Parallelized EM Algorithm in Cython: Gaussian Mixture Models and Application to Color Transfer in Imaging

Delanoue Pierre
ENSAE Paris
3rd Year Engineering degree
pierre.delanoue@ensae.fr

Meunier Dimitri
ENSAE Paris
3rd Year Engineering degree
meunier.dimitri@ensae.fr

Abstract

We demonstrate the use of Cython to speed up the computational time of the Expectation-Maximization algorithm for Gaussian Mixture Models approximation. As an application, we perform Optimal Transport on the colour space between images to perform colour transfer. All our codes can be found on the GitHub page https://github.com/DimSum2k20/Projet3A_ElementsLogiciels. The notebook EM *contains all our experiments (1D and 2D visualizations and color transfer).*

1. Introduction

The Expectation-Maximisation (EM) is an algorithm for maximizing the likelihood of a set of n independent variables $X_1, \cdots, X_n$. The EM algorithm is useful if we can write the likelihood of the model as

$$L(X_{1:n}|\theta) = \int_{\mathcal{Z}} L(X_{1:n}, z|\theta)dz$$

where $\mathcal{Z}$ is the space of latent variables and where $L(X_{1:n}, z|\theta)$ is easy to maximize in θ.
An example of such model is a mixture model where $\mathcal{Z} = \{1, \cdots, K\}$ is the set of component labels and $Z_i \in \mathcal{Z}$ assigns observation X_i to component Z_i.

After a presentation of the EM algorithm in the general case and in the case of Gaussian Mixture Models we explain how we implemented a GMM approximator and which part have been cythonized. Finally, we use the EM algorithm to perform approximated optimal transport on the color transfer task. For this, an innovative approach by [1] suggested to use GMM approximation (using the EM algorithm) as a pre-processing step of an Optimal Transport over the GMM space. This shows, among other things, that the EM algorithm is still widely used to solve modern problems.

2. Generic EM Algorithm

We denote by $l(\theta)$ the log-likelihood. We can marginalize each density over the latent space:

$$\begin{aligned} l(\theta) &= \log\big[L(\theta|x_{1:n})\big] \\ &= \log\big[\prod_{i=1}^{n} f(X_i|\theta)\big] \\ &= \sum_{i=1}^{n} \log(f(X_i|\theta)) \\ &= \sum_{i=1}^{n} \log \int_{\mathcal{Z}} f(X_i, z|\theta)dz \end{aligned} \tag{1}$$

The idea of the EM algorithm is the following: since it is difficult to directly maximize $l(\theta)$ we would like to repeatedly construct a lower bound on $l(\theta)$ (E-step) and then optimize the lower bound (M-step).
We can obtain a lower bound by using Jensen's inequality. For every observation i we put a density on $\mathcal{Z}$ that we denote by Q_i ($Q_i \geq 0, \int_{\mathcal{Z}} Q_i(z)dz = 1$). Then,

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n} \log \int_{\mathcal{Z}} f(X_i, z|\theta)dz \\ &= \sum_{i=1}^{n} \log\Big[\mathbb{E}_{Z\sim Q_i}\big[\frac{f(X_i, Z|\theta)}{Q_i(Z)}\big]\Big] \\ &\geq \sum_{i=1}^{n} \mathbb{E}_{Z\sim Q_i}\big[\log\big(\frac{f(X_i, Z|\theta)}{Q_i(Z)}\big)\big] \end{aligned} \tag{2}$$

This lower bound is valid for every sequence of distributions $(Q_i)_{i\in[1,\cdots,n]}$ over $\mathcal{Z}$. We want to find the sequence that gives the tighter possible lower bounds. We could consider a fixed point $\bar{\theta}$ and wants the inequality to be an equality at $\bar{\theta}$. The Jensen inequality becomes an equality if the term in the first expectation is constant with respect to Z.

