Autonomous Object Model Acquisition with a Robotic Arm

Victoria Colthurst, Kazuki Shin, and Joohyung Kim

Abstract— **This paper presents an autonomous object registration pipeline deployed on a self-contained camera and robotic arm hardware system that produces 3D object point clouds. Experimentation evaluates the quantitative and qualitative aspects of the produced point clouds, and we show that our methods produce representations comparable with state-of-the-art methods. It is our hope that the solution presented in this paper will aid in perception-based tasks that require identifying and interacting with objects in an unstructured environment.**

I. Introduction

In situations where a robot is to act autonomously, it must be able to identify and interact with objects in a cluttered environment. The problem of interacting is exacerbated when the object is obscured and the shape cannot be known completely. Humans have some concept of what an object looks like, even without seeing the entirety of the object. In a cooler filled with ice and drinks, we can identify the type and shape of a drink and pick the drink out of the cooler with only a partial view.

A common interaction is grasping, and current solutions fall into two categories, either geometry-based [1]–[3] or learning-based [4]–[10]. Both categories attempt to produce object agnostic grasps. Geometry-based grasping can achieve grasp success rates up to 86% [1]. Learning-based grasps can achieve even higher grasp success rates up to 97% on previously seen objects, but this number lessens by a few percent depending on the database and when encountering novel objects [7]. Additionally, learning-based methods require large training sets of objects and annotated grasps. The grasps produced by these methods are often limited to the input partial view of the object and all solutions mentioned previously involve an above workspace camera view, which we do not have. Geometry-based methods can generalize well without needing to collect training data and demonstrate predictable behavior compared to learning-based methods. The placement of our camera near the gripper, and the steps we take to grab the initial point cloud scan, allow us to generate both top-down and forward-facing grasps with a geometric-based method. There are solutions [11], [12] that attempt to algorithmically fill gaps in object meshes or even learn how to predict the unseen portions of an object mesh to aid in the identification or grasping tasks. However, the algorithmic filling cannot handle large chunks of missing object information, and the learned prediction must still overcome the sim-to-real gap.

Victoria Colthurst, Kazuki Shin, and Joohyung Kim are with KIMLAB (Kinetic Intelligent Machine LAB) at the University of Illinois Urbana-Champaign {vrc2, kazukis2, joohyung}@illinois.edu

Another approach to the problem is building up a database of objects, often with stored information on object properties, possible grasps, etc. Similar to a human using their experience and knowledge to pull a drink from a cooler, a robot can use a database to identify and interact with objects it has seen before. Matching database models to partial views of objects in clutter is a solved problem, with solutions varying mostly on the types of descriptors they use to make the matches [13]–[15]. Building up the initial database of objects remains a challenge. The existing state-of-the-art methods involve placing objects on a rotating platform [16] to gather known angles of an input object and align them together. These solutions require additional hardware, human intervention to evaluate the quality of the reconstruction, and manual changes to the registration. Other solutions tackle building the database by creating object models from 3D point scans using robot arms [17], [18]. In [18], the authors describe a hypothetical extension to their solution, which would set the object down and pick it up in a different orientation to fill in the areas of the scan obscured by the gripper in the first grasp. Unfortunately, this hypothetical extension was never realized and [17] excludes thin objects like bowls due to choice in representation.

A robot equipped with the ability to autonomously scan in and store object representations in a database can tackle the goals of object identification and interaction in the face of novel or unique objects. Our solution is an autonomous object registration pipeline that uses a self-contained robotic arm and camera hardware system. The robot grasps and rotates objects in place to get the full 360 degrees scans of the object before registering these scans together into one cohesive representation. Our solution uses 3D point clouds to represent the objects. This paper presents the results of the object registration pipeline both in isolation of and with robot grasping as the turning method. In addition, we present experimental results from geometry-based techniques for grasping and rotating objects tested on hardware.

This paper is organized as follows. Section II goes in-depth about the methodology of our solution, including our processes for point cloud registration and object grasping and rotation. Section III presents the results of our experiments trialing that methodology, and Section IV concludes and discusses possible future work.

II. OBJECT REGISTRATION PIPELINE

The object registration pipeline is an autonomous scanning solution for obtaining point clouds of novel objects. The general outline of the pipeline is as follows. First, the goal object is identified within an image of the robot's workspace.

Manuscript 253 submitted to 2023 20th International Conference on Ubiquitous Robots (UR). Received February 19, 2023.

Normally, scanning poses would be calculated in arcs around the goal object. In our trials we opt to use three hard-coded scan poses to reduce computation time and remain within the workspace limits of the hardware. Second, images are taken at each scanning pose, converted to point clouds, and registered together. Third, the goal object point cloud is used to generate grasps. The object is then grasped and rotated 30 degrees clockwise to present another view. This process is repeated until all 360 degrees have been observed and saved. Lastly, the collected point clouds are registered together to create one cohesive 3D model of the goal object.

A. *Static Scan Registration*

In this step we register together the RGBD images taken at each scan pose of one view angle. The collected point clouds are registered pairwise to create a pose graph, where each node is a point cloud and each edge describes the transformation between the clouds. The input point clouds are always pre-processed before being given to the registration algorithms described next. The pre-processing step consists of down sampling the point clouds at a given voxel size and estimating the normals of each point in the cloud. The voxel size decides the spacing of a 3D grid used for sampling the input point cloud, where a smaller voxel size produces a higher density point cloud.

