

Online Trajectory Replanner for Dynamically Grasping Irregular Objects

Minh Nhat Vu¹, Florian Grander², Anh Nguyen³

Abstract—This paper presents a new trajectory replanner for grasping irregular objects. Unlike conventional grasping tasks where the object’s geometry is assumed simple, we aim to achieve a “dynamic grasp” of the irregular objects, which requires continuous adjustment during the grasping process. To effectively handle irregular objects, we propose a trajectory optimization framework that comprises two phases. Firstly, in a specified time limit of 10 s, initial offline trajectories are computed for a seamless motion from an initial configuration of the robot to grasp the object and deliver it to a pre-defined target location. Secondly, fast online trajectory optimization is implemented to update robot trajectories in real-time within 100 ms. This helps to mitigate pose estimation errors from the vision system. To account for model inaccuracies, disturbances, and other non-modeled effects, trajectory tracking controllers for both the robot and the gripper are implemented to execute the optimal trajectories from the proposed framework. The intensive experimental results effectively demonstrate the performance of our trajectory planning framework in both simulation and real-world scenarios.

I. INTRODUCTION

Robots have become popular in the manufacturing industry because of the demand for increased automation and advances in computer processors. Besides, modular production setup allowing components to be changed has been trending in several industry sectors. Since robots are easily interchangeable for different purposes, they play a central role in rapidly adapting to new products and production processes. This study focuses on a proof-of-concept for a robot-assisted application. Specifically, grasping *irregular objects* with complex geometries in producing engine casing parts (Fig. 1). While typical robotic applications involve grasping objects from fixed positions [1], [2], this work investigates scenarios where the object’s placement is random, which increases the complexity but improves the production line’s flexibility. The research focuses on achieving *flexibility* and *dynamic grasping* capabilities where the robot remains in motion during the grasping process, which reduces application time and increases production efficiency.

A. Problem description

To facilitate the seamless grasping task, our equipment comprises a KUKA LBR iiwa R820 robot, SDH2 gripper, BASLER AVA 1000-100GC camera, and a grasping object, see Fig. 1. The camera, mounted on the robot end-effector,

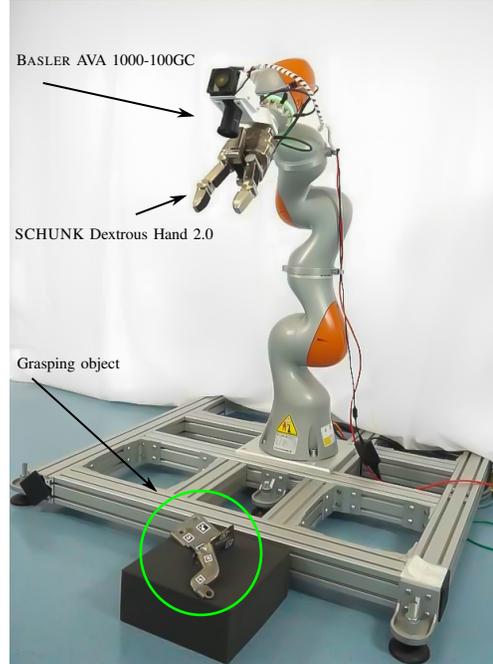


Fig. 1: An example setup of grasping irregular objects.

follows an eye-in-hand configuration. Without loss of generality, we consider the grasping pose to be located at O_g , which can be computed via a computer vision module at 30 Hz (e.g., Aruco marker and an extended Kalman filter). Note that since the object has an irregular shape and is not symmetrical, a small error caused by either the vision system or the trajectory planner can lead to unbalanced forces on the object’s surface, causing internal dislocation of the object between two gripper fingers. Therefore, the *replanning capability* of the proposed framework is critical.

B. Literature review

1) *Optimization-based trajectory optimization*: Motion planning can be addressed through numerical optimization to determine a locally optimal trajectory, considering all dynamic constraints of the system [3], [4], [5]. Two notably successful algorithms for this purpose are CHOMP [6] and TrajOpt [7]. In these algorithms, the trajectory is parameterized by its path progress or time. Then, the gradient-based methods are employed to identify the locally optimal trajectory. While these methods have proven successful in numerous applications [6], their computation time remains too long for real-time implementation on a standard electronic control unit, as observed in studies such as [8], [9]. In authors’ previous works [10], [11], [12], a distinction

¹ Minh Nhat Vu, is with the Automation & Control Institute (ACIN), TU Wien, 1040 Vienna, Austria vu@acin.tuwien.ac.at

² Florian Grander is with Automation and Control Center, Egger GmbH, florian.grander@egger.com

³ Anh Nguyen is with the Department of Computer Science, University of Liverpool, anh.nguyen@liverpool.ac.uk

is drawn between offline and online trajectory planning. Offline trajectory planning is conducted before the robot's movement to compute an optimal initial guess, whereas online trajectory planning occurs during execution. Recently, stochastic trajectory optimization (STO) methods, e.g., Via Point-STO [13], Chance-Constrained Via Point-STO [14], have been utilized to optimize over a continuous trajectory space defined by via points. Constraints such as system limits are specified implicitly and handled by the trajectory representation. Although these mentioned approaches have been successfully implemented in different robotic systems, the online capability is still challenging for this grasping task.

