Dense Robotic Packing of Irregular and Novel 3D Objects

Fan Wang and Kris Hauser, *Senior Member, IEEE*

***Abstract*—Recent progress in the field of robotic manipulation has generated interest in fully automatic object packing in warehouses. This paper presents a formulation of the robotic packing problem that ensures stability of the object pile during packing and the feasibility of the robot motion while maximizing packing density. A constructive packing algorithm is proposed to address this problem by searching over the space of item positions and orientations. Moreover, a new heightmap-minimization heuristic is shown to outperform existing heuristics in the literature in the presence of non-convex objects. Two strategies for improving the robustness of executed packing plans are also proposed: 1) conservative planning ensures plan feasibility under uncertainty in model parameters, and 2) closed-loop packing uses vision sensors to measure placement errors and re-plans to correct for them. The proposed planner and error mitigation strategies are evaluated in simulation and on a state-of-the-art physical packing testbed. Experiments demonstrate that the proposed planner generates high-quality packing plans, and the error mitigation strategies improve success rates beyond an open-loop baseline from 83% to 100% on five-item orders.**

***Index Terms*—Robotics, Bin packing, Warehouse Automation**

I. Introduction

With the unprecedented growth of the E-Commerce market, warehouse automation has attracted much interest and capital investments. Advances in the field are supported by the technical progress made in robotic manipulation and vision, as demonstrated by recent competitions like the Amazon Robotics Challenge. Compared to a conventional labor-intensive approach, an automated robot warehouse brings possible benefits such as increased uptime, higher total throughput, and lower accident rates. In particular, one emerging area of warehousing automation that attracts a lot of research attention is automated packaging or packing, a process during which robots stow objects into small confined spaces, such as shipping boxes.

In this paper, we focus on package fulfillment, which packs customer orders of arbitrary shapes and weights into a single shipping box. Currently, this is primarily performed by human workers in a warehouse, which leaves the container size selection largely to intuition. As a result, over-sized containers are often used, incurring waste and high shipping costs(Fig. 1). On the other hand, an automated system that optimizes container sizes and packing plans executable by a robot could lead to major cost savings for large warehouses.

F. Wang is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, 27708 USA. e-mail fan.wang2@duke.edu

K. Hauser is with the Department of Computer Science, University of Illinois at Urbana-Champaign. e-mail kkhauser@illinois.edu

This work is supported by an Amazon Research Award.

Manuscript received April 19, 2005; revised January 11, 2007.

(a) (b)

Fig. 1. (a) An example of poor space utilization in an actual shipping order delivered to the authors, while higher packing density can be achieved with algorithm that optimizes container sizes. (b) Five items are packed into a small container using our system

Robot pick and stow systems have been demonstrated in the past, but are insufficient to address the stated warehouse packing problem. Examples of such system include those from Amazon Robotics Challenge [1]–[3], where the stowing task asks a set of items to be placed in an over-sized container. Under this relaxed setting, packing can be addressed with simple heuristics or even dropping items from top center. Recent systems have also explores tight packing, like the *peg-in-hole* problem, where the robot employs vision and sensing capabilities to insert an object into a tight space [4], [5]. However, these systems focus on the success of inserting one object into a cluttered space and provide no guarantee if all items in a set will fit within the container.

Instead, we address the problem of dense placement planning for a set of complex-shaped objects into a single container, while ensuring the containment of all objects and feasibility of the packing motion when the packing motion is executed with a robot. Traditionally, problems that involve the placement of objects within a container or a set of containers are referred to as cutting and packing problems. The vast majority of prior work in packing addresses a restricted set of objects, e.g., rectilinear objects, under the two standard packing constraints:

1) *Noninterference.* Each object is collision free,
2) *Containment.* All objects are placed within the internal space of the container.

However, to perform automatic packing with robots, several real-world issues need to be addressed, such as stability under force of gravity, and kinematics and clearance issues for the robot. Instead, we consider the following *robot-packable* constraints:

3) *Stability.* Each object is stable against previously packed objects and the bin itself, and

4) *Manipulation feasibility.* A feasible robot motion exists to load the object into the target placement. The robot must obey kinematic constraints, grasp constraints, and collision constraints during this motion.

If stability is not considered, the pile may shift during execution, and therefore subsequent placements are unlikely to be executed as planned. If kinematics and clearance are not considered, the robot may be asked to perform infeasible motions (e.g., grip an item from underneath, bring an item to a target through another interlocked item, or pass through the container wall).

We present an offline packing planner that is able to address non-convex objects under robot-packable constraints. The contributions of this planner include:

1) A polynomial time constructive algorithm to implement a resolution-complete search amongst feasible object placements, under *robot-packable constraints*.
2) A 3D positioning heuristic named Heightmap-Minimization (HM) that minimizes the volume increase of the object pile from the loading direction.
3) A fast prioritized search scheme that first searches for robot-packable placement in a three-dimensional space that likely contains a solution, and falls back to search in a five-dimensional space.

Our algorithm is tested against others in a realistic physics simulator using 3D scanned models of real-world items. In these tests, approximately 97% of 10-item placement plans were successfully executed in the physics simulator, compared to an 86% success rate from a standard packing solver. Empirical results also show that the new Heightmap-Minimization heuristic finds more placements than existing strategies, such as the Deepest-Bottom-Left-Fill (DBLF) heuristic.

Beyond the packing algorithm, we develop a physical packing testbed to study the various sources of uncertainty and their effects on packing executions. These uncertainties include object modeling errors, such as incorrectly estimated object shapes, poses, deformation, etc., manipulation errors, such as failed grasps, object reorientation during grasping, and calibration error. Unstable placements and infeasible robot motions may result in failure to contain all items, and even cause damages to the robot and items. A failed packing is usually difficult and time-consuming to recover from, e.g., remove all placed items in the old box and replace them in a larger box.

We develop two strategies to mitigate the impact of uncertainties and increase the robustness of the packing execution. The first strategy is to plan conservatively during the offline planning phase to ensure execution feasibility under positioning error. The second strategy is to employ closed-loop vision and manipulation so that a robot can dynamically react to errors and correct them. Specifically, the system re-senses the object pile after each placement using visual and depth information and then replans the best subsequent placements if the pile deviates from the plan significantly.

We test these strategies in both Monte Carlo simulation as well as on the physical testbed. Simulation testing focuses on examining the results from controlled magnitudes of errors, while testing on the physical platform evaluates typical errors encountered using state-of-the-art perception algorithms (Fig. 2). They show that our strategies raise the packing success rates in 5-item packing to 98% from the open-loop baseline of 83%.

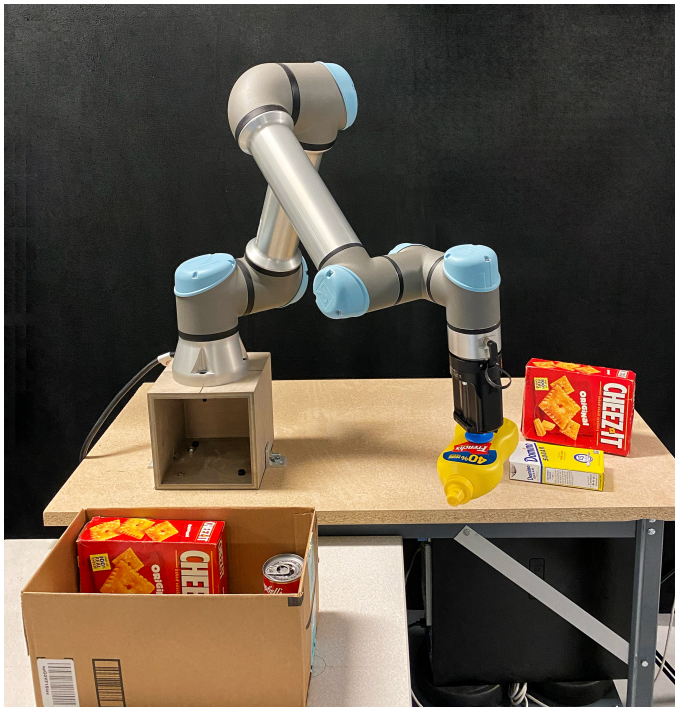

Fig. 2. The experimental packing setup consists of a UR5 robot equipped with E-pick suction gripper, and overhead cameras (not pictured). Errors from pose recognition, camera-robot calibration, box location calibration, grasp planning, the center of mass estimation, and geometric modeling affect the overall success rate of packing.

This paper is an extension of a prior conference publication [6]. It introduces experiments on a physical packing testbed, examination of execution success rates under disturbances, and novel uncertainty mitigation strategies (Secs. IV, V, VI.C, and VI.D).

II. Related Work

A. Robot picking and stowing

Recent years have seen increasing interest in end-to-end robot pick and stow system, driven by a high demand in warehouse automation and supported by the technical progress made in the field of robotic manipulation and vision.

