } \theta_t < \hat{\theta}_{min} \\ ||\hat{\theta}_{max} - \theta_t||_2 & \text{else if } \theta_t > \hat{\theta}_{max} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\hat{\theta}_{min}, \hat{\theta}_{max}$ adds a safety threshold from the actual bounds of the robot. In our experiments, we chose $k_{jl} = 0.1$.

5th Conference on Robot Learning (CoRL 2021), London, UK.

1.0.4 Avoiding Cartesian Local Minima

The manipulability score describes the ability of the end-effector to achieve any arbitrary velocity from a given joint configuration. It measures the volume of the ellipsoid formed by the kinematic Jacobian which collapses to zero at singular configurations. Thus, to encourage the robot to optimize control policies that avoid future kinematic singularities, we employ a cost term that penalizes small manipulability scores [1, 2]

$$\hat{c}_{\text{manip}}(\theta_t) = \begin{cases} 1.0 - \sqrt{J(\theta_t)J(\theta_t)^\top}, & \text{if} \sqrt{J(\theta_t)J(\theta_t)^\top} < k_m \\ 0.0, & \text{otherwise} \end{cases} \tag{4}$$

where we choose $k_m = 0.05$ based on values obtained by Haviland and Corke [3] for the Franka Panda robot.

1.0.5 Self Collision Avoidance

Computing self-collision between the links of the robot has previously shown to be computationally expensive, especially when we want to compute for many joint configurations [4, 5]. Hence, similar to previous approaches [4, 5], we train a neural network that predicts the closest distance [1] between the links of the robot given a joint configuration (θ). One difference in our approach is the use of positional encoding (i.e.,$[\sin(\theta), \cos(\theta)]$) as we found this to improve the accuracy of the distance prediction [6]. Our network (which we call jointNERF) has three layers with $[256, 128, 64]$ neurons and ReLU activations. We compute a cost term as shown below,

$$\hat{c}_{\text{self-coll}}(\theta_t) = \max(0, \text{jointNERF}(\theta_t)) \tag{5}$$

1.0.6 Environment Collision Avoidance

Safe operation in unstructured environments requires a tight coupling between perception and control for collision avoidance. General gradient-based trajectory optimization and MPC approaches [7] rely on either known object shapes or pre-computed signed distance fields that provide gradient information for optimization. However, our sampling-based approach can handle discrete costs and as such we explore collision avoidance without using signed distance. We specifically use a learned collision checking function from Danielczuk *et al.* [5] that operates directly on raw pointcloud data. The method classifies if an object pointcloud pc_l is in collision or not with the pointcloud pc_{env} given the object's pose X_l. The cost can be written as,

$$\hat{c}_{\text{coll}}(pc_l, pc_{env}, X_l) = \begin{cases} 1, & \text{if } collision, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Finally our running cost function is given by

$$\hat{c}(x_t, u_t) = \alpha_p \hat{c}_{\text{pose}} + \alpha_s \hat{c}_{\text{stop}} + \alpha_j \hat{c}_{\text{joint}} + \alpha_m \hat{c}_{\text{manip}} + \alpha_c(\hat{c}_{\text{self-coll}} + \hat{c}_{\text{coll}}) \tag{7}$$

In our experiments we chose large values for $\alpha_j = 100.0$ and $\alpha_{coll} = 1000$ to enforce joint and collision avoidance constraints. For pose reaching, $\alpha_p = [150.0, 20.0]$ was used with smaller value on orientation as it only needs to be maintained at the goal. For enforcing orientation constraints while reaching goals, however, $\alpha_p = [100.0, 100.0]$ was chosen.

2 Real-Time Control Implementation

We implemented our MPC pipeline using PyTorch [8] with manipulator forward kinematics adapted from the open-source implementation provided by Sutanta *et al.* [9] and all cost terms and update equations implemented in a batched fashion. Further, multiprocessing is used to run MPC in a seperate process to avoid latency issues. Table 1 shows a timing comparison of our system running on a Titan RTX GPU against leading manipulator control approaches in literature.

[1] Distance is positive when two links are penetrating and negative when not colliding.

