A

Project Report

On

**Credit Card Number Detection**

Submitted by

**Gargi Thakker L20371763**

1. Run program and check the outputs.
2. Write detailed steps used for text detection.

* Read the card image : This is done by simple use of matlab function, ‘imread’. Then, coloured image is converted to grey scale using function ‘rgb2gray’.
* Divide and consider only bottom half of image: Since, we know that the credit card number lies in the bottom half usually, this step lets us obtain the position of the area of interest (i.e., the card number). So we divide it in two equal parts and considered only bottom half. Also it will help to reduce pre-processing work. As we are considering only half image.
* Remove noise by applying 3x3 mean filter : Once the area of interest is obtained, the mean filter is applied to get a noiseless image.
* Apply sobel edge detector: This is used to detect the edges in the card.
* By applying fspacial(sobel ) we are doing sobel horizontal edge-emphasizing filter. creates a two-dimensional filter h of the specified type. fspecial returns h as a correlation kernel, which is the appropriate form to use with imfilter. type is a string having one of these values.
* We can also apply bwlabel(EdgeImage) edgepixel to calculate how many candidate regions are there. By setting threshold we segment the image.
* h = fspecial('gaussian', hsize, sigma) returns a rotationally symmetric Gaussian lowpass filter of size hsize with standard deviation sigma (positive). hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [3 3]; the default value for sigma is 0.5.
* imgaussfilt will filters image with a 2-D Gaussian smoothing kernel with standard deviation of 2.

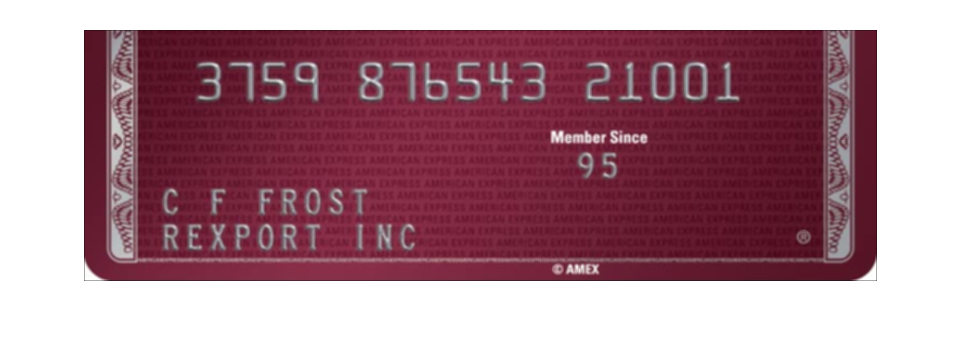
1. Test the program using the following credit card images: 3.jpg, 25.png, 26.jpg, 40.jpeg, 45.jpg and show the detection outputs.

* The outputs are given below :
* 3.jpg :

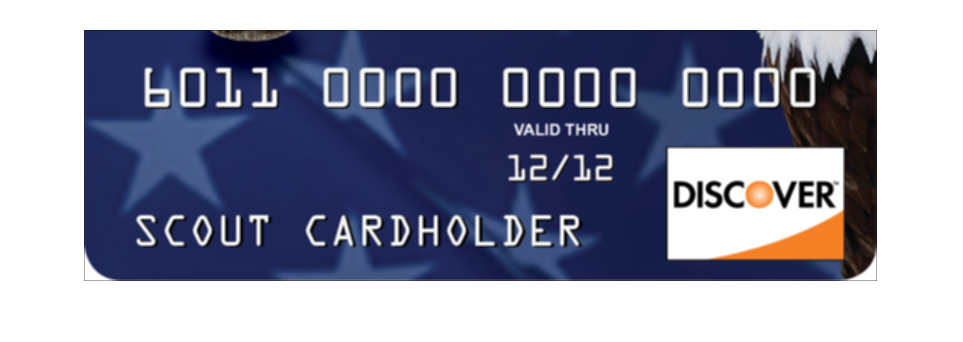
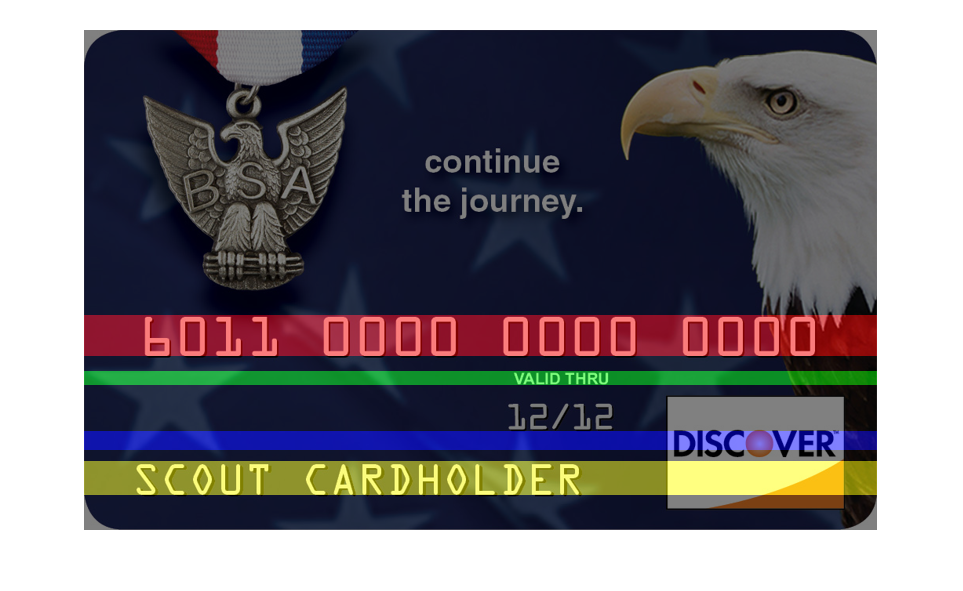




