Estimating the 3D center point of an object with Kinect sensor RGB-D images

1st Gustavo Fardo Armênio
Computer Engineering Course - UTFPR
Curitiba, PR, Brazil
gustavofardoarmenio@alunos.utfpr.edu.br

2nd João Alberto Fabro
PPGCA-Graduate Program on Applied Informatics
UTFPR-Federal University of Technology - Paraná
Curitiba, PR, Brazil, fabro@utfpr.edu.br

3rd Renzo de Rosa Tognella
Information System Course - UTFPR
Curitiba, PR, Brazil renzotognella@alunos.utfpr.edu.br

4th Felipe Pierre Conter
UNILA - University of Latin American Integration
Foz do Iguaçu, PR, Brazil - felipeconter.cc@gmail.com

5th Marlon Vaz de Oliveira
PhD Student at CPGEI-Graduate Program on Electrical Engineering and Industrial Informatics/UTFPR
Professor at IFPR- Federal Institute of Technology - Paraná
Pinhais, PR, Brazil - marlon.vaz@ifpr.edu.br

6th Everson de Souza Silva
PPGCA-Graduate Program on Applied Informatics
UTFPR
Curitiba, Brazil - everson@alunos.utfpr.edu.br

2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE) | 979-8-3503-1538-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/LARS/SBR/WRE59448.2023.10332949

***Abstract*——This article describes the estimation of a 3D point using a Kinect sensor and the Robot Operating System (ROS) along with You Only Look Once (YOLO) for object detection. The Kinect sensor provides RGB-D images, which are used to create a Point Cloud representing the geometry of the environment. ROS is used as a robotics development framework, while YOLO is employed to identify objects in the scene. The article presents the packages used, the datasets used for measurement, and the configuration of ROS and YOLO. Additionally, the functionalities of RViz, a 3D visualization tool used in the tests, are explored. Furthermore, it covers the methods employed, the acquired data, and an analysis of the error margin in relation to the measurement of the distance between the Kinect and the object. The findings and techniques presented in this study contribute to addressing the challenges faced in the RoboCup@Home competition, specifically in the context of object manipulation tasks.**

***Index Terms*—3D point, object manipulation, object detection, Kinect sensor, ROS, point cloud**

I. Introduction

Service robotics is a field that has been growing in recent decades with the application of mobile robots in commercial and domestic environments. The advances in Computational Processing, Artificial Intelligence, and Robotics bring a perspective of increasingly autonomous, reliable, and multifunctional service robots. However, many challenges remain open for the scientific and technological community to solve in a wide range of knowledge areas. RoboCup@Home [1] competition is an initiative that serves as a platform for developing technologies applicable to future service robots in home environments [2]. The competition focuses on tasks that simulate domestic work and encompass a wide range of human-robot interaction aspects. The primary objectives include Human-Robot Interaction and Cooperation, Navigation and Mapping in Dynamic Environments, Computer Vision and Object Recognition under Natural Light Conditions, Object Manipulation, Adaptive Behaviors, Behavior Integration, Environmental Intelligence, and System Standardization and Integration [1]. This paper specifically investigates the domain of Object Manipulation, a critical area for enabling autonomous and efficient performance of household tasks by robots. Successful object manipulation necessitates the robot's ability to perceive and comprehend its surrounding environment. An RGB-D (Red Green Blue – Depth) sensor gathers colored images and information about the distance of elements in its field of vision and can play a pivotal role in the robot's knowledge about the environment, allowing the application of several Computer Vision techniques sideby-side with the understanding of every image pixel's distance from the sensor. The use of a sensor such as Microsoft Kinect sensor [3] also facilitates the creation of a point cloud that accurately represents the three-dimensional geometry of the environment. Through this information, the robot can determine the position and orientation of objects, thus enabling the planning and execution of manipulation actions. Consequently, this paper explores the use of the Kinect sensor in conjunction with the Robotic Operating System (ROS) framework and the YOLO (You Only Look Once) [4] object detection model for object manipulation. The proposed study is motivated by the challenge approached in [5], estimating a 3D point for an object without the previous knowledge of its 3D format. The method proposed in [5] rendered promising results by

