ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

COLOR IMAGING PROJECT

Color transfer

Benoit Rat
Islam El-Sayed

The Images and Visual Representation Group (IVRG)

Prof: Sabine Süsstrunk

Introduction

Color Transfer consists in altering the colors of a photograph or an image (the target image) by using the color properties of another image (the atmosphere). As there are many ways to do so, we will concentrate our efforts on one solution that has been provided by Erik Reinhard, Michael Ashikhmin, and Peter Shirley [1]. But then the model provided on this paper was only a starting point, because as we will see later changes from the original method described in the paper had to be made in order to obtain a sustainable result. Our project is divided into two parts: the first part deals with the color transfer by simple scaling of the data statistics on the overall (target) image and the second part concerns the color transfer using clustering methods and image statistics.

I. The Theory

Because our goal will be to manipulate RGB images, which are often of unknown chromaticity, we have to convert the RGB signal in a perception based color space namely the *L alpha beta* color space discovered by [2]. We do so because in RGB space, most pixels will have large values for the red and green channel if the blue channel is large.
This implies that if we want to change the appearance of a pixel's color in a coherent way, we must modify all color channels in tandem. This complicates any color modification process. Thus what we need is an orthogonal color space without correlations between the axes. One color space that has the following wanted property is the *L alpha beta.*
This color space has little correlation between the axes, which lets us apply different operations in different color channels with some confidence that undesirable cross-channel artifacts won't occur. The *L* axis corresponds to the luminance and thus the achromatic channel whereas the *alpha* and *beta* channels are respectively achromatic yellow-blue and red-green opponent channels.
We describe here how the transformation is made from the RGB color space to *L alpha beta* using [1].
As Lαβ is a transform of the HVS cone space, namely the LMS, we first convert the RGB space to LMS.
The method described here converts first an RGB Image into the XYZ tristimulus values. This transformation is done using the M_itu matrix. Once in the device independent XYZ space, we convert the image to LMS space using another matrix provided in [1].

Combining both matrices, we obtain the following transformation:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The data obtained in the LMS color space contains a lot of skew, which can be eliminated by converting the data to logarithmic space. In our work we first did a scaling of the data in LMS from 1 to 100 in order to avoid having trouble with values near 0. Finally, from L'M'S' to L alpha beta, we use the following transformation indicated in [1]:

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1/\sqrt{3} & 0 & 0 \\ 0 & 1/\sqrt{6} & 0 \\ 0 & 0 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} L' \\ M' \\ S' \end{bmatrix}$$

After the color processing, [1] explains how to get back to the initial RGB space in order to display the images. Converting from *L alpha beta* to LMS then raising to the power of ten (because of the logarithmic multiplication in the RGB → L α β), followed by a conversion to RGB is done using these transformations:

$$\begin{bmatrix} L' \\ M' \\ S' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{3}/3 & 0 & 0 \\ 0 & \sqrt{6}/6 & 0 \\ 0 & 0 & \sqrt{2}/2 \end{bmatrix} \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

II. Implementation and main results

1. Simple color transfer model

The goal of this part is to use the statistical properties of the image by doing a simple but efficient color transfer which will be applied to the whole image, without any distinction in the luminance or in the colors that are in the target or source image, i.e. the deviations done to the target image will be the same for every pixel in this image.

For the purposes of the project, we will only use the mean and standard deviation of the image. The scheme implemented in this phase corresponds exactly to the one described in [1].
We first subtract the mean of only the target image along the three channels.

$$L^* = L - \overline{L}$$
$$\alpha^* = \alpha - \overline{\alpha}$$
$$\beta^* = \beta - \overline{\beta}$$

Then we scale the new values of the target image by the ratio of the standard deviations of both source and target image.

$$L' = L^* \frac{\sigma_{target}}{\sigma_{atmo}}$$
$$\alpha' = \alpha^* \frac{\sigma_{target}}{\sigma_{atmo}}$$
$$\beta' = \beta^* \frac{\sigma_{target}}{\sigma_{atmo}}$$

