Creating a Database of Objects and Their Positions in a Large-scale Indoor Environment

Suhas Dara, Sadhvi Darisipudi, Lucinda Nguyen, and Tracy Zhang
College of Natural Sciences
The University of Texas at Austin
Austin, Texas 78705

Abstract—**We programmed the BWI Bot to roam around the GDC and use its camera to record and create a database of recognizable objects and their respective positions in the building. We successfully utilized YOLO for object recognition and then stored the position of each object based on the GDC's preexisting map of the current floor. For movement, we built upon a preexisting functional program in the BWI repository to create the roaming and swivel movement to assist visualization. We tested the robot by making it traverse the lab by itself, capturing the different objects it sees along the way, and finally, we analyzed the results in the output database file.**

I. Introduction

Mobile robots today are expected to complete a variety of tasks. Many of these involve accessing and utilizing specific objects in a large environment. In order for this to be a feasible task, the robot must have the ability to either search for these objects or have a way of knowing where these objects are already located. Because many indoor robots tend to operate within a single environment, it is plausible to have the robot build up a database or list of objects and their respective locations within this environment autonomously. Using this information collected over time, the robot would be able to quickly recognize where an object is located and go to this location. This, as a result, would simplify the process of obtaining an object and allow for other tasks to be completed quicker and with relative ease. Our goal is to write a functional program that allows the robot to autonomously traverse the vast majority of an indoor environment and to collect and store information about objects within the environment and where they exist.

This project not only solves a problem on its own, but helps other tasks be completed quicker. For example, a task the robot may be programmed to do is to retrieve a certain object. In order to do this, the robot must either search for the object or using this approach, go to the location it has previously been known to be. Using this second approach not only saves time but energy as well because the robot will not have to physically maneuver through the environment looking at every possible location for where this object may be. Furthermore, using this approach reduces the amount of work the human operating the robot would have to do. Rather than having the user remember where something is located, the robot would be able to autonomously collect and store information about objects and their locations to access when necessary. This will be especially applicable in the context of home robots or other assistive robots in indoor environments. While this is not currently a common reality, with the introduction of non-mobile home robots such as Amazon's Alexa and Google Home among many others, mobile home robots that use object detection are not a distant future.

II. Background

Currently, for object recognition using computer vision, there are two primary approaches. This includes appearance-based and feature-based methods. Appearance-based methods make comparisons with an example or template image while feature-based models implement a search for visual matches. Many papers utilizing computer vision for object identification and database creation use a variety of methods in both areas such as a combination of recognition by parts, edge matching, divide-and-conquer search, gradient matching, pose consistency, histogram comparisons, geometric hashing, Scale-invariant feature transform (SIFT), Speeded Up Robust Features (SURF) and more [1]. These papers often discuss a specific combination of such techniques along with different softwares to find objects. Popular softwares for object detection include OpenCV, Deepdream, DeepPy, MATLAB, Halcon, among many others. Although the methods for object recognition may be varied in technique and softwares used, most papers demonstrate that they create databases of objects in similar way [2]. They all implement some type of a data structure to record objects once they are found and other relevant data pertaining to that specific object [3]. To expand upon this database further, many papers cite how they then use the database of images and objects they created for machine learning in order to increase the accuracy of the object detection [4] [5]. For example the creators of YOLO use images to train its model for accuracy in object detection and recognition [6].

Many other robots exist with functionality similar to what we hope to implement on the BWI Bot. For example, the Home Exploring Robotic Bulter (HERB) robot from Carnegie Mellon University (CMU) is a mobile robot that can complete a variety of household and domestic tasks. One of HERB's capabilities that parallels our project includes storing objects and their locations. HERBs map consists of two parts: a static, original map, and several lists of map objects with locations and orientations on the map. The list maintains the position and orientation of each object, as well as the history of object sightings which is very similar to our approach [7]. However, in our approach, we do not save a full history of object settings, instead overriding the output database file with new objects and their respective positions every time it is rerun [8].

