SAND2017-4756J

Multimodal Perception using Embedded Proximity and Force Sensors for Robotic Grasping

Rebecca Cox and Nikolaus Correll

***Abstract*—This paper presents our results towards fusing RGB-D images with data from contact and proximity sensors embedded in a robotic hand for improved object perception, recognition and manipulation. Optical depth information from multiple sensors is often inaccurate and inconsistent. These problems arise from problems with sensor calibration, but also occlusion of objects by other objects or the robot arm itself. In this paper, we propose to combine global pose information from RGB-D sensing with local proximity sensing during approach. Here, we use contact information based on a novel contact sensor and additional pose information provided by the arm's pose.**

***Index Terms*—Perception for Grasping and Manipulation, Reactive and Sensor-Based Planning, Force and Tactile Sensing, Sensor Fusion, Calibration and Identification.**

I. Introduction

PERCEPTION is a necessary component in grasping and manipulation that continues to be a challenge in robotics. Both 3D perception and tactile sensing have been explored, and both might be sufficient means on their own and complement each other [1]. Accuracy in object recognition depends heavily on the accuracy of calibration, successful point cloud segmentation, lighting conditions, and whether objects are occluded or not. Tactile sensing [2] can be much more accurate as the kinematics of a robot arm are usually well known and its encoders are precise, but requires active exploration of the environment.

In this paper, we describe our results from fusing 3D perception, in-hand proximity sensing, and touch into a single representation. Here, the key ideas are that tactile sensing can provide data-rich 3D representations with accuracy and proximity sensing can seamlessly bridge between the two sensing modalities, allowing us to gather data without disturbing the object's pose.

We are combining an Asus Xtion depth sensor with Robotic Materials' tactile sensors for the Kinova three-fingered hand and the parallel gripper for Rethink Robotics Baxter. We use the Point Cloud Library (PCL) to perform object recognition from the raw point cloud and find the approximate location of the objects relative to the arm. Our experimental setup is shown in Figures 1 and 2. Small errors in the camera-to-hand transformation result in a typical object location offset around 3 cm, shown in Table I. We show how we can use tactile sensing as a feedback mechanism to refine the camera-to-hand transformation for more accurate object location estimation.

R. Cox is a Master's student in the Department of Electrical, Computer, and Energy Engineering, University of Colorado, Boulder, CO (email: rebecca.e.cox@colorado.edu).

N. Correll is an Assistant Professor in the Department of Computer Science, University of Colorado, Boulder, CO (email: nikolaus.correll@colorado.edu).

Fig. 1. A typical system setup combining multiple sensor streams. The fingers on the Jaco arm from Kinova contain proximity and force sensors. An Asus Xtion is used for depth and RGB imaging.

Having accurate location information increases grasping performance, however, occluded features or objects such as an unseen handle of a cup can disrupt an attempted grasp if the gripper collides with them. We show how these tactile sensors can be used to scan objects without disturbing the environment to fill in missing points in the RGB-D point cloud demonstrated in Figure 2. Lastly, we show how the tactile sensors can be used to replace the RGB-D camera entirely through active scene exploration to locate objects, recognize them, and determine grasping poses.

II. Related Work

Using tactile sensing in perception has been widely explored, for example to plan motions to learn more about the geometry of an object or match it against a known 3D model [2], [4]–[6]. Incorporating this closed loop feedback can correct or fill in missing information from the low-cost depth cameras commonly used in research. However, tactile sensing requires touch which often unintentionally moves the object and changes the pose, requiring a new grasping pose to be determined. For this reason, proximity sensing using IR sensors in the fingers has also been explored for improving the pre-grasp pose prior to contact [7]–[9]. [10] explores using electric field sensors can be used for proximity, although this is susceptible to object composition. Even less have used the sensor data as a feedback loop to update the locations of objects [7], [11]. Little attention has been devoted on fusing proximity and depth information obtained from different means using tactile sensing to provide a common

