Car Damage Classification using Machine Learning Algorithms

Name: Soorya Nivedha Ashokan
Team: Individual Project

Objective of the Project:

I would like to create hassle free model to classify if the defects on the rental car is made by the user or they were preexisting on the car by giving the image of the car as input. This helps the user to avoid being charged for the defects that are not made by the user. Also, it helps the rental company to quickly verify the damages on the returned car.

GitHub Page: Please find the entire project uploaded in the following link
https://github.com/sooryanivedhaashokan/Car-Damage-Classification

Introduction

In this project, I would like to create classification model for identifying the damage on the surface of the car that we rent was already on the car or we made the damage. Most of us love travelling and when we plan for a trip to far destination. It is always convenient to reach the destination by flight and rent a car at the airport for comfortable commute during the trip. While renting the car, if we missed to notice any pre defects on the car surface, we will end up paying additional charges for the damage which we were not responsible for, while returning the car. In the internet I could find many projects on identifying the damage on the surface of the car. But as a consumer, we don't need to identify the pre defects of the car as we won't be buying them, instead we need a quick model to identify if the defects are done by us or not. Hence, I tried to use mean square error and structural similarity index of the image of the car before renting (IMAGE1) and image of the car while returning (IMAGE2) to classify the damaged car and the good car [1]. In this case, we need to make sure that the IMAGE1 and IMAGE2 are of the same car, else there is no use for finding the similarity index. Hence, I have used the svc model to detect the license plate of the car. Though this method gives a general idea of dissimilarities of the images, the accuracy of the prediction is low. Because this method primarily depends on the angle the picture is taken, also if the image of car has any dust or water, this will increase the mean square error, which is not our expectation. Hence, I tried to use histogram of gradient feature algorithm to produce the feature vectors of the image and then feed these feature vectors to image classification algorithms like SVM to produce better results. At this point I realized the significance of finding SSIM.

As mentioned previously, this report has three sections, which describes how the problem has been approached, then the issues that has been realized while working on each section and how to proceed further. Here, section1 explains the mean square error and structural similarity index calculation of the image. Then section 2 talks about the approach and algorithms used to detect the car number plate. Further section 3 summarizes the HOG method used to extract the features of the car image. Also, each section has is corresponding results obtained.

1. Mean Square Error and SSIM:

Digital image composed of finite number of elements where each have location and value called pixels. An image may be defined as two dimensional function f(x, y) where x and y are spatial coordinates and amplitude of function f at any pair of coordinate (x, y) is called the intensity or gray level of image at that point when x, y and amplitude value of f are all finite value, is called as digital image. Mean Square Error and SSIM are an objective analysis of image quality assessments. Here two images are compared pixel by pixel from left to right and top to bottom through a row and column. Then MSE is calculated by averaging square of difference between error of original and test distorted image. If x and y are two non -negative gray scale images, then MSE is calculated using,

$$\mathrm{MSE}(\mathbf{x}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2.$$

Thus, **MSE** represents the cumulative squared error between the test and the original image, ie the error signal $e_i = x_i - y_i$,. The lower the value of MSE, the lower the error and higher the chance for both images to be similar.

Suppose that x and y are local image patches taken from the same location of two images that are being compared. The local SSIM index measures the similarities of three elements of the image patches: the similarity l(x, y) of the local patch luminance (brightness values), the similarity c(x, y) of the local patch contrasts, and the similarity s(x, y) of the local patch structures[1]. The SSIM index is defined by following equation,

$$S(\mathbf{x}, \mathbf{y}) = l(\mathbf{x}, \mathbf{y}) \cdot c(\mathbf{x}, \mathbf{y}) \cdot s(\mathbf{x}, \mathbf{y})$$

$$S(\mathbf{x}, \mathbf{y}) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right) \cdot \left(\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) \cdot \left(\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}\right)$$

If the input images are identical, the index is 1; and if they are uncorrelated the index is very small. If one of the input images is considered the reference, the index gives the quality of the other image as compared to the reference. To capture the (dis-)similarity of the images better, the index is computed on windows sliding over the two images. Then all the resulted indexes are

averaged to give one index. In that case, x and y represent windows (located at the same places) of the two input images. μ x and μ y are the average intensities of pixels in x and y, with standard deviations σ x and σ y. However, the following results were calculated using the skimage python package [3] to measure the MSE and SSIM of the test image and original images in the list.

