An Autonomous Waste Sorting Robot

Hai Nguyen
University of Nevada, Reno
hainguyen@nevada.unr.edu

Abstract

This paper proposes a robotic grasping system for autonomously sorting waste items. The point cloud of multiple objects are transformed into the robot coordinate and path planning for grasping are performed. The proposed approach also handle grasping on certain objects defined by users. Convolutional neural network will perform the detection and select the object to be grasped. From the point cloud data, an algorithm will generate grasps and choose the best grasp to execute the robot. Experiments show the potentials of the proposed approach in the context of object grasping and waste sorting robots with robust and fast generated grasp.

1. Introduction

Waste sorting is one of the most dangerous occupations, and employees in this industry suffer a high risk of injuries and fatality. A study [1] by environmental, occupational safety, and community benefits experts reveal that recycling workers are two times more likely to suffer work-related injuries than other employees. For instance, between 2011 and 2013, seventeen workers in the US lost their lives while doing their jobs. Risky conditions around heavy machinery and close exposure to toxic material such as hazardous chemicals, hypodermic needles, and animal carcasses are some of the main reasons behind these unfavorable statistics. Recycling is the right thing to do but we would need a smart prevention strategy to protect human lives working in this industry.

A good strategy is to incorporate robots in the recycling process and let them handle all the dirty and dangerous parts of the job. The reasons why robots can be an excellent option for waste sorting can be numerous. Firstly, robots can easily handle hazardous material while processing waste. Secondly, waste sorting is also a tedious job which increases the possibility for a human to make an error, especially when working for long hours. A machine is ideal in this case as it can work tirelessly with little supervision. Finally, advancements in recent artificial technologies such as multi-layered neural networks, computer vision, and robot manipulation allow creating autonomous AI recycling robots possible with several existing waste sorting robots working in the industry.

Figure 1: Max-AI waste sorting robot.

One example is the Max-AI robot produced by Bulk Handling Systems as shown in Figure 1. Using cameras and deep learning algorithms trained to recognize specific objects, the robot can identify cans, plastic containers, glass, or any other kind of recyclable materials in a pile of trash within seconds, pick them up using suction cups or large tongs, and deposit them into the appropriate bin.

In this paper, we present our solution of turning an existing gripper in our lab into an autonomous waste sorting machine. The main idea is the integration of the latest technologies in object detection, grasp pose detection, and motion planning to create a viable solution. The proposed approach can also be applied to any manipulator and to any object that the manipulator can hold physically. Moreover, the work will also create a sound foundation so that it can be applied to other contexts relating to object grasping manipulation other than autonomous waste sorting robots.

2. Motivations

An autonomous waste sorting machine should have the ability to detect various types of wastes, and after that, it should be able to pick the detected items into the appro-

[1] Sustainable and Safe Recycling: Protecting Workers Who Protect the Planet

priate bins. Therefore, object grasping is an essential manipulation that these types of robots need to possess. This requires the robot to perform object grasping at different shapes, sizes, and positions in the scenes. There was multiple research going into autonomous garbage sorting robot with the different proposed approach for grasping. Most of them use multiple sensors such as RGB cameras and 3D sensors to achieve grasps on objects. In this paper, with a single RGB-D camera, we show that we can also achieve robust grasps on point clouds of objects. Moreover, using computer vision methods, we can direct the gripper to perform grasps specifically on a certain type of objects. To our knowledge, this has not been performed in previous research.

3. Literature Review

We divide the literature review into three smaller sections: grasp detection and filtering, manipulation, and existing sorting robots.

3.1. Grasp Detection and Filtering

Convolutional neural networks are frequently used to detect grasp positions. [5] uses two subsequent neural networks to detect robotic grasps from RGB-D images using a sliding window detection pipeline as shown in Figure 2. The top detection from the first network will be re-evaluated by the second. The strength of the paper is to use deep neural networks to replace heuristic features, which are traditionally cumbersome and time-consuming. Additionally, the usage of the two-step approach also reduces processing time while matching or even out-performing the performance of the single-state system. One problem of this approach is the exclusion of the properties of grippers as an input, therefore, even the detection algorithm might find a graspable pose but the gripper is unable to perform the grasp successfully.

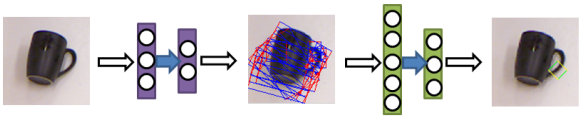

Figure 2: Illustration of two-state detection process.

[1] shows that using prior instance or category information about the object to be grasped could further improve the quality of detected grasp poses. They characterize the local geometry and/or appearance of graspable object surfaces using machine learning methodologies as illustrated in Figure 3. The convolution neural network is first trained on an online dataset comprised of RGB images of real scans and then fine-tuning continues with real data. The paper, however, fails to address the common difficulty faced by GDP algorithms when there is no direct way to distinguish between two adjacent objects as the algorithm avoids point cloud segmentation. As a result, there is also no direct way to grasp a specific object of interest.

Figure 3: Illustrations of grasp candidates found.

[7] uses models from geometry and mechanics to generate synthetic training datasets including synthetic point clouds, robot parallel-jaw grasps, and metrics of grasp robustness based on physics for thousands of 3D object models. Domain randomization is used to various objects, sensors, and physical parameters to bridge the gap between models and real-world systems. The biggest strength of this paper is the synthetic data which help save a lot of time and resources. Training on datasets collected by humans or physical experiments is more time-consuming and more prone to inaccuracies. Obviously, we have more control over the diversity and the accuracy of the simulation-generated datasets. In experiments with a real robot to perform grasps using RGB-D cameras, the proposed approach has two failure modes. The robot failed to attempt to grasp occluded objects and objects with adversarial geometry and adversarial material properties such as deformable or transparent items.

[10] also tries to find good grasps from an RGB view of the object. The algorithm has great speed when it is able to run at 13 FPS on a GPU. To achieve such a speed, the proposed network structure performs single-state regression to graspable bounding boxes instead of using sliding window or region-based techniques. Following this approach, even the network is comparatively larger but as it only runs once to an image, they get a significant boost in performance and speed.

3.2. Manipulation

After the detection of different types of waste is made, the robot will need to perform the pick-and-place operation. To accomplish this, a motion planner will generate a trajectory by performing optimizing over a set of trajectories that satisfy constraints. The planning starts with trajectories containing collisions and constraints violation. The planning should converge in a high-quality trajectory meeting

all constraints. In the context of waste sorting, the input to the motion planner will be the desired position and orientation of the best grasp poses. Outputs of the planner will be joint positions or torques making the gripper to move to pick and place those detected objects. Constraints might include no collisions with objects inside the working space or joint limits of the robot. There are several open-source implementation of various motion planning algorithms, including sampled-based planners from OMPL [3], CHOMP (Covariant Hamiltonian Optimization for Motion Planning) [9], and STOMP (Stochastic Trajectory Optimization for Motion Planning) [2]. In this paper, we use TrajOpt [11]. It is a motion planner with a similar approach to CHOMP's but with the advantage of faster computation time and the ability to solve a larger number of problems.

3.3. Existing Robots

In [4], a picking prototype robot by ZenRobotics Ltd is described targeting the industrial waste sorting application. The vision perception consists of a 3D camera, whose images are projected into an isometric heightmap defined on a 2D coordinate of a gantry-type robot. Handles (grasps) are defined as a data structure containing gripper (x, y, z) coordinate, gripper angle, and gripper opening. The algorithm searches exhaustively all closed handles which are gripper configurations where each finger of the gripper makes contact with the height map. Candidate handles are generated and the sample collection is expanded by duplicating for all possible openings allowed by the heightmap. After that, a random forest classifier will classify the handles into possible and impossible grasps. The system also monitors the gripper opening to feedback for autonomous training. The robot is able to actively learn to grasp better while working, however, the system needs to be paused periodically for data collection and learning. This is not desirable in the industrial context. Another disadvantage is that the system learns from only a small number of data samples each time, making the learning very slow.

ZenRobotics introduces another version of construction and demolition waste sorting robot - ZenRobotics Recycler robot in [6]. Multiple sensors such as 3D sensors, metal detector sensors, and RGB cameras are fused together to localize waste objects. Material classification is approached as a supervised learning problem and the grasping task is a reinforcement learning approach. To further increase the robustness, they also manually annotating mistakes done by the system to avoid next time.

Another prototype version for solid waste sorting is mentioned in [8] to work in landfill sites. An RGB camera and thermo-graphic camera are used, with the later gives more information about thermal properties which assists in detecting recyclable material. Training images first are used for feature extraction using SURF before being fed to a multi-class SVM classifier for material recognition. However, the grasp plan described is rather simple and only applies to simply shaped objects (soda cans and water bottles). The processing time will be another issue, as in a cluttered environment such as in landfill sites, SURF algorithms will cost much more time to recognize key-points.

4. The Proposed Method

The proposed approach included three stages:

- Selecting an object to grasp: In this stage, we are having multiple objects in the scenes and we need to decide which object we will need to grasp first. Therefore, we need an input of which type of waste that we want to grasp and if there are multiple objects of that type, we need a way to score them and grasp each item in a predefined order. The output of this step is the point cloud cluster of the selected object as we will generate grasps on it using point cloud data.

- Generating and filtering grasp on the selected object: Given that we are having the point cloud cluster of the object that we want to grasp, we need to generate grasps on the object and then select the best grasp to execute on the real robot. This requires a filter applied on candidate grasps to score them against each other. Only the highest-scored grasp will be chosen to execute.

- Executing the chosen grasp: We are now having a grasp pose to execute. The motion planner at this step will need to generate a trajectory that helps the robot's gripper to reach the desired grasp pose. More importantly, the trajectory needs to be collision-free to ensure the safety of the hardware. The motion planner, therefore, will need a model of the environment so that it can check for collisions when generating trajectories.

4.1. Object Selection

It is challenging when performing object detection directly using point cloud data as in point cloud data, some visual cues such as colors or shapes that we normally use to recognize an object is affected. To overcome this challenge, we combine color images and point cloud data to make the decision. The steps that are taken are illustrated in Figure 4. We start with RGB images of waste items and these images are fed to an image detector to put a bounding box around the selected object.

Currently, there are several existing neural networks that can help to localize objects from RGB images and put bounding boxes around them such as R-CNN, YOLOv3, and SSD. We test the performance of each network and decide to choose YOLOv3 as the final detection network.

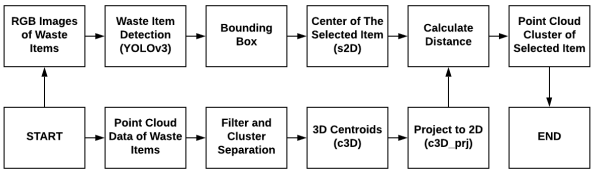

Figure 4: Flowchart of selecting an object to grasp.

Another feature of YOLO is that they also output the percentage of detection reliability. We use these percentages to score items of the same class in case that there are multiple items belonging to one class. By this mechanism, a single item is chosen at this step. The center of the bounding box $s2D$ (as shown in Figure 5) is calculated.

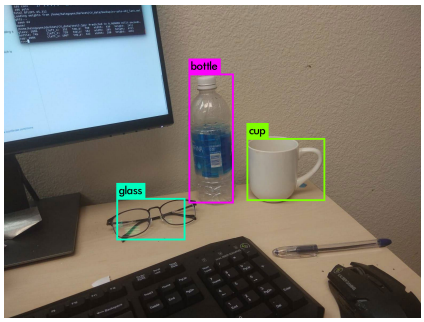

Figure 5: Objects detected by the YOLOv3 network.

On a parallel branch, point cloud data of waste items is first filtered to remove noise and outliers. To remove the outliers and further reduce the amount of point cloud data, we use a statistical filter, a voxel filter, and then a working space filter. A workspace filter will remove all the data points that do not belong to our predefined workspace. At this step, we use plane segmentation and target extraction to separate the items as shown in Figure 6.

Also shown in the figure, the 3D centroid coordinates of these objects $c3D$ are calculated. These centroids are then projected to get 2D points $c3Dp$ in the image plane coordinate (using the camera calibration parameters). At the last step, the Euclidean distance between $s2D$ and $c3Dp$ are calculated and compared to a threshold to determine whether the corresponding cluster belongs to the selected object:

$$\begin{cases} (s2D_x - c3Dp_x)^2 + (s2D_y - c3Dp_y)^2 < T & \text{Yes} \\ (s2D_x - c3Dp_x)^2 + (s2D_y - c3Dp_y)^2 > T & \text{No} \end{cases}$$

The process is illustrated in Figure 7 with the yellow squared being $c3Dp$, the white circle is $s2D$. In this picture, the calculated distance determine the cluster corresponding to the second object from the left belongs to the object to be grasped.

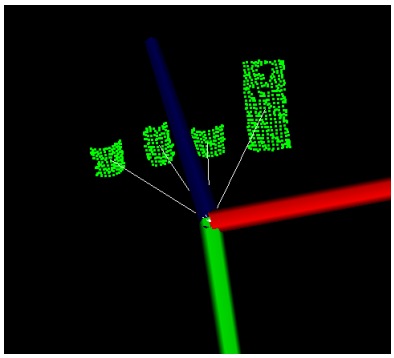

Figure 6: Cluster of objects are separated.

Figure 7: Determine the object to grasp.

4.2. Grasp Generation and Filtering

Figure 8: Grasp generation & filtering on the selected item.

The flowchart in this step is illustrated in Figure 8. After having the cluster of the object to be grasped, we run the algorithm Grasp Pose Detection [1] on it. Each point in the cloud is associated with a single viewpoint from which the point is captured. The algorithm also considers the geometric parameters of the robot gripper and a subset of points C_G belonging to objects to be grasped as inputs. To have the subset of points that belong to objects of interests C_G, we take those central points calculated earlier from the labeled bounding boxes and then sample spheres of points around them. Given providing the above-mentioned inputs, the algorithm will first sample points uniformly around C_G. After that, the algorithm calculates the surface normal and an axis of major principle curvature of the object surface. Potential grasp candidates are generated at regular orientations orthogonal to the curvature axis. These grasp poses are then pushed forward until the fingers make contact with the point cloud. Those grasps without any data points between the

fingers will be discarded. Next, remaining grasp poses will be classified as a grasp or not using a four-layered convolutional neural network. At this step, the algorithms will start scoring grasps against each other. The default algorithm gives high score for grasps that are at the top of the object because these grasp might have less effect on other objects. Additionally, we also give more priority to the grasp poses whose Euclidean distance to the current pose of the gripper is smaller. This is beneficial for the grasp execution later as the robot can go there quicker and easier than other poses.

4.3. Grasp Execution

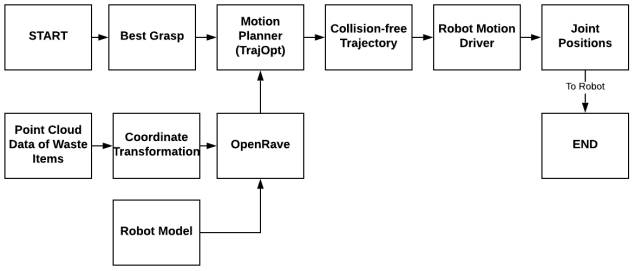

Figure 9: Flowchart of executing the chosen grasp.

To execute the chosen grasp on the real robot, we need to generate a collision-free trajectory that transforming the current gripper's pose to the desired grasp pose. To check for collisions, TrajOpt [11] needs an accurate simulated model of the environment. The environment includes the robot model and the point cloud data of objects in the robot base coordinate. As all the objects that the camera sees are in the camera coordinate, it is a must to find a transformation matrix from this coordinate to the robot base coordinate. We use Aruco tag to find this matrix.

- The Aruco tag is attached on the end-effector of the robot (the gripper is temporarily removed) so that the transformation matrix between the tag's coordinate C_{tag} and the end-effector coordinate C_e is known.
- The transformation matrix between C_{tag} and the camera coordinate C_{cam} can be computed using any Aruco tag software package. We use aruco_ros and the detection result is shown in Figure 10.
- The transformation matrix between the C_e and C_{base} is provided by the forward kinematics of the robot.

The homogeneous transformation matrix between C_{cam} and C_{base} is just the multiplication of the above matrices:

$$M_{cam}^{base} = M_{e}^{base} * M_{tag}^{e} * M_{cam}^{tag}$$

Using M_{cam}^{base}, we can easily convert the point cloud data in C_{cam} to C_{base} and put everything in OpenRave as shown in Figure 11.

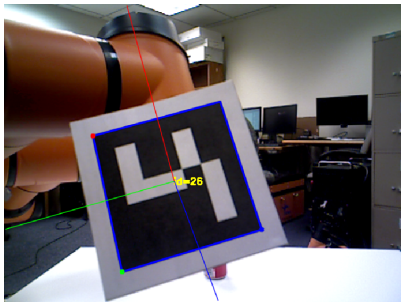

Figure 10: The tag coordinate in the camera frame.

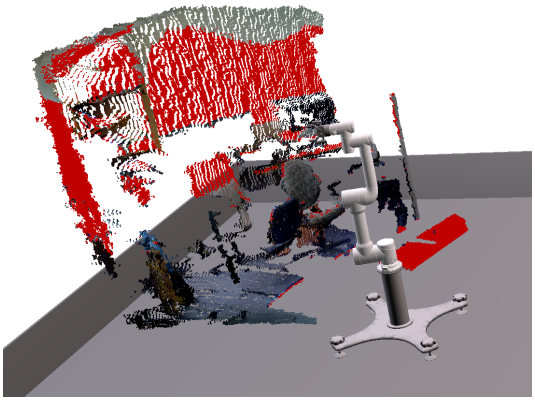

Figure 11: Simulated environment in OpenRave.

After we have everything transformed to C_{base} including the robot, the objects, and the desired grasp pose; TrajOpt will generate a collision-free trajectory. The trajectory is a series of 6 joint positions which can then be easily sent to the actual robot for replaying the trajectory.

Figure 12: Hardware setup.

5. Experimental Results

For the vision system, the camera we use is an Asus Xtion Pro RGB-D camera. The camera outputs 640x480

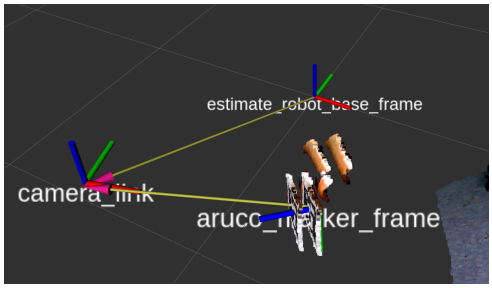

Figure 13: Mitigated depth error after a calibration.

RGB images along with point cloud data. The camera is fixed at a position on a table during the experiment. For the robot arm, we use an Aubo-i5 robot from AuboRobotics. It is a 6-DOF collaborative robot whose payload is 5 kg. We control the robot using a Python driver provided by the manufacturer. A Robotiq 2-finger 85 adaptive gripper is mounted on the arm. The gripper is position-controlled with positional feedback and adjustable gripping force from 20N to 235N. The whole setup is illustrated in Figure 12.

5.1. Camera Calibration

As grasping object will require very accurate positional information, it is essential for the camera to be properly calibrated. There are two different calibration procedures for RGB and depth images.

- RGB calibration: We use the procedure mentioned in this ROS tutorial [2]. A checkerboard of size 8x6 is used in this step.
- Depth calibration: The camera that we use is notorious for error in depth information without calibration. We also suffered the same error of about 5 cm at the distance of 50 cm. It can be seen in the Figure 13, the tag from point cloud data is moved 5 cm in front of the actual position. We follow the instructions for calibrating the depth sensor by using jsk_pcl_ros package [3]. The idea is to align the depth estimation from RGB images with the depth calculated from the depth sensor. We used the same size checkerboard with smaller grid in order to cover better our working space while performing the calibration. After calibration, the error is reduced to 1 cm.

5.2. Training YoloV3 Network

We collect data by scraping online images from 4 categories: eyeglasses, bottles, soda cans, and cups. To come up with this list, we need to test the performance of GDP on these objects to make sure that the algorithm can generate valid grasps on these objects. The neural network used in GPD was trained in certain objects, as a result, it might not generalize with many novel objects. We collect about 100 images for each class for training and validation. After that, we manually label the data and perform fine tuning with the instruction for training with custom objects from a github repo [4]. We use 80% of the total images for training and the remaining for testing. We begin with all the hyper-parameters are kept at their default values and the initial weight was taken from darknet53.conv.74. We continue training until the loss plateaus and the final performance is 84.39% mAP with Intersect Over Union (IOU) at 0.5. An example of detected objects on an RGB figure is shown in Figure 5.

5.3. Results

The trajectory generated by TrajOpt, given the transformed point cloud data, the robot model, and the desired grasp pose is shown in Figure 14. The scene with point cloud data is first transformed into simpler meshes for collision checking. After that, the path is generated to transform the current pose of the gripper to the desired grasp pose. At the final step, the gripper was able to reach the coke can and grasp it. The generated trajectory is replayed in the real robot, showing in Figure 15. The figure shows the accuracy of the generated path where the gripper was able to come close to the surface without making any collision. The coke can is also in the middle of the grasp, ready to be grasped. The complete video of object grasping and path planning can be seen in https://bit.ly/2HlsQqa and https://bit.ly/2LBAX7H. We also tested with several positions of the can, all resulting in efficient trajectories.

6. Discussion

In this paper, our key contributions are summarized as follows:

- We combine state-of-the-art algorithms in object detection, grasp pose detection, and motion planning to form a viable solution to an autonomous waste sorting robot.
- We modify the grasp filtering procedure of the existing grasp pose detection algorithm so that it can work robustly with our gripper and in the context of waste sorting. We validate the performance with an experiment of a complete autonomous trash sorting sequence with different types of waste items.

Finally, the results of the experiment grasping the coke can show that our proposed combination of vision algorithms, grasp detection & filtering, and the manipulator control method can achieve object grasping from point cloud data accurately and efficiently.

[2] http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration
[3] https://jsk-recognition.readthedocs.io/en/latest/jsk_pcl_ros/calibration.html
[4] https://github.com/AlexeyAB/darknet

Figure 14: Path planning by TrajOpt.

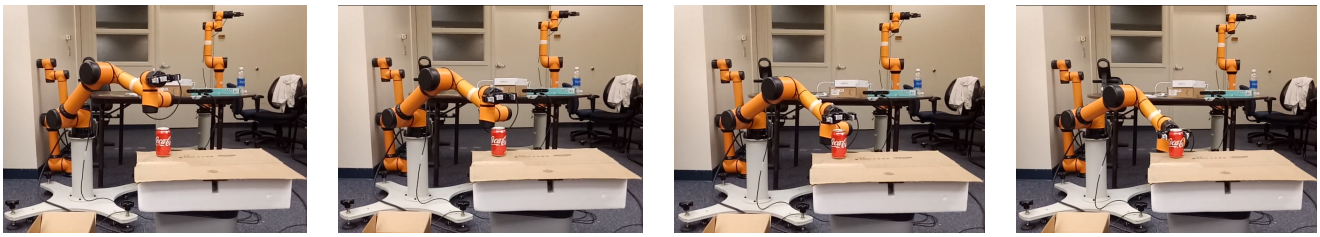

Figure 15: Generated trajectory replayed on the robot.

7. Conclusions and Future Work

In this paper, we present the result of our project to develop an autonomous waste sorting robot from an available gripper. The results show the potentials of the proposed approach in the waste sorting context as the proposed approach can make the gripper be able to grasp objects given the input of point cloud data of objects to be grasped. The approach can easily be applied to any gripper and objects. In the future, there are steps that we would like to take:

- Improving the grasp pose detection filtering so that the grasp selection can be more reliable.
- Create a single software package with the integration of all needed programs. It would increase the robustness and decrease the computation time.
- Deploying the code in an embedded computer instead of the current bulky desktop computer. This might include running the trained YOLOv3 network on a cheap neural compute stick instead of on an expensive GPU.

References

[1] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt. High precision grasp pose detection in dense clutter. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 598–605. IEEE, 2016.

[2] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[3] Z. Kingston, M. Moll, and L. E. Kavraki. Decoupling constraints from sampling-based planners. In *International Symposium of Robotics Research*, Puerto Varas, Chile, 2017.

[4] J. V. Kujala, T. J. Lukka, and H. Holopainen. Picking a conveyor clean by an autonomously learning robot. *arXiv preprint arXiv:1511.07608*, 2015.

[5] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[6] T. J. Lukka, T. Tossavainen, J. V. Kujala, and T. Raiko. Zenrobotics recycler–robotic sorting using machine learning. In *Proceedings of the International Conference on Sensor-Based Sorting (SBS)*, 2014.

[7] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

[8] S. G. Paulraj, S. Hait, and A. Thakur. Automated municipal solid waste sorting for recycling using a mobile manipulator. In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V05AT07A045–V05AT07A045. American Society of Mechanical Engineers, 2016.

[9] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. 2009.

[10] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE, 2015.

[11] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Citeseer, 2013.