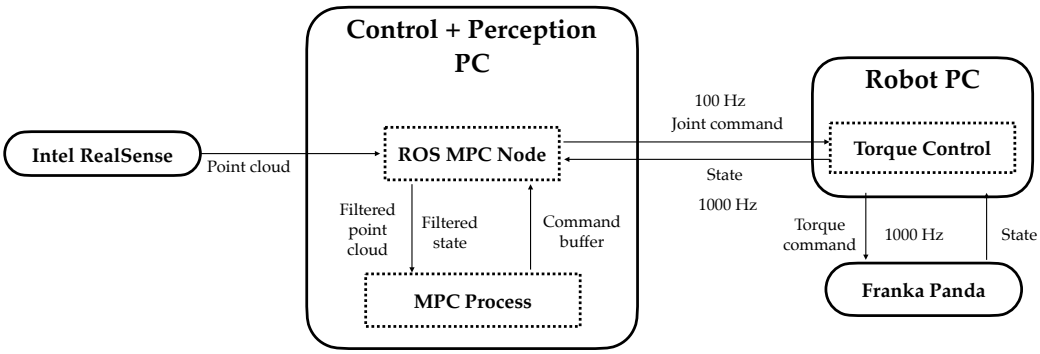


Figure 1: The compute graph shows the flow of information between the different components in our approach.

Table 1: Control Latency of methods that can handle collision avoidance in high dimensional systems is tabulated here. A more thorough description is available in Sec. 3.

Method	Latency (ms)	Horizon
OSC [10, 11, 3, 4]	1-17	1
Motion Planning [12, 13, 14]	140-1000	N/A
Custom Chip Motion Planning [15]	1	N/A
Gradient MPC [16, 17, 18]	20 - 140	$\leq$16
Sampling MPC [5]	100	40
Ours	10	30

2.1 Franka Panda Control System

The desired acceleration $\ddot{\theta}_t^d$ command from MPC is evaulated at 100Hz and integrated forward to obtain desired joint position (θ_t^d) and joint velocity $\dot{\theta}_t^d$ commands respectively. These commands are sent to a custom low-level torque controller that computes desired torque commands at 1000Hz to control the Franka robot,

$$\tau_t^{ff} = M(\theta_t)\ddot{\theta}_t^d + C(\theta_t)\dot{\theta}_t + K_p(\theta_t^d - \theta_t) + K_d(\dot{\theta}_t^d - \theta_t) \tag{8}$$

where $M(\theta)$ and $C(\dot{\theta})$ are the inertia and coriolis force matrices respectively provided by `libfranka` and K_p, K_d are gains for the position and velocity errors respectively. Fig. 1 shows the overall architecture of the control system.

2.2 State Estimation and Perception

We found the noise in the joint state read from `libfranka`, especially in joint velocities, to be prohibitive for precise control with MPC. In order to circumvent this issue we implemented a joint state filter that first predicts the state based on the previous commanded joint acceleration and then uses an exponential moving average filter to incorporate sensor readings.

For our perception setup, we use an Intel Realsense D455 Depth Camera placed at a fixed location in the workspace with a known camera-to-robot transform to obtain depth data in the form of point clouds. The raw point cloud is filtered to remove all the points lying inside the robot body to obtain the *scene* point cloud. The robot URDF is used to sample points along the robot body to create a *robot* point cloud. Both these point clouds are fed as input to SceneCollisionNet [5] for computing the collision cost in Eq. 6. Since we only consider static scenes for the purposes of this work, the *scene* pointcloud is computed once at the start of the run. For dynamic scenes, the pointcloud would need to be processed in real-time. We defer this to future work.

3 Related Work

Perception driven feedback control on high dimensional systems is a large field of research with several existing approaches [19]. Operation Space Controllers (OSC) are some of the fastest algorithms available for feedback control, with methods achieving control latency of 1-2 ms [10, 11, 3, 4]. However OSC methods rely heavily on a higher level planner for avoiding local minima (e.g.,obstacles).

A more global approach has been explored by reformulating standard motion planning methods to do feedback control via online replanning [20, 12, 13, 14, 15]. However most of these methods run at a slow rates on high dimensional systems, with control latencies between 140ms and 1000ms [20, 12, 13, 14]. Murray *et al.* [15] researched leveraging a custom chip to do fast parallel collision checking and use this with a PRM style planner to replan at 1ms for reaching Cartesian Poses in a semi-structured environment. Their chip based collision checker uses a complete pointcloud of the environment obtained by placing many cameras in the environment and combining their pointclouds. This is an highly unrealistic setting for real world manipulation in unstructured environments.

In the realm of Model predictive control approaches, only gradient based joint space MPC methods have shown to work on real manipulation systems as their control latency is in an acceptable range (20ms - 125ms) for feedback control [16, 17, 18]. Ishihara *et al.* [18] explore two stage hierarchical ILQR for fast MPC on a humanoid robot. Their two stage approach enables a very low control latency of 20ms. Erez *et al.* [17] explore ILQR on humanoid robots leveraging a simulator for the dynamics model. Fishman *et al.* [16] use gradient based MPC for finding trajectories for a manipulator while simultaneously predicting user intent in a human robot interaction setting. They solve the optimization problem leveraging Levenberg-Marquardt at a control latency of 140ms. Hogan and Rodriguez [21] explore gradient based MPC in the task space for planar non-prehensile manipulation leveraging a learned mode switcher to switch between different dynamic models. They are able to obtain a control latency of 5ms as their dynamic models are smooth.

