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

Differentiable Optimization-Based Modular Planning Framework for Pick-and-Place with Regrasp

Yejun Song, Seoki An, Somang Lee, Jeongmin Lee, Jeongseob Lee, Geunsu Yoo and Dongjun Lee†

***Abstract*— Robotic manipulation commonly involves pick-and-place tasks in which regrasp may be necessary for low-dexterity manipulators. Many existing approaches rely on sampling, which becomes inefficient when repeated regrasp is required in high-dimensional configuration spaces. We propose a modular planning framework that comprises differentiable optimization-based modules: grasp generation, stable pose prediction, inverse kinematics solving, and path planning. The modular design yields a systematic pipeline, enabling direct pick-and-place, static or non-static release, and repeated regrasp by solving each module as needed. Each module leverages differentiable geometric features to efficiently solve its corresponding optimization problem. Our framework explicitly accounts for grasp constraints across both task scenes and predicts stable poses for regrasp planning via optimization rather than expensive physics simulations, thereby improving the feasibility and efficiency of planning. We validated the framework in pick-and-place simulations and real-world experiments.**

I. Introduction

Pick-and-place is a fundamental task in robotic manipulation, yet planning a feasible sequence of actions for diverse object shapes and cluttered environments remains challenging. Even if a grasp is feasible in the pick scene, placing the object at the goal pose can cause collisions, making direct placement impossible [1]. Some manipulators with dexterous grippers can reorient the object in hand to achieve the desired pose for placement [2], [3]. However, many robotic systems rely on grippers with limited dexterity, which cannot perform full in-hand reorientation; examples include two-finger grippers [4], [5] and excavator-style grippers [6] used in industrial applications. In these cases, the object must be temporarily released and regrasped in a different pose, making regrasp planning essential for pick-and-place tasks. This challenge also arises on construction sites, where excavators must pick up rocks, release them, regrasp them, and place them to build stacked stone structures [7].

We propose a modular planning framework for pick-and-place tasks with regrasp. The framework follows a pipeline of optimization-based modules (Fig. 2), in which each module is solved sequentially and invoked as needed, enabling strategies such as direct pick-and-place, static or non-static release, and repeated regrasp. First, the grasp generation module (`GraspGen`) jointly considers both task scenes (e.g., pick/place or pick/release) to generate force-closure and collision-free grasps for a specified pick-and-place scenario.

This research was supported by HD Hyundai XiteSolution. The authors are with the Department of Mechanical Engineering, IAMD and IOER, Seoul National University, Seoul, Republic of Korea, 08826. {yj_song, seoki97s, hopelee, ljmlgh, overjs94, ygs1990, djlee}@snu.ac.kr. Corresponding author: Dongjun Lee.

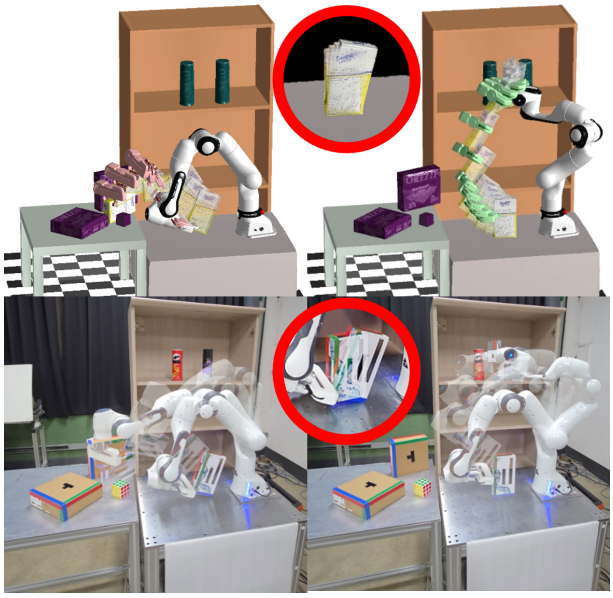


Fig. 1: Visualization of pick-and-place with regrasp, shown in both simulation and real-world experiments. The robot grasps the target object, releases and regrasps it on the table, and then places it on the shelf at the desired pose. Red and green grippers indicate the pick-to-release and regrasp-to-place stages, while magenta- and cyan-tinted objects represent the pick and place scenes, respectively. The figure also shows the object settling into a stable pose after release, prior to being regrasped (red circles).

Then, to define intermediate states for regrasp planning, we devise the stable pose prediction module (`StabPose`), which uses optimization rather than physics simulations for efficiency. We further show that `StabPose` solutions correspond to static equilibria on a flat surface. Leveraging `StabPose`, we extend `GraspGen` to `GraspGen`$^+$, allowing non-static release so that the object settles into the predicted stable pose, avoiding collisions between the gripper and the flat surface during release. Next, the inverse kinematics solving module (`IkSolve`) computes a manipulator configuration, and the path planning module (`PathPlan`) plans collision-free manipulator paths that execute the generated grasps while respecting joint limits. All optimization-based modules are fully differentiable, leveraging the differentiable geometric representations of object shapes [8] and inter-object distances [9]. We validated our framework in simulations and real-world experiments, demonstrating that reliable regrasp planning improves the success of pick-and-place tasks.