2) *Model predictive control-based trajectory optimization:* In recent years, there has been growing interest in extending trajectory optimization to online planning using receding horizons, encompassing gradient-based methods [15] and sampling-based approaches [16]. However, for manipulation tasks, simple point-to-point planning often falls short, necessitating the inclusion of additional constraints, such as pre-grasp points. Model Predictive Path Integral control [17], [18], [19], also called sampling-based MPC, has proven real-time performance on real robotic systems in challenging and dynamic environments. However, these methods are typically limited to short-horizon problems [13]. Toussaint et al. [20] introduced a sequence-of-constraints Model Predictive Control (MPC) approach to address task and motion planning (TAMP) in three stages. First, task planning generates waypoints; next, these waypoints are optimized in terms of timing to create a reference trajectory. Finally, MPC uses this reference to calculate collision-free paths over a short planning horizon. A global reference is essential for integrating waypoints into the MPC.

Inspired by the two phases mentioned above of trajectory planning and the MPC-based trajectory optimization, we first compute the offline trajectories for three phases, e.g., moving to the object, grasping the object, and moving to the target. Then, in an MPC fashion, a novel online trajectory planner is implemented to account for the new object pose update from the vision system.

C. Paper's contributions

Our contributions are threefold.

- *Firstly*, our method synchronizes offline trajectories for the robot and the gripper, considering the robot's dynamics, within a short computing time of 10 s, allowing dynamic grasping without disrupting the robot's motion.
- *Secondly*, our new online trajectory replanning method reactively adjusts the robot's and the gripper's trajectories within 100 ms from the real-time updates of the object's pose. This responsiveness is crucial for precisely grasping objects with complex geometry, minimizing the impact of minor errors in position detection.
- *Thirdly*, experiments are demonstrated to show the generalization of the proposed framework. The proposed framework can be generalized to different grippers and grasping a moving object. Videos of experiments are found at acin.tuwien.ac.at/39bb

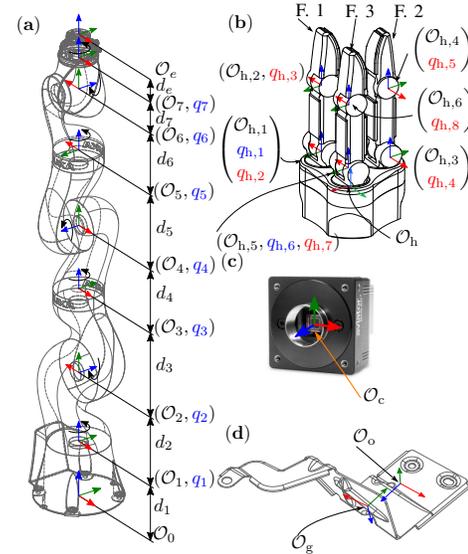


Fig. 2: Schematic drawing of (a) the KUKA LBR iiwa R820, (b) the SDH 2 hand, (c) the camera BASLER AVA 1000-100GC, and (d) the object. The x -, y -, and z -axis of each coordinate frame are depicted by red, green, and blue arrows, respectively. In the tuple of the rotation angle and its corresponding frame $(\mathcal{O}_{h,i}, q_{h,i})$, the illustrating color of the joint angle $q_{h,i}$ matches the corresponding rotate axis.

II. MODELING

This section briefly presents the modeling of the robot and the SCHUNK Dextrous Hand 2.0 (SDH2) gripper.

A. Modeling of the KUKA LBR iiwa R820 robot

The robot is modeled as a rigid-body system with the generalized coordinates $\mathbf{q}^T = [q_1, q_2, \dots, q_7]$, see Fig. 2(a), which are the rotation angles q_i around the z -axes (blue arrows) of each coordinate frame \mathcal{O}_i , $i = 1, \dots, 7$. Considering system state $\mathbf{x}^T = [\mathbf{q}^T, \dot{\mathbf{q}}^T]$ and \mathbf{v} is the vector of control inputs, the state-space form system dynamics is expressed as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}) = [(\dot{\mathbf{q}})^T, \mathbf{v}^T]^T \quad (1)$$

B. Modeling of SCHUNK Dextrous Hand 2.0 gripper

The system state of the SDH2 consists of four DoF $\mathbf{q}_h^T = [q_{h,2}, q_{h,3}, q_{h,7}, q_{h,8}]$, see Fig. 3(b). The grasping (contact) point G is chosen as the midpoint of the line between the points C_1 and C_3 . These points are the centers of arcs formed by the tactile sensor surfaces of fingers 1 and 3, respectively. Note that the choice of this grasping point is crucial to achieve the steady state movement of the object after grasping. This is because the contact points at the object's surface will be perpendicular to the arcs of tactile sensor surfaces. Thereby, the contact forces are equally distributed to the object's surface. The forward kinematics formulation

$$\mathbf{p}_h^G = [x_G, y_G, z_G]^T = \text{SHD2_FK}(\mathbf{q}_h) \quad (2)$$

describes the position of the grasping point G depending on the joint coordinates \mathbf{q}_h .

