Assignment 1

Ronak Doshi (IMT2017523)

1 Road Lane Detector

When we drive, we use our eyes to decide where to go. The lines on the road that show us where the lanes are act as our constant reference for where to steer the vehicle. Naturally, one of the first things we would like to do in developing a self-driving car is to automatically detect lane lines using an algorithm. To develop a self-driving car we need to be able to detect road lane lines so that the car is able to drive on the correct track.
In order to do this we use computer vision techniques. We try to detect road lanes by following this particular steps:

- Lane line clearance
- Canny edge detection
- Region of interest
- Hough Space

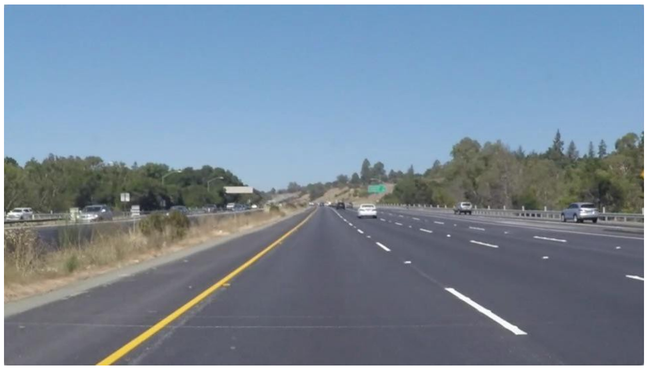
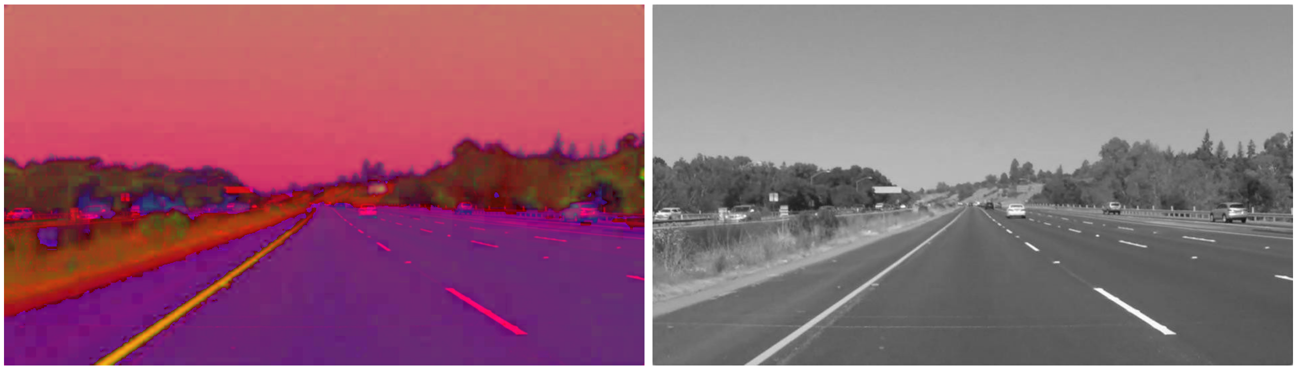


Figure 1: Original Image

1.1 Lane Line Clearance

The first step to working with our images will be to convert them to grayscale. Converting the image into gray scale will give intensity of each pixel. This is a crucial step for canny edge detector. Before applying edge detector, we need to isolate the lane lines. As Lane lines are always yellow and white in color, we apply mask to the original RGB image to return the pixels we're interested in.
After this we apply gaussian blur to suppress noise in our Canny Edge Detection.



