Flip the Switch

Efren Mendoza Enriquez, Elin Park, Jay Upadhyaya, Kevin Ayala, Sai Tanuj Madisetty

***Abstract*—Computer vision is an essential and interdisciplinary field that explores how computers detect and gain understanding from digital images or videos. Object detection and recognition is a subtopic under computer vision; the digital images or videos are processed and given meaning through semantic mappings.**

Essentially, we designed a program that uses grid-based convolutional neural networks with fixed bounding boxes (G-CNN) computer vision. In addition, the big ideas of object recognition were utilized to detect and find the depth orientation of a light switch. Our experimentation stage consisted of training a dataset in YOLOv5 (You Only Look Once) with a large assemblage of data, connecting with live footage, and finding the 3D orientation of the light switch to create transformations. The results indicate that our program is successful in processing data to eventually create interactions with a light switch.

I. Introduction

The problem at hand is that we need the robot to recognize a light switch in space and interact with it. Thus, turning them on or off autonomously proves to be a challenging task. We'd like to solve this issue to learn about object detection and how to implement something new ourselves - such as orienting the light switch in an orientation or pose that's easy to flip on and off.

In lieu of our problem, our goal is to essentially have the BWI bot go up to a light switch, recognize the switch through video input, and flip it on through a command. To compartmentalize the problem due to the time constraint, we will focus on simply the object recognition aspect. We will do this by creating an image segmentation of a light switch by receiving the color and depth image received by Azure Kinect, passing it through YOLOv5, creating a point cloud of the light switch, and publishing it to ROS.

In the real world, convenience is an important aspect. Laziness is fueled by inconvenience; thus, simple tasks such as getting up in the middle of the night to turn the light switch on or off is an issue we wanted to resolve. This is one such practical application of this experiment. As for future research implications, the object recognition coupled with transformations could prove to be quite useful in other contexts as well, such as opening or unlocking doors, therefore providing a way for these ideas to be further built upon.

II. Background

Image segmentation is the process of breaking down an image into subgroups/multiple segments (set of pixels, pixels in a region) called image segments based on color, intensity, locations, or boundaries of an object. This simplified image is more meaningful and easier to analyze, as it emphasizes a specific attribute or object within a frame. Examples of where image segmentation can be applied include face and fingerprint recognition, filtering of noisy images, or satellite images.

We previously worked with image masking in the colored cups program, in which we utilized ROS and OpenCV to mask and contour the color of the cups. This allowed us to differentiate between the cups by color and display each cup through various OpenCV transformations. In addition to that project, to prepare for this experiment, we read over the documentation for OpenCV, rospy, YOLO, and pyk4a and viewed supplementary videos about how to effectively use these packages together.

We have read previous works and experiments related to our problem including some papers related to the pandemic and masks (Y. Liu, B. H. Lu, J. Peng, and Z. Zhang). Image segmentation also took a large part in their project. Professional technologies such as R-CNN were utilized. Anchoring and loss are also indicators that this group of researchers used in their experiments.

III. Methodology

Our algorithm takes in a live video input displaying a light switch through Azure Kinect and detects the appearing object through YOLOv5. This pre-implemented grid-based CNN algorithm only looks at an image once and determines where a trained object is in the frame, if at all. We trained it with a light switch, and the algorithm outputs fixed bounding boxes, which work in coordination with pyk4a to display the frame with a box around the light switch.

However, due to the constraint of the light switch not being in the range of the Azure Kinect camera for a live video feed, a k4a recording was used in place of the live video for testing and experimental purposes. With PyTorch for the YOLO algorithm, pyk4a, and the rospy packages, the majority of the code was done in Python in a virtual environment.

A. Experiment Steps

The first step in our methodology consisted of taking pictures of various light switches in altered configurations and physical spaces. Lighting and the distance were also taken into consideration when gathering our data. After the compression of the pictures, we utilized this dataset to label the light switches using Makesense.ai, an online image annotation tool. We then created bounding boxes around the object we wanted to recognize and gave the object an appropriate label. These bounding boxes were saved as coordinates in text files, which would later be used by YOLO to train the algorithm.

We then split our data set in two: the training set and the validation set. The training set was used to develop the initial algorithm's ability to predict where the light switch is in a frame, and the validation set was subsequently used to tune the accuracy after each run of the algorithm.

Finally, we fed the data into YOLOv5 with varying epoch sizes. Epoch is a term under the umbrella of machine learning that indicates the number of times the entire dataset is passed forward and backward through the neural network. As the epoch size increased, the precision and recall values tended closer to 1 (which is ideal for accuracy), and as a result, our final algorithm was trained with an epoch size of 120.