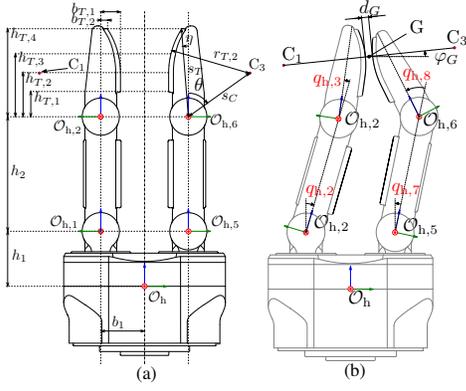


Fig. 3: (a) Side view of the SDH2 without the finger 2 with $q_{h,1} = -q_{h,6} = -\pi/2$. (b) Side view of the gripping hand with the SDH2 state variables \mathbf{q}_h and the grasping state \mathbf{q}_G .

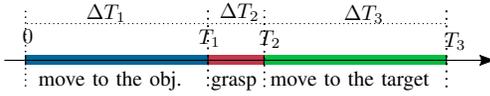


Fig. 4: Timeline

III. TRAJECTORY OPTIMIZATION

This section presents the overall optimization framework, which consists of two stages. In the offline stage, we consider three trajectory optimization phases; see Fig 4. In the first blue region of the timeline, the trajectory guides the robot from the initial configuration to the robot's target pose before the grasping process during the time $0 \leq t \leq T_1$. Subsequently, the second trajectory phase ΔT_2 , discussed in Section III-A.2, helps to move the robot from the robot's target pose before the grasping process to the target robot's pose after the grasping process during the time $T_1 < t \leq T_2$. Finally, Section III-A.3 describes the trajectory computation that brings the object to a user-defined target position during the time $T_2 < t \leq T_3$.

A. The offline trajectory optimization of the robot

In the i -th trajectory phase, $i \in 1, \dots, 3$, the trajectory that moves the robot from an initial configuration $\mathbf{x}_{i,S}$ to the target configuration \mathbf{x}_{i,T_i} is formulated as an optimization by discretizing the trajectory $\boldsymbol{\xi}_i(t)$, $t \in [0, \Delta T_i]$, with $N_i + 1$ grid points and solving the resulting static optimization problem

$$\min_{\boldsymbol{\xi}_i} J(\boldsymbol{\xi}_i) = \Delta T_i + \Delta t_i \sum_{k=0}^{N_i} \mathbf{v}_{1,k}^T \mathbf{v}_{1,k} \quad (3a)$$

$$\text{s.t. } \mathbf{x}_{i,k+1} - \mathbf{x}_{i,k} = \frac{1}{2} \Delta t_h \begin{bmatrix} \dot{\mathbf{q}}_{i,k+1} + \dot{\mathbf{q}}_{i,k} \\ \mathbf{v}_{i,k+1} + \mathbf{v}_{i,k} \end{bmatrix} \quad (3b)$$

$$\mathbf{x}_{i,0} = \mathbf{x}_{i,S}, \quad \mathbf{x}_{i,N} = \mathbf{x}_{i,T_i} \quad (3c)$$

$$\underline{\mathbf{x}} \leq \mathbf{x}_{i,k} \leq \bar{\mathbf{x}}, \quad (3d)$$

$$\underline{\boldsymbol{\tau}} - \mathbf{c} \leq (\mathbf{M}(\mathbf{q}_{i,k})\mathbf{v}_{i,k} + \mathbf{g}(\mathbf{q}_{i,k})) \leq \bar{\boldsymbol{\tau}} - \bar{\mathbf{c}} \quad (3e)$$

$$k = 0, \dots, N_i$$

for the optimal trajectory

$$(\boldsymbol{\xi}_i^*)^T = [\Delta T_i^*, (\mathbf{x}_{i,0}^*)^T, \dots, (\mathbf{x}_{i,N_i}^*)^T, (\mathbf{v}_{i,0}^*)^T, \dots, (\mathbf{v}_{i,N_i}^*)^T], \quad (4)$$

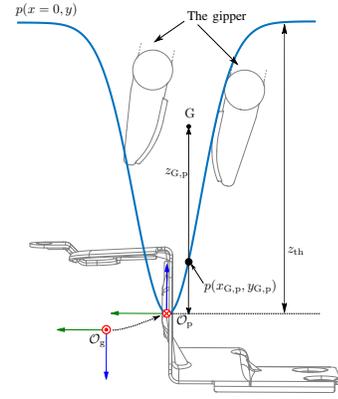


Fig. 5: Side view of the grasping object and fingertips with the potential function $p(0, y)$.

with the time step $\Delta t_i = \Delta T_i / N_i$. Note that the final time ΔT_i^* in (4) denotes the optimal duration of the trajectory from the initial state $\mathbf{x}_{i,S}$ to the target configuration \mathbf{q}_{i,T_i} . The system dynamics (1) is approximated by the trapezoidal rule in (3b). Additionally, $\underline{\mathbf{x}}$ and $\bar{\mathbf{x}}$ in (3d) denote the symmetric lower and upper bounds of the state, respectively, and (3e) considers the upper and lower torque limit $\bar{\boldsymbol{\tau}}$ and $\underline{\boldsymbol{\tau}}$. Note that the torque limits in (3e) are an expensive inequality constraint. Instead of fully neglecting the Coriolis matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, which is often done in industrial applications [21], upper and lower bounds, $\bar{\mathbf{c}}$ and $\underline{\mathbf{c}}$, for the values of $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ are determined according to [10]. Although the influence of the Coriolis matrix on the overall system's dynamics is insignificant, it is still advantageous to consider these physical limits in the optimization problem (3). Note that to create smooth transitions, the final state of a trajectory is chosen as the initial state of the subsequent trajectory. The following presents additional constraints to (3) for each phase.