We use a combination of feature-based global registration, GICP, color ICP, and point-to-plane ICP for the pairwise registration. An initial rough alignment between the source and target point clouds is made using Fast Point Feature Histograms (FPFH) feature-based global registration [19]. The table plane is then removed from the point clouds since the registration struggles with sparse scenery and excessive points belonging to the table create false high overlap alignments. Plane segmentation relies on random sample consensus (RANSAC) to fit a plane model to the largest possible group of points in the cloud [19]. After removing the table planes, the point clouds are re-registered using the FPFH feature method. After these two rounds of global registration, the alignment is refined in a loop of decreasing voxel sizes, $[0.005, 0.002]$, using color ICP and then GICP [19] [20]. After the loop, non-overlapping points are temporarily removed and the clouds are passed to point-to-plane to refine the final alignment using our smallest voxel size, 0.001 m^3. Non-overlapping points are defined as not within d of points in the opposing cloud, where d is the average distance between points within the point cloud.

The final transformations for each pair of clouds are used to create the total, cohesive point cloud representing the workspace environment, PC_global. PC_global is then processed to isolate the points that belong to the goal object. To do this, the table is segmented out of the input point cloud and the remaining points are divided by density into clusters using the Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [19]. The table plane's normal information is saved during this step. The scanning pipeline saves the depth value from the center of the camera frame (i.e. x=y=0) during the first scan. The distance from each cluster's center to the center of the frame and to the initial depth value is calculated and the cluster with the smallest distance is selected as the goal object. An example of this process is shown in Figure 1.

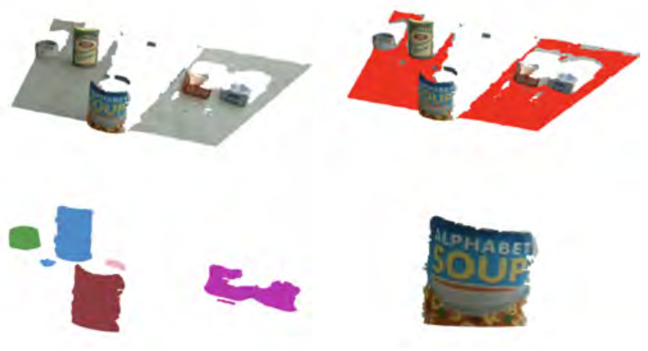


Fig. 1: From top left to bottom right: the initial input scan of the scene, the calculated table plane, the result of DBSCAN, the final isolated goal object.

B. *Object Grasping and Rotation*

The hardware setup requires that the object be rotated in place to see the object from all sides. The grasp generation process is based on the 2019 paper, 'Fast geometry-based computation of grasping points on three-dimensional point clouds' [1]. This method produces antipodal grasp points for parallel grippers, is object agnostic, operates on partial object views, and has a fast CPU runtime for large point clouds while maintaining an 85% grasp success rate. The modifications we make to this method for our setup will be described here.

1) Generate Grasp Candidates: Grasp candidates are pairs of points that represent where each pad of the parallel jaw gripper will be centered. Candidates are chosen to be close to the center of mass and perpendicular to the main axis of the object. Using principal component analysis (PCA) directly, as the paper does, often produces a main axis through the diagonal of the object cloud. Instead, we find the bounding box of the object point cloud, which produces a main axis parallel or perpendicular to the table plane. We rotate the point cloud and bounding box such that the longest side of the bounding box aligns with the x-axis.

We take a 7 mm slice of the object cloud perpendicular to the x-axis at a height central to the object, the central plane. To find points along the edge of the object, we take the maximum and minimum of the plane along the y and z axes. This produces four points which will represent the center of four bounding boxes. We define the side length of these boxes along the x-axis as one fingertip width, and along the y-axis and z-axis as the minimum of one fingertip width or one third of the object width in y or z.

We then use these four bounding boxes to crop out points from the original input object point cloud. Each collection of cropped points is a candidate region. The final pair of points

Manuscript 253 submitted to 2023 20th International Conference on Ubiquitous Robots (UR). Received February 19, 2023.

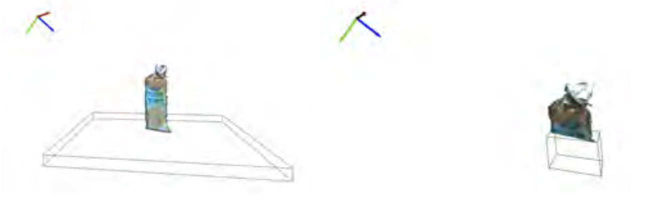


Fig. 2: The table bounding box is shown underneath a scan of the Orange Juice object on the left. The right shows the cropped equivalent for top-down grasping. The top left frame in each image represents the camera frame.

representing the grasp will be one from each candidate region of the same axis.

