Image & Video Processing - Assignment 4

Kelvin Likollari

1. Color Palette Extraction from Images

1.1. Linear RGB and sRGB Color Palettes

The image has been loaded, transformed to linear RGB.

Following, we cluster those colors according to color similarity in 2 spaces, namely RGB and sRGB spaces, meaning similar color values are grouped together thereby yielding a color palette of 7 distinct clusters (colors). For the decomposition of the queen's image into color layers, we select only the pixels of the queen's image which are included in each palette cluster, the rest are set to black. Lastly, an HSL-color-space conversion of the color layers has been implemented, using the **rgb2hsl** function. The results are depicted below:

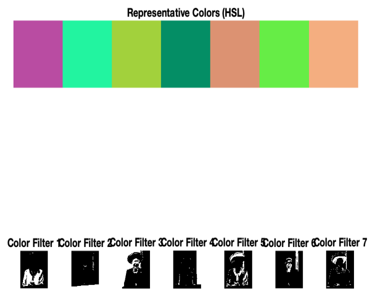

Figure 1: Linear Color Palette

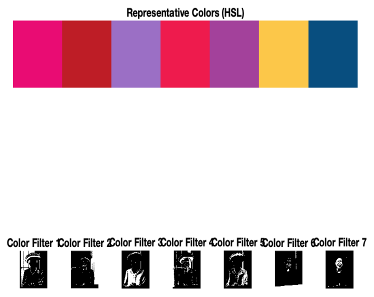

Figure 2: sRGB Color Palette

Comparison of the two color palettes: There are some evident differences between the two different color palettes, as seen from the image above (*in case you cannot see them, run the code in MATLAB and open the figure window in fullscreen - the differences shall then be clear*). The linear color palette, as seen above, yields better images, with a greater degree of accuracy of the color values, as taken from the original image. This means that certain places of the image that contain denser pixel distribution, both in terms of quantity of pixels but also in terms of variety of pixel colors, are represented slightly better using linear RGB palettes. Moreover, it can be seen that a greater amount of image details is preserved in linear RGB palettes compared to the sRGB ones, but also at the same time, perform better in terms of brightness as it supports a greater variety of brightness levels compared to the sRGB one.

1.2. CIELab Color Palette

Why is RGB, not the best color model for computing our color palette? First of all, RGB comes with limited color collections, meaning there might be a wide range of colors that cannot be represented using RGB color palettes. RGB stands for Red-Green-Blue, and as the name indicates, it is mainly used for tasks where a mixture of those 3 colors is selected. However, there are scenarios when those mixtures cannot generate accurately a color we would like to, thereby yielding in the invention of other methods that do this, such as CIELab.

We will be using CIELab instead of RGB for this section. Why is CIELab a better choice of color space in our case? CIELab offers a greater color range, conversely to RGB. CIELab can be used for tasks where color precision is very important, and in which the combination of the RGB colors is not enough. Moreover, CIELab is a better choice than RGB because it closely matches how we see colors. It measures the difference between colors in a way that aligns with our perception, making it useful for comparing colors and maintaining a consistent visual appearance. Last but not least, CIELab makes it easier to understand how colors relate to each other compared to RGB. It better matches how we see colors, which helps when working with color palettes for different tasks.

What is the expected impact of using CIELab instead of RGB? Using CIELab instead of RGB provides some advantages such as better, and more accurate representation of colors, yielding better and visually more pleasing color images. Moreover, CIELab will result in better relationships between the colors of the palette, as there shall be less dense changes in the variety of colors, yielding at the same time a better and smoother gradient color change. Lastly, with CIELab we can represent a greater range of colors, aiding us in a greater amount of applications.

