Building a simple image search algorithm

This project primarily includes two scripts: `histogram_image_search` and `knn_image_search`.

`histogram_image_search` leverages the OpenCV module to execute a straightforward image search based on normalized color histograms. The similarity or dissimilarity between images is measured using the Chi-Square metric.

`knn_image_search` uses a pre-trained model for feature extraction, and the similarity between images is calculated using the K-nearest neighbor algorithm, implemented through the scikit-learn library.

The project accepts an input image, which is compared against an image dataset. The output is a .csv file listing the top 5 images from the dataset that are most similar to the input image. Additionally, visualizations containing the selected target image and the top 5 most similar images are created through the matplotlib library.

In order to handle user input, the project employs the Tkinter module to create a simple GUI. This interface displays a pseudo-randomized sample from the image dataset, allowing users to select the target image for comparison.

Data

Dataset

The project's image search algorithm is based on the 17 Category Flower Dataset by the Visual Geometry Group at the University of Oxford. The dataset contains more than 1.000 images of flowers, divided amongst 17 distinct categories. The dataset should be placed in the `in` directory, and can be sourced here.

Model

By default, the project uses the VGG16 pre-trained model for feature extraction.

Project structure

```
└── simple_image_search
    ├── in
    │   ├── image_0001.jpg
    │   ├── ...
    │   └── image_1360.jpg
    │
    ├── out
    │   ├── histogram
    │   │   ├── csv
    │   │   └── plots
    │   └── knn
    │       ├── csv
    │       └── plots
    ├── src
    │   ├── histogram_image_search.py
```

```
│   ├── image_selection_gui.py
│   ├── knn_image_search.py
│   ├── utilities.py
│   └── visualize_results.py
│
├── README.md
├── requirements.txt
├── setup_unix.sh
└── setup_win.sh
```

Setup

Dependencies

Please ensure you have the following dependencies installed on your system:

- **Python**: `version 3.12.3`

Installation

1. Clone the repository

```
git clone https://github.com/apathriel/cds-vis-analytics
```

2. Navigate to the project directory

```
cd assignments
cd simple_image_search
```

3. Run the setup script to install dependencies, depending on OS.

```
bash setup_unix.sh
```

4. Activate the virtual environment (OS-specific)

```
source env/bin/activate
```

5. Run the main scripts from root project directory

```
python src/knn_image_search.py
python src/histogram_image_search.py
```

Usage

The image dataset should be placed in the 'in' directory. The image search algorithm should be run from the project root through the main scripts. The target is selected through the functions in `image_selection_gui.py`. Both scripts utilize functions from the `utilities.py` module. The `compare_images_in_dataset()` function takes a dataset input path, the image selected through the TKinter GUI, and an output path for the resulting .csv file.

Results

If you wish to compare the two scripts, you'd be tempted to examine the 'Distance' column of the CSV files. However, the `histogram` script utilizes Chi-Square distance, while the `knn` script utilizes Cosine similarity. Thus, while the project does not present any quantitative metrics on which to evaluate, the simple eye-test speaks to the efficacy of the results from the KNN script. Although not necessarily generalizable, but from personal experience, if you examine the two visualizations, you will find greater similarity between the flowers yielded by the `knn` script. Perhaps you could evaluate the algorithm performance through evaluating the categorical labels of the yielded flowers.

References

- VGG16 Model from Keras
- cv2 Histogram Comparison
- 17 Category Flower Dataset