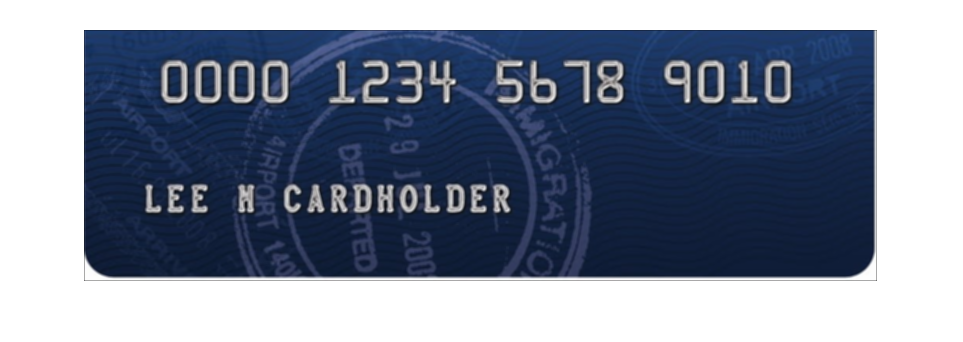
* 25.jpg :



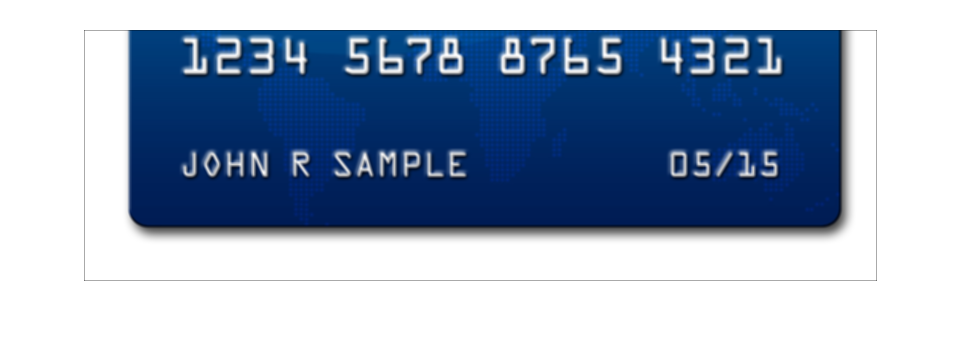
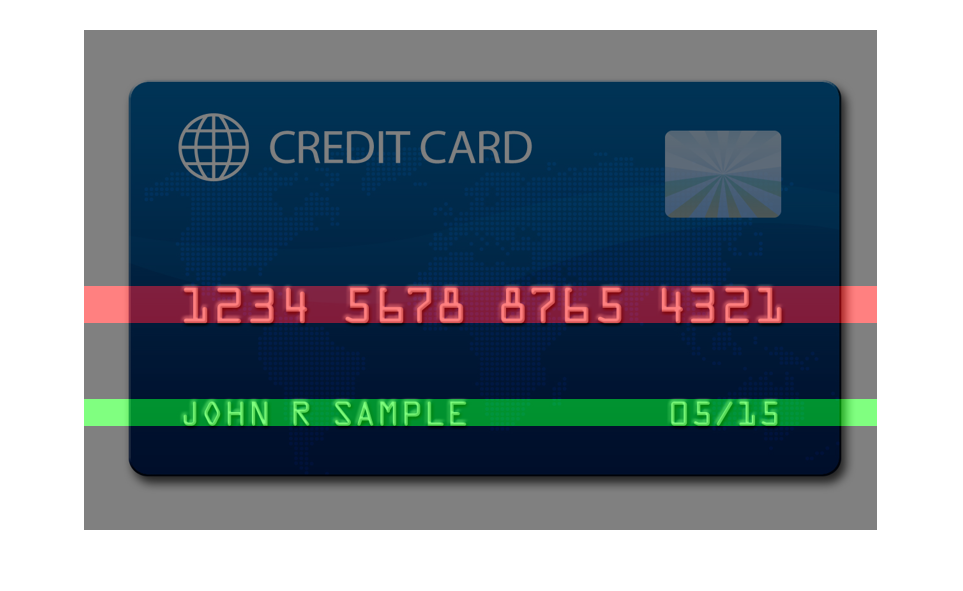
* 26.jpg :