(a) (b)

Figure 2: Image after (a) Applying Mask (b) Gray scale image

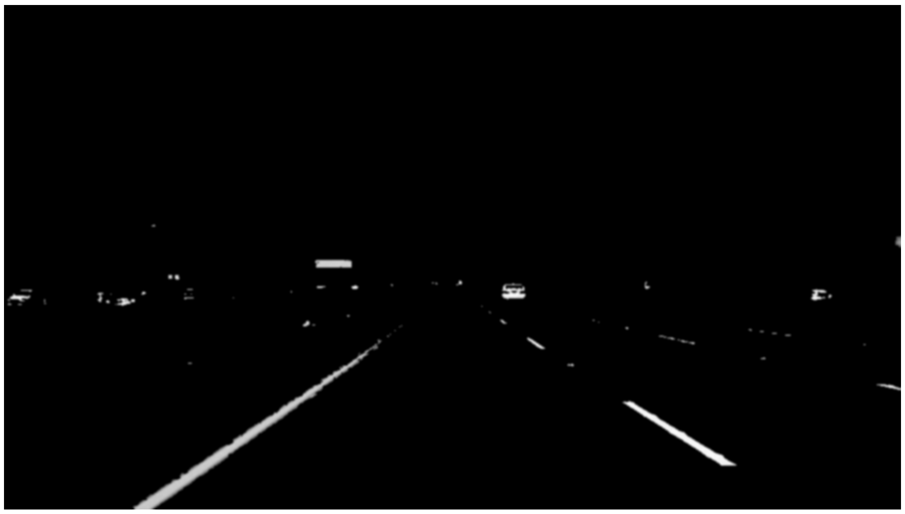


Figure 3: After Gaussian Blurringe

1.2 Canny Edge Detection

After enhancing the image for clear lane detection, we apply canny edge detection to detect all the edges especially the road lane edges. We try running the Canny Edge Detection algorithm on our cropped image with some reasonable starter thresholds.
Steps followed in canny edge detector:

- Filter image with derivative of Gaussian
- Finding magnitude and orientation of gradient
- Non-maximum suppression
- Linking and low, high thresholding (hysteresis)

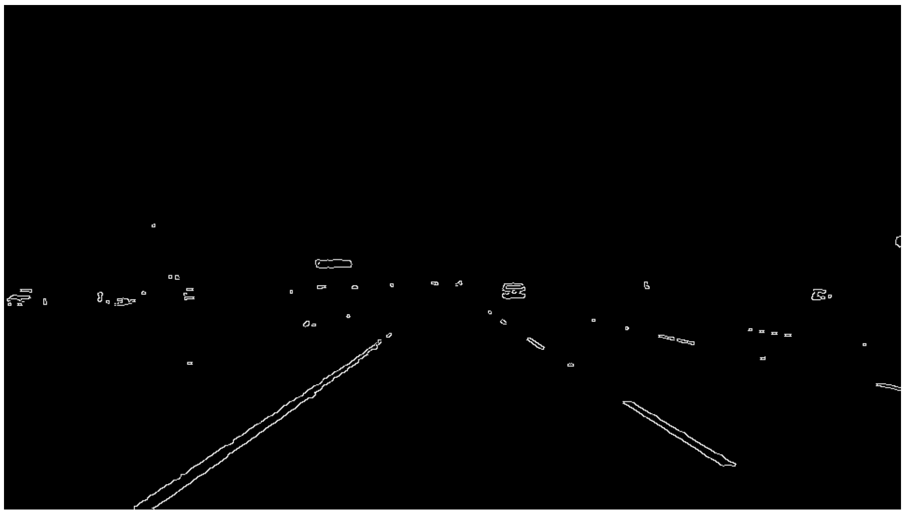


Figure 4: Image after applying canny edge detector

1.3 Region Of Interest

We want a region of interest that fully contains the lane lines. One simple shape that will achieve this goal is a triangle that begins at the bottom left corner of the image, proceeds to the center of the image at the horizon, and then follows another edge to the bottom right corner of the image.