Since the entire dataset is generally too large for the algorithm, the data is processed by the algorithm in batches. The batch size is a term that refers to the total number of training examples present in a single batch, and in our experiment, we ran the algorithm with a batch size of 16.

IV. Experimental Setup and Evaluation

At a high level, we were able to recognize the light switch using data streams from the Azure Kinect given in real-time. We trained an algorithm with YOLOv5 technology and used this algorithm while connected to Azure Kinect to transform our video input into a 2D depth map, which was then translated into a point cloud.

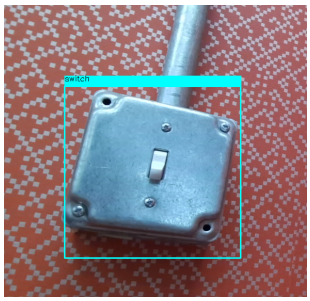

Fig. 1. YOLOv5 Bounding Boxes Around the Light Switch

A. Technologies Used

The technologies we utilized to retrogress our experiment are ROS, YOLOv5 object recognition, MakeSense.ai to label and train our dataset of images and videos of light switches, RVIZ, Azure Kinect, and the pyk4a python wrapper for the Kinect SDK. For the Azure Kinect, we made use of the Azure Kinect Viewer and the Kinect DK Recorder(k4arecorder). While our goal is to have the Kinect detect a light switch in real-time to simulate it being on a robot, the light switches were not within range of the test machines. Therefore, we had to use the k4arecorder to record a video to imitate a live feed.

However, the live feed from the Kinect utilizes a different encoding software than the k4arecorder; the recording is encoded with MJPEG, but the live feed and pyk4a methods process playbacks with BGRA32 encoding. As a result, some transformations were necessary to use the recorded playback instead of the live feed.

The Kinect playback receives a capture for each frame of a video (or each frame of a live feed), and it returns data from both the depth camera and the color camera, which were both used in this experiment. An RGB image is a 3D array; a channel for each of the three colors is stored in a 2D array. We also employed the depth camera, which uses the time of flight principle, is used to measure the distance from the camera to each of the points. Similar to the three color channels, the depth image is a 2D array, where each value in the array is the distance from the camera to the pixel. It is also important to note that the depth is in NFOV mode, meaning that there is pixel overlap between the depth and RGB camera. This allows for more accurate points and positioning of points of the point cloud in the point cloud map.

B. Data Analysis

The P-value was 1.0, which indicates the precision value of our training data; the R-value, otherwise known as the recall value, was 0.9. These results are significant in that for 100 percent of the experiments, there were 0 false positives returned by the algorithm; that is, there were no objects detected by the algorithm that were no light switches. Additionally, the recall value of 0.9 indicates a strong, positive association between the data points in the data sets. This signifies that there were very few false negatives returned by the algorithm, as there were only a few instances in which the algorithm did not detect and mark the light switch. As the epoch size increased, both the p-value and r-value increased (shown in fig. 2-3), showing that the more the algorithm runs through the dataset, the more accurate the output is.

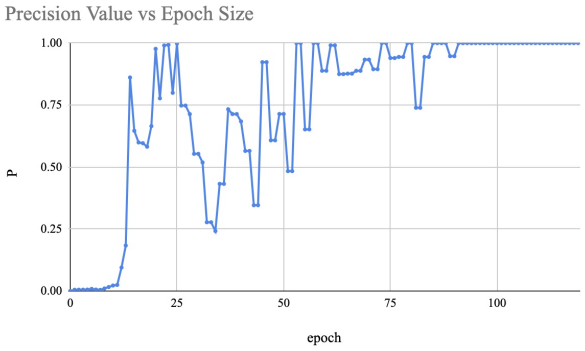

Fig. 2. YOLOv5 Point Cloud of the frame

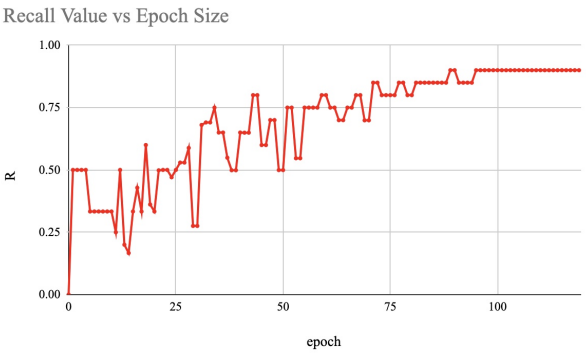

Fig. 3. YOLOv5 Point Cloud of the frame

V. Results

The overall result of our project was successful. Given five pre-recorded video our trained algorithm was able to correctly detect the light switch. Even as the video would pan left, right, up, and down, the light switch was still able to be detected. Taking the information from the depth camera and creating a point cloud turned out to also be pretty successful. While we were able to receive the point cloud of the frame (shown in fig. 4), our next step would be to isolate it and publish the point cloud of just the switch.