Some robot pick and stow system have been demonstrated in recent competitions like the Amazon Robotics Challenge (ARC). The competition requires stowing of arbitrary items into a storage system, picking specific items, and packing them into boxes. Several such end-to-end systems are published including system from team NimbRo [2], team MIT [1], etc. However, the packing problem was not so challenging in the ARC, because there was ample space in packing boxes and relatively few items (2-5). For non-dense packing, simple heuristics such as simply dropping objects into the container from above, or packing using object bounding boxes and heuristics, work well [2].

For dense packing, recent works employ vision and sensing capabilities to insert an object into a tight space as an extension to the *peg-in-hole* problem. For example, Ye et al. [4] develop a real-time state estimation system that can recover the pose and contact formation of a rectilinear object in grasp relative to its environment. By fusing visual and force sensing, they demonstrate the application of inserting an object picked by a suction cup into a tight space. Shome et al. [5] designed a complete robot packing system using minimalistic hardware stack of a single robot with suction gripper equipped with RGB-D cameras. They have employed different techniques to achieve robust packing that minimizes failure conditions, One example of such technique is using the suction gripper as a push finger that executes three pre-defined key manipulation primitives to tight packing in against perception and positioning errors. Dong et al. [7] used a high-resolution tactile sensors to control the insertion of objects while estimate and correct for residual position uncertainties to insert the object into a designated gap without disturbing the environment.

The key difference between peg-in-hole approaches and ours is the former focuses on the success of inserting one object into a cluttered space and nearly no planning is involved, while ours focus on the placement planning for all items in a known set and guarantee fitness of the items into the container chosen. And the packing planning for a set of irregular-shaped objects fall into the classical problem of cutting and packing.

B. Cutting and packing problem

Most existing research on cutting and packing handles floating 2D and 3D rectilinear objects. Under some simplified packability constraints, these problems can be formulated and solved optimally using exact algorithms. One example is the solution to the 3D bin packing problem using branch and bound, proposed by Martello et al. [8], [9], whose work is further extended by many including Boef et al. [10] and Crainic et al. [11]. Exact algorithms, although capable of finding the optimal solution if infinite time is spent, are strongly NP-hard [12] and do not guarantee optimal results within a reasonable amount of time, especially when a large number of instances are involved [9]. Heuristic methods and metaheuristic approaches have also been developed over the years, such as the popular Bottom-Left heuristic [13] and the Best-Fit-Decreasing heuristic [14].

More recent work has addressed irregular shape packing, and only heuristic methods are practical because the search space is infinite. Metaheuristics such as Simulated Annealing (SA) [15]–[17] and Guided Local Search (GLS) [18]–[22] are optimization methods that start with an initial placement and iteratively improve the placement by moving the pieces in the neighborhood while minimizing an objective function (e.g., overlap in the system). Other work has also proposed constructive positioning heuristics for 3D irregular objects, such as Deepest-Bottom-Left-Fill (DBLF), which places items in the deepest, bottom-most, left-most position; and Maximum Touching Area (MTA), which places an item in a position that maximizes the total contact area of its faces with the faces of other items [23].

Some research has taken aspects of stability into consideration during packing. Egeblad et al., for example, use a two-stage GLS packing algorithm that, in the first stage, optimizes for the center of gravity and inertia of the pile and, in the second stage, minimizes overlap in the system [19]; Liu et al. propose a constructive method that packs irregular 3D shapes using a Minimum-Total-Potential-Energy heuristic [17]. This method performs a grid search for the lowest gravitational center height Z for each placement. However, both proposed methods use heuristics only and do not verify the stability of each placement. In contrast from these works, our method enforces stability explicitly using constraints.

We also know of one packing work that takes into account robot manipulation feasibility [10], in which the author proposes a variant of the orthogonal 3D box packing scheme such that no prior packed box is in front of, to the right of, or above the current placing box, to avoid possible collision with a vacuum gripper. This placing rule, however, cannot guarantee collision-free placements with other gripper geometries and neglects robot kinematic constraints and graspability constraints.

To the best of our knowledge, ours is the first packing work to solve for stability and robot-feasibility constraints simultaneously. Moreover, these constraints can be solved for arbitrary shaped, non-convex 3D objects.

III. Packing Planner

This section describes the packing problem definition and the proposed packing algorithm. It also describes the new Heightmap Minimization (HM) placement heuristic.

A. Problem definition

The problem addresses offline packing of 3D irregular shapes into a single container while ensuring the stability of each packed item and feasibility of the placement with a robot gripper. Specifically, for a set N geometries $\mathcal{G}_1, \ldots, \mathcal{G}_\mathcal{N}$ where $\mathcal{G}_i \subset \mathcal{R}^3$, let $\mathcal{C}$ donate the free space volume of the container and ∂C as the boundary of the free space. Let $T_i \cdot \mathcal{G}_i$ denote the space occupied by item i when the geometry is transformed by T_i. The problem is to find a placement sequence $S = (s_1, \ldots, s_N)$ of $\{1, \ldots, N\}$ and transforms $\mathcal{T} = (T_1, \ldots, T_N)$ such that each placement satisfies non-overlapping constraints with geometries placed prior:

$$(T_i \cdot \mathcal{G}_i) \cap (P_j \cdot \mathcal{G}_j) = \emptyset,\ \forall i, j \in \{1, \ldots, N\}, i \neq j, \quad (1)$$

and the containment constraint:

$$T_i \cdot \mathcal{G}_i \subseteq \mathcal{C},\ \forall i \in \{1, \ldots, N\}. \quad (2)$$

Moreover, each placed item $k = 1, \ldots, N$ must satisfy stability constraints:

$$\texttt{isStable}(P_{s_k} \cdot \mathcal{G}_{s_k}, \mathcal{C}, T_{s_1} \cdot \mathcal{G}_{s_1}, \ldots, P_{s_{k-1}} \cdot \mathcal{G}_{s_{k-1}}) \quad (3)$$

and manipulation feasibility constraints:

$$\texttt{isManipFeasible}(P_{s_k} \cdot G_{s_k}, P_{s_1} \cdot \mathcal{G}_{s_1}, \ldots, P_{s_{k-1}} \cdot \mathcal{G}_{s_{k-1}}). \quad (4)$$

Stability is defined as the condition in which all placed items are in static equilibrium under gravity and frictional contact forces. We model the stack using point contacts with a Coulomb friction model with a known coefficient of static friction.

Manipulation feasibility checks feasibility of a packing pose when executed by a robot manipulator. This requires that the object be graspable from its initial pose and can be packed in the desired pose via a continuous motion, without colliding with environmental obstacles.

It is important to note that both stability and manipulation feasibility constraints must be satisfied for *every intermediate arrangement* of objects, not just the final arrangement.

B. Packing algorithm

Our constructive packing algorithm solves for the set of robot-packable constraints proposed. It accepts an itemset, a container dimension, a constructive positioning heuristic, and/or a packing sequence, to produce packing plans. The pipeline packs each item to its optimized feasible pose in sequential order, without backtracking.

Our pipeline primarily consists of four components, namely:

1) Sequencing heuristic
2) Generate transforms ranked by the positioning heuristic
3) Stability check
4) Manipulation feasibility check

The pipeline starts with a sequencing heuristic to sort all items into a packing sequence. It then attempts to plan an optimal feasible loading of each item into the container, where a grid-based search is used to exhaustively explore candidate placements, and placement quality is measured by a heuristic.

Because exploring a grid in the 6D space of transforms SE(3) is slow, we implement a prioritized method (Alg. 1) that first searches in a 3D space of *planar-stable* orientations (Lines 3–8). Planar-stable orientations are those in which a geometry G can stably rest on a planar surface, and are computed using the method of Goldberg et al. [24]. This speeds up the search for the common case of packing on the first layer and on horizontal support surfaces.

If a feasible placement is found in this space, then the item is considered packed, but if not, the item is placed on a fall-back list U. After placing all other objects, the planner attempts to place items in U (Lines 9–21) by seeking a 5D space including rolls and pitches. The 6th dimension, Z, is always determined analytically by finding the lowest loading height for any orientation and X-Y position.

The algorithm for selecting a placement for a single item is given in Alg. 8, which is parameterized by a set of rolls and pitches are given as input. It generates a set of collision-free placements $\mathcal{T}$, then sorts them by the positioning heuristic, and finds the first placement that satisfies stability and robot-packability constraints.

1) Placement sequence: The placement sequence can be user-specified or generated by a heuristic, such as the non-increasing bounding box volume heuristic, which is known to lead to fast convergence of branch-and-bound algorithms [9]. The generated sequence is subject to adjustment if a solution cannot be found in the specified ordering.