taking advantage of two estimations from different points of view but was greatly affected by the Kinect sensor's measurement errors, with a mean error of 9.4cm. To minimize inaccuracies and allow for more precise manipulation of the object, this paper tackles the evaluation of three selected methods for the calculation of the distance of an object from the camera, using Kinect RGB-D images from a single point of view, with the intent of reducing the impact of measurement errors in techniques such as the one proposed in [5]. The study encompasses various aspects regarding the techniques employed to reproduce and improve the estimation, presenting the employed methods, collected data, and an analysis of the error margin associated with Kinect's distance measurements to objects through the three distinct estimate methods: median, outlier removal and spiral filtering. This paper is divided into six sections, namely: Section II presents the relationship between 3D point estimation in space and its use in competition. Section III presents a brief overview of the hardware and software solutions used. Section IV discusses the strategies and solutions found. In section V, we discuss performance, challenges, and next steps. Finally, in Section VI, we summarize the discussion and present our final conclusions.

II. Uses in Competition and Domestic Tasks

RoboCup@Home offers various challenges for robots to perform, one of which is object manipulation and recognition [1]. To handle the competition's objects, it is necessary to identify them in the environment. The 2022 Latin-American edition rulebook outlines the following challenges [6]: the robot must identify, grasp, and correctly place multiple objects at different heights or positions and, optionally, find a hidden or occluded object; the robot must be at a random distance between 1.0m and 1.5m from a shelf, which has five levels ranging from 0.3m to 1.8m above the ground, and all objects are in their predefined locations; during the process, no people should be involved in the task. Object recognition performed by a model such as YOLOv3 [4] can identify the objects in a RGB image according to the trained dataset. The objects bounding boxes outline the region of interest of each object and allow focus at each object at a time. For example, if an orange is identified on a table the robot can try to pick it up using only the information about the orange's region of interest. The RGBD image from the Kinect can also be used to obtain depth images and determine a point close to the object's center. Considering the distance of every image pixel from the camera, simple trigonometry calculations can determine a point in 3D space for a certain pixel in relation to the camera. With the 3D point information, we can try and determine the position and distance of the object relative to the robot. Therefore, the studies presented in this article cover distances between 1.25m and 1.75m from the object, with the horizontal distance from the table where the object is located ranging from 1.0m to 1.5m, meaning the object is 25 cm from the edge of the table. These measurements are provided to give an understanding of the range for estimating the object's 3D point. For future domestic robot projects, methods for the robot to determine an object's location will be necessary. Our approach employed three methods of calculations to approximate the object's center, enabling its localization and further manipulation.

III. Hardware and Software Solutions

In this section, everything that is part of the necessary environment to test and extract the information obtained is introduced. The tools presented are directly responsible for the system's operation.

For robot integration, the Robot Operating System (ROS) provides communication between all parts of the robot. This is achieved through its embedded messaging system, which unifies communication between nodes through publishing and subscribing. This allows to reuse systems and maintain them more conveniently. Its tools facilitate the construction of applications for robots, enabling message exchange between processes and package management [6].

For the extraction of RGB-D images, a Kinect sensor is applied. Its operation consists of an infrared laser emitter and an infrared receiver (IR). The infrared projects a light that captures the deformation of this projection, allowing it to define the distance of each pixel from the cameras point of view, resulting in an image in RGB-D format, combining the color channels (red, green, and blue) with depth data [7]. However, the Kinect is vulnerable to noise and external lighting, what can usually lead to discrepancies in the object's centroid 3D point. The freenect_stack [8] driver for Kinect provides an interface between the sensor and the ROS ambient, publishing the RGB and depth images, as well as point clouds in ROS topics. The specific process "depth_registration" of the driver, if set as true when launched, performs the matching of the RGB-D images and the point cloud for overlapping their information, and is key to an accurate estimation.