$$\frac{f(X_i, Z|\bar{\theta})}{Q_i(Z)} = c(X_i, \bar{\theta}) \quad a.s.$$

This implies that $Q_i(z) \propto f(X_i, z|\bar{\theta})$ and since Q_i is a density we get

$$Q_i(z) = f(z|X_i, \bar{\theta})$$

We introduce the notation,

$$\begin{aligned} R_i^{\bar{\theta}}(\theta) &= \mathbb{E}_{Z\sim Q_i}\left[\log \frac{f(X_i, Z|\theta)}{Q_i(Z)}\right] \\ &= \mathbb{E}_{Z\sim f(.|X_i,\bar{\theta})}\left[\log \frac{f(X_i, Z|\theta)}{f(Z|X_i,\bar{\theta})}\right] \end{aligned} \tag{3}$$

By (2) we have,

$$\forall\theta \qquad l(\theta) \geq \sum_{i=1}^{n} R_i^{\bar{\theta}}(\theta) \qquad l(\bar{\theta}) = \sum_{i=1}^{n} R_i^{\bar{\theta}}(\bar{\theta})$$

The following proposition is the main motivation for the EM algorithm.

Lemma 2.1. $\forall t \geq 0 \qquad l(\theta_{t+1}) \geq l(\theta_t)$

Proof.

$$\begin{aligned} l(\theta^{(t+1)}) &\geq \sum_{i=1}^{n} R_i^{\theta^{(t)}}(\theta^{(t+1)}) \quad \text{by (2)} \\ &\geq \sum_{i=1}^{n} R_i^{\theta^{(t)}}(\theta^{(t)}) \\ &\quad (\theta^{(t)} \text{ is set to be the argmax of } \sum_{i=1}^{n} R_i^{\theta^{(t)}}(\theta)) \\ &= l(\theta^{(t)}) \\ &\quad \text{(we defined } Q_i \text{ such that the Jensen inequality} \\ &\quad \text{is an equality at } \theta^{(t)}) \end{aligned}$$

□

From this lemma we can make several remarks,

- A good stopping criterion is to fix a tolerance ϵ and stop when $l(\theta_{t+1}) - l(\theta_t) < \epsilon$
- There is no guarantee of convergence, we only know that the likelihood increase but it could be stuck in a local maximum.
- Finding the argmax in θ of $R_i^{\bar{\theta}}(\theta)$ is the same as finding the argmax of $\mathbb{E}_{Z\sim f(.|X_i,\bar{\theta})}\left[\log f(X_i, Z|\theta)\right]$

This enables us to introduce a generic version of the EM algorithm 1.

Algorithm 1: Generic EM Algorithm

Result: $\theta^{(T)}$ argmax of the log likelihood
Initialization: ϵ, $\theta^{(0)}$;
while $l(\theta^{(t+1)}) - l(\theta^{(t)}) > \epsilon$ **do**
  (E-Step);
  $\forall i \in [\![1, n]\!]$

$$Q_i(.) = f(.|X_i, \theta^{(t)})$$

  (M-Step);

$$\theta^{(t+1)} = \arg\max_{\theta} \sum_{i=1}^{n} \mathbb{E}_{Z\sim f(.|X_i,\theta^{(t)})}\left[\log f(X_i, Z|\theta)\right]$$

end

3. EM for Gaussian Mixture Models

In the GMM model, we consider observations $(x(1), \cdots, x(n)) \in (\mathbb{R}^d)^n$ sampled from a Gaussian Mixture Model:

$$f(x|\theta) = \sum_{i=1}^{K} p_k \phi(x, \mu_k, \Sigma_K)$$

where $\theta = (p_1, \cdots p_K, \mu_1, \cdots, \mu_K, \Sigma_1, \cdots, \Sigma_K)$, K is the number of components in the mixture, $\sum_{k=1}^{K} p_k = 1, p_k \geq 0$, $\phi(., \mu, \Sigma)$ is the density of the d-dimensional gaussian distribution with location parameter μ and scaling Σ. Directly maximizing the likelihood is difficult but we can introduce the latent variable $Z \in \{1, \cdots, K\}$ where $Z = k$ means that an observation comes from the component k of the mixture, $\mathbb{P}(Z = k) = p_k$. Thus,

