Aim:

To study the effect of different levels of reconstruction of a decomposed image using PCA (Principle Component Analysis) for the application of automotive object detection and derive its inferences.

Input:

- The Study focuses on analysis of automotive traffic image for the application of object detection of vehicle, Traffic signals and driving lane for autonomous driving application.
- The properties of the image is given below
    - Dimensions : 1440 x 815 pixels
    - Resolution : 100 dpi (dot per inch)
- Python has been used for performing PCA on the image, Initially the image data has been preprocessed by scaling and then for further analysis of the primary color components the image has been split into the RGB color channels

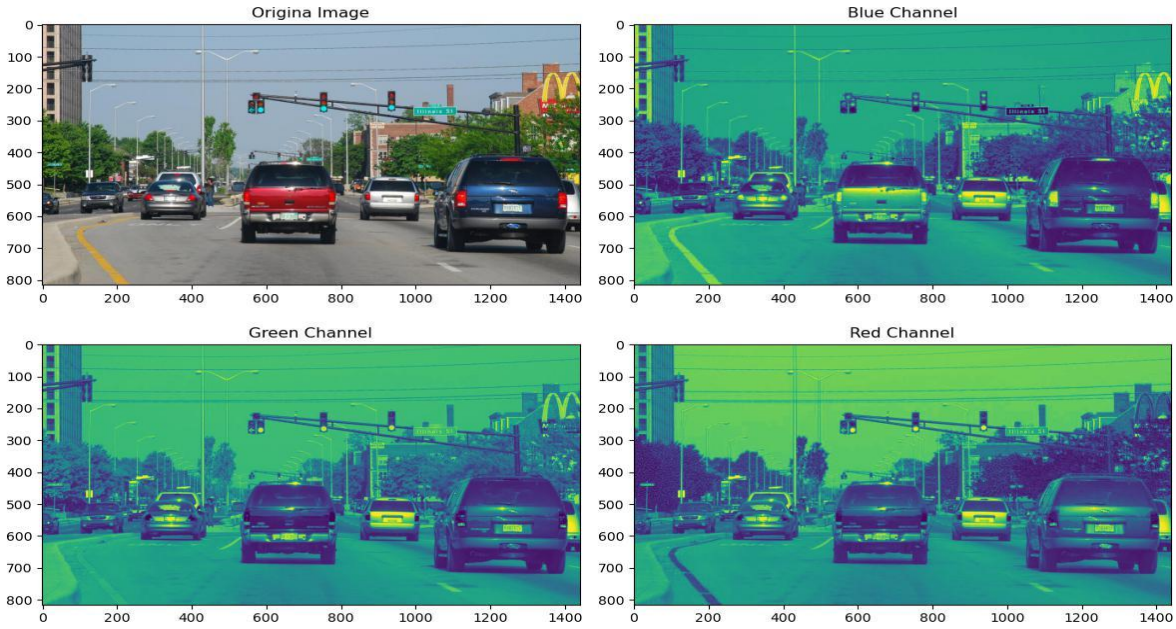

Figure 1 Original and RGB Channel Split Image

Output:

- The image has total N components of 1440 dimensions the principle component analysis of this image was done for set of reduced dimensions.
- i.e. from N/2 to N/32 which is set of N components of [720, 360, 180, 90, 45]
- The PCA analysis for these set of components was done and the Explained variance ratio is used for evaluating the performance
- Explained variance ratio of a PCA is equivalent to the ratio of eigenvalues to the sum of eigenvalues of all principle components. It is useful in determining how much information each principal component contributes to the overall structure of the data.

