2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Grasp-MPC: Closed-Loop Visual Grasping via Value-Guided Model Predictive Control

Jun Yamada[1,2] Adithyavairavan Murali[2] Ajay Mandlekar[2]
Clemens Eppner[2] Ingmar Posner[1] Balakumar Sundaralingam[2]

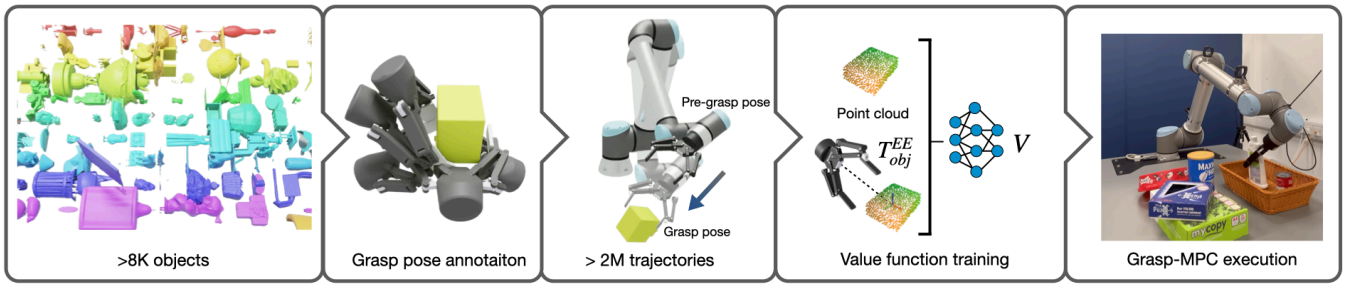


Fig. 1: *Grasp-MPC* overview. A large-scale synthetic grasp trajectory dataset is generated in simulation using a motion planner, collecting trajectories between pre-grasp and ground-truth grasp poses for 8K Objaverse objects. A value function is trained with sparse cost labels using the target object's point cloud and the end-effector pose relative to its center. The learned value is used as a cost in an MPC framework, enabling robust grasping of novel objects in cluttered environments.

***Abstract*— Grasping of diverse objects in unstructured environments remains a significant challenge. Open-loop grasping methods, effective in controlled settings, struggle in cluttered environments. Grasp prediction errors and object pose changes during grasping are the main causes of failure. In contrast, closed-loop methods address these challenges in simplified settings (e.g., single object on a table) on a limited set of objects, with no path to generalization. We propose *Grasp-MPC*, a closed-loop 6-DoF vision-based grasping policy designed for robust and reactive grasping of novel objects in cluttered environments. *Grasp-MPC* incorporates a value function, trained on visual observations from a large-scale synthetic dataset of 2 million grasp trajectories that include successful and failed attempts. We deploy this learned value function in an MPC framework in combination with other cost terms that encourage collision avoidance and smooth execution. We evaluate *Grasp-MPC* on FetchBench and real-world settings across diverse environments. *Grasp-MPC* improves grasp success rates by up to 32.6% in simulation and 33.3% in real-world noisy conditions, outperforming open-loop, diffusion policy, transformer policy, and IQL approaches. Videos and more at `http://grasp-mpc.github.io`.**

I. Introduction

Grasping is a foundational capability in robotics, serving as a prerequisite for physical interaction and subsequent complex manipulation tasks. However, despite decades of research, grasping remains unsolved, particularly when dealing with novel objects in cluttered scenes. State-of-the-art grasping methods can broadly be categorized into open-loop and closed-loop approaches, but both fall short of meeting the requirements of robust grasping in unstructured environments. Open-loop methods, which rely on grasp pose prediction models [1], [2], [3], [4], use motion planning to reach predicted grasp poses. Although demonstrating notable performance in grasping novel objects, these approaches are inherently unable to incorporate real-time feedback to adjust their goals, making them sensitive to grasp pose prediction errors and changes in object pose. On the other hand, closed-loop policies, including those based on reinforcement learning (RL) and imitation learning (IL), address some of these shortcomings by incorporating feedback during execution. However, prior approaches [5], [6], [7], [8] are often limited to simplified settings, such as tabletop environments, and exhibit poor generalization to novel objects, primarily due to the absence of large-scale grasp trajectory datasets for diverse objects. More importantly, safety considerations, such as collision avoidance in cluttered scenes, remain largely unaddressed during development.

To address these, we introduce *Grasp-MPC*, a framework that combines the strengths of open-loop and closed-loop methods for 6-DoF grasping in cluttered, novel-object settings. *Grasp-MPC* unifies model- and data-driven approaches by leveraging model predictive control (MPC) with a value function learned from data as a task cost function. *Grasp-MPC* uses a grasp prediction model and motion planning, similar to open-loop methods, to reach (noisy) pre-grasp poses. It addresses prediction errors and object pose changes by employing MPC for closed-loop execution, enabling real-time feedback while enforcing constraints such as collision avoidance and minimum jerk.

Furthermore, *Grasp-MPC* addresses a key challenge in applying MPC to grasping: designing a cost function that meaningfully captures grasp success. Traditional geometric cost functions based on distances to predicted grasp poses are sensitive to prediction errors and fail to exploit MPC's full closed-loop capability. To overcome this, *Grasp-MPC* leverages a vision-based value function, trained on

[1]Applied AI Lab, Oxford Robotics Institute, University of Oxford
Work done during internship at NVIDIA. [2]NVIDIA, USA
Correspondence to: `jyamada@robots.ox.ac.uk`

large-scale synthetic grasping trajectories with Objaverse objects [9] using visual observations and sparse success labels (Figure 1). This value function predicts the likelihood of grasp success and serves as a task cost in MPC, guiding the robot to explore the state space toward successful grasps.

