Hide and Seek: Object Detection using Lawn Mowing Algorithm and YOLO

Lara Flynn, Jennifer Suriadinata, Eric Tang, Tara Wijaya
Department of Computer Science
University of Texas at Austin
Austin, TX

***Abstract*—We designed a program which uses the You Only Look Once (YOLO) object detection system to make a robot find a specific object inside a room as well as a movement algorithm to move around the room efficiently. This was tested on the robots constructed for the University of Texas Building-Wide Intelligence project. These robots are built with a Xbox 360 Kinect camera for their vision and have a segway base to control their movement. We found that a simple rotation at the center of the room detected the object in some cases, but a more complex movement needed to be taken in order to find the object when it is in certain positions and orientations.**

I. Introduction

Object detection is an important robotic component for performing tasks such as following a person or identifying objects. Manipulating a robot to find an object is a complex task that requires integration of autonomous movement as well as object recognition. The robot would need to scout every section of the area at the same time as recognizing the objects within its vision. Developing an object-finding robot is an interesting and challenging research problem that is similar to the general exploration problem. The general idea between exploration and object finding is similar in the fact that the robot must explore the area until it finishes its goal, which differs for the two problems. The goal of exploration is to explore the entirety of an unknown area in order to build a map, while the goal of object finding is to explore an area until it acquires the location of the target object. With this, the robot does not necessarily have to explore the whole region to find an object, and will stop immediately upon detecting the target object.

The process of locating objects based on the position the robot detects it at is important in the sense that it will be able to help people find objects that are lost and take a step in the direction towards simply asking robots to obtain objects. This utilization of the robot can be very practical for a normal household robot. Another application to the problem could be for a search and rescue mission or finding objects inside a large warehouse. This can become very applicable for a more expansive search and rescue to allow for less stress for those who are on the mission as there will be a larger team available to search. If the robot's detection surpasses the human's, then having the robot find the object may be more efficient than searching for it oneself. The possible danger and fatigue from that comes with searching is eliminated.

Another challenge to the problem is developing an algorithm that would allow the robot to explore an area and find the object with the least amount of time possible. Although the robot is able to roam around an area aimlessly until it locates the object, it would be inefficient if this took place in a larger area, as well as if there were a time constraint. Instead we decided to find a more effective algorithm that will allow the robot to find the object in a more timely manner. The goal of such an algorithm is to ensure that the robot has seen each of a given set of points in the room from many different angles.

Because the world is always changing and objects do not stay in one place all the time, the robot needs to be able to adapt to the movement of objects and be able to locate them to the best of their abilities without the use of hard coded locations of objects within the robot's memory. By using autonomous object detection through the robot, the robot will be able to adapt to the what it sees as the world even as the current state of the world differs from previous ones.

II. Background

Autonomous search requires object detection, not only for finding an object, but also so that the robot acknowledges any obstacles on its way. Object detection in a local area can be conducted utilizing simultaneous localization and mapping (SLAM) to create a map of a given environment, where the robot autonomously navigates the local area and localizes itself within it [1]. The synergy of SLAM with an object detection system will create a detailed map complete with detected objects by estimating the predetermined objects' positions and localizing each in the position within the generated map where each were found. This method allows for simultaneous navigation and identification, which combined is the most basic level of object searching. One byproduct of generating a map including objects is the possibility for objects to be moved. Therefore, a less map-dependent method of searching is better-suited for a dynamic environment. Detect-SLAM overcomes this by recognizing unreliable objects within the map through a moving probability for every object during the SLAM process [2].

Neural networking (NN) as a method of object detection allows for autonomous identification of an object rather than fed identification by a user client or outdated identification from any previous searches. An approach includes object imaging and then the extraction of landmark features of an

object in the image to differentiate it from other objects in the same space, including statistical and geometrical ones [3]. Going hand in hand with NN is deep learning, which uses datasets like PASCAL VOC, COCO, or Imagenet improve the accuracy of object detection [4]. YOLO, for example, makes extensive use of these datasets. However, one repercussion of using datasets is the need to improve upon or replace existing datasets to guarantee updated data, improve accuracy, and avoid false positives.