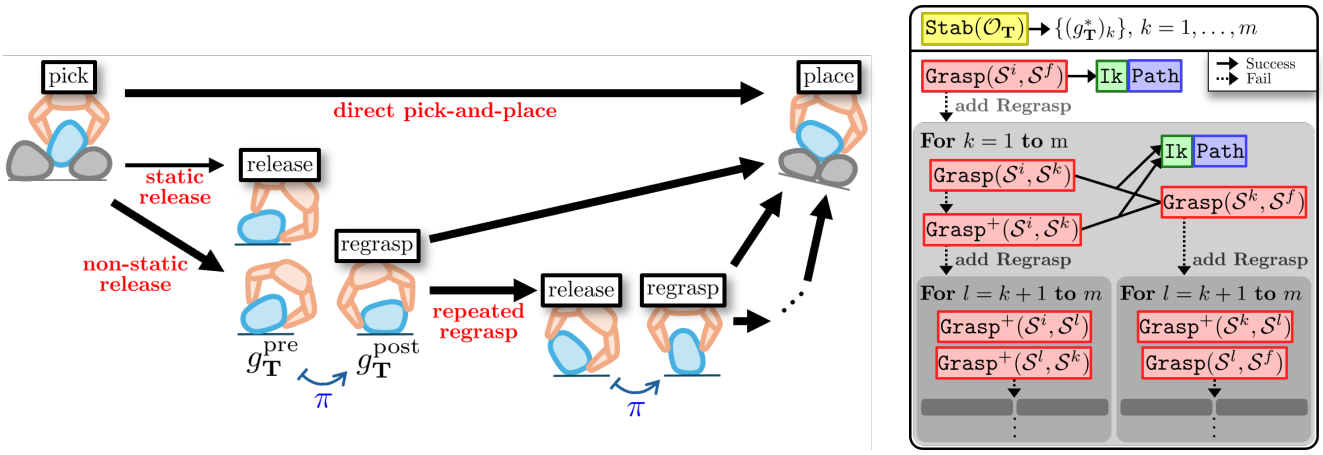


Fig. 2: (a) Illustration of grasps in the pick, place, release, and regrasp scenes, demonstrating pick-and-place strategies planned by the framework. (b) Overall pipeline of the framework, illustrating the modules and their conditional flow, enabling multiple planning strategies. Direct pick-and-place is attempted first; if unsuccessful, regrasp planning leverages predicted stable poses to enable static or non-static release and repeated regrasp. Module names are abbreviated for brevity.

The grasp generation problem has been widely studied using sampling [7], [10], [11], [12], optimization [13], [14], and learning-based methods [15], [16]. Most of these methods consider grasp quality in a single scene without accounting for place scene constraints. Shanthi et al. [1] demonstrate that joint reasoning across pick and place scenes enhances grasp success. In our approach, `GraspGen` generates feasible grasps for both task scenes by forming the union of their obstacles, using differentiable optimization without the need for extensive sampling.

Predicting stable intermediate poses of the target object is a key step in regrasp planning, but many existing methods use computationally expensive physics simulations [7], [11]. Learning-based methods [17], [18] require large datasets, limiting their performance on novel objects. In contrast, `StabPose` predicts stable poses via optimization over the object geometry, satisfying static equilibrium conditions without relying on physics simulations or learning.

For regrasp planning, Wan et al. [10] employ graph search combined with extensive sampling, covering various grasp and regrasp configurations. Wermelinger et al. [7] employ grasp sampling for stone stacking tasks with regrasp, achieving high experimental success rates. However, when regrasp is repeated or the configuration space is high-dimensional, these methods require a large number of samples, rendering them inefficient.

Learning-based reorientation planners, such as ReorientBot [11] and ReorientDiff [12], achieve high success rates but rely on large datasets of successful regrasp plans. Generating such feasible datasets typically requires extensive experiments or sampling-based methods [11], [12], a process that can be accomplished more efficiently with our framework.

To our knowledge, this is the first framework for pick-and-place with regrasp that is both sampling- and simulation-free, achieving efficient planning via fully differentiable optimization. Our contributions are summarized as follows:

- Joint consideration of constraints across both task scenes to generate feasible grasps for a specified pick-and-place scenario.
- Simulation-free stable pose prediction for regrasp, along with non-static release planning informed by it.
- Modular design enabling repeated regrasp and other strategies to improve feasibility, efficiently implemented via differentiable optimization-based modules.

The rest of the paper is organized as follows: Sec. II-A formulates the pick-and-place task with regrasp, Sec. II-B introduces the geometric representations adopted for differentiability, and Sec. III details each module of the planning framework. Sec. IV presents the ablation study and real-world experiments, and Sec. V concludes with future work.

II. Problem Formulation

A. Pick-and-Place with Regrasp

A key challenge in manipulation tasks is to find a grasp $X = (g_G, \theta_G)$, where $g_G \in \mathrm{SE}(3)$ and $\theta_G \in \mathbb{R}^{n_\theta}$ denote the gripper pose and finger joint configuration, respectively. The grasp should be collision-free and satisfy a grasp quality threshold $f_{\text{threshold}}$. Let $f(X; \mathcal{S})$ denote a grasp quality, where $\mathcal{S}$ represents the *scene* consisting of obstacles within the region of interest. A feasible grasp satisfies:

$$\begin{aligned} & f(X; \mathcal{S}) \geq f_{\text{threshold}} \\ & \operatorname{dist}\Big(\mathcal{C}_{\text{finger}}(X), \mathcal{O}_{\mathbf{T}}\Big) = 0 \\ & \operatorname{dist}\Big(\mathcal{C}(X), \mathcal{O}_{\text{obstacle}}\Big) \geq \delta_{\text{col}}, \ \forall \mathcal{O}_{\text{obstacle}} \in \mathcal{S} \\ & \underline{\theta} \leq \theta_G \leq \overline{\theta} \end{aligned} \tag{1}$$

where $\operatorname{dist}(\cdot, \cdot)$ denotes the distance between two bodies, δ_{col} represents the collision margin, and the last constraint enforces the finger joint limits. We use $\mathcal{C}$ and $\mathcal{O}$ to denote the links and the objects, respectively, with $\mathcal{O}_{\mathbf{T}}$ as the target.

To formalize this, we define the feasible grasp set $\mathcal{F}_G$ as:

$$\mathcal{F}_G(\mathcal{S}) := \{X \mid X \text{ satisfies (1) given } \mathcal{S}\}$$

In this paper, we consider pick-and-place tasks using a two-finger gripper without in-hand dexterity, where regrasp may be necessary to achieve the desired placement pose. Here, the goal of pick-and-place planning is to find initial (pick) and final (place) grasps, X^i and X^f, formulated as:

$$\begin{aligned} &\text{find } X^i,\ X^f \\ &\text{s.t. } \mathcal{F}_G(\mathcal{S}^i) \ni X^i = X^f \in \mathcal{F}_G(\mathcal{S}^f) \end{aligned} \tag{2}$$

where $X^i = X^f$ indicates the *equivalent grasp* defined as:

$$\left(g^i_{\mathbf{T}}\right)^{-1} g^i_G = \left(g^f_{\mathbf{T}}\right)^{-1} g^f_G, \quad \theta^i_G = \theta^f_G$$

which implies that the two grasps are identical in the frame of the target object, $g_{\mathbf{T}}$.