To this end, this work makes the following contributions:

1) *Grasp-MPC*, a safe closed-loop visual grasping policy for novel objects in cluttered environments.
2) *Grasp-MPC* unifies model-based control and data-driven approach by integrating a learned grasp value function into an MPC framework, enabling reactive, constraint-aware grasp execution in dynamic and cluttered environments.
3) A large-scale synthetic grasp trajectory dataset comprising over 2M trajectories, 115M states, and 8515 unique Objaverse objects, supporting scalable learning of a generalizable value function.
4) Extensive real-world and simulated evaluations show that *Grasp-MPC* significantly outperforms SOTA open-loop and closed-loop methods.

II. Related Works

Grasping. A significant body of research has focused on learning grasp prediction models [1], [3], [2], [10], [11], typically paired with motion planning for execution. These methods often rely on synthetic datasets with grasp annotations [12], enabling scalable training in simulation. While effective for novel objects, open-loop approaches suffer from prediction errors and lack feedback integration, limiting real-world robustness. In contrast, closed-loop policies learned via RL [5], [6], [7], [8] and IL [13], [14] address these issues, but often face challenges with sample inefficiency, limited generalization, and perform only in a clean tabletop.

Prior work [15], [16], [14] combines motion planning with learned policies for obstacle-aware manipulation. FetchBench [14] uses a Transformer-based IL policy [17] with motion planning for grasping in clutter, but performance is limited by dataset size and diversity. In contrast, we generate a large-scale grasping trajectory dataset, $100\times$ larger than [14], enabling better generalization.

Model Predictive Control for Robotic Manipulation. Model Predictive Control (MPC)[18], [19], [20] is a powerful framework for robotic control, enabling closed-loop optimization using real-time feedback. However, applying MPC to grasping is challenging, as it requires a cost function that captures the nuances of grasp success [21]. Even with a target grasp pose from a prediction model, MPC remains vulnerable to errors, much like open-loop methods. Recent work integrates learning with MPC to tackle challenges by learning dynamics [22], [23], [24], [25], cost functions [26], [27], [25], [28], [21], and sampling distributions [29], [30]. Chen et al. [21] propose a distance-based value function using a predicted grasp pose, but this overlooks key factors for grasp success, leading to suboptimal results. CV-MPC [26] learns value ensembles from few demonstrations but rely on low-dimensional states, limiting generalization. In contrast, *Grasp-MPC* learns a value function from large-scale synthetic trajectories using point cloud observations and sparse success/failure labels, addressing these limitations.

Offline RL Offline RL [31], [32], [33] enables policy learning using offline datasets without environment interaction, leveraging both successful and failure trajectories to improve policies. However, policy extraction often represents a bottleneck in the learning process rather than value function estimation, as discussed in prior work [34]. On the other hand, *Grasp-MPC* does not need to extract the policy from the learned value function because *Grasp-MPC* uses MPC as a policy that can explore and exploit states to minimize the learnt cost represented by the value function.

III. Preliminaries

Problem Formulation. We formulate grasping as a Partially Observable Markov Decision Process (POMDP). A trajectory is defined as $\tau = (\mathbf{x}_t, \mathbf{a}_t, c_t, \mathbf{x}_{t+1}, \mathbf{a}_{t+1}, c_{t+1}, \dots)$, where $\mathbf{x} \in \mathcal{X}$ are observations, $\mathbf{a} \in \mathcal{A}$ actions, and $c \in \mathcal{C}$ costs. The offline dataset contains N trajectories $\{\tau^i\}_{i=1}^N$, comprising both successes and failures. The objective is to minimize the discounted cumulative cost $J(\tau) = \sum t' = t^\infty \gamma^{t-t'} c(\mathbf{x}_t, \mathbf{a}_t)$, with discount factor γ.

Dynamics Model for Model Predictive Control. MPC samples action sequences and plans future states to select the next action that minimizes cost in real time, given a dynamics model. In this work, MPC optimizes for joint accelerations with Euler integration as the dynamics model. We assume that the real robot can track the generated joint position, velocity and acceleration targets accurately using a low-level controller, for example, an inverse dynamics controller, similar to prior work [35]. The environment is not explicitly modelled; thus, its dynamics, including those of objects, are unknown a priori.

IV. Approach

Grasp-MPC leverages a large-scale grasp trajectory dataset generated in a simulation (Section IV-A) to train a value function (Section IV-B). Then, the learned value function serves as a cost function within MPC (Section IV-C), enabling robust and safe closed-loop grasping that generalizes to novel objects. At deployment, *Grasp-MPC* is integrated with a grasp pose prediction model and motion planning to operate in cluttered environments (Section IV-D). Lastly, the implementation details of the value function and MPC are described in Section IV-E.

