# Rovio Mapping Using Color Based Object Localization

Alexander W. Mora, awm25

***Abstract*—I attempted to design and implement a procedure for localizing objects in a region using object colors. The platform used was the Rovio robot. To accomplish this task the image from the robot was analyzed to distinguish directions of color to correspond to objects within the scene. This was then combined with the position information from Rovio's navigation system to allow for specific direction information within the environment.**

## I. INTRODUCTION

THIS project focused on the use of Rovio to create a map of a scene. The method used for creating the scene was developed specifically for this application. Rovio's navigation information and web camera were used in combination.

The web camera's image was analyzed to divide it into vertical segments that indicated significant vertical attributes in the scene. This technique was designed in reaction to the realization that most objects within a scene have a significant vertical component. Some examples of this are doorways, boxes, chair legs, and trash receptacles. The vertical segments were produced using the general design of a color segmentation algorithm. This vertical information is then projected in directions determined by the camera's view.

Following this image analysis the projection is then combined with the navigation information to locate it on a 2D plane of the floor. By projecting this information onto the floor a general idea of object location can be derived. Combined with different perspectives on the same region specific locations can be determined by locations in which directions of the same color overlap.

With enough views and perspective the entire region within Rovio's navigation system could have been accurately mapped; however, this project did not make it that far towards implementation.

## II. PROCEDURE FOR MAPPING SYSTEM

### *A. Analyze Web Camera Image*

The image analysis technique stemmed from the graph based color segmentation algorithm. The image initially is smoothed by a Gaussian filter. Then each vertical line of pixels was distinguished as a node. The difference between each node and their neighbor is established by a simple squared difference calculation. This produces a weighted edge between each node. Afterwards, the edges are sorted and are inserted into the graph from lowest weight to highest. The edge is only inserted if the max difference of the two different connected components on each end of the edge plus a threshold value scaled by the size of the connected component is greater than the weight of the edge to be inserted.

After all the edges have been inserted into the graph what this creates is a collection of connected components of vertical lines that share generally similar attributes. This follows from the philosophy that the algorithm is looking for vertical features in the image like chair legs and door frames. After these initial groups are created the smaller groups are removed from the collection. This is necessary because at the border between various vertical features in the image there are liable to be significant changes in the vertical lines of pixels that aren't significant to the actual object we're looking to differentiate.

Each of these groups is then assigned a color based off of the center pixel row in the group. The pixel is taken from the top area of the pixel row in an attempt to select a color representative of the object. A better implementation would've selected the representative color more intelligently, but this implementation worked acceptably with such a simple selection process.

### B. *Projection of Colors*

Following the grouping of vertical lines the groups are then projected onto a 2D plane that is parallel to the ground. This is accomplished by calculating the area of the webcamera's vision occupied by each group. This area is given by the actual column location of the pixels. The vector is then calculated using a calibrated distance to the view window with a measured width of the view window. This vector is then followed until the edge of the mapped region. All along the vector the color of that group is applied to the floor map.

Ideally, this application of color would also include consideration for the application of the same color to the map as had already been drawn. This would indicate that an object had already been seen in this location from another direction. When the same color is recognize the cast length of the vector can be truncated to end at the given location of the object as determined by the separate viewing direction. In Figure 4 you can see the orange cone is being projected and overlapping with another viewing of the same area.

### C. *Figures*

The following figures show the application of this algorithm to a scene.


Figure1. View from Position (x=0, y=0, ang=0)

Figure 2. Smoothed Image


Figure 3. View from
Position (x=24, y=-20, ang=$\pi$/2)

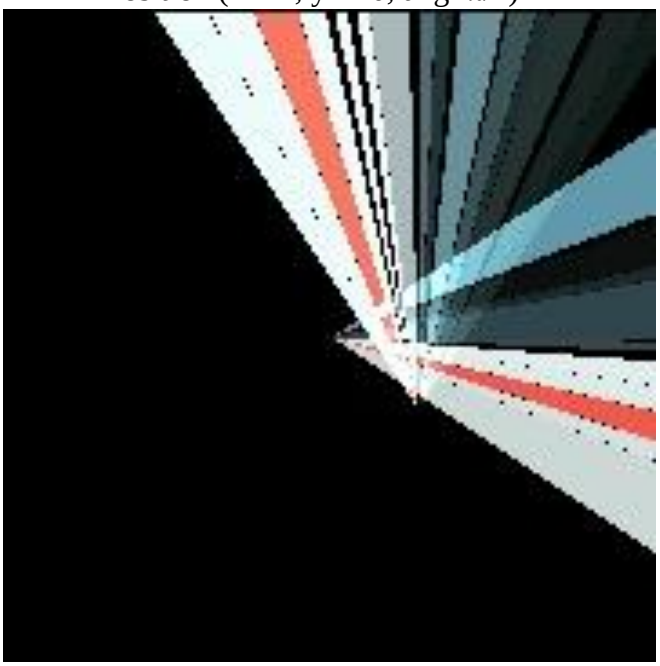

Figure 4. Results from Projection onto Floor

## III. Results

This method seemed to be generally effective; however, I was unable to apply this procedure to the location information given by Rovio's navigation system due to some of the inaccuracies of its measurement. As a result, the majority of the calculations had to be completed given very specific values not from the robot.