Algorithm 1: Robot-feasible packing with fall-backs

input : Item geometries $\mathcal{G}_1, \ldots, \mathcal{G}_N$, container C, and masses/frictions
output: Transforms $\mathcal{T}$ and sequence S, or **None**

1 Initialize $\mathcal{T}, S, U$ to empty lists;
2 Generate initial packing sequence $\{s_1, \ldots, s_N\}$;
3 **for** $s_i \in \{s_1, \ldots, s_N\}$ **do** `// 3D search`
4 Get planar-stable rolls and pitches for $\mathcal{G}_{s_i}$ with the top n highest quasi-static probabilities $O_{s_i} \leftarrow \{(\phi_1, \psi_1), \ldots, (\phi_n, \psi_n)\}$;
5 $T = \texttt{packOneItem}(G_{s_i}, C, O_{s_i}, S, \mathcal{T})$;
6 **if** T **then** Add T to $\mathcal{T}$, Add s_i to S;
7 **else** Add s_i to U;
8 **end**
9 **for** $u_i \in U$ **do** `// 5D fall-back search`
10 Let $\{(\phi_1, \psi_1), \ldots, (\phi_n, \psi_n)\}$ be the planar-stable orientations in O_{u_i};
11 **for** $t_r \in \{0, \Delta r, 2\Delta r, \ldots, 2\pi - \Delta r\}$ **do**
12 **for** $t_p \in \{0, \Delta r, 2\Delta r, \ldots, 2\pi - \Delta r\}$ **do**
13 $O^t = \{(\phi_1 + t_r, \psi_1 + t_p), \ldots, (\phi_n + t_r, \psi_n + t_p)\}$;
14 $T = \texttt{packOneItem}(G_{u_i}, C, O^t, S, \mathcal{T})$;
15 **if** T **then**
16 Add T to $\mathcal{T}$; Add u_i to S;
17 **continue** with Line 11
18 **end**
19 **end**
20 **return** ***None***
21 **end**
22 **return** $(\mathcal{T}, S)$

Algorithm 2: `packOneItem`

input : Item geometry $\mathcal{G}$, container $\mathcal{C}$, pitches and yaws O, sequence of packed items $\{s_1, \ldots, s_i\}$, transforms of packed items $\{P_1, \ldots, P_i\}$
output: Transform T or **None**

1 $\mathcal{T} \leftarrow \texttt{3DGridSearch}(\mathcal{G}, \mathcal{C}, O)$;
2 Score each T in $\mathcal{T}$ based on heuristic used;
3 **for** *up to* N *lowest values of* T *in* $\mathcal{T}$ **do**
4 **if** $\neg \mathit{isStable}(T \cdot \mathcal{G}, \mathcal{C}, P_1 \cdot \mathcal{G}_{s_1}, \ldots, P_i \cdot \mathcal{G}_{s_i})$ **then continue**;
5 Compute grasp poses $T_1^{\mathcal{G}}, \ldots T_n^{\mathcal{G}}$ compatible with T;
6 **if** $\mathit{isManipFeasible}(T \cdot \mathcal{G}, \{T_1^{\mathcal{G}}, \ldots T_n^{\mathcal{G}}\})$ **then return** T;
7 **end**
8 **return** ***None***

2) Generating ranked transforms: For a given item, a positioning heuristic identifies a free pose inside the container that is most preferred according to a specific criterion. Our pipeline accepts arbitrary positioning heuristics, but we have found best performance using the HM heuristic introduced in Sec. III-C.

The 3D search for collision-free placements of one object, given a set of rolls and pitches, is shown in Alg. 3. A grid search is performed for yaw, X, and Y at a given resolution, and the height Z of the placement is analytically determined as the lowest free placement. 2D heightmaps are used to accelerate the computation of Z to an efficient 2D matrix manipulation. Three heightmaps are computed: 1) a top-down heightmap H_c of the container and placed objects, 2) a top-down heightmap H_t of the object to be placed, and 3) a bottom-up heightmap H_b of the object to be placed. H_t and H_b are measured relative to the lower left corner of the orientated object. Raycasting is used to build these heightmaps, and rays that do not intersect with the object geometry are given height 0 in H_t and ∞ in H_b. The container heightmap is obtained once at the beginning of object placement search, and an object heightmap is computed once for each distinct searched orientation.

Given an object orientation and X, Y location, we calculate the lowest collision-free Z as follows:

$$Z = \max_{i=0}^{w-1} \max_{j=0}^{h-1} (H_c[x+i, y+j] - H_b[i,j]) \tag{5}$$

where (x, y) are the pixel coordinates of X, Y, and (w, h) to be the dimensions of H_t.

Algorithm 3: `3DGridSearch`

input : Geometry $\mathcal{G}$, container $\mathcal{C}$, rolls and pitches O
output: All collision-free candidate transforms

for $(\phi, \psi) \in O$ **do**
  for $\theta \in \{0, \Delta r, 2\Delta r, \ldots, 2\pi - \Delta r\}$ **do**
    Let $R \leftarrow R_z(\theta) R_y(\phi) R_x(\psi)$;
    Discretize legal horizontal translations of $R \cdot \mathcal{G}$ into grid $\{(X_1, Y_1), \ldots (X_n, Y_n)\}$;
    for (X, Y) *in* $\{(X_1, Y_1), \ldots (X_n, Y_n)\}$ **do**
      Find the lowest collision free placement Z at translation X, Y;
      $T \leftarrow (R, (X, Y, Z))$;
      if $T \cdot \mathcal{G}$ *lies within* $\mathcal{C}$ **then** Add T to $\mathcal{T}$;
    end
  end
end
return $\mathcal{T}$

Once all collision-free candidate transforms are obtained, they are scored by the positioning heuristic. For example, the Deepest-Bottom-Left-First heuristic can be formulated as the score:

$$Z + c \cdot (X + Y) \tag{6}$$

where c is a small constant.

The candidates are then ranked by score (lower is better). If only robot-packable constraints are required, the placement candidate with the lowest score is returned.

After a new object has been placed, we update the heightmap of the container H_c. This subroutine is also used in our heightmap minimization heuristic. Given a pose X, Y, Z of the object to be packed, and the top heightmap H_t at the given orientation, we calculate an updated heightmap H'_c adding the placed object as follows:

For all $i = 0, \ldots, w-1$, $j = 0, \ldots, h-1$, we let:

$$H'_c[x+i, y+j] = \max(H_t[i,j] + Z, H_c[x+i, y+j]) \tag{7}$$

if $H_t[i,j] \neq 0$, and otherwise

$$H'_c[x+i, y+j] = H_c[x+i, y+j]. \tag{8}$$

3) Stability checking: Let the set of contact points be denoted as $c_1, \ldots, c_K$, which have normals $n_1, \ldots, n_N$, and friction coefficients $\mu_1, \ldots, \mu_K$. For each contact c_k, let the two bodies in contact be denoted A_k and B_k. Let $f_1, \ldots, f_K$ denote the contact forces, with the convention that f_k is applied to B_k and the negative is applied to A_k. We also define m_i as the mass of object i, and cm_i as its COM. We take the convention that the container has infinite mass.

The object pile is in static equilibrium if there are a set of forces that satisfy the following conditions.

Force balance: $\forall i = 1, \ldots, N$,

$$-\sum_{k \mid i = A_k} f_k + \sum_{k \mid i = B_k} f_k + m_i g = 0. \tag{9}$$

Torque balance: $\forall i = 1, \ldots, N$,

$$-\sum_{k \mid i = A_k} (cm_i - c_k) \times f_k + \sum_{k \mid i = B_k} (cm_i - c_k) \times f_k = 0.$$

Force validity: $\forall k = 1, \ldots, K$,

$$f_k \cdot n_k > 0, \tag{10}$$

$$\|f_k^{\perp}\| \leq \mu_k (f_k \cdot n_k). \tag{11}$$

where $f_k^{\perp} = f_k - n_k(f_k \cdot n_k)$ is the tangential component (i.e., frictional force) of f_k.

For a given arrangement of objects, an approximate set of contact points is obtained by slightly scaling geometries and determining the set of geometric features that overlap when scaled. We also perform a clustering step in which we merge all contact points within a grid size of 1 cm, which reduces the number of contacts to a more manageable number. A pyramidal approximation for the friction cone is used, and the conditions above are formulated as a linear programming problem over $f_1, \ldots, f_N$, solved using the convex programming solver CVXPY [25]. If no such forces can be found, the arrangement is considered unstable.

4) Manipulation feasibility: Our planner is limited to checking top-down placement trajectories, as robots performing pick and place (e.g., box packing) commonly use vertical motion [10]. We also assume the existence of a grasp generator that produces some number of candidate end effector (EE) transforms, specified relative to an object's geometry that may be used to grasp the object. The instantiation of this generator used in our implementation is described in Sec. IV-D. The pseudo-code for the manipulation feasibility checker is given in Alg. 4.

