**REPORT FOR CMIT SUMMER SCHOOL**

**SIMPLE LINEAR ITERATIVE CLUSTERING FOR FURTHER MODIFICATION IN USAGE OF PEBBLE MOSAIC SEGMENTATION**

Shavarsh Melikyan

22 August 2022

The following report represents information related to the Simple Linear Iterative Clustering Algorithm(s) and with examples and their usage in further.

Code of the programme is written in Python.

Used libraries for the SLIC basic algorithm:

* Skimage (segmentation, util, io); functions: slic, mark\_boundaries, img\_as\_float
* Argparse\*
* Matplotlib (pyplot)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Main Problem*

The aim of this report and the project itself is to make/try/test dividing pictures into superpixels by the implementation of Simple Linear Iterative Clustering Algorithm

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Imported Libraries*

Most important library used for solving the problem is the Skimage with its util and segmentation sublibraries’ functions called slic, mark\_boundaries, img\_as\_float. The algorithm itself segments the image (no difference if it is a JPG or PNG or any other sort of image file) using k-means clustering algorithm; mark\_boundaries is responsible for returning the image with boundaries between regions labeled and img\_as\_float converts the image into floating point format, more precisely: from an unsigned 8-bit integer to a floating point number data with pixel values between [0, 1]. IO is responsible for the Input-Output of the process.

Matplotlib’s role is just to show up results on a screen and argparse is an optional element which is usually being used for activating the code through terminal so the computer can find the locations of directories where images are saved and/or have to be saved for skipping the “hard coding” (in this case i.e. giving strict addresses of files).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Process*

First of all in the code it can be seen that the libraries are being imported for the use, afterwards, because the work is done in an interactive environment (Spyder), the “hard coding” was used by making a dictionary and indicating the a key for accessing the name of the image (in the case of the provided code that key is “image”).

The next step is, by using IO and img\_as\_float, to change the data type of the image for usage in the following part of the code.

Last step is running a loop where the list of numbers used in a loop are representing the number of segments into which the image has to be divided; function slic works which returns a 2/3D array (in this case: 2D) which is then being plotted and returned with the image boundaries by mark\_as\_boundaries function.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Observations and other possible codes*

For the observations part it would be interesting to note what is happening when using additional parameters for the slic function (all parameters both optional and required ones can be found [here](https://scikit-image.org/docs/dev/api/skimage.segmentation.html?highlight=slic#skimage.segmentation.slic)); interesting effects occur when trying to play with the values of sigma and compactness. Results show that high compactness value makes superpixels look more easily for eyes in terms of being more systemized/squary; for the case of sigma value’s increase it tends to show the division of different RGB zones more strictly, hence making easy the segmentation of particular objects from the image; however it makes the programme to work slower and possibly increases its complexity (need to be checked). The reason for it is (possibly) the increased number of iterations which has to be done for making the programme implement the clustering algorithm.

Also it is interesting to mention other possible ways of programming the algorithm. Skimage looks like a fast-working library with considerably smaller complexity, however there is a faster one called OpenCV which, even though it is an image patterns analysis package, provides SLIC’s variant, which is considered faster (but less detailed and low accuracy).

Besides all there are some codes that are written from scratch which, based on the codes, have to have smaller complexity, but are working slower and in some specific versions of Python programming language\* (e.g. 2.7)

\*This report includes observations ONLY using Python’s libraries/capabilities; there are other libraries in R or MatLAB or Julia as well and mainly they all are the same kind.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Conclusion*

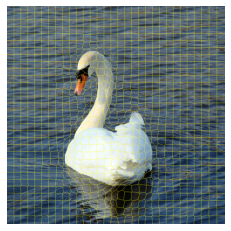
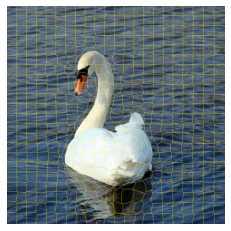
SLIC Algorithm works with quite high accuracy and speed by using the skimage library of python.

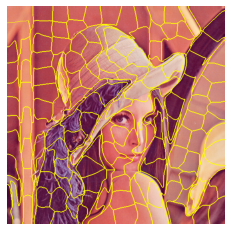
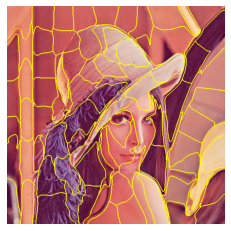
The code passed through all required tests and possible changes.

The algorithm is ready for its further usage

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Examples of Results (upper left to right and down 1,2,3,4,5,6)*





1 – Picture without any modifications

2 – high Sigma level

3 – high Compactness level

4, 5, 6 – different number of clusters made

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Further Steps*

1. Understand the complexity of the code and compare it with others and other parameters too for knowing the trade-off
2. Continuing working through the paper of Pebble Mosaics by then examining the smoothing of boundaries, pebble geometry and 3D look of images