A. Data Generation

A diverse set of grasp trajectories is generated using 8515 Objaverse objects [9] (see Figure 1). *Grasp-MPC* operates from pre-grasp poses, estimated using an off-the-shelf grasp pose prediction model (Figure 2 (1)), which provides a rough estimate of viable grasp poses. Thus, each grasping trajectory is generated to move from a pre-grasp pose to the corresponding ground-truth grasp pose. The grasp pose annotations are from the GraspGen dataset [36], where candidates are generated via antipodal sampling, similar to ACRONYM [12]. For each object, 2K grasp transformations

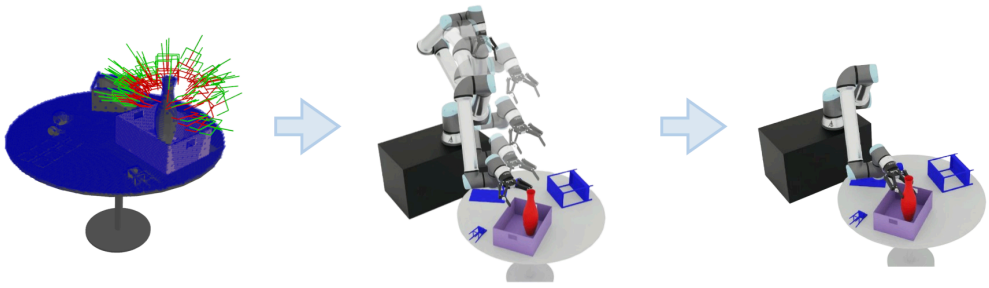
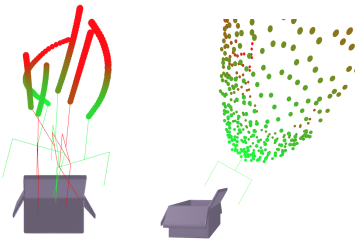


Fig. 2: *Grasp-MPC* Pipeline. *Grasp-MPC* seamlessly integrates off-the-shelf grasp prediction and motion planning with an MPC that incorporates a learned grasp value function, enabling grasping in cluttered scenes. The pipeline involves: (1) predicting grasp and pre-grasp poses using a fixed offset and filtering out in-collision poses via IK; (2) planning a trajectory to a collision-free pre-grasp pose; and (3) executing actions from the pre-grasp to grasp the object.

are uniformly sampled around the mesh. All sampled poses are evaluated in Isaac Sim to determine physical feasibility and are labeled accordingly as feasible or infeasible. Pre-grasp poses are derived by applying a fixed 15cm offset from each annotated grasp pose. To increase data coverage, we add random translation noise sampled from $\mathcal{U}(-0.04\text{cm}, 0.04\text{cm})$ and orientation noise from $\mathcal{U}(-0.04\pi, 0.04\pi)$.

We generate motions from the perturbed pre-grasp poses to the grasp poses using motion planning with CuRobo [37]. Trajectories that successfully reach physically feasible grasp poses are labelled as successful, whereas those planned toward infeasible grasp poses are labelled as failures. Incorporating both successful and failed trajectories allows the value function to learn grasp success likelihoods across the entire distribution of candidate poses (see Section IV-B). We do not validate these trajectories with simulation to accelerate data collection. Up to 256 trajectories are collected per object, with early termination if motion planning repeatedly fails. Each sample includes object poses T^{obj}_{world} and end-effector poses T^{EE}_{world}. In total, we collect $2,105,296$ trajectories ($115M$ data points), averaging 55 steps per trajectory, with lengths ranging from 31 to 233 steps. 70.2% of the collected trajectories reach a successful grasp, and the remaining trajectories are labeled as failures.



(a) (b)

Fig. 3: Visualization of the learned value function. (a) Costs along trajectories for feasible (green) and infeasible (red) grasps, with higher final costs for infeasible poses. (b) Estimated values around a target grasp pose by varying x and y while fixing orientation; red indicates higher values and green lower values.

B. Value Function Training for Grasping

Grasp-MPC uses a value function as a cost for guiding MPC in grasping. The value function takes as input $\mathbf{x}$ consisting of a segmented object point cloud and the end-effector pose relative to the point cloud centroid, T^{EE}_{obj}. To standardize inputs, we center the point cloud by subtracting its mean. This setup allows *Grasp-MPC* to generalize across the workspace using only local information. The collected trajectories are labeled with sparse costs, with terminal and near-terminal states in successful grasp trajectories labeled as 0, and all others as 1. In particular, the cost c_t as timestep t is defined as:

$$c_t = \begin{cases} 0 & |q_{goal,i} - q_{t,i}| \leq 5e^{-3},\ \forall i,\ \textit{and}\ \mathbb{1}_{feasible} = 1, \\ 1 & \text{Otherwise} \end{cases} \tag{1}$$

where $q_{t,i}$ and $q_{goal,i}$ are the i-th joint positions at time t and the goal, respectively, and $\mathbb{1}_{feasible}$ indicates whether the trajectory corresponds to a feasible grasp.

A value function $V(\mathbf{x}_t)$ is then trained to approximate the expected cost-to-go, defined as $V(\mathbf{x}_t) = \mathbb{E}_\tau[J(\tau)]$. The value function is trained using the Bellman error objective [38]:

$$y_t = c_t + \gamma V_{\phi'}(\mathbf{x}_{t+1}) \tag{2}$$

$$\ell(\phi; \mathbf{x}_t, c, \mathbf{x}_{t+1}) = \big(y_t - V_\phi(\mathbf{x}_t)\big)^2 \tag{3}$$

$$\phi^* = \arg\min_\phi\ \mathbb{E}_{(\mathbf{x}_t, c, \mathbf{x}_{t+1})}\left[\ell(\phi; \mathbf{x}_t, c, \mathbf{x}_{t+1})\right] \tag{4}$$

where y_t is the one-step target consisting of the immediate cost c_t and the discounted value of the next state $V_{\phi'}(\mathbf{x}_{t+1})$ from the target value function with exponential moving average of parameters ϕ. We set the discount factor to $\gamma = 0.99$, and the exponential moving average uses a rate of 5×10^{-3}. Figure 3 presents a visualization of the learned value function. The estimated value (cost) is lower (depicted in light green) around the target grasp pose, facilitating MPC in guiding the robot toward a successful grasp pose.