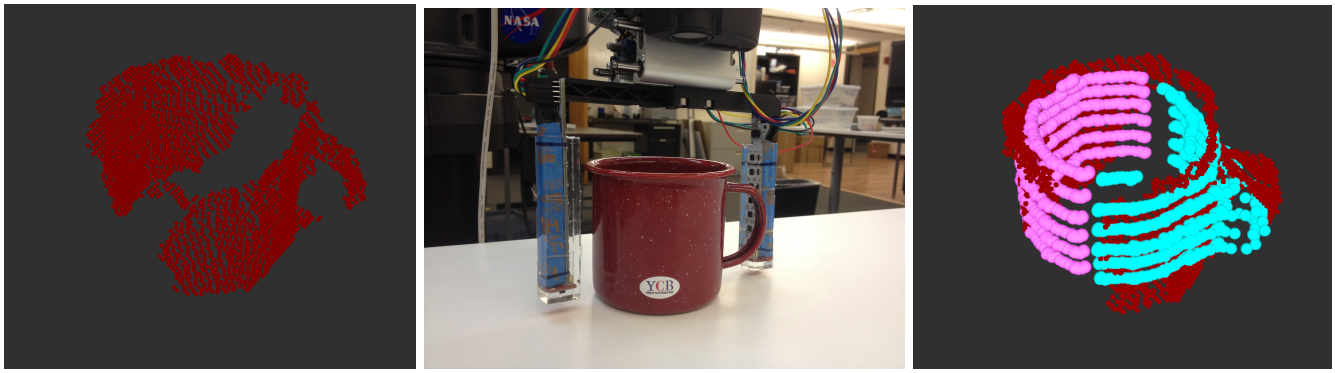

Fig. 2. All three images feature the same cup from the YCB data set [3] with the handle on the right side. Left image: Segmented point cloud from the RGB-D camera showing the cup from the YCB dataset. Middle image: Proximity sensors embedded in the Baxter parallel grippers are being rotated 180°around the cup to capture a full 360°scan. The proximity measurements are actively generating a new point cloud. Right image: Merged point clouds of the cup from the depth camera and from the 3D finger sensor scan. Pink points are from the left finger and cyan points are from the right finger. The handle of the cup is clearly shown from the scan, as well as the accurate alignment of the merged point clouds.

reference frame. Such refined models could then be used to improve grasp planning.

III. Proximity and Contact Sensors

We use combined proximity, contact and force sensors described in [12] to get proximity measurements from the robotic hand to an object with a range of up to 5 cm. The sensors are available commercially[1] and easy to integrate with existing hardware. They consists of a digital infrared distance sensor that emits infrared and measures proximity. For protection from abrasion and to allow light to pass through, they are embedded in a soft transparent polymer, which doubles as a spring for force measurements. Due to an inflection point in the sensor signal at contact, contact can be robustly estimated independently of the surface reflectivity. This information is used to improve gripper alignment during grasping and determine when contact with objects is made, which we describe in [13].

A. Contact Sensing

Unlike proximity measurements, detecting contact using these sensors is not affected by surface properties due to an inflection point that is observed in the signal [12]. Regardless of the initial amount of reflected signal, as proximity distance between the sensors and object decreases, the nonlinear increase in reflection experiences a peak rate of change. This real-time feedback has already shown to be beneficial in our experiments during grasping by providing immediate feedback of the object's location relative to the fingers when contact occurs. In previous work, this contact event was used for object detection for improving grasping and 3D reconstruction of where contact was observed. For example, thin objects in a cup such as a straw or spoon are detected from Jaco's three fingered hand using a search motion above the cup. This is often necessary in grasping to find these narrow objects for the reason that they are typically not detectable by the low-resolution depth cameras due to them not reflecting enough depth information. When contact is observed, the fingers are aligned and closed for the ideal grasp on the object for manipulation. However, this search motion requires that the sensors be moved forward slowly to prevent accidentally bumping that object before it can be grasped.

Unintentionally bumping objects is a common problem in tactile sensing that can be prevented by the combined use of proximity to detect the oncoming contact to slow the end-effector. Initially, 3D models were constructed using only contact sensing in the array of sensors embedded Baxter parallel grippers. This configuration was used for 3D modeling over the Jaco fingers due to the well known location of all seven internal sensors, as well as there being more sensors available for higher resolution models from less motions. The Jaco fingers each have one embedded sensor, where the Baxter grippers have seven internal sensors and one tip sensor that was used to detect the surface of the table. These seven sensors were touched along stacks of blocks shown in Figure 3. The 3D model clearly shows the staircase, as well as the offset in the scene calibration. However useful, making enough contact points on an object for reconstruction or object recognition is not always practical. For this reason, we explored using proximity measurements for 3D reconstruction and contact detection for optimizing scene calibration discussed below.

B. 3D Modeling using Proximity

The proximity sensors were previously used to generate crude 3D models of objects discussed in [12]. Albeit noisy, surface features of objects could be clearly scene. We hypothesized that the models were noisy due to the sensors being calibrated for properties of a different surface and the non-orthogonal incidence angles at irregular surface features. To test this, the sensors were calibrated for the surface of the cup shown in Figure 2 by moving the cup to known distances from the sensors and recording the measured reflection in the sensors. A best fit curve was found and is shown in Figure 4. It was found to be $y = a * b^x + c$, where x is the proximity distance in cm, y is the sensor measurement, and c is the base value that is unique to each sensor. The function can be inverted to:

[1]www.roboticmaterials.com

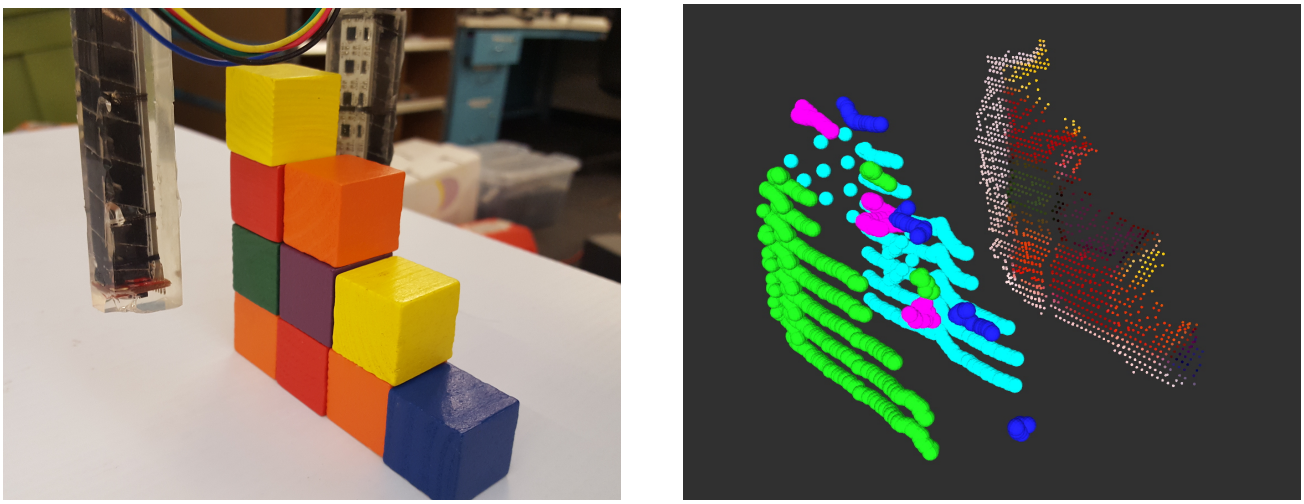

Fig. 3. Left: Scanning a staircase of blocks using the finger sensors on the custom designed Baxter gripper. Right: 3D point cloud from contact detection in the array of finger sensors compared with 3D point cloud from Asus Xtion depth sensor. The finger sensors provide precise information where the surface of the object is relative to the grippers. The offset in this badly calibrated scene can be clearly seen by the far distance ($5cm$) between these two point clouds.

$$x = \frac{log(y-c) - log(a)}{log(b)} \tag{1}$$

In the Baxter parallel gripper shown in the middle image in Figure 2, there are seven sensors embedded in the side of each finger directing inwards and a sensor embedded in each fingertip that points downward. The kinematics of Baxter's fingers are well known, as well as the exact location of each sensor in the finger. This information, as well as the proximity measurements to each sensor, allows us to construct 3D point clouds shown in Figure 2. When a sensor measurement is received above the noise floor, Equation 1 is used to calculate the location for the new point at an x perpendicular distance from the sensor and add it to the object's point cloud. In an accurately calibrated scene reconstructed in RVIZ in the right image in Figure 2, we show that we can construct a full 3D model from the sensors that aligns with the RGB-D image. In the experimental section, we demonstrate that the sensors can be used to increase grasping performance by using these 360°scans to find hidden features such as the handle of the cup.

IV. Global 3D Object Recognition

The system setup allows the RGB-D camera to be placed in an arbitrary location around the robot. To find the camera-to-robot transform, an augmented reality (AR) tag is rigidly mounted to the wrist of the robot. When visible to the camera, the system can estimate the transformation using the known location of the robot's wrist. This leaves a typical scene calibration error around 3cm that will be corrected in a later section. Objects that are visible to the camera can now be related to a location reachable by the arm.

To provide simple object detection, we built a perception pipeline using PCL that takes a raw point cloud and RGB image as input and publishes found objects and their poses. Using PCL's built in feature detection libraries, we are able to successfully perform object labeling on individual objects, but don't always have the information available in a single frame to be able to estimate object pose accurately, especially in a cluttered environment with occluded views. In such cases, the center-of-mass of the point cloud is not congruent with the actual center, making grasping difficult. Worse, key geometries that are important for grasping and manipulation are hidden and might not be inferred from partial models.

While the approach described here is basic, we believe its challenges and limitations to be ubiquitous in manipulation, even when using more advanced methods for perception.

A. Object Segmentation

The first part of the pipeline involves removing noise and narrowing down the point cloud to regions that contain the

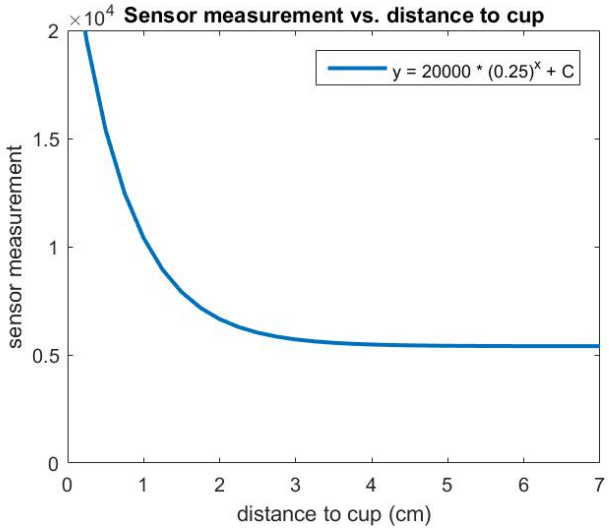

Fig. 4. Best fit function for the measured sensor values for set distances away from the cup shown in Figure 2. c, a base value that is unique to each sensor, was calculated by taking the average of ten sensor readings when no object was present.

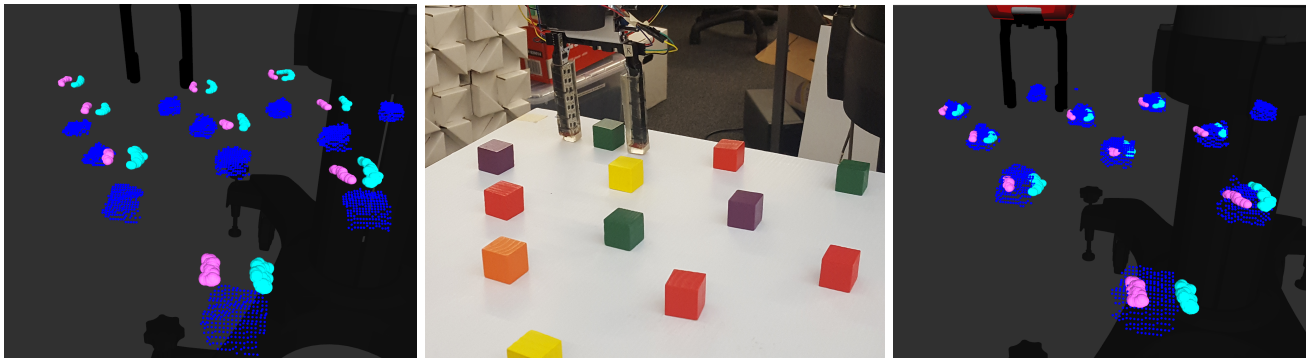

Fig. 5. The above images demonstrate the typical scene calibration error and how contact sensing is used to correct it. Left: Typical offset between the robot's end-effector and the RGB-D point cloud from the camera. The blue point clouds are the cubes shown in the middle image. The pink and cyan points show the point cloud generated from proximity sensing from the left and right fingers. Middle: This image shows the real-world view that is being reconstructed in the left and right images. Right: After applying a rigid transformation calculated from touching cubes on the table, the RGB-D point cloud aligns with the robot's grippers.

objects. We can assume that the objects will always be placed on a table or flat surface that will fill a large portion of the field of view in the camera. For this reason, the initial step is to find the flat surface and extract it from the point cloud using Random Sample Consensus (RANSAC), a simple method of separating inliers of the 2D plane from everything else in the cloud.

The remaining point cloud is clustered into smaller objects using a nearest neighbor approach. This is calculated by taking a sphere of radius r_s around every point in the cloud and grouping points contained in the same sphere. Objects that are placed further than the distance r_s from each other will be in different neighborhoods and will get processed separately. Objects that are stacked or within the spherical radius become clustered into a single object and require further segmentation techniques such as color-based segmentation.

B. Object Recognition

After the objects have been segmented, we individually analyze each one and compare it against the known data set. Due to the redundancy in point cloud information of a single object, the objects are down-sampled to their keypoints. These keypoints are sparse enough to make processing much more efficient while still containing enough information for surface reconstruction. 3D descriptors from [14] are calculated from these keypoints and used in searching for correspondence with an object from the data set. When enough matching descriptors are found, the object is labeled and a pose is estimated based on the translations from matching descriptors. This approach is described in more detail in [1].

C. Tracking Found Objects

The final step in the perception pipeline aims to improve object recognition during run time by decreasing the likelihood that an object found previously will be lost due to two main reasons: objects being moved or occluded by a robotic arm during manipulation and noise in a point cloud that results in not enough correspondences to be found that were observed previously. When an object is found, we record a geometric centroid and a timestamp. In the likely case that in the next few seconds an unidentified object is found to have a near centroid, but failed correspondence matching, then it is assumed to be the same object as identified before. We found that correctly identified objects often lost their label every few frames, even without moving the object. Small fluctuations in the received point cloud and down-sampling reduce the number of matched descriptors found. Similarly, descriptor matching can fail due to the robotic arm partially occluding the view of an object during manipulation. Keeping track of found objects in this way greatly reduced false negatives seen.

V. Merged Representation

Discussed in Section III, the point cloud generated from proximity and contact sensing is a similar point cloud to ones produced from an RGB-D camera, minus the color information. When scanning continuous surfaces, we show that the point clouds are nearly identical, like the cylindrical cup shown in Figure 2. In this section we demonstrate that we can use these sensors to improve global perception by optimizing the scene calibration transform, accurately estimating an object's pose, and performing active scene exploration in regions not visible to the camera.

A. Scene Calibration

As camera calibration is a recurrent problem in robotics, we investigated whether tactile information can be used to correct calibration error of a RGB-D camera. In addition to calibrating the intrinsic non-linear parameters of a camera, camera calibration requires calculating a transformation to find the location of the objects from the camera's viewpoint to a location relative to the robot arm. Discussed in Section IV, we use the widely used AR_tracker library available in ROS to locate the AR tag used for calculating the camera-to-robot transformation. The low resolution of the RGB-D camera and slight errors in AR tag placement result in a typical scene offset of 3cm.

To mitigate these offset errors, we investigate how contact at the end-effector can be used to correct the transformation.

We chose to focus on updating the camera pose over the poses of objects due to most of the offset errors propagating from the estimated camera pose and being similar across all detected objects. It can be assumed that the estimated pose of the objects may also have an offset from the camera due to the centroid being derived from the center of mass (COM) which is calculated from the point cloud that will contain at most three sides of the cubes, biasing the object's location towards the more visible sides. However, this offset from using the COM as the object centroid is minor in small objects such as a cube ($0.5cm$) compared to the scene calibration offset ($3cm$).

To conduct scene calibration, centroids of objects from the vision system are compared against centroids found using touch discussed Algorithm 1. The centroids consist of an (X, Y, Z) coordinate location for both the vision and the contact system allowing us to use simple, widely used algorithms to calculate the transformation correction. For example, after finding the actual centroid of an object relative to the hand, there exists a translation from the vision system centroid of the object that can be easily calculated using subtraction. When more centroids are found using touch, an algorithm based on a linear least squares approach can be applied to find a rigid transformation. The following variables are introduced in Algorithm 1:

- SAI_{0-15}, contact detection for each sensor. Sensors 0-6 are along the inside of the left finger, with 0 being closest to the palm. Sensors 7 and 15 are the tip sensors. Sensors 8-14 are along the inside of the right finger, with 8 being closet to the palm.
- S_{0-15}, sensor (x, y, z) locations using the same indexing.
- P_{0-3}, (x, y, z) coordinates
- C_o, C_f, (x, y, z) initial centroid estimation, (x, y, z) final centroid.
- $\{C_o\}, \{C_f\}$, the sets of initial and final centroids
- x_{0-15}, calculated proximity using the sensor value plugged into Equation 1
- ε, proximity threshold to the surface of an object
- $\Delta_{X,Y,Z}$, some offset in the X,Y, and/or Z direction
- R, t, rotation matrix, translation matrix
- T_c, camera-to-robot transformation

Algorithm 1: Scene Calibration

Function *find_object_centroid(object_pose)*
  move_hand_to($object\_pose + \Delta_Z$);
  while $x_{7,15} > \varepsilon_{table}$ **do**
    move_hand_to($current\_pose - \Delta_Z$);
  while $!SAI_6$ **do**
    move_hand_to($current\_pose + \Delta_X$);
    if *timeout* **then**
      C_o = $object\_pose + \Delta_{X,Y}$;
      return find_object_centroid(C_o);
  while $(x_6 < x_{14})||(x_{14} < x_6)$ **do**
    move_hand_to($current\_pose \pm \Delta_X$);
  close_grippers();
  $P_0 = S_6$;
  $P_1 = S_{14}$;
  open_grippers();
  while $x_{6,14} < \varepsilon_{cube}$ **do**
    move_hand_to($current\_pose + \Delta_Y$);
  close_grippers();
  $P_2 = S_6$;
  $P_3 = S_{14}$;
  return COM(P_{0-3});
Function *calibrate_scene*($\{C_o\}$)
  for C_o *in* $\{C_o\}$ **do**
    C_f = find_object_centroid($C_o - \Delta_{X,Y}$);
    $\{C_f\}$.add(C_f);
  if $size(\{C_o\}) < 3$ **then**
    R = identity_matrix(3x3);
    $t = ave(\{C_f\} - \{C_o\})$;
  else
    $R, t = LLS(\{C_f\}, \{C_o\})$;
  $T_c = T_c * R + t$;
  return T_c;

B. Feature Detection

In order for robots to operate in unconstrained environments, they must be able to cope with occluded objects and features of objects that are not visible from global perception. Proximity sensing has a great advantage over tactile sensing in that it does not require contact to observe surface properties. Proximity in the hand can be used to find the most ideal grasp without having to disturb and potentially dislocate the object.

When the surface of an object is parallel to the array of IR sensors in the Baxter grippers, or perpendicular to the sensor signal, an accurate calculation on the sensor measurement can be used to generate a 3D model of the object as shown in 2. The cup is an ideal object to construct a 3D model of to search for the handle due to the minimal irregular surface features present. The single irregularity, the handle, is not often identifiable in the point cloud from the camera, even when clearly visible in the RGB image due to the low resolution of the depth camera and the handle being so narrow.

To the scan the cup, Baxter grippers are lowered around the centroid until the table is sensed from the tip sensors. The wide grippers are able to freely rotate around the cup without catching on the handle. We use the known location of the sensors as well as the calculated proximity to the cup to estimate the diameter of the object in between the sensors on each finger stepped through in Algorithm 2. When the diameter measured is greater than the cylindrical diameter, we know that we have found the handle between the fingers. For this object, the handle is always closest to the sensors that measure the least proximity due the cup centroid from vision being within $1cm$ of the true centroid. For other objects where distance to the sensor is not enough information, irregularities can be detected from the rate of change of proximity measurements during scanning. As shown in Figure 6, sensor measurements scanning over the handle experience

a high rate of change.

Additional variables not introduced in Algorithm 1:

- θ, angle used to rotate the wrist
- D_{cup}, measured diameter of the object between the grippers
- D_{handle}, diameter in cm of the cup at the widest part where the handle is located

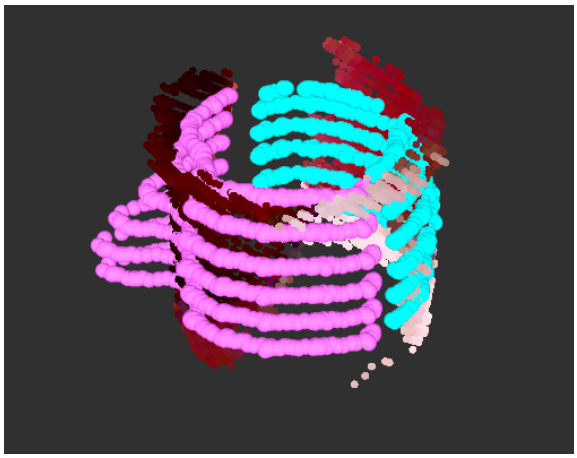

Fig. 6. Missing features in the RGB-D point cloud are found using the proximity sensors embedded in the robotic hand. Even when in full view of the camera, the cup's handle is often not discernible in the point cloud.

Algorithm 2: Proximity Sensors Handle Search

Function *find_cup_handle(cup_pose)*
  move_hand_to($object\_pose + \Delta_Z$);
  while $x_{7,15} > \varepsilon_{table}$ **do**
    move_hand_to($current\_pose - \Delta_Z$);
  while *!handle_found* **do**
    rotate_grippers(θ);
    $x_{left} = max(x_4, x_5, x_6)$;
    $x_{right} = max(x_{12}, x_{13}, x_{14})$;
    $D_{cup} = abs(x_{left} - x_{right})$;
    if $D_{cup} \geq D_{handle}$ **then**
      $handle\_found = True$;
    else if $\theta > 180°$ **then**
      return $(0, 0, 0)$;
  if $x_{left} < x_{right}$ **then**
    return $S_{left} + x_{left}$;
  else
    return $S_{right} + x_{right}$;

C. Occluded Object Detection

In an unconstrained environment, there will often be situations where no vision information is available to the camera such as the basket shown in Figure 7. Another example is the Amazon pick and place challenge where many contestants mounted the RGB-D camera on the arm so it could be moved to a position that enables it to capture images directly inside the storage bins. However, this limited perception is lost when the end-effector is moved inside the storage bin, blocking the camera's view. In both situations, perception at the end-effector is necessary for detecting objects relative to the hand and determining an optimal grasp.

Using similar scanning trajectories to those used to find objects on a table, occluded objects in an unseen region can be detected by moving the sensors slowly in a defined manner until objects or the boundaries are detected. Shown in Figure 7, to detect the salt shaker in the basket, the grippers were slowly lowered down into the basket repeatedly until the inner finger sensors detected an object surface or the tip sensors detected the bottom of the basket. If one of the tip sensors was lowered directly onto the top of the salt shaker, a "false positive" for the bottom of the basket was identified from the large difference between the locations of the known bottom and where the sensor detects it. The grippers could then be moved horizontally until they can lower without crushing the shaker.

The point clouds generated from the in-hand sensing contain the same type of points as those obtained from a depth camera, although at a the lower resolution. Compared to a depth camera which captures 640x480 points per frame, the sensors obtain at most 16 points per frame (1 point per sensor). However, many sensor frames can be combined to generate 3D models comparable and more complete than those from the camera as shown in Figure 6. Although not investigated in this paper because Rebecca needs to graduate eventually, the 3D models from the sensors are compatible with the perception pipeline from Section IV. When object recognition is not successful due situations such as overlapping objects, methods such as [15] that use the point cloud alone to find grasping poses can still be applied due to the point cloud compatibility.

VI. Experimental Validation

To verify the improvements from fusing sensory data from local perception with that found in global perception we conducted two long running experiments described in this section.

A. Optimizing the camera-to-robot transform

In Section V, we showed how touch can be used as a feedback mechanism to improve the camera-to-robot transformation. Conducting scene calibration on either system, Jaco or Baxter, we often experience a 3cm or greater error offset from the centroid calculated by the point cloud COM from vision to the actual centroid relative to the end-effector. This error can be visually seen in the point cloud overlay with the robot model in RVIZ. However, to quantify the offset and measure improvement, we used the Baxter parallel grippers to scan many cubes on a table and find their actual centroid relative to the robot.

Shown in 5, wooden cubes $2.5cm^3$ are distributed on a table with enough space in between to allow for the grippers to scan without bumping into other cubes. To keep the scanning trajectory consistent, the cubes were given identical orientations, where the sides align with the X,Y,Z axis of the robot base.

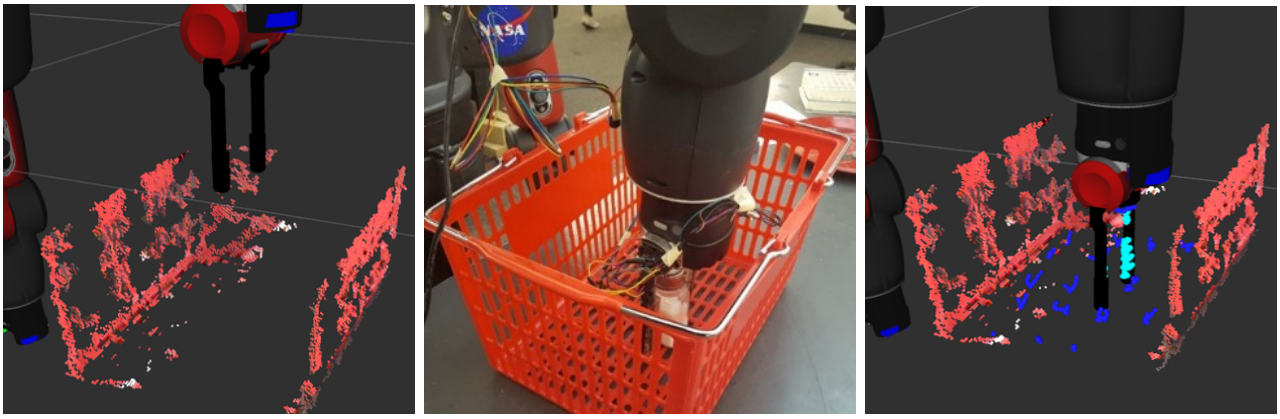

Fig. 7. Exploring areas where no global perception information is available.

Cube Centroids Verified	Transformation Update Performed	Average Offset of Centroid (cm)
0	N/A	3.39
1	Translation	1.48
2	Averaged Translation	1.36
3	Rigid Transformation	0.54
4	Rigid Transformation	0.54
5	Rigid Transformation	0.49

TABLE I
SCENE CALIBRATION RESULTS USING CONTACT SENSING TO VERIFY CUBE CENTROIDS FOUND FROM THE CAMERA

Although the cubes' orientations could be estimated using methods available in PCL, slight errors in estimation or noise in the depth image may result in a miscalculated orientation. An incorrect orientation will cause the grippers to move the cube during touch, invalidating the experiment. For this reason, only the orientation shown was used.

To compensate for the scene calibration error, a radius of $3cm$ is scanned around each cube until the cube is found, stepped through in Algorithm 1. To measure incremental improvements from verifying more cube centroids, we first measured the average offset of every cube recognized by scanning each one. Shown in Table I, the average centroid offset was $3.39cm$, which was on target to the expected offset.

We now have a set of object centroids from vision and a set from contact. When choosing one centroid at random to derive a translation matrix, we saw an average improvement in offset error of $1.48cm$. The average of two translations from two cube centroids saw roughly the same improvement, within a millimeter difference. These improvements were better than expected despite the camera-to-robot transformation often having a large rotational error, as much as $15°$around an axis. Shifting the camera's pose could cause only the selected cube to achieve a more accurate position while simply shifting the other cubes to different offset errors. However, these experiments repeatedly showed improvement from translation alone even when there existed a large rotational error. This may be due to the cubes being placed closely together on the table.

Sensor Data	Handle Found
RGB-D Image	78.13%
Proximity Sensing	100%
RGB-D Image and Proximity Sensing	100%

TABLE II
USING PROXIMITY MEASUREMENTS IN THE BAXTER PARALLEL GRIPPERS, WE

To correct a rotational offset, at least three cube centroids are required to solve the linear least squares method based on singular value decomposition (SVD). As shown visually in Figure 5, the derived rigid transformation corrected the centroid offsets to within $0.5cm$. The remaining offset is likely due to the vision centroids being calculated from the COM of the point clouds which will always be biased towards the sides that are more visible to the camera. This accurate scene calibration allowed us to align the point clouds of the cup shown in Figures 2 and 6.

B. Improved Grasping

To verify that we do in fact see improvement in grasping when we utilize proximity sensing, we tested the system's ability to detect the handle of the cup shown in Figure 2 when vision fails to do so. The wide Baxter grippers shown in the middle image are just wide enough to perform a full rotation around the cup's centroid without bumping the handle. However, the wider grippers, even when closed fully, are too wide to grasp the cup around its cylindrical center. This leaves the only successful grasping pose to be around the handle, so if the handle is not found, it can be assumed that these grippers will almost always fail to grasp the cup.

