2D to 3D point Mapping for Object Detection

Emerald Zhang[1], Tapaswini Kodavanti[2], Varsha Ravi[3], Phuc Nguyen[4]

***Abstract*— 3D point clouds and 2D image data are two different visual data formats for representing the same space in the physical world. This introduces the problem of multiple overlapping visual data inputs that need to be mapped together. In our paper, we propose a system that applies image segmentation on a 2D image and determines the 3D point cloud correspondence through transformation algorithms. Our segmentation pipeline successfully segments objects from the 2D image, converts the depth image to a point cloud, and registers the segmented image to the correct points in the point cloud.**

I. Introduction

Object detection techniques often leverage different deep learning algorithms to identify and recognize objects in images and videos. In the case of robotic vision, visual data may be consumed by various types of cameras, resulting in 2D image data and 3D depth image data of the same scene. While human vision can differentiate between both formats and create correlations between the same object represented in both formats, computer vision models need varying processes to clean and interpret the data. The current BWI bots have cameras that produce both 2D and 3D data, but there is not a formal way of associating both image formats together. Our project aims to register incoming 2D image data with point-cloud data, so an object in one format corresponds to the same object in the other format.

There are three main steps in this process: (1) Applying image segmentation on the 2D images, (2) Computing depth image from point cloud, and (3) Registering the segmented image with the point cloud. We utilize three methods of image segmentation for step 1, choosing from basic color-blob detection methods, residual networks, or vision transformers.

II. Background

Image segmentation and object detection are imperative components of a robotic vision system, allowing robots to focus on a specific object of interest and perform operations related to it. The most common method is using image processing algorithms, most notably YOLO residual networks [1] and CLIPseg image transformers [2].

Image transformers essentially process 16x16 pixel patches of the image and produce context embeddings containing information about the color pattern and location in the entire image. These embeddings are fed into a transformer architecture which uses the attention mechanism [2] to produce bounding boxes around the object. Some algorithms use Large Language Model (LLM) prompting [3] , which then produces an encoding to segment any given image. Prompt engineering tactics are utilized to ease the pretraining process and to transfer zero-shot to new image distributions and tasks.

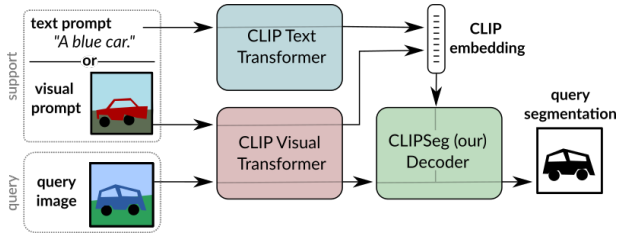

Fig. 1: CLIPSeg Architecture

Previous work in this area utilized deep learning techniques within the transformation process to produce a more accurate image to point cloud mapping algorithm.

A. I2P mapping algorithm

In order to map 2D image data point-by-point to its corresponding point in a 3D plane, work has been done to implement cross-modality projection between an image and a point cloud. Using an RGB image and a general point cloud from a 3D Lidar scanner captured at different locations in the same scene, their method estimates the relative rigid transformation between the two coordinate frames[4]. Other works focus on generating a feature-based correspondence[5] between a 2D and 3D image. Specifically, given a pair of a 2D image and a 3D point cloud, they are first transformed into high-dimensional feature spaces, then the resulting features are fed into a symmetric overlapping region detector to determine the region of overlap, establishing dense 2D-3D correspondence.