C. Integrating a Value Function as a Grasp Cost within MPC

Grasp-MPC uses learned value function as costs to guide MPC in minimizing grasping cost during online deployment.

The value function approximates the expected cost-to-go, which are integrated into the MPC objective to select control inputs. However, MPC may sample out-of-distribution actions, leading to unreliable cost estimates and reduced performance. To mitigate this, prior work [26], [39], [33] constrains MPC using pessimistic upper bounds derived from ensembles to avoid unsupported states. In our setting, ensembles do not yield performance improvements, as the large-scale synthetic data already covers a wide distribution of states (see Section V-D). Thus, in this work, the following objective is employed:

$$C_{grasp}(\mathbf{x}_{h\in H}) = \Sigma_{t'=t}^{t+H} \gamma^{t'-t} V_\theta(\mathbf{x}_{t'}) \tag{5}$$

We use Model Predictive Path Integral (MPPI) [40], a sampling-based MPC, implemented in CuRobo [37] to achieve real-time control. At each control step, MPPI samples N action sequences from a Gaussian distribution, rolls them out in parallel, evaluates the trajectories under a cost function, and computes an importance-weighted update to refine the control sequence. The first action of the optimized sequence is executed, and the horizon is shifted forward in a receding-horizon manner. *Grasp-MPC* augments standard MPC costs (e.g., minimum jerk, collision) with a value-based grasp cost. In particular, we integrate the grasp cost into CuRobo, and define the final cost as:

$$C_{\textit{Grasp-MPC}} = C_{curobo} + \omega C_{grasp} \tag{6}$$

where C_{curobo} is a set of CuRobo's default costs. ω is a weight for the pessimistic cost function, which we set to 1000. The cost term C_{curobo} in CuRobo consists of three main constraints: (1) joint limits on position, velocity, acceleration, and jerk; (2) self-collision avoidance; and (3) robot-world collision avoidance.

D. Grasp-MPC Deployment

At deployment, *Grasp-MPC* is combined with an open-loop grasp pose prediction model [1] and motion planner [37]. The grasp prediction model generates grasp poses for the target object. Pre-grasp poses are obtained by moving a negative distance along the gripper's approach vector. For real-world experiments, the standard 10cm offset distance [1], [41] is used. For simulation experiments in Fetchbench [14], 6cm offset is used, reduced from the standard 10cm due to insufficient clearance in many benchmark scenes, but increased from the original paper's 4cm, which is too restrictive. Feasible, collision-free pre-grasp poses are verified via a goalset motion plan, and the robot moves to the selected pre-grasp pose using the motion planner. Once positioned, *Grasp-MPC* grasps the object with feedback of the robot state, segmented object pointcloud, and the world signed distance field.

E. Implementation

While *Grasp-MPC* is compatible with any sampling-based MPC library, we specifically employ MPPI [40] implemented in CuRobo[37], a GPU-accelerated MPC framework, to enable fast and efficient real-time control. The value function

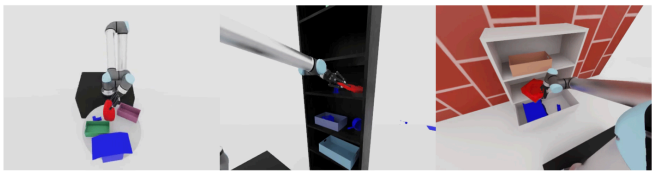


Fig. 4: Simulated environments. *Grasp-MPC* is extensively evaluated in FetchBench [14] environments.

consists of a PointNet++ [42] for point cloud input and an MLP for proprioception. Their outputs are concatenated and passed to an MLP head to output a value. A *softplus* activation ensures positive value predictions. Training uses both full and partial point clouds, with partial views rendered from randomly placed virtual cameras. Gaussian noise is added for robustness to real-world noise. The value function is trained with a mini-batch size of 1536, where 32 distinct object point clouds are sampled and 48 states are sampled for each object point cloud. The training procedure was conducted on a single RTX 4090 GPU for six days, using a learning rate of 1×10^{-4}.

V. Experimental Results: Simulation

In this section, we evaluate *Grasp-MPC* and competitive baselines in simulated environments designed to test grasping capabilities in cluttered settings. The experiments address: (1) How well does *Grasp-MPC* grasp unseen objects using ground-truth grasp poses? (2) How robust is it to perturbed ground-truth poses? (3) How does it perform with predicted grasp poses?

A. Experiment Setup

Grasp-MPC and baselines are evaluated in simulation using a UR10 robot with a Robotiq 2F-140 gripper. Evaluations are based on FetchBench [14] (Figure 4), adapted to use Isaac Sim for improved physics and closed-loop gripper modeling. All objects are novel and consist of both procedurally generated objects and the ACRONYM objects [12]. Three cameras capture point clouds, and the one with the greatest coverage of the target object is selected to provide observations to the policy. Experiments span 90 scenes (60 problems each), resulting in $5,400$ test cases. A $6cm$ offset is used to compute pre-grasp poses from the grasp poses.

