Data Visualization and Analysis

BINF4245

Prof. Dr. Renato Pajarola

Exercise and Homework Completion Requirements

- Exercises are **mandatory** and they must be completed successfully to finish the course with a passing final grade.
- Exercises are graded coarsely into categories **pass** or **fail**.
  - *A **fail** is given to submissions which do not fulfill the submission rules or partial solutions, and no points are awarded.*
  - *A **pass** indicates that the exercise is sufficiently good to receive the corresponding points.*
  - *Late submissions (up to one day) will result in "-1" point.*
- The four exercises are allocated to the following point distribution: 2 – 3 – 5 – 5
  - A **minimum of 7 points** must be achieved to pass the module.
  - Thus, at least two exercises must be solved correctly, including at least one from the advanced ones.
  - *Failure to achieve this minimum will result in a failing grade for the entire module*
- We award **bonus points** for students who have collected more than 8 points from all the exercises.
  - Thus, **7 points** from the exercises is required, **8 points** is still normal passing, **9 and above** would give 1 or more extra bonus points.
  - Only the bonus points can and will be added directly to the final grade.
- Do not copy assignments, tools to detect copying and plagiarism will be used.
  - *The exercise results are an integral part of the final course grade and therefore the handed in attempts and solutions to the exercises **must be your personal work**.*

Submission Rules

- Submitted code must run without errors using the indicated Python environment, included libraries, packages and frameworks. If additional libraries/packages are needed, please specify these in a *readme.txt* file accompanying your submission.
- The whole project source code must be zipped and submitted before the given deadline, including exactly the files indicated on the bottom of this exercise sheet.
- Submit your .zip archive named *dva_ex2_Firstname_Lastname_MATRIKELNUMBER.zip* (e.g. dva_ex1_Hans_Muster_01234567.zip) through the OLAT course page.
- **Deadline is Monday, 09 November 2020 at 23:59h**

Exercise 2

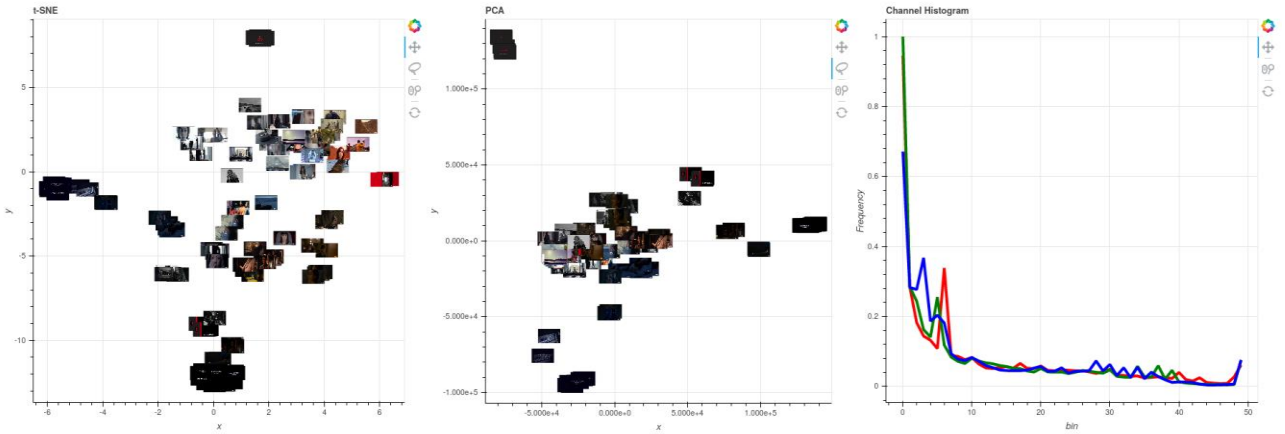
Dimensional reduction techniques are an important tool when working with datasets, which have more dimensions than we can convey using the coordinates, colors, shapes, or other visual cues. In such cases, a dimensional reduction technique may be applied to the high-dimensional dataset to project it into the low-dimensional visualization space (2D in our case since we have a two-dimensional coordinate system).

The aim of this exercise is to practice different dimensionality reduction processes and to learn how to plot them. Your task will be to read the color information from images, process and filter it and present the images based on their color content in a dashboard, using three different but connected plots. The arrangement of the widgets in your dashboard should look like the figures below. Exact tasks and hints are provided in the code skeleton.