Results for Section1:

To find the structural difference between the car image before renting and the car image while returning. When SSIM for two images are similar then MSE helps to find the difference and vice versa. If the rental agency has the already taken the images of the cars they have, then when the customer returns the car, they can input the returned car image as test image and find the dissimilarities on the car surface. Below image shows the MSE and SSIM of the images in the dataset with respect to test image given.

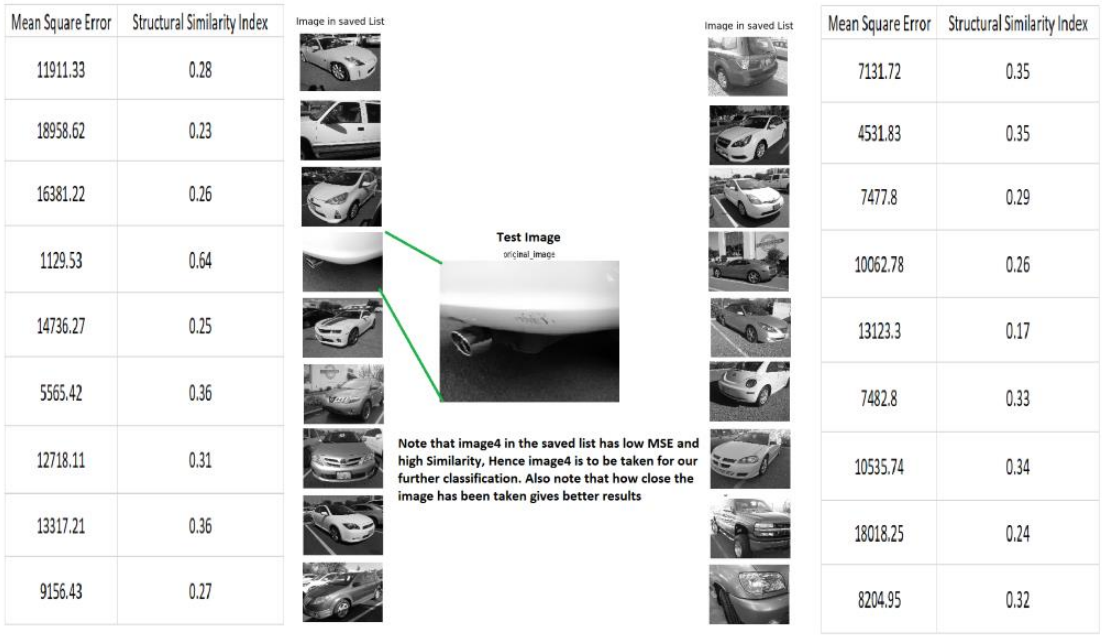

Figure1: Car images with MSE and SSIM values

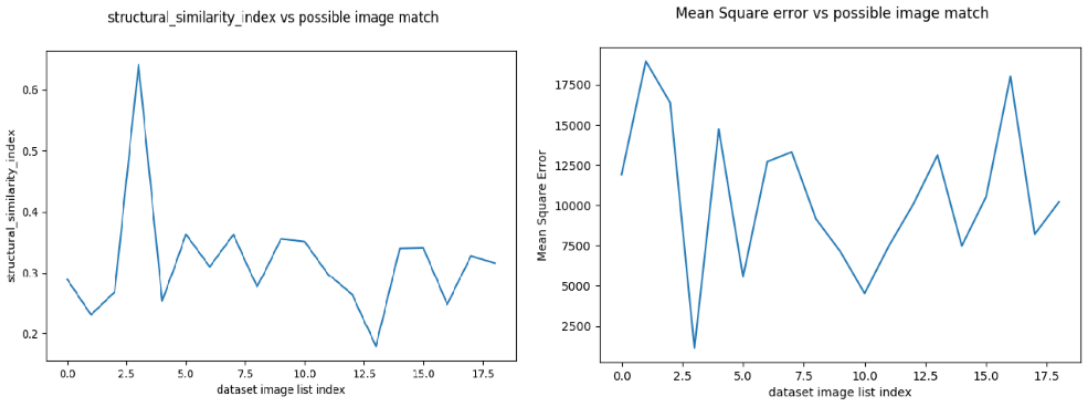

Figure2: SSIM and MSE comparison results

Observation:

Note that in the figure1 image4 matches better with the test image given. It is likely to be the same car but has some defects which can be concluded by the MSE and SSIM value.

Issues:

1. In order to use this method effectively, we need to take the image of each car in the agency from all the angles (minimum of front, back and two sides of the car). In this project, since I could not collect 4 sides of the image of a same car with and without damage. I have tried to calculate the results using single image. Accuracy can be higher if we input the tool with proper dataset
2. Image quality and alignment plays a major role in making any conclusion based on this method.

2. Car Name Plate detection:

Automatic vehicle license plate detection is a key technique in many of the security and traffic related applications. There are various methods, techniques and algorithms have been developed for license plate detection and recognitions. In this project, we know that we need to compare images of the same car before renting and after returning. One way of identifying the car is by detecting its license plate and then look for any damages or dissimilarities between the car images. Below is the flowchart for the approach used to detect the letters in the license plate of the car.