Sampling-based methods have a rich history in MPC. Model-Predictive Path Integral Control (MPPI) [22] is one of the leading sampling-based MPC approaches that has shown great performance on real-world aggressive off-road autonomous driving by leveraging learned models [23] and GPU acceleration [24]. Wagener *et al.* [25] analyze MPC algorithms from the perspective of online learning and show connections between different methods such as Cross-Entropy method (CEM) and MPPI and have also demonstrated control rates of 40Hz with 1200 samples and a horizon of 2.5 seconds for off-road driving using GPU acceleration.

However, in the context of manipulation, sampling-based control in joint space has only been explored by optimizing in the joint position space without considering velocity and acceleration limits [5, 26, 27]. Danielczuk *et al.* [5] learn a collision classifier and do online replanning in joint space leveraging an inverse kinematic function to get a goal joint configuration and find straight line paths in joint space to reach the goal while avoiding collisions. Their approach doesn't handle different task spaces directly in the form of cost functions and also has a much larger control latency of 1000 ms. Hyatt *et al.* [26, 27] compare sampling based MPC with gradient based MPC on large dimensional robots with piecewise linear functions. They show that sampling based MPC can run at 200Hz (5ms) even with large number of dimensions due to it's parallelizability on the GPU. However, they do not explore collision avoidance or task space cost terms in their approach and leave it for future work. Pinneri *et al.* [28] provide a method for adding correlated noise to action samples in Cross-Entropy Method to reduce the number of particles required for obtaining good performance on high-dimensional control tasks. However, their multi-threaded CPU implementation is unable to achieve real-time performance. Their experiments are also limited to simulation with access to true dynamics.

Sampling based optimization has also been used for motion planning in high dimensional systems [29, 30, 31]. Kalakrishnan *et al.* [29] formulate planning as a stochastic trajectory optimization problem and plan over the joint position space to reach Cartesian positions while orientation constraints on the end-effector. They structure their co-variance matrix based on the finite difference matrix to sample delta joint positions that start and stop at 0 with a smooth profile in between. This sampling along with projecting the weighted action through this matrix, pushes the optimization towards a smooth trajectory for execution on the real robot. Kobilarov [30, 31] explored using cross-entropy optimization for motion planning and showed global convergence in very tight environments on quadrotors and simple planar environments.

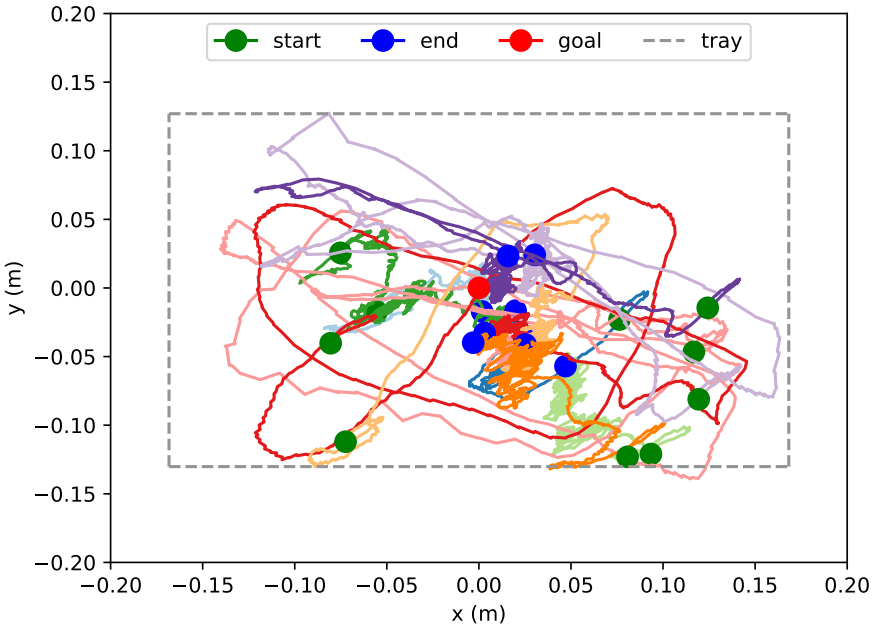


Figure 2: Trajectories of ball in tray frame for 10 different episodes of the dynamic balancing task. At the start of each episode the robot starts from the home configuration and the ball is placed at an arbitrary location on the tray by the human operator. Our control framework is able to achieve a median error of 3.9 cm. Note that due to perception errors, sometimes the center of the ball is detected to be outside the tray when near the edge.