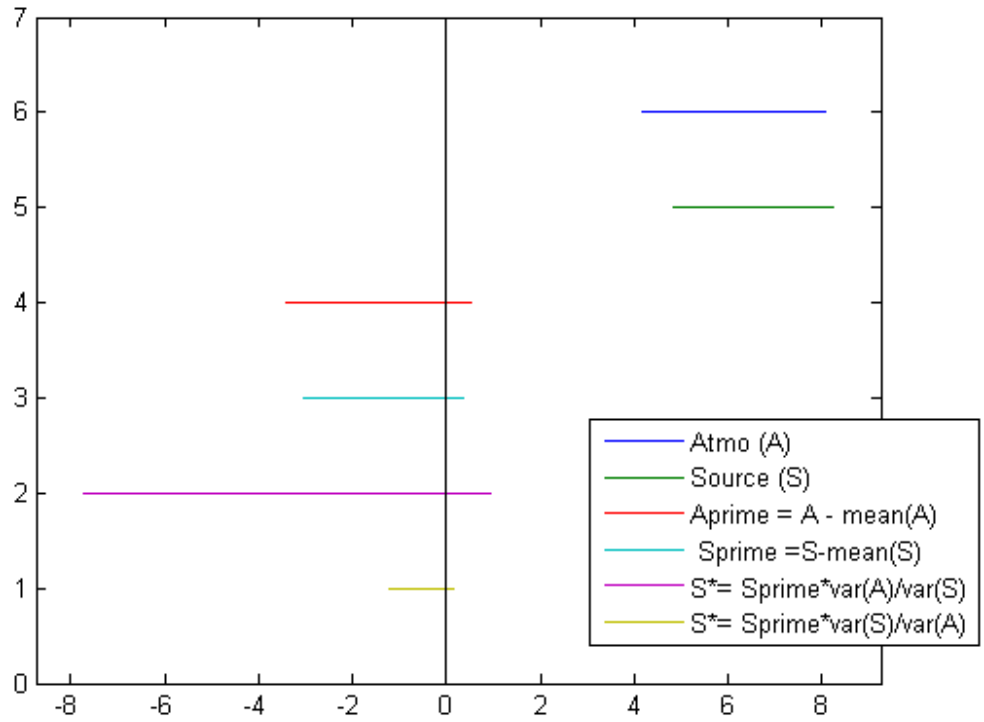


Figure 1 representation of the Lαβ vectors (with max and min values) scaled by the ratio of standard deviation around the mean values

Once all these operations are done, we do not simply add the mean of the atmosphere image as it is described in [1], but instead we take a flavor of both the initial mean of the target and the source image. The whole aim of the process is to have an image without color cast at the end, which depends greatly on the initial values of the target and atmosphere mean. For doing so, in order to preserve some of the original properties of the initial image without loosing the information we added by the scaling factor (K_color), this solution described below has been found to be the best one. The following example is for the alpha channel, but the calculations are done exactly in the same way for the two other channels.

$$\alpha_{final} = \alpha'_{target} + R * (K_{color} * \bar{\alpha}_{target} + (1 - K_{color}) * \bar{\alpha}_{atmo})$$

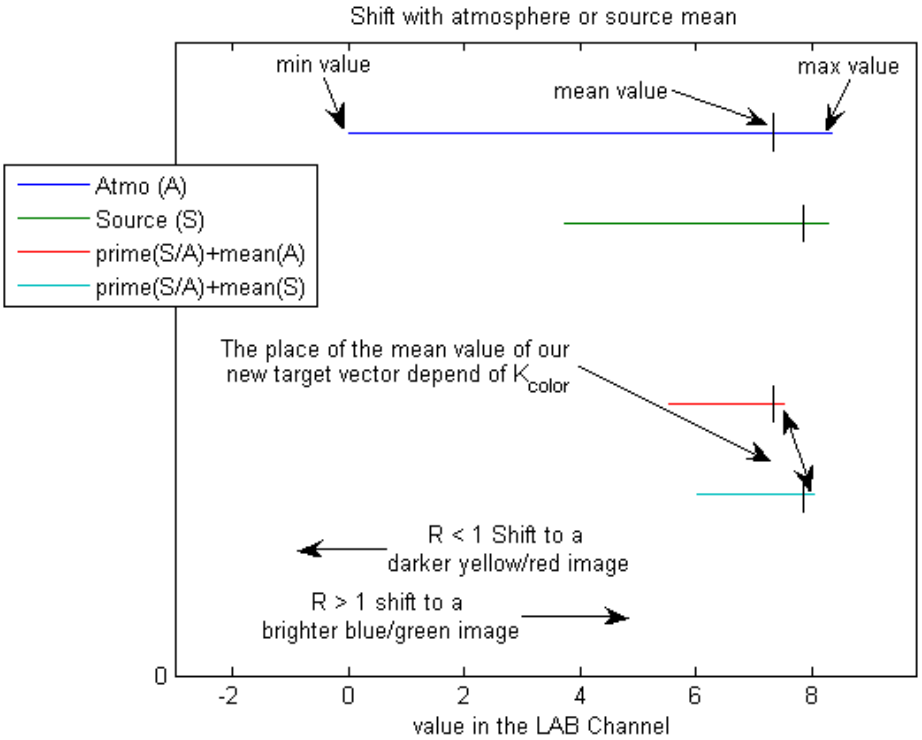


Figure 2 explaining of the shift

One of the problems raised by this formula for the value of the color channel is the value of the R and K.

After many tests over the images, we have noticed that K= 2/6 and R=1 is a good compromise. But in certain cases, although we have a “good” color transfer, there is a color cast over the image so we use another value for R.