Algorithm 4: `isManipFeasible`

input : Desired placed geometry $T \cdot \mathcal{G}$ and a set of grasp candidates $\{T_1^{\mathcal{G}}, \dots T_n^{\mathcal{G}}\}$

1 **for** $T^{\mathcal{G}} \in \{T_1^{\mathcal{G}}, \dots T_n^{\mathcal{G}}\}$ **do**
2 Compute top-down EE path $\mathcal{P}_{ee}$ interpolating from an elevated pose to a final pose $T \cdot T^{\mathcal{G}}$;
3 **for** $P_{ee} \in \mathcal{P}_{ee}$ **do**
4 **if** $\neg($`IKSolvable`$(P_{ee}) \wedge$ `inJointLimits`$(P_{ee}) \wedge$ `collisionFree`$(P_{ee}))$ **then go to** Line 1;
5 **end**
6 **return** True
7 **end**
8 **return** False

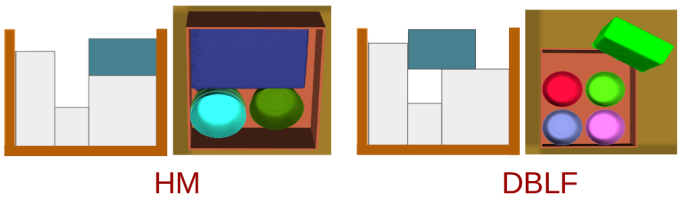

Fig. 3. Example packing placements obtained by the heightmap (HM, ours) and Deepest-Bottom-Left-Fill (DBLF) heuristic orders. The HM heuristic nests the four bowls inside one another, leaving room for the box.

C. Heightmap-Minimization Heuristic

The performance and solution quality of a multi-dimensional packing problem is highly susceptible to the item-positioning rule [26]. However, existing positioning heuristics for 3D packing are scarce and are commonly adapted directly from 2D packing, and therefore result in poor space utilization in the 3D container [11], [27]. To address these shortcomings, we propose a novel positioning heuristic called the Heightmap-Minimization (HM) heuristic, which favors item placements that result in the smallest occupied volume in the container, as observed from the loading direction.

Specifically, HM scores a placement as follows. Given the candidate transform $T = (roll, pitch, yaw, X, Y, Z)$, compute a tentative container heightmap H'_c using the update routine described in Sec. 12. Suppose its shape is (w, h). The score for the placement using the HM heuristic is:

$$c \cdot (X + Y) + \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} H'_c[i, j] \tag{12}$$

where c is a small constant.

HM favors positions and orientations that result in good space utilization as it minimizes wasted space and holes that cannot be filled. HM also favors stable placements since the bottom of the object is encouraged to match the shape of the supporting terrain (Fig. 3).

IV. Packing System Implementation

Next, we describe the implementation of our packing algorithm on a physical robot testbed. The system components used are representative of current state-of-the-art in sensing, perception, and planning for warehouse manipulation.

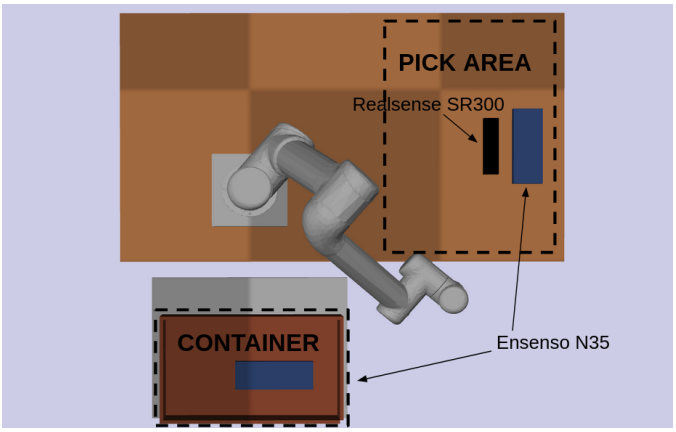

Fig. 4. Top-down diagram of the packing experiment. Objects to be packed are arranged in the picking area, and must be packed in the container (e.g., a shipping box). Overhead cameras (Realsense SR300 and Ensenso N35) provide color and depth information.

A. System overview

A 6R industrial robot (Universal Robot UR5e) is equipped with a suction gripper (Robotiq EPick) and a compliant suction cup (Fig. 2). One RGB-D camera (Realsense SR300) and two high-resolution stereo / structured light black-and-white depth cameras (Ensenso N35) are used. An RGB-D / stereo pair is mounted overhead looking down over the picking area, and the other stereo camera is mounted looking down the packing area, shown in Fig. 4. This setup allows us to combine both the color information from the RGB-D camera and the high-quality point clouds from the stereo cameras. The high-quality point cloud enables the pose recognition pipeline to achieve better accuracy compared to the RGB-D data.

The entire system layout is illustrated in Fig. 4. In the picking area, objects are placed in arbitrary locations. The system assumes that objects are drawn from a known set, whose 3D models are available. In addition, all objects in the picking area will be packed, with their initial locations not touching. In the packing area, a predetermined box footprint area is marked off, and a box with matching dimensions is placed by hand to align with the footprint.

The software pipeline consists of calibration, perception, and planning. The calibration steps include:

- Factory-specified tool center point and camera intrinsic parameters were used.
- Camera extrinsic calibration using robot-mounted checkerboard
- The box footprint location is calibrated by jogging the robot to touch the footprint corner locations. The footprint corners are set to the tool center point coordinates relative to the robot base.

Object identification and pose estimation are then performed with the following steps:

1) Plane segmentation removes the tabletop from the point cloud.

2) Region growing segmentation [28] is used to segment object point clouds.
3) Pointcloud segments are projected onto the RGB image to obtain an object ROI.
4) The object is identified using a convolutional neural network model [29], trained on the experiment objects.
5) A CNN pose estimation model gives an initial estimate of object pose [30].
6) If fitness [31] of the CNN estimation is below a threshold, point-cloud only template matching is performed.
7) The pose is fine-tuned with iterative closest points (ICP) [32].

The packing algorithm presented in the prior section plans nominal object packing sequences, grasp locations, packing poses, and top-down loading paths assuming a known location of the container and all items. Two forms of sensor feedback are available in this system:

- Force feedback, via the force/torque sensor on the UR5e's wrist, is used for guarded moves during loading. A force threshold, set to 20 N in our experiments, prevents crushing items.
- RGB-D and stereo cameras provide an overhead image of how items have been packed in the container.

B. Pose recognition

The pose recognition pipeline estimates P_i^0 for $i \in \{1, \ldots, N\}$ for use in the planner.

First, we acquire partial pointcloud segments for $\mathcal{G}_1, \ldots, \mathcal{G}_\mathcal{N}$. This is done through a process of plane segmentation and distance calculation that removes all observation not on the picking station. The remaining pointcloud is segmented by Point Cloud Library (PCL)'s [33] implementation of region growing segmentation [28]. The algorithm partitions the points that are close enough in terms of a smoothness constraint.

Next, the partitioned pointcloud segments are projected on the color image obtained with a RealSense camera. Pixels inside the bounding box of the projects are cropped into a color image of each individual object. The ID of the object is classified with ResNet18 [29], trained on the experiment dataset using Pytorch [34].

For the experiment, we assume that $\forall i \in \{1, \ldots, N\}, \mathcal{G}_i$ is sitting on a tabletop with pose P_i^0, where the orientation component of P_i^0 is a *planar stable* orientation for $\mathcal{G}_i$. Therefore, pose recognition estimates the translation of $\mathcal{G}_i$ while R_i^0 is from a finite subset modulo rotations about the vertical axis. We initialize the estimate R_i^0 from the Dense Fusion [30] algorithm. Inputs to Dense Fusion include a pointcloud, cropped color image, and ID for the corresponding object segment. If fitness [31] for this estimation is below a certain threshold, a template matching technique is used as a fallback. Template object point clouds are generated at planar-stable orientations, and the best-fit match to the point cloud observation is selected. Whether Dense Fusion or template matching is used for the initial alignment, we follow this with a dense point-to-plane Iterative Closest Point (ICP) [32] alignment for fine-tuning. This is performed using the implementation from Open3D [31].

Fig. 5. Dynamic acquisition of 3D object models from a single top-down scan. The sensed upper surface is completed with a flat bottom.

C. Dynamic model acquisition

In a practical warehouse setting, a packing system may need to cope with hundreds or thousands of unique items, where accurate 3D models may not be available in advance. To address this setting, strategy V4+DMA modifies the system pipeline to perform dynamic model acquisition at packing time. The acquired model may not be as accurate as a pre-acquired model, but it avoids the need for a separate object identification and 6D pose estimation step. Furthermore, dynamic model acquisition can better cope with shape differences between prior models and the presented item, such as variations in packaging or damaged objects.

Our dynamic model acquisition approach requires no additional sensor setup under the assumption that items are not touching in the pick area. 3D models are acquired under the assumption that the item has a flat bottom and sits on the tabletop. Plane removal and point cloud segmentation are performed, and then each pointcloud segment is completed by concatenating surface points with its projection on the underlying plane. The completed point cloud then goes through a triangulation algorithm such as Poisson surface reconstruction [35] to obtain a watertight triangular mesh model (Fig. 5). An object's mass and center of mass are estimated from the acquired model assuming a constant uniform density of 0.5 g/cm^3

Drawbacks of dynamic model acquisition include noise and missing depth data from certain materials (e.g., reflective, transparent) that appear invisible to the depth sensor. Moreover, our current system infers the occluded portion of items using a somewhat naïve approach of assuming a flat supporting surface, which can overestimate the shape of irregular-shaped, concave items (e.g., bowls). 3D shape completion algorithms [36] can improve the inference somewhat, but errors are still likely to exist in the occluded regions. These shape estimation errors may lead to potential instability in packing plans and loss of packing density, but these problems may be avoided by ensuring that occluded surfaces are placed onto planar areas during loading.