4 Further Experimental Details

4.1 Dynamic Object Balancing

Experimental Setup: In this task, the real Franka Panda robot is required to balance a ball on tray grasped by the parallel jaw gripper. Every episode starts with the ball placed at an arbitrary location on the tray with the robot trying to center the ball without dropping it. Each episode lasts for 30 seconds after which the ball is placed at an arbitrary location by the human user. The position of the ball is measured at 30Hz using perception system that uses the RGBD input from a RealSense camera. The ball is detected from the RGB images using a blob-tracker and the corresponding depth is queried from the aligned depth image. The camera intrinsics are then used to compute the 3D coordinates. Fig. 3 shows a snapshot of the robot performing the task.

Ball Dynamics: We use a simple kinematic model of rolling on plane under acceleration due to gravity to predict the future positions and velocities of the ball in the frame of the tray given the end effector positions and orientations obtained from our arm model. In this simplified model, we do not explicitly account for friction or perform any system identification. In order to predict the future states of the ball given accelerations, we can leverage our tensorized forward model and maintain a high control frequency. Let the state of the ball in world frame (as input by the perception system) be denoted by

$$x^w_{ball} = [x, y, z] \qquad \dot{x}^w_{ball} = [\dot{x}, \dot{y}, \dot{z}]$$

Further, let the gravity vector in world frame be denoted by g^w. From forward kinematics, we can calculate the homogeneous transform of the end effector with respect to world frame as

$$T^w_{ee} = \left(\begin{array}{c|c} R^w_{ee} & d^w_{ee} \\ \hline 0 & 1 \end{array}\right)$$

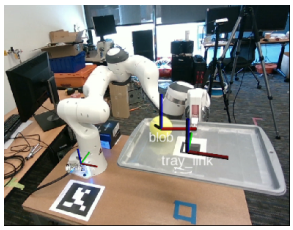


Figure 3: Snapshot of Ball Balancing Task

Table 2: Comparison of STORM with baselines on pose reaching. Reported values are median across 5 poses excluding the first pose which was unreachable by MMC. Max Joint Velocity is the maximum velocity achieved throughout the run.

Method	**Pos. Error** (mm)	**Quat. Error** (%)	**Joint Path Length** (rad)	**EE Path Length** (m)	**Max Joint Vel** (rad/s)
RRTCONNECT	0.5328	0.054	3.4527	1.0086	1.3547
RRTSTAR	0.3433	0.0393	1.7967	0.7626	0.7626
MMC	0.0661	0.00267923	2.3394	0.74351317	0.268
Ours	4.822	0.4619	3.41792	1.087688	1.2701

Given a batch of such end-effector poses (obtained from our tensorized arm model rollouts), we can calculate the accelerations due to gravity of the ball in the end effector frame as $g^{ee} = R_{ee}^{wT} g^w$ and $\ddot{x}_{obj}^{ee} = \left[g_x^{ee}, g_y^{ee}, 0\right]$. Here, we made the assumption that the ball does not lose contact with the plate which is reasonable at lower speeds. Similarly, the position and velocity of the ball in end effector frame are calculated as $x_{ball}^{ee} = T_{ee}^{w-1} * x_{ball}^{w}$ and $\dot{x}_{ball}^{ee} = R_{ee}^{wT} * \dot{x}_{ball}^{w}$ with $\dot{x}_{ball}^{ee}[2] = 0$. Finally, given the initial state and a batch of acceleration inputs in the end effector frame, we can simply employ our tensorized kinematic model to predict the future states of the ball.

Analysis: Fig. 2 shows a superimposed plot of the (x,y) trajectories ball with respect to the tray frame for 10 different episodes of the experiment. We observe that, even with our highly biased model and noise due to perception and state estimation, our MPC framework is able to achieve a median error of 3.9 cm in positioning the ball at the center of the tray. Moreover, the robot did not drop the ball in any episode. This experiment demonstrates the efficacy of MPC in correcting for model bias while maintaining reactivity and handling complex task constraints. In future work, we wish to leverage machine learning in the form of dynamics models and terminal Q-functions for MPC to achieve even better accuracy. An exciting extension of the task is to combine it with goal position reaching and handling multiple objects.

4.2 Reaching Cartesian Poses

We test the accuracy and path length of Cartesian pose reaching by selecting a sequence of six hard poses for the robot to reach. We compare against baseline sampling based planners RRTCONNECT and RRTSTAR using MOVEIT! and, Manipulability Motion Control [3], an operation space controller. In these experiments, the MOVEIT! and MMC baselines use the position and velocity controllers provided by `franka_ros` and `libfranka` respectively which have been extensively tuned and verified for accuracy. Hence, MMC represents the best accuracy achievable by a feedback controller.

