Visual Wall Detection for Safe Robot Navigation

Mounika Alluri
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623
ma8658@rit.edu

Abstract—Object detection is done using image classification or template matching methods. In this project, both these methods are investigated to detect glass walls using images. Good accuracy is obtained using both the techniques. The advantages, limitations and reasons for failures for both these methods are discussed.

Index Terms—Glass Wall Detection; Robotic Vision; RGB-D Image classification

I. INTRODUCTION

Robot navigation problems can be classified into indoor and outdoor navigation based on the type of environment. Indoor navigation in hallways often consists of glass walls which can be only be detected by laser and sonar at certain angles of the robot from the wall. They fail to detect transparent and reflective obstacles effectively. An occupancy grid of this environment considers these few observations as noise, representing the glass wall as free space in the map. Hence, if the robot tries to navigate around the glass walls, it is highly likely that it crashes into the glass.

Different techniques have been proposed to mitigate this problem. One of them is sensor fusion where laser scan data is combined with other sensor data to represent glass in occupancy grids. There are studies where the angles at which the glass behaves like a regular obstacle are used to detect glass. Also, the frames of the glass are used to detect the glass. However, the problem has not been solved completely.

The aim of this project is to solve the problem of glass wall detection for robot navigation by using computer vision techniques. Three different approaches are used - Image classification using RGB-D pixels, Convolutional Neural Networks and Template Matching using reflections of the robot in the glass wall. Image classification is done using Support Vector Machines(SVM) and Convolutional Neural Network(CNN) is implemented with Tensorflow library.

II. LITERATURE REVIEW

A. Sensor Failures

Laser is one of the most widely used and reliable sensor in robots, autonomous driving vehicles. The laser sensors in robots detect the distance of the obstacle by using time of flight between emitted and reflected rays. Light is refracted with transparent materials like glass, so the sensor reading gives distance of the obstacle behind it.

Figure 1 shows the assumption of light reflections by lidar and how glass materials deviate from this assumption. It assumes that the incident light is spread around the surface. But in glass, most of the light is transmitted and some of it is reflected back unlike regular opaque materials. Hence, glass and mirrors can be detected by laser only if they strike the surface along the perpendicular direction.

When multiple reflections are received at the same time, the sensor can choose the closest, brightest or average of them. LIDAR reports the range of the brightest reflection. When a diffuse wall is behind the wall, if the LIDAR is perpendicular to glass wall, it gives the range of glass wall as it is brightest. At all other angles, the reflection from the diffuse wall is brighter. Hence, LIDAR gives the range of the diffuse wall.

The reflections of diffuse wall and glass wall have same intensities at critical angle. At these critical angles and few angles around it, either range may be reported. This critical angle is dependent on the distance between the walls, noise on the surface of the glass. [1]

B. Image Classification using Support Vector Machines

Support Vector Machines (SVM) gives a optimal hyperplane with maximum margin. SVM performs well in high dimensional spaces where the dimensions of the space is greater than dimensions of input. Images are high dimensional, hence SVM is the preferred classification technique for images. Optimal Hyperplane can be represented by

$$|\beta_0 + \beta^T x| = 1$$
Distance between the support vectors and hyperplane is given with wider-margin than fit all the examples. It leads to under-fitting as SVM tends to choose hyper-plane if the margin is relatively small. If the value of C is too small, SVM is too high, it will choose to fit all the training data better even and the no. of correctly classified examples. If the value of C is too small, it is another activation function that is used but leads to vanishing gradients problem. The dimensions remain unaffected in this layer.

2) Rectified Linear Unit(RELU) layers: RELU layers are used after every convolutional layer. This layer implements the following activation function.

\[ f(x) = \max(0, x) \]

It essentially takes in the output of convolutional layer and replaces all the negative pixels with zero. The sigmoid function is another activation function that is used but leads to vanishing gradients problem. The dimensions remain unaffected in this layer.

3) Max Pooling Layer: This is done after a series of convolutional and ReLU layers to decrease the no.of dimensions.

4) Fully Connected Layer(FC): It computes the class scores. The output of this layer has dimensions 1 x 1 x n where n is the no.of classes. Convolutional and FC layers transform the input volume using activation functions, weights and biases of neurons. Gradient Descent is used to train the parameters in Convolutional and Fully Connected Layers to minimize the loss function.

5) Hyperparameters: Strides: No.of pixels through which the filter is slipped. In general, a stride of 1 or 2 is used. Padding: The amount of zeros added to the image.

3) Accuracy of Classifier: Confusion matrix is the most common metric used to find the accuracy of classifiers. It is a 2x2 matrix that has the following information of the true condition and predicted condition. Table I shows the columns of confusion matrix as defined in sci-kit learn.