In terms of object recognition software, the application of the YOLOv3 model provides real-time object detection, helping to distinguish between objects desired for testing [4]. YOLO is based on two-stage Convolutional Neural Networks (CNN), which follows the pattern of first generating bounding boxes and then performing object classification and predicting their location information. One-stage detection is a single-stage inference from input to output, meaning everything happens in this single stage, making it faster but with a higher chance of imprecision [2].

The darknet_ros package provides an integration of the YOLO model with ROS topics and messages. The package adapts the inputs and outputs of the original model implementation to the ROS data communication format [9]. Using ROS tools, it is easy to use sensor data like the Kinect's camera to pass RGB images to the darknet_ros node, so it can perform the inferencing of the detected objects and output the information regarding them, like bounding boxes, detected classes and percentages of certainty.

Available at https://github.com/ros-drivers/freenect_stack
Available at https://github.com/leggedrobotics/darknet_ros

This way, it is possible to combine Kinect RGB-D data to detect objects and perform calculations to extract 3D points from an object, being the center point of the object the focus of the study. Two custom ROS nodes were built to accomplish the object segmentation through its bounding box and estimate the 3D centroid of the segmented object using the Kinect depth image.

For visualization of the point cloud and images obtained by the Kinect, detected bounding boxes and the calculated 3D center point, RViz [10] was a powerful tool. RViz enables real-time visualization of topics from various ROS data types.

A. *Strategies and Solutions*

First, the basic conditions set for the experiments are addressed. Several configurations were tested for the evaluation of methods across different distances from the object, different objects and different datasets for training. Two distances were used: 1.5m and 2.0m horizontally from the wall to the camera, with 0.25m from the wall to the object, resulting in 1.25m and 1.75m from the camera to the object in the horizontal axis. As for the heights, 1.22m from the ground to the camera and 0.83m to the surface where the object is located on. Two sets of weights trained on two distinct datasets were used for object recognition with YOLO: the pre-trained COCO weights provided by YOLOv3, and the weights generated on a custom dataset from LARC 2022. COCO contains around 320.000 images for 80 classes of objects [11]. Classification inference examples on COCO dataset can be seen in Figures 1 and 2.

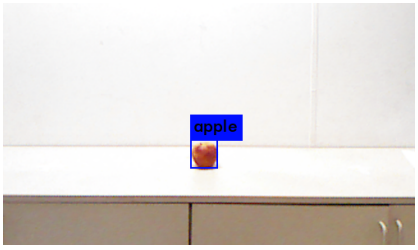

Fig. 1. YOLOv3 classification of an apple in the COCO training.

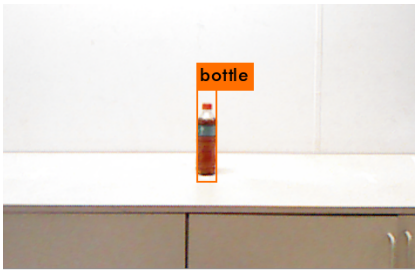

Fig. 2. YOLOv3 classification of an Iced Tea bottle in the COCO training.

Available at https://github.com/UtBotsAtHomeUTFPR/utbots_vision/blob/master/vision_tools/src/detected_person_manager.py

Available at https://github.com/UtBotsAtHomeUTFPR/utbots_vision/blob/master/vision_tools/src/extract_3d_centroid.py

The LARC2022 dataset is made of approximately 200 images that contain several classes of objects in each image, including occluded ones, from 20 classes. A small number of classes was selected from the intersection of the two datasets, since both they have a big number of objects in common, aiming for future tests with object manipulation. Four objects were selected: an apple, an orange, an iced tea bottle and a soda can. Classification inference examples on LARC2022 dataset can be seen in Figures 3 and 4.

Fig. 3. YOLOv3 classification of an iced tea bottle in the LARC2022 training.

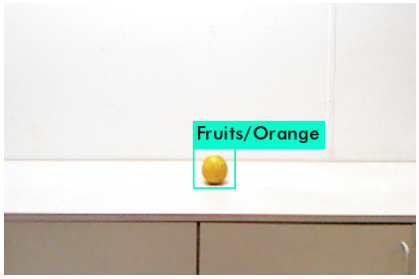

Fig. 4. YOLOv3 classification of an orange in the LARC2022 training.

