Robot Visual Mapper

Hung Dang, Jasdeep Hundal and Ramu Nachiappan

***Abstract*— Mapping is an essential component of autonomous robot path planning and navigation. The standard approach often employs laser range finders, however, they are expensive. Cameras can be used but it is difficult to extract accurate range information. For our project, we develop a simple method for local planar mapping of a robot's surrounding environment using only monocular camera images. We utilize SURF to calculate the change in a robot's orientation and implement a simple segmentation method to identify obstacles in an image. A local map of the robot's surrounding can be built by applying a pin hole model to the segmented image and combining the result with the calculated orientation. In addition, using Singular Vector Machine, we implement a simple classifier that detect uniquely colored fluorescent objects. We test our methods in different indoor environments with a Rovio equipped with an on-board web camera. The results demonstrate that our mapping algorithm is able to produce a local map with a decent accuracy on the level of a sonar sensor and that our SVM target classifier performs well in detecting and locating bright color objects. We attempted to integrate both into a complete path planning algorithm but were not successful because of our inability to localize accurately.**

I. INTRODUCTION

The objective of our project in essence is to implement SLAM. Our ambitious goal is to have Rovio roam its environment mapping and identifying targets of interest. Our more humble goal though is to do just that but in a much more simplistic indoor environment filled with orange cones as landmarks and targets of interest marked with green tags. Figure 1 shows a typical image, taken using Rovio's on-board camera, of the environment that our Rovio operates in. Though the application and implementation of SLAM have already been demonstrated, we felt that our project is unique nonetheless because of Rovio, which only has a web-cam that we can really use to implement SLAM. As such, there are several stages to our project. They are: mapping, object recognition and path planning, all of which are discussed to a great extent in subsequent sections.

The rest of the paper is divided into four section and is organized as follows. In Section II, we discuss our approach to the first major stage of our project - local 2D mapping from an image. Object recognition is presented in section III. Section IV details both numerical and qualitative evaluation of all of our implemented algorithms. Finally, we close with a few remarks in Section V and give a general idea of how we would have approached path planning if we were able to solve the global mapping problem.

Hung Dang is with the School of Mechanical and Aeronautics Engineering, Cornell University. Jasdeep Hundal and Ramu Nachiappan are with the School of Computer Science, Cornell University. {hvd2,jsh263,rn54}@cornell.edu

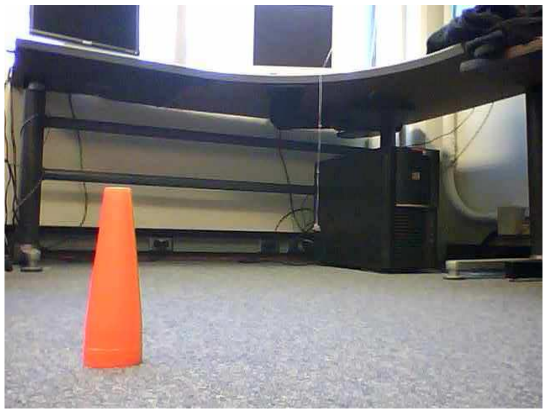

Fig. 1: A typical image of Rovio's environment

II. LOCAL 2D MAPPING FROM AN IMAGE

As mentioned in the introduction, one of the major goals of the project is to map the surrounding environment of Rovio. Using only the camera on-board Rovio and through the application of SURF, a simple carpet segmentation method, and pin hole model, we solve the problem of mapping the free space in Rovio's field of view. The overall architecture of the local 2D mapping is summarized in Figure 2.