Another robot is Bossa Nova which is currently being implemented in Walmart stores across the nation. It traverses the store aisles taking pictures and recording item inventories [9]. It can discern missing, broken, or misplaced items on the shelves and replace or return them to the correct locations. Because Bossa Nova is a retail or commercial robot and is not used for academic or research purposes little to no information can be found on its software implementation, object recognition, and location storage. However, the functioning of the robot is similar to what we have implemented as a part of this project. So while other existing projects similar to ours exist, they are different in terms of more detailed specific functioning.

III. Method

This project has two components to it, object detection and movement of the BWI Bot. For object detection, the initial approach considered was to use the Movidius Neural Compute Stick (NCS) which implements a version of real-time object detection software called tiny-YOLO. This approach was chosen due to its ability to process incoming image frames quickly on a GPU. However, due to technical difficulties getting the Movidius NCS to work, our team was unable to carry forward with this approach. The next approach chosen was to use YOLO itself which is also known as Darknet-ROS. YOLO does not require a GPU to run, but it is significantly slower than the Movidius NCS. However, YOLO pertains very well with our project as it is very accessible and usable as an open-source unlicensed software. Furthermore, many tutorials exist online teaching both the basics and specifics of implementing it.

With YOLO, image processing on the BWI Bot will be relatively slow as the BWI Bot must analyze each frame using the Microsoft Xbox Kinectv2 (the camera) for recognizable objects without a powerful GPU. YOLO offers detection of eighty different classes of objects through pre-trained models, including but not limited to people, cars, bottles, chairs. Our project focuses on stationary objects as we only want to store objects that probably will remain in their positions in the near future, thus excluding classes such as people, and cars. This limits to fifty-eight different classes that will be detected. YOLO divides the image into several regions and weighs the calculated probabilities in each region. YOLO then publishes all the objects detected in one image frame, their confidence levels (recorded as decimals between 0.0 and 1.0), and their position (bounding box) in the image frame in pixels, which will be used to get their position with respect to the map of the floor.

Our object detector is able to publish a topic with positions of all the objects it has found until now, along with a database of all these positions in the form of a text file. To accomplish this, our object detector subscribes to the YOLO publisher, which publishes bounding boxes of objects that it detects in the current frame. To find the depth of the object from this image frame, we also subscribe to the point cloud data of the camera, which provides us the extrapolation of a 2D pixel into 3D. Once we have the position of the object with respect to the camera, we try to restrict the positions that make it to the database which is represented by an output file with all the positions detected so far, saving the data points in a vector. We do not want positions that are too similar to positions detected in previous image frames. This usually denotes a repeated detection of the same object. Hence, we enforced a restriction that the current object's position is within ten centimeters in all directions of any other object previously detected, it will be excluded from our database. YOLO's pre-trained models do not always provide the best predictions, and incorrectly identify one object as another. We used the confidence level of each bounding box as a threshold to limit objects that make it to the database. This threshold, which we set to forty-five percent, allows only objects that were detected with a higher confidence level than forty-five percent to enter the database. Once we have an object which matches all the criteria, we transform the object's position (which is currently in the camera's frame of reference) to the map's frame of reference.

The second part of the project, as mentioned previously is the movement of the robot to cover larger ground and find objects across the floor instead of at a stationary place. Due to time constraints, in order to implement the roaming movement, we utilized and altered a preexisting program from the BWI Bot repository. We used a program from an older project called ”visit_door_list” that visits a few different doors in a cycle motion. We chose this in particular because it allows the BWI Bot to roam the entirety of the AI Lab on GDC 3rd floor. To visualize this movement we utilized RVIZ (see fig. 2). We altered the functionality of this program to interrupt the BWI Bot for seven to fifteen seconds while it is pursuing a navigation goal, before it resumes its trajectory toward the goal again. We opted to interrupt the robot because of YOLO's limited computation power making it lag a few frames behind when not having access to a GPU.