<table>
<thead>
<tr>
<th></th>
<th>Actual= True</th>
<th>Actual= False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted= True</td>
<td>True Positive</td>
<td>False Positive</td>
</tr>
<tr>
<td>Predicted= False</td>
<td>False Negative</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

TABLE I

Confusion Matrix

Other metrics can be derived from confusion matrix.

\[ \text{Accuracy} = \frac{\text{TruePositives} + \text{TrueNegatives}}{\text{Total}} \]

C. Convolutional Neural Networks

Convolutional Neural Networks are modified Neural Networks. In Neural Networks, neurons in one layer are connected to all the neurons in the previous layer. In CNN, expect in the Fully Connected(FC) layers, neurons in one layer are not connected to all neurons in the previous layer.

1) Convolutional layer: Convolutional layer slides a filter across the image. Generally filter of odd dimensions(width and height) are used. The depth of the filter is equal to depth of the image. The filter is slipped across the input image to compute dot product at all the possible locations. Each filter results in a 2D activation map. If there are f filters, they produce f number of 2D activation maps which are stacked along the depth dimension. Hence, if the input has dimensions NxNx3 and f filters are used, the output has dimensions NxNxf.

2) Rectified Linear Unit(RELU) layers: RELU layers are used after every convolutional layer. This layer implements the following activation function.

\[ f(x) = \max(0, x) \]

It essentially takes in the output of convolutional layer and replaces all the negative pixels with zero. The sigmoid function is another activation function that is used but leads to vanishing gradients problem. The dimensions remain unaffected in this layer.

3) Max Pooling Layer: This is done after a series of convolutional and ReLU layers to decrease the no.of dimensions.

4) Fully Connected Layer(FC): It computes the class scores. The output of this layer has dimensions 1 x 1 x n where n is the no.of classes. Convolutional and FC layers transform the input volume using activation functions, weights and biases of neurons. Gradient Descent is used to train the parameters in Convolutional and Fully Connected Layers to minimize the loss function.

5) Hyperparameters: Strides: No.of pixels through which the filter is slipped. In general, a stride of 1 or 2 is used. Padding: The amount of zeros added to the image.
D. Template Matching

The reflections in the glass can be used to detect glass in the image using template matching. It is often used to detect road signs. Preprocessing of the RGB images is done before template matching. If template matching is done on RGB image, it will be color sensitive and fails to detect the sign for slight changes in lighting.

RGB images are first transformed into HSV color space. A binary image is formed by thresholding the Hue, Saturation and Value channels. This segmentation is not affected by variations in light. This gives a contour of the sign to be detected. This region is filled with white pixels so that the complete sign region is detected and not just the contour. The sign is extracted from the segmented image by finding the maximum and minimum values with labels along both the dimensions of the images. This extracted sign is used for template matching.

Normalized cross-correlation computes the degree of similarity between two images.

Normalized cross-correlation of two images $f$ and $g$ is computed by using the following formula

$$\gamma(u,v) = \frac{f - \bar{f}}{\sqrt{\Sigma(f - \bar{f})^2}} \frac{g - \bar{g}}{\sqrt{\Sigma(g - \bar{g})^2}}$$

$\bar{f}$ and $\bar{g}$ represents the mean of intensities of images $f$ and $g$ respectively.

The first and second terms represent the normalized intensities. To normalize, mean is subtracted and divided. The range of normalized cross-correlation is $(-1,1)$ as it computes the dot product between two unit vectors. [4]

III. METHODS

A. Kinect RGB-D Images

The images are obtained using iRobot Create2 mounted with Microsoft Kinect XBOX 360 (Kinect v1) by navigating the robot around the hallways. Figure 3 shows the robot used for the project. ROS (Robotic Operating System) is used to interface with the hardware. This uses publisher-subscriber system to communicate between images. Kinect v1 has two cameras for RGB, depth of field and an IR projector. RGB and Depth cameras give images of resolution $640 \times 480$. Figure 4 shows the internal layout of Kinect.

RGB and Depth images are recorded for 17 s using rosbag record command on the topics `/camera/rgb/image_rect_color` and `/camera/depth_registered/hw_registered/image_rect`. These images are saved in bag files as ROS image type. Rosbag play command is used to play the images, while subscribers to the same topics are used to save the images. RVIZ is used to visualize the images when recording and playing.

Brighter the pixel i.e. higher the value of pixel, the farther it is from the Kinect. The Kinect fails to detect objects that are too close or too far. A pixel with zero intensity corresponds to these locations at which the data is unavailable. A distance of 1 m is represented by an intensity of 5000 on the depth image.