PCA Comp	Blue Channel	Green Channel	Red Channel
N/2	0.99999385	0.999993898	0.999995183
N/4	0.998203758	0.998230865	0.998642742
N/8	0.988466216	0.98871677	0.991391403
N/16	0.967409878	0.96835722	0.976166986
N/32	0.93272332	0.935407269	0.951458077

Table 1 Explained Variance Ratio

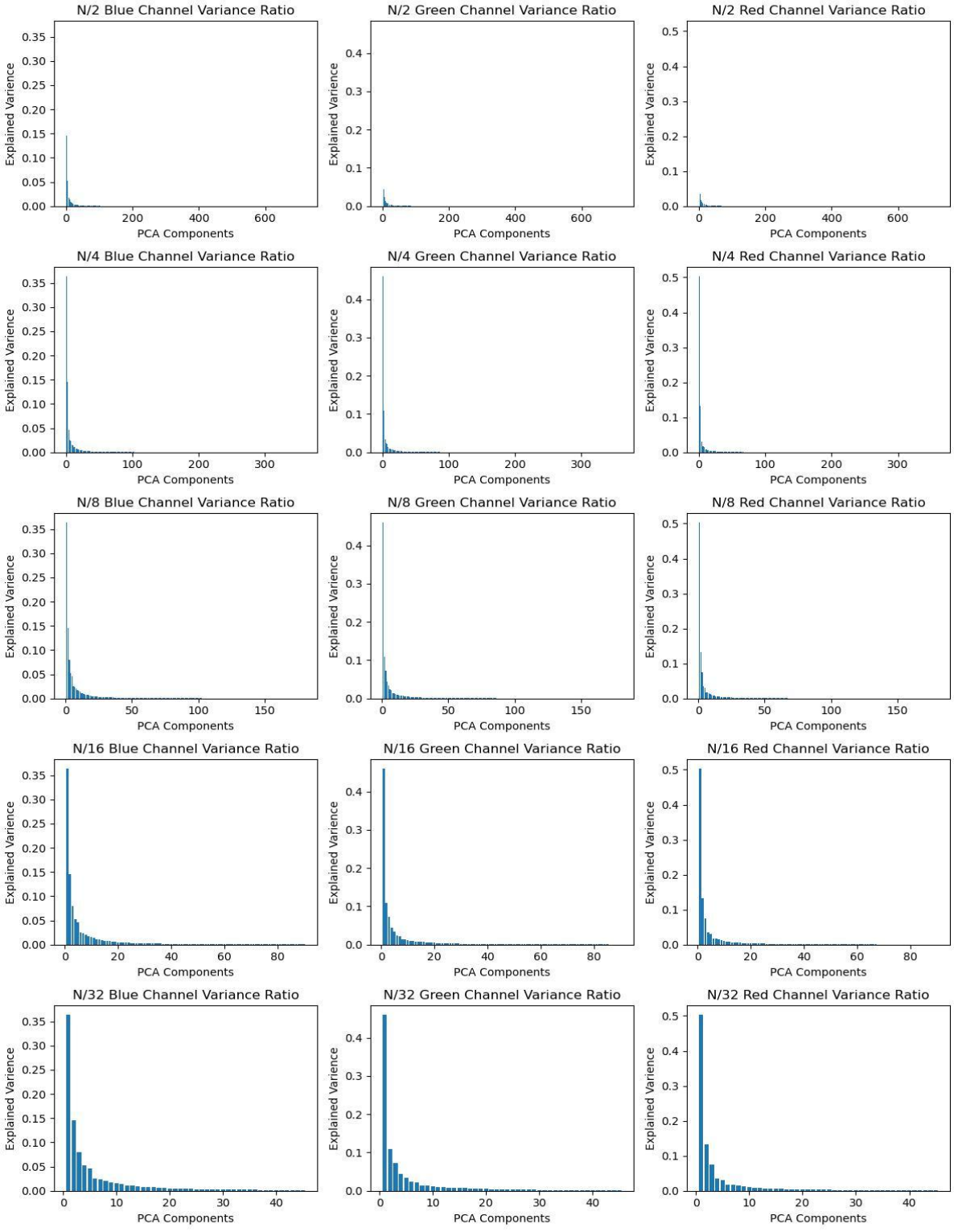

Figure 2 Explained Variance Ratio for set of n_components

- After completing the PCA analysis of the image, Now to visualize the dimensionality reduction the image has been reconstructed for this the PCA components has been reverse transformed and the data of all 3 channels been merged.