IV. Evaluation

To test our system, we first created a rosbag input stream, which recorded different objects in the lab using the camera. The rosbag video was created to contain clear images of objects that YOLO recognizes such as cell phone, mouse, bottle, etc. (see Fig. 1). We ran YOLO's object detection software concurrently with the rosbag to check if objects were detected fairly accurately and to ensure that YOLO was setup accurately. Upon achieving desired results, we tested YOLO on live feed from the robot, to decide on the duration of the interrupt for movement.

Prior to any testing on the robot, we used ROS Visualizer (RVIZ) to localize the BWI Bot to avoid collisions and for it to record its initial position and for it to calculate and store the positions of all the objects it finds. For testing whether

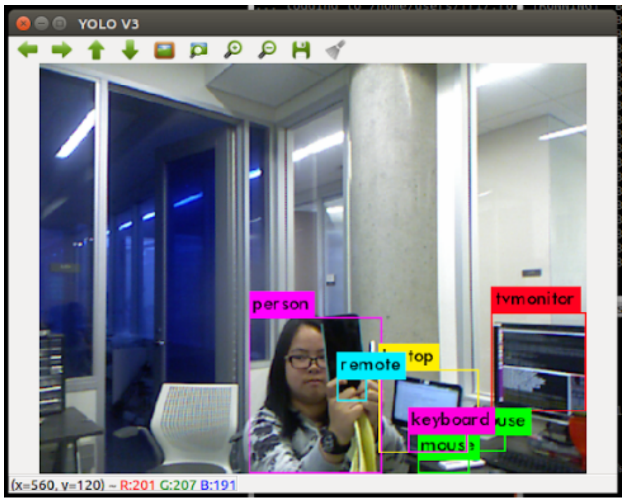

Fig. 1. An example of YOLO working with input from the Microsoft Xbox Kinect in a indoor setting. It recognizes most objects from its databases and then publishes bounding boxes around them.

our object detector was accurately publishing the positions, we created a topic listener that helps us utilize RVIZ to view all the positions of different objects on the map. This topic listener was dynamic as it ran at the same time as the publisher. The robot did not have enough processing power to handle the YOLO publisher, our publisher and our listener together causing the robot's laptop (command line interface) to lag to an unworkable extent. This is when we realized that we need a static listener that was able to access a database of all positions, and added a feature of writing all the positions to a text file. This static listener can be run independently of our publisher, but can only visualize objects in RVIZ that were recorded in the previous run of the publisher.

For our final testing on a small sample, we used the second listener along with our publisher and movement to test whether our position data was accurately visualized in RVIZ. The positions were published accurately on the map whenever the robot was correctly localized on the map, although the label given to the object was not always correct, which is a limitation of YOLO itself and not our object detector.

V. Results

After our final testing, we let the robot do an entire round of the lab by itself, capturing the different objects it sees along the way. We made the best attempt of localizing the robot perfectly before running our entire object detector setup. This allowed the robot to give relatively accurate positions of each object. YOLO publishes new bounding boxes in the latest frame at a rate of about 0.3 to 0.5 FPS, giving us access to new data every 2 to 3 seconds. During this period, the object detector would evaluate the positions of all the new objects found in the last dataset received from YOLO, and write them into an out file that would allow our listener (second) to later grab the positions data and visualize them accurately in RVIZ (example shown in Table 1 and Fig 2).

Table 1 lists a few of the objects detected as the robot was roaming around. The robot identified some things that did not actually exist at the location, such as recognizing the carpet as a bed. This illustrates the limitations of YOLO's object detection, even with the forty-five percent threshold we chose for the confidence level. However, we were able to accurately detect the refrigerator in the kitchen of the BWI lab. When we compare this to the position of a nearby detected chair in the lounge, we can see that the differences in their x and y position (11.2 and 6.8) represented the Euclidean distance between the chair and refrigerator.

