

# CS663 Project Report

Yash Shah, 160050002 & Utkarsh Gupta, 160050032

November 2018

## Automatic Image Colorization

### 1 Problem statement

**Input** : A given black and white image and a related colored image.

**Output** : A visually acceptable coloring of the black and white image.

### 2 Procedure

- Convert the set of training images from RGB color space to LAB.
- Randomly choose some pixels from the images and extract SURF + FFT + position features at those pixels.
- Reduce the dimensions of the features using PCA.
- Choose some fixed number of chrominance values (between 6 and 32), and use the  $(a, b)$  values at the pixels to perform kmeans clustering. After convergence is attained, quantize the image by assigning each pixel to the corresponding centroid.
- Train a set of SVMs on the extracted feature vectors — one SVM for each chrominance value — using one vs. all classification. Each SVM classifies whether a pixel should have that color or not.
- Take the input (test) image, extract the same features at each pixel, project them on the PCA basis vectors and predict for each pixel the probability that it will have a color — in this implementation, this probability is approximated by the classification margin of that SVM.
- Use the above probability values and gradient of image to define an *energy* (objective) function at each pixel, and perform minimum energy image labeling using graph cut algorithm.
- After labelling the input image, convert it into RGB format by taking its luminance value and using the chrominance values of the labelled centroid to obtain the result.

### 3 New Results

We modified our approach by disabling quantization of color space, removing the SVMs for classification and removing the graph-cut labeling algorithm. Instead, we used the entire  $L-a-b$  color space, replaced the classification part with *k-nearest neighbors* search and used the inverse distances (obtained from kNN search) as weights for averaging the  $(a, b)$  values. In this way, we were able to get significantly better results as shown.

### 3.1 Same image used as reference and test image

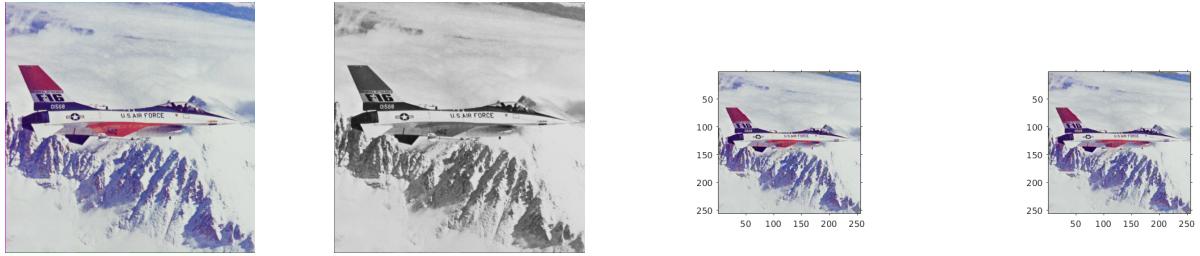


Figure 1: Airplane (reference, test, best neighbor, averaged neighbor)

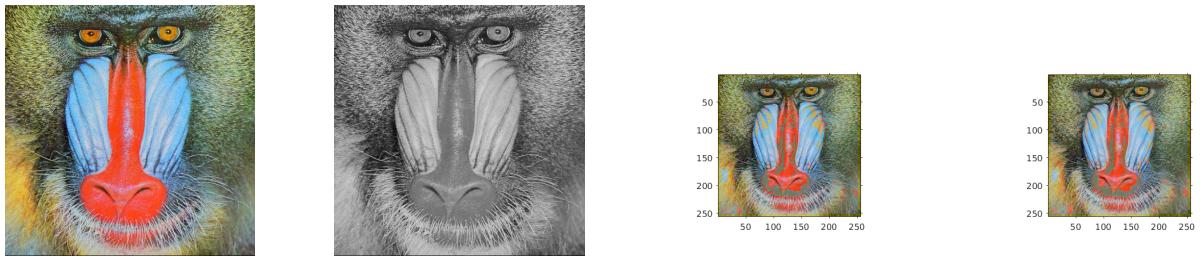


Figure 2: Baboon (reference, test, best neighbor, averaged neighbor)

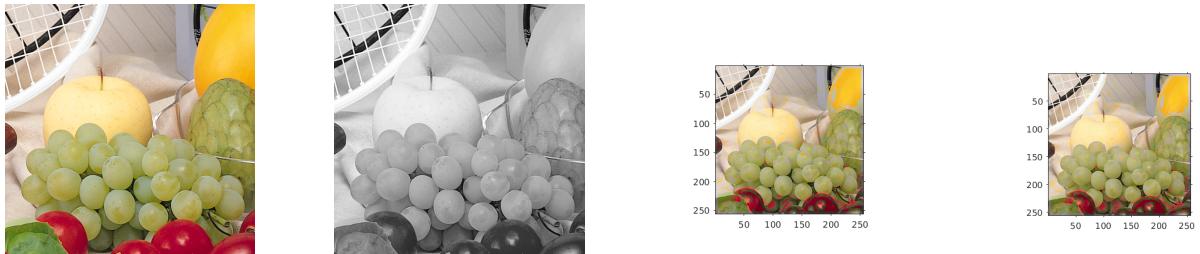


Figure 3: Fruits (reference, test, best neighbor, averaged neighbor)