CVBridge library acts as an interface between ROS and OpenCV. It is used to convert ROS image type into CV image type. RGB images are saved as 8 bit .jpg files and depth images are saved as 16-bit .png files. Listing 1 shows the python code to save depth images in OpenCV. The depth image is first
Fig. 5. Image Labels: Green squares represent wall pixels and Red squares represent not wall pixels

converted to numpy array and the pixels are scaled to 0-255 to save them as grayscale images.

Listing 1. saving depth images

```python
d_array = np.array(depth_image, dtype=np.float32)
d_array[np.isnan(d_array)] = 0
cv.normalize(d_array, d_array, 0, 1, cv.NORM_MINMAX)
cv.imwrite(image_name, d_array*255)
```

B. Label Images for Classification

LabelMe is the most widely used tool to label images for computer vision applications. For this project, OpenCV mouse click events are used to label the images into two classes - wall pixels and not wall pixels. Each left or right mouse click event registers the 13 x 13 region around it as wall pixels (class 1) and not wall pixels (class 0) respectively. Figure 5 shows the labelling for an RGB image. The depth registered image that corresponds to the current RGB image is loaded and depth pixel information of the selected pixels is extracted and appended to the rgb feature vector to form a RGBD feature vector.

C. Training and testing

Sci-kit learn is used to implement SVM classification. 1000 feature vectors are used for training SVM, 500 of them from class 1 and 500 from class 0. The parameters of SVM are penalty term and kernel functions. C is varied from 0.01 to 1000 and the three kernel functions-linear, gaussian and polynomial are used to compare.

D. CNN Architecture and Tensorflow

Tensorflow library is used for implementing Convolutional Neural Networks. and Figure 6 shows CNN Architecture for RGB-D classification. A simple architecture that is used to classify CIFAR -10 data sets is used for RGB-D image classification.

INPUT - [CONV- RELU - POOL]*2 - FC- SOFTMAX

The architecture is also shown in Figure 6 Gradient Descent Optimizer is used.

E. Template Matching

A bright green wallpaper is used on the laptop. When the robot navigates in the direction of the glass walls, the reflection can be seen in the glass walls. Regular RGB images from mobile camera are used for this project. Figure 7 shows reflection of laptop on the robot in the glass wall.

MATLAB is used as it has functions to implement template matching. imfill is used on the binary image to get a perfect contour. The function normxcorr2 is used to find 2D normalized cross-correlation between the template and the original image. A tiny green image cropped from the wallpaper is used as template.

IV. Results

A. SVM

The classifier is tested on images that have glass in a different orientation and under different lighting conditions. For the accuracy metric, the predicted class labels are compared with the ground truth labels and the percentage is calculated. The penalty term is varied from 0.01 to 1. The four kernels-linear, radial basis function, polynomial are used. An accuracy of 88% is obtained using SVM Classifier with linear kernel. Table III shows the accuracy of SVM classification. Table IV shows the accuracy of SVM with different kernels. There are many pixels in depth images with intensity 0. This is because the laser cannot sense anything that is too close or too far. Glass pixel is always has intensity 0 in depth images. This can make the classifier based. Most of the training images have glass pixels on the right. The classifier is not able to differentiate between the pixels that are behind and the glass.
This leads to false positives and false negatives. The images do not have much color information, they are mostly gray, white and black. The robot has a glass on it as shown in figures 3 and fig:label. This sometimes overlaps with the glass walls and sometimes this is detected as noise. Some part of this glass may overlap with glass pixels in the images and some region with non glass pixels.

**B. Template Matching**

Figure 8 shows the normalized cross-correlation between the input image and the template. The peak corresponds to the region where the reflection is found. It implies that a glass wall is present in front of the robot and should be avoided. Table IV-B shows the results of template matching.

<table>
<thead>
<tr>
<th>Total images</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>6</td>
</tr>
<tr>
<td>True Negatives</td>
<td>3</td>
</tr>
<tr>
<td>False Positives</td>
<td>1</td>
</tr>
<tr>
<td>False Negatives</td>
<td>1</td>
</tr>
</tbody>
</table>

For the template matching to work the robot should be able to see the reflection. When there is too much light in front and before the robot,

**C. Integration with the robot**

The techniques discussed above can be integrated with the robot.

**V. CONCLUSION**

The classification techniques give good accuracy. But they can fail for different datasets which differ significantly from the training dataset. Indoor environments differ significantly from place to place.

The template matching method is more reliable than classification. In this project, the template used can be easily detected when there is light before and away from the lab. The system can be made robust to illumination in the environment, though it fails under occlusion. The template matching can be used to detect the angle of the robot from the wall. This approach can also be used to detect whether the robot is going towards the glass wall or away from it.

**REFERENCES**


