Voronoi Tessellations

stochastic13

July 30, 2018

Contents

1 General working

The script takes an input image, converts it into a `numpy` array for further processing. There are several ways to initialize the clusters for which the Voronoi maps will be computed - random (default), based on a predefined cluster-map (x,y) or a predefined probability map. Following this, the script computes distances for each pixel to all the cluster centers and picks the least distance and assigns that cluster number to it (like in k-means). This is followed by either averaging of RGB channels for each cluster or exchange of channels as specified by options. The two methods to compute the actual maps are `low_ mem` and `fast`, of which the former is recommended.

2 Syntax

The function `VoronoiMain.py` is called in the terminal with additional options.

```
VoronoiMain.py [-h] [--rescale RESCALE] [--border BORDER]
                   [--method METHOD] [--threshold THRESHOLD]
                   [--clusmap CLUSMAP] [--probmap PROBMAP]
                   [--channel {r,g,b,rand,rb,rg,bg,randdual}]
                   [--verbose VERBOSE] [--seed SEED]
                   [--gaussianvars [GAUSSIANVARS [GAUSSIANVARS ...]]]
                   input output cn

positional arguments:
  input                 Input Image file
  output                Output Image file
  cn                    Number of clusters (default = 0.1*size)

optional arguments:
  -h, --help            show this help message and exit
  --rescale RESCALE     Rescaling factor for large images
  --border BORDER       Make border [1/0]?
  --method METHOD       fast vs low_mem methods. Default is low_mem.
  --threshold THRESHOLD
                        Only for borders. Threshold distance.
  --clusmap CLUSMAP     Load a specific cluster map as tab-separated text file
  --probmap PROBMAP     Load a 2D probability map for cluster generation
  --channel {r,g,b,rand,rb,rg,bg,randdual}
                        Whether to tessellate along only R,G,B or
                        combinations?
  --verbose VERBOSE     Print progress?[1/0]
  --seed SEED           Seed for PRNG
  --gaussianvars [GAUSSIANVARS [GAUSSIANVARS ...]]
                        Only for gaussian probmap
                        (mx,my,sigmax,sigmay,corr(opt),spacing(opt))
```

3 Options

3.1 Required Arguments

The three positional arguments are required to run the program. All other options are optional.

input Full path (or just the name for files in the same directory) to the input image file readable by `Image.open` command in `PIL`.

output Full path (or just the name for files in the same directory) to the output file. The file will be saved as a `.jpg` file

cn The number of cluster centres. **Not all clusters will have pixel membership at the end**. Furthermore, when using `probmaps`, if the cluster lies outside the image region, it is still a valid cluster but might not have any membership. For explicitly loaded cluster maps, try to keep the clusters within the image region.

3.2 Method

There are two methods available to compute Voronoi memberships. **Recommended method is** `low_ mem`. The option can be specified as `fast` or `low_ mem` following the `--method` option.

low_ mem This takes the input array and the cluster centres, initializes an empty array the size of the image and iterates over all the pixels to compute their membership. It uses a simple nested-for loop structure and is not vectorised, and hence, slow. But this occupies significantly less memory than the other option which makes this the recommended choice. *The progress is printed if* `verbose` *command is set. The printing uses a carriage-return symbol to print live percentages, which may not give the desired output in Linux,Mac or many other terminals.*

fast This takes the input array and the cluster centres, initializes an empty array the size of the image and uses a vectorized code to compute the distances and the memberships with `numpy` broadcasting. It is faster than the above method but uses significantly higher memory and may cause the computer to crash/hang.

3.3 Cluster-maps

A custom clustermap can be loaded by providing the file name to `--clusmap` option. The file must have x-values in column 1 and y-values in column 2 with tab-separated columns. Try to keep the clusters in the image region since the algorithm excepts outside-region centers which may give weird results.