1) Phase 1: Trajectory of approaching the object from an initial configuration

Since the object to be grasped has an unconventional shape, see Fig. 2(d), the potential function is employed that restricts the motion of the gripper while approaching the object. In Fig. 5, a simple description of the potential function is illustrated where a side view of a potential function is drawn in blue. Therein, the motion of the grasping point G of the gripper is restricted in the inverted Gaussian bell shape (illustrated in blue color) when the height of the point G is smaller than a threshold value z_{th} . In the following, the potential function is described in detail. The potential function is formulated as an inverted Gaussian function in the form

$$p(x, y) = z_{th} \left(1 - \exp \left(-\frac{1}{2} \begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix} \right) \right), \quad (5)$$

where z_{th} is the threshold value, $\boldsymbol{\Sigma}$ is the covariance matrix, and $\boldsymbol{\mu}^T = [\mu_x, \mu_y]$ is mean value of the Gaussian function. In Fig. 5, a side view of the potential function $p(x=0, y)$ in the frame \mathcal{O}_p is depicted with the variances are chosen to be $\sigma_x^2 = 5 \text{ mm}$ and $\sigma_y^2 = 2 \text{ mm}$. The mean value is $\boldsymbol{\mu}^T = [0, 0]$.

Thereby, to guarantee that the grasping point is inside the region of the Gaussian bell shape created by the potential function, the following condition

$$h(\mathbf{q}) = z_{G,p} - p(x_{G,p}, y_{G,p}) \geq 0 \quad (6)$$

must be added in (3) for the phase $i = 1$.

2) *Phase 2: Trajectory of the robot during grasping action of the gripper*

In addition to (3) for $i = 2$, to ensure the object is lifted from the ground at the end of this second trajectory phase, the z -axis velocity \mathbf{v}_{0,T_2}^h of the gripper w.r.t. the world frame \mathcal{O}_0 is larger than 0, corresponding to

$$\mathbf{e}_z^T \mathbf{v}_{0,T_2}^h \geq 0, \quad \text{with} \quad \mathbf{v}_{0,T_2}^h = \left. \frac{\partial \mathbf{p}_0^h}{\partial \mathbf{q}} \right|_{\mathbf{q}=\mathbf{q}_{2,T_2}} \dot{\mathbf{q}}_{2,T_2}. \quad (7)$$

3) *Phase 3: Trajectory of the movement from the grasping object to the target position*

In this phase (green region in Fig. 4), the optimal trajectory ξ_3 obtained from (3) drives the robot from the final state \mathbf{x}_{2,T_2} of phase 2 to a predefined target state \mathbf{x}_{3,T_3} . Note that the target state is a stationary point, i.e., $\mathbf{x}_{3,T_3} = [\mathbf{q}_{3,T_3}, \mathbf{0}_3]$.

B. Online trajectory replanner

Due to the complexity of optimization problems, the optimization (3) requires longer computation times (≥ 3 s) with dense grid points $N_i = 100$, $i = \{1, 2, 3\}$. Therefore, the following procedure presents the online trajectory replanner to adapt the robot trajectories ξ_i^* , $i = \{1, 2, 3\}$. To precisely grasp the object, the pose of the object \mathbf{H}_0^o is updated using the computer vision module at the rate 10 Hz. Thereby, the new starting $\hat{\mathbf{x}}_{S_i}$ and target states $\hat{\mathbf{x}}_{T_i}$, $i = \{1, 2, 3\}$ are computed accordingly. Here, the online trajectory replanner is used to minimize the deviation from the current optimal trajectory while satisfying the dynamic constraints of the system and adapting these new target states. Since the following procedures are identical for three trajectory phases ξ_i , $i \in \{1, 2, 3\}$, the subscript i is omitted for compact notations. The current optimal trajectory is expressed as

$$(\xi^*)^T = [\Delta T^*, (\mathbf{x}_0^*)^T, \dots, (\mathbf{x}_N^*)^T, (\mathbf{v}_0^*)^T, \dots, (\mathbf{v}_N^*)^T]. \quad (8)$$

Note that this optimal trajectory satisfies the dynamical constraints approximated by the trapezoidal rule in the form

$$\mathbf{x}_{k+1}^* = \mathbf{x}_k^* + \frac{\Delta T}{2N} (\mathbf{f}_k^* + \mathbf{f}_{k+1}^*). \quad (9)$$

For the online trajectory replanner, small deviations

$$\delta \xi^T = [\delta T, (\delta \mathbf{x}_0)^T, \dots, (\delta \mathbf{x}_N)^T, (\delta \mathbf{v}_0)^T, \dots, (\delta \mathbf{v}_N)^T]$$

need to be taken into account to compute the new trajectory connecting the current starting state $\hat{\mathbf{x}}_S$ with the updated target state $\hat{\mathbf{x}}_T$. The discrete-time system dynamics (1) w.r.t. the interpolated optimal state ξ^* reads as

$$\begin{aligned} \mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\Delta T^* + \delta T}{2N} & \left(\mathbf{f}_k^* + \mathbf{\Gamma}_k^x \delta \mathbf{x}_k + \mathbf{\Gamma}_k^v \delta \mathbf{v}_k \right. \\ & \left. + \mathbf{f}_{k+1}^* + \mathbf{\Gamma}_{k+1}^x \delta \mathbf{x}_{k+1} + \mathbf{\Gamma}_{k+1}^v \delta \mathbf{v}_{k+1} \right), \end{aligned} \quad (10)$$