TABLE I
Objects and Their Respective Locations

Object Class	X-coordinate	Y-coordinate	Z-coordinate
Oven	-44.7483	-3.09647	0.387882
Chair	-32.7629	-3.90937	0.449987
Refrigerator	-40.8273	-10.6016	0.176422
Chair	-29.6026	-3.82291	0.447743
Bed	-32.0595.	-12.42	-0.0363596
Cup	-13.4071	-10.1501	-0.00217486
Etc.	Etc.	Etc.	Etc.

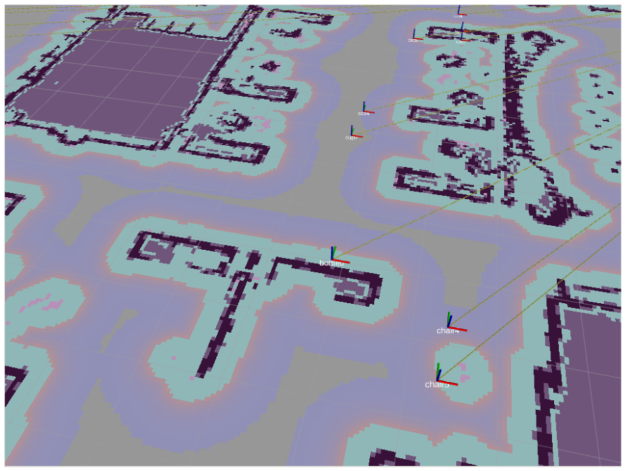

Fig. 2. The TF frames of the objects detected viewed in RVIZ.

VI. Discussion

This project is important because it helps robots complete other tasks more quickly and efficiently. For example, a task a robot may be programmed to do is to retrieve or grasp a certain object. Thus, the robot must either search for the object or, using our approach, go the location it had been previously seen at. Storing objects and their locations saves both time and energy because the robot will not have to move through the environment, roaming and searching for the object dynamically. Additionally, using this approach reduces the amount of work the human operating the robot would have to do since the robot is autonomously collecting and storing information about objects it sees once it is running. This approach will hopefully make assistive robots more helpful in many different real-world, indoor settings.

One example of an application of this project in the context of a home environment is finding lost objects. Even if the robot's human user cannot remember where an object was last seen or where it is located, the robot would be able to keep a detailed and updated log on the location of the object. In this scenario, the user could input the name of an object and the robot would be able to access the object's last recorded location which would make finding lost items easier.

While our object detector, accomplished many of its goals, there are limitations to our approach. For example, the movement of the BWI Bot relies on existing behaviors, which are not optimized to the extent desired. For example, we rely on the fact that after the robot has gone into sleep, it will have to re-localize itself and as a result will swivel or rotate. It cannot be predicted, however, how long the robot will swivel or in what direction. A future project could pursue making the movement aspect of this project better.

Our object detector also has limitations in the dataset. Because we store a database of poses for each object and continue to add poses to it, if the object were to be moved, our object detector would not recognize that there is not an object in a pose that we previously stated there was; it would simply add more poses to the database of where the object has been moved to. This means that our object detector will have to reset it's database regularly in order to update the position of the objects in the environment.

Although this approach provides a working method to solve the problem at hand, it has several limitations in its ability to do so. One limitation in our object recognition is that YOLOs object recognition algorithm is not perfect, and may not always recognize an object. Furthermore, a complete search of the room to collect the positions of object can be difficult. For example, if the robot is done roaming and collecting data, the user has to manually stop the robot. This is because of its inability to distinguish whether an area has been visited yet or if data for certain objects have already been collected. Another limitation of our approach is that roaming in a semi-random manner through visiting every door in the AI lab not be an effective way to search as there is no clear strategy involved.