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n} \log \sum_{k=1}^{k} f(X_i, \{z = k\}|\theta) \\ &= \sum_{i=1}^{n} \log \sum_{k=1}^{k} \phi(X_i|\mu_k, \Sigma_k) p_k \end{aligned} \tag{4}$$

The advantage of this model is that we can explicitly derive the formulation of the (E-step) and the (M-step). The EM algorithm in the case of GMM is the algorithm 2.

4. Implementation

The core of our implementation consists in building our own python packaged version of the EM algorithm. We took the same structure as the *Gaussian Mixture* class of scikit-learn ([2]), meaning that we have fit and predict functions.

Algorithm 2: EM for GMMs

Result: Parameters of the GMM
initialization: $\epsilon, p_k^{(0)}, \mu_k^{(0)}, \Sigma_k^{(0)}$;
while $l(\theta^{(t+1)}) - l(\theta^{(t)}) > \epsilon$ **do**
(E-Step);
$\forall i \in [\![1, n]\!], \forall k \in [K]$

$$Q_i(k) = \frac{\phi(X_i|\mu_k^{(t)}, \Sigma_k^{(t)})p_k^{(t)}}{\sum_{l=1}^{K} \phi(X_i|\mu_l^{(t)}, \Sigma_l^{(t)})p_l^{(t)}}$$

(M-Step);
$\forall k \in [K]$

$$p_k^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} Q_i(k)$$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} Q_i(k)X_i}{\sum_{i=1}^{n} Q_i(k)}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{n} Q_i(k)(X_i - \mu_k^{(t+1)})(X_i - \mu_k^{(t+1)})^T}{\sum_{i=1}^{n} Q_i(k)}$$

end

The challenge of this project was to develop a parallelised version of the EM algorithm. For this we used Cython. As explained in the previous sections, we took advantage of the fact that the E step is parallelizable to increase the learning speed. We obtained a speed-up by a factor 100 after having cythonized the E-step. As shown in the next section, it allows to derive GMM approximation in less than a minute on large cloud points.

5. Application: Color Transfer with Optimal Transport

Optimal transport ([3], [4]) is a set of mathematical tools to solve the problem of transporting masses from a source distribution to a target distribution in a mass preserving manner with minimum cost. Initially created to solve the problem of transporting an amount of dirt, it has been extended to the transport of any distribution onto another.

One fun application of Optimal Transport is color transfer. The task is to transfer the colors of one source image to a target image while preserving structure. For this, we consider the color space of each image. Our source and target samples are clouds of 3D points, each of whom encodes the RGB color of a pixel in a standard test image. We can then define a pair of discrete probability measures on our color space $[0, 1]^3$.

$$\alpha = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i} \qquad \beta = \frac{1}{M}\sum_{i=1}^{M} \delta_{y_i}$$

Figure 1 (top row) illustrates the distribution α and β in the colour space for two paintings. Both images are of sized (600*600), thus α and β are discrete distributions of 360,000 Dirac masses. Due to the size of the cloud points it is inconceivable to solve directly the vanilla optimal transport distance (called Kantorovich problem) as it requires to solve a linear programm of size (360,000 times 360,000). [1] suggested a two-step solution for this problem. In a first step we approximate α and β by GMM using the Expectation-Maximization algorithm (this is where our algorithm is helpful) and then we fit a specific optimal transport distance for GMM (called the Mixture Wasserstein distance MW_2). Figure 1 (second and third rows) illustrates the GMM approximation of the distributions. We compare our approximation (second row) with Scikit-Learn's implementation (third row). Results and computational time are almost the same.
Figures 2, 3, 4 and 5 present convincing color transfer results after having performed GMM approximation with our algorithm. For more details, see the reference Jupyter Notebook on the Github page of the project (https://github.com/DimSum2k20/Projet3A_ElementsLogiciels).