D. Grasp planning

For grasp planning, we use a point cloud-based planner that generates a set of vacuum grasp candidates, ensuring that the grasp location is nearly flat, the suction direction is nearly normal to the surface, and the center of mass is almost underneath the grasp point (Fig. 6).

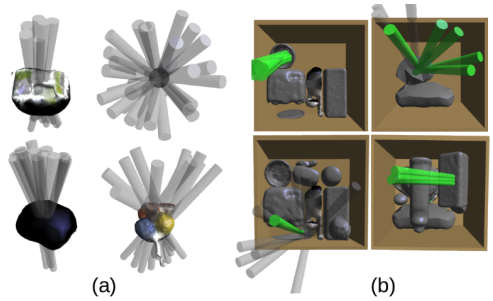

Fig. 6. (a) Grasp poses generated satisfying vacuum graspbility constraints. (b) Compatible gripper poses with candidate object orientation are checked for clearance with the container and the object pile. Collision-free grasps are shown in green and colliding grasps are colored in transparent grey.

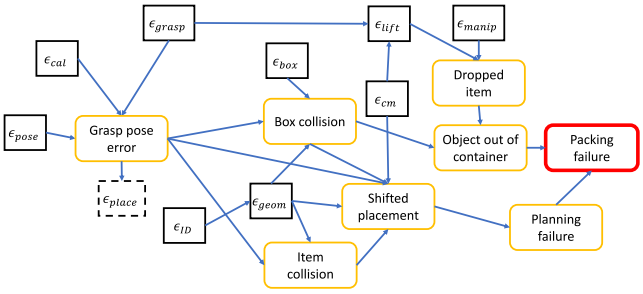

Fig. 7. A diagram of how error sources (rectangles) contribute to unexpected events (rounded rectangles) and ultimately to packing failure. We use ϵ_{place} as a measure for grasp pose error.

Specifically, the grasp point must lie within a radius r = 2 cm in the horizontal plane to the centroid of the observed point cloud segment. The areas under the gripper should be planar areas (80% of the surface points directly below the tool are within 0.3 cm to the estimated plane), which helps the vacuum opening to grasp normal to the surface.

For use in the planner, the set of grasp candidates is tested one by one for manipulation feasibility, i.e., with the chosen grasp pose, the robot can transfer $\mathcal{G}_i$ from P_i^0 to P_i without collision and with successful IK solutions being found along the trajectory. Moreover, in the packing pose, the resulting gripper axis needs to be within a tilting angle $\theta = \pi/4$ to the Z-axis to prevent excessive torques being applied at the gripper.

V. Packing Errors and Mitigation Strategies

Here we identify sources of error in a physical packing system that may lead to failed packing. We also present two mitigation strategies tested in our experiments.

A. Factors affecting packing success

Packing success is defined by a final state having all objects unbroken and packed within the container's dimensions.

Because the planner performs collision and stability checking, if the world behaved exactly as modeled, then the executed plan would be guaranteed to yield a successful packing. However, several errors are introduced by the execution pipeline.

We identify and describe nine key **error sources**:

1) Camera calibration error ϵ_{cal}.
2) Object identification error ϵ_{ID}.
3) Pose recognition error ϵ_{pose}.
4) 3D geometry modeling errors ϵ_{geom}.
5) Center of mass specification errors ϵ_{cm}.
6) Grasp positioning error ϵ_{grasp}.
7) Grasp acquisition (lifting) error ϵ_{lift}.
8) In-transit manipulation error ϵ_{manip}.
9) Box placement error ϵ_{box}.

Camera calibration error in the picking area causes deviations in the robot-relative object pose estimates, adding to problems acquiring a grasp. Camera calibration over the container leads to collisions with objects and walls of the container in the closed-loop packing strategy.

Object identification error leads the planner to choose a different 3D reference model than the actual object's geometry. In packing setups where the correct subset of items must be picked from a larger collection, this could lead to more serious errors where the wrong item is packed. Our problem definition assumes that all presented objects must be packed, so object identification error is correctable during packing.

Pose recognition error leads to misalignment of the true object geometry and the 3D reference model used in grasp and packing planning. This causes grasp failures and inadvertent collisions during loading.

The 3D geometry of each object was acquired using a 3D scanner and multiple viewpoints. There is inevitable noise in these models, particularly noticeable in the underside of the object. Moreover, the scanned items were newly bought objects, and as items exhibited wear and tear during our experiments, some items underwent slight deformations. Although we do not have ground truth, we expect overall geometry errors to be less than 5 mm.

Centers of mass of objects are not measured precisely, and may also change with the object orientation (e.g., product settling in a box). This causes potential problems with grasp acquisition and inaccurate stability tests in the planner.

Grasp positioning errors occur when the suction gripper shifts the object during grasping. The related grasp acquisition errors occur when the gripper cannot achieve adequate suction to lift the object. In-transit manipulation errors occur when the robot prematurely loses grip of the object, e.g., by causing large accelerations.

Finally, box placement errors can cause collisions with the sidewalls during packing.

Overall, these errors form cascading effects down the pipeline to contribute to unexpected events, such as toppling object placements or unexpected collisions, which lead to packing failures, as shown in Fig. 7. Informally, we refer to events in which an object hits a container wall or other objects prematurely as "smashing," and events in which an object falls over or shifts as "toppling." Note that the packing loop is performed N times, where N is the number of objects, so the system has an opportunity to measure and correct for errors accumulated during packing.

B. Error reduction methods

We propose two methods to mitigate the impacts of error on the execution success of packing plans. The first strategy is closed-loop packing, which makes use of sensing feedback to adjust the packing plan so that if an object is grasped incorrectly or falls over during placement, subsequent steps can adapt to the new knowledge. The second strategy is robust planning, which predicts the future effects of uncertainty so that unexpected behaviors are less likely to occur. To evaluate the effectiveness, we perform experiments with five variations of planning and executing strategies.

1) **V1: Baseline:** Plans are generated and executed, assuming perfect models.
2) **V2: Robust planning:** The packing planner is modified to avoid tight fits.
3) **V3: Closed-loop vision:** The container sensor re-senses the object pile after each placement. If the heightmap of the pile deviates from the plan, then a replan is triggered.
4) **V4: Robust planning and closed-loop vision:** Both V2 and V3 strategies are employed.
5) **V4+DMA: Dynamic model acquisition:** The V4 strategy is used, but with object identification and pose estimation replaced with dynamic model acquisition.

These strategies do not consider significant rearrangement of objects within the container (e.g., repacking), although strategies V3 and V4 do allow the robot to push objects that have stuck outside the container. We describe these strategies in more detail below.

We implement a grasping controller based on end effector force feedback, which is used in all experiments. During grasping, if the robot's force sensor detected that the object was not successfully lifted, the robot will attempt the next computed grasp pose that can achieve the planned object placement with a collision-free loading motion. Moreover, while packing an object into the container, if an immediate contact force exceeding 20N is sensed by the force/torque sensor, the object will be released. This avoids crushing objects during "smash" events.

C. Dynamic re-planning

In closed-loop packing, we monitor and remodeled the object pile after each placement. A failure is detected if the pile shifted significantly from expected, preventing subsequent items from being executed as planned. In case of failure, we replan the packing locations of the remaining items while maintaining consistency in the previous plan where possible. We also detect whether an object is lying partially outside of the container, and perform a *push maneuver* to attempt to push the object back in.

Specifically, with the overhead depth camera, we capture a pre-placement depth map of the container. If the incoming item's 3D model at its planned packing location is in collision with the pre-placement heightmap, we count the number of colliding pixels. A "significant deviation" is triggered when this count exceeds a threshold of 20% of pixels belonging to the item.

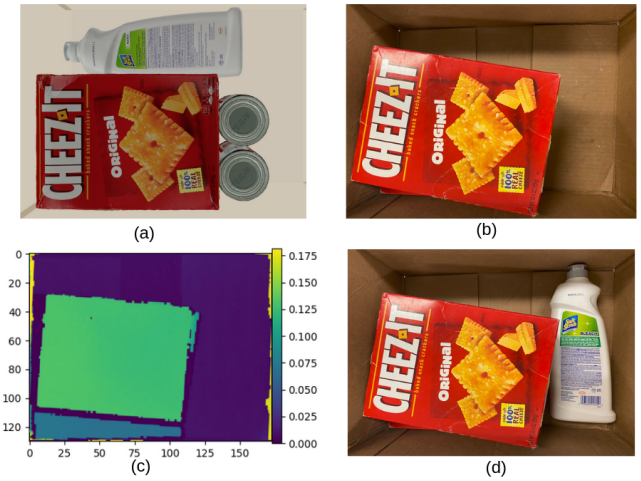

Fig. 8. Closing the loop around vision leads to more robust packing. (a) Offline plan. The two *cracker boxes* are stacked first, followed by *bleach* and two *soup cans*. (b) The second *cracker box* is shifted during packing, blocking the access for *bleach*. Open-loop packing would lead to a failure. (c) The closed-loop strategy captures a heightmap of the placed pile, detects a violation for the *bleach* placement, and plans a new location. (4) *Bleach* is placed successfully in the empty location.

If no significant deviation is determined, the item is placed to the planned transform as usual, which helps maintain coherence with the existing plan. If a significant deviation is detected, the strategy replans a new optimal placement for the infeasible object given the pre-placement heightmap. Rather than trying to identify the pose of each object in the pile, which is prone to significant errors, we simply treat the heightmap as the geometry of an object rigidly-fixed to the container. The use of the sensed heightmap as a collision geometry suffices for collision checking between the robot and gripper, and stability checking still verifies whether there is sufficient friction between a rigid pile and subsequent placed objects. However, treating the pile as a rigid, unmovable object incurs some loss of information for the stability constraint checks, so the planner may decide to put the object in a location an underlying object may shift or topple.

To perform the push maneuver, we capture a post-placement heightmap, and detect whether any object's horizontal extent surpasses the dimensions of the container. If so, we aim the robot's gripper to push in a straight line perpendicularly to the box wall, with the end effector's lowest portion slightly above the box's maximum height. This movement is aimed at the geometry center of the portion of the object outside the container dimensions.

D. Robust planning

Robust planning avoids tight fits with a threshold distance of δ. Placements closer than δ to the container wall and placed objects will be penalized in the offline planner.

Previously, a candidate placement is scored as follows as in 12. In robust planning, we add an extra term to the scoring

Fig. 9. Conservative geometry margins reduce inadvertent collisions. (a) A non-conservative plan causes a failure during the execution of the plan (b). Specifically, the shifted *cracker box* caused a failure and crash of the *sugar* item planned on top of it. (c) A conservative plan with δ =1 cm margin. During execution, all items are placed within the container without causing an inadvertent collision (d).

function to penalize tight fit:

$$c \cdot (X + Y) + \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} H'[i,j] + C/\max(d_{min}, \delta) \quad (13)$$

Where d_{min} is the minimum horizontal distance between the item and the pile or the container wall. This is computed from the heightmaps of the pile and the item being placed. To compute the minimum distance, we only consider pixels in the pile/container heightmap whose pixel height is larger than object placement height Z and x-y location within and near the object 2D projection onto the x-y plane. The minimum of all horizontal distances between object pixels and the pile/container pixels is taken as d_{min}. The value C is taken to be a large constant such that C/δ dominates the other two terms in the sum.

It should be noted that conservative planning may fail to find a plan when non-conservative planning would succeed. To address these cases, the planner tries to replan with a smaller δ that decreases linearly until it is 0. If the $\delta = 0$ case fails, we declare planner failure.

E. Closed-loop packing and robust planning

In strategy V4, we combine both closed-loop repacking and robust planning. This is mostly a straightforward combination, except that we note that when δ is decreased due to a planning failure, we maintain the decreased value for subsequent planning steps.

VI. Analysis and Experimentation

We evaluate the efficacy of the planning algorithm compared to other algorithms. We also evaluate the proposed packing system to study the sources and magnitudes of errors. We quantify the impact of such errors on the overall packing success rate in Monte Carlo simulation and on the physical testbed.

Fig. 10. 15 items from the YCB object dataset are used in experiments.

Algorithm parameters used in the experiments include: heuristic constant $c = 1$; heightmap resolution 0.002m; step size in both X and Y 0.01m; $\Delta r = \pi/4$ in range $[0, \pi)$; friction coefficient $\mu = 0.7$. Contact points are obtained using the exact geometry with a scale factor of 1.03. The top four planar-stable rolls and pitches with the highest quasi-static probabilities are used, and candidate number N = 100. Offline planner tests are conducted on Amazon Web Services instance type m5.12xlarge, and computation times are measured on a single thread. Simulation tests use the Bullet physics simulator [37] to simulate executions, and are run on several desktop PCs with 32GB of RAM and Intel I7 processors.

For simulation testing, objects were drawn at random from a set of 94 real-world object meshes from the YCB [38] and the APC 2015 object set [39]. On average each mesh contains 10,243 vertices. For testing on the robot, 15 real-world items from the YCB object dataset [40] are selected that are mostly rigid, opaque, graspable, visible to the depth camera, and in a wide range of shapes and weights (Fig. 10).

A. Comparing planner performance in simulation

The robot model used to verify robot feasibility constraints is a Staubli TX90 robot, equipped with a cylindrical vacuum gripper of 30 cm length and 2 cm diameter.

We perform stress tests on itemsets of size 10. A tall container of size $32 \times 32 \times 30$ cm is chosen. 1,000 itemsets of size 10 are generated and verified with all tested methods to have a non-overlap packing within the chosen container. Since the tilted gripper is likely to collide with the tall container chosen, we assume the gripper can grasp object of any orientation at the center of object's top surface, with the gripper axis vertically aligned to the Z axis.

We compare our HM heuristic against the DBLF and MTA heuristics [23], as well as an implementation of a guided local search (GLS) method as described by Egeblad et al. [19]. The fast intersection area theorem in Egeblad's paper was not implemented. Therefore, for the fairness of the comparison, GLS was run with five random restarts, and each restart was

Fig. 11. Examples of packing plans for itemsets of size 10.

TABLE I
COMPARING PLANNING TECHNIQUES ON 10-ITEM ORDERS WITH AND WITHOUT ROBOT-PACKABLE CONSTRAINTS

	HM	DBLF [23]	MTA [23]	GLS [41]
Success, non-overlap (%)	99.9	98.4	88.9	78.9
Time, non-overlap (s)	15.7	14.2	14.1	502
Success, all constraints (%)	97.1	96.3	86.3	—
Time, all constraints (s)	34.9	50.1	95.4	—

terminated after 300 s if a solution could not be obtained. GLS is also not tested for all constraints, as implementing robot-packable constraints in GLS methods is very challenging. Table I reports the percentage of solutions found and the average computation time.

Empirically, HM finds more solutions than any other method in comparison. With robot-packable constraint, HM finds 99.9% of all feasible solutions, leading the 2nd place DBLF heuristic by 1.5%, while MTA and GLS are not as competitive. After adding all constraints, each technique drops in success rate by a few percents, but HM still leads the other methods. The mean running time of HM is also 30% shorter, indicating that the highest ranked placements are more likely to be stable than the other heuristics.

In addition, only 3.2% (320 out of 10,000) of the items are packed with the fallback procedure, Of these, 51% were packed by adjusting the packing sequence of the item, and the other 49% were packed by performing the search in 5D. Indicating the 3D space searched is indeed highly likely to contain robot-packable solutions. And the prioritized search in 3D space is significantly more efficient than always searching in 5D as our other tests indicate that simply performing the 5D search has an average running time of 522 s, which is 15 times slower.

The performance of the algorithm is also impacted by the parameters chosen. With finer rotation granularity Δr and more candidates to check against robot-packable constraints, the success rate can be further improved at the cost of increased computation time (Table II).

To compare feasibility of the packing plans produced by ours and other planners, we test the open-loop execution feasibility in the Bullet physics simulator. The simulated robot places one item after another using a top-down loading direction, and if a 20 N resisting force is reached, the robot will drop the item and retract. The robot places all items 1 cm elevated from their planned transform; so items are expected to drop 1 cm. We allow 20 s for an item to settle before the next item is placed. A plan is considered a success if all items are fully within the container's bounding box.

TABLE II
IMPACT OF Δr AND CANDIDATE NUMBER N ON THE RESULTS

Rotation granularity Δr	$\pi/4$	$\pi/4$	$\pi/8$
Number of candidates N	100	500	500
Success, all constraints (%)	97.1	97.5	98.7
Time, all constraints (s)	34.9	70.05	89.40

In the 10 item case, 948 out of 971 (≈98%) of robot-packable plans obtained using the HM heuristic are executed successfully. This is significantly higher than the 83% success rate without robot-packable constraints.

B. Uncertainty evaluation

We collected a dataset of all experiment items placed in various poses and positions on the picking area, complete with RGB images, depth images, and pointclouds, taken by Realsense and Ensenso cameras in their overhead mounting positions. Around 200 poses and positions are taken for each object. 70% of the dataset was used for training, and the remaining 30% of the dataset was withheld for testing.

1) Object identification: In our test set, object segmentation was 100% successful because the input objects were placed in isolation. Our object recognition pipeline achieved a 98.5% top-1 accuracy on the testing set, which indicates that for 7.2% of 5-item order sets, at least one of the objects will be misidentified. Note that because all objects in the packing area need to be packed, a misidentified object will not necessarily lead to failed packing. Instead, it contributes to a (potentially substantial) geometry modeling error.