Evaluation Metrics: The evaluation uses the metric *Grasp Success*, defined as lifting the object at least 1cm. Since the task involves motion planning to reach the pre-grasp pose, this metric accounts for motion planning failures by excluding trials where moving to the pre-grasp pose failed.

Baselines: *Grasp-MPC* is compared to the following:

- **Open-Loop Linear**: The end-effector moves linearly from the pre-grasp to the desired grasp pose. Operational Space Control [43] implemented in FetchBench is used.
- **Transformer Policy** A Transformer-based policy, inspired by the architecture used in OPTIMUS [17], and included as a baseline in FetchBench [14].

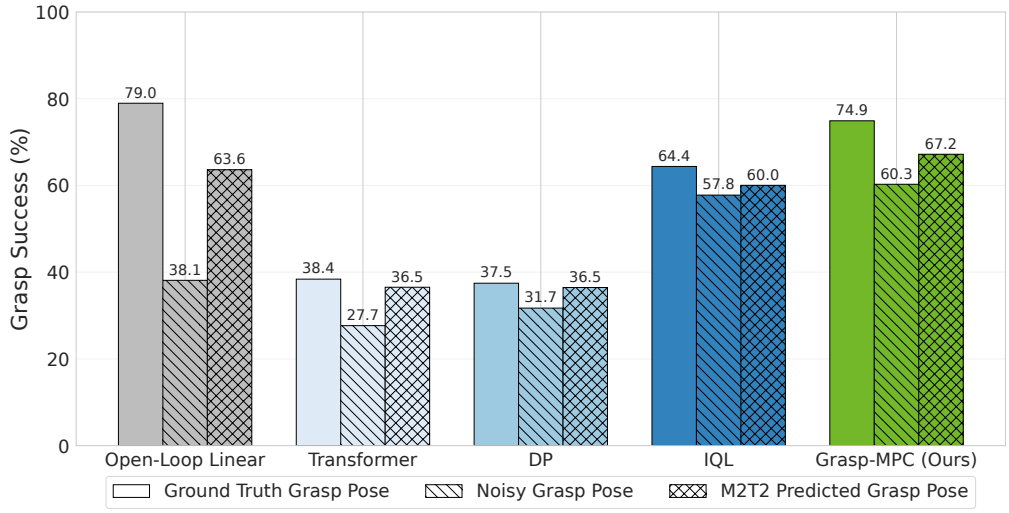


Fig. 5: Grasp performance comparison across different methods given ground truth, noisy, and M2T2-predicted grasp poses. Solid bars represent performance with ground truth grasp poses, diagonally hatched (///) bars show results with noisy perturbations ($\pm$2cm, $\pm$18deg), and cross-hatched (xx) bars indicate M2T2-predicted grasp poses. While *Grasp-MPC* maintains a high grasp success rate across all perturbations, Open-Loop Linear, an open-loop baseline, significantly drops with perturbed grasp poses.

- **Diffusion Policy (DP)**: A diffusion policy [44] that takes point cloud observations as input, which is considered one of the state-of-the-art IL approaches.
- **IQL**: A policy trained with Implicit Q-Learning (IQL) [32], a state-of-the-art offline RL method.

All methods, including *Grasp-MPC*, use CuRobo's motion planner to generate a trajectory from the robot's initial position to a pre-grasp pose. The robot then attempts to grasp the object using one of the above methods. IL policies are trained only on successful trajectories.

B. Grasp Execution with Grasp Pose Annotations

Grasp pose annotations for objects in the Fetchbench environments were generated for the Robotiq 2f-140 following ACRONYM [12]. Up to 200 kinematically valid, collision-free grasp poses are randomly sampled during evaluation, from which one is selected for the robot to move to the corresponding pre-grasp pose using a goalset motion planner in CuRobo [37].

***Grasp-MPC* nearly matches oracle performance while surpassing closed-loop baselines.** As shown in Figure 5, *Grasp-MPC* demonstrates competitive performance compared to *Open-Loop Linear*, an open-loop approach, which serves as an oracle baseline. *Grasp-MPC* achieves a grasp success rate of 74.9%, closely matching the oracle baseline *Open-Loop Linear* at 79.0%. Moreover, *Grasp-MPC* significantly outperforms closed-loop baseline methods.

Closed-loop baselines underperform due to suboptimal data and domain mismatch. IQL achieves a grasp success of 64.4%, below *Grasp-MPC* (73.6%), likely due to limitations in its policy extraction process [34]. IL-based approaches also perform poorly, likely because motion planning, while efficient for data collection, often yields suboptimal demonstrations. Additionally, the gap between the empty scene used for data collection and the object rich evaluation environments introduce MDP mismatches that further degrade performance.

***Grasp-MPC* is robust to noisy pre-grasp poses.** *Grasp-MPC* and baseline methods are also evaluated using ground-truth grasp poses perturbed with random noise sampled from $\mathcal{U}(-2\text{cm}, 2\text{cm})$ in translation and $\mathcal{U}(-18\text{deg}, 18\text{deg})$ in orientation. As shown in Figure 5, *Open-Loop Linear* fails to recover from these perturbations due to its open-loop nature, resulting in a 40% drop in performance. In contrast, *Grasp-MPC* achieves a 60.3% grasp success rate (14% drop), outperforming baseline closed-loop control policies.