* 40.jpg :

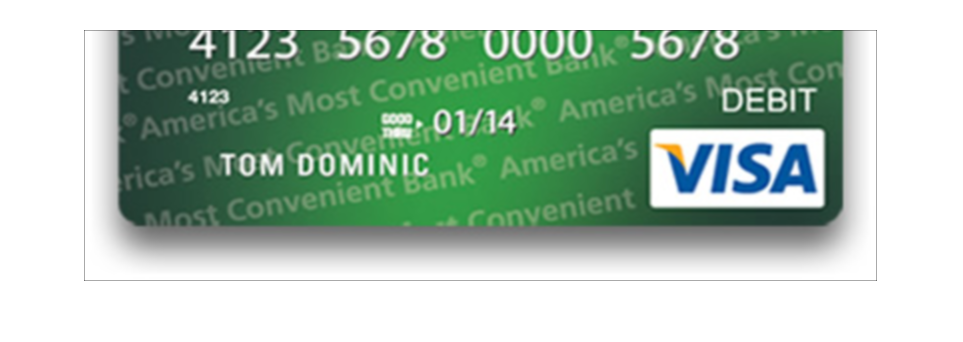


* 45.jpg :



1. Test the program using the following credit card images: 14.jpg, 29.jpg, 30.png, and compare the performance with previous step’s. Discuss how to improve the performance.

* 14.jpg :



* 29.jpg :



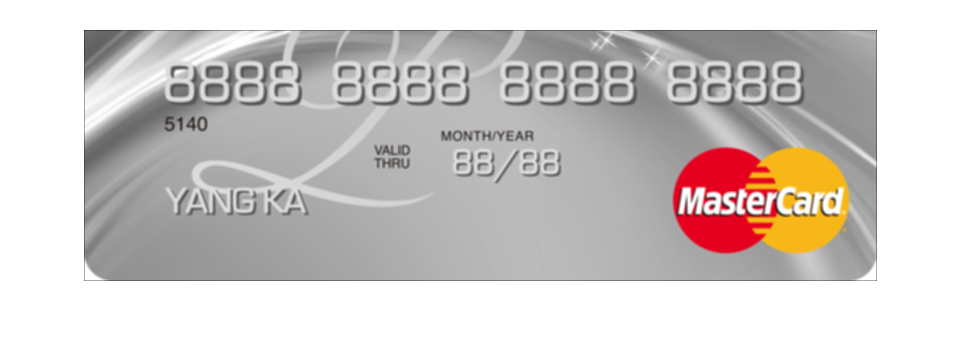
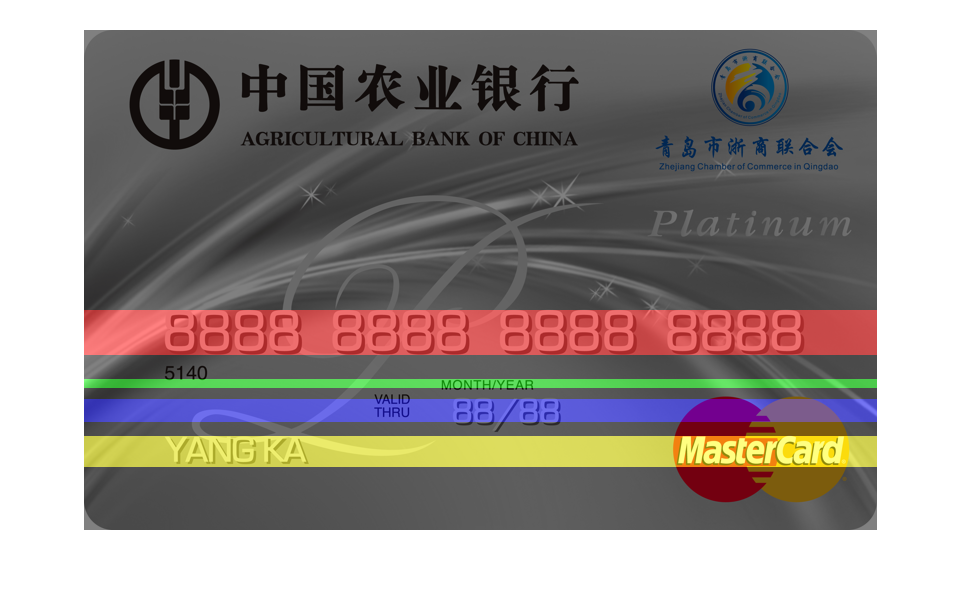
* 30.jpg :



