****

May 27th

2021

November 7th

2019

Jakob Grøhn Damgaard, 201808996

Visual Analytics – Teacher: Ross Deans Kristensen-McLachlan.

Exam Portfolio for Visual Analytics

Table of Contents

[Introduction 2](#_Toc72191498)

[General Instructions 2](#_Toc72191499)

[A2 – Histogram Comparisons 4](#_Toc72191500)

[Description 4](#_Toc72191501)

[Usage 4](#_Toc72191502)

[Methods 4](#_Toc72191503)

[Discussion of Results 4](#_Toc72191504)

[A3 – Object Recognition using Edge Detection 4](#_Toc72191505)

[Description 4](#_Toc72191506)

[Usage 4](#_Toc72191507)

[Methods 4](#_Toc72191508)

[Discussion of Results 4](#_Toc72191509)

[A4 – MNIST Image Classification GUI 5](#_Toc72191510)

[Description 5](#_Toc72191511)

[Usage 5](#_Toc72191512)

[Methods 5](#_Toc72191513)

[Discussion of Results 5](#_Toc72191514)

[Critical Evaluation 5](#_Toc72191515)

[A5 – Classifying Paintings 5](#_Toc72191516)

[Description 5](#_Toc72191517)

[Usage 5](#_Toc72191518)

[Methods 5](#_Toc72191519)

[Discussion of Results 5](#_Toc72191520)

[Unbuilding LEGO Creations using Computer Vision 6](#_Toc72191521)

[Description 6](#_Toc72191522)

[Usage 6](#_Toc72191523)

[Methods 6](#_Toc72191524)

[Discussion of Results 6](#_Toc72191525)

[Critical Evaluation 6](#_Toc72191526)

[References 6](#_Toc72191527)

# Introduction

This document constitutes my collection of assignments from Spring 2021 course *Visual Analytics*, part of the bachelor's elective in Cultural Data Science. These assignments all revolve around conducting computational analysis of visual (image) data with varying purposes in Python (Van Rossum & Drake, 2009). Four assignments were assigned by the teacher throughout the semester while the last one comprises a self-assignment project. All source scripts for each assignment are coded according to principles of object-oriented programming. Hence, each script consists of a class that houses different functions and is initialized when the script is executed from a command line.

## General Instructions

Link to overall repository:

This section provides a detailed guide for locally downloading the code from GitHub, initialising a virtual Python environment, and installing the necessary requirements. In order to maximise user-friendliness and ease the processes mentioned above, all projects have been collated into one single GitHub repository. Therefore, all code can be fetched into one local folder and as there are no overlapping dependencies, only a single virtual environment is needed. Please note, a local installation of Python 3.8.5 or higher is necessary to run the scripts.

To locally download the code, please open a terminal window, redirect the directory to the desired location on your machine and clone the repository using the following command:

git clone <add repo>

Then, proceed to execute the Bash script provided in the repository for initialising a suitable virtual environment:

./create\_venv.sh

This command may take a few minutes to finalise since multiple packages and libraries must be collected and updated. When it has run, your folder should have the following structure (folder depth of 2):

.

├── A1-Basic-Image-Processing

│   ├── README.md

│   ├── W1-Basic-Image-Processing.py

│   ├── data

│   ├── new\_data.csv

│   └── org\_data.csv

├── A2-Histogram-Comparisons

│   ├── README.md

│   ├── W3-Histogram-Comparison.py

│   ├── chi\_sqr\_comparisons\_image\_0001.jpg.csv

│   └── data

├── A3-Edge-Detection

│   ├── README.md

│   ├── W5-Edge-Detection.py

│   ├── data

│   ├── image\_with\_ROI.png

│   ├── image\_with\_contours.png

│   └── only\_ROI.png

├── A4-Image-Classification

│   ├── GUI\_utils

│   ├── README.md

│   ├── digit\_classification\_GUI.py

│   ├── graphics

│   ├── model\_out

│   ├── models

│   └── test\_images

├── A5-CNN-on-Paintings

│   ├── README.md

│   ├── cnn\_artists.py

│   ├── data

│   ├── output

│   └── utils

├── LICENSE.md

├── README.md

├── Visual\_Analytics\_Exam.docx

├── creat\_venv\_unix.sh

└── requirements.txt

You can verify this structure by running the following command:

tree -I vis\_analytics\_venv -L 2

If everything checks out, you should be ready to execute the code scripts located in the respective assignment folders. Always remember to activate the virtual environment before executing scripts in the terminal command line:

source vis\_analytics\_venv/bin/activate

And the same goes for deactivating the environment when use is ceases:

deactivate

All code has been tested in Python 3.8.5 on a 2020 MacBook Pro 13’’, 2 GHz Quad-Core Intel Core i5, 16 GB Ram running macOS Big Sur (11.2.6). More detailed instructions regarding the execution of the individual scripts can be found in the sections on the respective assignments.

# A2 – Histogram Comparisons

Link to the assignment folder (embedded in the overall repository):

## Description

Chart, bubble chart

Description automatically generatedColours constitute key markers for the visual system when faced with tasks such as object recognition and memory consolidation (Wichmann and Sharpe, 2002). Furthermore, colours can be numerically represented in e.g., a 3D RGB colour space. Thus, by being both relevant neural features and computationally manageable, colours are an obvious feature to focus on when performing a simple analysis on visual data. As an example, image similarity can be analysed by calculating the differences in the colour composition between two different images. To conduct such an analysis, the various colour nuances present in an must be quantified using a 3D colour histogram. Such a histogram represents the distribution of colour present in an image.