C. Grasp Execution with a Grasp Pose Prediction Model

M2T2 [1] is used as an off-the-shelf grasp prediction model to generate target grasp poses. We attempted to train M2T2 using grasp pose annotations specific to the Robotiq gripper, but could not train an optimal grasp pose prediction model. We instead use the publicly available M2T2 model trained on Franka Panda gripper, adding a 10cm offset to adapt the predicted grasp poses for the Robotiq gripper. The top-512 scored grasp poses are selected.

***Grasp-MPC* maintains strong performance despite noisy grasp poses from M2T2, outperforming all baselines.** IL-based approaches perform poorly, achieving only 36.5% grasp success rate as shown in Figure 5. The standard *Open-Loop Linear* approach achieves 63.6%, dropping by 15% from using ground truth grasp poses. *Grasp-MPC* achieves 67.2% success, highest among methods, dropping only by 8% from using ground truth grasp poses. This result highlights *Grasp-MPC*'s robustness to prediction errors from an off-the-shelf grasp prediction model and shows promise for real-world deployment, which is evaluated in the next section.

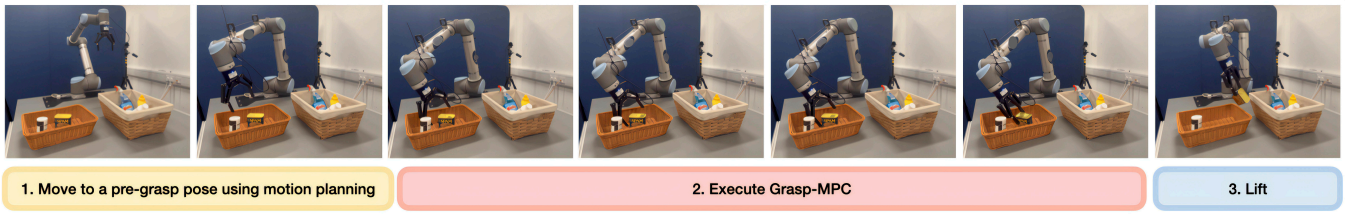


Fig. 6: *Grasp-MPC* execution in the Table Clutter scene. *Grasp-MPC* effectively grasps a novel object from the bin.

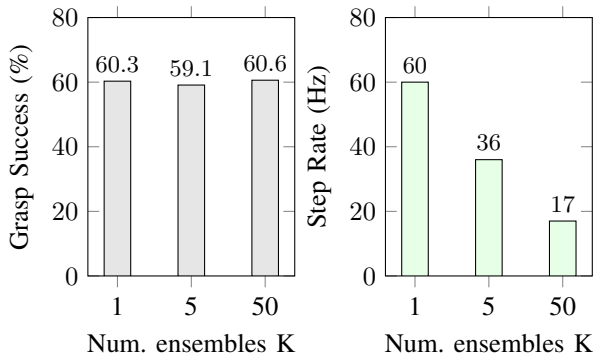


Fig. 7: Ablation on Ensemble showing Grasp Success and Step Rate for different number of ensembles K.

D. Ensemble Ablation Study

We present results using a single value function, as ensembles did not yield meaningful performance improvements. To validate this choice, we use the risk-averse objective for ensemble value functions, following the prior work [26]:

Impact of Value Function Ensembles. Figure 7 reports grasp success rates across all scenes. Using an ensemble improves performance by only 0.3%, suggesting that the dataset provides sufficient coverage for training a single value function. Although ensembles would promote safer behavior via pessimistic cost estimation, a single value function still generalizes well and enables effective grasping. The MPC step runs at $60hz$ for $K = 1$ and $17hz$ for $K = 50$, indicating the computational trade-off of using larger ensembles.

VI. Experimental Results: Real-World

Grasp-MPC is evaluated on novel objects using a UR10 robot arm with a Robotiq 2F-140 in real-world environments. The experiments are designed to address: (1) How effectively does *Grasp-MPC* grasp novel objects in challenging real-world environments compared to open-loop approaches? (2) How well does it adapt to dynamic perturbations when the target object pose changes after reaching the pre-grasp position?

A. Experiment Setup

Perception System. Two RealSense L515 cameras with known extrinsics capture 640×480 depth images to generate point cloud observations. Target object segmentation is performed using SAM-Track [45], which combines Grounding DINO [46] for detection and SAM [47] for segmentation,

Fig. 8: Representative real-world environments: (Left) Table Empty, (Middle) Table Clutter, and (Right) Shelf Clutter.

producing input point clouds for *Grasp-MPC*'s value function. To handle obstacles in cluttered scenes, NVBlox [48] is used to represent the environment for CuRobo's motion planning and MPC modules.

Success Criteria. Grasp success is defined as lifting the target object and returning the arm to its home position without dropping it during execution.

Grasp Pose Prediction. As in the simulated experiments, M2T2 predicts grasp poses, with a 10cm offset applied to adapt them for the Robotiq gripper.

Baselines. *Open-Loop Linear*, implemented in CuRobo-GraspAPI, is used for comparison, as it provides a simple and reliable grasp execution pipeline suitable for real-world deployment. Other closed-loop control policies are excluded from real-world experiments due to safety concerns, as they lack collision avoidance mechanisms and are prone to collide with surrounding obstacles such as a shelf or clutter. In contrast, *Grasp-MPC* minimizes collision and learned grasp task costs, enabling safe and effective operation in cluttered real-world environments.

