

Exercise 3: Histograms

This exercise contains three main exercises. I encourage you to implement your own methods to increase your understanding of the algorithms. However, it is also possible to solve the exercises using OpenCV's implementation.

Note that all these exercises may take quite some time to complete. In general, if time is sparse you should at least write down a solution in pseudo code, such that you reflect on the methods presented at the lecture.

1) Load `legoHouse.jpg` and visualize a histogram over the hue channel. Explain what you see. (You can either implement your own function for visualizing histograms or use the provided code on `itsLearning`)

2) Implement histogram equalization and apply it to either a) the luminance channel, or b) a gray-scaled version of `TrinityCampanile3.jpg`

3) Write down pseudo code for implementing k-means clustering for data of dimension 1 (each datapoint is a single value).

Optional: Implement the k-means clustering method to cluster the hue channel values of `rainbow.jpg`. Estimate the correct number of classes by inspecting the histogram. Verify that you find the desired “peaks” in the histogram.

Optional: Segment the red spoons in the `BabyFood` image by means of histogram back-projection on the hue channel of the image

Things to look out for

1) Read the documentation to OpenCV

https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#aa5d20fc86d41d59e4d71ae93daee9726

Keep in mind that the size identifier used in the `at` operator cannot be chosen at random. It depends on the image from which you are trying to retrieve the data. The table below gives a better insight in this:

If matrix is of type `CV_8U` then use `Mat.at<uchar>(y,x)`.

If matrix is of type `CV_8S` then use `Mat.at<schar>(y,x)`.

If matrix is of type `CV_16U` then use `Mat.at<ushort>(y,x)`.

If matrix is of type `CV_16S` then use `Mat.at<short>(y,x)`.

If matrix is of type `CV_32S` then use `Mat.at<int>(y,x)`.

If matrix is of type `CV_32F` then use `Mat.at<float>(y,x)`.

If matrix is of type `CV_64F` then use `Mat.at<double>(y,x)`.

So if the image is `CV_16U` you have to use `at<ushort>`

Things to look out for

Check if uncertain about depth, number of channels or dimensions

Note that `histogram.size()` outputs (width,height) = (cols,rows)

```
void cv::calcHist (const Mat *images, int nimages, const int *channels, InputArray mask, OutputArray hist, int dims, const int *histSize, const float  
**ranges, bool uniform=true, bool accumulate=false)  
Calculates a histogram of a set of arrays. More...
```