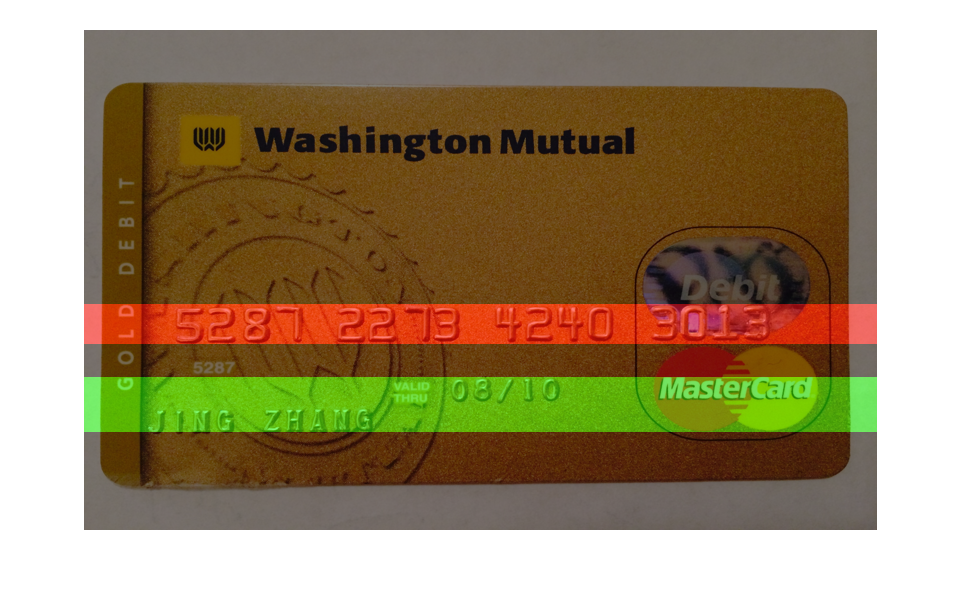


1. Test the program using the following credit card images: 18.jpg, 37.jpeg, 49.jpg, and compare the performance with previous step’s. Discuss why the program can detect text region but cannot extract numbers correctly.

* 18.jpg :