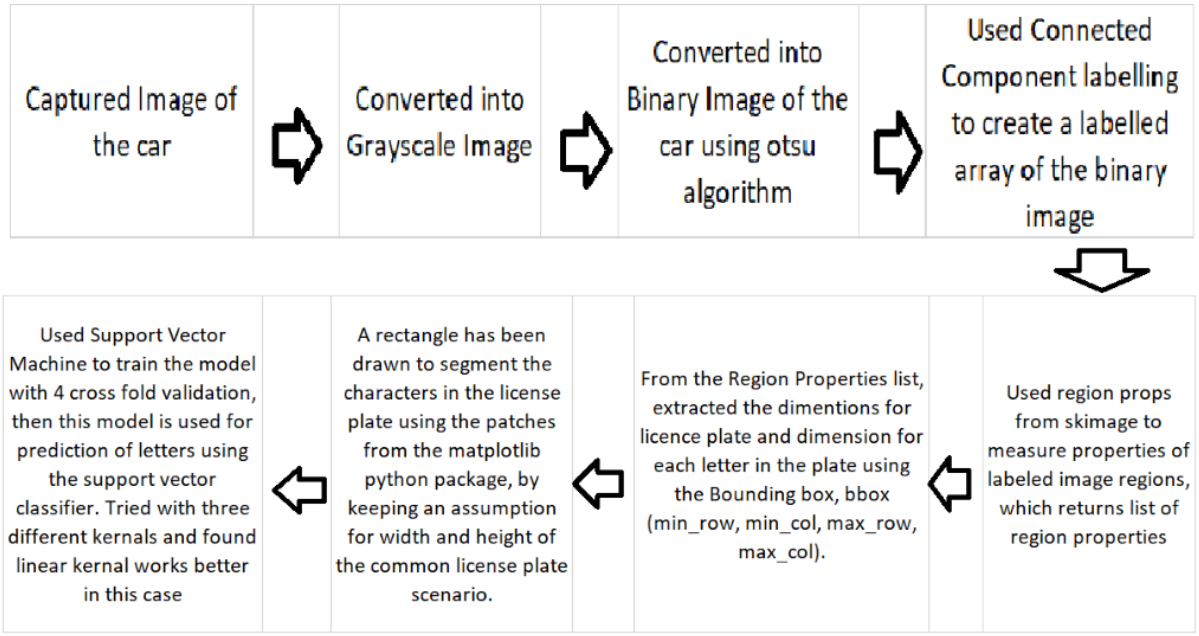

Figure3: Flowchart

Thresholding is the simplest method of image segmentation[4]. From a grayscale image, thresholding can be used to create binary images. The Otsu algorithm returns a single intensity threshold that separate pixels into two classes, foreground and background. The **Connected Component Labelling** is used to create a labelled array of the binary image. Two pixels are connected when they are neighbors and have the same value. In 2D, they can be neighbors either in a 1- or 2-connected sense. The value refers to the maximum number of orthogonal hops to consider a pixel/voxel a neighbor. In order to train the model, **Support vector machines (SVMs)** are a set of supervised learning methods used for classification, regression and outliers detection. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. It is also versatile, since different Kernel functions[5] can be specified for the decision function. I have followed the blog [6] to complete this part of the project.