2) 6D pose estimation: To obtain the pose estimation error, the 6D pose estimation result from our pipeline is further corrected by manual alignment of the projected and ground truth (using the Meshlab software). We considered using an alternative approach to pre-define a ground truth pose and manually align the object to this pose, but we found that symmetric objects (e.g., boxes, cylinders) induced large pose error measurements, even though these errors would have virtually no effect on picking and packing. On our test set, the average translation and rotation error is 0.7 cm and 0.3 rad.

3) Rare errors: Grasp acquisition errors were not observed in our setup (so $\epsilon_{lift} \approx 0$), in large part because we chose objects from the YCB dataset that were more easily lifted by suction. In-transit manipulation errors never occurred during our experiments ($\epsilon_{manip} \approx 0$). Box placement error is also quite low ($\epsilon_{box} \approx 0$) compared to other sources of error, e.g., calibration, because it is relatively simple to align the box to within a millimeter or two of the calibrated footprint. Moreover, it would be a simple matter to add rigid fixtures to the packing area to constrain the box even further. Since these errors are so low, we do not include them in our analysis.

4) Summary statistic: placement error: Observe that calibration, pose estimation, and grasp positioning errors all contribute to the pose of the object-in-hand, which leads to an ultimate error in where the object is packed. Specifically, this is the final translation and rotation error of an object from a planned packing location P_i, assuming no collisions occurred during loading. We call this summary error *placement error*

TABLE III
SUMMARY OF EMPIRICALLY EVALUATED ERROR SOURCES

Error source	Magnitude
ϵ_{ID}	1.5%
ϵ_{pose}	0.7 cm (translation), 0.3 rad (orientation)
ϵ_{geom}	Estimate $\approx$ 0.5 mm
ϵ_{place}	1.02 cm (translation), 0.41 rad (orientation)
ϵ_{lift}	≈ 0
ϵ_{manip}	≈ 0
ϵ_{box}	≈ 0

ϵ_{place}, and can be evaluated experimentally. The evaluation performs 80 single item manipulations by a known transform that transforms object from one planar orientation to another. Before the manipulation, the pointcloud of the object is captured and the pose of the segment is estimated. After manipulation, the pointcloud of the object is captured again, and the observation is manually aligned to the projected object pose. ϵ_{place} is taken to be the RMSE of these alignment errors. Our experiments show that placement error is on average 1.02 cm in translation and 0.41 rad in rotation.

A summary of all error sources is given in Tab. III.

C. Monte-Carlo simulation testing

A more extensive set of experiments were conducted in simulation that introduce different magnitudes of errors and larger test sets. Varying degrees of error from ground truth object poses and depth sensor readings are used as input to each of our packing strategies.

For these tests, we use the same object sets as in the previous simulation test and for consistency we also use the same item sets across the range of packing strategies. For picking, we rigidly attach objects to the robot's gripper with a randomly-generated pose error, using the same standard deviations as ϵ_{place}, but scaled with an error scaling factor from 0-200%. The object is dropped when the robot reaches the desired placement pose, or a 20 N force threshold is reached.

The packing success rates for strategies V1–V4 are plotted in Fig. 12. For V2 and V4, a δ value of 1 cm is used. At 0% error, all experimental variations performed similarly, just under 98% success rate, while closed-loop strategy performs slightly worse. This can be largely attributed to the treatment of the pile as a rigid, infinite-mass object during replanning, which fails to capture the nuances of the pile's stability.

As the error increases beyond 100%, the success rate for the baseline strategy drops off sharply. Meanwhile, the rate of drop off is noticeably slower for the other three comparison strategies that employ robustness measures. The best performer is V4 that uses both robust and dynamic replanning, achieving the highest or near highest success rate at all positioning errors introduced, and maintain a packing success rate at 94.5% at 2.04 cm translation error and 0.82 rad in rotation error.

It should be noted that increasing δ does not always lead to better performance under uncertainties. A δ of 2 cm was also tested in V2 and V4, achieving similar and slightly worse performance than a δ of 2 cm. With a higher margin, planning is more likely to fail, and hence our system will fall back to use a smaller margin.

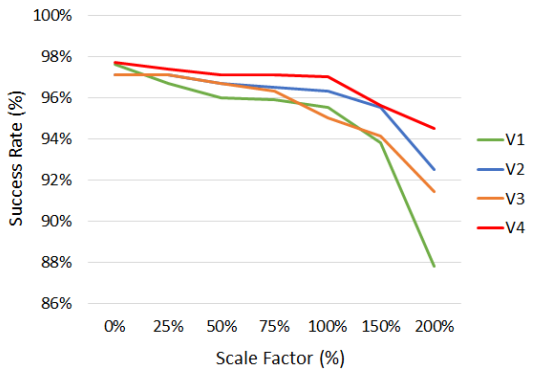

Fig. 12. Simulation packing success rates for 10-item orders with varying pose error. Pose error scaling factor is given on the horizontal axis, and the success rate is given on the vertical axis. The baseline (V1) success rate drops off dramatically as pose error increases, while the robust technique (V4) is much less sensitive to error.

TABLE IV
SUCCESS RATES ON THE PHYSICAL PACKING PLATFORM, 5-ITEM ORDERS

	V1	V2	V3	V4	V4+DMA
Success rate (%)	83	95	95	98	100
Max force exceeded (# per order)	2.22	1.40	1.12	0.7	0.53
Push performed (# per order)	—	—	0.07	0.03	0.03

D. Testing on physical platform

For testing on the physical platform, we generated 100 sets of 5-item orders from our experiment items. The robust planning strategies (V2 and V4) use δ =1.02 cm. Tab. IV summarizes the results. Both closed-loop and robust methods (V2 and V3) perform significantly better than the baseline, while the combined method (V4) beats the baseline success rate 98% to 83%. V4 with dynamic model acquisition performs best of all, with 100% of packing problems successfully executed.

We also show the number of times the maximum loading force (20 N) was exceeded, which indicates approximately how often a smash event was observed and how much damage the object sustained during packing. The same pattern is observed, showing that both closed-loop and robust strategies decrease the rate of smashing. The last row shows how many times a closed-loop push maneuver was performed, showing that these are rather rare, but are still helpful to improve success rates by a few percentage points.

We recorded object classifications, estimated poses, and planned and final placements, which allows us to examine causes of cascading failures more closely. In V1, nine failures occurred with one or more items protruding outside the box footprint, while seven failed for objects within the container but exceeding the maximum height. A final case failed when an object tipped over and stood up underneath the gripper, causing the robot to report a "maximum load exceeded" fault. Some failure cases can be traced to one predominant factor. Four failures can be traced back to failed object identification, which leads to wrong shape estimation of the object to pack. Three cases can be traced back to significant pose estimation error (e.g., rotation of 90 degrees, etc.). One case can be traced back to object tipping, which led to sudden force changes too

Fig. 13. Example of failed packing executions, for each experimental condition. In the baseline (V1), all items are planned tightly, and the *cracker box* hit a container wall, causing it to tilt, all subsequent items placed on top were smashed. In robust planning (V2), although margins were enforced in planning, a substantially incorrect pose estimation caused the *bleach* to stick out of the container, and no action is taken to correct it. In closed-loop packing (V3), the first-placed *cracker box* caught on the side and failed to be pushed in. This was the only case when the push maneuver did not work in our examples. Although this caused a failed plan, re-planning prevented the subsequent items from smashing onto the *cracker box*. With both closed-loop packing and robust planning (V4), a replan was triggered after the *cracker box* and before the *bleach* item. Because the pile was treated as a fixed, rigid object, the *bleach* was placed on top of the *cracker box*, but since it was unstable, this caused it to tip over.

rapid to be caught by the force control loop. The remaining failure cases are caused by cascading errors, e.g., a slightly misplaced object tipped over the object underneath, resulting in subsequent objects exceeding the container height.

In V2, three cases failed when one or more item protruding outside the box footprint, and the other two failed during toppling events, causing items to exceed the max height. Among the failure cases, two can be traced back to misidentified objects, while one can be traced back to significant pose estimation error.

In V3, all five cases failed due to exceeding the max height. All objects protruding outside the box footprint were corrected by pushing maneuvers. Among the failure cases, two can be traced back to misidentified objects, and two can be traced back to significant pose estimation error.

In V4, the two failure cases appear similar – the *cracker box* item toppled and exceeded the max height. One case was caused by a significant pose estimation error, while in the other, the *bleach* item was pushed onto the *cracker box*, and the impact caused it to tip over.

When dynamic model acquisition is used for perception for each underlying packing strategy, we have observed a similar trend of performance increase from V1 to V4. As a matter of fact, with the dynamically acquired model, we achieved 100% packing success in V4. We note that the 15 items from the YCB object dataset are mostly convex, and hence their shapes can be estimated well with our model acquisition pipeline.