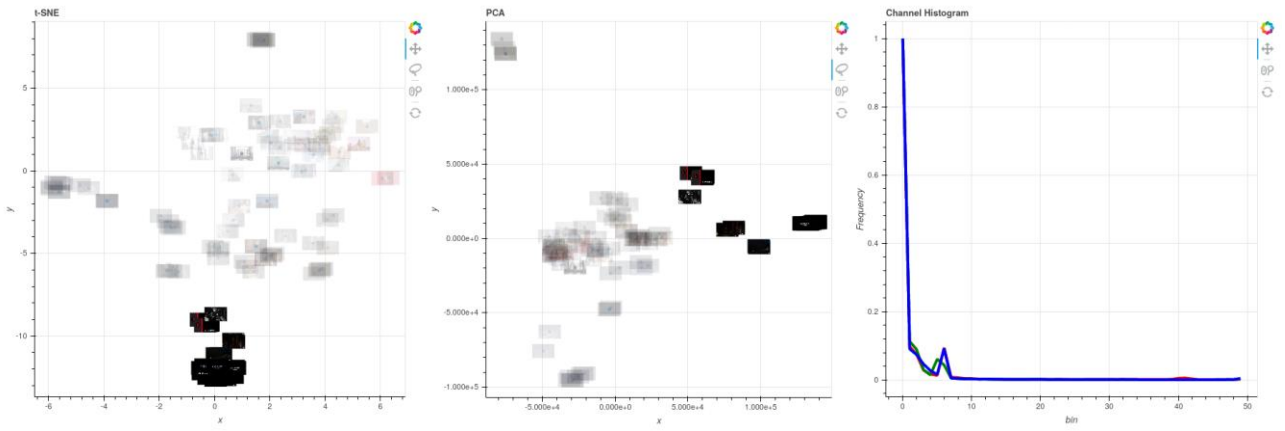
Important: To run the application, you will need to deploy it as a *directory format* application, such that the images can be loaded by the visualization. In essence you must make sure that the python script is named **main.py** and you run in the directory format:

```
# bokeh serve --show .
# python -m bokeh serve --show .
```

Otherwise the images will not appear in your visualization. More information about the directory format of the bokeh server can be found here: https://docs.bokeh.org/en/latest/docs/user_guide/server.html#directory-format



Screenshot 1: Full selection at startup



Screenshot 2: Partial selection using the lasso tool

Task 1: Preprocessing:

Your first task is to calculate two different histogram types for each image. First you need to compute a high-dimensional **color histogram**, which you will later input into the dimensional reduction method. Essentially, a color histogram is computed by splitting the complete RGB color space, which is a cube, into a number of smaller cubes (bins) and then counting for each bin, how many pixels of an image fall into it according to their color value. Luckily, NumPy provides such a multi-dimensional histogram functionality. A color histogram with 16 bins per edge would thus result in a high-dimensional feature vector with $16^3 = 4096$ bins. To place the histograms into the 2D visualization space, we will apply two different dimensional reduction techniques, namely t-SNE and PCA, for which we will use the scikit-learn library. The second histogram is a so-called **channel histogram**, sometimes also referred to as **image histogram**, which essentially computes the frequency per bin color channel individually (red, blue and green).

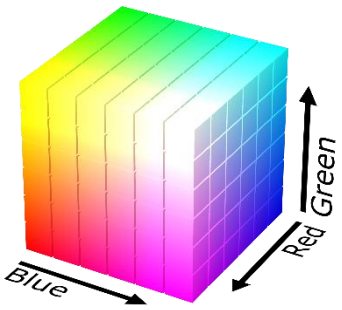


Figure 1: A color histogram splits the complete color space into a set of smaller cubes.

Task 2: Visualization

We want to plot the t-SNE and PCA results using the images as glyphs in the plots. The channel histograms should be aggregated over all images and plotted as three lines in a third figure. Additionally, a lasso tool is added to the t-SNE and PCA plots to select subsets of images. Each time a selection is made, an update function has to recalculate the aggregation of the channel histograms based on the currently selected image subset.

Important: All deliverables of this exercise must be submitted before the deadline. The absence of any required files will automatically lead to a **FAIL**.

Submission:

- clean version of your code file with proper comments (**.py** format. **No** .ipynb files!) that runs without any error. (Have a look at the jupyter_to_python example on OLAT)

- static folder that contains the images required to run your code

- readme.txt – Use this file for your comments or remarks (if necessary). If you used any additional libraries or packages that are not imported in the skeleton, explain why you added them and what you use them for. This file may be empty if you have no comments and only used the provided libraries.

- An export of the final dashboard in .pdf or .jpg format. (A screenshot is also accepted.)

- Put all required files into a .zip archive using the naming scheme detailed on the first page of this document. Put the files directly into the archive and do not package the root folder.

- Send late-submissions until the 10. November 2020 23:59 to thomas.huber2@uzh.ch with halter@ifi.uzh.ch in the CC.