If (2) has no solution, the object should be released at a predefined, obstacle-free location for regrasp. Releasing the object in an unstable pose causes it to settle into a stable pose, and regrasp is performed only after the object stabilizes. Let $\pi : \mathrm{SE}(3) \to \mathrm{SE}(3)$ denote a transition from a released pose to its resulting stable pose. Pick-and-place with repeated regrasp is then formulated by extending (2) as:

$$\begin{aligned} &\text{find } X^j,\ j = 1, \cdots, N \\ &\text{s.t. } \mathcal{F}_G(\mathcal{S}^j) \ni X^{j,\text{post}} = X^{j+1,\text{pre}} \in \mathcal{F}_G(\mathcal{S}^{j+1}) \\ &\quad \pi(g^{j,\text{pre}}_{\mathbf{T}}) = g^{j,\text{post}}_{\mathbf{T}} \end{aligned} \tag{3}$$

where $j = 1$ and $j = N$ correspond to the initial and final states, respectively, while the remaining $N - 2$ states represent regrasp steps. The superscripts *pre* and *post* indicate the released object's states before and after settling, respectively. The first constraint ensures that regrasp is performed only when the target object is in a stable pose. Fig. 2a illustrates grasps in the pick-and-place tasks with regrasp.

B. Geometric Representations

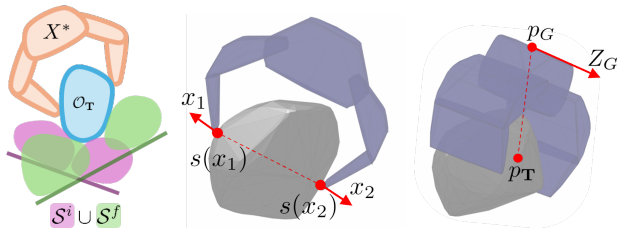
Solving (2) or (3) is challenging because $\mathcal{F}_G$ lacks a closed-form expression and π is non-differentiable, requiring costly dynamic simulations. In this work, we adopt differentiable geometric representations to efficiently address these challenges through differentiable optimization.

To this end, we represent the object geometries ($\mathcal{C}$ and $\mathcal{O}$) using the Differentiable Support Function (DSF) [8], which represents a wide range of convex shapes while providing differentiable features. DSF is based on the support function, defined for a convex body $\mathcal{O}$ as:

$$h_{\mathcal{O}}(x) = \sup_{s \in \mathcal{O}} x^\top s$$

The support point $s(x) \in \partial\mathcal{O}$ satisfies $h_{\mathcal{O}}(x) = x^\top s(x)$, where $\partial\mathcal{O}$ is the boundary of $\mathcal{O}$. The formulation of DSF [8] provides gradients of support points and surface normals.

Leveraging the differentiability of DSF, the framework in [9] computes collision distances and their derivatives with respect to the $\mathrm{SE}(3)$ poses of the bodies, attaining fast convergence via Riemannian optimization. These differentiable features facilitate optimization for robotic manipulation, including collision avoidance and contact enforcement [19].



(a) (b) (c)

Fig. 3: (a) Union of the two scenes, expressed in the target frame. (b) Illustration of the antipodal term. (c) Illustration of the alignment term.

Module	`GraspGen`	`StabPose`	`IkSolve`	`PathPlan`
Domain	$\mathrm{SE}(3) \times \mathbb{R}^{n_\theta}$	$\mathbb{S}^2$	$\mathbb{R}^{n-n_\theta}$	$\mathbb{R}^n$
Objective	grasp quality	support function	pose error	smoothness
Inequality	world col., finger lim.	-	world col., self-col., joint lim.	world col., self-col., joint lim.
Equality	target contact	-	-	-
Solver	AL + RTR	RTR	AL + TR	

TABLE I: Summary of the optimization problems for each module. The manipulator and gripper have n and n_θ degrees of freedom, respectively. Here, AL, TR, and RTR denote Augmented Lagrangian, Trust Region, and Riemannian TR, respectively; "col." denotes collision and "lim." denotes limit.

III. Differentiable Optimization-Based Modules

The pick-and-place problem in (2) and its regrasp extension in (3) can be addressed efficiently via differentiable optimization using features from Sec. II-B. Since the planning requires not only feasible grasps but also the corresponding manipulator configurations and paths, we decompose it into four modules. The `GraspGen` generates feasible grasps and invokes `StabPose` for intermediate stable poses if regrasp is required, while `IkSolve` and `PathPlan` compute manipulator configurations and plan paths for the goal grasp. All modules are formulated as differentiable optimization problems (Table I), forming the pipeline (Fig. 2b).

A. Grasp Generation

Our `GraspGen` generates collision-free grasps for both task scenes (e.g., pick/place or pick/release), but computing $\mathcal{F}_G(\mathcal{S}^i) \cap \mathcal{F}_G(\mathcal{S}^f)$ is intractable as $\mathcal{F}_G$ has no closed-form expression. We therefore construct the union $\mathcal{S}^i \cup \mathcal{S}^f$ in the target object's frame (Fig. 3a), and then map the resulting optimal grasp $X^* \in \mathcal{F}_G(\mathcal{S}^i \cup \mathcal{S}^f)$ back to each scene. The `GraspGen` module solves (1) via the optimization:

$$\max_X f(X; \mathcal{S}^i \cup \mathcal{S}^f) \text{ s.t. the constraints in (1)} \tag{4}$$

For two-finger grippers, we define a differentiable grasp quality using the contact normals x_1 and x_2:

$$\begin{aligned} f &= f_{\text{anti}} + f_{\text{align}} \\ f_{\text{anti}} &= -x_1^\top x_2 + x_1^\top \frac{s(x_1) - s(x_2)}{\|s(x_1) - s(x_2)\|} \\ f_{\text{align}} &= -\frac{1}{2}\left(Z_G^\top \frac{p_{\mathbf{T}} - p_G}{\|p_{\mathbf{T}} - p_G\|}\right)^2 \end{aligned}$$