Considering they all have different geometries, their center point is located at different heights; the orange and the apple have their center at 0.87m from the ground, the Iced Tea bottle at 0.93m and the soda can at 0.89m. The results obtained from these two training weights were then compared.

A depth image in a complex environment has a large range of distances from the camera for all points, therefore it is a smart move to reduce as much as possible the points that do not belong to the detected object of interest. Based on that idea, object segmentation was applied from the objects' bounding boxes, basically cropping a selected objects' region of interest from the depth image in a ROS custom node. The RGB and depth image do not necessarily overlap correctly, because the field of view of the RGB and IR cameras are not precisely the same. To adequately match the RGB and depth images, the "depth_registration" provided by freenect_stack driver was essential.

Having reduced the number of points evaluated to the bounding box pixels, some irrelevant pixels can remain, because the bounding box is not usually the exact silhouette of the object and is highly sensible to the manual labeling of the dataset. Therefore, three different methods were tested to calculate the object's centroid.

The first and simplest method consists of calculating the median of the object's pixels. However, it is possible that some pixels do not correspond to the object. Therefore, we used two additional methods to evaluate this issue.

The next method involves removing outlier values, meaning it analyzes all the points and identifies those that likely do not belong to the object. This is done using quartiles (percentiles) of the depth values found, to determine upper and lower limits that define an interquartile range. This interquartile range is calculated from the difference between the third quartile (Q3), corresponding to the 75th percentile, and the first quartile (Q1), corresponding to the 25th percentile. Values within the limits of 1,5 times the interquartile above Q3 and below Q1 are used to calculate the average distance of these points. Below, Algorithm 1 and Algorithm 2 applied for outlier removal are displayed.

Algorithm 1 CalculatePercentile

Require: $array, p$ {requires the values array and the percentile value}
  $array \leftarrow array.sort()$
  $n \leftarrow array.size()$
  $lowerIndex \leftarrow \lfloor \frac{p}{100} \times (n-1) \rfloor$
  $upperIndex \leftarrow \lceil \frac{p}{100} \times (n-1) \rceil$
  $lowerValue \leftarrow array[lowerIndex]$
  $upperValue \leftarrow array[upperIndex]$
  $percentile \leftarrow lowerValue + (upperValue - lowerValue)$
  return $percentile$

Algorithm 2 RemoveOutliers

Require: $pixelDistances$ {requires the array containing distances for every pixel}
Ensure: $pixelDistances.size() > 0$
  $q3 \leftarrow CalculatePercentile(pixelDistances, 75)$
  $q1 \leftarrow CalculatePercentile(pixelDistances, 25)$
  $interquartile \leftarrow q3 - q1$
  $max \leftarrow q3 + (1.5 \times interquartile)$
  $min \leftarrow q1 - (1.5 \times interquartile)$
  $filteredPixels \leftarrow []$
  for $pixel \in pixelDistances$ **do**
    if $pixel > min$ and $pixel < max$ **then**
      $filteredPixels.insert(pixel)$
    end if
  end for
  return $filteredPixels.mean()$

The last calculation method consists of an algorithm that spirals through the image from the central point and compares each pixel with the filtered mean of the previous pixels. If the difference between the current value and the filtered mean is less than or equal to the defined threshold, the pixel is considered valid and added to the filtered pixel list. Following, Algorithm 3 shows the calculations used for the last discussed method.