References

[1] A. D. Julie Delon. A wasserstein-type distance in the space of gaussian mixture models. 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] G. Peyré and M. Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11:355–607, 2018.

[4] C. Villani. *Optimal transport – Old and new*, volume 338, pages xxii+973. 01 2008.

A. Appendix

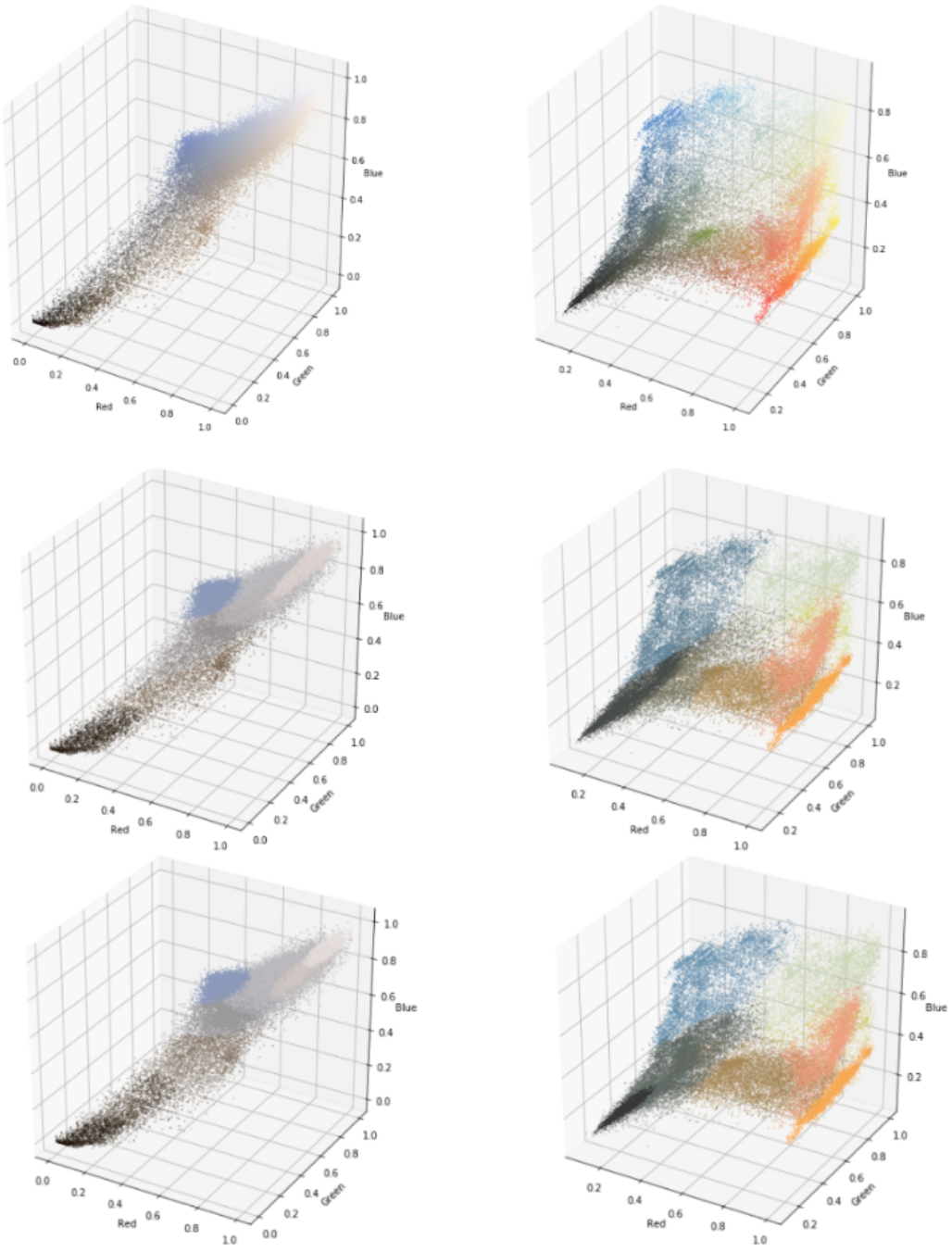

Figure 1. Comparison of our EM implementation with scikit-learn to cluster the color histogram of two images (see figure 5): true distribution (up), our EM implementation results with 8 clusters (middle), scikit-learn with 8 clusters (bottom).