B. Grasping Performance Across Different Scenes

Comprehensive evaluation across three environments with increasing complexity. We compare *Grasp-MPC* against *Open-Loop Linear* using CuRobo-GraspAPI across three progressively challenging environments: *empty tabletop*, *cluttered tabletop*, and *cluttered shelf scenes* as shown in Figure 8. Each environment utilizes 5 different objects, with distinct object sets for each environment to ensure comprehensive evaluation across varying complexity levels. For each object, three different poses are considered, and two evaluation trials per method are performed to assess consistency and reliability. In total, 30 trials are conducted for each environment.

Consistent outperformance over open-loop baselines across all environments. Figure 9 presents the grasp success rates of *Grasp-MPC* and *Open-Loop Linear* across different scenes. The open-loop baseline frequently fails to grasp target objects when the predicted grasp pose deviates from

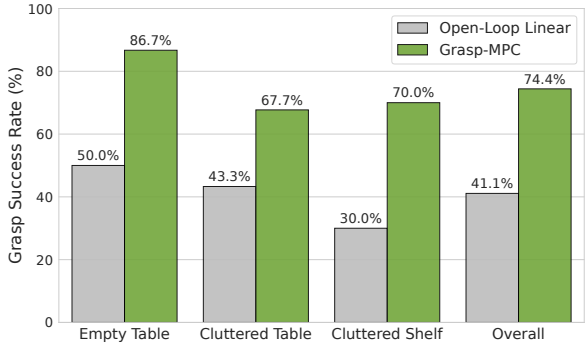


Fig. 9: Grasp performance comparison between Open-loop Linear and *Grasp-MPC*. *Grasp-MPC* consistently achieves higher grasp success rates across all scenes, highlighting its robustness in real-world environments.

the ideal configuration, as it cannot adapt during execution. In contrast, *Grasp-MPC* continuously adjusts the gripper pose to minimize the task grasp cost function while avoiding obstacles such as a shelf, demonstrating safe and robust grasp execution across all environments. *Grasp-MPC* outperforms the baseline in all scene types, with greater performance improvements observed in complex environments.

C. Real-Time Adaptation to Object Pose Perturbations

To demonstrate the benefits of *Grasp-MPC* as a closed-loop control policy, perturbations are added to the target object pose after the robot reaches the pre-grasp pose. Since open-loop approaches cannot handle such dynamic changes by design, we evaluate only *Grasp-MPC* in this experiment to assess its real-time adaptation capabilities.

Successful grasping despite large object perturbations during execution. We evaluate 5 objects with 6 trials per object, resulting in 30 trials total. Figure 10 illustrates an example rollout demonstrating *Grasp-MPC*'s adaptability. *Grasp-MPC* achieves a 60% success rate, demonstrating its capability to adapt in real-time even when large perturbations are applied to the target object pose. *Grasp-MPC* exhibits this global behavior while having trained on grasping motions that are only 15cm away from a grasp. We are planning on training with larger grasping motions in the future to study if we can get even higher success at this task.

VII. Discussion

This work presents *Grasp-MPC*, a 6-DoF closed-loop grasping policy for novel objects in cluttered environments. The approach learns a value function from both successful and failed grasping trajectories, which is subsequently integrated into an MPC framework to generate actions during deployment. Through its modular design, *Grasp-MPC* can address deployment-time constraints such as clutter without requiring retraining. Moreover, leveraging both successful and failed trajectories enhances data efficiency.

Grasp-MPC is validated on the simulated benchmark FetchBench [14] across 5,400 grasping problems, demonstrating superior performance compared to IL methods. It also outperforms open-loop planning-based approaches in scenarios with noise in the grasp pose or when the grasp pose was provided by a learned model (e.g., M2T2). On a real robotic system, *Grasp-MPC* achieves a 30% higher success rate than a planning-based approach in cluttered table-top and shelf settings, despite being trained exclusively on empty scenes. These results demonstrate the ability of the method to operate with noisy point clouds and to handle real-world contact physics without relying on physically simulated training data.

While shown to be effective, *Grasp-MPC* presents several limitations that highlight opportunities for future improvement and extension. First, although higher success rates are achieved compared to existing methods, performance in real-world deployments does not yet reach 100%. We hypothesize that leveraging physics simulation during data generation to generate success/failure labels will improve the performance of *Grasp-MPC*. *Grasp-MPC* can also be readily finetuned with real-world data as only success/failure labels are required for trajectories. Both of these explorations are left to future work.

A further limitation is that validation is restricted to grasping tasks. Although the approach could extend to other manipulation tasks given suitable demonstration pipelines [49], evaluating value function learning across diverse tasks is left for future work.

ACKNOWLEDGMENT

This work was supported by a UKRI/EPSRC Programme Grant [EP/V000748/1]. We acknowledge the use of the University of Oxford Advanced Research Computing (ARC) and SCAN facilities in carrying out this work. We would like to thank Stan Birchfield for providing feedback on this paper.

References

[1] W. Yuan, A. Murali, A. Mousavian, and D. Fox, "M2t2: Multi-task masked transformer for object-centric pick and place," 2023.

[2] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," 2019, pp. 2901–2910.

[3] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes." IEEE, 2021, pp. 13 438–13 444.

[4] J. Carvalho, A. T. Le, P. Jahr, Q. Sun, J. Urain, D. Koert, and J. Peters, "Grasp diffusion network: Learning grasp generators from partial point clouds with diffusion models in so(3)xr3," *arXiv preprint arXiv:2412.08398*, 2024.