Figure 3 Reconstructed images

Inferences:

- The reduction of dimensions from the original has made it more compressed representation while allowing the most of the information in the image data retained in higher PCA components.
- In the set of N/2 to N/8 the expected variance ratio is around 98-99% which denotes by using 720, 360 or 180 n_components, the 98-99% of the variance can be kept in the image data. Which retains more information in the reduced image.
- In the set of N/16 and N/32 the expected variance ratio is around 93-97% which denotes by only using 90 or 45 n_components, the 93-97% of the variance can be kept in the image data. Which retains information in the reduced image with a loss of 3-7% variance
- The effect of dimensionality reduction through PCA has found significantly less impact on this study's image data on its image information and quality, No significant large variation till N/32 i.e. 45 component the image has been observed blurry and some loss of information.

Code:

```
#Import required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
from sklearn.decomposition import PCA
from scipy.stats import stats
import matplotlib.image as mpimg


#Load image file
image = cv2.cvtColor(cv2.imread('Vehicle100.jpeg'), cv2.COLOR_BGR2RGB)

#Image Splint in base channels
Blue_layer, Green_Layer, Red_Layer = cv2.split(image)
image_data = (('Origina Image', image), ('Blue Channel', Blue_layer), ('Green Channel', Green_Layer), ('Red Channel', Red_Layer))

#Visualize the image
fig = plt.figure(figsize=(12,8))
for i in range(1,5):
    ax = fig.add_subplot(2,2,i)
    ax.set_title(image_data[i-1][0])
    ax.imshow(image_data[i-1][1])
    plt.tight_layout()
fig.savefig('Channel_split.jpg')

#Scale the values of channel between 0 and 1
Blue_Scaled = Blue_layer/255
Green_Scaled = Green_Layer/255
Red_Scaled = Red_Layer/255
Scaled_Channel = [Blue_Scaled, Green_Scaled, Red_Scaled]

fig, k = plt.figure(figsize=(12,16)),1
Reconstructed_Image = dict()
for i in range(1,6):
    N_comp = int(image.shape[1]/ 2**i)
    print("PCA Analysis for N/{0} i.e. dimension (815, {1})".format(2**i,N_comp))
```

```
    Inverted_data = []
    for j in range(3):
        pca = PCA(n_components=N_comp)
        pca = pca.fit(Scaled_Channel[j])
        Trans_Channel = pca.transform(Scaled_Channel[j])
        print('\t-{0} Explained Variance Ratio : {1}'.format(image_data[j+1][0], sum(pca.explained_variance_ratio_)))
        #Plot variance ratio for each channel
        ax = fig.add_subplot(5,3,k)
        ax.set_title('N/'+str(2**i)+' ' + image_data[j+1][0] + ' Variance Ratio')
        ax.set_xlabel('PCA Components')
        ax.set_ylabel('Explained Varience')
        ax.bar(range(1,N_comp+1),pca.explained_variance_ratio_)
        plt.tight_layout()
        k = k+1
        Inverted_data.append(pca.inverse_transform(Trans_Channel))
    Reconstructed_Image[2**i] = (cv2.merge(Inverted_data))
    print()
fig.savefig('Explained_Varience.jpg')

#Visualize the Reconstructed image
fig ,k = plt.figure(figsize=(12,18)), 1
for i in range(1,6):
    ax = fig.add_subplot(5,2,k)
    ax.set_title('Origina Image')
    ax.imshow(image)
    plt.tight_layout()
    k = k+1

    ax = fig.add_subplot(5,2,k)
    ax.set_title('Reconstructed Image '+'N/'+str(2**i))
    ax.imshow((Reconstructed_Image[2**i] * 255).astype(np.uint8))
    plt.tight_layout()
    k = k+1
fig.savefig('Reconstructed.jpg')
```