Figure 1 - 3D visualisation of a 3D colour histogram produced by the notebook from class 3 (https://github.com/CDS-AU-DK/cds-visual/blob/main/notebooks/session3.ipynb)

For this assignment, we were provided with a data set consisting of images of 17 different common British flowers. The data set contains 80 images for each category. We were then asked to compare the 3D colour histogram of a self-chosen target image with each of the other images in the corpus one-by-on using chi square distance as a similarity measure. More specifically, the task was to produce a script that, for a given input image, outputs a single *.csv* file containing a column for the filenames of the compared images and a column with the corresponding distance scores. Lastly, the script should print out the filename of the image found to be most similar to the input target image.

## Usage

If not already open, open a terminal window and redirect to the home folder of the cloned repository (see *General Instruction*). Remember to activate the virtual environment. Then, jump into the folder called *A2-Histogram Comparisons* using the following command:

cd A2-Histogram-Comparisons

Now, it should be possible to run the following command in to get an understanding of how the script is executed and which arguments should be provided:

# Add -h to view how which arguments should be passed

python3 src/A2-Histogram-Comparison.py -h

usage: A2-Histogram-Comparison.py [-h] [--ti target\_image]

[INFO] Image similarity using color histograms

optional arguments:

-h, --help show this help message and exit

-ti --target\_image [DESCRIPTION] Name of the target image

[TYPE] str

[DEFAULT] image\_0001

[EXAMPLE] -ti image\_0001

It should now be clear that the script can be executed using the following command.

python3 src/A2-Histogram-Comparison.py -ti image\_0001

If there is no input to the *--ti* argument, *image\_0001.jpg* is used as the default target image. Feel free to choose your own target image. As the script is already specialised to this specific assignment and this specific dataset, I’ve simplified the command line argument as much as possible to increase user-friendliness. Therefore, please note that one only has to input the image name of the desired target image and no file path or *.jpg* suffix is needed.

### Structure

The structure of the assignment folder can be viewed using the following command:

tree -L 2

This should yield the following graph:

.

├── README.md

├── data

│   └── jpg

├── output

│   └── chi\_sqr\_comparisons\_image\_0001.csv

└── src

└── A3-Histogram-Comparison.py

The following table explains the directory structure in more detail:

|  |  |
| --- | --- |
| **Folder** | **Description** |
| data | A folder containing the data set used for the analysis. In this folder, the subfolder *jpg* holds 1382 *.jpg* files along with a *.txt* file listing all the filenames. |
| src | A folder containing the *.py* script (*A2-Histogram-Comparison.py*) created to solve the assignment. |
| output | A folder containing the output produced by the Python script. The script yields a *.csv* file with the file name *chi\_sqr\_comparisons****\_<image name>****.csv* |

## Methods

As stated, the script is coded using the principles of object-oriented programming. The main class of the script includes an \_\_init\_\_ method which holds the set of statements used for solving the desired tasks. This collection of statements is executed when the class object is created. Furthermore, the class holds a series of utility functions which are called when needed in the \_\_init\_\_. The class object is created - and thus, the tasks are performed -whenever the main function is executed. This happens every time the module is executed as a command to the Python interpreter.

The script takes the name of a target image as an input variable, reads in this image, and creates a 3D colour histogram with 8 bins in each dimension. It then obtains the paths for all the image file located in the *data* folder. Looping through all the images in the list paths one-by-one (and skipping a file if the path is identical to that of the target image), the script proceeds to generate a 3D colour histogram for each respective comparison image before calculating the chi square distance between it and the target image histogram. This value along with the given file name is then appended as a row to a Pandas (McKinney, 2010) data frame, which is exported as .*csv* file to the *output* folder when the loop has finished. Finally, the script prints the name and chi square value of the image with the highest similarity. Note that to make images directly comparable and to account for outliers, varying light intensity, image sizes etc., the histograms are normalized using min-max normalisation before similarity is calculated.

The main library utilised for this assignment is OpenCV (Bradski, 2000).

## Discussion of Results

The script runs swiftly and outputs a tidy data frame as expected. When running the script with *image\_0001* set as the target image, the most similar image is found to be *image\_0597* which has a chi square distance of 1242.

python src/A2-Histogram-Comparison.py -ti image\_0001

The most similar image is image\_0597.jpg

This image has a chi square distance of 1242

Figure 2 - When image\_0001 is set as the target image, image\_0597 is found to be the most similar image in the dataset when analysing 3D colour histograms



image\_0001

image\_0597

From this example run, it is apparent that the algorithm to a certain degree captures similarity between images; the target image and the image with the lowest chi square value both depict yellow flowers photographed with a mixed green/brown background. However, color histograms only capture the proportion of the number of colours in an image and do not account for the spatial location of the colors. The target image comprises several flowers and the flower visible in *image\_*0597 - though yellow - is clearly not of the same species. Hence, the method will often be insufficient to solve a more complex task like object recognition and serves better when used in combination with other visual analysis approaches.

# A3 – Object Recognition using Edge Detection

## Description

## Usage

## Methods

## Discussion of Results

# A4 – MNIST Image Classification GUI

## Description

## Usage

## Methods

## Discussion of Results

## Critical Evaluation

# Unbuilding LEGO Creations using Computer Vision

## Description

## Usage

## Methods

## Discussion of Results

## Critical Evaluation

TO DO:

Fixe kode:

* CNN paintings
* GUI!!!
* Friske resten op

Udfylde afsnit for alle pre-made assignments

Komme i gang med LEGO project

# References

Bradski, G. (2000). The OpenCV Library. Dr. Dobb’s Journal of Software Tools.

McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

Wichmann, F. A., Sharpe, L. T., & Gegenfurtner, K. R. (2002). The contributions of color to recognition memory for natural scenes. Journal of Experimental Psychology: Learning, Memory, and Cognition, 28(3), 509.

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace.