Towards Robust Product Packing with a Minimalistic End-Effector

Rahul Shome*, Wei N. Tang*, Changkyu Song, Chaitanya Mitash, Hristiyan Kourtev,
Jingjin Yu, Abdeslam Boularias, and Kostas E. Bekris

Abstract— Advances in sensor technologies, object detection algorithms, planning frameworks and hardware designs have motivated the deployment of robots in warehouse automation. A variety of such applications, like order fulfillment or packing tasks, require picking objects from unstructured piles and carefully arranging them in bins or containers. Desirable solutions need to be low-cost, easily deployable and controllable, making minimalistic hardware choices desirable. The challenge in designing an effective solution to this problem relates to appropriately integrating multiple components, so as to achieve a robust pipeline that minimizes failure conditions. The current work proposes a complete pipeline for solving such packing tasks, given access only to RGB-D data and a single robot arm with a vacuum-based end-effector, which is also used as a pushing finger. To achieve the desired level of robustness, three key manipulation primitives are identified, which take advantage of the environment and simple operations to successfully pack multiple cubic objects. The overall approach is demonstrated to be robust to execution and perception errors. The impact of each manipulation primitive is evaluated by considering different versions of the proposed pipeline, which incrementally introduce reasoning about object poses and corrective manipulation actions.

I. Introduction

The past decade has witnessed a vast growth of intelligent robotic solutions for logistics and warehouse automation tasks, with the *Amazon Kiva* mobile robotic fulfillment system serving as a prime example [1]. Nevertheless, the completion of many tasks still rely on the use of repetitive human labor, such as for picking and packing of products and building customer orders. In particular, tightly packing products that are picked from an unstructured pile, the focal task of this work, still remains primarily manual, even though it is integral to warehouse automation and manufacturing.

Packing objects to fit in confined spaces, such as a small shipping box as the one in Fig. 1, is highly challenging. It can be argued that is more difficult than clearing clutter since packing requires placing objects in close vicinity to each other, in an ordered manner and also to be well aligned with the boundary of the container. This demands high levels of accuracy from the perception component as well as the manipulation strategy. Indeed, surprisingly little prior work seems to exist that explicitly addresses this problem, let alone using a simple, suction-based end-effector.

Fig. 1. The product packing problem for cuboid products: initial configuration (left), and achieved goal configuration (right).

To help narrow the application gap and enable the reliable, fully autonomous execution of such tasks, this system paper:

A. Proposes a complete hardware stack and an accompanying software pipeline for developing and testing algorithmic solutions for robot-enabled packing tasks. The hardware setup involves a single robotic arm as shown in Fig. 2, which depends only on depth-imaging technology to sense its environment. The result is a fully autonomous integrated system for picking objects from unstructured piles and placing them to satisfy a desirable packing arrangement, such as the one shown in Fig. 1.

B. Explores the use of a suction-based end-effector, which is also treated as a pushing finger, for product packing. The placement of objects, given the packing objective, requires the vacuum-based end-effector to pick objects from specific orientations, which may not be immediately accessible. Nevertheless, the paper demonstrates that the end-effector can still address such challenges in a reliable manner.

C. Develops and evaluates corrective prehensile and non-prehensile manipulation primitives that increase the pipeline's robustness despite uncertainty regarding the pose of the objects. The uncertainty arises from the end-effector's minimalistic nature and the use of only visual input. A critical aspect of the primitives is the intentional use of collisions, which exploit the inherent compliance of objects, the bins, and the end-effector for enhancing accuracy. Furthermore, the proposed primitives are tightly integrated with sensing to achieve in real-time:

 i) toppling of objects in the initial unstructured bin to expose a desirable surface of the object for picking;

 ii) pulling an object towards its target placement while pushing neighboring objects to further pack by operating directly over point cloud data; and

 iii) real-time monitoring of potential failures and corrective pushing to achieve tight packing.

The evaluation uses the platform of Fig. 2. The experiments execute the pipeline in real-world scenes and show that the proposed manipulation primitives provide robustness despite the minimalism of the setup.
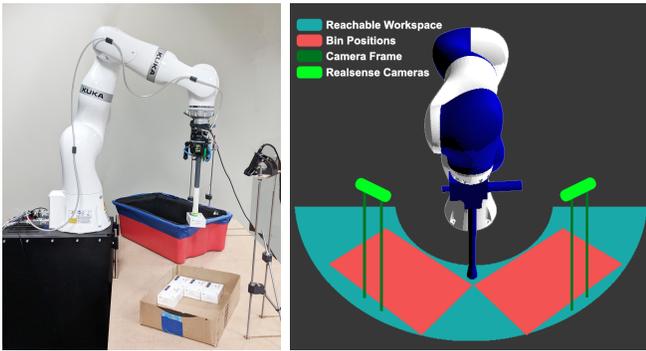
Fig. 2. Experiments are performed using a *Kuka LBR iiwa* arm equipped with a suction-based end-effector and depth-sensing cameras *SR300*. Two bins are within the reachability region of the arm given overhand picks.

II. RELATED WORK

Picking Objects in Clutter Most efforts in robot picking relate to grasping with form- or force-closure using fingered hands [2]. While early works focused on standalone objects [3], recent efforts are oriented toward objects in clutter [4]–[6]. Either analytical or empirical [4], grasping techniques typically identify points on object surfaces and hand configurations that allow grasping. Analytical methods rely on mechanical and geometric object models to identify stable grasps. Empirical techniques rely on examples of successful or failed grasps to train a classifier to predict the success probabilities of grasps on novel objects [7]. Analytical methods can be applied in many setups but are prone to modeling errors. Empirical methods are model-free and efficient [8] but require a large number of data. The heuristic picking strategy presented here inherits the properties of analytical methods while reducing computational burden.

Bin Packing The 3D bin packing problem, which is NP-hard [9], [10], requires objects of different or similar volumes to be packed into a cubical bin. Most strategies search for ε-optimal solutions via greedy algorithms [11]. While bin packing is studied extensively, to the best of the authors' knowledge there are few attempts to deploy bin packing solutions on real robots, where inaccuracies in vision and control are taken into account. Such inaccuracies have been considered in the context of efforts relating to the Amazon Robotics Challenge [12] [13] [14] [15] [16] [17] but most of these systems do not deal with bin packing. Most deployments of automatic packing use mechanical components, such as conveyor trays, that are specifically designed for certain products [18], rendering them difficult to customize and deploy. Industrial packing systems also assume that the objects are already mechanically sorted before packing. While this work makes the assumption that the objects have the same dimensions, the setup is more challenging as the objects are randomly thrown in the initial bin.

Non-prehensile Manipulation Non-prehensile manipulation, such as pushing, has been shown to help grasping objects in clutter [19], [20]. In these works, pushing is used to reduce the uncertainty of a target object's pose. Through pushing, target objects move into graspable regions. This work follows the same principle and relies on pushing actions

to counter the effects of inaccurate localization and point cloud registration. The problem is different because pushing is used for placing instead of grasping objects. The proposed method also uses pushing actions for toppling picked objects to change their orientations as well as to re-arrange misaligned objects. Other efforts have also considered pushing as a primitive in rearrangement tasks [21], [22]. The proposed system takes advantage of the compliance properties of the end-effector and leverages collisions with the environment to accurately place objects or topple them. A closely related approach [23] performs within-hand manipulation of a grasped object by pushing it against its environment.

6D Pose Estimation Recent efforts in 6D pose estimation use deep learning. In particular, regression over quaternions and centers can predict the rotations and centers of objects in images [24]. An alternative first predicts 3D object coordinates, followed by a RANSAC scheme to predict the object's pose [25]. Similarly, geometric consistency has been used to refine the predictions from learned models [26]. Other approaches use object segmentation to guide a global search process for estimating 6D poses [27]–[29]. This work is based on the authors' prior effort [27], where uncertainty over pixel labels returned by a convolutional neural network is taken into account when registering 3D models into the point cloud. The approach used here is different from the previous work as it can handle multiple instances of the same category of objects.

III. PROBLEM SETUP AND NOTATION

Consider a robotic arm in a workspace $\mathcal{W}$ with known static obstacles, two bins $\mathcal{B}_{init}$ and $\mathcal{B}_{goal}$, as well as n movable, uniform objects $\mathcal{O} = \{o^1, \ldots, o^n\}$ of known cubic dimensions. The bins $\mathcal{B}_{init}$ and $\mathcal{B}_{goal}$ are static but also compliant bodies in known poses that define a cuboid volume in $\mathcal{W}$ where the objects can be placed.

A labeled arrangement $A = \{p^1, \ldots, p^n\}$ is an assignment of poses $p^i \in SE(3)$ to each object o^i. Initially, the objects are in a start arrangement A_{init}, where the objects are inside $\mathcal{B}_{init}$ in random but "stable" poses, i.e., the objects are stably resting and not moving. The A_{init} arrangement is *not* known a-priori to the robot.

The objective is to move $\mathcal{O}$ to an unlabeled arrangement $\hat{A}_{\text{goal}} = \{\hat{p}^1, \ldots, \hat{p}^n\}$, which achieves a tight packing in $\mathcal{B}_{goal}$. $\hat{A}_{\text{goal}}$ depends on the pose of $\mathcal{B}_{goal}$, its dimensions and the object dimensions. The target arrangement and the picking order is input for the proposed process. The target arrangement maximizes the number of objects inside $\mathcal{B}_{goal}$, while the objects rest on a stable face, and minimizes the convex hull of the volume the objects $\mathcal{O}$ occupy in $\mathcal{W}$. This is typically a grid-like regular packing for cuboid objects. The unlabeled nature of $\hat{A}_{\text{goal}}$ means it is satisfied as long as one of the objects is placed at each target pose, i.e.,

$$\forall \hat{p}^j \in \hat{A}_{\text{goal}} : \exists\, o^i \in \mathcal{O} \text{ so that } D(\hat{p}^j, p^i) < \epsilon, \quad (1)$$

where ϵ is a threshold for achieving the target poses; $D(\cdot, \cdot)$ is distance between object poses, which should consider the 3-axis symmetry of the cubic objects. In particular, if two
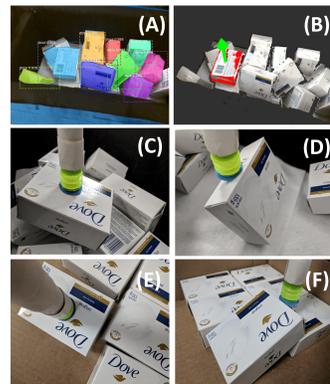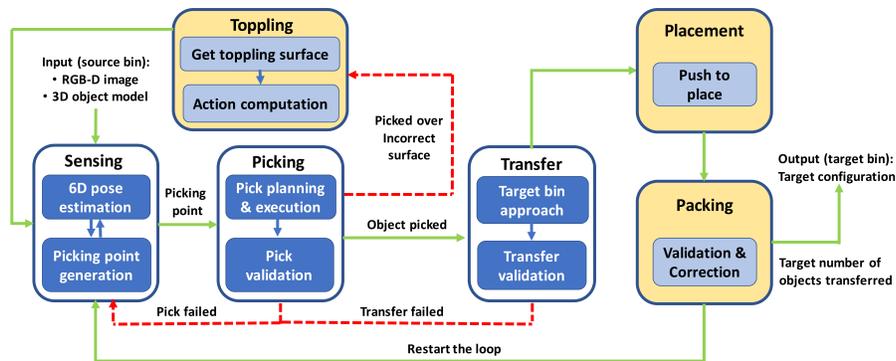
Fig. 3. *Left*: Pipeline in terms of control, data flow (green lines) and failure handling (red lines). The blocks identify the modules of the system. Sensing receives an RGBD image of $\mathcal{B}_{init}$ and object CAD models to return a grasp point. Based on the picking surface, the object is either transferred to $\mathcal{B}_{goal}$ or is handled by the `Toppling` module, which flips the object and places it back in $\mathcal{B}_{init}$. When the object is transferred, a robust `Placement` module places the object at the target pose $\hat{p}^i$. The Packing module validates and corrects the placement to achieve tight packing. *Right*: a) Instance segmentation. b) Pose estimation and picking point selection are provided by sensing, c) Picking d) Toppling e) Placement and f) Packing.

poses result into an object occupying the same volume, then their distance is 0. For instance, rotation by π about the vertical axis for a stably resting cube on a flat surface results in distance 0. A popular distance metric for 6D poses is the ADI metric [30]. The evaluation section will describe the distance used in the experimental process for evaluating the error of the final arrangement given point cloud data.

The arm has d joints that define the arm's configuration space $\mathbb{C}_{full} \subset \mathbb{R}^d$, which has a collision-free subset $\mathbb{C}_{free}$. Valid arm motions correspond to a continuous C-space curve $\pi : [0, 1] \rightarrow \mathbb{C}_{free}$. The arm has an end-effector, such as a suction cup, for which discrete operations $\{\texttt{pick}, \texttt{release}\}$ give rise to discrete modes: $\mathcal{M} = \{\texttt{transfer}, \texttt{transit}\}$. No within hand manipulation operations are available. The state space of the arm is: $\mathcal{X} = \mathbb{C}_{free} \times \mathcal{M}$. Sensing is used to reason about the current object poses. Overall, the robot operations involve (i) moving the joints, (ii) picking or releasing objects and (iii) sensing.