Algorithm 3 SpiralFiltering

Require: $pixelMatrix, threshold$ {requires the matrix containing the distance for every pixel in the image and the threshold value for distance filtering}
Ensure: $pixelMatrix.size() > 0$
  $h \leftarrow pixelMatrix.height()$ {height}
  $w \leftarrow pixelMatrix.width()$ {width}
  $hH \leftarrow \lfloor h/2 \rfloor$ {half height}
  $hW \leftarrow \lfloor w/2 \rfloor$ {half width}
  $dx \leftarrow 1$ {initial direction in the horizontal axis is forward}
  $dy \leftarrow 0$ {initial direction is zero, it does not move in the vertical axis}
  $x \leftarrow hW$
  $y \leftarrow hH$
  $filtPixels \leftarrow []$
  for $pixel \leftarrow 0$ while $pixel < (max(w,h))^2$ **do**
    if $0 < x < w$ and $0 < y < h$ **then**
      if $x = hW$ and $y = hH$ **then**
        $filtPixels.insert(pixelMatrix[y][x])$
        $filtMean \leftarrow pixelMatrix[y][x]$
      else if $|pixelMatrix[y][x] - filtMean| \leq threshold$ **then**
        $filtPixels.insert(pixelMatrix[y][x])$
        $filtMean \leftarrow filtMean + \frac{(pixelMatrix[y][x] - filtMean)}{filtPixels.size()}$ {moving average}
      end if
    end if
    $cx \leftarrow x - hW$ {centered horizontal}
    $cy \leftarrow y - hH$ {centered vertical}
    if $cx = cy$ or ($cx == -cy$ and $cx < 0$) or ($cx == -(1 + cy)$ and $cx \geq 0$) **then**
      $dx, dy \leftarrow dy, -dx$
    end if
    $x \leftarrow x + dx$
    $y \leftarrow y + dy$
  end for
  return $filtMean$

Having estimated the distance of the object's center from the camera, applying trigonometry considering the horizontal and vertical aperture (57 degrees and 43 degrees, respectively, for the Kinect) of the camera makes it possible to estimate a 3D point. First, a simple remapping of the point's x and y coordinates in the image is necessary to consider the center of the camera as the origin of the Cartesian coordinate representation, subtracting half the width and half the height of x and z, respectively. Considering the center point as origin, the maximum angle of each corner of the image is half the aperture in the respective axis, being negative at the negative segments of the axis. Figure 5 presents the important angles and measures for this calculation in 3D space perspective. It is possible then to reach the Equations 1, 2, 3, 4 and 5, in which ρ is the estimated distance:

$$\theta_{max} = \frac{57}{2} = 28.5^\circ \quad \phi_{max} = \frac{43}{2} = 22.5^\circ \tag{1}$$

$$\theta = \theta_{max} \times \frac{x - \frac{width}{2}}{\frac{width}{2}} \qquad \phi = \phi_{max} \times \frac{y - \frac{height}{2}}{\frac{height}{2}} \tag{2}$$

$$x_{3D} = \rho \times \cos\phi \times \sin\theta \tag{3}$$

$$y_{3D} = \rho \times \cos\phi \times \cos\theta \tag{4}$$

$$z_{3D} = \rho \times \sin\phi \tag{5}$$

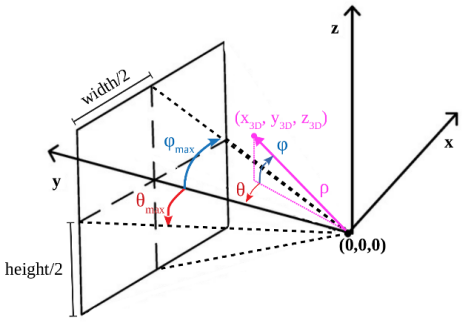

Fig. 5. Camera FOV and point angles in the cartesian 3D coordinate system.

IV. Results and Margins of Error of the Study

This section will demonstrate the obtained results. Figures 6, 7, 8 and 9 display some resulting estimated 3D points from the point cloud perspective.

Tables I and II contain the found values and their respective 3D points, considering the actual distances of the objects and those calculated by the algorithm by each method - median, outlier removal and spiral filtering (with a threshold of 10cm) - and configuration of experiment. Notably, discrepancies arise, particularly for iced tea and soda. These variations stem from lighting conditions impacting point cloud generation. For instance, aluminum cans may reflect ambient light, and iced tea's translucency allows infrared rays to penetrate its transparent container. The filtering method excels by selectively processing pixels through a defined threshold and moving average, effectively removing outliers and noise. However, it necessitates precise pixel alignment with the object's bounding box center and threshold specification. The outlier removal method, while effective in eliminating discrepant values, is less potent than filtering, particularly with translucent objects, where the values may still fall within an acceptable range.