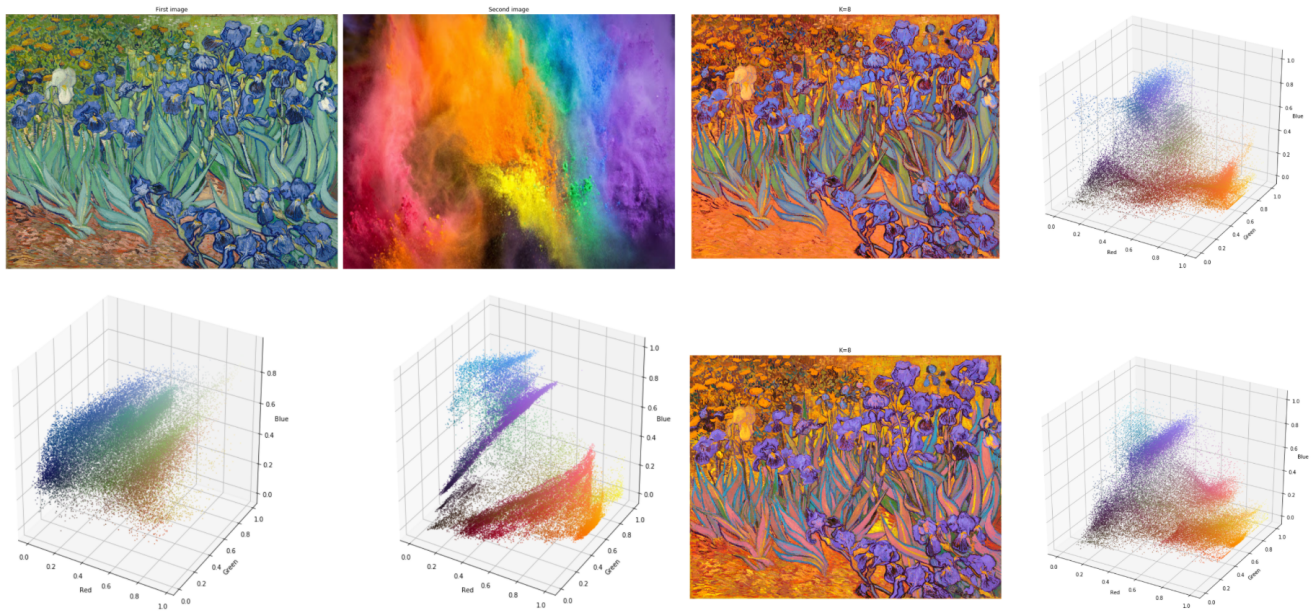

Figure 2. Application of the color transfer technique from an abstract colored image to a VanGogh painting: base images (upper left), true distributions (bottom left), our EM approximation and the target image after colour transport (upper right), scikit-learn (bottom right).