Algorithm 1 `GraspGen` for pick-and-place

Input: Pick and place scenes $\mathcal{S}^i, \mathcal{S}^f$
Output: Feasible grasps X^i, X^f
Construct union scene $\mathcal{S}^i \cup \mathcal{S}^f$ in target object's frame
Initialize $k = 0, X_0$ and dual variables
repeat *AL loop*
    Initialize $l = 0$, Hessian approximation B_0
    repeat *RTR loop*
        Compute AL $\mathcal{L}$ and $\frac{\partial \mathcal{L}}{\partial X}$ from (4)
        Update $X_{k,l}$ via RTR method
        Update B_l by RBFGS algorithm
        $l \leftarrow l + 1$
    until $\min \mathcal{L}$ converged
    Update dual variables and penalty parameters
    $k \leftarrow k + 1$
until $f(X; \mathcal{S}^i \cup \mathcal{S}^f)$ converged and constraints satisfied
Map X^* to pick and place frames using $g^i_{\mathbf{T}}, g^f_{\mathbf{T}}$

where $s(x)$ denotes the support point of $\mathcal{O}_{\mathbf{T}}$ along the normal x, with p_G and $p_{\mathbf{T}}$ representing the centers of the gripper and the target object, respectively. The antipodal term, f_{anti} (Fig. 3b), is designed to enforce the force-closure condition under the assumption of two soft-finger contacts [20], [21]. The alignment term, f_{align} (Fig. 3c), is added to improve grasp alignment along the gripper's width direction Z_G.

The `GraspGen` solves the nonlinear constrained optimization problem (4) using the Augmented Lagrangian (AL) method [22], [23], which effectively handles constraints and ensures robust convergence. The AL subproblem, $\min \mathcal{L}$, is solved via the Riemannian Trust Region (RTR) method [24], with the Hessian approximated using the Riemannian BFGS (RBFGS) algorithm [25]. These Riemannian methods properly handle the $\mathrm{SE}(3)$ structure of the grasp X, which standard TR and BFGS do not consider. Consequently, `GraspGen`, summarized in Algorithm 1, generates optimal grasps robustly and efficiently without relying on sampling.

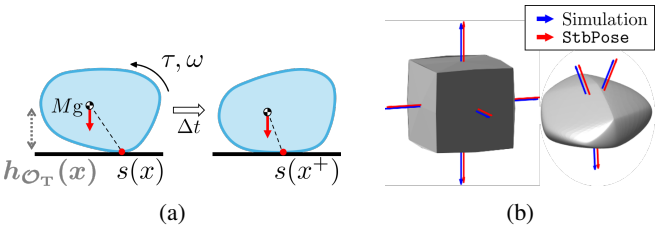
B. Stable Pose Prediction for Regrasp

When a regrasp is required—i.e., when no feasible grasp is found in $\mathcal{S}^i \cup \mathcal{S}^f$ by solving (4), as described in Sec. II-A—stable poses of the target object must be predicted for regrasp planning. To avoid costly simulations, we propose a geometry-based stable pose prediction module, `StabPose`.

On a flat surface, finding a stable pose reduces to determining the supporting plane normal, while the in-plane $\mathrm{SE}(2)$ components (planar translation and rotation about the z-axis) do not affect the grasp expressed in the target object's frame (Fig. 3a). Treating the surface as a supporting plane, we define x as the negative plane normal (Fig. 4a) and represent the object using the DSF, where $h_{\mathcal{O}_{\mathbf{T}}}(x)$ denotes the support function of $\mathcal{O}_{\mathbf{T}}$ along the direction x. Then, the target object orientation $R_{\mathbf{T}}$ is chosen such that:

$$R_{\mathbf{T}} \in \mathrm{SO}(3) \text{ s.t. } (R_{\mathbf{T}})_{3,:} = -x^\top \tag{5}$$

where $(\cdot)_{3,:}$ denotes the third row of a matrix. Under this approach, the gravitational potential energy is $\mathrm{Mg}\, h_{\mathcal{O}_{\mathbf{T}}}(x)$,



(a) (b)

Fig. 4: (a) Illustration of an object settling on a flat surface during Δt, with red arrows showing x and x^+ in (8), in the global frame. (b) Comparison plot of the physics simulation and `StabPose` for a cube and an arbitrary shape, with a small offset added to the arrows for clarity.

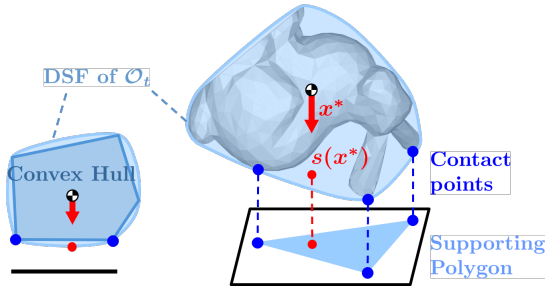


Fig. 5: Illustration of a stable pose satisfying the static stability criterion [26], with the supporting polygon (light-blue triangle). Blue and red dots show contact points and the DSF support point $s(x^*)$, along with their projections onto the plane. The inset shows the object's convex hull and DSF in 2D.

with Mg is the gravitational force on the object. This yields the unconstrained optimization problem for `StabPose`:

$$x^* = \underset{x \in \mathbb{S}^2}{\arg\min}\, h_{\mathcal{O}_{\mathbf{T}}}(x) \tag{6}$$

We solve (6) via RTR [24], [9] with fast convergence, seeking local minima of the object's center-of-mass height above the plane.

Note that the solution x^* from (6) may correspond to an unstable equilibrium, whereas stability can be assessed geometrically using principal curvatures [27]. For this purpose, we use the DSF [8]. A pose is considered stable if the minimum eigenvalue of the projected Hessian on the tangent space $T_{x^*}\mathbb{S}^2$ is numerically non-negative [9]; we check:

$$\Lambda_{\min}\left(P^\top \mathrm{Hess}\; h_{\mathcal{O}_{\mathbf{T}}}(x^*) P\right) > -\epsilon \tag{7}$$

where $P \in \mathbb{R}^{3\times 2}$ denotes a projection matrix onto $T_{x^*}\mathbb{S}^2$ and $\epsilon > 0$ is a sufficiently small constant used to handle near-zero values. Since (6) returns a local minimizer, we sample multiple initial vectors on $\mathbb{S}^2$ to explore all stable poses. We then perform clustering based on cosine similarity and filter out unstable directions using (7). Additionally, stable poses are ranked according to $\Lambda_{\min}(\cdot)$, enabling regrasp planning to begin from the most stable pose. The complete procedure is summarized in Algorithm 2.