One of these poses requires a large change in the end-effector's orientation as shown in Fig. 4. We found that MMC was unable to handle this pose and would repeatedly result in self collisions owing to its local nature and no consideration of self-collision avoidance, whereas both STORM and MOVEIT! baselines are able to reach the it. For the other poses, we found the accuracy of STORM to be worse than the baselines (shown in Table 2) although the path lengths and max joint velocities are comparable. Even though our method obtains millimeter level accuracy, the performance is limited by the lower level controller which was not tuned extensively. As future work, we intend to further tune and improve the lower level controller to overcome these issues.

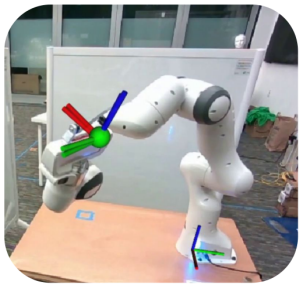


Figure 4: We show the robot trying to reach a very hard orientation Sec. 4.2. The MMC baseline is unable to reach this pose and continuously enters states of self collision due to its inability to account for collision avoidance while both MPPI and MOVEIT! baselines are able to reach the pose.

4.3 SceneCollisionNet Training

For the current experiments, we used a pre-trained SceneCollisionnet model (https://github.com/NVlabs/SceneCollisionNet) provided by the authors of [5]. The model was trained for table-top environments which is appropriate for our setting as well.

5 Ablation Studies

We performed extensive quantitative ablation studies for individual components of our system for the reaching task in simulation. We also present qualitative demonstrations of dynamic obstacle avoidance and effect of horizon on our website https://sites.google.com/view/manipulation-mpc/further-experimental-results.

5.1 Effect of Number of Particles

First, we study the effect of number of trajectories sampled per iteration of optimization (or particles) on the controller performance in simulation. For this experiment, 10 end-effector pose targets were used that require significant changes in orientation. Each episode is 700 timesteps long after which the manipulator is reset to the base orientation. The horizon is kept constant at 30 timesteps and all cost function weights are also fixed.

5.1.1 Position Accuracy

The box plot in Fig. 5a shows the median (solid line) with confidence interval (box) of position errors in the last 50 timesteps of every episode as a function of changing number of particles. We chose the last 50 timesteps to show the convergence of the controller to the goal. From the plot it can be seen that, increasing number of particles the controller is able to achieve more accurate median error with a tighter confidence interval.

5.1.2 Orientation Accuracy

The box plot in Fig. 5b shows the median (solid line) with confidence interval (box) of quaternion errors in the last 50 timesteps of every episode as a function of changing number of particles. Our framework achieves a confidence interval over quaternion errors within 5

5.1.3 Jerk

The box plot in Fig. 5c shows the median (solid line) with confidence interval (box) of the jerk in the robot motion over all the timesteps and episodes. Our sampling strategy generates smooth (low jerk) motions even with 200 particles.

5.1.4 Maximum Joint Velocity

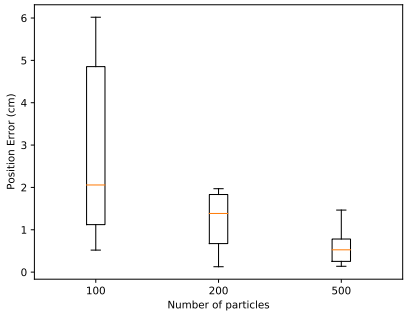
Fig. 5d demostrates that with increasing number of particles, our B-spline sampling strategy is able to ramp up the robot's joint velocity while maintaining low jerk.

5.2 Cost Terms

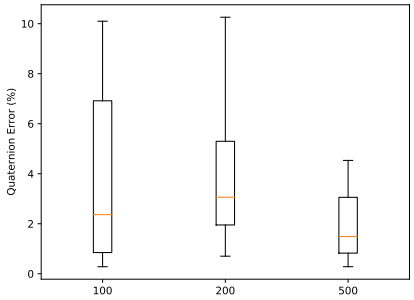
We study the effect of different cost terms on the controller performance. We test the constraint-based self collision and joint limit avoidance terms and behavior-based manipulability and stop costs.