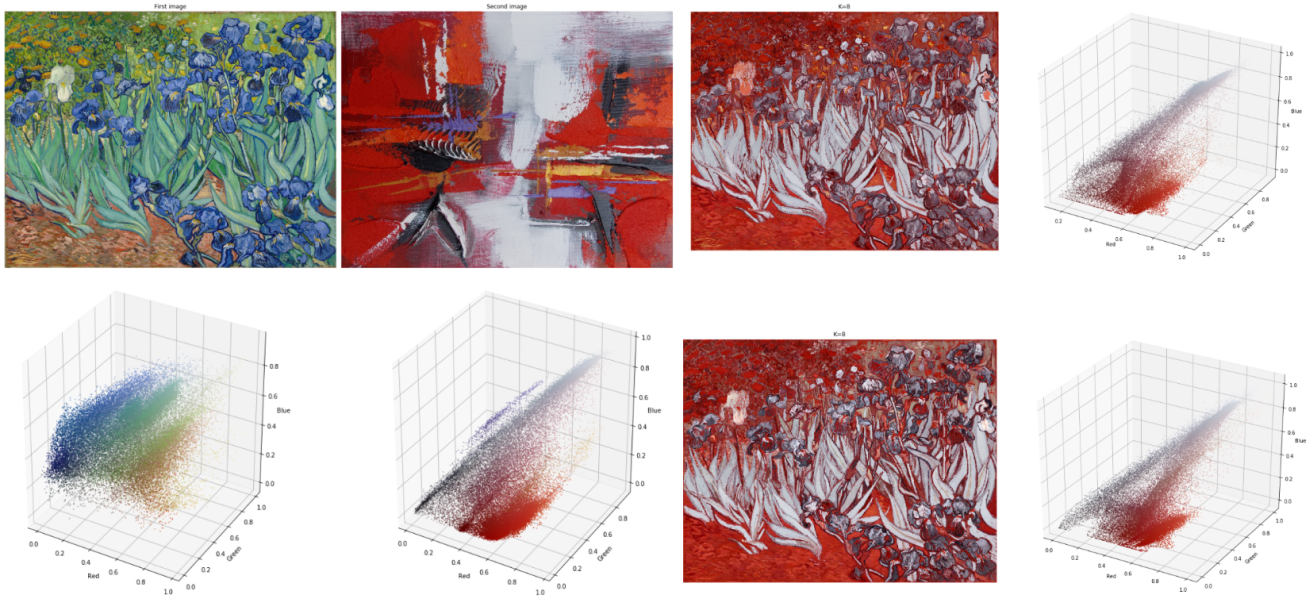

Figure 3. Application of the color transfer technique from a red painting to a VanGogh painting : base images (upper left), true distributions (bottom left), our EM approximation and the target image after colour transport (upper right), scikit-learn (bottom right).