with $\delta \mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_k^*$, $\delta \mathbf{v}_k = \mathbf{v}_k - \mathbf{v}_k^*$, $\delta t = \Delta T - \Delta T^*$, and

$$\mathbf{\Gamma}_k^x = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}_k^*, \mathbf{v}_k^*}, \quad \mathbf{\Gamma}_k^v = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \right|_{\mathbf{x}_k^*, \mathbf{v}_k^*}$$

for $k = 0, \dots, N-1$. Subtracting (9) from (10) and neglecting terms containing a product of deviation variables, we have

$$\begin{aligned} \delta \mathbf{x}_{k+1} = \delta \mathbf{x}_k + \frac{\Delta T^*}{2N} & \left(\mathbf{\Gamma}_k^x \delta \mathbf{x}_k + \mathbf{\Gamma}_k^v \delta \mathbf{v}_k + \mathbf{\Gamma}_{k+1}^x \delta \mathbf{x}_{k+1} \right. \\ & \left. + \mathbf{\Gamma}_{k+1}^v \delta \mathbf{v}_{k+1} \right) + \frac{\delta T}{\Delta T^*} (\mathbf{x}_{k+1}^* - \mathbf{x}_k^*). \end{aligned} \quad (11)$$

In a more compact form, (11) is rewritten as

$$\mathbf{C}_{k+1} \mathbf{s}_{k+1} = \mathbf{A}_k \mathbf{s}_k, \quad (12)$$

where

$$\begin{aligned} \mathbf{C}_{k+1} &= \begin{bmatrix} \mathbf{I} - \frac{h^*}{2} \mathbf{\Gamma}_{k+1}^x & -\frac{h^*}{2} \mathbf{\Gamma}_{k+1}^v & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \\ \mathbf{A}_k &= \begin{bmatrix} \mathbf{I} + \frac{h^*}{2} \mathbf{\Gamma}_k^x & \frac{h^*}{2} \mathbf{\Gamma}_k^v & \mathbf{x}_{k+1}^* - \mathbf{x}_k^* \\ \mathbf{0} & \mathbf{0} & \frac{\Delta T^*}{1} \end{bmatrix}, \\ \mathbf{s}_k &= \begin{bmatrix} \delta \mathbf{x}_k \\ \delta \mathbf{v}_k \\ \delta T_k \end{bmatrix}, \end{aligned}$$

and $h^* = \frac{\Delta T^*}{N}$. For simplicity, only one variable for the final time δT in (10) was introduced instead of δT_k , $k = 0, \dots, N-1$. Thus, $\delta T_{k+1} = \delta T_k$ was used in (12). The deviation vector ξ_k is obtained as the solution of a linear constrained quadratic program (LCQP) of the form

$$\min_{\mathbf{s}_k} \frac{1}{2} \sum_{k=1}^{N-1} \mathbf{s}_k^T \mathbf{Q}_k \mathbf{s}_k \quad (13a)$$

$$\text{s.t. } \mathbf{C}_{k+1} \mathbf{s}_{k+1} = \mathbf{A}_k \mathbf{s}_k, \quad k = 0, \dots, N-1 \quad (13b)$$

$$\underline{\mathbf{s}}_k \leq \mathbf{s}_k \leq \overline{\mathbf{s}}_k, \quad k = 0, \dots, N \quad (13c)$$

and the positive definite weighting matrix

$$\mathbf{Q}_k = \text{diag}(\mathbf{Q}_{\mathbf{x}_k}, \mathbf{Q}_{\mathbf{v}_k}, Q_{\Delta T}). \quad (14)$$

With the choice of $Q_{t_F} > 0$ and the submatrices $\mathbf{Q}_{\mathbf{x}_k}$ and $\mathbf{Q}_{\mathbf{v}_k}$, the deviation of the online trajectory from the trajectory (8) can be weighted explicitly in the objective function (13a) w.r.t. the traversal time ΔT , the state \mathbf{x}_k , and the control input \mathbf{v}_k , respectively. The inequality condition (13c) allows the admissible tolerances of the online trajectory, where

$$\begin{aligned} \underline{\mathbf{s}}_k^T &= [\underline{\mathbf{x}}_k^T - (\mathbf{x}_k^*)^T, \underline{\mathbf{v}}_k^T - (\mathbf{v}_k^*)^T, \underline{\delta T}], \\ \overline{\mathbf{s}}_k^T &= [\overline{\mathbf{x}}_k^T - (\mathbf{x}_k^*)^T, \overline{\mathbf{v}}_k^T - (\mathbf{v}_k^*)^T, \overline{\delta T}], \end{aligned}$$

for $k = 1, \dots, N-1$, and $\overline{\delta T}$ and $\underline{\delta T}$ is a sufficiently large upper and lower bound for δt_F , respectively. Additionally, the equality constraints must be taken into account by

$$\begin{aligned} \underline{\mathbf{s}}_0^T &= [\delta \mathbf{x}_0^T, \delta \mathbf{v}_0^T, \underline{\delta T}], \quad \overline{\mathbf{s}}_0^T = [\delta \mathbf{x}_0^T, \delta \mathbf{v}_0^T, \overline{\delta T}], \\ \underline{\mathbf{s}}_N^T &= [\delta \mathbf{x}_N^T, \delta \mathbf{v}_N^T, \underline{\delta T}], \quad \overline{\mathbf{s}}_N^T = [\delta \mathbf{x}_N^T, \delta \mathbf{v}_N^T, \overline{\delta T}]. \end{aligned}$$