[5] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation." PMLR, 2018, pp. 651–673.

[6] L. Wang, Y. Xiang, W. Yang, A. Mousavian, and D. Fox, "Goal-auxiliary actor-critic for 6d robotic grasping with point clouds." PMLR, 2022, pp. 70–80.

[7] R. Singh, A. Allshire, A. Handa, N. Ratliff, and K. Van Wyk, "Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands," *arXiv preprint arXiv:2412.01791*, 2024.

[8] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. V. Wyk, "DextrAH-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics," 2024.

[9] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre *et al.*, "Objaverse-xl: A universe of 10m+ 3d objects," vol. 36, 2024.

[10] K. R. Barad, A. Orsula, A. Richard, J. Dentler, M. Olivares-Mendez, and C. Martinez, "Graspldm: Generative 6-dof grasp synthesis using latent diffusion models," *IEEE Access*, 2024.

Fig. 10: *Grasp-MPC* execution for moving objects. *Grasp-MPC* adapts in real time to track and grasp moving target objects, capabilities that open-loop approaches lack.

[11] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," 2020, pp. 11 441–11 450.

[12] C. Eppner, A. Mousavian, and D. Fox, "Acronym: A large-scale grasp dataset based on simulation." IEEE, 2021, pp. 6222–6227.

[13] S. Song, A. Zeng, J. Lee, and T. A. Funkhouser, "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations," vol. 5, pp. 4978–4985, 2019.

[14] B. Han, M. Parakh, D. Geng, J. A. Defay, G. Luyang, and J. Deng, "Fetchbench: A simulation benchmark for robot fetching," in *Conference on Robot Learning*. PMLR, 2025, pp. 3053–3071.

[15] J. Yamada, Y. Lee, G. Salhotra, K. Pertsch, M. Pflueger, G. Sukhatme, J. Lim, and P. Englert, "Motion planner augmented reinforcement learning for robot manipulation in obstructed environments." PMLR, 2021, pp. 589–603.

[16] J. Yamada, J. Collins, and I. Posner, "Efficient skill acquisition for insertion tasks in obstructed environments," in *Proceedings of the 6th Annual Learning for Dynamics and Control Conference*, vol. 242. PMLR, 2024, pp. 615–627.

[17] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, "Imitating task and motion planning with visuomotor transformers," 2023.

[18] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, pp. 1608 – 1639, 2010.

[19] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," 2016, pp. 1433–1440.

[20] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," 2018, pp. 1–9.

[21] Y.-C. Chen, A. Murali, B. Sundaralingam, W. Yang, A. Garg, and D. Fox, "Neural motion fields: Encoding grasp trajectories as implicit value functions," *arXiv preprint arXiv:2206.14854*, 2022.

[22] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," 2017, pp. 2786–2793.

[23] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," vol. 28, 2015.

[24] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 09–15 Jun 2019, pp. 2555–2565.

[25] N. A. Hansen, H. Su, and X. Wang, "Temporal difference learning for model predictive control." PMLR, 2022, pp. 8387–8406.

[26] N. Jawale, B. Boots, B. Sundaralingam, and M. Bhardwaj, "Dynamic non-prehensile object transport via model-predictive reinforcement learning," 2024.

[27] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, "Value function approximation and model predictive control," in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2013, pp. 100–107.

[28] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," *arXiv preprint arXiv:1811.01848*, 2018.

[29] J. Sacks and B. Boots, "Learning sampling distributions for model predictive control." PMLR, 2023, pp. 1733–1742.

[30] J. Yamada, S. Zhong, J. Collins, and I. Posner, "D-cubed: Latent diffusion trajectory optimisation for dexterous deformable manipulation," 2024.

[31] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[32] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[33] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," vol. 33, pp. 1179–1191, 2020.

[34] S. Park, K. Frans, S. Levine, and A. Kumar, "Is value learning really the main bottleneck in offline rl?" *arXiv preprint arXiv:2406.09329*, 2024.

[35] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation." PMLR, 2022, pp. 750–759.

[36] A. Murali, B. Sundaralingam, Y.-W. Chao, J. Yamada, W. Yuan, M. Carlson, F. Ramos, S. Birchfield, D. Fox, and C. Eppner, "Graspgen: A diffusion-based framework for 6-dof grasping with on-generator training," *arXiv preprint arXiv:2507.13097*, 2025.

[37] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos *et al.*, "Curobo: Parallelized collision-free minimum-jerk robot motion generation," *arXiv preprint arXiv:2310.17274*, 2023.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[39] C.-A. Cheng, T. Xie, N. Jiang, and A. Agarwal, "Adversarially trained actor critic for offline reinforcement learning," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 17–23 Jul 2022, pp. 3852–3878.

[40] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," 2016, pp. 1433–1440.

[41] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics (T-RO)*, 2023.

[42] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[43] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[44] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2023.

[45] Y. Cheng, L. Li, Y. Xu, X. Li, Z. Yang, W. Wang, and Y. Yang, "Segment and track anything," *arXiv preprint arXiv:2305.06558*, 2023.

[46] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[47] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[48] A. Millane, H. Oleynikova, E. Wirbel, R. Steiner, V. Ramasamy, D. Tingdahl, and R. Siegwart, "nvblox: Gpu-accelerated incremental signed distance field mapping," *arXiv preprint arXiv:2311.00626*, 2024.

[49] C. Garrett, A. Mandlekar, B. Wen, and D. Fox, "Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment," 2024.