B. 3D Reconstruction

An effort has been made to reconcile the image recognition and depth mapping problem. Current research focuses on conducting alignment between CLIPencoded point cloud and 3D category texts. A point cloud is projected into multi-view depth maps without rendering, and the view-wise zeroshot predictions are aggregated to achieve knowledge transfer from 2D to 3D. Another approach explores single-view 3D reconstruction by learning generalizable representations inspired by advances in self-supervised learning [6]. The

[1]Emerald Zhang is with the College of Natural Sciences, The University of Texas at Austin, Austin, TX 78712, USA emerald.zhang@utexas.edu

[2]Tapaswini Kodavanti is with the College of Natural Sciences, Austin, TX 78712, USA tk24428@utexas.edu

[3]Varsha Ravi is with the College of Natural Sciences, The University of Texas at Austin, Austin, TX 78712, USA vravi@utexas.edu

[4]Phuc Nguyen is with the College of Natural Sciences, The University of Texas at Austin, Austin, TX 78712, USA pnn329@utexas.edu

algorithm compresses the input appearance and geometry to predict the 3D structure by querying a 3D-aware decoder to transform RGB-D videos to 3D points and scenes.

III. Method

We propose a pipeline to incorporate fine-tuned image segmentation for 2D object detection with a transform from 2D to 3D point cloud as a means of achieving 3D object detection. The processing is done on a live video stream frame by frame from a Azure Kinect DK Camera. Our project contains three methods of image segmentation, each listed with higher segmentation accuracy: Color blob detection, YOLO v8 models, and CLIPSeg models.

A. Colorblob Detection

In its raw form, the inputted image from the 2D image camera consists of 3 matrices, each matrix representing one color pixel from the standard RGB (Red, Green, Blue) formatting. In this method, the only objects the program can detect must be a single saturated color object that is red, green, or blue. The algorithm iterates over each RGB matrix and determines which pixels pass a thresholding test, thus identifying all contours that are a given color. From here, the largest contour by area is selected and assumed to be the object in question. In order to truly segment the object in the image, a bitwise-and operation is performed between the mask and the original image - the original color in the image is only preserved if and only if both pixels are greater than zero.

B. YOLOv8 Segmentation

Using a medium-sized YOLOv8 segmentation model pretrained on the COCO dataset, each frame was provided as input to the model. Upon prediction, a series of masks of any object in the frame was generated. Then, a mask specific to the queried object was found by filtering the list by class id. Because the COCO dataset has 80 pretrained classes . Performing the same bitwise-and operation on the original frame returned a fully processed and segmented image.

C. CLIPSeg Segmentation

The desired object, including attribute specifications such as color and size, were provided to a pre-trained CLIP model along with the given image frame as input. After prediction, each pixel was associated with a probability that measured how close to the given input it was. Using a sigmoid function to range these probabilities between 0 and 1, each pixel was compared to a likelihood threshold of 0.1 [7] in order to create an image mask.
The algorithm first processes the camera capture and computes an image mask covering the specific object we are looking for. The mask is applied to the input image and converted to a depth image format. Using the Azure Kinect API, we apply the *depth_image_to_point_cloud()* method to obtain the point cloud format of the input 2D image.

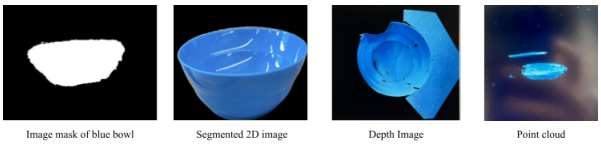

Fig. 2: Image Processing Pipeline Example

IV. Experimental Setup

To measure the accuracy of our method, we collected qualitative data, conducting visual tests with the 3 algorithms using the same selected objects in various settings. The orientation of the objects and camera were kept under control. The objects were selected using three criteria: solid, irregular patterned, and color-blocked. For solid objects, rgb bowls, red apples, and blue water bottles were used. Irregular patterned objects were utilized to test if an entire object was segmented or if it was just the solid color. For this we used a pringles can and a tape dispenser. Finally, color-blocked objects(a white cup with a red bottom) contain a distinct separation of color and allow a test between the full object or just segments of the object.

The first batch of evaluations were done with 3 background types - noiseless, controlled noise, and noisy/cluttered. The noiseless background acts as a control. The neat background was chosen for controlled interference, allowing for some parts of the background to be the same color as the object to detect. The final level simulates the real world where the amount of a given color or object is unpredictable in a setting, allowing us to test the robustness and limits to the system.