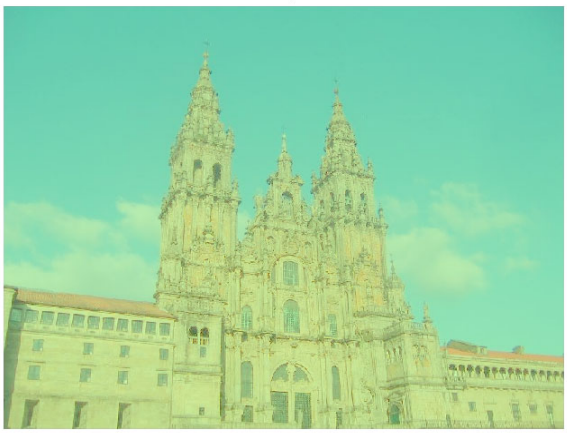
Here we give a good example where we had to set R = 0.4 to remove the greenish effect on the image. The R scaling can be assimilated to a gamma correction without affecting the color transfer itself (standard deviation).

Results

Original target image

Atmosphere image

Target image with R=0.4



Target image with R=1

Original target image

Atmosphere image

Typical image with huge color cast because of color dissimilarity, even with different values of R and K.

The problem here is that we do not make a clear distinction between the different colors that are present in the two images, i.e. if we have an image which is half green and half blue; this method is not suitable for the colors transfer because the mean of such an image would cause the tow colors to “melt” which has a bad effect on the image. Thus we introduce the clustering of the colors in the images.

2. Color transfer using clustering

In this section we introduce not only the notion of clustering for the images but also additional features such as cluster edge smoothing and image masks closing which will be discussed later on.

1. The clustering

The clustering here is basically done using the matlab function k-means which partitions the data points of the image into k clusters.
Given an initial number k of wanted clusters, the algorithm is implemented in the following way:
We first partition the data points into k clusters, and then compute seed points as being the centroids of the clusters of the current partition. The centroid is the means point of the cluster and is calculated as the arithmetic mean of the points in the cluster. After having computed the centroid, assign each coordinate point to the cluster with the nearest seed point. The algorithm stops when no assignment occurs.

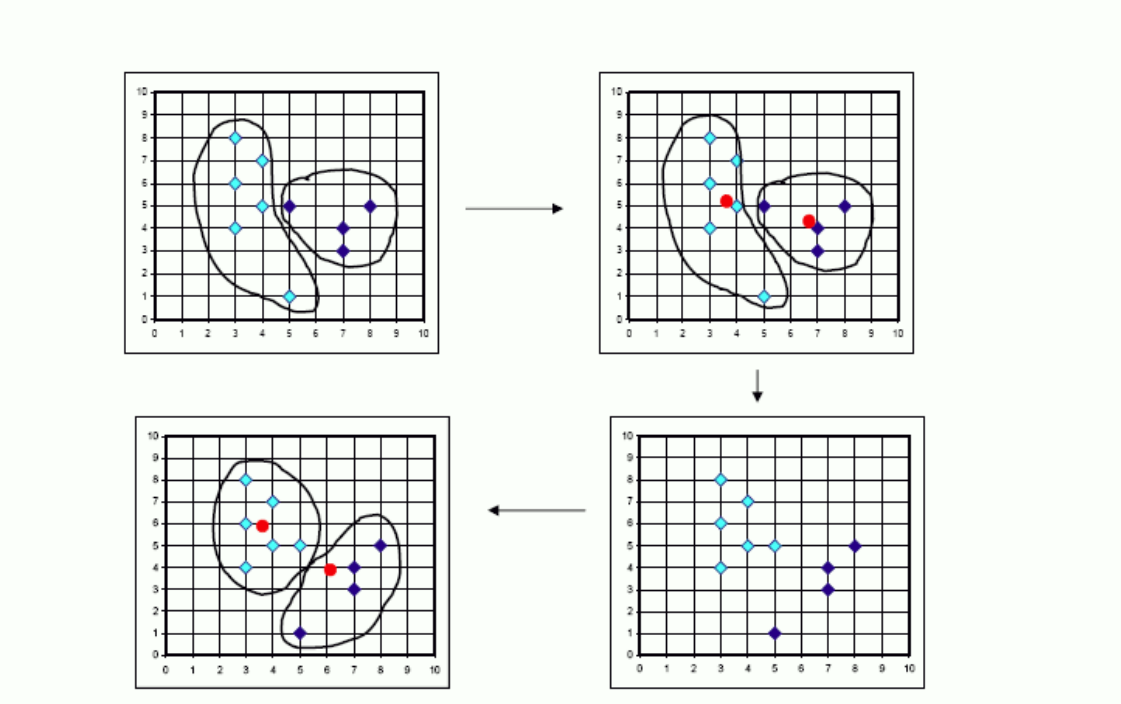
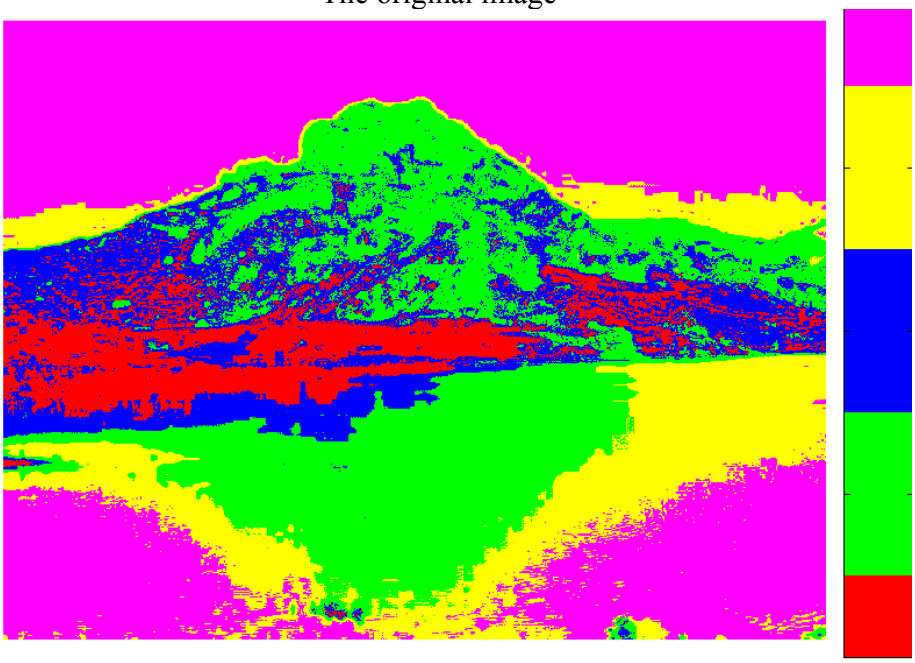