2) Rank Grasp Candidates: This step takes as input two candidate regions at a time and their associated normal and curvature values. The ranking function we use is identical to the reference paper's. It combines four metrics to choose the input point pair that will produce the most successful parallel jaw grasp: the pair's distance to the central plane (grips close to the center of mass), curvature of each point (places gripper pads on flat parts of the object), how antipodal the point pair is (keeps gripper pads parallel), and whether the vector connecting the two points is perpendicular to the main axis of the object (grip parallel to the table). We filter for grasps wider than the max width of the parallel jaw gripper and return the highest ranked point pair from this step. We run this step twice for candidate regions on the y and z axes, which produces two highly ranked point pairs.

3) Filter Collisions: We filter the best grasps from the last step for collisions with the gripper and the table or the object itself. The first step in the filtering process is to create two types of bounding boxes that represent the table and the gripper pads.

To create the table box, we take as input the object point cloud, the table plane's normal n, and the table plane's approximate center, c. We first rotate the object cloud and c to align n with the x-axis. We want the x-axis to represent height off of the table; however, n may point up from the table towards the object or down towards the floor. To remedy this, we count how many object points have positive or negative x-values and rotate n accordingly. Then, we find the center of the object at table height, p, by replacing x of the object cloud mean with the x from c. We translate the cloud such that p lies at the origin. Finally, the table bounding box is defined as a square with a side length of 0.5 m and a height of 0.025 m centered below the yz plane.

We represent the parallel jaw gripper using two bounding boxes for the pads that will make contact with the object itself. We take as input a grasp candidate (p_1, p_2), the object cloud, and the table normal n. First, we rotate the inputs such that both grasping points lie on the x-axis and then translate the inputs to have one of the two input points lie on the origin. Next, we define the orientation $R = (x, y, z)$ of the bounding boxes. We have y represent the vector from the smaller x-valued point p_1 (the origin) to larger x-valued

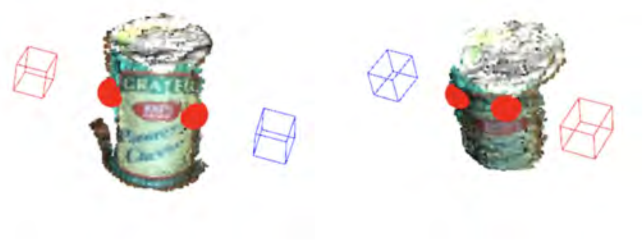


Fig. 3: Example gripper pad bounding boxes from a given grasp.

point p_2. The orientation of the bounding boxes is defined as: $R = [n \times y, p_1 - p_2, x \times y]$.

The center of each gripper box, c_1 and c_2, are defined along the x-axis as shown below, where g is the gripper opening width in meters.

$$c_1 = (p_1[0] - g/2, 0, 0)$$

$$c_2 = (p_2[0] + g/2, 0, 0)$$

We offset the centers to allow extra padding for motion errors. The extent of bounding boxes is defined exactly as that of the physical gripper pads based on the measurements from [21]. We combine these three elements of center (c_1 and c_2), extent, and orientation R to create the two bounding boxes that represent each pad of the parallel jaw gripper.

Given the table bounding box, we loop over every potential grasp candidate and produce their associated gripper pad bounding boxes. Within the loop, we check if either gripper bounding box intersects the table bounding box or intersects the object cloud itself. The intersection of bounding boxes is defined as whether there is overlap on all 3 axes when the boxes are collapsed onto each axis of x, y, and z, after both boxes have been oriented to be axis-aligned. If there is no overlap on any one axes, that means that the objects do not collide.

4) Generate Grasp Poses: We calculate grasp poses for any grasp candidate that makes it through collision filtering, and return these grasps and their poses in order of their rank. The input to this grasp pose generator is the object point cloud, the non-colliding grasp point pair (p_1, p_2), and the table normal n and center c. The output grasp poses are in the form of a position (x_p, y_p, z_p) and an orientation of quaternion type $q = (x_o, y_o, z_o, w_o)$. We produce nine poses: the original grasp pose, two angular offset grasp poses, three associated pre-grasp poses, and three associated rotation poses.

The grasp position describes the center of the parallel jaw gripper, or the area between the pads of the gripper when it is fully closed and extended to its longest shape, during the grasp. We define this position $p = (x_p, y_p, z_p)$ as the midpoint mp between (p_1, p_2) with a 0.05 m adjustment along the x-axis of the grasp, where R is the rotation of the gripper.

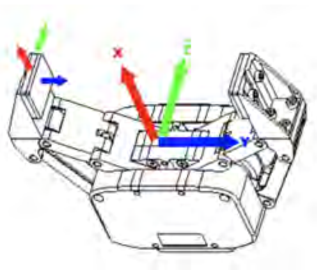


Fig. 4: This modified image shows the frame of the gripper [21]. The x-axis points in the direction of the grasp, the y-axis points across the fingers, and the z-axis points up.

$$mp = (p_1 + p_2)/2$$

$$p = (mp[0] + R * 0.05, mp[1], mp[2])$$

The gripper does not fully close while grasping an object, so the center ends up much farther into the object. Without the adjustment we notice more slippage of the grasps as they are too shallow.

The orientation of the gripper in rotation matrix form $R = (x_R, y_R, z_R)$ describes what pose the gripper will take before closing its jaws around the object. Figure 4 shows the frame we are trying to orient in this step. We start with y_R as the vector from p_2 to p_1 which is the axis between the finger pads. We define x_R as perpendicular to n and y_R, which leads to a grasp parallel to the table, or a front facing grasp. The orientation is defined as: $R = [n \times y_R, p_1 - p_2, x_R \times y_R]$.