Results for Section2:

Image Plots:

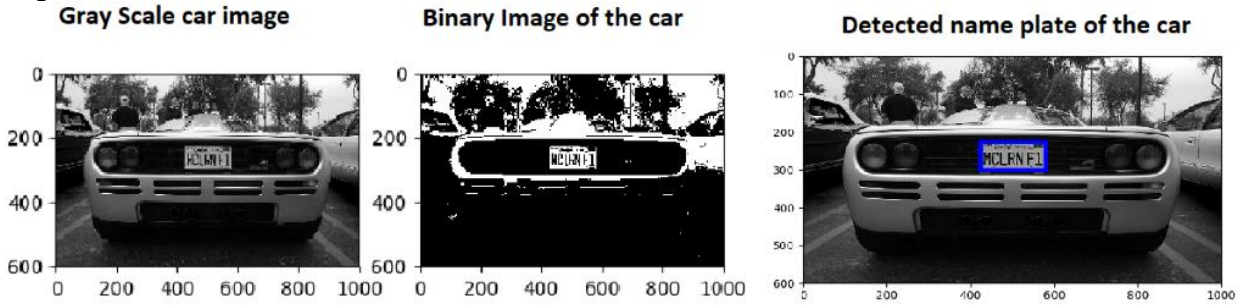

Figure4: Image of the car used for License plate detection

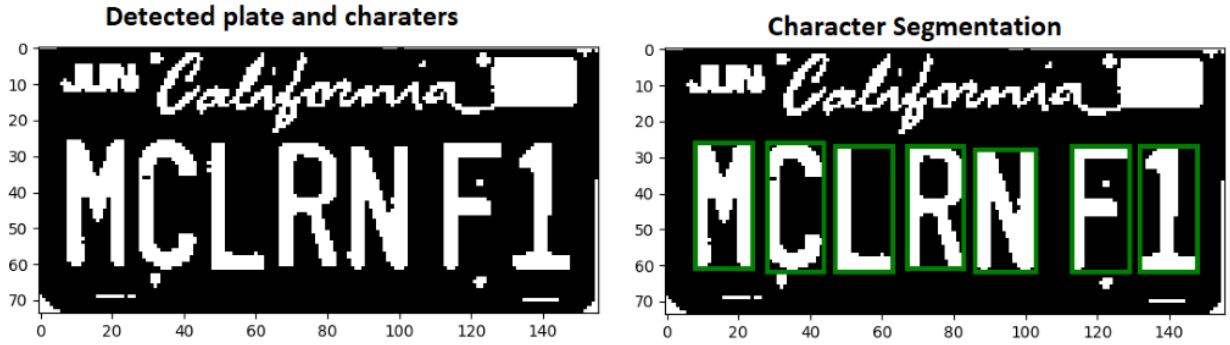

Figure5: Segmentation of characters in the License plate for prediction

Python Code Results:

Accuracy with poly kernel and 4 folds = [0.81372549 0.85294118 0.85294118 0.82352941]
Accuracy with rbf kernel and 4 folds = [0.92156863 0.94117647 0.94117647 0.98529412]
Accuracy with linear kernel and 4 folds = [0.96078431 0.99019608 0.97058824 1.]
Classified array list = [['M'], ['C'], ['L'], ['R'], ['F'], ['1'], ['N']]
Predicted license plate letters = MCLRF1N

Observation and Issues:

1. Arrangement of characters in the list were not proper, probably need to have a global copy of the column list after segmentation
2. Dimensions of the license plate should have a standard form in order to use this model effectively. Figure6 shows the results obtained with a different car with small size license plate. Note that the model with given dimensions detects 3 different spots.