Figure 3 Red points represent the centroids whereas colored points the data points. Here we only needed two iterations to converge.

As we are using a color space where colors which are close in distance are close in color, the Lαβ color space is the ideal space for our calculations. The initial matrix that we give to the k-means function is the reshaped α*β matrix. We do not need the L matrix as it provides no information about the colors.

The original image



The image partitioned in 5 clusters

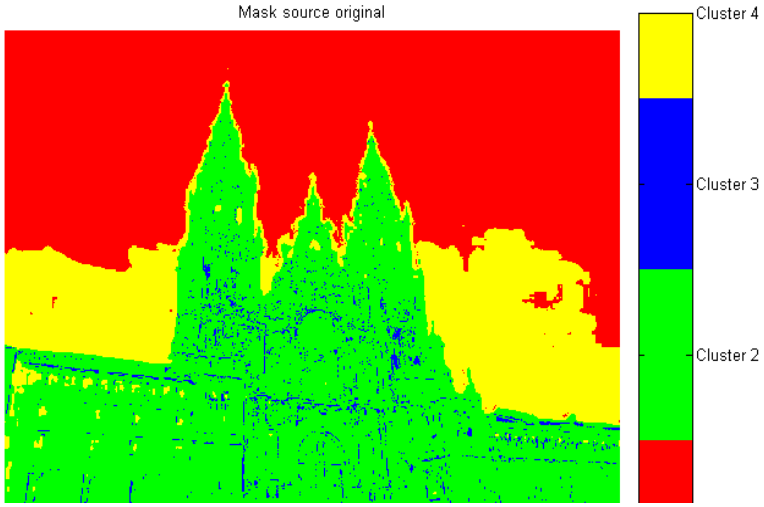
We found out that the way the algorithm assigns the colors with the clusters differs from one image to another. For example if in the source image cluster number 1 represents the

green color, it is possible that on the target image the green color is represented by the cluster number 3 for instance. This is why we had to swap the cluster positions in one image to match the clusters of the other image. The basic idea of the algorithm of the following:

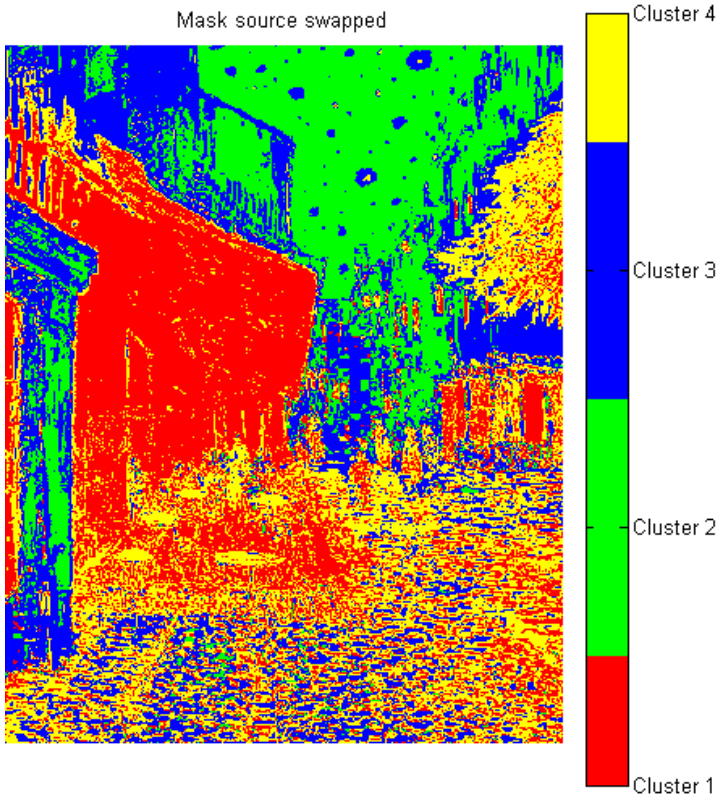
The algorithm avoids mapping two clusters of image A with only one cluster of image B: First, we compute the Euclidian distance between all the cluster's centroids in A with all clusters centroids in B: thus we obtain a centroid Euclidian distance matrix D.
After finding each mapping (a_map# -> b_map#), using the minimum value in D, we set the row of b_map# and the column of a_map# at some maximum value to avoid redundancies in mapping.

Original target Image

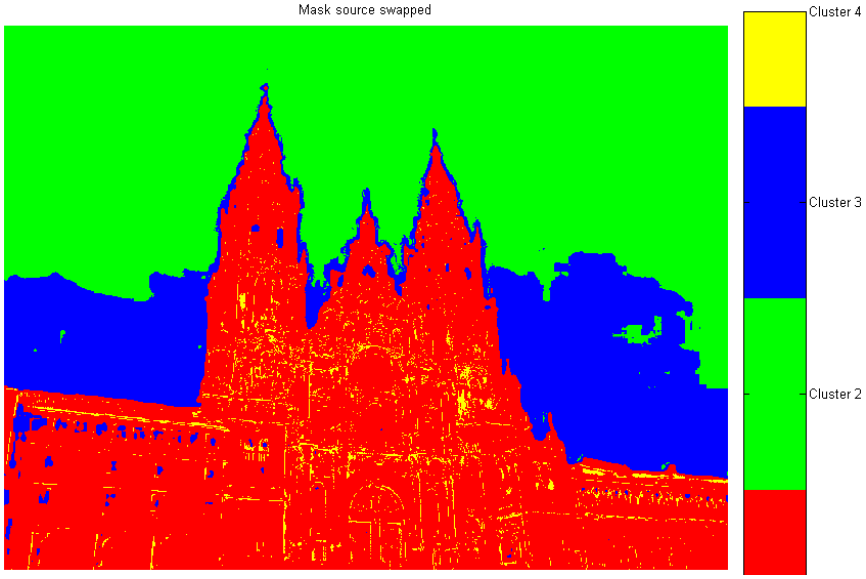
Atmosphere image



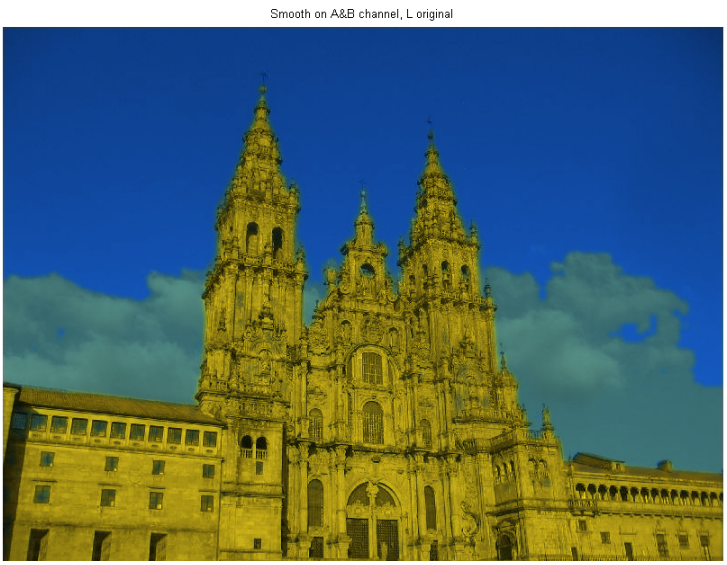
Original (random) target mask



Atmosphere mask



Swapped target mask



Target image with swapped masks

Target image with random masks

This algorithm works properly, but sometimes we can't choose a good mapping at the last step because we map the lasting two clusters together, which does not necessarily represent a minimal Euclidian distance.

Original target image

Source image

Swapping that failed

2. The smoothing of the image

Smoothing is a process by which data points are averaged in a relative way with their neighbors. This has the effect of blurring the sharp edges in the smoothed data. Smoothing is sometimes referred to as filtering, because it has the effect of suppressing high frequency signal and enhancing low frequency signal. There are many different methods of smoothing but in our project we only use Gaussian smoothing with the matlab function “fspecial.m”

The basic process of smoothing is very simple. We proceed through all the pixels of the image. For each data point we generate a new value that is some function of the original value at that point and the surrounding data points.