We then convert this rotation matrix into a quaternion representation $q = (x_o, y_o, z_o, w_o)$ and combine it with the position found previously to complete the original grasp pose, the first of nine. All associated pre-grasp poses modify the position by -0.086 m, the measured length of the gripper's fingers when fully extended, along the x-axis to put the gripper in front of the object.

After grasping and lifting the goal object, we want to turn the object 30 degrees in place and set it back down on the table to view a new angle of the object. To do this, we obtain R_{30}, a rotation matrix that describes a turn about the table normal n by 30 degrees. We multiply the orientation from earlier R with R_{30} to get the final desired orientation of the gripper.

The calculation for the rotated gripper position $p_r = (x_p, y_p, z_p)$ that keeps the object center in the same place on the table requires rotating the midpoint mp about the approximate center of the object scan by 30 degrees. The calculation for approximate center takes as input an object cloud where its main axis is aligned to the x-axis. It takes random slices of the object in the yz-plane, fits those slices to a circle, and averages the centers of those circles. The function to fit a slice to a circle is based on [22], which uses least squares to approximate the parameters of the circle given a list of points. The average of the circle centers is the approximate center of the object.

After applying the 0.05 m adjustment and the R_{30} rotation, we have the complete rotated gripper position p_r. Combining this position with the previously calculated orientation we have the complete pose for the rotated grasps.

The final step of the grasp pose generation process is to disturb the basic and rotated grasp poses by 15 degrees around the y-axis of the gripper. This is to account for the limited mobility of the robot hardware and give the arm more options for possible grasp approach angles. All nine poses are returned at the end of this step.

5) Top Down Grasping: In addition to creating grasps that approach parallel to the tabletop we create top-down grasps that approach along the table normal n towards the object. The modifications to the previous procedure are listed here.

1) The object point cloud input to the grasp generation process is a cropped to leave only the top 0.086 meters of the object.
2) The table box represents the remainder of the object underneath the cropped input. The length and width of the box are limited to the size of the object in the yz plane and the table box sits under the cropped input, not at table height. An example is given in Figure 2.
3) The orientation of the gripper pad boxes is modified as: $R = [x_R, y_R, z_R] = [-z_R \times y_R, p_1 - p_2, n \times y_R]$
4) The 0.05 m adjustment of the grasp position changes from the x-axis to the z-axis of the gripper.
5) We incorporate an additional 30 and 45-degree offset around the y-axis to accommodate robot hardware limitations.

C. Dynamic Scan Registration

The final step in our object registration pipeline is dynamic scan registration, the process of registering together 12 isolated object scans taken 30 degrees apart. There are three general steps to the dynamic scan registration process: pre-processing object scans, registering object scan pairs, and combining scans with the computed transformations.

1) Pre-processing Object Scans: In the Experiments section, four modifications to the dynamic scan registration will be compared to find the most optimal. The first step is to set the two parameters that determine which configuration will be run: register with regions and favor overlap. Registering with regions means that the object scan will be segmented into regions [23]. The most similar regions will be registered together and compared against a whole scan registration. Favoring overlap means that the best registration between two scans where non-overlapping points were removed will be favored over registrations between complete scans.

Next, the goal object is segmented out of each of the 12 statically registered scans using the techniques discussed previously, which are generally: removing the table plane, segmenting objects by density, and choosing the object most central to the camera frame. Then, we pre-rotate each isolated object scan by its estimated pose between 0 and 330 degrees and stack them together using their estimated centers.

Many of the objects in our test set share a similar shape and only differ in color or pattern, so we prioritize color ICP registration. We found through experimentation that color ICP can be sensitive to the number of colors it needs to align, doing better with fewer. To simplify the color profile of the objects, we run every point cloud through a color quantizer. The color quantizer uses k-means to cluster colors in the input array into n goal clusters.

The last step is used when the input parameter concerning the use of regions is true. We segment each point cloud into regions and calculate the global orthogonal object descriptor (GOOD) for each region [23] [24]. GOOD is invariant to object orientation and has a spacing constant n, where a higher n will lead to a fine descriptor and a lower n will lead to a coarse shape descriptor. Using the GOOD algorithm, we produce descriptors for every region from each point cloud using three values for n: 10, 30, and 50. By comparing these three descriptors we find the top 3 most similar pairs of regions between the input source and target point clouds.

2) Registering Object Scan Pairs: We loop over the list of 12 object point clouds and run the registration algorithm on the neighboring clouds. This is because the registration of scans taken 60+ degrees apart does not produce useful results.

The outer function takes in the pair of point clouds being registered and their color reduced versions, their top 3 most similar regions, and the estimated center of the stacked scan c. It makes 5 attempts at registration for the input pair to account for the starting randomness of the global registration functions. Over the 5 attempts, the highest fitness transformation is tracked and eventually returned as the transformation that best aligns the input pair. The fitness metric describes the overlapping area between a source and target point cloud, or the number of inlier correspondences divided by the total number of points in the target cloud. Within each loop, the outer function compares the registration of the color corrected point clouds, the top 3 most similar regions, and a cropped version of the color corrected point clouds. These three registrations are compared with the current maximum fitness transformation at the end of each loop.