Additionally, the segmentation algorithm which was the majority of the work needed some careful attention to some of the variables controlling the smoothing window width and height, the threshold difference, and the minimum group size. The initial values that were chosen ended up being too low resulting in far too many groups. This was corrected using a very simple learning optimization algorithm.

The image analysis was performed with a specific initial set of parameters to the analysis. Then based off of the error between the discovered number of groups and the expected number of groups for a given image, the parameters were adjusted in an attempt to reduce the error. The training values of the expected number of groups for a given image were inserted by me and the algorithm was set to distinguish more appropriate values. This learning approach was necessary because there were four variables to optimize over. The difference in error for the training set of 15 of the 40 images can be seen in Table 1. Additionally, Figure 6 and Figure 7 show the differences in the vertical segmentation and grouping given by the trained and untrained parameters on Figure 5. The training may have over reduced the segmentation.

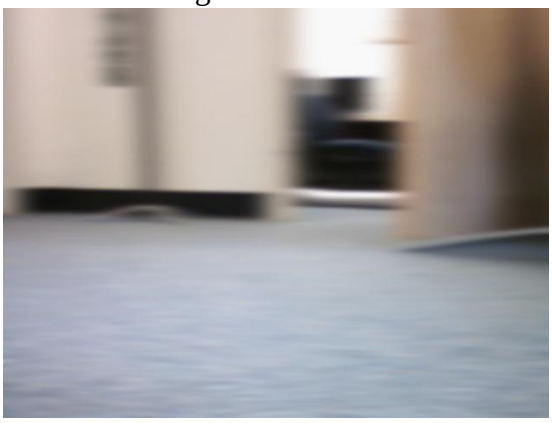

Figure 5. One of the Training Scenes

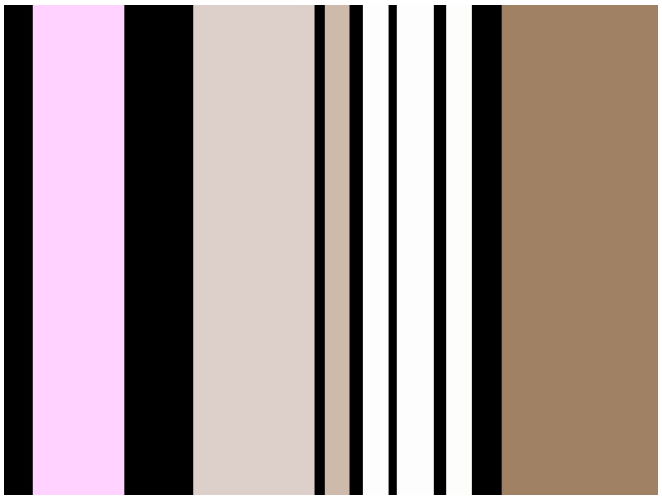

Figure 6. Untrained Parameters (black is ungrouped area)

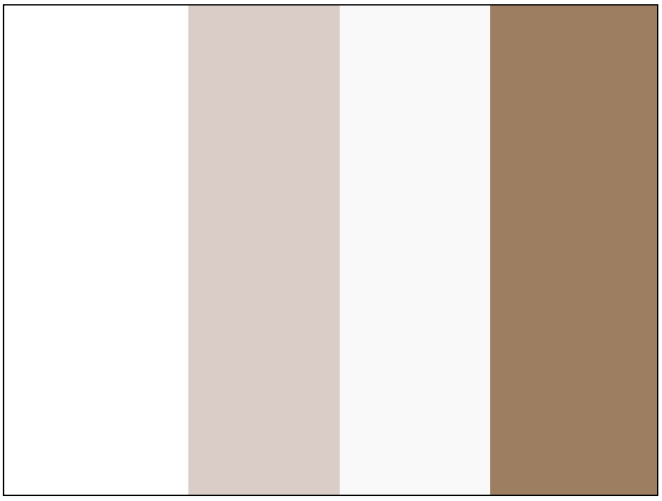

Figure 7. Trained Parameters

Table 1. Sample of Training Results

| 15 of 40 Test Cases | Untrained Absolute Error | Trained Absolute Error |
|---|---|---|
| 1 | 4 | 1 |
| 2 | 3 | 1 |
| 3 | 2 | 1 |
| 4 | 9 | 3 |
| 5 | 2 | 3 |
| 6 | 8 | 1 |
| 7 | 5 | 2 |
| 8 | 9 | 4 |
| 9 | 5 | 4 |
| 10 | 8 | 1 |
| 11 | 7 | 1 |
| 12 | 7 | 3 |
| 13 | 4 | 5 |
| 14 | 6 | 2 |
| 15 | 6 | 2 |
| SUM : | 85 | 34 |