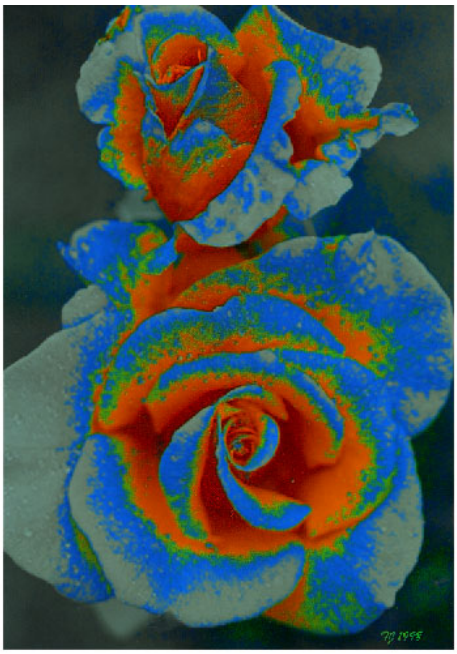
With Gaussian smoothing, the function that is used is a Gaussian curve with a mean=0 and a given variance. The greater the variance is, the more the image is blurred.

The reason we used the smoothing in our project is because in some images we had single pixels that belonged to a cluster that were surrounded by a majority of pixels belonging to another cluster. The tricky part was to find the value of the variance so as to remove this effect but without having too much blur effect. We found out that a variance of 3 was a good value.

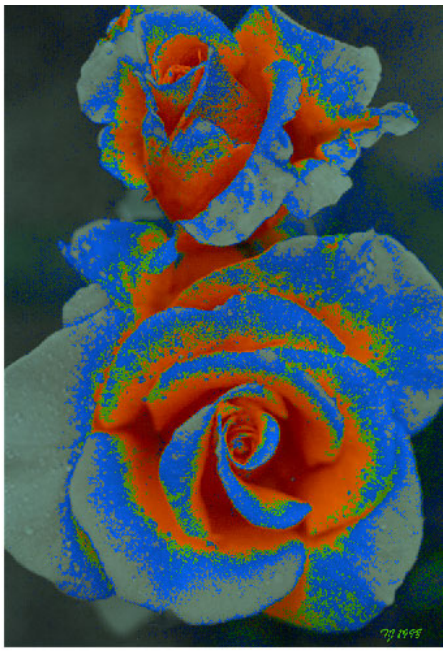
The smoothing is done only for the α and β channels, because there is no point in smoothing the luminance channel.