Finally, for each trajectory phase, the optimal trajectory of the online trajectory replanner ξ^* reads as

$$\xi^* \leftarrow \xi^* + \delta\xi^*, \quad (15)$$

where $\delta\xi^*$ results from the solution of (13) in the form

$$(\delta\xi^*)^T = [\delta T^*, (\delta\mathbf{x}_0^*)^T, \dots, (\delta\mathbf{x}_N^*)^T, (\delta\mathbf{v}_0^*)^T, \dots, (\delta\mathbf{v}_N^*)^T].$$

IV. RESULTS

The experimental setup is illustrated in Fig. 1, driven by the 4.0 GHz Intel Core i7-10700K PC with 32 GB RAM. Three network interface controllers (NIC) are used for communication with the robot, the SDH2 gripper, and the camera via EtherCAT, RT-Ethernet, and Ethernet, respectively. Since the joint velocities of the complete system cannot be measured directly, differential filters with a time constant of $T_1 = 12$ ms are used for each joint. The sampling time for the controller is $T_s = 125$ μ s. The offline trajectory optimization (3) and the online replanner (13) are solved using the *Interior Point Optimizer* (IPOPT) [22] with the linear solver MA57 [23], [24]. In these optimization problems, the numbers of grid points N_i , $i \in \{1, 2, 3\}$ are set to 100, and the average computing times are listed in Table I.

A. Simulation results

Statistical results. We report the average computing times are taken on 988 successful trials over the total 1000 trials of random uniformly distributed object poses $(\hat{\mathbf{p}}_0)^T = [p_{0,x}^o, p_{0,y}^o, p_{0,z}^o] \in [\underline{\mathbf{p}}_0^o, \overline{\mathbf{p}}_0^o]$ with $(\underline{\mathbf{p}}_0^o)^T = [0.6 \text{ m}, -0.1 \text{ m}, 0.1 \text{ m}]$ and $(\overline{\mathbf{p}}_0^o)^T = [0.7 \text{ m}, 0.1 \text{ m}, 0.1 \text{ m}]$. In 12 failed trials, the optimization problems reach the iteration limits set to 100. In simulations, the proposed framework shows excellent results with a success rate of approximately 98.8%. In Tab. I, the computation times of the offline phases do not exceed 10 s.

TABLE I: Average computation time comparison.

Phase	Ours	VP-STO [13]
phase 1 opt. traj. (3)	2.7 ± 0.4 (s)	7.2 ± 1.6 (s)
phase 2 opt. traj. (3)	1.2 ± 0.1 (s)	9.8 ± 3.7 (s)
phase 3 opt. traj. (3)	4.3 ± 0.9 (s)	11.5 ± 2.3 (s)
online opt. traj. (13)	53.5 ± 7.9 (ms)	NA

Comparison. In addition, we utilize the via-point stochastic trajectory optimization (VP-STO) [13] to solve the offline trajectory planning phases. Although the success rate of VP-STO is 100%, our computational speed for offline trajectory planning outperforms VP-STO significantly. Note that the average computation time of the online replanner is below 100 ms, significantly faster than the computing time of the offline trajectory planning.

B. Real robot results

Static scenarios. The initial and the target configuration of the robot are chosen as

$$\mathbf{q}_T^T = [55^\circ, 39.5^\circ, -6.6^\circ, -22^\circ, -36.3^\circ, 110.6^\circ, 134^\circ]. \quad (16)$$

and

$$\mathbf{q}_T^T = [0^\circ, 0^\circ, 0^\circ, -40^\circ, 0^\circ, 100^\circ, 90^\circ]. \quad (17)$$

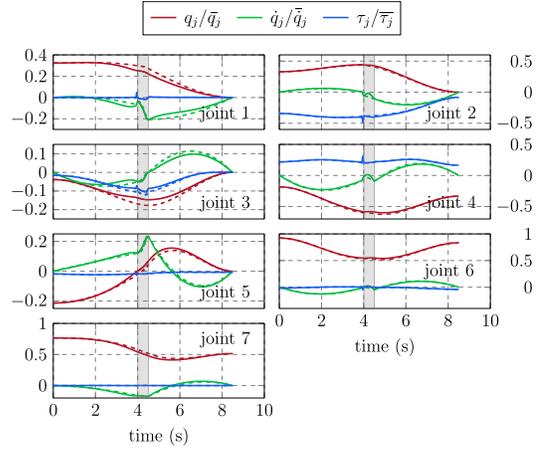


Fig. 6: Time evolution of the scaled offline and the online trajectory optimization with the initial robot configuration (16). The dashed lines are the scaled offline trajectories ξ_i , $i \in \{1, 2, 3\}$ obtained from (3). The solid lines represent the scaled online trajectories obtained from (15). The grey areas indicate the second trajectory phase when the robot grasps the object. The subscript j indicates the j^{th} joint of the robot.