The cropped version of the color corrected point clouds removes potentially non-overlapping points. We translate the point clouds so c lies at the origin and then project the clouds onto a unit sphere. The pre-processing done to rotate the objects to their expected pose gives us our estimated overlap when the clouds are projected. We then remove the non-overlapping points and return the two clouds.

The registration function used by the outer function first pre-processes the input clouds at a voxel size of 0.008 m. Then a global registration technique, called fast Super 4PCS, is applied to get an initial rough estimate for the transformation [25]. The results of the Super 4PCS implementation we use are heavily dependent on the input parameter: overlap, a value between 0.0 and 1.0, which describes the approximate known overlap of the input point clouds. We run Super 4PCS three times with three different overlap values: 0.9, 0.85, and 0.8. The resulting fitness are compared and the best transformation is recorded as part of the final transform.

Then, local registration algorithms are applied in a for loop of decreasing voxel sizes: [0.008, 0.0045, 0.001] and multiplier values: [3, 2, 1]. In each loop, the source and target point clouds are pre-processed at the loop voxel size and run through the color ICP algorithm. If color ICP fails more than 2 times (out of 4 in the entire registration), we disregard the entire registration attempt and return None for the transformation. Then we remove non-overlapping points and run GICP to get a tighter fit. A multiplier determines how far away a point can be while being considered overlapping as a function of the average nearest neighbor distance in a cloud. The resulting total transform is recorded as part of the final transform.

Finally, after this loop has exited, we do an additional iteration of color ICP and Point-to-Plane ICP, at a voxel size of 0.001 m, to refine the transformation. The accumulated final transform and its associated fitness is returned as the best alignment to the outer function mentioned earlier.

3) Combining Scans with Computed Transformations: After finding all transformations between neighboring object scans, we use them to combine the scans into a complete representation of the object. The transformation between the 0-degree scan and the 330-degree scan is the 12th transformation computed. Only 11 transformations are necessary to complete the representation, but we use the 12th transformation as an alternative. During the final combination, we remove the lowest ranked transformation by fitness from the 12 transformations. During this process we transform the source cloud using the given transformation, then crop the overlap of the current two object scans, and refine the registration once more using the Point-to-Plane ICP. We notice a significant improvement in the alignment after adding this final refinement step.

III. Experiments

A. Experimental Setup

1) Hardware Setup: We use a 6-DOF arm from the PAPRAS (Plug-and-Play Robotic Arm System) [26] with a D435 Intel Realsense Depth Camera attached on top of a parallel jaw gripper. The arm system is tethered to an off-board power source and multicore CPU compute system. We assume an isolated object is centered on the table.

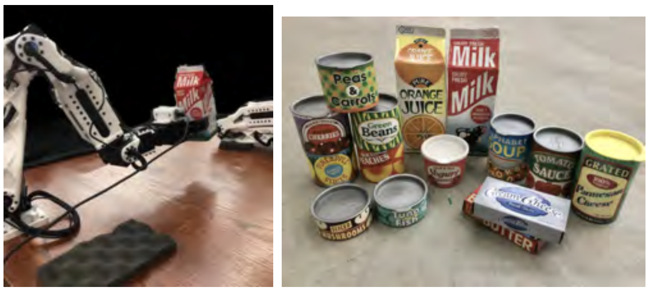


Fig. 5: Experimental setup showing the PAPRAS arm mounted on the table and the RGBD camera mounted on the gripper at the end of the arm. The image on the right shows the superset of 15 objects from the HOPE Dataset.

Manuscript 253 submitted to 2023 20th International Conference on Ubiquitous Robots (UR). Received February 19, 2023.

TABLE I: Frontal Object Grasping

Object	Grasps	Slips	Avg Rank	Avg Grasp	Grasp Rate
Milk	3	3	7.965	1.50	50%
Parm	4	0	7.604	1.50	66%
Tomato	3	0	8.568	1.33	50%
Beans	2	1	6.303	0.33	33%
Yogurt	2	0	7.666	0.33	33%
Mushr	1	0	8.205	0.33	17%
Cream	4	3	5.753	1.11	44%

TABLE II: Frontal Object Grasping, Tallest Poses Only

Object	Grasps	Slips	Avg Rank	Avg Grasp	Grasp Rate
Milk	3	2	7.706	2.00	100%
Parm	3	0	7.545	2.00	100%
Tomato	3	0	8.568	2.00	100%
Beans	2	1	6.303	0.66	66%
Yogurt	2	2	7.666	0.66	66%
Mushr	1	0	8.205	0.33	33%
Cream	4	2	5.753	1.11	66%

TABLE III: Frontal Object Grasping 360s

Object	Grasps	Slips	Avg Rank	Avg Grasp	Grasp Rate
OJ	9	4	7.620	2.00	90%
Parm	8	3	8.175	2.00	100%
Tomato	8	1	8.106	2.00	88%

TABLE IV: Top Down Object Grasping

Object	Grasps	Slips	Avg Rank	Avg Grasp	Grasp Rate
Milk	3	2	8.032	1.50	50%
Parm	5	0	8.315	1.67	83%
Tomato	4	1	7.346	1.50	66%
Beans	1	2	6.776	1.17	17%
Yogurt	1	2	6.783	1.67	17%
Mushr	1	2	5.318	1.33	17%
Cream	6	0	6.899	1.44	66%

The RGBD camera has an optimal range of 0.3 to 3.0 m and cannot record depth for translucent or overly shiny textures. Evaluation is conducted using a subset of 9 objects from the Household Objects for Pose Estimation (HOPE) dataset: milk carton, orange juice carton, alphabet soup can, green beans can, tomato sauce can, mushrooms can, yogurt cup, Parmesan cheese can, and cream cheese box [27]. Many objects in the HOPE dataset repeat shapes while only changing the coloring of the object, so we use one or two representations from each category of shape.

2) Software Setup: The methods are implemented in Python and integrated with robot hardware using ROS and MoveIt. We use two ROS nodes to exchange messages that execute the object registration pipeline on the robot. One node represents the software pipeline and communicates information like grasp poses and collision objects to the hardware node. The hardware node tells MoveIt about objects to avoid during motion planning and moves the arm to desired poses. This node also modifies the gripper positions with a constant value to lift and set down the objects in the world frame.

B. Evaluation of Object Grasping

The experimental results for both grasping types, frontal and top-down are presented here. For each object grasping method we perform between 6 to 9 grasp attempts on the subset of 9 HOPE objects, each at a different pose.

In Tables I to III we record the number of successful grasps (Grasps), the number of times the gripper slipped off the object (Slips), the average rank of the top-rated grasp (Avg Rank), the average number of generated grasps (Avg Grasp), and the total grasp success rate taken out of the number of attempts (Grasp Rate) for frontal object grasping. Table II only considers the poses that leave the objects at their tallest heights.

We note the significantly better success rate in the tallest only table, Table II. Short objects produce grasps that are close to the tabletop which are often filtered out by our collision detector. In Table I we confirm this by noting that the average number of successful grasps generated is less than 1 for the shortest objects and more than 1 for the tallest objects. If we include all 6-9 pose types, the average rate of grasp success is about 42%. If we remove the short poses and limit the evaluation to 3-6 poses only, the average rate of grasp success jumps to 76%.

Table III depicts the same grasp metrics as before but taken during the 360-degree scan collections on a subset of the 9 objects. Short objects are removed from this 360 scan trial due to high grasping failure rates. The number of attempts jumps to 9-12 and the average grasp success rate goes from 76% to 93%.

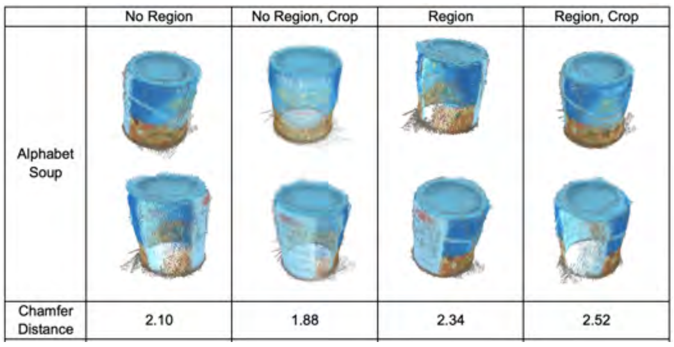
In Tables IV to V we record the same metrics as before, but for the top-down grasping method. We can see in Table IV that for all objects using the top-down grasping method the average number of successful grasps generated is greater than 1. Despite the top-down grasp poses being generated, we still see a high failure rate for grasping short objects. This is due to MoveIt motion planning failures.

The average grasp success rate is about 45% for all poses in Table IV and about 75% for the 360 grasps in Table V.

Next, we discuss the success of rotating the object 30 degrees after grasping it successfully. Using all 6-9 poses for each object, we see an average of an 89% rotation success rate for both methods. This number reflects how often MoveIt fails to plan a path to the rotated pose. When the rotation is tested only on the most successful initial pose, we see a 100% rotation success rate for both methods. For frontal grasping, we note a 41.5 average degree change from the first trails and a 44.1 average degree change in the second. The degrees are highly skewed by large numbers from when objects slipped or bounced on the table and rotated more or less than the pose should have allowed. We note the rotations are especially off for the box objects as the parallel jaw gripper would align the box with the gripper and cause the box to rotate in the gripper before being rotated and set down. For the

TABLE V: Top Down Object Grasping 360s

Object	Grasps	Slips	Avg Rank	Avg Grasp	Grasp Rate
Milk	8	3	7.255	1.50	66%
Soup	10	1	7.636	2.00	83%



	No Region	No Region, Crop	Region	Region, Crop
Alphabet Soup				
Chamfer Distance	2.10	1.88	2.34	2.52

Fig. 6: Final Alphabet Soup representation from four method configurations for hand turned method.

top grasping method, we see a 26.0 average degree change from the first trials and a 32.2 average degree change in the second. We note the top down rotation method suffers less from overturning than the frontal method.

We conclude that our gripper cannot effectively grasp short objects. Excluding short objects, the frontal grasping and rotation method had a slightly better average performance than the top-down grasping and rotation method. In contrast, the paper the grasping method was based off of uses strictly a top-down grasping strategy with an above table camera view and achieves an average of 86% grasping success on simple objects [1]. We are able to achieve a 93% success rate using a gripper mounted camera view on grasping simple objects from the HOPE dataset using the frontal grasping method.

C. *Evaluation of Point Cloud Registration Methods*

An example result of the object registration pipeline is shown in Figure 6. It shows the final point cloud representations and the resulting Chamfer distance metrics, which describe the average distance between the ground truth model and the reconstructed model. To get the Chamfer distance we align the entire reconstruction with the ground truth model from the HOPE dataset and take the average distance between the closest points of the input clouds. The first table goes over the results where a human turned the objects approximately 30 degrees between scans. The next table goes over the results where the robot turned the objects.

The 'No Region' column describes the base implementation without turning on either parameter for the use of regions or favoring overlap. The 'No Region, Crop' column describes the base implementation while favoring the overlapped alignment. The 'Region' column describes the base implementation while computing and comparing against transformations from extracted regions. The 'Region, Crop' column describes the base implementation with both parameters used.

Table VI compiles the average Chamfer distance per method configuration for the hand-turned set. We give the average Chamfer distance value inclusive of all alignments and also the value exclusive of the worst alignment. We note the average Chamfer distances, even inclusive of the worst alignment, are all below 5 mm.

TABLE VI: Average Chamfer Distances, Hand Turned

	Inclusive (mm)	Exclusive (mm)
No region	4.94	2.71
No region, crop	3.67	2.89
Region	**2.58**	**2.42**
Region, crop	3.18	2.70

TABLE VII: Average Chamfer Distances, Robot Turned

	Inclusive (mm)	Exclusive (mm)
No region	2.26	2.08
No region, crop	5.44	4.18
Region	**2.23**	2.01
Region, crop	2.97	**1.87**

The method in [23] only quantifies the best alignments from 4 objects, ignoring partial or incomplete scans. These objects for their best registration have a Chamfer distance of 2.9 mm, 4.2 mm, 4.3 mm, and 3.1 mm. Our average Chamfer distances are of comparable quality to the best Chamfer distances from [23]. The Chamfer distance for our best alignments are - Yogurt: 2.43 mm, Tomato Sauce: 1.86 mm, Parmesan: 2.39 mm, Orange Juice: 1.61 mm, Mushrooms: 2.17 mm, Milk: 2.32 mm, Green Beans: 2.02 mm, Cream Cheese: 3.36 mm, Alphabet Soup: 1.88 mm.

Our average Chamfer distance for the top alignments from our four methods is about 2.23 mm, whereas the average of the four objects from [23] is 3.63 mm.

Table VII compiles the average Chamfer distance per method configuration for the robot turned set. The results for the robot-turned alignments present slightly worse qualitatively than their hand-turned counterparts. We note that the average degree change after a rotation was mostly larger than the expected 30 degrees. Despite the inaccurate degree turns, recognizable reconstructions are made for every tested object in the 360-degree scan experiment.

The best alignments for the robot-turned set are - Tomato Sauce: 1.59 mm, Parmesan: 2.04 mm, Orange Juice: 2.88 mm, Milk: 1.66 mm, Alphabet Soup: 1.38 mm. The average of best alignments is 1.91 mm, which is a better quantitative result than was produced by the hand-turned object models.

We find there is no clear winner between the four method configurations. If the processing time can be long, we would recommend running all four and choosing the alignment with the highest fitness score as the final object representation. Otherwise, the region only alignments have the best scores in Chamfer distance for both hand-turned and robot-turned scans.

IV. Conclusions and Future Work

This paper has developed a successful object registration pipeline solution on a self-contained robotic arm and camera system. This pipeline is capable of registering a complete object representation using RGBD scans taken at 30-degree intervals around an entire object. We achieve state-of-the-art quantitative results for the final object point clouds when compared with a 2022 paper [23]. Our solution is advantageous because it works on a contained system with a mounted camera, unlike its counterparts which require a separate camera that overlooks the robot workspace. Our solution uses a point cloud representation for the scans throughout, which mitigates some of the problems seen with TDSF representations.

The object registration pipeline has some limitations surrounding the type of inputs it can handle. The camera used cannot create correct depth scans for translucent objects or objects with reflective surfaces. The solution cannot handle deformable objects due to limitations in the grasping solution and the possibility for the objects to change shape between scans. Additionally, highly symmetric objects in both shape and color are not handled by our solution. The dynamic registration step of the pipeline has a long computation time as it compares multiple avenues of registration to achieve the best results. For this reason, we suggested running this last step offline.

In the future, it would be advantageous to optimize the solution for speed and improve computation time using a GPU. Additionally, we want to extend the solution to compare the four method configurations to find the best model autonomously. Extensions to this solution include registering multiple objects from one tabletop, developing a database to effectively search and store existing point cloud models, and using this database in a perception task to identify objects in the environment. It is our hope that the solution presented in this paper aids in the development of robots that can effectively interact with their environment and adapt to their surroundings over time.