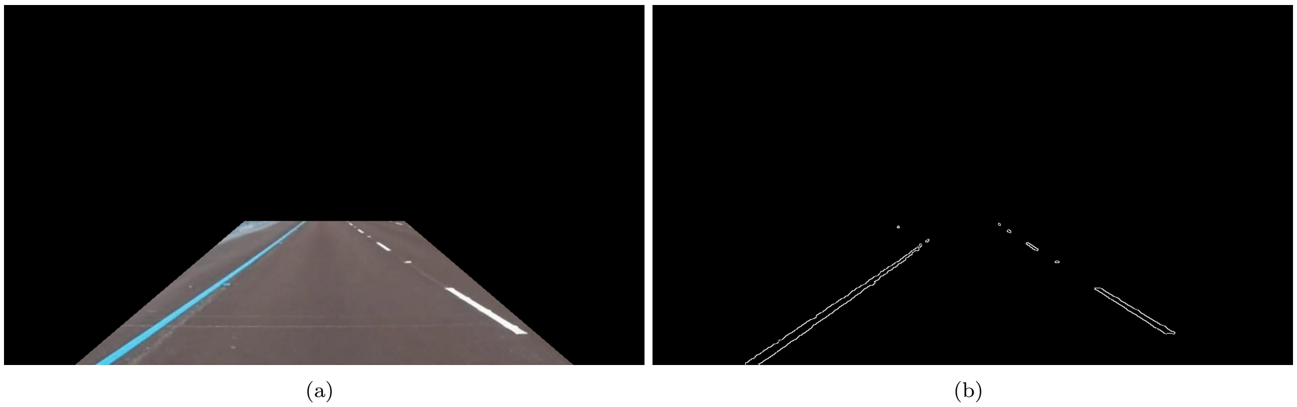


Figure 5: (a) Region of Interest window (b) Applying the region of interest window on the canny edge detection

1.4 Hough Space

As we can see from Figure 5 (b) that the road lane lines are straight lines. Therefore we need a straight line detector for detection of road lanes. One of the most popular algorithm for straight line detection is Hough Line Transform.

The Hough Transform is a method that is used in image processing to detect any shape, if that shape can be represented in mathematical form. For our problem we need a line detector and a line can be represented as $r = xcos + ysin$ or $y = mx + c$. Way hough line transform works is:

- First it creates a 2D array or accumulator (to hold values of two parameters) and it is set to zero initially
- Let rows denote the r and columns denote the ()theta.
- Size of array depends on the accuracy you need. Suppose you want the accuracy of angles to be 1 degree, you need 180 columns(Maximum degree for a straight line is 180).
- For r, the maximum distance possible is the diagonal length of the image. So taking one pixel accuracy, number of rows can be diagonal length of the image.

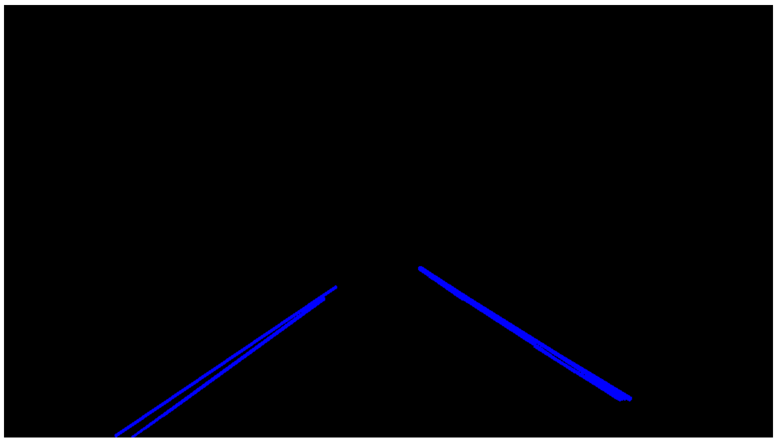


Figure 6: Hough Lines

1.5 Result

As the hough line transform was successfully able to detect the road lanes we just apply this hough lines to the original image for self-driving car assistance.
The oly ptoblem with this algorithm is that the lane detection region of interest (ROI) is not flexible. When driving up or down a steep incline, the horizon will change and no longer be a product of the proportions of the frame.