Figure 4: Lena (reference, test, best neighbor, averaged neighbor)

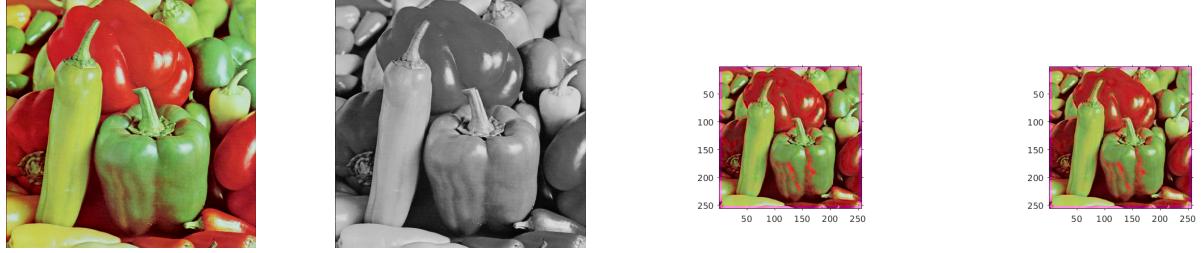


Figure 5: Peppers (reference, test, best neighbor, averaged neighbor)

### 3.2 Different reference and test image

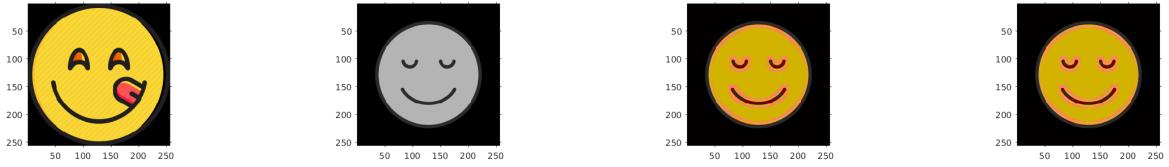


Figure 6: Peppers (reference-1, test, best neighbor, averaged neighbor)

### 3.3 Multiple reference images

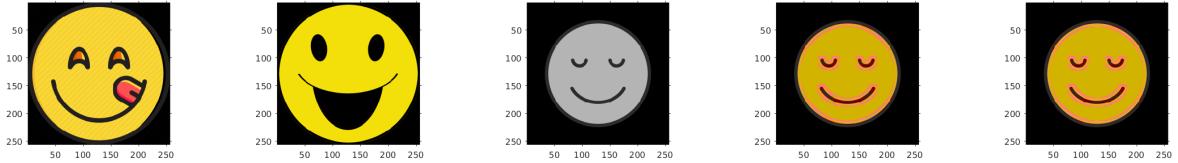


Figure 7: Peppers (reference-1, reference-2, test, best neighbor, averaged neighbor)

## 4 Original Results

For some reason (which we were not able to figure out), our SVMs were not getting trained properly and were giving almost constant scores for each color value. We tried with various image sizes, number of samples, feature sizes, etc. but were unable to get any sensible results — most of our results were monochromatic images.. Furthermore, when we tried running the code published along with the paper, we found it to be outdated; we therefore couldn't check if our code itself worked properly.

We provide the results of our implementation on various images below.



Figure 8: Airplane (original, grayscale, result)

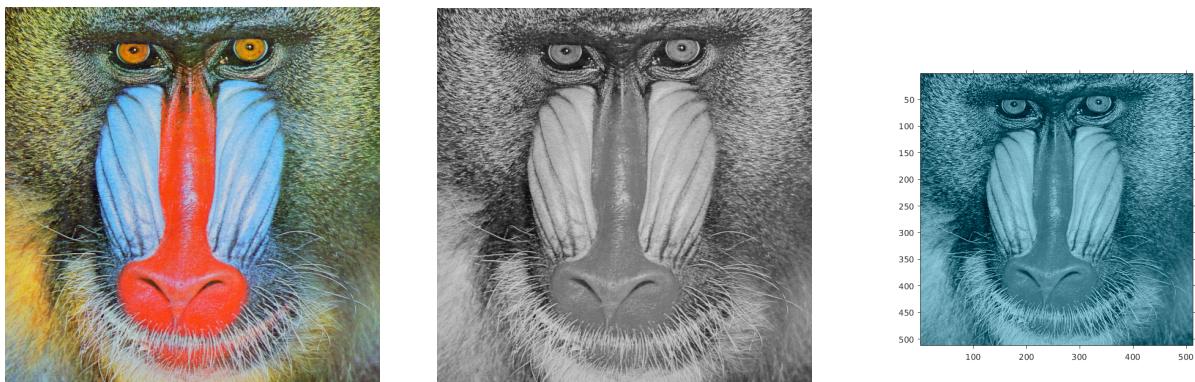


Figure 9: Baboon (original, grayscale, result)



Figure 10: Fruits (original, grayscale, result)



Figure 11: Lena (original, grayscale, result)

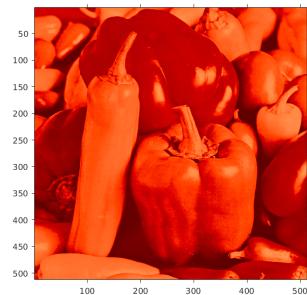


Figure 12: Peppers (original, grayscale, result)