5.2.1 Self-Collision Cost

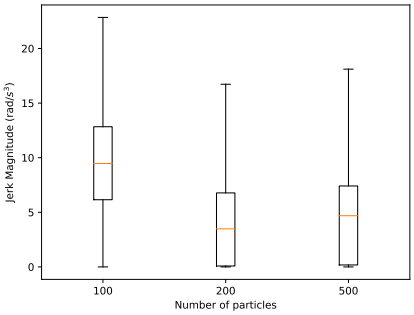
For this experiment, we chose 5 end effector target poses that are in collision with the robot and report the number of timesteps the robot spent in self collision. Table 3 shows that when the self collision cost is used, the robot never enters a state of self collision at the cost of not reaching the goal pose.



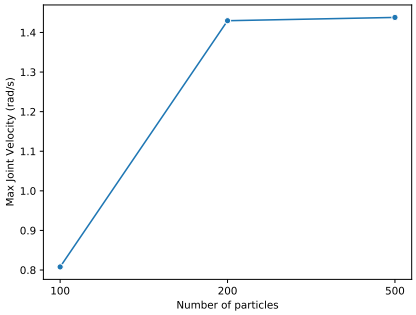
(a) Median position error with confidence bounds as a function of number of particles.



(b) Median quaternion error with confidence bounds as a function of number of particles.



(c) Jerk in robot motion as a function of number of particles.



(d) Maximum joint velocity reached for different number of particles.

Figure 5: Results for ablation study for number of sampled particles in MPPI.

Weight	Number of Self Collisions
0	2730
50	0
500	0
1000	0
5000	0

Table 3: Number of timesteps spent in self collision for changing weight on self collision cost

Weight	Number of Joint Limits Violations
0	1624
50	218
100	49
500	0
1000	0

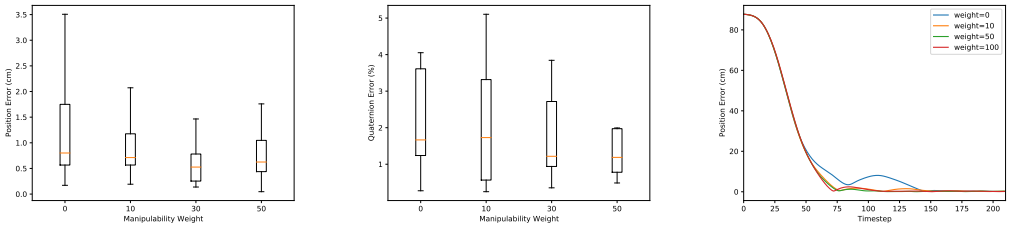
Table 4: Number of timesteps spent in joint limit violation for changing weight on joint limit cost

5.2.2 Joint Limit Avoidance Cost

For testing joint limit avoidance 10 end-effector targets with 500 particles were used with a varying weight on the joint limit cost. Table 4 shows that as the weight on the joint limit avoidance cost is increased, the number of violations steadily decreases, going to zero for a large weight of 500 and above.

5.2.3 Manipulability Cost

The manipulability cost acts as a regularizer to keep the manipulator away from singular configurations. The box plots in Fig. 6a and Fig. 6b show the median value with confidence bounds of the position and orientation errors over the last 50 timesteps in 10 different pose reaching runs. We chose the last 50 timesteps and not just the last timestep to test the convergence to the goal. Here we see that as the weight on the manipulability cost is increased, the pose reaching accuracy improves. However, after a certain threshold, the manipulability cost interferes with pose reaching and the position accuracy decreases. Maintaining high manipulability allows the robot to reach different end effector orientations accurately.

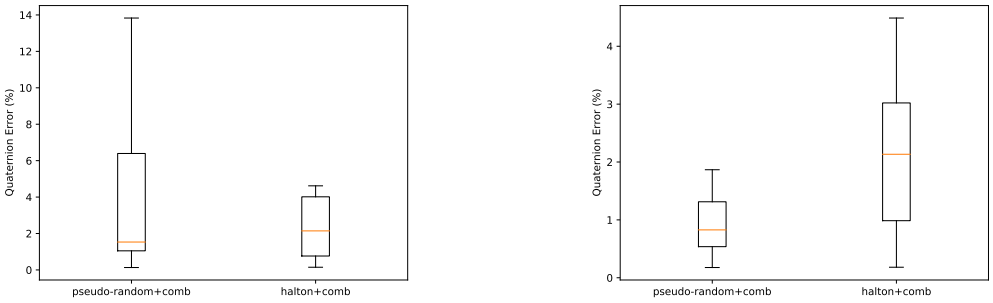


(a) Median position error with confidence bounds as a function of manipulability cost weight.

(b) Median quaternion error with confidence bounds as a function of manipulability cost weight.

(c) Position error over time for changing stop cost weight. Higher weight avoids oscillation near goal.