VII. Conclusion

In this paper, we present a pipeline for automated packing in a warehouse settings that can pack geometrically complex, non-convex objects with stability while satisfying robot constraints. A new Heightmap-Minimization heuristic is proposed as a positioning heuristic for efficient 3D irregular shape packing. Simulation results on exhaustive datasets demonstrate the effectiveness of the pipeline and the advantage of the new heuristic in finding stable and robot-packable plans. Robot-packable plans are shown to be far more successful in open-loop execution than non-overlap methods used in prior work. Moreover, the robustness of the planner on a physical robot was investigated in an integrated system, and robust planning and closed-loop vision and force feedback are shown to greatly improve the success rates of executed plans.

Our work has a few limitations that can be addressed in future research. If object geometry, masses, or surface friction are poorly modeled, the placed pile is still likely to be disturbed and collapse. A more conservative planner, such as using a robust stability checker [42], could improve the likelihood of plan stability. It is still a challenge to ensure robustness with uncertainty in centers of mass and surface friction. We are also interested in improving stability tests during replanning by identifying the object pile's shifted configuration rather than treating it as a fixed, infinite-mass object. Finally, we are very interested in the use of manipulation to re-position items after they have been placed, such as compressing items toward a corner to make more space, or to repack items that have fallen.

Our current method is also mostly based on rigid objects with diffuse appearance. In some scenarios, a robot needs to handle translucent or transparent packaging and reflective surfaces, which poses challenges for perception. Deformable objects may be encountered in apparel packing and soft packaging, as well as when using dunnage (e.g., bubble wrap or paper) prevent damage to the objects during shipment. Our system can handle slightly deformable objects, but significant deformation requires further research, e.g., to exploit deformation during planning to achieve tighter packing, and to predict how the object changes shape during manipulation.

Fan Wang is a Ph.D. student at Duke University in the Department of Electrical and Computer Engineering. She received her M.Sc. from the University of Delaware and B.Sc. (Hons.) from the University of Edinburgh, UK. Her research interests include robotics, planning, manipulation, machine learning, computer vision, and sensor fusion for perception-based control of autonomous systems.

Kris Hauser is Associate Professor at University of Illinois at Urbana-Champaign in the Department of Computer Science and the Department of Electrical and Computer Engineering. He received his PhD in Computer Science from Stanford University in 2008, bachelor's degrees in Computer Science and Mathematics from UC Berkeley in 2003, and was a postdoc at UC Berkeley in 2008. He has also held faculty positions at Indiana University from 2009–2014 and Duke University from 2014–2019. He is a recipient of a Stanford Graduate Fellowship, Siebel Scholar Fellowship, Best Paper Award at IEEE Humanoids 2015, and an NSF CAREER award.

REFERENCES

[1] K. Yu, N. Fazeli, N. C. Dafle, O. Taylor, E. Donlon, G. D. Lankenau, and A. Rodriguez, "A summary of team mit's approach to the amazon picking challenge 2015," *CoRR*, vol. abs/1604.03639, 2016. [Online]. Available: http://arxiv.org/abs/1604.03639

[2] M. Schwarz, C. Lenz, G. Martin Garcia, S.-Y. Koo, A. S. Periyasamy, M. Schreiber, and S. Behnke, "Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing," 05 2018.

[3] A. Zeng, S. Song, K. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morena, P. Qu Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3750–3757.

[4] K. Yu and A. Rodriguez, "Realtime state estimation with tactile and visual sensing for inserting a suction-held object," *CoRR*, vol. abs/1803.08014, 2018. [Online]. Available: http://arxiv.org/abs/1803.08014

[5] R. Shome, W. N. Tang, C. Song, C. Mitash, H. Kourtev, J. Yu, A. Boularias, and K. E. Bekris, "Towards robust product packing with a minimalistic end-effector," *CoRR*, vol. abs/1903.00984, 2019. [Online]. Available: http://arxiv.org/abs/1903.00984

[6] F. Wang and K. Hauser, "Stable bin packing of non-convex 3d objects with a robot manipulator," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8698–8704.

[7] S. Dong and A. Rodríguez, "Tactile-based insertion for dense box-packing," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7953–7960, 2019.

[8] S. Martello and D. Vigo, "Exact solution of the two-dimensional finite bin packing problem," *Management Science*, vol. 44, no. 3, pp. 388–399, 1998.

[9] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations Research*, vol. 48, no. 2, pp. 256–267, 2000.

[10] E. den Boef, J. Korst, S. Martello, D. Pisinger, and D. Vigo, "Erratum to "the three-dimensional bin packing problem": Robot-packable and orthogonal variants of packing problems," *Operations Research*, vol. 53, no. 4, pp. 735–736, 2005.

[11] T. G. Crainic, G. Perboli, and R. Tadei, "Extreme point-based heuristics for three-dimensional bin packing," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 368–384, 2008.

[12] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," 1979.

[13] B. Baker, E. Coffman, Jr., and R. Rivest, "Orthogonal packings in two dimensions," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 846–855, 1980.

[14] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 3, no. 4, pp. 299–325, 1974.

[15] T. Kämpke, "Simulated annealing: Use of a new tool in bin packing," *Annals of Operations Research*, vol. 16, no. 1, pp. 327–332, Dec 1988.

[16] D. Zhang and W. Huang, "A simulated annealing algorithm for the circles packing problem," in *Computational Science - ICCS 2004*, M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 206–214.

[17] X. Liu, J.-m. Liu, A.-x. Cao, and Z.-l. Yao, "Hape3d—a new constructive algorithm for the 3d irregular packing problem," *Frontiers of Information Technology & Electronic Engineering*, vol. 16, no. 5, pp. 380–390, May 2015.

[18] O. Faroe, D. Pisinger, and M. Zachariasen, "Guided local search for the three-dimensional bin-packing problem," *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 267–283, 2003.

[19] J. Egeblad, "Placement of two- and three-dimensional irregular shapes for inertia moment and balance," *International Transactions in Operational Research*, vol. 16, pp. 789 – 807, 06 2009.

[20] C. Voudouris and E. P. K. Tsang, *Guided Local Search*. Boston, MA: Springer US, 2003, pp. 185–218.

[21] J. L. Viegas, S. M. Vieira, E. M. P. Henriques, and J. M. C. Sousa, "A tabu search algorithm for the 3d bin packing problem in the steel industry," in *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control*, A. P. Moreira, A. Matos, and G. Veiga, Eds. Cham: Springer International Publishing, 2015, pp. 355–364.

[22] J. A. Bennell, L. S. Lee, and C. N. Potts, "A genetic algorithm for two-dimensional bin packing with due dates," *International Journal of Production Economics*, vol. 145, no. 2, pp. 547 – 560, 2013.

[23] L. Wang, S. Guo, S. Chen, W. Zhu, and A. Lim, "Two natural heuristics for 3d packing with practical loading constraints," in *PRICAI 2010: Trends in Artificial Intelligence*, B.-T. Zhang and M. A. Orgun, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 256–267.

[24] K. Y. Goldberg, B. Mirtich, Y. Zhuang, J. Craig, B. Carlisle, and J. F. Canny, "Part pose statistics: estimators and experiments," *IEEE Trans. Robotics and Automation*, vol. 15, pp. 849–857, 1999.

[25] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[26] A. Lodi, S. Martello, and D. Vigo, "Tspack: A unified tabu search code for multi-dimensional bin packing problems," *Annals of Operations Research*, vol. 131, no. 1, pp. 203–213, Oct 2004. [Online]. Available: https://doi.org/10.1023/B:ANOR.0000039519.03572.08

[27] ——, "Heuristic algorithms for the three-dimensional bin packing problem," *European Journal of Operational Research*, vol. 141, no. 2, pp. 410–420, 2002. [Online]. Available: https://EconPapers.repec.org/RePEc:eee:ejores:v:141:y:2002:i:2:p:410-420

[28] R. Adams and L. Bischof, "Seeded region growing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 6, p. 641–647, June 1994. [Online]. Available: https://doi.org/10.1109/34.295913

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[30] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[32] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," *Efficient Variants of the ICP Algorithm*, pp. 145–152, 02 2001.

[33] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[35] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[36] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5868–5877.

[37] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[38] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, Apr. 2017.

[39] Rutgers APC RGB-D Dataset, 2015, (Accessed Jan 28, 2019). [Online]. Available: http://pracsyslab.org/rutgers_apc_rgbd_dataset

[40] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *CoRR*, vol. abs/1711.00199, 2017. [Online]. Available: http://arxiv.org/abs/1711.00199

[41] J. Egeblad, B. K. Nielsen, and A. Odgaard, "Fast neighborhood search for two- and three-dimensional nesting problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1249 – 1266, 2007. [Online]. Available: https://doi.org/10.1016/j.ejor.2005.11.063

[42] Y. Or and E. Rimon, "Computation and graphical characterization of robust multiple-contact postures in two-dimensional gravitational environments," *The International Journal of Robotics Research*, vol. 25, no. 11, pp. 1071–1086, 2006.