The CIELab color palette looks like this:

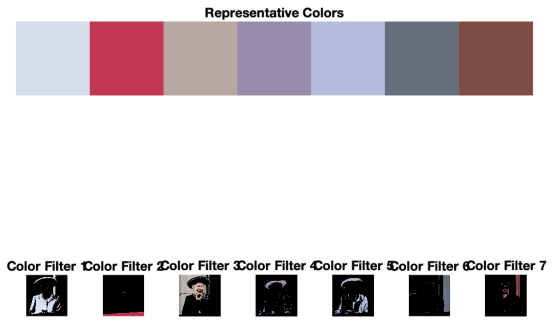

Figure 3: Queen CIELab Color Palette

Taking into account the CIELab color palette, but also the linear RGB and sRGB color palettes, it is fairly evident that the CIELAB one generates better images, with colors that correspond to the original queen image. All those images, if added, will yield in the end the original queen image. Moreover, we can see that the CIELab color palette contains colors that correspond to the colors present in the original queen image. Last but not least, those generated images are of higher quality compared to the original 2 methods, thereby confirming our original assumption that the CIELab color space is more efficient than the linear and sRGB ones.

To improve the quality of our palette, we can, first of all, choose images that contain balanced colors, meaning they have just a specific variety of colors (black, white, and gray). This will yield a nice balance of the colors in the color palette. Another technique would be to limit the number of clusters (colors) present in the color palette. Using fewer colors in our palette can yield a stronger visual impact. and can aid in creating a more focused result. For this reason, I chose the *cod4* image which contains a variety of balanced and harmonious colors. The results are shown below:

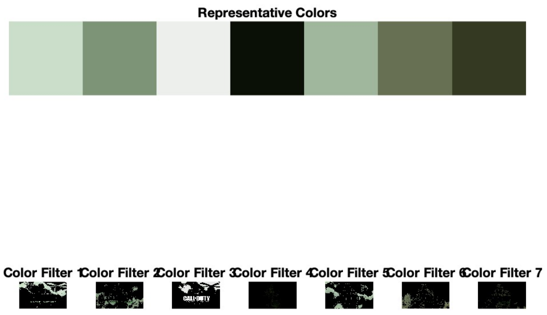

Figure 4: Cod4 CIELab Color Palette

Bonus: For this task, I chose the cod4 image which I tried combining with the CIELab color palette and applying some simple operations on it such as contrast enhancement or brightness increase. The results are shown below:

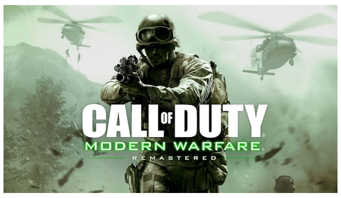

Figure 5: Original Image

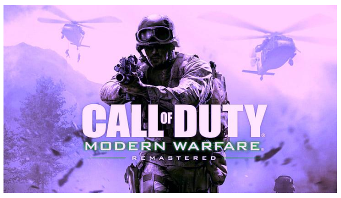

Figure 6: Pixel Corrected

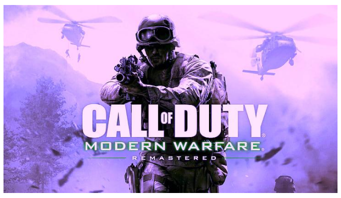

Figure 7: Gray-World

The final image, alongside the color palette, is visualized below:

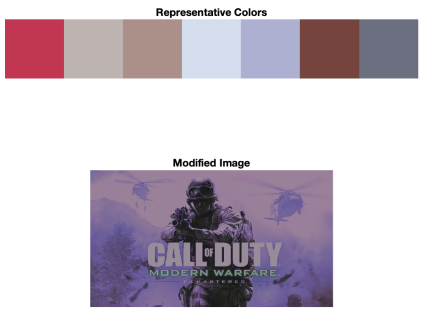

Figure 8: Final Image

2. Color Quantization and lookup tables (LUTs)

For the color quantization method, to reduce the size of the queen's image, I decided to use k-means clustering, where k will be the number of colors, set to 32. I essentially used k-means clustering, and then find the nearest centroid for each pixel. The image is then transformed back to the original queen's image size. Following this, I implemented a lookup table that transformed the compressed RGB image into a grayscale image. We take the mean of the centroids for the lookup table and then convert the quantized color image to grayscale using the LUT.