The arm's forward kinematics define a mapping $FK : \mathbb{C}_{full} \rightarrow SE(3)$, which provides the pose $g \in SE(3)$ of the end-effector given $q \in \mathbb{C}_{full}$. The reachable task space defines the end-effector poses the arm can reach without collisions: $\mathcal{T} = \{\forall \ q \in \mathbb{C}_{free} : FK(q) \in SE(3)\}$. For the arm to pick o^i at p^i, it has to be that the arm's end-effector pose g satisfies a binary output function: $\texttt{is\_pick\_feasible}(o^i, p^i, g)$, where $g \in \mathcal{T}$. For instance, the pose g of a suction cup must align it with at least one of the surfaces of an object o^i at p^i. Then, it is possible to define the set of end-effector poses, which allow to pick an object at a specific pose:

$$\mathcal{G}(o^i, p^i) = \{g \in \mathcal{T} : \texttt{is\_pick\_feasible}(o^i, p^i, g) = true\}.$$

Assume the sets $\mathcal{G}(o_i, p_i)$ are non-empty for all objects $\mathcal{O}$ and poses in A_{init} or $\hat{A}_{\text{goal}}$ (and their vicinity inside bins $\mathcal{B}_{init}$ and $\mathcal{B}_{goal}$). Otherwise, the task is not feasible. Note that it may be necessary to reconfigure the objects inside the initial bin so as to pick them from the appropriate face before placing them. This is due to the lack of within-hand manipulation.

Given the above definitions, the task is to identify a sequence of motions for the arm as well as end-effector and sensing operations to transfer the objects $\mathcal{O}$ from the unknown initial stable arrangement A_{init} inside $\mathcal{B}_{init}$ to a tight, grid-based packing inside $\mathcal{B}_{goal}$ defined by an unlabeled arrangement $\hat{A}_{\text{goal}}$, so as to satisfy Eq. 1.

IV. SYSTEM COMPONENTS

This section describes critical components that influence the design of the proposed pipeline:

a) Hardware Setup: The robot used in the setup is the `Kuka IIWA14` 7-DoF arm (Fig. 2). A customized end-effector solution extrudes a cylindrical end-effector that ends with a compliant suction cup, to engage vacuum grasps. Two *RealSense* SR300 cameras are mounted on a frame and pointed to containers $\mathcal{B}_{init}$ and $\mathcal{B}_{goal}$ from the other side relative to the robot as Fig. 2 shows. While far from the robot, the cameras' frame is statically attached to the robot's base such that calibration errors are minimized in estimating the camera poses in the robot's coordinate system.

b) Workspace Design: Fig. 2 shows the setup designed for the target task. The annular blue region represents the subset of the reachable workspace that allowed for top-down picks with the robot's end-effector. This region is computed by extensively calling an inverse kinematics (IK) solver for top-down picks with the end-effector. The IK solutions indicate that the radial region between $40cm$ and $70cm$ from the robot center maximizes reachability and IK solution success given the setup. The bins (red rectangles) are placed so that they lie inside the optimal reachable region.

c) Software: `MoveIt!` [31] is used for motion planning. Most of the motions are performed using *Cartesian Control*, which guides the arm using end-effector waypoints. Ensuring the motions occur in reachable parts of the space increases the success of *Cartesian Control*, simplifies motion planning and speeds-up execution. To decrease planning time, motion between the bins is precomputed using the RRT* algorithm [32] and simply replayed at appropriate times.

V. PROPOSED SOLUTION

The proposed pipeline, described in Fig. 3, provides the steps undertaken to perform the desired packing. The baseline steps correspond to:

a) *(Sensing)*: sense and select a target object o^i to pick up;

b) *(Picking)*: execute action {pick};

c) *(Transfer)*: move o^i to the next available target pose $\hat{p}^j$ and execute action {release}.

The experimental section considers this baseline pipeline, where it is observed that it performs poorly due to multiple sources of uncertainty, ranging from pose estimation and calibration errors, to object non-uniformity and object interaction with the bin and other objects, among others. These issue necessitate the introduction of remedies, which actively reduce the impact of the uncertainty. To this end, 3 manipulation primitives for a simplistic end-effector are designed to increase robustness and are integrated with the overall architecture:

i) Toppling;

ii) Robust Placement; and

iii) Corrective Packing.

A. Baseline: Pose Estimation and Picking

Given an RGB-D image of the source bin and a CAD model of the object, the objective is to retrieve (o^i, p^i) such that it maximizes the chance of achieving target configuration $\hat{p}^j$, where $D(p^i, \hat{p}^j) \leq \epsilon$. To achieve this, the image is passed through a MaskRCNN convolutional neural network [33], which is trained to perform segmentation and retrieve the set of object instances $\mathcal{O}$. An image segment is ignored if it has a number of pixels below a threshold. It is also ignored, if MaskRCNN has small confidence that the segment corresponds to the target object. Among the remaining segments, instances $o^i \in \mathcal{O}$ are arranged in a descending order of the mean global Z-coordinate of all the RGBD pixels in the corresponding segment. Then, 6D pose estimation is performed for the selected instance [34] [35].

If, given the detected 6D pose of the instance, the top-facing surface does not allow the placement of the object via a top-down pick, the next segment instance is evaluated in order of the mean global Z-coordinate. If no object reveals a top-facing surface, then the first object in terms of the maximum mean global Z-coordinate is chosen for picking.

For the selected object, a picking point, i.e., a point where the suction cup will be attached to the object, is computed over the set of points registered against the object model. It utilizes a picking-score associated in a pre-processing step with each model point, which indicates the stability of the pick on the object's surface. The score calculates the distance to the center of the object mesh. A continuous neighborhood of planar pickable points is required to make proper contact between the suction cup and the object surface. Thus, a local search is performed around the best pick-score point to maximize the pickable surface.
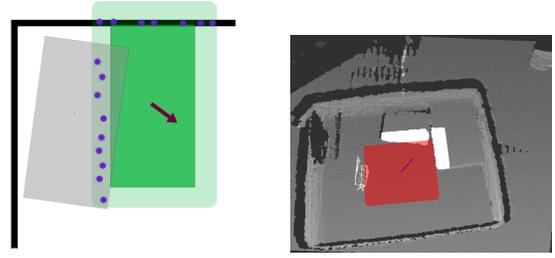

Fig. 4. Adaptive pushing: (left) The black border is the bin and the gray rectangle a previously placed object. The green rectangle represents the target pose for the current object. The light green boundary represents an ε-expanded model that intersects the point cloud at the purple points. These points result in the black vector that pushes the object away from them. (right) A screen shot of a scene's point cloud, where the white points are collision points with previously placed objects and the red volume shows the computed pre-push pose for the new object.

B. Toppling

The toppling primitive is invoked if there exists no object that exposes the desirable top-facing surface, or if the object was erroneously picked from the wrong face. The latter is detected after the pick by performing pose estimation once the objet is attached to the suction cup. For instance this can happen for the soaps shown in Figure 3 (right)(d), if the thin side is available for pick but the soap needs to be placed on their wider side. In this case, a toppling primitive is used to reorient the object.

Given a starting pose of an object p_{start} and a toppling action of the arm, the object ends up at a new pose p_{topple}. The objective is to allow the existence of a pick $\hat{g} \in \mathcal{G}(o^i, p_{topple})$ so that there is a transfer action from pick $\hat{g}$ that achieves the final placement p_{end} close enough to the desired target placement $\hat{p}^j$, i.e., $D(p_{end}, \hat{p}^j) \leq \epsilon$. For the considered setup, this means that the top-facing surface of the object at p_{topple} and $\hat{p}^j$ is the same.

The toppling module inspects the visible point cloud in the source bin to identify the best toppling plane, which is sufficiently empty and flat. The accompanying implementation restricts actions to ones that change the pose of the cubic objects by shifting the most upward facing surface to only one of the adjacent surfaces. While this does not guarantee toppling the object to all possible poses, the symmetry of cubic products resolves this issue.

Prior work [36] has shown the efficacy of minimal end-effectors used in tandem with the environment to achieve toppling. In the previous work, the friction against a conveyor belt is used to topple an object about a resting surface. The conveyor belt's motion is parallel to the initial resting surface plane. In the current setup, the compliance of the suction cup is used to emulate the same effect using a lateral motion on the same plane as the top-surface along the direction of the desired transformation between p_{start} and p_{topple}. Due to symmetry, at least one neighboring surface allows top-down picks, so a successful toppling action exists. Using the pose of the object, the lateral motion direction is executed once the object is placed on the detected plane, and the object is released during this action. Results show that this is highly effective in the target setup.

C. Point Cloud Driven Adaptive Pushing

Directly placing objects at the goal pose $\hat{p}^i$ into bin $\mathcal{B}_{goal}$ is prone to placement failures due to errors in perception as well as prior placements. This may result in damaging the objects. A safer alternative is to drop the object from a certain height, right above the goal pose, so as to avoid pressing against previously erroneously placed objects. Still, however, this alternative results in low quality packing. A key realization is that during placement, the object being manipulated will inevitably approach or even collide with other objects or the target container. To sidestep undesirable collisions, an *adaptive pushing* primitive is developed, which directly operates over point cloud data for the target bin.

The process is shown in Fig. 4. The adaptive pushing begins by growing the object model at the target pose $\hat{p}^i$. Given the uncertainty value ε, the model is enlarged by 2ε along each dimension. The enlarged model is intersected with the point cloud to retrieve collision points. A collision vector is computed as a vector pointing from a collision point to the center of the object model. Summing all collision vectors yields the displacement vector. By iteratively moving the object model along the unit displacement vector, where the displacement vector is recomputed after each movement, a collision-free pre-push pose for the object is obtained. During the execution, the robot first moves the object above the pre-push pose, then lowers the object to the pre-push pose and finally pushes the object to the target pose.

D. Fine Correction using Push and Pull Primitives

The final primitive deals with the remaining failure cases. Fine corrections are required because objects can be placed in incorrect poses due to unexpected collisions as well as calibration and pose estimation errors. The proposed corrective manipulation procedure continuously monitors the scene and triggers corrective actions whenever necessary.

The process first removes the background, the box, the robot's arm and end-effector from the observed point cloud and computes its surface normals. The observed point cloud is then compared against the desired alignment

Fig. 5.   Fine layout adjustment and correction.

of the objects in their target poses. As shown in Fig. 5, two types of misalignment errors can occur. The first type occurs when the top surface normal of an object is not perpendicular to the support surface. This error is corrected by pushing the object along a direction and for a distance computed based on its surface normal in a manner that makes it aligned with the support surface. The second type of error happens when a peripheral object is not entirely within the desired footprint of the pile. The proposed procedure systematically detects pivot points that are outside the desired footprint of the pile and pulls their objects inside accordingly. The correction is repeated until the point cloud is aligned with the desired goal poses given a threshold ϵ, or a timeout occurs.

VI. EVALUATION

To evaluate the performance of the proposed pipeline, extensive experiments were executed on the Kuka platform of Fig. 2, to accurately represent the motivating complications that arise in real-world setups. The experiments are designed to showcase the hardness of tight packing as well as the benefits of adding robust environment-aware manipulation primitives that aid in increasing success rate and accuracy.

For consistency, an identical version of the problem is tested, with "dove soap bars" that are randomly thrown into the source bin $\mathcal{B}_{init}$, which is placed on one side of the robot's reachable workspace (Fig 2). Only top-down grasps are allowed within a given alignment threshold. The start arrangement A_{init} of objects is intended to reflect a random pile, with 10 repetitions of each experimental condition. The target bin $\mathcal{B}_{goal}$ contains a 3×3 grid arrangement of 9 objects, on the same plane, with the stable face of the object targeted for placement. The complete pipeline uses a) corrective actions for fine adjustments, b) push-to-place actions for robust placement, c) toppling actions for increasing successes, and d) pose estimation for adjusting the object. The improvements introduced by these strategies are evaluated through the following comparison points, within the context of the proposed pipeline:

V1 - Full pipeline: The complete pipeline with all the primitives achieves the highest accuracy and success rate.

V2 - No corrective actions: The experiment corresponds to the use of **V1** without the fine correction module of Fig. 5.

V3 - No push-to-place actions: This version is **V2** without the use of the robust placement module (Fig. 4) that performs push actions to achieve robust placement.

V4 - No toppling actions: These experiments used **V2** without considering toppling actions to deal with objects not exposing a valid top surface that allows the target placement.

V5 - (Baseline) No push-to-place, toppling, pose-estimation: The naive baseline that solely uses a pose-unaware grasping module that reports locally graspable points and drops the grasped object at an end-effector pose raised from the center of the desired object position, with no adjustment in orientation.

The metrics evaluated include the fraction of successful object transfers that succeed in moving objects to the target bin. The accuracy is captured in the threshold mentioned in Eq. (1) that is expressed in terms of a percentage of unoccupied volume within the ideal target placement volume. This was measured with a voxel discretization sufficient to elucidate the difference between the methods. The average data recorded is reported in Fig. 6. This error measure is proportional to the accuracy. The points to note for every version are detailed as follows.

The low error for **V1** corroborates the final bin placement evidence. On average 7 corrective actions per experiment were invoked to achieve the high degree of accuracy.

The accuracy improvement obtained from corrective actions is evaluated in **V2**. While this version succeeded in dropping all the objects close to the correct target poses,
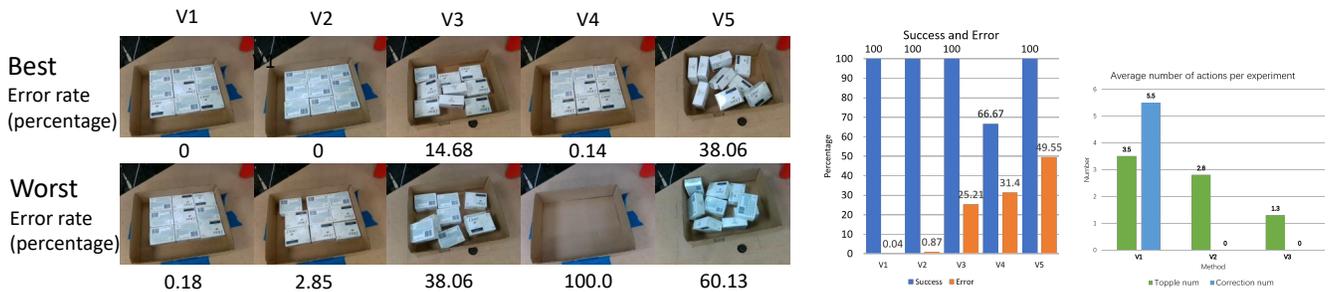
Fig. 6. (left)The final set of object poses in the target poses at the end of every experiment. Different column represents different versions. The top row is the best case, and the bottom row is the worst case. (middle) the blue bar represents the fraction of successful object transfers, the orange bar represents the percentage of unoccupied volume within the ideal target placement volume.(right)the blue bar represents the average number of correction actions happened per experiment, the green bar represents the average number of toppling actions happened per experiment.

application use-cases where a higher degree of accuracy is desired motivate the use of corrective actions. The integration of the corrective actions was done with higher error threshold during intermediate steps, and a much finer one for the final adjustment. Errors can typically arise from execution failure and pose misalignments. The less accurate these underlying processes are, the more important corrective actions become.

V3 only performs adjustments using pose estimation, and toppling. While, this is sufficient to successfully transfer all the objects, any difference of accuracy to **V2** would be introduced by the lack of push-to-place actions. Here there is complete reliance on the exactness of the execution and pose adjustments. Due to the proximity of adjacent object surfaces in the target grid arrangement, even minor errors get aggravated. However, due to the ability to reason about toppling, all the objects can be transferred to the target bin, even with this low accuracy. This is demonstrated in the occurrence of the failure to transfer all the objects.

In **V4** any object that does not expose an permitted picking surface that makes the prehensile placement possible, is not picked. Any instance of the source bin, which ends with no such objects results in no valid picking actions that can make the approach proceed, and a failure is declared. The current behavior of **V4** drops the object if it is mistakenly grasped from the wrong surface. This can itself be used as a naive toppling primitive. It is important to note that there might be other alternative strategies that can deal with this failure, but the intent of this comparison is to demonstrate the importance of having a deliberate toppling strategy in the pipeline, that can change the object's orientation in the context of random starting arrangements of the object. On average, over **V1, V2,** and **V3** the toppling primitive was required 4 times per experiment. This highlights the necessity of this reasoning. Deliberate toppling however requires at least one additional pick action. The number of pick attempts per successful object transfer was 2.56 for **V4**, whereas, in **V2** the same was 1.98. This indicates that toppling is indeed necessary both in terms of success and efficiency of actions.

Expectedly, **V5** has the lowest accuracy. However, since there is no reasoning about the pick surface, every object was transfered to the space of the bin. This has no guarantee to work if the object is larger. This drives the motivation for using a set of robust primitives for the packing problem.

Overall, the time for the experiments show a trend of increasing with the increasing complexity of the pipeline. The trade-off of accuracy versus time persists. On average, **V1** ran for $945s$ while **V5** ran for $323s$.

Multi-layer packing: With a sufficiently tall bin, objects can be packed in multiple layers. Running the method beyond the first plane of the grid effectively shifts all the operations to the higher plane. This is demonstrated in a standalone run.

Different objects: To validate the applicability of the method to other cuboidal objects, **V1** was performed for toothpastes. Over 5 experiments, with 4 objects, every run succeeded in placing the object inside the bin.

Fig. 7. Result for toothpaste and two-layer packing

VII. DISCUSSION

The proposed pipeline indicates intriguing nuances of the packing problem. The use of a minimal, suction-based end-effector is a cost-effective, simple and relatively robust way to pick objects but does not easily allow for complex grasp reasoning, regrasping, or within-hand manipulation. The proposed pipeline achieves high accuracy by leveraging the compliance of the suction cup and the environment, while the object is attached. It uses robust reasoning to incrementally correct errors, instead of compounding them. While it can be argued that better baseline components can be developed to minimize uncertainty, the overall philosophy of robust, minimal, and compliant reasoning remains unchanged.

The proposed system and primitives can also deal with cubic objects with different sizes and can be extended to non-cubic objects by adapting the object models. The key adaptation corresponds to identifying an appropriate packing arrangement $\hat{A}_{\text{goal}}$ (potentially labeled in this case) in the target bin and the corresponding picking order from the initial bin. Future work will also explore speeding up performance and dealing with algorithmic challenges: the combinatorial reasoning over possible placements, physics-based reasoning to further improve pushing and toppling, as well as extending to more adaptive end-effectors. The platform can also be utilized as a training testbed for reinforcement learning to automatically discover robust primitives for solving packing tasks.

REFERENCES

[1] J. J. Enright and P. R. Wurman, "Optimization and coordinated autonomy in mobile fulfillment systems," in *Proc. of the 9th AAAI Conference on Automated Action Planning for Autonomous Mobile Robots*, 2011, pp. 33–38.

[2] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, Mar. 2012.

[3] K. Shimoga, "Robot grasp synthesis algorithms: A survey," *The International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.

[4] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis: A survey," *Trans. Rob.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[5] A. Boularias, J. A. D. Bagnell, and A. T. Stentz, "Efficient optimization for autonomous robotic manipulation of natural objects," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, November 2014, pp. 2520–2526.

[6] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, pp. 1336–1342. [Online]. Available: http://dl.acm.org/citation.cfm?id=2887007.2887192

[7] D. Morrison, J. Leitner, and P. Corke, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[8] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 515–524.

[9] J. O. Berkey and P. Y. Wang, "Two-dimensional finite bin-packing algorithms," *Journal of the Operational Research Society*, vol. 38, no. 5, pp. 423–429, May 1987. [Online]. Available: https://doi.org/10.1057/jors.1987.70

[10] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Oper. Res.*, vol. 48, no. 2, pp. 256–267, Mar. 2000. [Online]. Available: http://dx.doi.org/10.1287/opre.48.2.256.12386

[11] S. Albers and M. Mitzenmacher, "Average-case analyses of first fit and random fit bin packing," in *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '98. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998, pp. 290–299. [Online]. Available: http://dl.acm.org/citation.cfm?id=314613.314718

[12] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, 2016.

[13] M. Schwarz, C. Lenz, G. M. García, S. Koo, A. S. Periyasamy, M. Schreiber, and S. Behnke, "Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3347–3354.

[14] D. Morrison, A. W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, *et al.*, "Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7757–7764.

[15] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, *et al.*, "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)*. IEEE, 2018, pp. 1–8.

[16] Z. Littlefield, S. Zhu, C. Kourtev, Z. Psarakis, R. Shome, A. Kimmel, A. Dobson, A. Ferreira De Souza, and K. E. Bekris, "Evaluating end-effector modalities for warehouse picking: A vacuum gripper vs a 3-finger underactuated hand," in *CASE*, 2016.

[17] C. Rennie, R. Shome, K. E. Bekris, and F. A. De Souza, "A dataset for improved rgbd-based object detection and poe estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters (RA-L)*, 2016.

[18] S. Hayashi, S. Yamamoto, S. Tsubota, Y. Ochiai, K. Kobayashi, J. Kamata, M. Kurita, H. Inazumi, and R. Peter, "Automation technologies for strawberry harvesting and packing operations in japan 1," *Journal of Berry Research*, vol. 4, no. 1, pp. 19–27, 2014.

[19] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[20] M. Dogar and S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, vol. 33, no. 3, pp. 217–236, Oct 2012.

[21] J. E. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2508–2515.

[22] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4627–4632.

[23] N. Chavan-Dafle and A. Rodriguez, "Prehensile pushing: In-hand manipulation with push-primitives," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 6215–6222.

[24] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[25] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[26] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother, "Global hypothesis generation for 6d object pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 462–471.

[27] C. Mitash, A. Boularias, and K. Bekris, "Robust 6d object pose estimation with stochastic congruent sets," *arXiv preprint arXiv:1805.06324*, 2018.

[28] V. Narayanan and M. Likhachev, "Discriminatively-guided deliberative perception for pose estimation of multiple 3d object instances." in *Robotics: Science and Systems*, 2016.

[29] C. Mitash, A. Boularias, and K. E. Bekris, "Improving 6d pose estimation of objects in clutter via physics-aware monte carlo tree search," *arXiv preprint arXiv:1710.08577*, 2017.

[30] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.

[31] S. Chitta, I. Sucan, and S. Cousins, "Moveit!" *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, 2012.

[32] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *IJRR*, vol. 30, no. 7, June 2011.

[33] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[34] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," in *IROS*, 2017.

[35] C. Mitash, A. Boularias, and K. E. Bekris, "Robust 6d object pose estimation with stochastic congruent sets," in *British Machine Vision Conference*, 2018.

[36] K. M. Lynch, "Toppling manipulation," in *Proceedings 1999 IEEE International Conference on Robotics and Automation*, vol. 4. IEEE, 1999, pp. 2551–2557.