Original target image

Atmosphere image



lowpass filter of 4 pixels and variance of 3

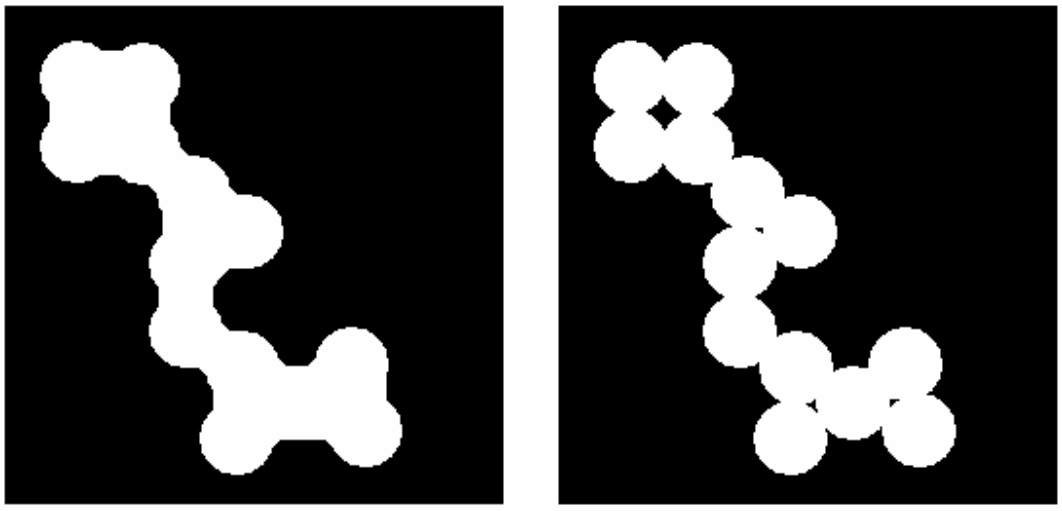


no filtering done

3. Closing of the image

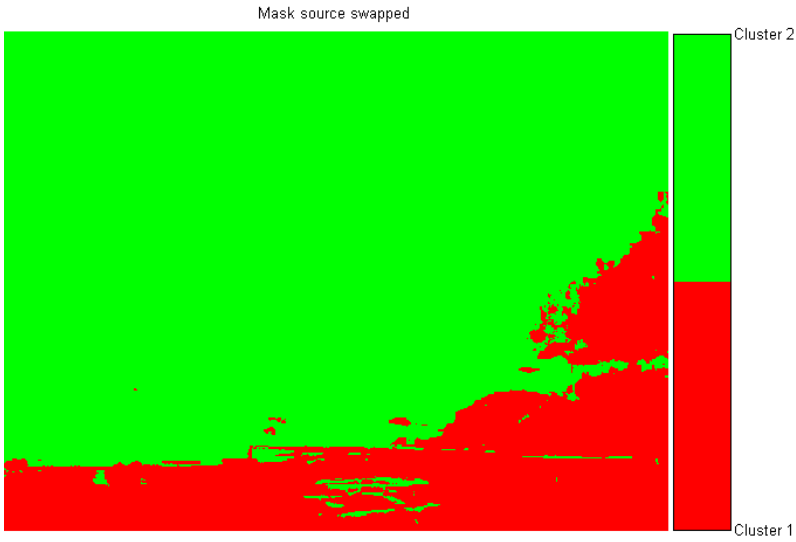
In this part we investigated another way to smooth the image, as was done previously in part 2. We used the matlab function `imclose` to join pixels that are surrounded by another cluster in the image by filling in the gaps between them and by smoothing their outer edges.

But then this smoothing technique works fine only for binary images and since we do the smoothing on the masks of the image, which in our case translates to only two clusters for this method, this is not very interesting because in most of the cases we dealt with images where the number of clusters is more than two.

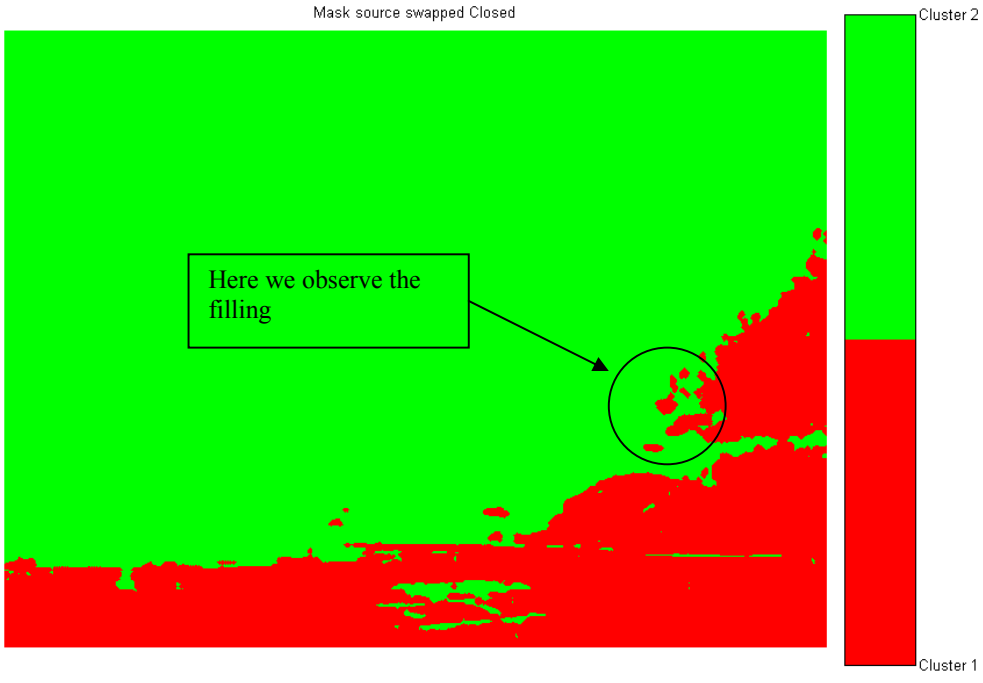


Morphological close operation on the image

Original image



No closing



Closing on the masks using a disk of diameter of 2 pixels

4. Luminance transfer

The luminance transfer is one of the hardest tasks to achieve in the transfer of colors. Several approaches have been studied which give different results in terms of final quality of image.

The first and simplest approach was to keep the original luminance of the target image, i.e. do the color transfer only for the alpha and beta channels. The transfer of colors in this case was not very successful, especially for images that contained very bright colors, which had as an immediate effect of saturating transferred colors. The reason for this is that the luminance contains also information about the image, and colors need this information to appear in the desired way. See images 4.1.x for a detailed view.