The results of both the original color quantization and the improved quantization one are shown below:

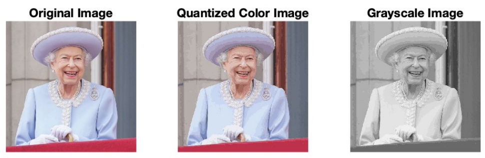

Figure 9: Color Quantization

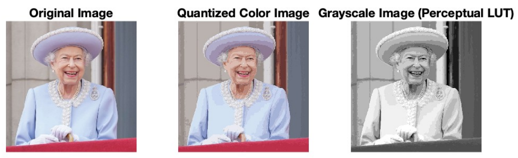

Figure 10: Improved Quantization

The comparison between the generated images between the original color quantization and lookup tables with the improved/enhanced color quantization can be distinguished by comparing the previous two generated images. Effectively, it can be clearly seen that the new improved color quantization method yields better results which can be seen by comparing the middle image of the 2 subplots. The middle image on the improved method contains fewer artifacts/noise patterns (especially in the blue vest of the queen). For the last image, comparing the grayscale image yielded from the original color quantization with the perceptual LUT grayscale image, also shows that the improved grayscale image contains fewer artifacts/noise than the original color quantization methods (zoom onto the PDF to see the differences because they are not clearly visible).

3. Gaussian and Laplacian Pyramids

The 4 levels of the Laplacian pyramid of the *happy* image are visualized below: (Please make sure you have the dark mode on because they do not look good on light mode - alternatively, you can execute the respective MATLAB script).

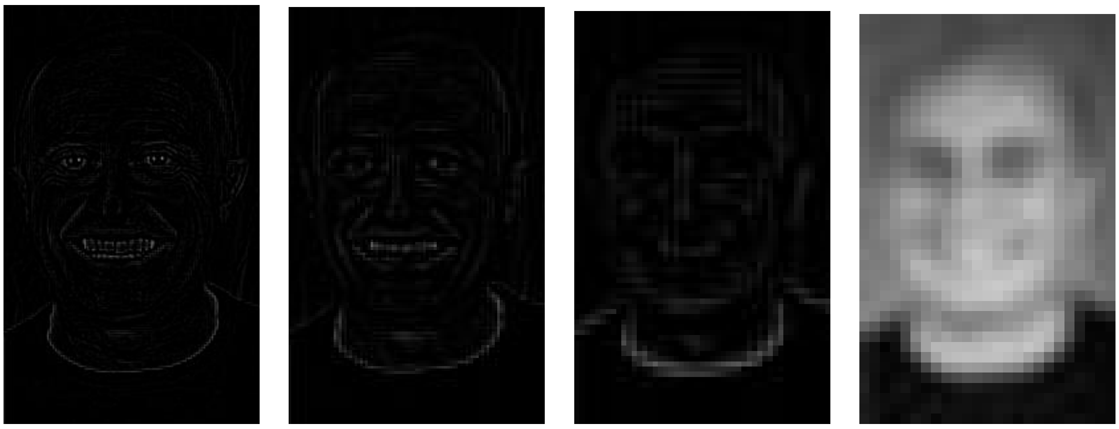

Figure 11: Laplacian Pyramids for *happy*

Similarly, for the *sad* image, the respective Laplacian pyramid is visualized below:

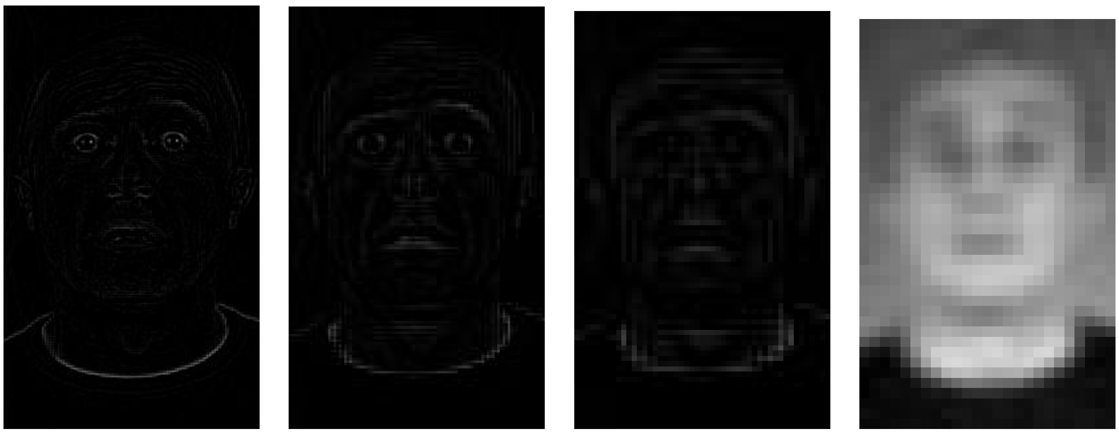

Figure 12: Laplacian Pyramids for *sad*

No special work is needed for generating the different levels of the Laplacian pyramid as seen from the respective MATLAB script. Namely, we start with a grayscale version of the provided image(s), followed by a Gaussian filter, which in turn is followed by a call to the function that generates the actual Laplacian pyramid, in which, the major operations happening are a convolution of the previous pyramid level with the spatial filter, followed by a downsample of two.

However, one thing to be noted is that if we combine (sum) those 4 levels of each image, it will yield the original happy/sad images, which is one of the properties of the Laplacian Pyramids.

4. Hybrid Images

For creating a hybrid image, I chose to implement image filtering using high and low-pass filters. High and low-pass filters allow us to enhance specific frequency components in both of the images. We start by grayscaling both images and then applying the high-pass filter is then applied onto the happy image, to extract the high-frequency details, such as edges etc. Conversely, the low-pass filter is applied on the sad image, extracting this time the low-frequency details and preserving the details of the image. Both filters use FFT to perform their respective filtering operations on the frequency domain, and both filtered images are combined to create a final image that combines both high and low-frequency components. Some observations to be made for the sigma value are that the lowest the sigma value is, the wider the Gaussian filter will be, meaning more low-frequency information from the *sad* image passes through and vice versa (for increasing the size of sigma).

As shown in the slides, some properties of the hybrid images are that, depending on the distance the human observer sees different parts of the signal. For example, when the viewer is far, he sees low spatial frequencies, and vice versa, when the user is at a close distance, he sees high spatial frequencies. Given the fact that the low-pass filter was applied to the sad image, this means that the sad image should be viewed at a greater distance than the happy image, in which the high-pass filter was applied. Thereby, this yields the conclusion that the distance to see the sad face must be greater than the distance needed to see the happy face.

In particular, to calculate the distance to see the input images, we effectively have to use the following formula: $\alpha = 2\arctan\left(\frac{W}{2d}\right)$. With W being the weight of each respective image, and d is the distance from which we view the image. Given the fact that the happy image has a viewing distance of 1m and the sad image has a viewing distance of 8m, we can replace the values and retrieve the angle α. By replacing the values, we obtain for the happy image $\alpha_1 = 0.26$ and $\alpha_2 = 0.02$ for the sad image. To obtain the level of the Laplacian pyramid need, we can use the following formula: $f_1 = \frac{2}{d}\tan\left(\frac{\alpha}{2}\right)$. Substituting for the two values of α, we get $f_1 = 0.497$ cycles/pixel and $f_2 = 0.0013$ cycles/pixel. Given $(0.5)^k$, with k being the level corresponding to the Laplacian pyramid and we will try to approximate as much as possible. According to the slides, the bottommost (first) level of the Laplacian pyramid, has 0.5 cycles/pixel, and given $f_1 = 0.497$ approximates 0.5, we say that for the happy image, we get the first level of the Laplacian pyramid, whereas for f_2=0.0013 we get 0.0078, meaning the 7th level. Regarding the size of the image, and considering we are dealing with a combination basically of images, the sizes of the high pass and low pass filters are to be considered. Therefore, the size of the final hybrid image will be 319x193.