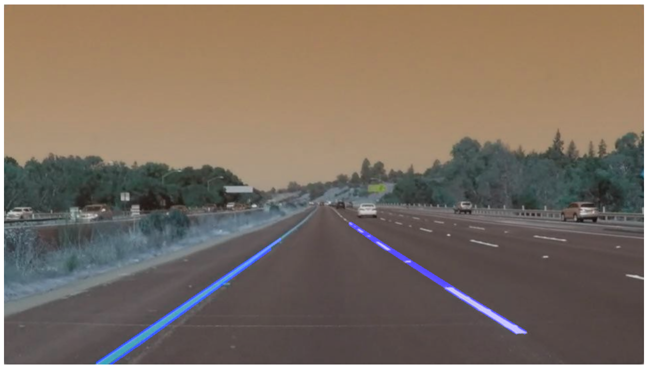


Figure 7: Final Image detecting the Road Lanes

1.6 Road Region Segmentation

For road region segmentation we use KMeans algorithm. Kmeans algorithm groups pixels with similar intensity. For road road segmentation, KMeans separates out the green grass on the side of the road with the road itself forming two different groups. This two groups are the nothing but the segmentation we want.

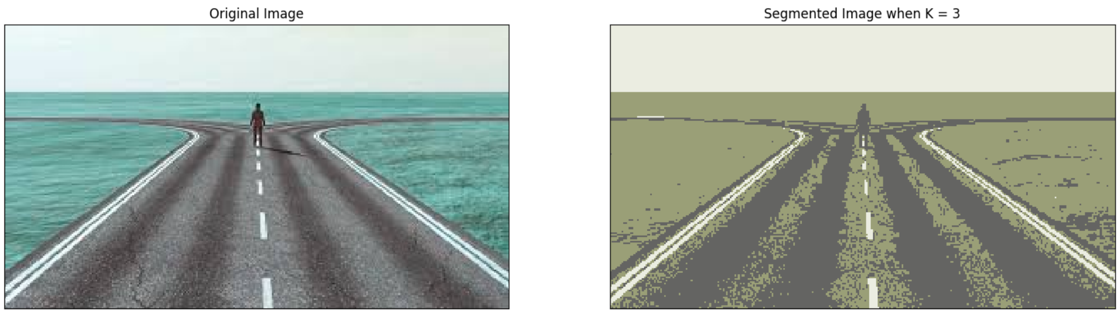


Figure 8: (a) Original Image (b) Image after separating road and surrounding

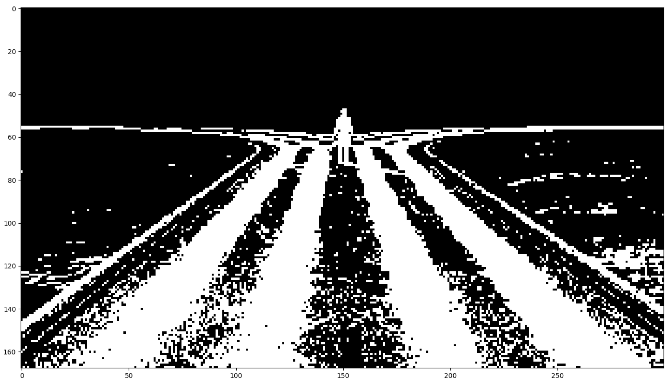


Figure 9: Road region segmentation

References

[1] https://towardsdatascience.com/tutorial-build-a-lane-detector-679fd8953132

[2] http://www.ieee-jas.org/fileZDHXBEN/journal/article/zdhxbywb/2018/3/PDF/jas-5-3-645.pdf