The second model consisted in transferring the luminance with respect to clusters. This meant that for every cluster, we transferred the luminance of the colors contained in the cluster. The results show that this is not the best model because the separation between the luminance levels in some cases is too sharps which leads to "bad" images. This effect could not be eliminated with the smoothing algorithm. The cases where this approach worked well is when, in the original source and target images, there was a clearly defined border between the colors in the images as it is seen in 4.2.x.
The third model is the luminance transfer using the overall statistics of the image, exactly in the same way as it was done in the color transfer without clustering. This transfer lead to the best results compared to the two models described above. See 4.1.x for a detailed view

4.1 –1st luminance examples:

Original target image

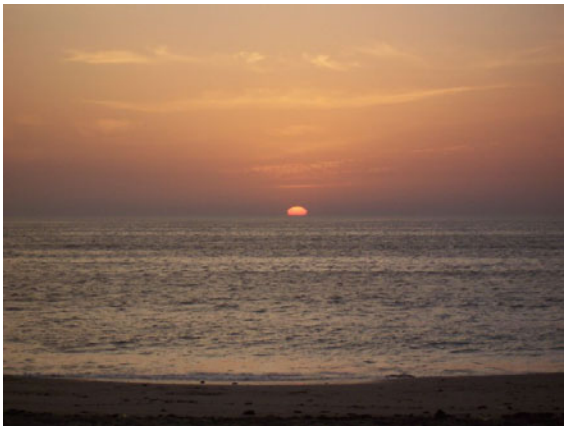
Atmosphere image

4.1.1 - Keeping the original luminance (saturation in the blue)

4.1.2 - Luminance transferred using clustering

4.1.3 - Luminance transferred using overall statistics

4.2 –2nd luminance example:



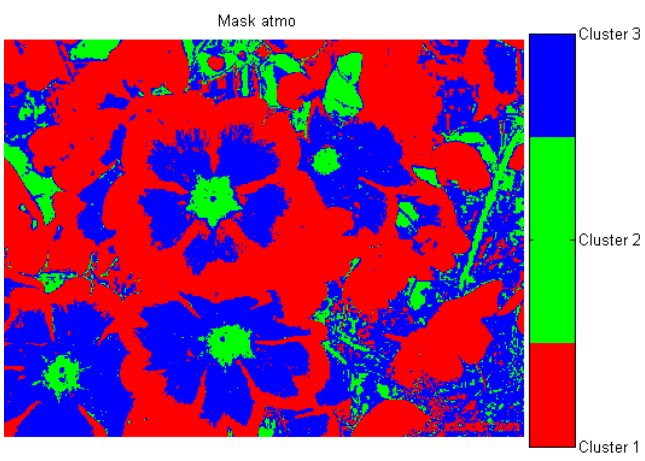
4.2.2 - Keeping the original luminance

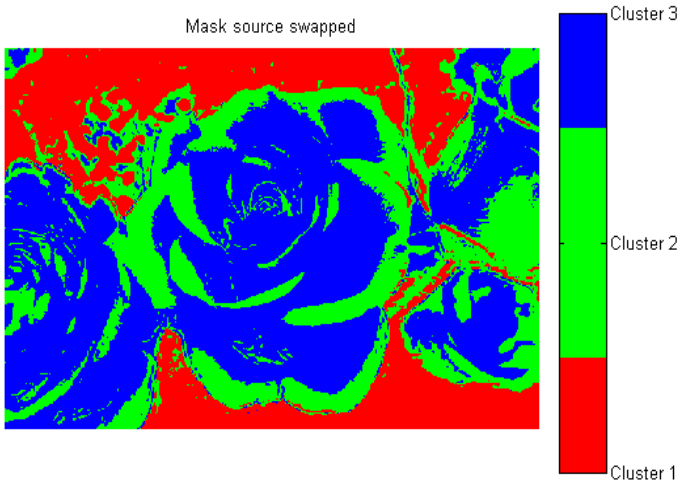
4.2.2 Luminance transferred using clustering

4.2.3 - Luminance transferred using overall statistics

5. Final results

The final image shown below is obtained using the best models (smoothing, optimal mean shift correction and overall luminance transfer and 3 clusters).The result is very satisfactory!





Smooth on A&B channel, L->luminace transfert

III. Conclusion

In this project we have seen that in order to manipulate colors we have to use a color space that is decorrelated. We have also seen that by simple scaling of the mean and standard deviation of an image, we are able to transfer one image's color properties to another. But then the result differs drastically depending on the parameters (R, K, number of clusters, smoothing factor, etc.) and the methodology used for the color transfer. We have also seen that initial properties of the image such as luminance, color saturation of both source and target images play an important role in the success of the transfer.
For instance, if the atmosphere has initial colors that are saturated and the target image is bright, the color transfer works very well.

Future improvements of the model proposed would be to detect the luminance and act accordingly, detect saturation of the image, map the clusters in a more appropriate way.

And finally….. We did it ourselves

Smooth on A&B channel, L->luminace transfert

[1] Erik Reinhard, Michael Ashikhmin, and Peter Shirley
[2] Ruderman et al.
[3] http://www.mrc-cbu.cam.ac.uk/Imaging/Common/smoothing.shtml
[4] http://www.mathworks.com/access/helpdesk/help/toolbox/images/imclose.html