In order to begin searching, a robot must have a path-planning algorithm. One such algorithm constructs a path utilizing a slave agent that uses the A-star heuristic algorithm and have that slave agent communicate with a mobile agent that controls the low-level aspects of the robot. The slave agent will send the present coordinates and the goal coordinates to the mobile agent which then plans a path based on a map of the current environment. This method allows for an efficient way of searching [5]. A space-filling curve (SFC) is another algorithm than allows for a quite efficient path-planning by turning a multi-dimensional map into a one-dimensional one. SFCs characterize a linear order for arranging and planning objects that lie in the multi-dimensional space. SFCs as the basis for path-planning has numerous advantages over other methods, such as its versatility as far as the quantity of scheduling parameters, and the simplicity of code development and maintenance [6].

Several other goal selection strategies include a greedy approach (GA) that that optimizes the immediate next payoff by selecting the candidate that is reached with the lowest cost, the traveling salesman strategy (TSS) that finds the shortest Hamiltonian path in the compete graph of all goal candidates, and the traveling delivery strategy (TDS) which is uses an altered version of the depth-first search algorithm in order to speed it up. For the TDS approach, the alteration made is that only n-nearest (determined using Euclidean distance) goal candidates form the graph and only k-nearest (determined using Dijkstra's algorithm) neighbors for each node are expanded. This modification allows the user to set n and k to an arbitrary number, allowing the search algorithm to be completed in a matter of milliseconds without needing a super computer. It is found that the TSS and TDS outperform GA in the average time needed to traverse through the entirety of the room. [7]. However, the GA outperforms the TSS and TDS when the experiment is done in an environment free of obstacles because the GA enables the robot to move freely in a quick manner while TSS and TDS goes through smaller portion of the room at a time.

III. Approach

In order to approach this problem, we broke down the process into several goals that will mark milestones for the project. Initially, we worked on the recognition of the object to be found. We then determined how to specify to the robot that the object has been found in the space after the user has hid the object. Next, we determined an algorithm to have the robot move around the space to locate the item. Finally, we integrated these components together in order to have a complete system.

First we wanted to work on the recognition of the initial object. The robot needed to determine the object that it needs to find based on an object that the user shows upon the startup of the program. To do this, we made use of the YOLO object detection system, a code base that can be used to take note of the objects that are found within an image. Utilized on a robot, this can act as the eyes for knowing what the robot is seeing in the world, and how it should act upon what in front of them. For this project, the robot was able to determine the object to be found by having the user hold the object in front of its camera. Knowing that there will be background objects that the robot detects, we had the robot determine who the user was based on the largest bounding box of a person was, and then decided what the item to be found was based on what was held in front of that person. The object was then stored in the robot's database based on the classification of that object from the YOLO database. A future implementation may use neural networks to determine which object is most likely the object the user wants to find based on the location relative to the robot, or human-robot interaction confirming the found object. It would be a good idea to explore where one would usually show the robot the item they are trying to find, and congregate the data to have a more accurate reading of the initial object. And on the other hand, it would be very applicable to interface a human-robot interaction that solidifies what has been detected as the initial object. It can be by a simple user and console, but can become a more fluid communication through speech from the human to the robot.

After the initial object has been detected, the user then places it somewhere in the room and returns to the robot. The time for this to occur is simply counted by the robot, much like in a game of hide and seek where the seeker would countdown to let people hide, and it is assumed that the user understands that this must occur. Of course there could be a more developed system in this for the robot to know that the user has come back, or for the user to notify that it has hid the object, instead of simple counter in the robot. However, with the time constraints of this project, we focused more on the object detection than this simple interaction that could be improved upon later.

We then worked on finding the object for the second time, but this time within the room. We focused on making sure that the robot was able to tell that the objects it has seen by exploring a room were not the object it was looking for. Upon the detection of any objects, the robot compared the tags of those items with that of the initial object. If they matched, the object had been found. However, understanding that YOLO does contain errors within its system, we had intended to improve upon this system by having the robot move closer to the object and confirm that it truly was the object that it was looking for, making sure that the confidence score for that item is above around 70% which indicates that it should be the correct object. Unfortunately we did not achieve this point, and then attempted to obtain a pose estimate of the object based on

the initial image of the item. Using the depth image from the Kinect, we were able to obtain a relative location of the object to the robot by pulling points out of the bounding box from where the object was located based on the YOLO interface. Using this relative location to the robot, we were able to find a concrete pose estimate of where the object was located on the map. We then stored this location to have the robot to return to in order to show the user that it has found the object. However, this again has more to do with human interaction rather than object detection, and so it was not given high priority in this particular project.