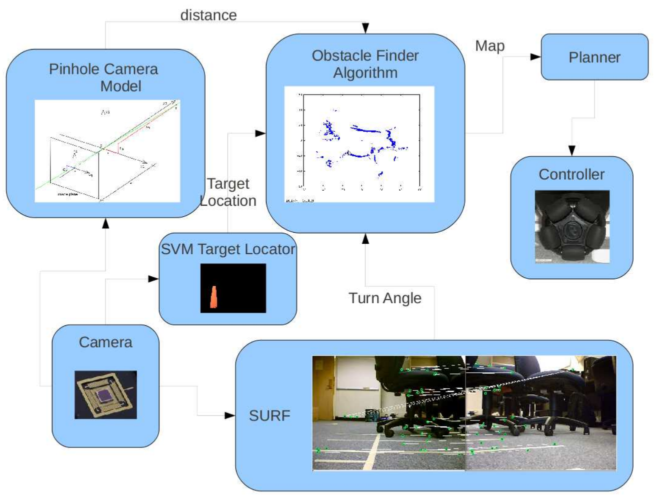

Fig. 2: Overall architecture of local 2D mapping

A. SURF (Speeded Up Robust Features)

SURF is used to detect and describe features in images, much like the SIFT algorithm. Developed in 2006, it is purported to be more robust than SIFT at identifying features, along with clearly being the faster algorithm [1].

We experimented with both SIFT and SURF, and settled on the OpenSURF implementation written by Christopher Evans to determine which was more robust for the environment in our project. We compared the performance of SIFT and SURF across several pairs of images of the Robot Lab taken by the Rovio. It was quickly seen that SIFT matched features in the carpet between the images, and that most of those features were not matched to the correct location in the carpet. SURF picked up at most one to two carpet pixels in each image, and produced a significant number of solid matches otherwise, so it was picked as our feature detection algorithm. An example of such is shown in Figure 3

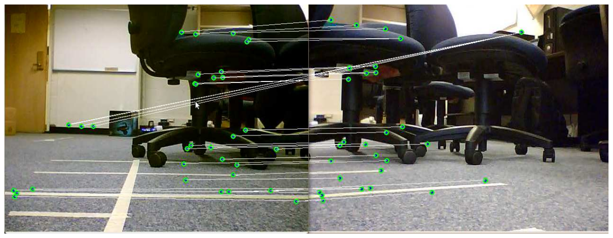

Fig. 3: A typical output of OpenSURF

Despite SURF's apparent robustness compared to SIFT, we were unable to use it in combination with the pinhole camera model to directly map the location of objects. Distance measures using SURF features were not reliable, mostly due to the fact that SURF did not pick up many features that were right along the floor. These would be the ones that would be most accurate with the pinhole model. Most of the features were beyond the four foot of the pinhole camera model. An extension of finding distance change using a set of two images and matching features with SURF was proven useless as nearly zero features along the floor matched between the images taken after forward movement.

However, SURF did prove useful for orienting the Rovio by determining change in angle. The well-matched set of features between two images taken by the Rovio shifting angles gave a reliable pixel shift between the images by taking the median pixel shift among all matched features. With the assumption that the shift in pixels had a roughly linear correspondence to the shift in angle, the change in angle can be computed as $\frac{p_\Delta \theta_w}{p_w}$, where p_Δ is pixel shift, p_w is the pixel width of the images, and θ_w is the field of view of the Rovio in degrees (found through measurement).

B. *Carpet Finder Algorithm*

A major component of our project is obstacle detection. Without it, robot movement would be very restrictive and fragile. There are many techniques that can be used for obstacle avoidance. The best technique depends on the specific environment and the equipment. For our project, the task of obstacle avoidance is executed within an indoor environment hence a carpet segmentation approach is deemed to be the most stable approach.

Since the carpet or floor plane contains more than one pixel color, we can make an assumption that the immediate foreground of the robot is obstacle free. If we were to sample the colors in the lowest part of the image which is the immediate space in front of the robot we could use these color samples and find them in the rest of the image. By searching for all pixels who share the same or similar color to those pixels in this sample space we can theorize that those pixels are also part of the carpet.