3.4 Probability-maps

Instead of the custom clustermaps, probability maps can also be loaded to allow for center initialization according to that probability. An inbuilt function converts the probability map into a Cumulative Distribution Function and applies Inverse Transform Sampling to sample from this distribution using a uniform random distribution (`numpy.random.rand`). The functions are custom made and hence slow, inefficient and only moderately accurate. :)

An inbuilt function to have a multivariate (2-dimensional) gaussian distribution of the probabilities is also available via the `--gaussian` option. The algorithm first generates a probability array using the following:

$$p = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-c^2}} \exp\frac{-1}{2-2c^2}\left[\frac{(x-mx)^2}{{\sigma_x}^2} + \frac{(y-my)^2}{{\sigma_y}^2} - \frac{2c(x-mx)(y-my)}{\sigma_x\sigma_y}\right]$$

`mx`,`my`,`c`,`sigma` are the means for x and y, the correlation, and the sd for either respectively.

probmap An input 2D array with probabilities at different points. The size is free and will be coerced to the size of the image internally. Given a uniform random number, first a marginal distribution of x is computed from which a marginal CDF of x allows inverse transform sampling to get x. Given a x, the probability distribution of y given x is used to get the corresponding CDF and y. Currently, matching the random uniform variable to the CDF value is done without any interpolation, i.e. the closest value available in the given array is chosen. Hence, larger the array, more accurate the sampling. But larger arrays might cause memory problems. Enter `gaussian` to use the internal gaussian function to generate the cluster centres.

gaussianvars Allows changing the mx,my,sigmax,sigmay,correlation,spacing for the inbuilt gaussian function. Ignored unless the `probmap` is set to `gaussian`. `spacing` is used to generate the size of the probability array and is directly put into `numpy.linspace` to generate x and y, followed by `numpy.meshgrid` before feeding into the equation.

3.5 Channels

This allows more complicated averaging once the clusters have been computed. Several options are available.

r,g,b Any of these options averages only one of these channels. Each cluster has 50% chance to be averaged and 50% chance to stay unchanged

rg,rb,bg Each of these options exchanges the values of different channels, eg. rg exchanges the red-green channel values in half of the clusters at random

rand For each cluster, randomly averages one of the channels.

randdual For each channel, randomly exchanges any two channels. Can also keep the cluster unchanged

3.6 Border

This allows computing black borders of the clusters. Currently, a border point is a point for which the difference in distances to the closest two clusters is less than the `threshold` value. This creates uneven sized borders but in a more geometrically regular sense. More options in future versions perhaps!

border Set 1 for true. Default is 0 for false

threshold Threshold value. Coerced to int internally

3.7 Other Options

verbose Whether to print messages on the standard output or no. 1 for true,0 for false. Default is true. The percentage progress uses a carriage return character to allow for printing the next update on the same line. This may cause this printing to not function properly in some systems.

rescale The input image is rescaled by this factor.

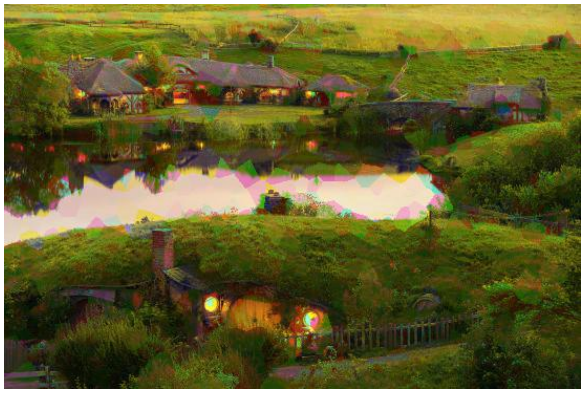
seed The seed for the Pseudo Random number generator. Fed to `numpy.random.seed`.

4 Samples

Figure 1: source image

(a) Default options, 2000 cn

(b) Gaussian, vars = 0.3,0.8,90,150



(c) Channel rand

(d) Channel randdual