Next, we had the robot implement an algorithm to explore the room and find the object. We first explored a simple base case of having the robot move towards the center of the room and rotate around in place. We presumed that this would allow the robot to take note of the entire area and detect the object in the space. However, it was soon discovered that this would not be that simple. Although this algorithm ensures that every point in the room is in the camera's field of view at some point, this alone would not guarantee object detection. The resolution of the camera, the accuracy of YOLO, and the orientation of the objects all limit its effectiveness. Upon realizing this, we decided to look into the different types of exploration algorithms that the robot could use to explore a place.

Initially we looked into frontier exploration, an algorithm used to create maps by building a costmap based on the local costmap of the robot. We discovered that we could obtain the original occupancy grid of our lab, and convert that into a map of viable locations to explore. Then based on the angle of the Kinect and the distance that YOLO can accurately detect an object, we could determine what the robot has seen and record that data to make sure the robot does not explore the same space again. In a similar manner to frontier exploration, the robot would only search in the areas it has not yet explored. We then looked into different space filling curves that could have the robot move around a space but not record what has been seen. This would have the robot explore every area without having to calculate what has been seen or not. It would make up for the fact that YOLO may not see the object upon the initial look, but would use extra time to explore the area thoroughly.

We decided upon using a simple type of space filling curve that we called the "lawn mowing algorithm." Similar to how one mows their lawn, the robot will traverse from one side of the room to another and slowly make its way across the width until it has completed the entire room. This was done by generating an array of points that could then be used by the robot as navigation goals to respond to. Upon the completion, the robot started at one corner of the room and made its way across.

A theoretical analysis of the lawn mowing algorithm shows that it can help to compensate for some of the shortcomings of the object recognition system. With a distance of 1 meter between rows and a camera angle of 70 degrees, it can guarantee that every point in the room is viewed from two sides at a distance of less than 1.735 meters and at an angle of less than 35 degrees.

Fig. 1. The set up of the robot and the clock used for experimentation.

Fig. 2. The set up of the clock for our two test cases. The purple dot indicates our first condition at which the object to find is located near the center of the room. The red dot illustrates the position of the clock in our second condition which is where we have the object to find be at one of the corners of the room.

Lastly, we brought this all together to have the robot move and search for an object. In order to connect these two systems, we had to publish a node from one part of the program that could then be interpreted by another. This would allow for a direct communication of information that may have been known, without combining these two separate functions into one code base. With this, our robot was able to smoothly transition from simple object detection and movement in order to complete a system that can search for an object.

IV. Experimental Setup

For our experiment, we first had a base case in which we would have the robot go to the middle of the room and rotate in place to look for the object. In our experiment, we utilized a clock since YOLO can detect a clock relatively well compared to other items we have tried such as a water bottle or a cup. During our experiment we would have one member of the group stand in front of the robot while holding the object we want the robot to find. After the robot stores that initial item to look for, we then proceeded to place the clock on top of a chair that is situated in different areas around the room, one being the center of the room and another being in one of the corner of the room. The robot then moves towards the center of the room and rotates around in place to try to look for the

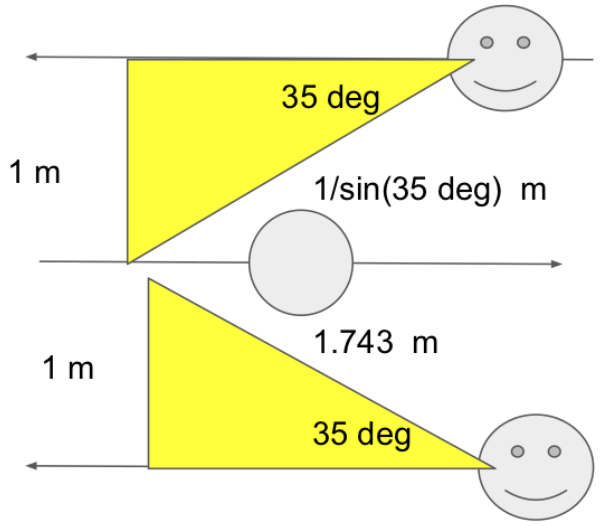

Fig. 3. The lawn mowing algorithm. The faces represent the robot at different stages of the search, the circle represents the object, and the yellow triangles represent the robot's field of view.

object inside the indicated room. Once the robot sees the initial item that it stored at the beginning of the experiment, it will return back to the user, which would be the initial position of the robot when it was first asked to store the object to find.