Algorithm 2 `StabPose`

Input: Geometry of the target object $\mathcal{O}_{\mathbf{T}}$
Output: Ranked stable orientations $(R^*_{\mathbf{T}})_1, \cdots, (R^*_{\mathbf{T}})_m$
Sample initial vectors on $\mathbb{S}^2$
Solve (6) via RTR method for each initial vector
Cluster solutions based on cosine similarity
Filter and rank candidates using (7)
Convert m stable directions x^* into $R^*_{\mathbf{T}}$ by (5)

We compared the stable directions $x^*_1, \ldots, x^*_m$ obtained by Algorithm 2 with those from a physics-based simulation [28] and observed close agreement (Fig. 4b). Consequently, `StabPose` efficiently predicts stable poses and ranks them according to curvature-derived stability metrics, without relying on physics simulations.

Although the proposed formulation (6) is geometry-based, it implicitly captures the dynamic behavior of the object. More precisely, the `StabPose` module approximates the quasi-dynamics formulation [3], [29], which is widely used and can be expressed for objects settling on a surface as follows (Fig. 4a):

$$\begin{aligned} \tau(x) = & -\mathrm{Mg}\, s(x) \times x = \mathbf{I}\dot{\omega} + \omega \times (\mathbf{I}\omega) \\ x^+ \leftarrow & \; e^{[\omega \Delta t]_\times} x \end{aligned} \tag{8}$$

where τ is the torque due to the gravitational force, $\mathbf{I}$ is the object's moment of inertia, and ω is its angular velocity. The contact point of the object with the flat surface, denoted by $s(x)$, also corresponds to the support point in the x direction used in (5). The following proposition establishes that the `StabPose` module (6) approximates the quasi-dynamics equation (8).

Proposition 1: The optimality condition of (6) coincides with the equilibrium condition of (8), and under a first-order approximation, their update steps are identical.

Proof: The first-order optimality condition of (6) is given by:

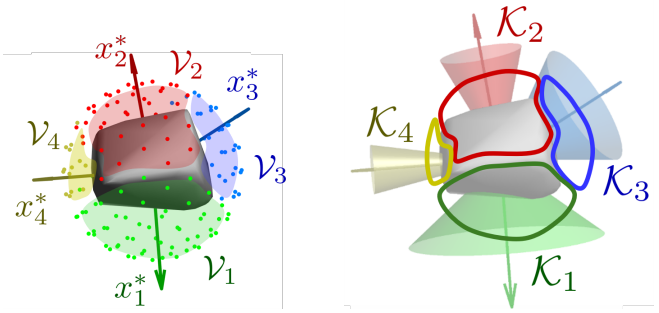
$$\mathrm{grad}\; h_{\mathcal{O}_{\mathbf{T}}}(x) = (I_3 - xx^\top) s(x) = 0$$

which is equivalent to the equilibrium condition of (8), i.e., $s(x) \times x = 0$. Moreover, the update step Δx in (8) can be written under a first-order approximation as:

$$\Delta x = \mathrm{Mg}\Delta t^2 \left(\mathbf{I}^{-1} (s(x) \times x) \right) \times x$$

showing that it is parallel to $\mathrm{grad}\; h_{\mathcal{O}_{\mathbf{T}}}(x)$. ■

Additionally, `StabPose` is applicable to non-convex objects. Consider such an object on a flat surface (Fig. 5), where x^* denotes the solution of (6). As shown in Proposition 1, x^* satisfies $s(x^*) \times x^* = 0$, implying that x^* is parallel to $s(x^*)$, the contact point with the plane. The supporting polygon, defined as the convex hull of actual contact points with the plane, corresponds to the convex hull face in contact with the plane. Due to the DSF's strict convexity, the projection of $s(x^*)$ onto the plane lies inside this polygon. Since static stability [26] requires the center of mass projection to lie within the supporting polygon, x^* also qualifies as a stable pose for the non-convex object.



(a) (b)

Fig. 6: (a) For each nominal direction x^*_k (arrows), the corresponding set $\mathcal{V}_k$ (dots) is shown in matching colors, exhibiting irregular boundaries. (b) Each set $\mathcal{V}_k$ is then conservatively modeled as an inscribed cone $\mathcal{K}_k$.

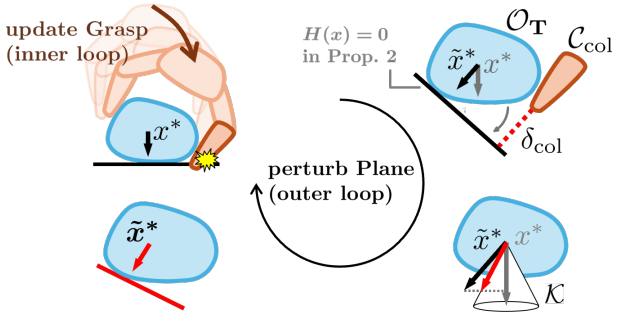


Fig. 7: Illustration of the plane perturbation scheme in `GraspGen`$^+$. If a collision exists between the release plane and any gripper link at the end of `GraspGen`'s inner loop, a collision-margin-aware perturbation is applied following Proposition 2, and the perturbed plane normal $\tilde{x}^*$ is then projected onto the cone $\mathcal{K}$. This process is executed iteratively within the outer loop of `GraspGen`$^+$.

C. Stable-Pose-Informed Non-Static Release

The stable poses from `StabPose` are used in `GraspGen` for the pick-to-release and regrasp-to-place stages. Meanwhile, restricting release to strictly stable poses (i.e., $g^{\mathrm{pre}}_{\mathbf{T}} = g^{\mathrm{post}}_{\mathbf{T}} = g^*_{\mathbf{T}}$) is often overly conservative, leading to infeasibility in regrasp planning. To mitigate this conservatism, we allow the object to be released in an unstable pose and then regrasped after it stabilizes (i.e., $g^{\mathrm{pre}}_{\mathbf{T}} \overset{\pi}{\mapsto} g^{\mathrm{post}}_{\mathbf{T}} = g^*_{\mathbf{T}}$), leveraging `StabPose` to incorporate settling behavior π. By appropriately relaxing the constraints, `GraspGen` is refined to yield `GraspGen`$^+$, enabling feasible non-static release.