Figure 6: Results for ablation study for behavior based manipulability and stop costs.



(a) 100 particles .

(b) 500 particles

Figure 7: Median quaternion error with confidence bounds for psedurandom and Halton sampling with 100 and 500 particles

5.2.4 Stop Cost

The stop cost penalizes joint velocities that are too high for the robot to safely stop within horizon based on a maximum acceleration threshold. An important consequence of this term is that it allows the robot to smoothly stop at the goal even with a short horizon. We demonstrate this effect in the plot in Fig. 6c where a lower weight on the stop cost leads to undesirable oscillations near the goal.

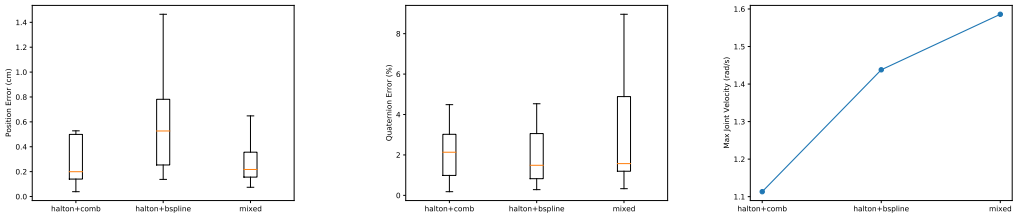
5.3 Sampling Strategy

5.3.1 Pseudo-random vs Halton Sampling

Halton Sampling is a Quasi Monte-Carlo method that provides better coverage of the action space as compared to pseudo-random sampling that exhibits and undesired clustering of samples. We study the improvement gain by using Halton sampling in a low particle regime. The plots in Fig. 7 show the quaternion errors achieved by pseudo random and Halton samples with 100 and 500 particles respectively. With lesser number of particles, pseudo-random sampling with comb filter is unable to achieve a quaternion error with a confidence interval within 5% whereas Halton sampling achieves less than 5% quaternion error with both 100 and 500 particles.

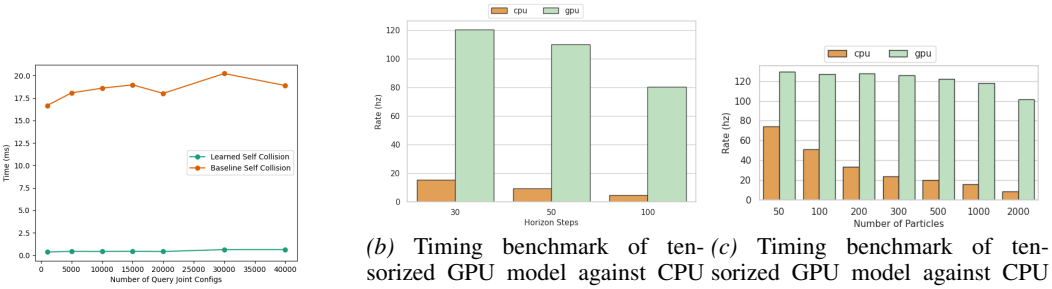
5.3.2 Comb Filtering v/s B-Splines

Fig. 9 shows a comparison of using comb filtering versus bsplines in conjunction with Halton sampling. Comb filtering smoothens out the sampled trajectories using user defined filtering coefficients. In order to ensure smooth (low jerk) motions a very strong filtering is required which prevents the robot from ramping up this velocity. Fitting B-Splines to sampled actions is able to ensure smooth acceleration profiles which allows the robot to smoothly ramp up the velocity but comes at the price of reduced pose accuracy. However, since our sampling strategy allows us to arbitrarily mix different kinds of trajectories, we create and test a hybrid sampling strategy (denoted as ”mixed”) with



(a) Position Error *(b)* Quaternion Error *(c)* Max Joint Velocity

Figure 8: Results for the comparison of pose reaching accuracy and max joint velocity achieved for different smoothing techniques in conjunction with Halton sampling.



(a) Timing benchmark of learned versus baseline self-collision.

(b) Timing benchmark of tensorized GPU model against CPU baseline as a function of horizon with 500 particles.

(c) Timing benchmark of tensorized GPU model against CPU baseline as a function of number of particles with horizon of 30 steps.

Figure 9: Timing benchmarks of different key components

a ratio of 0.6 and 0.4 of B-Spline and comb filtering. This provides comparable accuracy to comb filtering while also achieving high joint velocities.

5.4 Timing Benchmarks

5.4.1 Learned v/s Baseline Self Collision Detection

We quantify the computational gains obtained by using a learned self collision detector. In Fig. 9a we present a timing benchmark for an increasing batch size of query configurations. The learned function is over 40x faster on average than baseline self collision detection that uses forward kinematics to compute link poses and calculates minimum distance between them. Further, the learned self-collision detector maintains a very low latency of 0.4-0.6ms even for large batch sizes.