5. Theory: Hybrid Images

Given two input images, we want to combine two levels of their respective Laplacian pyramids to create the final hybrid image which will consist of two levels, one from each Laplacian pyramid of each input image. One of the images should be visible from a distance of 1 meter, and one from a distance of 8 meters. For the first image, visible from the closest distance, we use a high-pass filter, and respectively for the other image we use a low-pass filter. When we are closer to an object, we are more sensitive to higher frequencies, thereby the image is clearer. For this reason, we acquire a high-pass filter as high-pass filters allow high-frequency details to pass through while removing the low-frequency components. For the second image (the one intended to be visible from a distance of 8 meters), we use a low-pass filter, as a low-frequency pass allows the low-frequency details to pass through while removing the high-frequency components. For the chosen levels of the pyramid, we can reason that from the image that is closer to the viewer, we can acquire the first (bottommost) pyramid level, as the first level contains a greater quality of the image and contains a greater level of details, needed for the high-pass filter. Conversely, for the image that is farthest away, we use the highest level of the pyramid, as the top (last) level of the pyramid contains fewer details, which is also the case when using the low-pass filter.

6. Bonus: Create your own Instagram Filter!

There is a series of transformations performed in order to obtain the final image that looks like this: In the beginning, the original image is taken and edge detection is performed using the **Sobel** filter. The edge detection performed on the original image is visualized below:

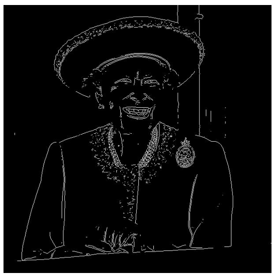

Figure 13: Edge Detection

As a following step, we performed a dilation operation in order to make the extracted edges thicker and give them a cartoonish black color, as seen below.

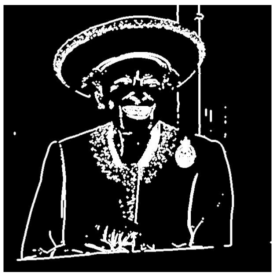

Figure 14: Edge Dilation

Following, we perform Color Quantization. Color Quantization helps in reducing the number of colors in an image by mapping similar colors to a smaller set of discrete values. The color quantization process starts with converting the image to the CIELab color space, which is a better space for performing color quantization. Next, the colors of the image are clustered using the k-means algorithm, before determining the color that will represent each cluster. Following, the pixel's color is calculated by using the color of the nearest neighbor, thereby yielding similar colors being put in the same clusters. As a last step, the image is converted back into the original color space, in order to be able to visualize it, as seen below.

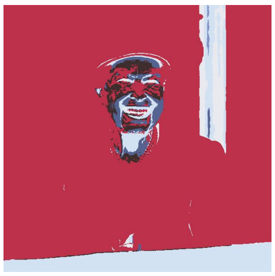

Figure 15: Color Quantization

For the next stage, we increase the color saturation in order to boost the intensity of the colors already present in the image. This is performed by converting the image onto the HSV space, then adjusting the saturation channel, and as a last step, we convert the HSV image back onto the RGB color space.

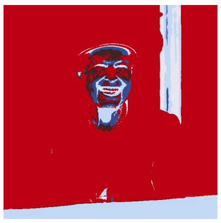

Figure 16: Increased Color Saturation

Onto the next stage, we add thick lines as an extra filter feature.

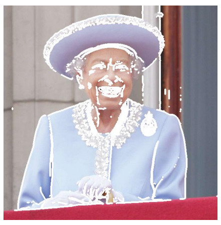

Figure 17: Thick Edges

Next, we combine the increased color saturation image with the filtered image that already contains the aforementioned thick edges. This is done in order to apply saturation to the edge regions that are not captured previously.

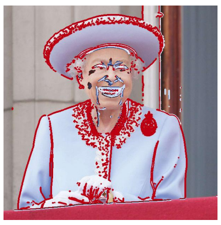

Figure 18: Saturation With Filtered Image

As a last step, I implement some additional operations, such as image adjustment, image sharpening, and Gaussian smoothing, in order to obtain an aesthetically and visually pleasing filter, as shown below.

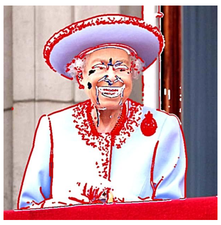

Figure 19: Final Filter