To test our results, we took a recording using k4a-recorder with Azure Kinect. Due to the disparity in encoding software, the color image transformation didn't work, only able to transform the point cloud. The next step would be to transform MJPG to BGRA32 In each recording, we tested whether the YOLO algorithm detected the light switch

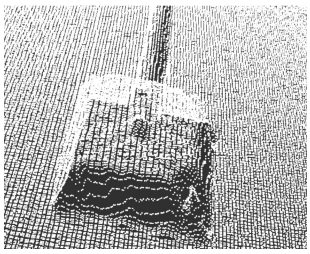

Fig. 4. YoloV5 Point Cloud of the frame

However, with more time and effort, having the UR5 Robotic-arm perform the action of flipping the switch on or off would have made this project more impressive and interesting.

Because our project was not focused on obtaining raw statistical data, our project results were interpreted more abstractly rather than in mathematical terms.

VI. Discussion

This experiment and its results are important in that our algorithm can be implemented, translated, and utilized in different contexts and settings. Different robotic arms and systems can adopt this algorithm and implement different actions of flipping light switches on or off after the 3D orientation is calculated through pyk4A.

While this experiment is restricted to light switches, it would not be difficult to translate these results to other objects by creating a YOLO algorithm with a dataset of different objects. For example, if the goal was to identify and turn door handles for a user, a new algorithm with a dataset of door handles would allow the program to detect door handles instead of light switches. Therefore, the results of the experiment could be much more valuable than simply finding a light switch.

Additionally, this experiment may be beneficial for the future because we can implement these techniques and this type of algorithm to find other objects. Since the basic steps to reproduce these results are the same (train object model, find depth point cloud, and get the spatial transformation). This is beneficial since service robots will need to depend on this type of tech and steps.

In hindsight, it was interesting to see the conjoining of simple codes into more complex codes to achieve our goal. For example, combining code led to capturing individual frames from Azure Kinect and transforming those frames into the point cloud system.

VII. Conclusion

Our experiment was significant in that different light switches can be detected in different physical spaces. For instance, a light switch in a large gymnasium can flip on or off as well as potentially a sliding switch in a kitchen in a home.

Our results and experiment can be used in the future with different systems and algorithms for different physical areas. Different rooms may contain different orientations of light switches that can be detected with our algorithm. For example, a switch in a classroom is oriented at a dissimilar height and physical placement than the light switches in a warehouse or a classroom. Moreover, as robots - especially service robots - become overwhelmingly important in our society, there is a higher need for robots with object detection software. They increase efficiency and ameliorate the state of convenient living.

For example, another variation of detection such as voice recognition algorithms in Alexa and Google Home allows for an improved quality of life. While this is slightly different, eliminating the act of getting up and flipping a light switch on or off will undoubtedly decrease the inconvenience in the quality of life.

References

[1] "Image segmentation techniques 1," CiteSeerX. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.227.6638. [Accessed: 10-May-2022].

[2] D. Davies, "Yolov5 object detection on windows (step-by-step tutorial)," Wamp;B, 20-Sep-2021. [Online]. Available:https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on-Windows-Step-By-Step-Tutorial—VmlldzoxMDQwNzk4what-is-yolo. [Accessed: 10 May 2022].

[3] L. doleron, "Detecting objects with Yolov5, opencv, python and C++," Medium, 02-Mar-2022. [Online]. Available:https://medium.com/mlearning-ai/detecting-objects-with-yolov5-opencv-python-and-c-c7cf13d1483c. [Accessed: 10-May-2022].

[4] "S3," Amazon, 2002. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 10-May-2022].

[5] Y. Liu, B. H. Lu, J. Peng, and Z. Zhang, "Research on the use of YOLOV5 object detection algorithm in mask wearing recognition," World Scientific Research Journal. [Online]. Available: https://www.airitilibrary.com/Publication/alDetailedMesh?docid=P20190709001-202011-202010280002-202010280002-276-284. [Accessed: 11-May-2022].