Statistics computation description

May 22, 2024

1 Maximum

99th percentile of luminance values.

Function `numpy.percentile()`

2 Minimum

1th percentile of luminance values

Function `numpy.percentile()`

3 Mean

Arithmetic mean - sum of luminance values divided by their number, which here is the size of the image

Function `numpy.mean()`

4 Variance

Variance of the array elements, measure of the spread of a distribution of luminance values in image. Computed as average of the squared deviations from the mean:

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)$$

where μ is an expected (mean) value. Maximum likelihood estimate of the variance is used (by keeping the `ddof` parameter equal to 0)

Function `numpy.var()`

5 Skewness

Skewness is calculated as Fisher-Pearson coefficient of skewness:

$$g_1 = \frac{m_3}{m_2^{3/2}}$$

where m_i is the i-th central moment:

$$m_i = \frac{1}{N}\sum_{n=1}^{N}(x[n]-\mu)^i$$

where μ is the expected (mean) value.
Function `scipy.stats.skew()`

6 Kurtosis

Kurtosis is computed as fourth central moment. Fisher's definition is used, so 3 is subtracted (to give 0 result for normal distribution).

$$g_2 = \frac{m_4}{m_2^2} - 3$$

where m_i is the i-th central moment:

$$m_i = \frac{1}{N}\sum_{n=1}^{N}(x[n]-\mu)^i$$

where μ is the expected (mean) value.
Function `scipy.stats.kurtosis()`

7 Directionality

Directionality describes the stregth of material orientation. First Fourier amplitude spectrum is computed from the luminance values of the image. Complex values are computed with function `numpy.fft.fftn`. Then absolute values are computed by using `numpy.abs(values)**2`. This spectrum is then shifted so that low frequencies are at the middle of the spectrum.

Because the spectrum is symmetrical, only half of the spectrum is used in the computations. The spectrum is divided into circular sectors (here we use 24 for the whole spectrum - so 12 on the half) and these sectors are then divided by circles from the center to the edge (here we use 32). In each of these areas mean value of the frequencies is computed and multiplied by the size of the sector. These values are stored in a matrix called Discrete Fourier Polar Coordinates Matrix (DFPCM).

The directionality uses sums of these columns. First maximum value of these sums is found by function `numpy.max()`. Then directionality is counted as:

$$D = \frac{\sum_{i=1}^{N}(f_{max} - f(i))}{N \times f_{max}}$$

where $f(i)$ is the sum of the ith column of the matrix and f_{max} is the max value of these sums and N is the number of columns.

8 Low, middle, high frequencies

Frequency values are computed from Power Spectral Density (PSD) defined as strength of the different features at different resolutions. The higher the value the more the variance of the features on a specific resolution and the higher the amplitude values of the frequency signal.

First Fourier transform is computed on given image luminance values, using function `numpy.fft.fftn()`. Then from complex values of the transform amplitudes are extracted using `numpy.abs(values)**2`.

Then these amplitudes need to be sorted into wave number bins representing the same frequency values. First the wave values corresponding to the amplitude values need to be found. Wave vectors are calculated by function `numpy.fft.fftfreq(size) * size` where size is the size of the image. These vectors are given to `numpy.meshgrid` to match the two-dimensional Fourier spectrum. Then we calculate their norm. Bin number is prepared by calculating `numpy.arange(0.5, size//2+1, 1.)` (maximum wave number will equal half the pixel size of the image - half of the Fourier frequencies can be mapped back to negative wave numbers that have the same norm).

Then binning is done by using `scipy.stats.binned_statistic()` with statistic set as `mean`. This calculates the mean values in each bin. Values of the bins are then multiplied by the space of each bin. That's the final PSD.

We extracted the information about the spectrum by dividing the PSD into *low*, *middle*, *high* frequencies. The borders for these frequencies were chosen artificially. We are working with images size 512x512 so the values were chosen as 0-8 for *low*, 8-64 for *middle*, 64-256 for *high*.

9 Mean chroma

Mean value of the *chroma* channel. Uses values weighted (multiplied) by the luminance values - the mean is calculated from these multiplied values. This accounts for visibility or brightness of the color information stored in the chroma channel. Uses arithmetic mean - sum of values divided by their number, which here is the size of the image using function `numpy.mean()`.

10 Pattern strength

Part of the computation is the same as *Directionality* computation (7). Uses the DFPCM matrix and its frequency spectrum for examining pattern behaviour.

Means of all the columns are calculated (describing a specific circular sector - direction - through all frequencies) using function `numpy.mean()`. Maximum of these values is found and ratios of the mean values to the maximum are computed as `max_mean / mean`. Then mean of these ratios is computed and describes how on average are the frequency values in a specific direction different from the maximum. That is, how visible is the pattern in the whole picture, how much the frequencies change in perspective to the maximum values.

11 Pattern number

Pattern number should correspond to the number of directions of the pattern. That is number 1 for linear pattern, number 2 for checkered pattern etc.

Part of the computation is the same as *Directionality* computation (7). Uses the DFPCM matrix and its frequency spectrum for examining pattern behaviour.

First sums of columns are computed using function `numpy.sum()`. We are interested in the peaks of these values, so they are scaled to (0, 1) interval for easier manipulation. Then function `findpeaks(method="topology")` from the `findpeaks` package is used (docs). It uses persistent homology for finding peaks in the data with lowering sea-level mechanic while also quantifying the "significance" of the peaks. Here peaks with score higher than 0.5 are selected. Score is calculated as a distance between levels of the data values, so the score value is proportional to the data size (0, 1).

Lastly we need to ensure the circular nature of the data is accounted for and if peaks on both ends of the spectrum are found, only one of them is selected (as normally they would be connected and only one of them would be a peak).

The found peaks signify the directions, where the frequency values peaked over their neighbourhood. In these directions the pattern is significantly visible. So we count those peaks and get the pattern number.

12 Colors number

Number of colors in the image.

It is difficult to differentiate colors the same way a human would because generally, colour is a spectrum. Setting the boundaries where human would regard values as two different colours is difficult. Here we take similar approach to finding significant pattern directions.

First we represent image in the LAB colour space and take values from channels A and B which represent the color spectrum values. These are combined to 2D histogram with 129 bins from -128 to 128 in both directions.

In this 2D histogram we find peaks using function `findpeaks` from package `findpeaks` for two-dimensional data (docs). Same function for one-dimensional data was used for finding *Pattern number* (11). Here the score of the peaks is proportional to the amount of values in the bin. Limit parameter for the score was chosen artificially based on experiments on peak values for few colorful materials as 30.

We count the number of colors as the number of peaks found.