Let $\mathcal{V} \subset \mathbb{S}^2$ be the discrete set of initial vectors sampled for (6), and let $\mathcal{V}_k \subset \mathcal{V}$ denote the discrete subset of initial vectors that converge to the k-th stable direction x^*_k. Thus, $\mathcal{V}_k$ can be regarded as a discrete sampling of the basin of attraction of the nominal direction x^*_k (Fig. 6a), i.e., any released pose $g^{\mathrm{pre}}_{\mathbf{T}}$ corresponding to $x \in \mathcal{V}_k$ settles into $(g^*_{\mathbf{T}})_k$. Since $\mathcal{V}_k$ is discrete with an irregular boundary, it does not admit a closed-form representation; we compute an inscribed cone $\mathcal{K}_k$ with axis x^*_k as an inner approximation of this basin of attraction, using the alpha-shape algorithm [30] on $T_{x^*_k}\mathbb{S}^2$ (Fig. 6b).

Algorithm 3 `GraspGen`$^+$ for non-static release

Input: Pick scene $\mathcal{S}^i$ and k-th stable pose $(g^*_{\mathbf{T}})_k$
Output: Relaxed release pose $g^{\text{pre}}_{\mathbf{T}}$
Initialize `GraspGen`
Compute cone $\mathcal{K}_k$ from $\mathcal{V}_k$ using alpha-shape algorithm
while AL loop not converged **do**
    while RTR loop not converged **do**
        Update grasp X by RTR step
    end while
    if collision between the release plane and $\mathcal{C}_{\text{col}}$ **then**
        Compute $\tilde{x}^*$ by solving (9)
        Project $\tilde{x}^*$ onto $\mathcal{K}_k$
    end if
end while
Convert perturbed direction $\tilde{x}^*$ into release pose $g^{\text{pre}}_{\mathbf{T}}$

As shown in Fig. 6a, each $\mathcal{V}_k$ is a discrete set of scattered unit vectors; therefore, solving `GraspGen` individually for each $x \in \mathcal{V}_k$ is inefficient, as it neglects their distribution near the nominal direction x^*_k. Instead, we perturb the release plane within the basin of attraction, avoiding collisions with $\mathcal{C}_{\text{col}}$ while maintaining contact with $\mathcal{O}_{\mathbf{T}}$ (Fig. 7). The following proposition enables efficient computation of this update.

Proposition 2: A common tangent plane of two disjoint bodies $\mathcal{O}$ and $\mathcal{C}$ always exists with a positive margin δ.

Proof: Define the residual support function:

$$H(x) := h_{\mathcal{O}}(x) - h_{\mathcal{C}}(x) - \delta + x^\top (p_{\mathcal{O}} - p_{\mathcal{C}})$$

where $p_{\mathcal{O}}$ and $p_{\mathcal{C}}$ denote the centers of the bodies, and $H(x) = 0$ defines the desired supporting plane. Consider two unit vectors:

$$x_0 := \frac{p_{\mathcal{O}} - p_{\mathcal{C}}}{\|p_{\mathcal{O}} - p_{\mathcal{C}}\|}, \quad x_1 := -x_0$$

so that $H(x_0) > 0$ and $H(x_1) < 0$. Let $\gamma : [0,1] \to \mathbb{S}^2$ be a smooth curve with $\gamma(0) = x_0$ and $\gamma(1) = x_1$. Since $H \circ \gamma$ is continuous on $[0,1]$ due to DSF properties, the Intermediate Value Theorem ensures some $t \in (0,1)$ with $H(\gamma(t)) = 0$. We obtain the solution $\tilde{x} = \gamma(t)$. ∎

Proposition 2 guarantees the existence of the desired supporting plane, which can then be obtained by solving:

$$\min_{x \in \mathbb{S}^2} \frac{1}{2} H(x)^2 \tag{9}$$

via RTR method. This solution yields the collision-margin-aware plane closest to the nominal stable plane corresponding to x^*_k. However, (9) alone does not guarantee that perturbations stay within the basin of attraction of x^*_k. To address this, we obtain the desired vector $\tilde{x}^*$ by projecting it onto the previously constructed inscribed cone $\mathcal{K}_k$ (Fig. 6b). As shown in Algorithm 3 and Fig. 7, this process is performed iteratively to update the plane for release. Thus, our support-function-based plane perturbation scheme extends `GraspGen` to `GraspGen`$^+$, enabling non-static release while guaranteeing that the object reliably reaches the nominal stable poses.

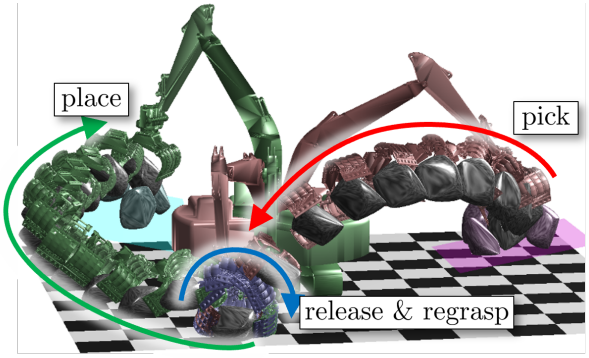


Fig. 8: Visualization of the excavator scenario for stone manipulation. The target stone (gray) is released for a subsequent regrasp. Colors follow the same convention as in Fig. 1.

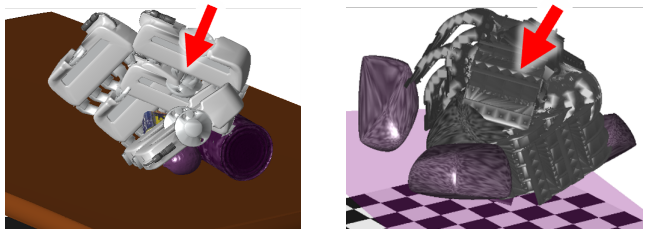


Fig. 9: Examples of sampled grasps used in the baseline for the two scenarios. Red arrows indicate the grippers' approach directions, and only 18 grasps are shown for clarity.