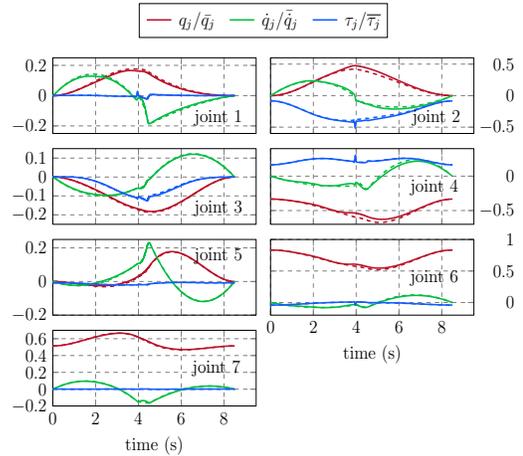


Fig. 7: Time evolution of the experiment on the comparison of the proposed algorithm with and without the online replanning in (13). The dashed lines illustrate the computed trajectory without the online replanner. The solid lines depict the computed trajectory from (15).

The object is placed in the camera's field of view on the ground. With a given object pose from the computer vision system, the three-phase trajectory of the robot, illustrated by dashed lines in Fig. 6, is computed offline using (3). During the robot's motion, the online trajectories, depicted by solid lines, are updated periodically. In Fig. 6, the grey areas indicate the second phase, i.e., the grasping process. It is worth noting that three parts of the robot's trajectory are smoothly connected. Since the trajectories in Fig. 6 do not surpass the ± 1 horizontal line, hence, all the state and input constraints according to (3d) and (13c) are respected.

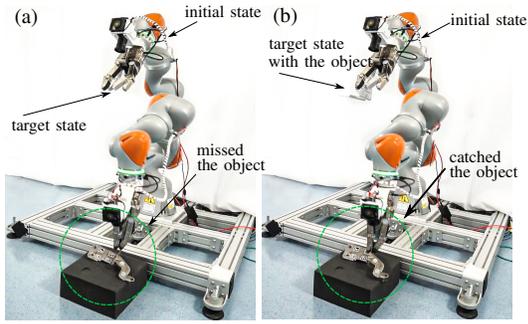


Fig. 8: Snapshots of experiment on the comparison of the proposed trajectory optimization framework with (a) and without (b) the online replanning.

TABLE II: Results of grasping moving objects over 20 trials.

	Ours	RRTConnect [25]
success rate (%)	95	75
comp. violation (num. of trials)	0	2
missed object (num. of trials)	1	3

Comparison between with and without the online replanner.

To further validate the effectiveness of the online replanner, a comparison with only the offline solution is conducted. To this end, two trials in which the initial and the target configuration of the robot are both chosen as

$$\mathbf{q}_S^T = \mathbf{q}_T^T = [0^\circ, 0^\circ, 0^\circ, -40^\circ, 0^\circ, 100^\circ, 90^\circ], \quad (18)$$

and the object is randomly positioned in the camera’s field of view, as shown in Fig. 8. Only the offline trajectories computed from (3) are sent to the robot controller in the first trial. On the other hand, in the second trial, the online replanner (13) periodically updates the trajectory according to the object pose’s update. The scaled trajectories of the first and second trials are illustrated by the dashed lines and the solid lines in Fig. 7. Therein, during $0 \leq t \leq 3$, the robot mainly follows the offline optimal trajectories. Once the robot is close to the object and a more precise object pose is acquired, minor deviations occur at $3 < t \leq 4$. Although the deviations are minor, the online trajectory replanner helps the robot to successfully grasp the object while only executing the offline trajectory failed to grasp the object, illustrated in Fig. 8 (a) and (b), respectively.

Generalization to dynamic scenarios. The proposed framework is generalized to a *different gripper* (Robotiq 2F85) and can grasp a dynamically moving object. In Fig. 9, overlay images of an experiment are illustrated. When the robot nearly reaches the object, the object is suddenly relocated (see sub-figures 2,3 in Figure 9). The online replanner can still adjust the offline trajectories and precisely grasp the object (see sub-figure 6 in Fig. 9). We further run a small-scale Monte Carlo simulation of 20 trials to compare the performance of our proposed framework with RRTConnect [26] implemented in MoveIt [25]. Statistical results are presented in Tab. II. Note that the Time-Optimal Trajectory Generation (TOTG) algorithm [27] is applied to obtain the time parametrization for the RRTConnect algorithm. We chose RRTConnect over

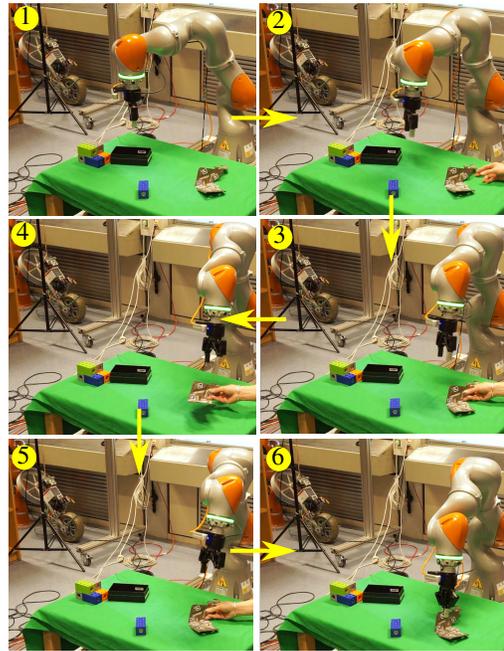


Fig. 9: A demonstration of grasping the moving object.