Afterwards, we implemented a lawnmower algorithm in an attempt to improve the robot's object finding ability. With a lawnmower algorithm, the robot traverses through the entire room starting at one of the corners. Unlike the simple case, the robot is able to obtain more data points for what it looks at. We implemented the same locations in which the object was placed, one near the center and one at the corner. And to the same extent as the previous, once the robot has found the object, it returns back to the initial position where the user is waiting. Upon completing the two conditions we then compared the time it took for each algorithm to finish under the two different locations of object to find.

V. Results

In order to record the effectiveness of our different movement algorithms, we compared the time it took the robot to start exploring and return back to the initial position. For the object near the center of the room, surprisingly enough, the robot ended up returning back to its initial position in around 40 seconds for the lawn mowing algorithm while the it took around 50 seconds for the rotating robot to finish looking and return using the base case. This is due to how the two algorithms are set up. The rotating robot only starts looking for the object once it has reached the center of the room and starts rotating. However, the lawn mowing algorithm allows the robot to start at one corner of the room where it initially starts and then immediately traverses through the room using the programmed path. This allowed the lawn mower robot to immediately find the object it is looking for even from a distance.

On the other hand, for the object near the corner of the room, the robot could not detect the object when it only rotated in the center, thus the time was infinite. This is due to how YOLO struggles to detect objects accurately from a distance greater 3 meters. However the lawn mowing algorithm found the object and return to the initial position in around 43 seconds.

VI. Discussion

Our result indicates that lawn mowing is a viable algorithm to address the problem of object finding in the robotics realm because this algorithm enables the robot to explore the whole entire area quite thoroughly. This is in spite of the fact that the experiment was run with low resolution cameras and a low frame rate. With a more sophisticated hardware, it is very likely that significantly improved performance can be gained with minimal changes to the algorithm. To make it more applicable to real-world problems requiring searching for objects, the lawn mowing algorithm can be extended from two to three dimensions. This way, aerial or submarine robots with more freedom than the BWI bot can be put to work in various environments doing tasks such as looking for an object in a cluttered space or looking for a person among ruins.

VII. Conclusion

The problem we are trying to solve is using object detection to find an object within a specified area and how to do it in a way that is efficient instead of having the robot roam around aimlessly. We have implemented two algorithms that will determine what object it needs to find based on the YOLO tag associated with it, in which the user will physically show the robot the object, then it will determine a path to follow and effectively search an area for the desired object. The robot will either move to the center of the room and take advantage of this centralized location to rotate and find the object, or move in a lawnmower path in order to traverse the room to find the object. These were both implemented through the MoveBase class and the map of the area stored within the robot. Our solution will allow a robot to have the tools necessary to find lost or otherwise hard to find objects for someone who is not able to find it but knows it to be in a particular area. Some future implementations to this problem would include applying a space filling curve. The drawback to the lawn mowing algorithm is that it is only able to see two sides of an object, never seeing the sides which are not facing along the mowing lines. This behavior of the lawn mowing algorithm will cause the robot to fail to find an object that is covered by another object within the the robot's limited line of view with lawn mowing.

References

[1] S. Ekvall, P. Jensfelt, and D. Kragic, "Integrating active mobile robot object recognition and slam in natural environments," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 5792–5797.

[2] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang, "Detect-slam: Making object detection and slam mutually beneficial," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1001–1010.

[3] K. Kim, J. Cho, J. Pyo, S. Kang, and J. Kim, "Dynamic object recognition using precise location detection and ann for robot manipulator," in *2017 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)*, May 2017, pp. 237–241.

[4] X. Zhou, W. Gong, W. Fu, and F. Du, "Application of deep learning in object detection," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, May 2017, pp. 631–634.

[5] C. Popirlan and M. Dupac, "An optimal path algorithm for autonomous searching robots," *Annals of the University of Craiova-Mathematics and Computer Science Series*, vol. 36, no. 1, pp. 37–48, 2009.

[6] M. Ali and S. Ladhake, "Overview of space- filling curves and their applications in scheduling," *International Journal of Advances in Engineering and Technology*, vol. 1, no. 4, pp. 148–154, 2011.

[7] M. Kulich, L. Přeućil, and J. J. M. Bront, "Single robot search for a stationary object in an unknown environment," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5830–5835.