D. IK Solving and Path Planning

From the previous modules, we obtain feasible grasps for each scene. The `IkSolve` module computes a manipulator configuration to achieve the desired gripper pose via AL with TR, minimizing the gripper's SE(3) pose error while respecting joint limits and collision constraints.

The resulting goal configurations for each sub-task are interpolated by the `PathPlan` module, which formulates an optimization using differentiable features to generate a smooth path while avoiding collisions among the grasped object, manipulator, and surrounding obstacles.

IV. Evaluation

We conducted an ablation study in simulation using randomly generated scenes for the following two scenarios:

- The Franka PANDA manipulator handles YCB objects [31] (Fig. 1) with 8 degrees of freedom. In the pick scene, randomly selected objects, including the target, are placed on a table in random poses; in the place scene, they are placed upright on a shelf.
- The excavator model [32], equipped with a stone gripper [6], has a total of 7 degrees of freedom including the gripper. Multiple stones are positioned on inclined surfaces in the pick and place scenes, with each stone generated as an arbitrary convex shape.

Release and regrasp actions were performed in an obstacle-free area between the pick and place locations for both scenarios.

A. Ablation Study

We evaluated the following strategies:

1) **Baseline:** We replace `GraspGen` with a sampling-based method used in regrasp research [7], [10], [11], [12], sampling 50,000 grasps over gripper orientation, approach distance, and finger joints (Fig. 9).
2) **Direct pick-and-place:** This strategy attempts direct pick-and-place without any regrasp.
3) **Regrasp with static release:** Regrasp is performed using a strictly stable pose from `StabPose`, without any non-static release.
4) **Regrasp allowing non-static release:** This strategy employs `GraspGen`$^+$, enabling non-static release.
5) **Repeated regrasp:** Non-static regrasp is applied repeatedly as necessary (Fig. 2b).

Table II lists each strategy's modules: direct pick-and-place uses `GraspGen` only, static regrasp adds `StabPose`, and non-static or repeated regrasp adds `GraspGen`$^+$.

Strategy	`Grasp`	`Stab`	`Grasp`$^+$	`Grasp`$^+$(repeated)
Direct pick-and-place	✓			
Static regrasp	✓	✓		
Non-static regrasp	✓	✓	✓	
Repeated regrasp	✓	✓	✓	✓

TABLE II: Comparison of module usage across ablation-study strategies, with names abbreviated for brevity.

B. Results and Analysis

Table III shows that the success rates increase as we move from direct pick-and-place to static, non-static, and repeated regrasp. Our framework outperforms the sampling-based baseline in all scenarios. In complex scenes, the baseline struggles, as sampling can neither leverage prior information (e.g., object geometry) nor jointly consider both pick and place scenes. Examples of successful regrasp plans are shown in Fig. 1 and Fig. 8.

Strategy	Franka PANDA with YCB objects	Excavator with Stones
Baseline	75/300	43/300
Direct pick-and-place	120/300	103/300
Static regrasp	170/300	209/300
Non-static regrasp	201/300	263/300
Repeated regrasp	224/300	288/300

TABLE III: Success rates of pick-and-place tasks with different strategies in simulation. Baseline refers to the sampling-based grasping approach.

On an Intel Core i5-13600KF at 3.50 GHz, the computational time per trial in our C++ implementation was 180 ms for `GraspGen` and 2.66 ms for `StabPose`, while the remaining modules depend on the path discretization. The main causes of failure were an empty feasible grasp set in `GraspGen` due to obstacle enclosure, a numerically induced stable pose mismatch in `StabPose` due to DSF strict convexification, and kinematic unreachability in the remaining modules due to excavator joint limits.

Fig. 10: Example of stable poses for YCB objects exhibiting various shapes. Arrows indicate the stable normals from `StabPose`, with their corresponding cones. For the non-convex mug, a slightly enlarged convex hull (blue) is shown.

Fig. 10 illustrates examples of stable poses of YCB objects predicted by `StabPose`. For stones, we evaluated arbitrary convex shapes to validate the approach (Fig. 4b and Fig. 6). The stable poses obtained by `StabPose` serve as inputs to `GraspGen`$^+$, enabling non-static regrasp planning. By integrating geometry-based stable pose prediction with grasp optimization, our method enhances regrasp planning while maintaining collision-free and force-closure grasps in challenging environments.

To evaluate the computational efficiency of our approach, Table IV reports the average number of **eval-grasp()** calls per trial for each strategy. Each call performs collision detection between the relevant body pairs, including robot links and objects. Although the baseline invokes **eval-grasp()** once per sampled grasp, achieving comparable success rates would require orders of magnitude more samples (Table III). By contrast, our optimization-based strategies execute multiple **eval-grasp()** calls during optimization, yet the total number remains substantially lower than that of the baseline, highlighting the efficiency of our approach.

Strategy	Franka PANDA with YCB objects	Excavator with Stones
Direct pick-and-place	44	63
Static regrasp	247	570
Non-static regrasp	341	918
Repeated regrasp	379	1389

TABLE IV: Average number of **eval-grasp()** calls per trial for each strategy.

C. Real-World Experiment

To validate our approach, we conducted real-world executions of our planning framework. Given the geometry and poses of the target object and surrounding obstacles, the framework planned pick-and-place with regrasp, and the robot executed the resulting plan to pick, release, regrasp, and place the object while maintaining collision avoidance and force closure. Fig. 11 shows static release, and Fig. 1 illustrates non-static release during object settling.