Table III shows the margin of error for each obtained result in centimeters, following the sequence of samples displayed in Tables I and II.

As can be confirmed, the filtering method was the most consistent in its measurements, with little variation among them, a mean absolute error of 4.256cm, and a maximum absolute error of 7.6cm. Therefore, the spiral filtering presents a comparable result to the results of [5] (9.4cm mean error).

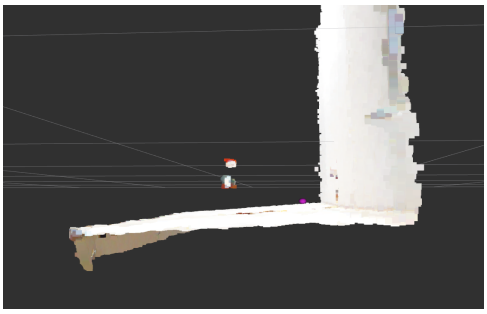

Fig. 6. Side view of the estimated center point of the Iced Tea bottle (seen in pink) from a point cloud 3D perspective.

Fig. 7. Front view of the estimated center point of the Iced Tea bottle (seen in pink) from a point cloud 3D perspective.

Fig. 8. Side view of the estimated center point of the apple (seen in pink) from a point cloud 3D perspective.

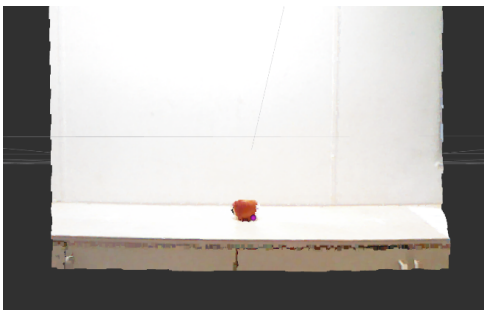

Fig. 9. Front view of the estimated center point of the apple (seen in pink) from a point cloud 3D perspective.

TABLE I
COCO TRAINING RESULTS WITH MEDIAN, REMOVE OUTLIERS AND SPIRAL FILTERING (THRESHOLD OF 10CM), IN METERS

Object	Real Distance		Median		Remove Outliers		Spiral filtering	
	1.5m	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*
Apple	1.278m	1.780m	1.21m	1.72m	1.21m	1.82m	1.22m	1.71m
Orange	1.278m	1.780m	1.21m	1.72m	1.245±0.035m	1.82m	1.22m	1.71m
Iced Tea	1.260m	1.765m	1.34±0.14m	2.00m	1.35±0.02m	1.91m	1.23m	1.71m
Soda	1.272m	1.776m	1.22m	1.71m	1.21m	1.81m	1.22m	1.70m

TABLE II
LARC2022 TRAINING RESULTS WITH MEDIAN, REMOVE OUTLIERS AND SPIRAL FILTERING (THRESHOLD OF 10CM), IN METERS

Object	Real Distance		Median		Remove Outliers		Spiral filtering	
	1.5m	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*
Apple	1.278m	1.780m	1.23m	1.99m	1.31m	1.86m	1.22m	1.725±0.005m
Orange	1.278m	1.780m	1.22m	1.815±0.025m	1.32m	1.85m	1.22m	1.72m
Iced Tea	1.260m	1.765m	1.49m	2.00m	1.37m	1.965±0.035m	1.22m	1.715±0.005m
Soda	1.272m	1.776m	1.335±0.015m	1.80±0.04m	1.35m	1.85m	1.23m	1.72m

TABLE III
COCO AND LARC2022 TRAINING DISTANCE ESTIMATION ERRORS WITH MEDIAN, REMOVE OUTLIERS AND SPIRAL FILTERING, IN CENTIMETERS