This process can be accomplished by performing the following image processing steps. We first sample a small rectangular region in the lowest center part of the image. Pixels within this rectangular region are used to understand what pixel colors are likely to be floor pixels. Iterating through all the pixels inside the sample space, we find the the maximum and minimum pixel value for each of the three color channels. With these ranges known, we iterate through the rest of the image and classify any pixel within these ranges as carpet pixel and those outside of the ranges as obstacle pixels. The result is a binary image as shown in Figure 4.

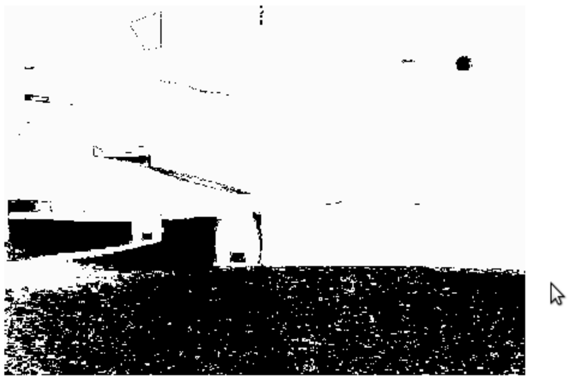

Fig. 4: Binary image

The black pixels in the binary image now represent all pixels in the image that are similar to those found in the sample space. We can see that this works quite nicely to segment out the carpet but the method is not perfect since it does not take into account the effects of shadow and other global features. Thus, we then dilate the image with a 3 by 3 mask of all ones to remove those small noisy holes in the segmented carpet as in Figure 5.

Then we label all of the connected components and discard any connected component with size less than 80 pixels (Figure 6). This removes many false negative carpet pixels. Finally, we iterate through the columns of the resulting binary image saving the lowest row index (height) of the carpet space at that column, which corresponds to the closest object in that orientation. This vector is input to the pin hole model, from which the obstacle boundary can be calculated.

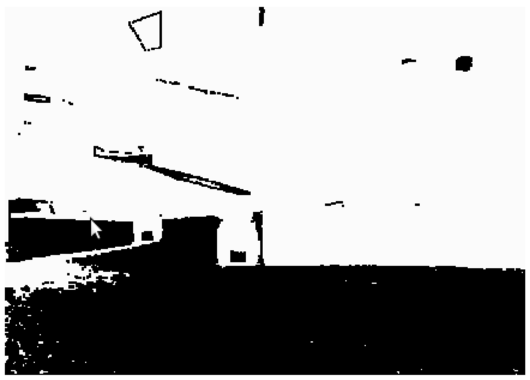

Fig. 5: After dilation

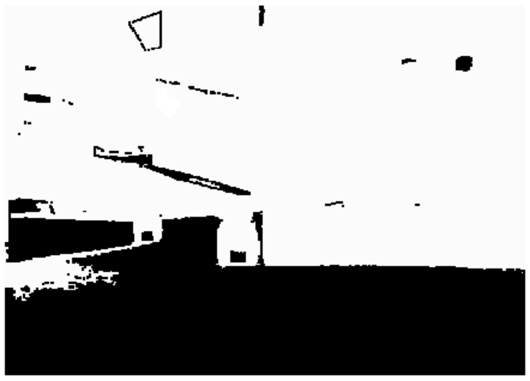

Fig. 6: After removal of small connected components

C. Pin Hole Model

For a given pixel point in an image, we want to know the corresponding coordinates of the location represented by that pixel with respect to the camera. We develop a method to do just that by using the pinhole camera model. The pin hole model describes the mathematical relationship between the coordinates of a three dimensional point and its projection onto the image plane. It is a first order approximation with the assumption that the camera aperture is described as a point without any lenses. It does not take into account lens distortion, which occurs in real cameras. Its accuracy depends on the quality of the camera and decreases from the center of the image to the edges [2]. The geometry related to the pin hole model is illustrated in Figure 7.