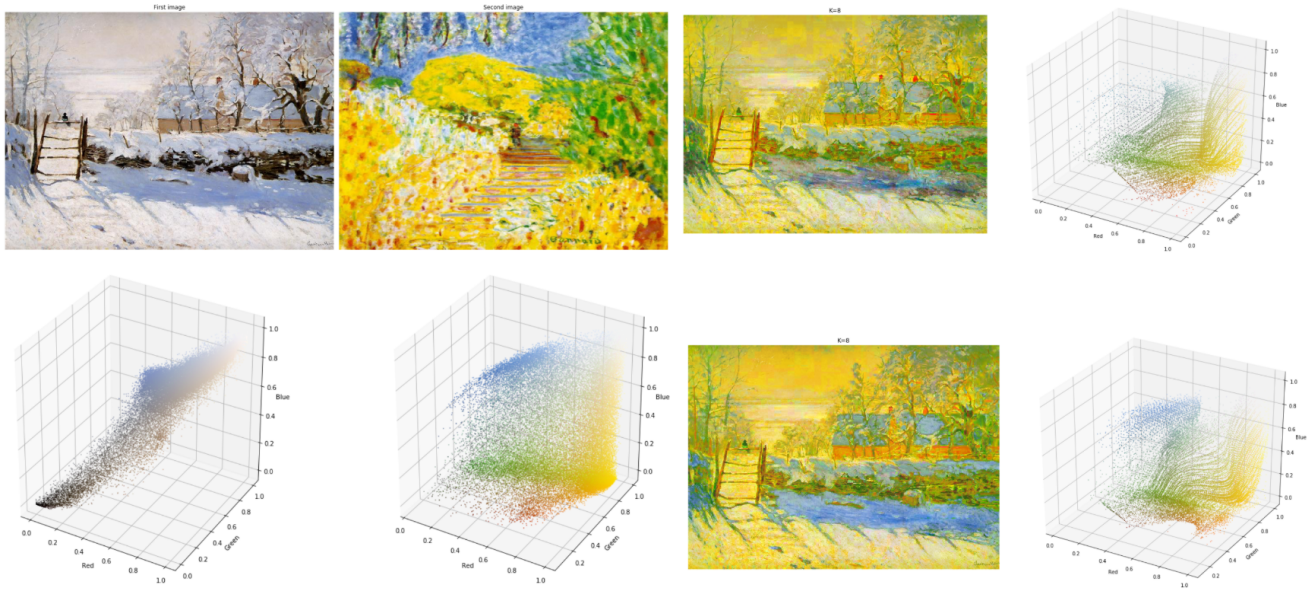

Figure 4. Application of the color transfer technique from a Bonnard painting (mainly yellow) to a Monet painting (mainly white) : base images (upper left), true distributions (bottom left), our EM approximation and the target image after colour transport (upper right), scikit-learn (bottom right).

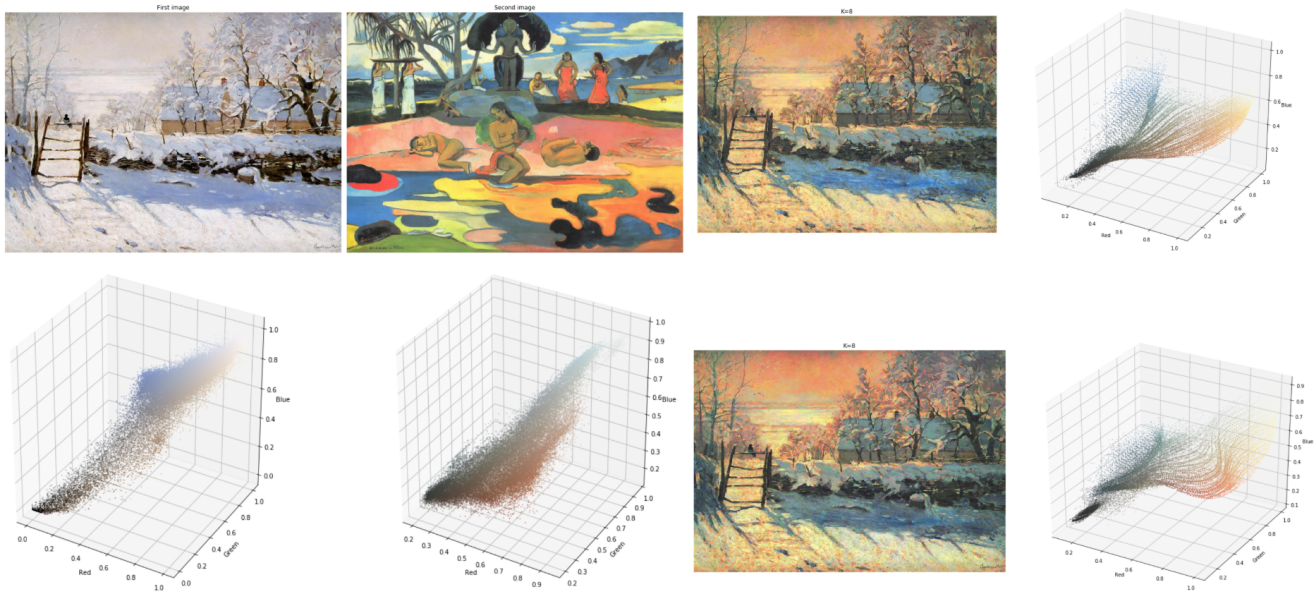

Figure 5. Application of the color transfer technique from a Gauguin painting (mainly yellow) to a Monet painting (mainly white) : base images (upper left), true distributions (bottom left), our EM approximation and the target image after colour transport (upper right), scikit-learn (bottom right).