References

[1] B. S. Zapata-Impata, P. Gil, J. Pomares, and F. Torres, "Fast geometry-based computation of grasping points on three-dimensional point clouds," *International Journal of Advanced Robotic Systems*, vol. 16, no. 1, p. 1729881419831846, 2019. [Online]. Available: https://doi.org/10.1177/1729881419831846

[2] T. de Haan, P. Kulkarni, and R. Babuska, "Geometry-based grasping of vine tomatoes," 2021. [Online]. Available: https://arxiv.org/abs/2103.01272

[3] S. Zhu, X. Zheng, M. Xu, Z. Zeng, and H. Zhang, "A robotic semantic grasping method for pick-and-place tasks," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 4130–4136.

[4] D. Yang, T. Tosun, B. Eisner, V. Isler, and D. Lee, "Robotic grasping through combined image-based grasp proposal and 3d reconstruction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6350–6356.

[5] A. ten Pas and R. Platt, *Using Geometry to Detect Grasp Poses in 3D Point Clouds*. Cham: Springer International Publishing, 2018, pp. 307–324. [Online]. Available: https://doi.org/10.1007/978-3-319-51532-8_19

[6] M. Gualtieri, A. ten Pas, K. Saenko, and R. P. Jr., "High precision grasp pose detection in dense clutter," *CoRR*, vol. abs/1603.01564, 2016. [Online]. Available: http://arxiv.org/abs/1603.01564

[7] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9626–9633.

[8] P. Ni, W. Zhang, X. Zhu, and Q. Cao, "Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3619–3625.

[9] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "Pointnetgpd: Detecting grasp configurations from point sets," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3629–3635.

[10] H. Duan, P. Wang, Y. Huang, G. Xu, W. Wei, and X. Shen, "Robotics dexterous grasping: The methods based on point cloud and deep learning," *Frontiers in Neurorobotics*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2021.658280

[11] W. Agnew, C. Xie, A. Walsman, O. Murad, Y. Wang, P. Domingos), and S. Srinivasa, "Amodal 3d reconstruction for robotic manipulation via stability and connectivity," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1498–1508. [Online]. Available: https://proceedings.mlr.press/v155/agnew21a.html

[12] G. Gou, H. Sui, D. Li, Z. Peng, B. Guo, W. Yang, and D. Huang, "Limofilling: Local information guide hole-filling and sharp feature recovery for manifold meshes," *Remote Sensing*, vol. 14, no. 2, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/2/289

[13] M. Nahangi, T. Czerniawski, C. Rausch, and C. Haas, "Arbitrary 3d object extraction from cluttered laser scans using local features," in *Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC)*, A. A. U. Sattineni, S. A. U. Azhar, and D. G. T. U. Castro, Eds. Auburn, USA: International Association for Automation and Robotics in Construction (IAARC), July 2016, pp. 366–373.

[14] G. Arvanitis, E. I. Zacharaki, L. Váša, and K. Moustakas, "Broad-to-narrow registration and identification of 3d objects in partially scanned and cluttered point clouds," *IEEE Transactions on Multimedia*, vol. 24, pp. 2230–2245, 2022.

[15] Y. Li, A. Dai, L. Guibas, and M. Nießner, "Database-assisted object retrieval for real-time 3d reconstruction," *Computer Graphics Forum*, vol. 34, no. 2, pp. 435–446, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12573

[16] (2018) "this autonomous 3d scanner figures out where it needs to look. [Online]. Available: https://techcrunch.com/2018/02/15/this-autonomous-3d-scanner-figures-out-where-it-needs-to-look/

[17] S. Lu, R. Wang, Y. Miao, C. Mitash, and K. Bekris, "Online object model reconstruction and reuse for lifelong improvement of robot manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1540–1546.

[18] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3d object modeling," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, 2011. [Online]. Available: https://doi.org/10.1177/0278364911403178

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

[20] "Fast gicp library," https://github.com/SMRT-AIST/fast_gicp/, 2020.

[21] (2021) Robotis e-manual. [Online]. Available: https://emanual.robotis.com/docs/en/platform/rh_p12_rn/

[22] I. Bucher. Circle fit. MATLAB Central File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/5557-circle-fit

[23] J. Li, F. Qian, and X. Chen, "Point cloud registration algorithm based on overlapping region extraction," *Journal of Physics: Conference Series*, vol. 1634, no. 1, p. 012012, sep 2020. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1634/1/012012

[24] S. H. Kasaei, A. M. Tomé, L. Seabra Lopes, and M. Oliveira, "Good: A global orthographic object descriptor for 3d object recognition and manipulation," *Pattern Recognition Letters*, vol. 83, pp. 312–320, 2016, efficient Shape Representation, Matching, Ranking, and its Applications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865516301684

[25] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4pcs library," https://github.com/nmellado/Super4PCS, 2017.

[26] J. Kim, D. C. Mathur, K. Shin, and S. Taylor, "Papras: Plug and play robotic arm system," in submission. [Online]. Available: https://uofi.box.com/s/dhftd4jyb3zym7da4n960n30616ss1m9

[27] S. Tyree, J. Tremblay, T. To, J. Cheng, T. Mosier, J. Smith, and S. Birchfield, "6-dof pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark," 2022. [Online]. Available: https://arxiv.org/abs/2203.05701

Manuscript 253 submitted to 2023 20th International Conference on Ubiquitous Robots (UR). Received February 19, 2023.