another variant of the RRT algorithm due to the computation speed, which is fast enough for this moving object scenario. The update frequency for the trajectory replanner is 100 ms. With our proposed approach, there is a 1 failed trial when the robot misses grasping the object, i.e., 95% success rate. On the other hand, the RRTConnect planner violated the time limit in 2 trials and failed to grasp the moving object in another three trials. Video of the experiments is provided at acin.tuwien.ac.at/39bb

V. DISCUSSION

This paper presents the two-stage trajectory optimization framework for grasping irregular objects with complex geometry using a novel online trajectory replanning. To account for the object’s pose estimation errors from a computer vision module, a novel online update of the robot trajectory is implemented to react to the change of the object’s pose during the grasping motion. This application has been verified and tested in several experiments and outperforms the current state-of-the-art trajectory planning method in computational speed. The proposed framework is also generalized with different grippers and in moving object scenarios. With this extreme scenario, the proposed framework can successfully grasp the object in 19 over 20 trials. The main limitation of the proposed method is that it relies on offline trajectories requiring 10s for computation. Therefore, we are currently extending this two-stage approach to the online version, where we consider the shorter planning horizon for the offline phase and the incremental update when approaching the goal. Another direction is to compute and learn motion primitives for the offline planning phase via generative AI techniques, e.g., diffusion models, and variant autoencoder. As a result, this method could be used in an interactive scenario like language-driven human-robot collaboration tasks.

REFERENCES

- [1] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Preparatory object reorientation for task-oriented grasping," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 893–899.
- [2] A. D. Vuong, M. N. Vu, B. Huang, N. Nguyen, H. Le, T. Vo, and A. Nguyen, "Language-driven grasp detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 902–17 912.
- [3] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.
- [4] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.
- [5] M. N. Vu, F. Beck, M. Schwegel, C. Hartl-Nesic, A. Nguyen, and A. Kugi, "Machine learning-based framework for optimally solving the analytical inverse kinematics for redundant manipulators," *Mechatronics*, vol. 91, p. 102970, 2023.
- [6] 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*, vol. 32, no. 9-10, pp. 1164–1193, 2013.
- [7] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [8] S. Iftikhar, O. J. Faqir, and E. C. Kemgan, "Nonlinear model predictive control of an overhead laboratory-scale gantry crane with obstacle avoidance," *Proceedings of the Conference on Control Technology and Applications (CCTA)*, pp. 382–387, 2019.
- [9] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.
- [10] M. N. Vu, C. Hartl-Nesic, and A. Kugi, "Fast swing-up trajectory optimization for a spherical pendulum on a 7-dof collaborative robot," in *Proceedings of the International Conference on Robotics and Automation*, 2021, pp. 10 114–10 120.
- [11] M. Vu, P. Zips, A. Lobe, F. Beck, W. Kemmetmüller, and A. Kugi, "Fast motion planning for a laboratory 3d gantry crane in the presence of obstacles," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9508–9514, 2020.
- [12] M. N. Vu, A. Lobe, F. Beck, T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Fast trajectory planning and control of a lab-scale 3d gantry crane for a moving target in an environment with obstacles," *Control Engineering Practice*, vol. 126, p. 105255, 2022.
- [13] J. Jankowski, L. Bruder Müller, N. Hawes, and S. Calinon, "Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 125–10 131.
- [14] L. Bruder Müller, G. Berger, J. Jankowski, R. Bhattacharyya, and N. Hawes, "Cc-vpsto: Chance-constrained via-point-based stochastic trajectory optimisation for safe and efficient online robot motion planning," *arXiv preprint arXiv:2402.01370*, 2024.
- [15] T. Schoels, P. Rutquist, L. Palmieri, A. Zanelli, K. O. Arras, and M. Diehl, "CIAO*: MPC-based safe motion planning in predictable dynamic environments," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6555–6562, 2020.
- [16] 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," in *Proceedings of the Conference on Robot Learning (CoRL)*, vol. 164, 2022, pp. 750–759.
- [17] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1423–1430, 2021.
- [18] K. Honda, N. Akai, K. Suzuki, M. Aoki, H. Hosogaya, H. Okuda, and T. Suzuki, "Stein variational guided model predictive path integral control: Proposal and experiments with fast maneuvering vehicles," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 7020–7026.
- [19] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.
- [20] M. Toussaint, J. Harris, J.-S. Ha, D. Driess, and W. Hönig, "Sequence-of-constraints MPC: Reactive timing-optimal control of sequential manipulation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 13 753–13 760.
- [21] E. E. Binder and J. H. Herzog, "Distributed computer architecture and fast parallel algorithms in real-time robot control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 4, pp. 543–549, 1986.
- [22] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
- [23] HSL. (2018) A collection of fortran codes for large scale scientific computation. [Online]. Available: <http://www.hsl.rl.ac.uk/>
- [24] I. S. Duff, "Ma57—a code for the solution of sparse symmetric definite and indefinite systems," *ACM Trans. Math. Softw.*, vol. 30, no. 2, pp. 118–144, June 2004. [Online]. Available: <https://doi.org/10.1145/992200.992202>
- [25] D. Coleman, I. A. Sukan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a MoveIt! case study," *Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 3–16, 2014.
- [26] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.
- [27] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," in *Proceedings of Robotics: Science and Systems*, 2012, pp. 1–8.