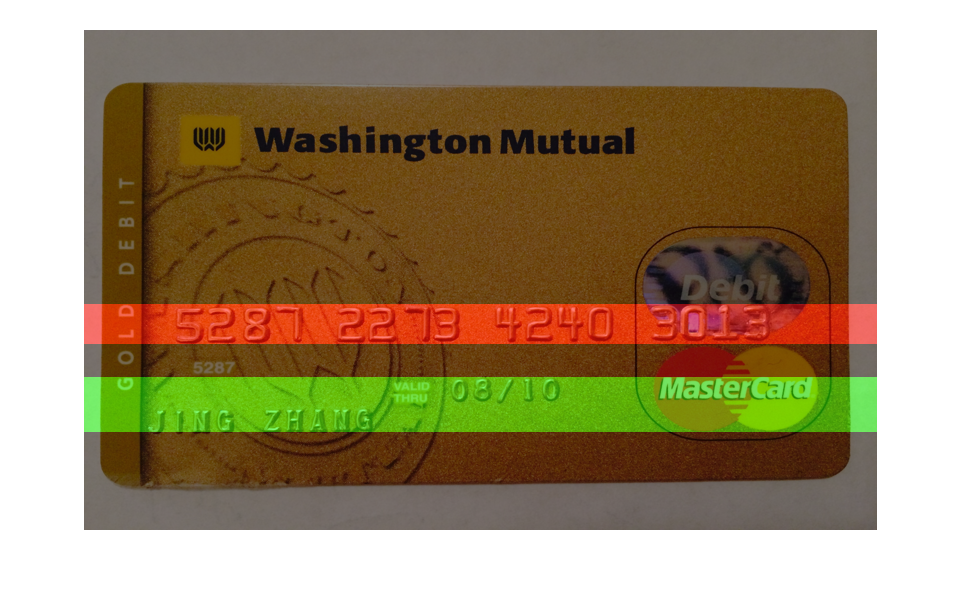


* 37.jpg :





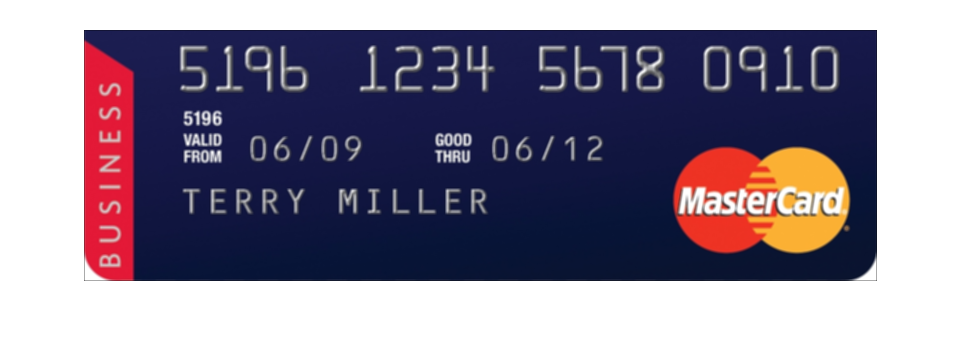
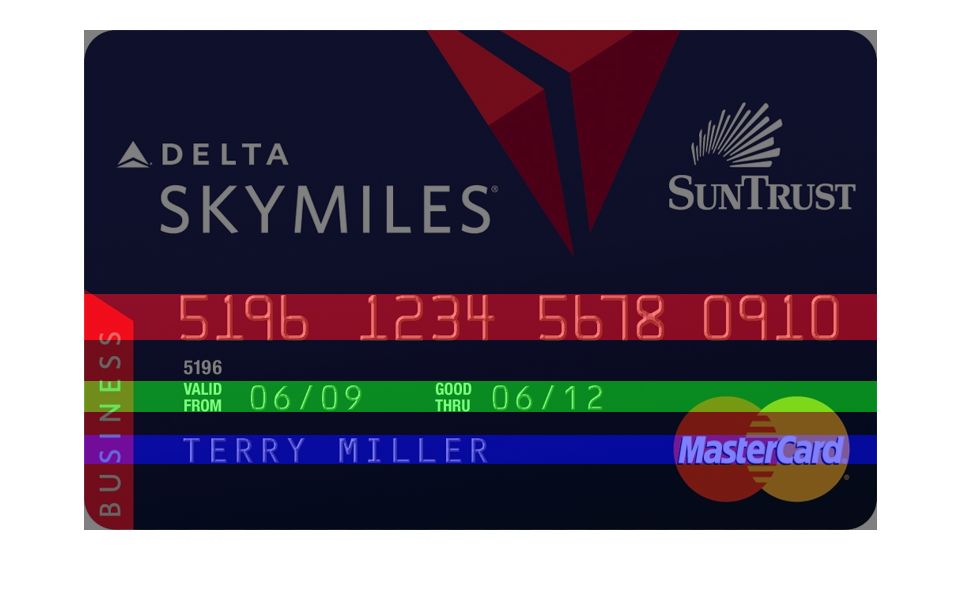
* 49.jpg :



Hence, it is observed that all the images other than img 14 gives a good output.

1. Test the program using the following credit card images: 6.jpg, 36.jpg, and compare the performance with previous step’s. Discuss why the program can detect text region correctly.

* 6.jpg :



* 36.jpg :





Here even if background noise is greater or we can find the candidate region correctly. Candidate region height is perfectly match here.

1. Discuss which step(s) is/are the hardest for you and why ?

* The way the steps were explained, it as no part that was found so hard, specifically. But, applying bwlabel, was just a bit tricky.
* Calculating the ledge value is difficult because that value does not give the efficient output for every image we pass.