QoS and Multimedia TP1: Image Manipulation with Python

This project is part of the QoS and Multimedia TP1, focusing on setting up tools and starting image manipulation using Python.

Table of Contents

Project Description

The goal of this project is to provide tools and resources for working with multimedia data, specifically images, using Python. You will learn how to set up the necessary tools, manipulate images, and explore various multimedia-related tasks.

Installation

Provide instructions on how to install and set up your project, including any prerequisites.

Windows Installation

1. Download Python.
2. Run the installer and follow the prompts. Be sure to check "Add Python x.x to PATH."

Linux Installation

1. Check if Python is already installed by running `python --version`. If not, use your distribution's package manager to install Python.

Setting Up a Virtual Environment

1. Install `virtualenv` (if not already installed) using `pip install virtualenv`.
2. Create a virtual environment using `virtualenv venv` (replace 'venv' with your desired environment name).
3. Activate the virtual environment:
   - **Windows Command Prompt:**
     `venv\Scripts\activate`

- **Windows PowerShell:**
  `.\venv\Scripts\Activate`
- **Linux:**
  `source venv/bin/activate`

Installing Dependencies

1. Install NumPy using `pip install numpy`.
2. Install OpenCV using `pip install opencv-python`.

Usage

Explain how to use your project, including code examples and screenshots if applicable.

Reading an Image

```
import cv2

# Read an image using OpenCV
image = cv2.imread('input_image.jpg')

# Check if the image was successfully loaded
if image is None:
    print("Error: Unable to read the input image.")
else:
    # Display the original image
    cv2.imshow('Original Image', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Creating a Random Image

```
import numpy as np
import cv2
import random

# Create a random image using NumPy
height, width, channels = 500, 500, 3  # Adjust these values as needed
random_image = np.random.randint(0, 256, (height, width, channels), dtype=np.uint8)

# Display the random image
cv2.imshow('Random Image', random_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
# Save the random image
cv2.imwrite('random_image.jpg', random_image)
print("Random image saved as 'random_image.jpg'")
```

Using Jupyter Notebook

Jupyter Notebook is an interactive computing environment that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is a powerful tool for data analysis, scientific computing, and running code interactively.

Installing Jupyter Notebook

To use Jupyter Notebook with your project, you first need to install it. If you haven't already installed it in your virtual environment, you can do so with pip:

```
pip install jupyter
```

Launching Jupyter Notebook

1. Make sure your virtual environment is activated.
2. Open a terminal or command prompt.
3. Navigate to your project directory.
4. Start Jupyter Notebook by running the following command:
   `jupyter notebook`
   This will open a new tab in your web browser with the Jupyter Notebook interface.

Creating and Running Notebooks

With Jupyter Notebook, you can create interactive notebooks with code cells, markdown cells, and more. Here's how you can create and run a new notebook:

In the Jupyter Notebook interface, click the "New" button and select "Python 3" to create a new Python notebook.

You can add code cells and markdown cells to your notebook. Code cells are where you can write and execute Python code, while markdown cells are for documentation and explanations.

To run a code cell, select it and press Shift + Enter. The output will be displayed below the cell.

Benefits of Using Jupyter Notebook

Interactive Development: Jupyter Notebook allows you to run code cells interactively, making it ideal for experimentation and data analysis.

Documentation: You can include detailed documentation, explanations, and visualizations alongside your code, making it easier to understand and share your work.

Data Visualization: Jupyter Notebook supports various data visualization libraries, making it convenient for creating plots and charts.

Reproducibility: Notebooks capture the entire workflow, making it easy to reproduce and share your analysis with others.

Collaboration: You can share your Jupyter notebooks with colleagues or collaborators, enabling collaborative coding and analysis.

Feel free to create Jupyter notebooks within your project directory to document and run code related to your project. It can be a valuable tool for showcasing and sharing your work.

Dependencies

List the dependencies and versions required for your project.

. Python x.x . NumPy x.x . OpenCV x.x . Jupyter Notebook (optional)