Object	COCO training						LARC2022 training					
	Median		Remove Outliers		Spiral filtering		Median		Remove Outliers		Spiral filtering	
	1.5m	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*	*1.5m*	*2.0m*
Apple	-6.9cm	-6.1cm	-6.9cm	3.9cm	-5.9cm	-7.1cm	-4.9cm	20.9cm	3.1cm	7.9cm	-5.9cm	-5.5±0.5cm
Orange	-6.9cm	-6.1cm	-3.3±3.5cm	3.9cm	-5.9cm	-7.1cm	-5.9cm	3.5±2.5cm	4.1cm	6.9cm	-5.9cm	-6.0cm
Iced Tea	8±14cm	23.4cm	9±2cm	14.4cm	-3.0cm	-5.6cm	23.0cm	23.4cm	11.0cm	20±3.5cm	-4.0cm	-6.5±0.5cm
Soda	-5.3cm	-6.6cm	-6.3cm	3.4cm	-5.3cm	-7.6cm	6.3±1.5cm	2.4±4cm	7.7cm	7.4cm	-4.3cm	-5.6cm

V. NEXT STEPS

In the setup of this study's experiments, the spiral filtering method outperformed the estimations of [5]. In order to compare the approaches with more data, the next step will be to evaluate its performance in similar conditions to [5], and evaluate the performance of the application of two estimations from distinct points of view using the spiral filtering to improve the results. Additionally, it is relevant to conduct tests with a physical or simulated robotic arm, to verify the performance in a practical application. Lastly, it is intended to test the challenges and rules proposed in the RoboCup@Home, to be able to perform more tasks in the environment.

VI. FINAL CONSIDERATIONS

Based on the data analysis, the filtering method stands out as the most promising approach for further exploration. It's feasible to establish a threshold, given the object's geometry and its placement surfaces, and assume that the bounding box center contains an object pixel. The median method yielded the poorest results. The outlier removal approach showed promise and even outperformed filtering in some aspects, but it lacked the same level of consistency. A potential improvement lies in reevaluating the outlier boundaries. To refine these algorithms, additional tests will be conducted with more objects. In the context of the competition, the goal is to extract essential information from these methods to precisely determine the robot arm's positioning for object grasping.

VII. ACKNOWLEDGMENT

The authors would like to thank UTFPR, Federal University of Technology - Paraná, for the financial support.

REFERENCES

[1] Robocup, "Robocup@Home," https://athome.robocup.org/(accessed Jun. 19, 2023).
[2] T. Wisspeintner, T. Van Der Zant, L. Iocchi, S. Schiffer, "RoboCup@Home: scientific competition and bench-marking for domestic service robots",Interaction Studies.Vol.10(3),pp.393–428,2009.
[3] Microsoft, "Kinect Sensor,". msdn.microsoft.com. http://msdn.microsoft.com/en-us/library/hh438998.aspx (accessed Jun. 19, 2023).
[4] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement", arXiv preprint arXiv:1804.02767, 2018.
[5] F. P. Conter, "Object Detection and 3D Position Estimation through the Application of Convolutional Neural Networks" (in portuguese). 2022. Dissertation (Master's in Applied Computing) - Federal University of Technology - Paraná, Curitiba, 2022.
[6] F. Pimentel, A. P. Magalhaes, K. Kappel, J. Dyonisio, K. H. Guimarães, "Robocup@Home Rulebook – LARC", https://github.com/RoboCupAtmeLatinAmerica/RuleBook/blob/2023_draft/build/Rulebook.pdf (accessed Jun. 19, 2023).
[7] L. Cruz, D. Lucio and L. Velho, "Kinect and RGBD Images: Challenges and Applications," 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials, Ouro Preto, Brazil, 2012, pp. 36-49.
[8] P. Khandelwal, J. O'Quin, "freenect_stack" , http://wiki.ros.org/freenect_stack (accessed Jun. 19, 2023).
[9] M. Bjelonic, "YOLO ROS: Real-Time Object Detection for ROS", https://github.com/leggedrobotics/darknet_ros (accessed Jun. 19, 2023).
[10] D. Hershberger, D. Gossow, J. Faust, W. Woodwall, R. Haschke. "Rviz" http://wiki.ros.org/rviz (accessed Jun. 19, 2023).
[11] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. Springer International Publishing, 2014.