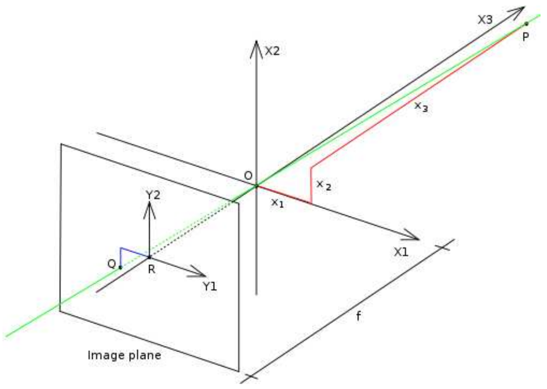

Fig. 7: Pin hole model illustration

Mathematically, the pin hole model is expressed as follows.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -\frac{f}{x_3} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{1}$$

We used a number of calibration images where the distance to the object was known to determine the focal length. After the focal length was determined we solve for x_1 and x_3 in the image above using y_1 and y_2. x_2 is always the height of the camera above the ground. This turned out to be 3.5 or 6.0 inches based on the Rovio's camera position, down and up respectively.

We actually implemented two versions of the pinhole camera model. The first one assumed that the pixel being measured was at the height of the floor and could from a single image determine the x1 and x3 coordinates of the object in relation to the robot. To determine the position of an object, we inputed the bottom most pixel of the object adjacent to a carpet pixel. Hence we can assume it is at the plane of the floor. This was usually determined from the output of our carpet classification code.

The second algorithm could determine the three dimensional position of an object with relation to the robot but required stereo images and corresponding points in both images. Theoretically any point in the image should only be shifted vertically between the images captured with the camera in the up and the down position. The corresponding points in the two images were determined using SURF. This mostly held true, but some horizontal shifting was detected probably due to flaws in the camera arm position. Ignoring the horizontal differences, the size of the vertical shift will depend only on the distance of the object from the camera. The solved version of the pinhole camera model is below:

$$x_3 = f\frac{camera\_height\_up - camera\_height\_down}{y_2^{down} - y_2^{up}} \tag{2}$$

$$h_{obj} = \frac{y_2^{up} x_3}{f} + camera\_height\_up \tag{3}$$

$$x_1 = \frac{y_1^{down} x_3}{f} \tag{4}$$

Our initial goal with the 3D camera model was to be able to map unique landmarks and then use them for localizing the robot. While SURF turned out to match surprisingly few false positives between images, most of the matches were in the background beyond the range of the pinhole camera model. The location of foreground features could be determined to within a few inches in all three dimensions. However unlike the background features, these were much less invariant to movement and could not be matched between translations of the robot. The problem with foreground objects is that the features found on the edges depended on the background behind them. Hence the same location on the cone might have brown pixels from a door behind it when viewed from one location and white wall pixels behind it when viewed from another location. This meant that while a nearby landmark could be placed on the map quite accurately, it could not be matched for purposes of localization after the robot moved significantly. If these issues could be addressed, perhaps through filtering, then this could be a very promising tool to solve the localization problem.

III. SVM TARGET CLASSIFIER

Vision has been our main focus for the entire project, in particular, finding the best learning algorithm to identify orange cones and green tags in an image. We created a dataset of images of cones and boxes with green tags under various lighting conditions and at a number of distances taken from Rovio's camera. We manually segmented the orange cones and green tags for all images in the dataset (Figure 8). We wrote a K Nearest Neighbor classifier algorithm but abandoned the approach since it was computationally very expensive. Consequently, we switched to using Support Vector Machine[3] since it is much faster.