2 Night, Face, Landscape Classifier

Task in hand is to classify whether the given input image is a night image or face image or a image of a landscape. For classification, we can use SIFT algorithm to detect key-points and apply bag of visual words algorithm to generate a feature vector for each image. After generating the feature vector we can apply any classifier. This method works very well in classifying between and face and a night image or face and a landscape image, but it doe not a good job in classifying between night image and a landscape image.
Due to this we not only use SIFT algorithms followed by bag of visual words algorithm for feature vector generation, we also create another feature which nothing but the average intensity of the image,
Steps followed for classification are:

- Apply SIFT algorithm
- Use Bag of Visual Words to generate feature vector for the image
- Add average intensity to the feature vector
- Apply any classifier

(a)

(b)

(c)

Figure 10: (a) Face Image (b) Night Image (c) Landscape Image

2.1 SIFT

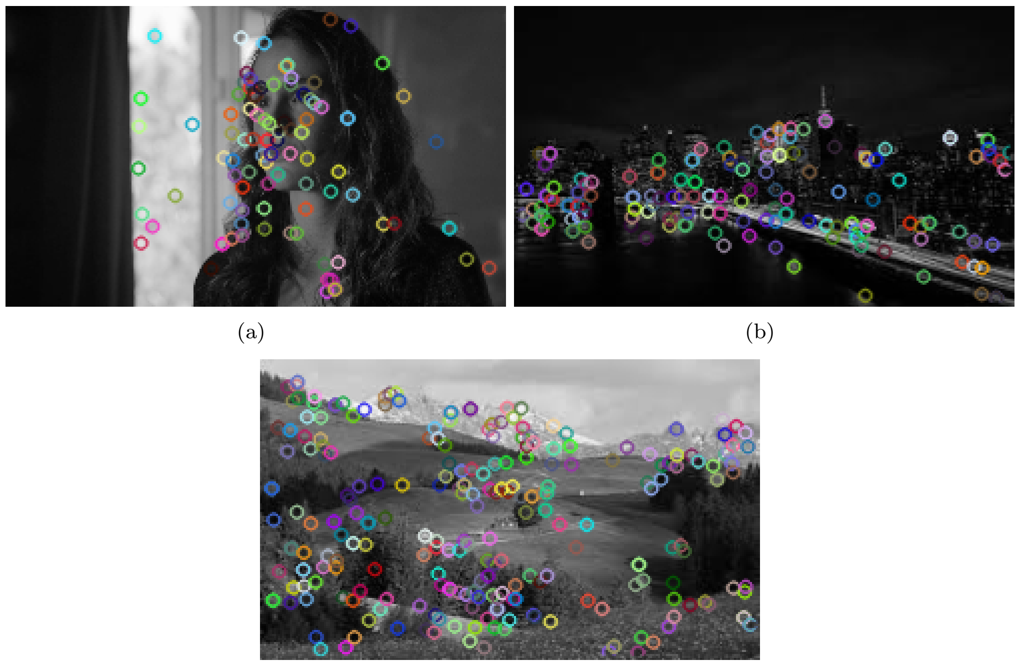
SIFT algorithm is a corner detection algorithm. It extracts the key-points from an image and computes its descriptors. Keypoints are the "stand out" points in an image, so no matter the image is rotated, shrink, or expand, its keypoints will always be the same. The descriptor is the description of the keypoint. There are mainly four steps involved in SIFT algorithm:

- **Scale-space Extrema Detection:** To detect corners, scale-space filtering is used. It finds the Laplacian of Gaussian for the image with various values.
- **Keypoint Localization:** Once the location of potential keypoint is found, they have to be refined to get more accurate results. If the intensity at this keypoint is less than a threshold value (0.03 as per algorithm), it is rejected.
- **Orientation Assignment:** A neighbourhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created.

- **Keypoint Descriptor:** Now keypoint descriptor is cre- ated. A 16x16 neighbourhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available.

.

