# ADM - Project 4
# Paint-by-numbers image generation using $k$-means clustering

Gonzalo Solera

May 2020

## 1 Introduction

I wanted to make a personalised gift to my girlfriend. She likes to paint and I wanted my gift to be related to her hobby. So I thought of buying a personalised paint-by-numbers product from a company like this one: https://www.miicreative.com. Their service basically consists on uploading a picture, converting it in a paint-by-numbers template and ship to the customer the template and a set of numbered paints, each of them assigned to a number of the template.

However, I decided to code myself a similar algorithm than the one that this company must use to convert the images, and make the gift myself, making it related with my personal hobby as well.

The algorithm should basically consist on a $k$-means algorithm. But as I supposed beforehand, it actually needs some non-trivial features to make it work.

## 2 The algorithm

The picture that I wanted to convert was a picture of our dogs:

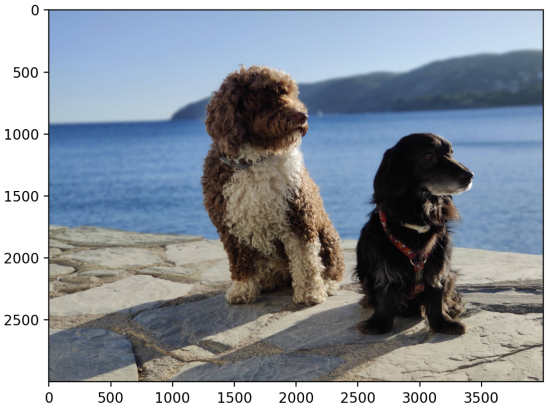


Figure 1: Original image

The desired output of the algorithm would consist of a modified picture that contains just $k = 10$ distinct colours. So the original picture should be simplified to just using $k$ colors. We first find which $k$ colors would be representative of the original image, which is accomplished by a simple $k$-means algorithm:

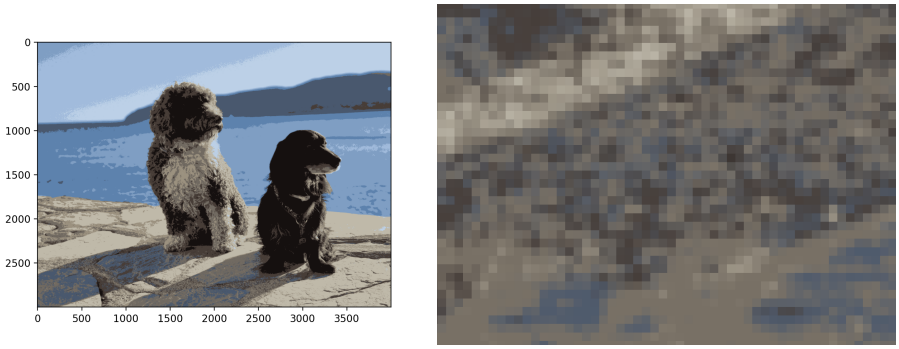


Figure 2: $k$-means clustering without any post-processing step

The original image contains $12M$ pixels, and to choose the $k$ representative colors out of all of them is not very efficient. We sample a parameterized proportion of all the pixels and run the $k$-means algorithm only on these to speed up the computation.

That however, is not enough since if we naively replace each pixel's color by one of the $k$ representative colors, the resulting image still contains too much

detail, and separated areas of the resulting picture are too small to paint or even to write a number inside:

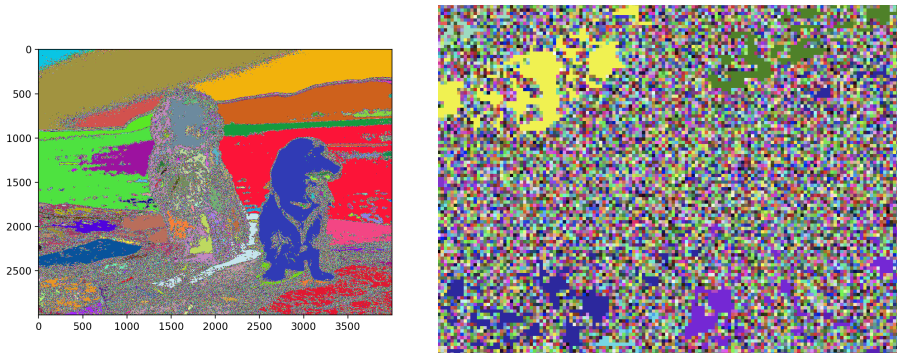


Figure 3: Pixel groups of $k$-means clustering without any post-processing step

To solve it, we have to merge separated pixel groups of different representative colors such that there are not too-small pixel groups. This requires the following two steps:

- Convert the image to a graph representation and find out a way of choosing two neighbour pixel groups to merge into one.
- Find out a way of merging two given neighbour pixel groups into one. We want the transformation to preserve the essence of the original picture, making the resulting image painted by only $k$ colors as close as possible to the original one.

The way to convert the image into a graph representation is very costly, since it requires to:

- Run a breadth-first-search starting from each pixel to find its respective pixel group.
- Create a node for each pixel group.
- Find the contiguous pixel group's nodes to create an edge connecting both pixel groups using another breadth-first-search starting from each pixel group.

The graph representation needs to allow for an efficient way of choosing two neighbour pixel groups to merge, since a naive any-to-any inspection would have a cubic asymptotic cost, making it inviable for large images like mine: $(12 \cdot 10^6)^3 = 1.728 \cdot 10^{21}$. Hence, the proposed algorithm uses an elaborated way of reusing information to allow for an inspection in cuadratic cost instead. By applying such algorithm, we obtain much bigger pixel groups (above a parameterized threshold size):

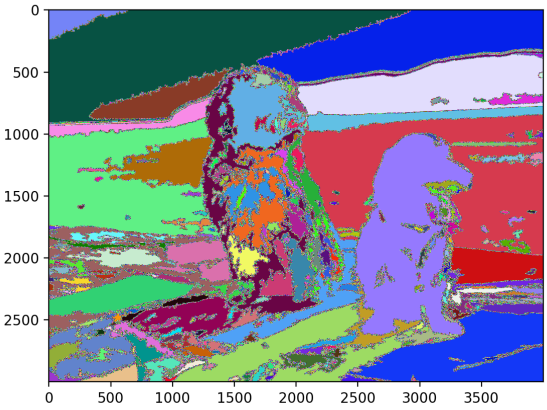


Figure 4: Pixel groups of $k$-means clustering after post-processing step

We then obtain a much more simplified image of the original one, containing only $k$ distinct colors and big pixel groups:

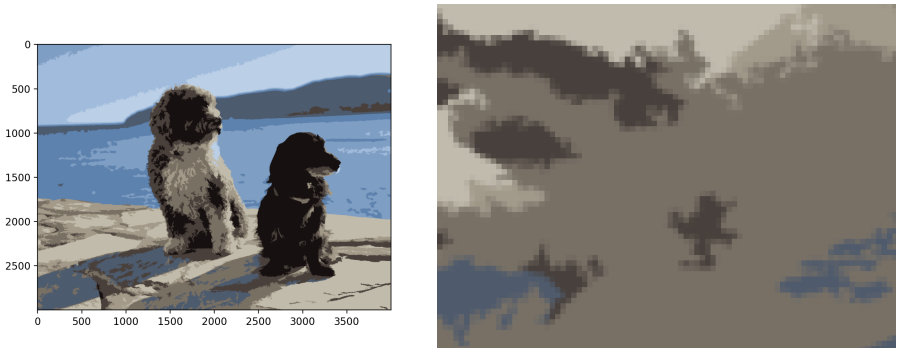


Figure 5: Resulting image

# 3 Future work

- Find a metric to evaluate the complexity of the shapes of the pixel groups and optimize it. There shouldn't be complex and strange shapes to paint, since they will be painted by hand and with tools aimed for a detail level of several orders of magnitude greater.
- Find out a proper way to automatically tune the hyperparameters of the algorithm.

# 4 Conclusion

The project has been fun to do, since it involved a real gift with a personal touch. It also has a beautiful use of a simple machine learning algorithm. Although the application of $k$-means is just one of the multiple steps, the data mining component of the project is still present, since the challenge of the other steps reside in the difficulty of writing efficient algorithms for massive data.