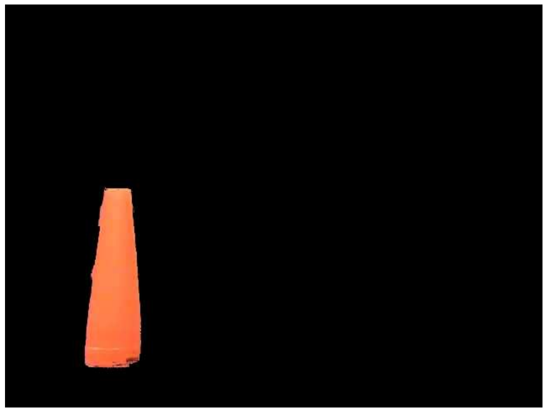

Fig. 8: Positive sample for SVM

For now we are using brightly colored objects such as an orange cone and fluorescent pieces of paper as labels. Since these colors are rare in our testing environment, they relatively easy to recognize using pixel color alone. Our algorithm for locating these objects works on a pixel by pixel level. Our classifier's goal is to identify all the pixels in the unknown image as members of the target object. To train the classifier we manually labeled the pixels that were part of the target object in a series of training images using the magic wand in GIMP. These pixels that are part of the target object were extracted and saved in separate files. For the training of some classification algorithms we also needed the negative pixels so that we can estimate the distribution of negatively pixels as well. This can easily be obtained by subtracting the extracted pixels from the original image to obtain the negative image. Initially, we worked with a kNN classifier to label all the pixels in our image corresponding to the cone. However, this turned out to be quite slow since just one photo has more than 300,000 pixels. So we have switched to using a linear classifier. Using a SVM to classify the pixels results in faster classification with a success rate given by SVM light program to be about 98.5% (Figure 9).

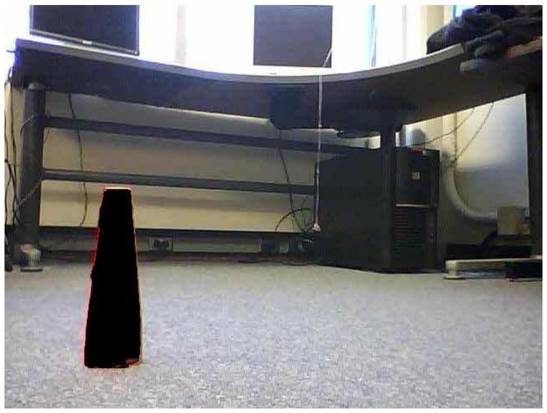

Fig. 9: Segmented output using SVM

IV. EXPERIMENTS

A. Pin Hole Model

We performed a number of experiments to assess the validity of the distance measurement using the pin hole model during calibration. We found the distance measurements usable for navigation within the range of 3-5 feet. The errors exponentially with distance from the robot. Less than 2 feet, the errors were about 1-2 inches. At 3 to 4 feet, these rose to 3 to 4 inches. At 6 feet the errors typically were on the order of 1 foot which about the size of the robot and hence we found at this range the measurements were no longer useful for navigation. Beyond 10 12 feet we found in some cases the errors could exceed 100% on the positive direction. What was clear from the long distance measurements was that the error distributions were not symmetric. This exponential scaling of errors could be explained by the mapping of forward distances on the floor from 0 to infinity onto a finite number of pixels in our image. Hence, while camera noise causing a single pixel error might represent only a few fractions of an inch near the camera, it would mean an error of many feet closer to the horizon.

While initially we expended great effort on calibration of the camera parameters, we realized that due to differences between robots it was a futile effort. Both horizontal and vertical camera angles differed slightly among the robots. This would translate into different horizon heights in our model and as well as some non-linear errors we could not correct. Even the parameters for the same robot changed over time due to mishandling by users. To reduce the impact of bad camera orientations on the 2D pinhole camera model , we resorted to using the camera in the down position where we expected less robot to robot variance.

B. Local 2D Mapping

We tested our local 2D mapping method in several environment settings. An example of such is shown in Figure 10. For each environment, we rotated Rovio about $20°$ and repeated it to make a complete $360°$ scan. At each rotation, we took a picture and used the local 2D mapping method together with the orientation as calculated by SURF and the pin hole model to calculate the x and y location of the obstacles.