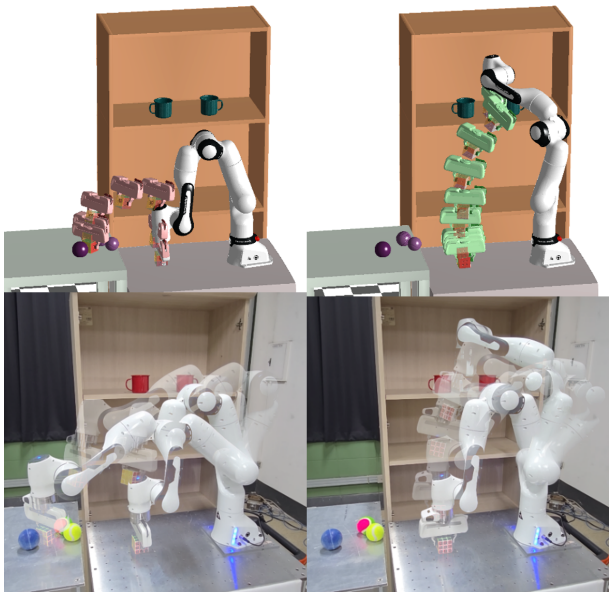


Fig. 11: Real-world execution demonstrating successful regrasp when direct pick-and-place was infeasible.

V. Conclusion

We presented a modular framework for pick-and-place with regrasp, enabling non-static and repeated regrasp planning through a sequential pipeline. By jointly considering both task scenes and predicting stable poses without simulation, it achieves efficient regrasp planning via differentiable optimization, as validated in simulation and real-world experiments.

A limitation is the requirement for predefined target objects and goal poses, as well as the inability to handle non-target rearrangement. Future work will enable the determination of target objects and goal poses for task-oriented manipulation. Its sampling- and simulation-free design allows the generation of large, feasible datasets for learning-based methods without extensive experiments.

References

[1] M. D. Shanthi and T. Hermans, "Pick and place planning is better than pick planning then place planning," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2790–2797, 2024.

[2] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*, pp. 297–307, PMLR, 2022.

[3] T. Pang, H. J. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4691–4711, 2023.

[4] Franka Emika PANDA. https://robodk.com/robot/Franka/Emika-Panda. Accessed: 2026-02-09.

[5] ROBOTIQ 2F-85. https://robotiq.com/products/adaptive-grippers#Two-Finger-Gripper. Accessed: 2026-02-09.

[6] engcon® SG26. https://engcon.com/en_us/tools/hydraulic-tools/stone-and-sorting-grapples.html. Accessed: 2026-02-09.

[7] M. Wermelinger, R. Johns, F. Gramazio, M. Kohler, and M. Hutter, "Grasping and object reorientation for autonomous construction of stone structures," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5105–5112, 2021.

[8] J. Lee, M. Lee, and D. Lee, "Uncertain pose estimation during contact tasks using differentiable contact features," in *Robotics: Science and Systems*, 2023.

[9] S. An, S. Lee, J. Lee, S. Park, and D. Lee, "Collision detection between smooth convex bodies via Riemannian optimization framework," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 12464–12471, 2024.

[10] W. Wan, H. Igawa, K. Harada, H. Onda, K. Nagata, and N. Yamanobe, "A regrasp planning component for object reorientation," *Autonomous Robots*, vol. 43, no. 5, pp. 1101–1115, 2019.

[11] K. Wada, S. James, and A. J. Davison, "ReorientBot: Learning object reorientation for specific-posed placement," in *2022 IEEE International Conference on Robotics and Automation*, pp. 8252–8258, 2022.

[12] U. A. Mishra and Y. Chen, "ReorientDiff: Diffusion model based reorientation for object manipulation," in *2024 IEEE International Conference on Robotics and Automation*, pp. 10867–10873, 2024.

[13] A. H. Li, P. Culbertson, and A. D. Ames, "Toward an analytic theory of intrinsic robustness for dexterous grasping," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2992–2999, 2024.

[14] J. Chen, Y. Ke, and H. Wang, "BODex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization," in *2025 IEEE International Conference on Robotics and Automation*, 2025.

[15] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic, *et al.*, "Deep learning approaches to grasp synthesis: A review," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3994–4015, 2023.

[16] H. Sekkat, O. Moutik, L. Ourabah, B. ElKari, Y. Chaibi, and T. A. Tchakoucht, "Review of reinforcement learning for robotic grasping: Analysis and recommendations," *Statistics, Optimization & Information Computing*, vol. 12, no. 2, pp. 571–601, 2024.

[17] R. Li, C. Esteves, A. Makadia, and P. Agrawal, "Stable object reorientation using contact plane registration," in *2022 IEEE International Conference on Robotics and Automation*, pp. 6379–6385, 2022.

[18] X. Pang, F. Li, N. Ding, and X. Zhong, "Upright-Net: Learning upright orientation for 3D point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14911–14919, 2022.

[19] J. Lee, S. Park, M. Lee, and D. Lee, "Efficient gradient-based inference for manipulation planning in contact factor graphs," in *2025 IEEE International Conference on Robotics and Automation*, 2025.

[20] V. D. Nguyen, "Constructing force-closure grasps in 3D," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 240–245, 1987.

[21] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC Press, 2017.

[22] M. R. Hestenes, "Multiplier and gradient methods," *Journal of Optimization Theory and Applications*, vol. 4, no. 5, pp. 303–320, 1969.

[23] M. J. D. Powell, "A method for nonlinear constraints in minimization problems," *Optimization*, pp. 283–298, 1969.

[24] P. A. Absil, C. G. Baker, and K. A. Gallivan, "Trust-region methods on Riemannian manifolds," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 303–330, 2007.

[25] C. Qi, K. A. Gallivan, and P. Absil, "Riemannian BFGS algorithm with applications," in *Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization*, pp. 183–192, Springer, 2010.

[26] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, 1968.

[27] R. Mason, E. Rimon, and J. Burdick, "Stable poses of 3-dimensional objects," in *Proceedings of International Conference on Robotics and Automation*, vol. 1, pp. 391–398, 1997.

[28] J. Lee, M. Lee, and D. Lee, "Modular and parallelizable multibody physics simulation via subsystem-based ADMM," in *2023 IEEE International Conference on Robotics and Automation*, 2023.

[29] M. T. Mason, *Mechanics of robotic manipulation*. MIT Press, 2001.

[30] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 2003.

[31] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *2015 International Conference on Advanced Robotics*, pp. 510–517, IEEE, 2015.

[32] DEVELON DX225. https://na.develon-ce.com/en/construction-equipment/crawler-excavators/dx225lc-7-crawler-excavator. Accessed: 2026-02-09.