Università della Svizzera italiana

Faculty of Informatics

Image and Video Processing **2023**

Students: Filippo Casari, Alessandro De Grandi

Report for Assignment 4

1. Color Palette Extraction from Images

1.1. Linear RGB and sRGB Color Palettes

- We implemented the function to convert to sRGB to linear RGB using the formula shown in the slides.
- We used kmeans to cluster the image colors into 7 clusters, for both sRGB and linear-RGB
- We created a subplot showing the color pallettes on first row, second row shows the layers where each pixel gets the color of that centroid, third row shows the original pixels assigned to that cluster, There is not much difference between RGB and sRGB clustering.

1.2. CIELab Color Palette

RGB is not the best color space to use when we calculate kmeans, because this algorithm relies on euclidean distances. In contrast CIELab should be better because it is perceptually uniform, so colors that are perceptually similar will have similar distances and will be clustered together.
We played around in CIELab color space by increasing the luminosity of the third layer in image 4 to give the queen a more shiny lipstick.
Also we chose to modify our favourite Van Gogh Painting, Starry Night. By modifying the first and third layer in image 5 by increasing luminosity and shifting the a* channel from greens towards reds and the b* channel form blues to yellows we obtain a very nice artistic effect transforming Starry Night to Starry Dawn.


(a) sRGB


(b) Linear-RGB


(c) sRGB


(d) Linear-RGB


(e) CIELab

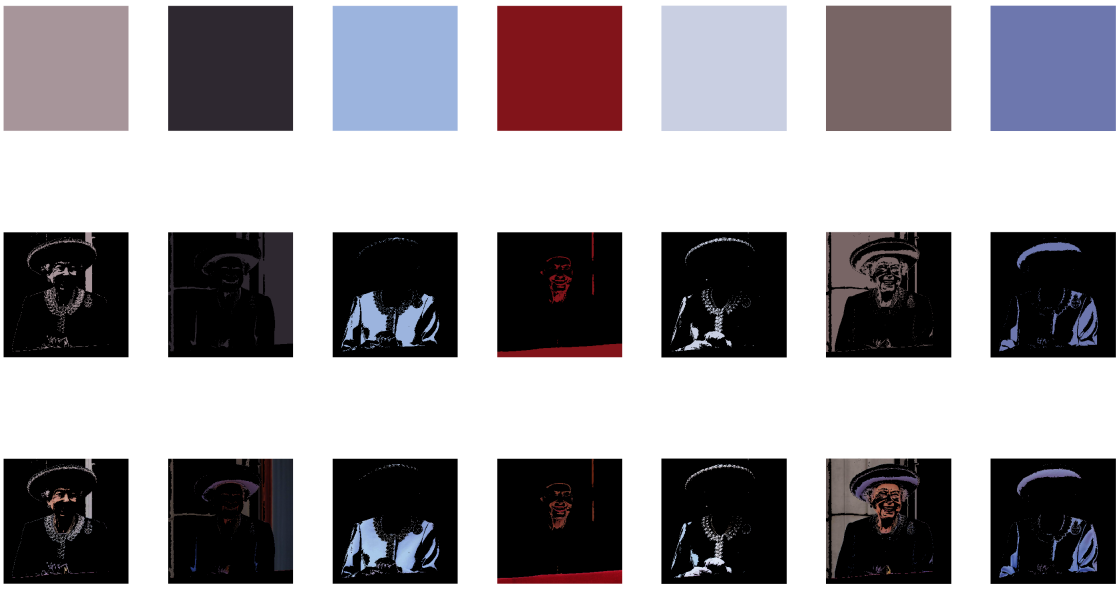


Figure 2: Decomposition Linear

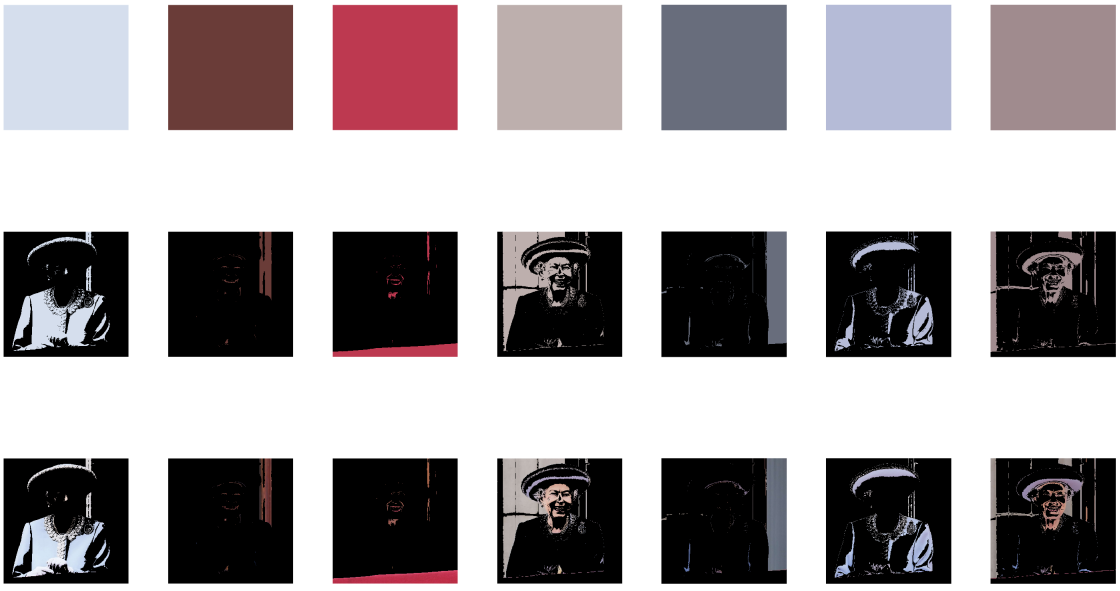


Figure 3: Decomposition sRGB

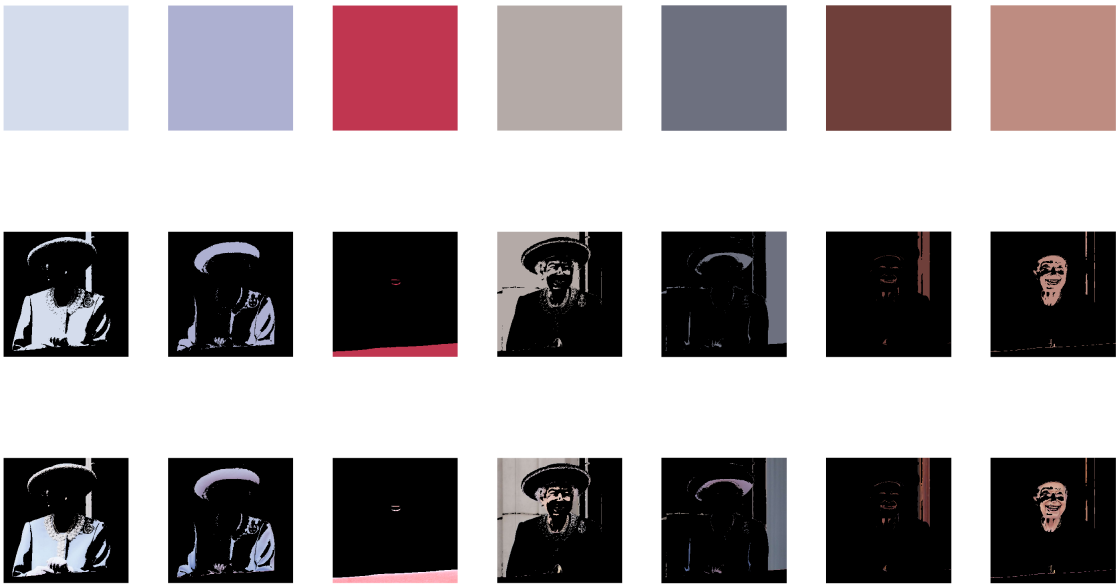


Figure 4: Decomposition CIELab
increased luminosity of 3rd layer

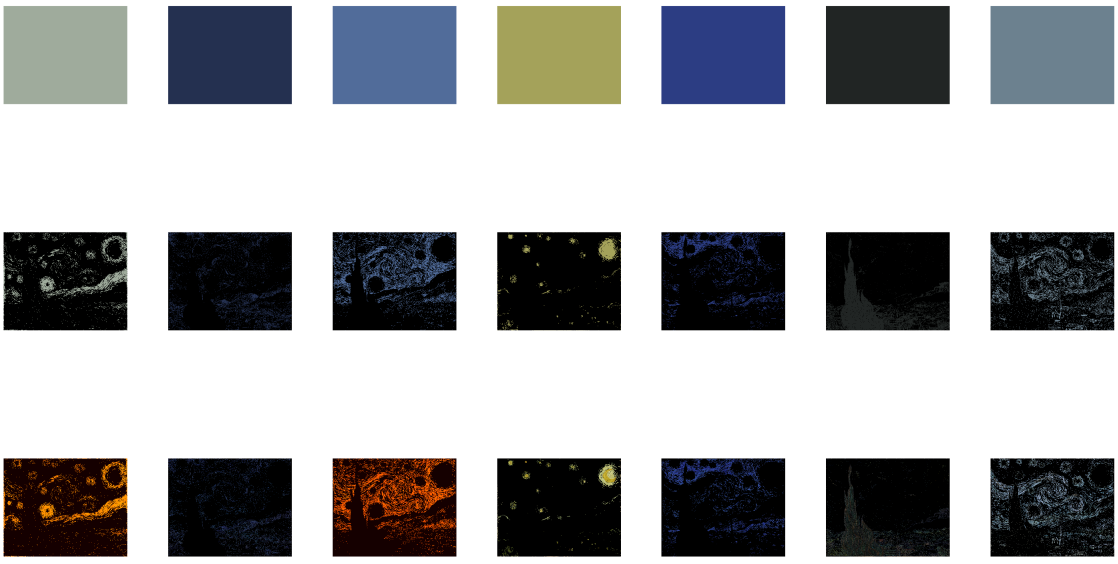


Figure 5: Decomposition CIELab Starry night modified

Figure 6: Starry Night

Figure 7: Starry Dawn

2. Color Quantization and lookup tables

We used 2 methods for performing color quantization:

- Thresholding
- Clustering

For thresholding method we first created a matrix constaining 32 possible combinations of colors between 0 and 1 for each channel (tab. 1)

R	G	B
0	0	0
0	0	0.667
0	0.334	0
0	0.334	0.667
0	0.667	0
0	0.667	0.667
0	1	0
0	1	0.667
0.334	0	0
0.334	0	0.667
...	...	...

Table 1: 32 possible combinations of colors

Then, we computed the distances between each pixel and all the possible combinations. After that, we assigned to the new image the colors from the table nearest to the original pixel.
We noticed, as expected, that this method is the easiest but the crudest one (fig. 8).

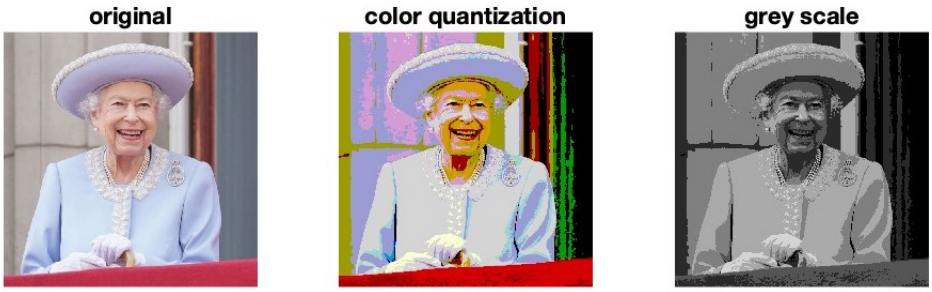


Figure 8: thresholding

In our second approach, we used Kmeans clustering to find 32 centroids. Having found them, we computed the distances (like before) and assigned at each new image pixel a value corresponding to the closest centroid.

Figure 9: clustering

For both methods we implemented the conversion from color image to grey scale image using the below formula applied for each pixel. We built a LUT (tab. 2) using HashMap data structure in which the keys are the RGB colors and the corresponding values are the grey values. Then, we converted the image to grey scale using the LUT.

$$grey\ value\ = \frac{max(Centroid) - min(Centroid)}{2}$$

As one can easily see, the second method performs much better because takes into account only the colors present within the original image, and groups colors very similar to each other.

Key			**Value**
R	*G*	*B*	*Grey*
0.6320	0.8438	0.6869	0.7379
0.7380	0.1887	0.2903	0.4634
0.3857	0.4143	0.4723	0.4290

Table 2: first 3 entries LUT

3. Gaussian and Laplacian Pyramids

We created a function called **laplacian_pyramid**. In figures 11, 12, 13 are shown the layers produced by our function. Additionally, in fig. 10 is shown the reconstructed image. In fact, we implemented the reconstruction phase in order to verify the correctness of the pyramid. The reconstructed image matches the original one perfectly. We omitted to show the happy face image because we got the same effects and results.