V. Results

We found that certain segmentation algorithms performed better in some situations than others, regardless of the quality of the algorithm.

A. Colorblob Detection

When displaying a colored object in a controlled, noiseless environment, the algorithm identified the correct outline of the image. When overlapping the object with other distractors (background environment or other separate objects), the algorithm failed to segment only the object. Instead, only the intersecting color blocks were also detected, since the algorithm iterates over all pixels and simply partiones each based on a constant threshold. For objects that had multiple color blocks, this meant only the requested color shade blocks were segmented.

B. YOLOv8

The YOLOv8 model was much more sensitive to noisy backgrounds in comparison to the other segmentation methods. The object was clearly segmented in noiseless environments but failed frequently when other objects were present in the image. However, object segmentation overall was much more accurate than color blob detection. The algorithm is not sensitive to particular color shades, so the entire object was frequently segmented. However, segmentation is limited

to a certain list of detectable objects, so detecting any object based on pure colors is not possible.

C. CLIPSeg

The CLIPSeg model was the best model in terms of application practicality, since each object was detected based on specific text prompts. This model was not sensitive to noisy backgrounds or irregular multicolor patterns on the object, so the model frequently segmented the overall object from the image accurately. The model was not able to segment out holes within the object (such as the space between the handle and the cup parts of the coffee mug).

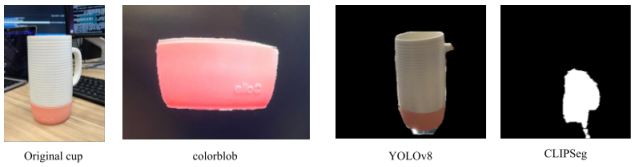

Fig. 3: Segmentation Algorithm Variation with color-block cup

VI. Discussion

We found that object detection in three-dimensional space could be achieved by applying frame-by-frame transformations overlaying a 2D color image on top of a depth map. Thus, not only is the proper object identified, but its properties can be maintained and utilized as information, and its spatial positioning can be discerned.

We expect that, with our technique, robots will be able to integrate it into their systems as a basis for discovering objects. For example, task completion can be efficiently done as there is now a way to find physical objects in space. We also see it being used in other tasks such as obstacle avoidance and searching devices.

VII. Conclusion

We presented an approach to object detection through the segmentation of 2D rgb images transformed onto a 3D image that can be adapted to any distinct object or any object through further training. Specifically, we explored different strategies for image segmentation to produce a masked image with the desired object in 2D. Once the object was detected in 2D, we then applied a series of k4a transformations to construct the object in a 3D space using OpenGL. Beyond that, we showed that using pre-trained models work with significant accuracy.

The system detailed in this work demonstrates that arbitrary object detection in a 3D space is a capability that robots can achieve. With the capacity to mask and identify almost any object, the information could be published to a ROS topic to either localize the object in the given environment or pass the point cloud into a manipulation pipeline, allowing for physical interaction and acknowledgment of the object. This system can be extended and adapted as more powerful segmentation models are created to enhance vision potential for robots.

References

[1] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," 2023.

[2] T. Lüddecke and A. S. Ecker, "Image segmentation using text and image prompts," 2022.

[3] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[4] J. Li and G. H. Lee, "Deepi2p: Image-to-point cloud registration via deep classification," 2021.

[5] S. Ren, Y. Zeng, J. Hou, and X. Chen, "CorrI2p: Deep image-to-point cloud registration via dense correspondence," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 3, pp. 1198–1208, mar 2023. [Online]. Available: https://doi.org/10.1109%2Ftcsvt.2022.3208859

[6] C.-Y. Wu, J. Johnson, J. Malik, C. Feichtenhofer, and G. Gkioxari, "Multiview compressive coding for 3d reconstruction," 2023.

[7] X. Chen, Q. Hu, K. Li, C. Zhong, and G. Wang, "Accumulated trivial attention matters in vision transformers on small datasets," 2022.