After appplying SIFT, we get a 2D array of size (number of key points, 128)



(a) (b) (c)

Figure 11: (a) Face Image Keypoints (b) Night Image Keypoints (c) Landscape Image Keypoints

2.2 Bag Of Visual Words

Bag of visual words (BOVW) is commonly used in image classification. Its concept is adapted from information retrieval and NLP's bag of words (BOW).
The reason behind using BOVW is that the SIFT detects many keypoints of an image and each keypoint has a corresponding descriptor vector of size 128. Now to convert all these descriptor vectors into one n sized features vector, we use bag of visual words approch.
The general idea of bag of visual words (BOVW) is to represent an image as a set of features. Features consists of keypoints and descriptors. We use the keypoints and descriptors to construct vocabularies and represent each image as a frequency histogram of features that are in the image. From the frequency histogram, later, we can find another similar images or predict the category of the image.

To build a bag of visual words, we first extract keypoints and descriptors using SIFT algorithm. Next, we make clusters from the descriptors (using K-Means algorithm). The center of each cluster will be used as the visual dictionary's vocabu- laries. Finally, for each image, we make frequency histogram from the vocabularies and the frequency of the vocabularies in the image. Those histograms are our bag of visual words (BOVW).

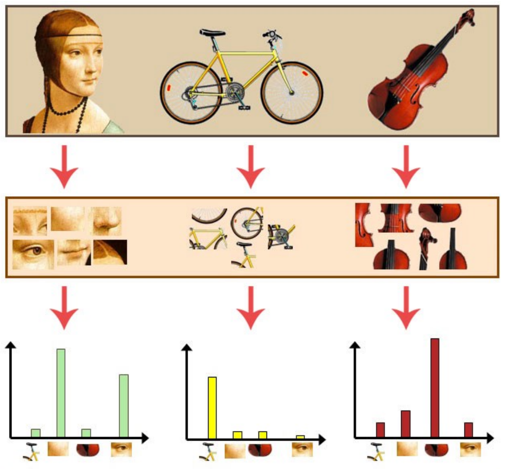
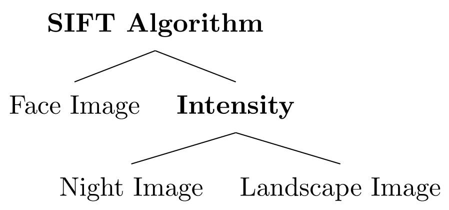


Figure 12: Bag Of Visual Words approach

2.3 Intensity Averaging

After getting the feature vector for an image using SIFT and Bag of visual words, we add two more features to this feature vector. We calculate the mean intensity of the image and intensity variance and add them to the feature vector of that image.
Reason behind using average intensity and intensity variance of each image as a feature is for better classification across night image and landscape image. As SIFT just captures intensity gradient (or edges), it makes it difficult to differentiate between most of the night and landscape images as they might have the similar intensity variation along a direction.
One major feature that helps in differentiate a landscape image and a night image is it's intensity value. Generally average intensity of night image turn out out to be low (as it is dark) as compared to day landscape image. Even a night image containing lightnings such as street lights won't show much effect on the average intensity over the average intensity of a landscape image which will be comparatively very high due to daylight.
Just to provide a simple idea using decision tree:



2.4 Model Building

After generating the feature vector, all we need to do is to apply a machine learning algorithm for classification.
To test the accuracy, I use two different models:

1. k-nearest neighbours Model
2. Random Forest Model

Model was trained using 500 images of depicting face, night and landscape. And the model was tested using 120 mixed images of face, night and landscape.

Table 1: Accuracy Table

Model	**Accuracy**
k-Nearest Neighbours with $k = 3$	78.33%
k-Nearest Neighbours with $k = 7$	80.83%
Random Forest Classifier	89.16%

2.5 Conclusion

From the above accuracy table we can conclude that using Random Forest Model we are able to achieve approximately around 90% accuracy score.
The corresponding confusion matrix is as following:

Table 2: Confusion Matrix for Random Forrest Classifier

	Predicted 0	**Predicted 2**	**Predicted 2**
Actual 0	39	0	0
Actual 1	0	40	3
Actual 2	2	8	28

References

[1] SIFT : https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html