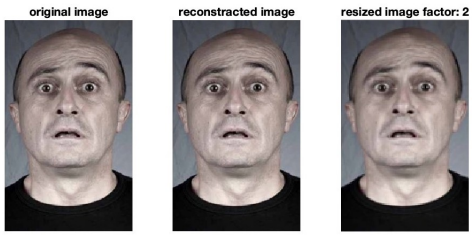


Figure 10: Original, Reconstructed, last layer

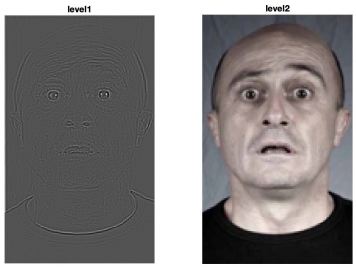


Figure 11: layers, pyramid of 2

Figure 12: layers, pyramid of 3

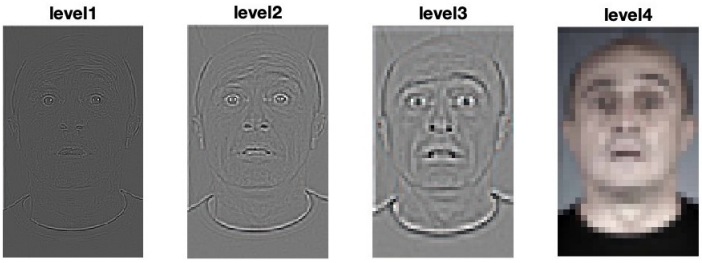


Figure 13: layers, pyramid of 4

4. Hybrid Images

To accomplish our goal, we used Laplace pyramid method. We extracted low frequencies from the sad image using 2 layers, and high frequencies form the happy image using 4 layers. We used 2 different σ for the 2 pyramids: 1.5 for the first one, and 0.01 for the second one. The successful result is shown in fig. 14.

Figure 14: Result

Ideal viewing conditions:

- the hybrid image of 4x3 square cm
- 30 cm to see the happy face
- about 2 m to see the sad face

5. Theory: Hybrid Images

We assume that we see at 1 meter 0.5 cycles/pixel. Then, if we were distant from the screen 8 meters, we would see 0.5/8 cycles/pixel, that is, 0.0625 cycles/pixel. Looking at the cycles/pixel corresponding to Laplacian pyramid, we can say that we should combine the first layer of the Laplacian for the first image and the 4^{th} one for the second image. By doing that, we see the the first image (higher frequency) at 1 meter, and the second one (lower frequency) at 8 meters. Indeed, humans perceive better higher frequencies at shorter distances, and lower frequencies at further distances like our case.

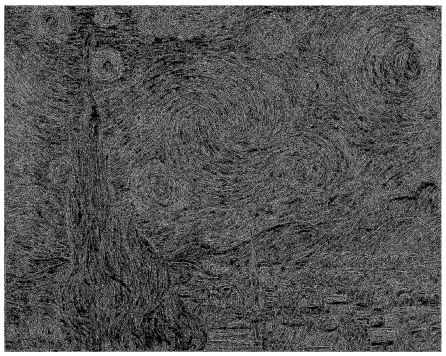
6. BONUS - Van Gogh Starry Night - Style Transfer Filter

Our idea is to create a filter to make any image look like a painting with similar style of Starry night. The filter can be summarized in two steps: extracting the brush style of Van Gogh painting and transferring colors of the painting to the image. The filter is plug and play, usage is shown in plugPlay.m

Filter steps:

- Apply Gaussian Filter to greyscale of Original Starry Night painting and subtract to obtain high frequencies, binarize resulting image.
- Summing a cropped region of these High frequencies of Starry night to any image,then rescaling and increasing slightly the luminosity in CIELab color space, to get Van Gogh effect.
- Then transfer colors by using kmeans clustering on CIELab Starry Night with 32 clusters. Then for each pixel in the Original image assign that pixel to the nearest centroid of the Starry Night clustering.

(a) StarryNight



(b) Starry Night high frequencies

(a) Original Image of Lugano

(b) Vang Gogh effect on Lugano

(a) Lugano Starry Night Style Transfer

(b) Queen Van Gogh Style Transfer