A Real-time, Scalable and Extensible Object Filtering and Detection System Using Kinect Sensor and ROS2 Foxy

C.M. Gunasekara
Department of Computer Science and Engineering
University of Moratuwa
Moratuwa, Sri Lanka
ORCID: 0009-0007-7049-5087

Keywords— Kinect, ROS2, Real-time, Shape filter, Color filter

I. INTRODUCTION

Object detection and filtering based on shape and color is an important capability for many robotics applications. For example, sorting objects by shape and color is a common industrial application. Service robots also need to detect and track objects based on visual properties. While powerful deep learning approaches like YOLO have emerged for general object detection, they require large datasets and extensive training. A simple shape and color filtering provide a lightweight and customizable alternative.

This work aims to provide a real-time modular and lightweight system that can identify objects of basic shapes and colors and allow extensibility of functionality by incorporating more custom color and shape filters. The proposed system for real-time shape and color filtering using a Microsoft Kinect RGB-D sensor and Robot Operating System (ROS2) can identify an array of regular shapes like circles, rectangles, and triangles over a spectrum of different colors. New shape and color filters can be added dynamically at runtime thanks to the modular, ROS-2-based implementation.

II. LITERATURE REVIEW

Segmentation and filtering by color is a well-established computer vision technique [1]. Basic shape detection approaches like contour analysis are also commonly used [2]. More advanced techniques use neural networks to identify shapes [3]. ROS1 has been used in several works for object sorting using RGB-D data [4]-[5]. There is limited prior work leveraging ROS2 for similar applications. A custom ROS2 package for detecting basic shapes like circles and squares was developed in [6]. This work builds on such efforts to create a ROS2 system for shape and color filtering using a Kinect.

III. SYSTEM DESIGN

A. *Hardware*

The system uses a Microsoft Kinect v2 RGB-D camera as the primary sensor. It captures 1920x1080 RGB images and 512x424 depth images at 30 FPS. The Kinect is mounted on a fixed location overlooking the workspace. All processing is done on a laptop with an Intel Core i5 CPU running ROS2 Foxy on Ubuntu 20.04.

B. *Software*

The overall system is implemented using ROS2 Foxy in C++. A publisher-subscriber model incorporates different color and shape filters to the input image and depth data streams sent from the Kinect camera. The kinect_ros2 library is used to interface the system with the Kinect sensor. OpenCV2 library is used for image processing and contour analysis.

IV. SYSTEM ARCHITECTURE

The nodes first subscribe to the raw image data published by the Kinect camera node. First, the color filter operations are done. Each color filter performs its filtering operation on the image received by subscribing to the image. Finally, the combined results of the color filter are published on the *filtered_rgb* topic. The shape filters then subscribe to this topic and perform their shape-filtering process on this data stream. All the shape filter outputs are then published on a single topic.

The following *rqt_graph* output gives an overview of the topics and nodes used in the workflow.

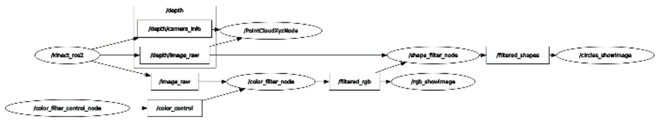

Figure 1.1- Topic and Node List

V. MATHEMATICAL REPRESENTATIONS

The system workflow goes through several crucial operations to perform the above functionality.

A. *Conversion from RGB Color Space to HSV Color space*

Given the values of the image from the RGB, space be *R*, *G, and B* the following transformation is used to convert the color space from RGB color space to HSV

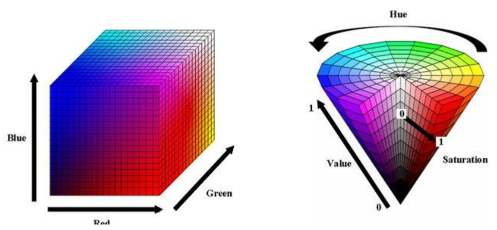

Figure 1.2 RGB color space vs HSV color space[7]

color space where *H, V, and S* represent *hue, value* and *saturation* values.

$$M = max(R,G,B) \quad m = \min(R,G,B)$$

$$H_0 = 60 * \begin{cases} 0 + \dfrac{G-B}{M-m}, & M = R \\ 2 + \dfrac{B-R}{M-m}, & M = G \\ 4 + \dfrac{R-G}{M-m}, & M = B \end{cases}$$

$$H = \begin{cases} H_0\,, & H_0 \geq 0 \\ H_0, & H_o < 0 \end{cases}$$

$$S = \frac{M-m}{M}$$

$$V = Max(R,G,B)$$

B. *Filtering Image stream based on color filter*

Define 2 bounds of HSV values that the extreme color values for the required color range. Define them using *OpenCV Scalar*. Then threshold the image using these limits. Let the source image be *I*, and the function *f* performs the filtering. *lower* and *upper* are the lower threshold and upper threshold values respectively. *I`* will represent the resulting image. Let *x, y* be the coordinates of the pixel considered.

$$I'(x,y) = f(x) = \begin{cases} I(x,y), & lower \leq I(x,y) \leq upper \\ 0, & otherwise \end{cases}$$

C. *Determine the bounding boxes*

Use *boundingRect* method of OpenCV to determine the bounding box of the required image. Filter out possible disturbances by defining the *width: height* ratio of the bounding box.

VI. RESULTS

The system can seamlessly identify images according to the color filters and shape filters provided. The following image shows how the system identifies *red* and *green circular objects* in the image stream. The depth value (untransformed) is shown in the raw form as they are received from the Kinect Sensor. These depth values are not calibrated or represent the actual distance to the objects.

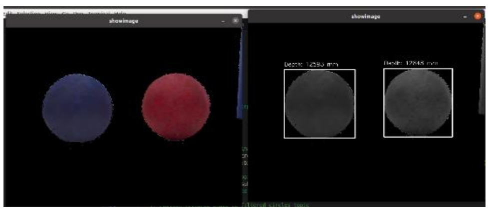

Figure 2.2 - Results of color filter and shape filter

ACKNOWLEDGEMENT

I would like to extend my thanks to Dr. Sulochana Sooriyarachchi for supervising this project.

REFERENCES

[1] J. Serra, Image Analysis and Mathematical Morphology. Academic Press, 1982.

[2] S. Suzuki et al., “Topological structural analysis of digitized binary images by border following,” CVGIP, vol. 30, pp. 32-46, 1985.

[3] L. P. Cordella et al., “An improved algorithm for matching large graphs,” in Proc. 3rd IAPR-TC15 Workshop Graph-based Representations in Pattern Recognition, 2001, pp. 149–159.

[4] A. Vannucci et al., “ROS2Learn: a ROS2-based framework for machine learning and computer vision,” in Proc. IEEE/SICE International Symposium on System Integration, 2019, pp. 926–931.

[5] E. Erdal and O. E. Erdinc, “ROS based multi-layer cnn object sorting,” in Proc. IEEE International Conference on Advanced Robotics and Mechatronics, 2019, pp. 243–248.

[6] P. Inigo-Blasco et al., “Robotics software frameworks for limited resources systems,” in Proc. IEEE Global Engineering Education Conference, 2020, pp. 1–8.

[7] Chen, Rui & Wang, Meiling & Lai, Yi. (2020). Analysis of the role and robustness of artificial intelligence in commodity image recognition under deep learning neural network. PLOS ONE. 15. e0235783. 10.1371/journal.pone.0235783.