Figure6: Wrong license plate detection

3. Histogram of Oriented Gradients

The image gradient vector is defined as a metric for every individual pixel, containing the pixel color changes in both x-axis and y-axis. To compute the gradient vector of a target pixel at location (x, y), we need to know the colors of its four neighbors. The Histogram of Oriented Gradients (HOG) is an efficient way to extract features out of the pixel colors for building an object recognition classifier. Below is the flowchart of HOG algorithm.

Preprocess the image, (resizing and color normalization)	⇨	Compute the gradient vector of every pixel	⇨	Divide the image into many 8x8 pixel cells. In each cell, the magnitude values of these 64 cells are binned and cumulatively added into 9 buckets of unsigned direction	⇨	Slide a 2x2 cells (thus 16x16 pixels) block across the image. In each block region, 4 histograms of 4 cells are concatenated into one-dimensional vector of 36 values and then normalized to have an unit weight.	⇨	Final HOG feature vector is the concatenation of all the block vectors. It can be fed into a classifier like SVM for learning object recognition tasks.

Figure7: HOG algorithm Flowchart

I tried to use histogram of gradient feature algorithm to produce the feature vectors of the image and then feed these feature vectors to image classification algorithms like SVM to see better results. As I could collect only very few images (18 images) of the cars with and without damage. The analysis that is been done here is very limited and the accuracy obtained is fair.

Image Plot Results:

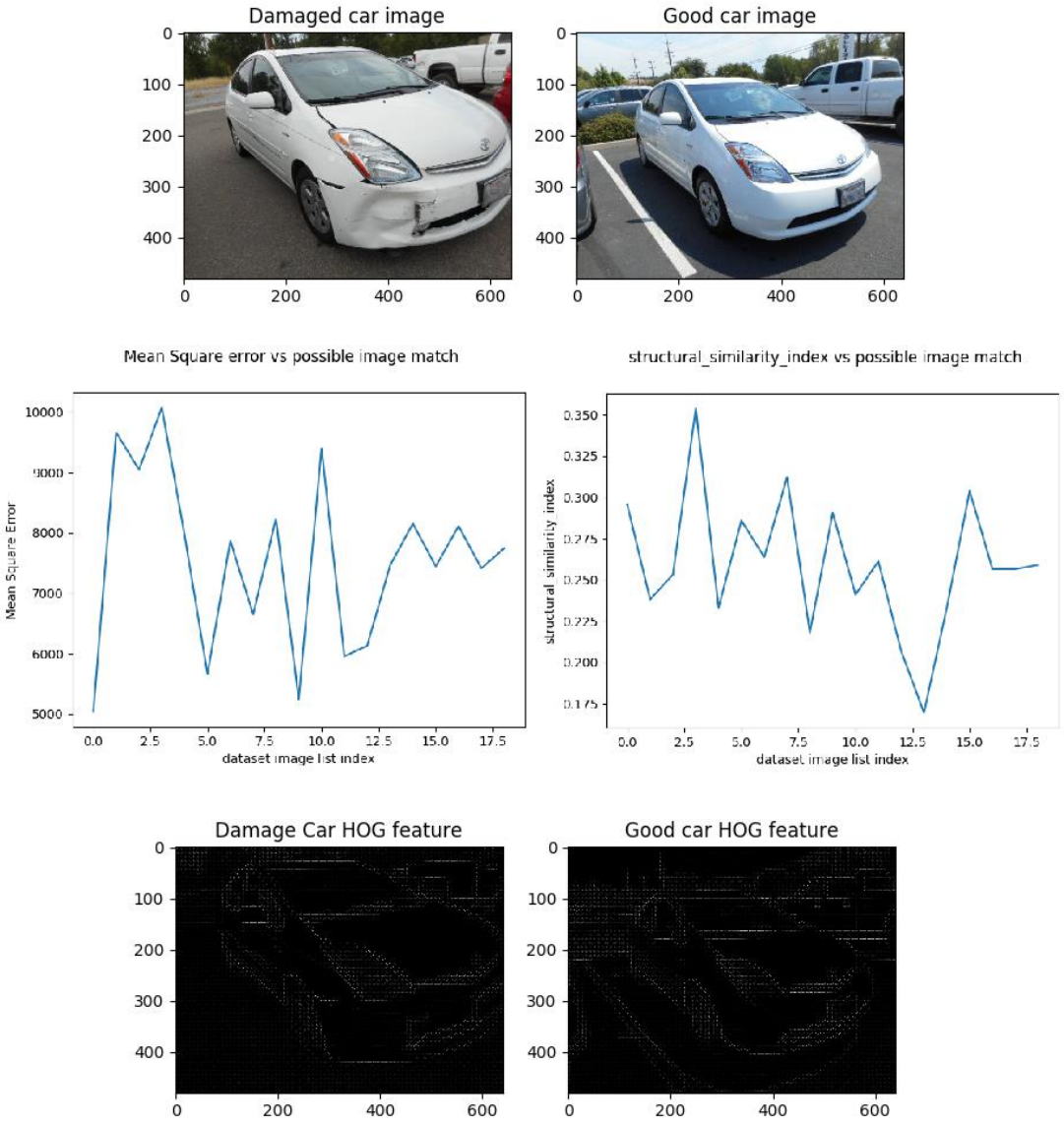

Figure8: Damaged car vs Good car

Observation:

Note that here MSE and SSIM does not give clear idea for car image number 7 with and without damage, hence tried to train the model with HOG features to identify if the car rented and returned are same or not. But the prediction accuracy is low because of very few training samples used.

Python code results:

Total number of damaged car images: 18
Total number of good car images: 18
18.99 Seconds to extract HOG features...
Size of the Training data set: 28
Size of the Testing data set: 8
Classification accuracy using the Linear Support Vector Machine Classifier:
1.17 Time in sec to train Support Vector Machine Classifier...
Test score of my Support Vector Machine Classifier = 0.625

Conclusion:

In this project, I tried to approach the problem based on my understanding by learning the course EE646. I have found many advanced approaches available online like mask CRNN to detect the damage on the surface of the car more accurately. With this project, I understood that when we use image as a data to the machine, we need to realize the significance of image processing and dataset. We need to consider two significant parameters such as,

1. SSIM between the images taken before renting and while returning, ie., park the car in the same exact way and location, this is possible if we rent and return in same place
2. We need lot more image dataset of car before damage and after damage for training the machine

Since I could only collect very few images of the car before and after damage, the analysis I could do is limited. In the future, need to explore more advanced neural networks to approach this problem with enough training data to get best results.

References

[1] https://pdfs.semanticscholar.org/dbe6/3a546c9931af7f97c1838d693f4748a0d124.pdf
[2] Z. Wang and A. C. Bovik, "Mean squared error: love it or leave it? - A new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98-117, Jan. 2009.
[3]. https://scikit-image.org/docs/dev/api/skimage.measure.html
[4]. https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_thresholding.html
[5]. https://scikit-learn.org/stable/modules/svm.html#svm-kernels
[6].https://blog.devcenter.co/developing-a-license-plate-recognition-system-with-machine-learning-in-python-787833569ccd
[7]. https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html
[8]. https://export.arxiv.org/pdf/1804.11207
[9]. https://ieeexplore.ieee.org/document/8260613