Other limitations entail our implementation of the Object Detector. When it writes new positions found into a datafile, it overrides the existing datafile. This is either a limitation or an advantage depending on the circumstances under which the detector is being used. If we want a latest representation of the lab, this is an advantage as the memory is automatically re-allocated to the new data, but if we want to update the original database, this approach will not work.

One architectural limitation of the BWI Bot is that after multiple circuits of the lab, the localization of the robot is not exactly the same as when it started its first circuit, which may lead to duplicate positions that are slightly different from each other but actually represent the same object. Also, after all these multiple circuits we have to end the object detector's runtime manually. With the time constraint, the program controlling movement could not be altered enough to take this into consideration. Eventually though we hope to improve upon these limitations to make this project more easily functional and comprehensive.

VII. Conclusion

After scouting and researching object detection software options, we chose YOLO for our project, and utilized it to recognize and collect data for the positions of objects, while the BWI Bot roams around the AI Lab. The program controlling movement is built on existing BWI repository that makes the BWI Bot visit different doors and rooms in the lab, altering it to make the robot stop every few seconds for a specific duration of time to process images and store the objects it sees. The entire setup was tested on the BWI Bot in the AI Lab and was observed without any physical interference. After the BWI Bot completed a circuit of the lab, we stopped it manually and analyzed the database accumulated, while visualizing it in RVIZ.

Our project is significant because in an increasingly technologically advanced world, robots are expected to easily complete a variety of tasks, some of which require object detection and recognition within a real world setting. Further advancements in this general direction will lead to the development of a variety of assistive robots in different settings. As such mobile robots in all sorts of different environments are becoming more like reality and less of a distant future.

While this is our current approach to the project given the constraints of time and the robots processing capabilities without the Movidius Neural Compute Stick, this project has potential to be extended with new goals, including but not limited to better movement capabilities, better efficiency and using the object detector to return to a previously detected object's position.

Acknowledgment

The authors of this paper would like to thank Dr. Justin Hart for guidance along this project. They would also like to knowledge the lab mentors and teaching assistants for their guidance, support, and advice. Finally, a special thanks goes to lab mentor Jamin Goo for helping them outside of his office hours.

References

[1] Shaikh, Soharab Hossain, Khalid Saeed, and Nabendu Chaki. *"Moving Object Detection Approaches, Challenges and Object Tracking." Moving Object Detection Using Background Subtraction SpringerBriefs in Computer Science,* 2014, 5-14.

[2] Rahesh Mohan and Rakamant Nevatia *"Perceptual organization for scene segmentation and description"* IEEE Trans Pat Anal Mach Intell. 1992.

[3] D.G. Lowe *"Distinctive image features from scale-invariant keypoints"* International Journal of Computer Vision.

[4] Tony Lindeberg *"Scale invariant feature transform"* 2012: Scholarpedia.

[5] Herbert Bay*"Speeded-Up Robust Features (SURF)". Computer Vision and Image Understanding.* 2008: Harvard University Press.

[6] Redmon, Joseph, and Ali Farhadi. *"YOLOv3: An Incremental Improvement."* ArXiv, 2018.

[7] Srinivasa, Siddhartha S., Dave Ferguson, Casey J. Helfrich, Dmitry Berenson, Alvaro Collet, Rosen Diankov, Garratt Gallagher, Geoffrey Hollinger, James Kuffner, and Michael Vande Weghe. *"HERB: A Home Exploring Robotic Butler." Autonomous Robots 28, no. 1 (2009): 5-20. doi:10.1007/s10514-009-9160-9.*

[8] Gallagher, Garratt, Siddhartha S. Srinivasa, J. Andrew Bagnell, and Dave Ferguson. *"GATMO: A Generalized Approach to Tracking Movable Objects." 2009 IEEE International Conference on Robotics and Automation, 2009. doi:10.1109/robot.2009.5152863.*

[9] "Bossa Nova Robotics." Bossa Nova Robotics. Accessed May 10, 2018. http://www.bossanova.com/.