5.4.2 Speedup from Tensorized Forward Model on GPU

The timing benchmark in Fig. 9b and Fig. 9c show the computational gains from our tensorized GPU implementation of forward model versus a CPU baseline for varying horizon and number of particles used in MPC respectively. Fig. 9b shows that our model is over 5x faster than CPU baseline for different horizons with 500 particles. In Fig. 9c, we can see that the our model maintains similar latency for increasing number of particles owing to it being completely tensorized.

References

[1] C. A. Klein and B. E. Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The international journal of robotics research*, 6(2):72–83, 1987.

[2] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann. Manipulability analysis. In *2012 12th ieee-ras international conference on humanoid robots (humanoids 2012)*, pages 568–573. IEEE, 2012.

[3] J. Haviland and P. Corke. A purely-reactive manipulability-maximising motion controller. *arXiv e-prints*, pages arXiv–2002, 2020.

[4] D. Rakita, B. Mutlu, and M. Gleicher. RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi:10.15607/RSS.2018.XIV.043.

[5] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox. Object rearrangement using learned implicit collision functions. *arXiv preprint arXiv:2011.10726*, 2020.

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

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

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[9] G. Sutanto, A. Wang, Y. Lin, M. Mukadam, G. Sukhatme, A. Rai, and F. Meier. Encoding physical constraints in differentiable newton-euler algorithm. volume 120 of *Proceedings of Machine Learning Research*, pages 804–813, The Cloud, 10–11 Jun 2020. PMLR. URL http://proceedings.mlr.press/v120/sutanto20a.html.

[10] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger. Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom. *IEEE Robotics & Automation Magazine*, 19(2):20–33, 2012.

[11] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff. Rmpflow: A computational graph for automatic motion policy generation. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 441–457. Springer, 2018.

[12] L. G. Torres, A. Kuntz, H. B. Gilbert, P. J. Swaney, R. J. Hendrick, R. J. Webster, and R. Alterovitz. A motion planning approach to automatic obstacle avoidance during concentric tube robot teleoperation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2361–2367. IEEE, 2015.

[13] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

[14] K. V. Alwala and M. Mukadam. Joint sampling and trajectory optimization over graphs for online motion planning. *arXiv preprint arXiv:2011.07171*, 2020.

[15] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. D. Konidaris. Robot motion planning on a chip. In *Robotics: Science and Systems*, 2016.

[16] A. Fishman, C. Paxton, W. Yang, D. Fox, B. Boots, and N. Ratliff. Collaborative Behavior Models for Optimized Human-Robot Teamwork. *arXiv e-prints*, art. arXiv:1910.04339, Oct. 2019.

[17] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov. An integrated system for real-time model predictive control of humanoid robots. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 292–299, 2013. doi:10.1109/HUMANOIDS.2013.7029990.

[18] K. Ishihara, T. D. Itoh, and J. Morimoto. Full-body optimal control toward versatile and agile behaviors in a humanoid robot. *IEEE Robotics and Automation Letters*, 5(1):119–126, 2019.

[19] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018. doi:10.1109/LRA.2018.2795645.

[20] A. Kuntz, C. Bowen, and R. Alterovitz. Fast anytime motion planning in point clouds by interleaving sampling and interior point optimization. In *Robotics Research*, pages 929–945. Springer, 2020.

[21] F. R. Hogan and A. Rodriguez. Reactive planar non-prehensile manipulation with hybrid model predictive control. *The International Journal of Robotics Research*, 39(7):755–773, 2020.

[22] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[23] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[24] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[25] N. Wagener, C.-A. Cheng, J. Sacks, and B. Boots. An online learning approach to model predictive control. 2019.

[26] P. Hyatt, C. S. Williams, and M. D. Killpack. Parameterized and gpu-parallelized real-time model predictive control for high degree of freedom robots. *arXiv preprint arXiv:2001.04931*, 2020.

[27] P. Hyatt and M. D. Killpack. Real-time nonlinear model predictive control of robots using a graphics processing unit. *IEEE Robotics and Automation Letters*, 5(2):1468–1475, 2020.

[28] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius. Sample-efficient cross-entropy method for real-time planning. *arXiv preprint arXiv:2008.06389*, 2020.

[29] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011. doi:10.1109/ICRA.2011.5980280.

[30] M. Kobilarov. Cross-entropy randomized motion planning. In *Robotics: Science and Systems*, volume 7, pages 153–160, 2012.

[31] M. Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012.