Finding the handle through vision largely depends on the noise in the point cloud and the point of view from the camera. When it is clearly visible, we can accurately estimate the location of the handle. We iterate through the point cloud and measure the distance from each point to the centroid. When we encounter distances that are greater than the radius of the cylindrical center, we know that we have found the handle and publish its location. To provide ground truth, a human is used

to verify the results between each run. It was found that vision was always accurate when the handle is visible to the camera and only failed when the cup occluded the handle. Table II shows that the probability of the handle being found from this was 78.13%. We were expecting lower success due to the handle not always being visible in the point cloud, even when it is clearly visible to the camera. However, over the course of a few frames the likelihood of one frame containing points on the handle is dramatically increased. These vision methods could easily be substituted for a more complex orientation estimation algorithm.

After a handle position is received, we test the sensors ability to find it in the same place, or if not found, be able to accurately estimate the position. Using Algorithm 2, the recognized cup is assumed to be in a well calibrated scene so that no search motion is required. The grippers are lowered to the cup and rotated until the handle is scene or 180°. As mentioned previously, the diameter of the slice of the object between the grippers is measured using proximity sensing on both fingers. When a diameter greater than the threshold is encountered, we know that the handle is between the grippers. Whichever sensors measured the least distance, meaning that the object surface is much closer to that finger, was assumed to be the side that the handle is on. When the handle is found, again the human in the loop is asked to verify the results to provide ground truth. We found that this was successful 100% of the time. This was expected since the sensor values were calibrated for this object and the surface remained parallel to the fingers throughout scanning.

Finally, to test the handle location found, the grippers would grasp the cup and rotate a random angle to put the handle in a new location. The arm would then move to a home location to prevent occluding the cup from the camera and the experiment is rerun.

VII. Conclusion

The presented techniques for fusing sensor data from the camera and in-hand sensing are simple yet effective for improving perception and grasping performance. As researchers will continue to purchase lower-cost sensors, we have shown that such sensors embedded in the hand greatly improved object pose estimation from the camera by providing a feedback mechanism to correct scene calibration errors. We also showed how proximity sensing can be used to detect irregular features and generate a 3D point cloud that can be merged with the depth camera point cloud and is compatible with widely used point cloud processing libraries.

Now that we have demonstrated the improvement from using proximity sensors that are susceptible to surface properties and sharp features of objects, we would like to continue this work using less susceptible sensors that measure time-of-flight rather than reflection. Finally, we will test the object recognition methods such as correspondence matching on the point clouds from the sensors to determine if we can design a comparable system that can perform grasping and manipulation without the need for global perception in the loop.

References

[1] R. Patel, R. Cox, B. Romero, and N. Correll, "Improving grasp performance using in-hand proximity and contact sensing," *arXiv preprint arXiv:1701.06071*, 2017.

[2] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Robust grasping under object pose uncertainty," *Autonomous Robots*, vol. 31, no. 2-3, pp. 253–268, 2011.

[3] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols," *arXiv preprint arXiv:1502.03143*, 2015.

[4] H. Dang and P. K. Allen, "Stable grasping under pose uncertainty using tactile feedback," *Autonomous Robots*, vol. 36, no. 4, pp. 309–330, 2014.

[5] J.-P. Saut, S. Ivaldi, A. Sahbani, and P. Bidaud, "Grasping objects localized from uncertain point cloud data," *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1742–1754, 2014.

[6] L. Ma, M. Ghafarianzadeh, D. Coleman, N. Correll, and G. Sibley, "Simultaneous localization, mapping, and manipulation for unsupervised object discovery," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 1344–1351.

[7] E. Luberto, Y. Wu, G. Santaera, M. Gabiccini, and A. Bicchi, "Enhancing adaptive grasping through a simple sensor-based reflex mechanism," *IEEE Robotics and Automation Letters*, 2017.

[8] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng, "Reactive grasping using optical proximity sensors," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2098–2105.

[9] S. Ye, K. Suzuki, Y. Suzuki, M. Ishikawa, and M. Shimojo, "Robust robotic grasping using ir net-structure proximity sensor to handle objects with unknown position and attitude," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3271–3278.

[10] B. Mayton, L. LeGrand, and J. R. Smith, "An electric field pretouch system for grasping and co-manipulation," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 831–838.

[11] J. Tegin and J. Wikander, "Tactile sensing in intelligent robotic manipulation–a review," *Industrial Robot: An International Journal*, vol. 32, no. 1, pp. 64–70, 2005.

[12] R. Patel and N. Correll, "Integrated force and distance sensing using elastomer-embedded commodity proximity sensors," in *Proceedings of Robotics: Science and Systems*, 2016.

[13] R. Patel, J. C. Alastuey, and N. Correll, "Improving grasp performance using inhand proximity and dynamic tactile sensing," in *Int. Symp. on Experimental Robotics (ISER)*, Tokyo, Japan, 2016.

[14] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European conference on computer vision*. Springer, 2010, pp. 356–369.

[15] A. ten Pas and R. Platt, "Using geometry to detect grasp poses in 3d point clouds," in *Intl Symp. on Robotics Research*, 2015.