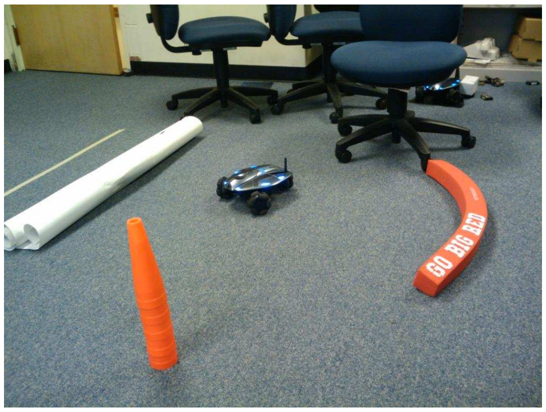

Fig. 10: An experimental setup

The resulting map is shown in Figure 11. Qualitatively, one can see that our developed local 2D method indeed maps the general outline of the environment. The tube of paper is clearly mapped as well as the curved up foam piece. The location and orientation with respect to Rovio of the stack of cones is shown correctly and even the chairs show up in the map. However, our method also picks up random noises and this is expected due to the nature of our segmentation method and the pin hole model approximation. Overall, the error of the map as compared to the actual setting is on the order of half a foot. Some of this error can be attributed to the error in the estimation of Rovio's orientation using SURF. On average, the error in the orientation measured by SURF is about $\pm 2°$

V. CONCLUSIONS

The goal for our project is to simultaneously localize Rovio and map its environment and target objects. In some measure, we succeeded all of the major tasks of SLAM even though we weren't able to implement a full SLAM. We successfully developed a stable method to map all objects within a circle close to the robot for indoor environment using a simple segmentation approach. We were able to train the robot to recognize objects by color, and we have an analytical solution to find the distance to an object that is within four feet of the robot.

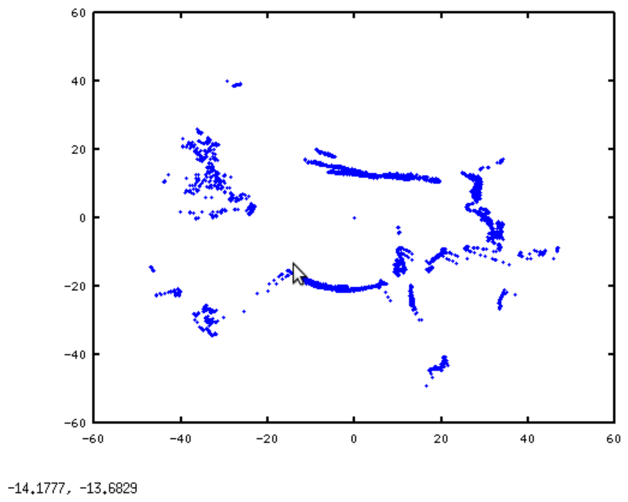

Fig. 11: Mapping result

The next step in regards to recognition is to implement a learning algorithm that fits distance data to the known equation for finding distance. We think that this approach will help us tune the robot to account for any regular noise that causes the result of the equation to deviate from actual distance. For future work, we may use a feature detection approach to recognize more complicated and realistic objects, such as chairs. A definite major task to be completed in the future is the unification of our object recognition and distance finding approaches into full SLAM specifically solving Rovio's localization problem

We intend to use cell decomposition approach to path planning with obstacles represented as polygons and way-points as midpoints of the borders of free space. This would ensure that the robot has as much space as possible to move around. Dijkstra algorithm would be used to generate the shortest path given the starting way-point and the final way-point.

VI. ACKNOWLEDGMENTS

We would like to thank Jonathan Diamond for his assistance and our fellow classmates for the fun time shared in the Rovio lab.

REFERENCES

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool. "URF: Speeded Up Robust Features", *Computer Vision and Image Understanding (CVIU)*, Vol. 110, No. 3, 2008, pp. 346-359.
[2] M. Sonka, V. Hlavac and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Thomson-Engineering; 2007
[3] T. Joachims, "Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning", B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.