Colorization of Grey Scale Images

Dominik Benk
ČVUT–FIT
benkdomi@fit.cvut.cz

May 11, 2021

1 Introduction

There exist many methods that address problem of colorization of gray scale images. There has also been a strong tendency to automate this process, ideally not to require any additional human assistance. Historically the images had to be segmented and manually coloured, which not only consumed a lot of time, but also caused many inaccuracies due to human error. Even though many fully automated techniques were developed in recent years thanks to machine learning algorithms (see for example [1]), in this paper we will not focus on them as they require a non trivial amount of data (images) as well as computing power to train them.

We will present a method that requires only a little human interaction, involving painting of a few so called scribbles, and possibly making the whole process of colorization entertaining thanks to a simple user interface. This project was mainly inspired by Colorization Using Optimization [2], whose authors presented this approach of marking the grey scale images and provided its Matlab implementation.[1]

2 Colorization Algorithm

As authors [2] stressed, the whole idea of optimization is based on assumption, that pixels that are close to each other and have similar intensities, are also expected to have similar colours. Based on this they introduced a cost function, that if minimized yields the coloured image. Algorithm itself performs the following steps:

1. Collects both grey scaled as well as the scribbled image and converts RGB to YUV, i.e. the intensity and two chrominance channels respectively

2. Determines the colour mask, which is the difference between grey and scribble images

3. Calculates the weight matrices for the system of linear equations. This is given by the loss function that minimizes sum of squared distances of all pixel from weighted sum of its neighbours. Mathematically this can be expressed as

$$loss(U) = \sum_{i,j}^{m,n}(U_{i,j} - \sum_{k,l \in N_{i,j}} W \cdot U_{k,l})^2$$

   where $U_{i,j}$ is the colour channel on i,j coordinate on $m \times n$ image, $N_{i,j}$ represents the indices of neighbouring pixels and W is the weighting function. Inspired by the original article, weight is determined by intensity Y as

$$W = e^{-(Y_{i,j}-Y_{k,l})^2/(2\sigma_{i,j}^2)}$$

4. Runs the optimization algorithm that minimizes both loss(U) and loss(V). This can be achieved in many ways, but for our purposes we utilized `scipy.sparse.linalg.spsolve` that assumes sparse matrices.

5. Converts YUV back to RGB

[1] Note that [2] also considered the time dimension as they also colorized videos, but we will focus only on images.

3 Results

We managed to create a functional graphical interface, that allows user to paint and colorize in the same window. Unfortunately, we can not directly compare the results with the paper as authors provided only marked images without their grey counterparts. Nevertheless, results appear to be very close to reality if accurate scribbles are provided. To give an example of how well this algorithm can estimate the true colours, we show colorization of python and comparison to its original image (see 1, 2, 3)

4 Issues, Discussion and Conclusion

We also encountered several issues, namely the limitation of `opencv-python` GUI, that does not for example support interactive buttons and overall it is quite basic (which is not necessarily a bad thing).

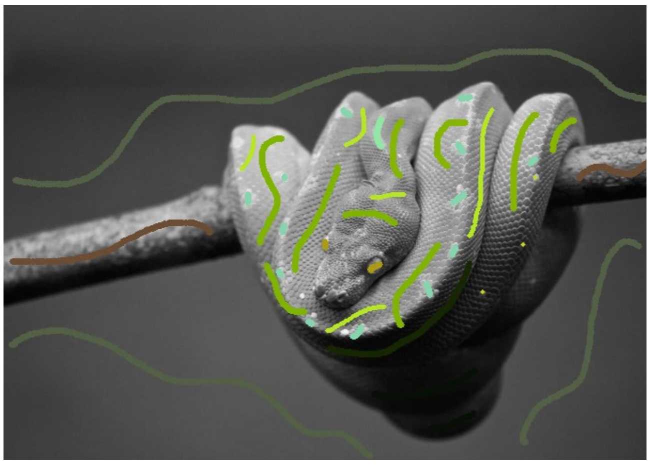

Figure 1: Scribbled

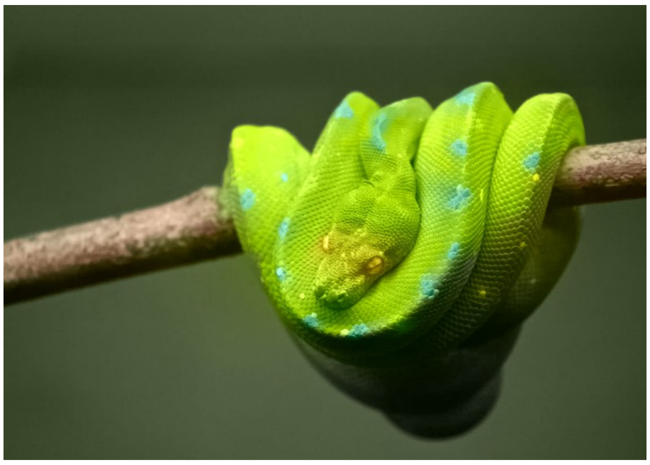

Figure 2: Colorized

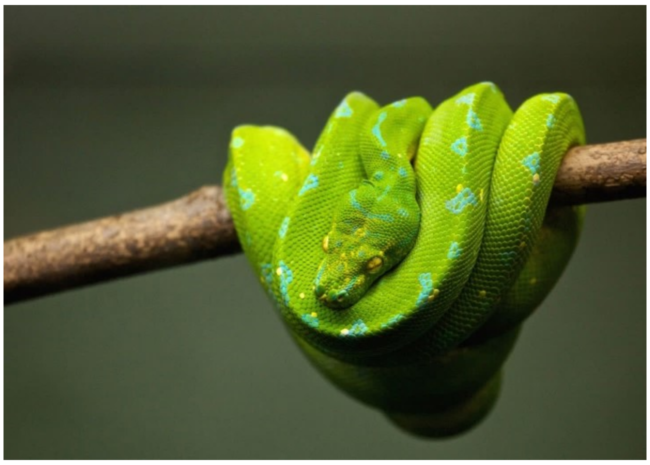

Figure 3: Original

Also, as the algorithm is quite computationally demanding, in case of larger images it can take 10s of seconds to colorize. This offers a space to parallelize the iteration over each image pixel or to find a solver that is more suitable for this task.

5 Conclusion

Even though there are more automated methods nowadays, optimization still provides a great alternative. It could also for example help with labelling some of grey scale images for which there exists no coloured counterpart and could then represent a gold data for neural network training.

References

[1] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European conference on computer vision*, pages 577–593. Springer, 2016.

[2] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. https://www.cs.huji.ac.il/ yweiss/Colorization/, 2004.