Project 2: Content-based Image Retrieval

Team Members

- **Sangeeth Deleep Menon** | NUID: 002524579 | MSCS - Boston | CS5330 Section 03 (CRN: 40669, Online)
- **Raj Gupta** | NUID: 002068701 | MSCS - Boston | CS5330 Section 01 (CRN: 38745, Online)

Project Description

This project implements a content-based image retrieval (CBIR) system that, given a target image, finds the most visually similar images from a database. The system supports multiple feature extraction methods (baseline pixel matching, color histograms, multi-region histograms, texture+color, deep network embeddings) and distance metrics (SSD, histogram intersection, cosine distance). A custom retrieval method combining HSV color analysis with texture and spatial features is also implemented.

Building the Project

This project uses CMake and requires OpenCV, GLFW, and a C++20 compatible compiler.

1. **Navigate to the project directory.**
2. **Create a build directory and run CMake and make:**

   ```
   mkdir -p cmake-build-debug && cd cmake-build-debug
   cmake ..
   make
   ```

   This will create two executables inside the `cmake-build-debug` directory: `Project2` (command-line) and `Project2_GUI` (interactive GUI).

Running the Applications

Command-Line Application

The executable takes the following arguments:

```
./cmake-build-debug/Project2 <task> <target_image> <num_results>
```

Example:

```
./cmake-build-debug/Project2 baseline olympus/pic.1016.jpg 4
```

GUI Application

The GUI provides a more user-friendly way to use the system.

1. **Navigate to the project's root directory.**
2. **Execute the command:**

   ```
   ./cmake-build-debug/Project2_GUI
   ```

3. **How to Use:**
   - Click **"Browse..."** to open a file dialog and select a target image.
   - The selected image will be displayed in the window.
   - Select the desired **"Task"** and **"Number of Results"** from the dropdown menus.
   - Click **"Execute Search"** to run the query.
   - Results will be displayed in a grid below.
   - Click **"Close"** to exit.

Executable Files

This project generates three main executable files, each with a specific role:

1. **`Project2` (Command-Line Interface - CLI)**
   - **Purpose**: This is the basic application run directly from the terminal. It takes all search parameters as command-line arguments and prints results to the console.
2. **`Project2_GUI` (Graphical User Interface - GUI)**
   - **Purpose**: This is the interactive application with a graphical window. It provides a user-friendly way to select images, choose tasks, and view results visually.
3. **`generate_embeddings` (Utility Application)**
   - **Purpose**: This is a standalone tool designed to create feature data. It processes images through a Deep Neural Network (DNN) model and saves the resulting feature vectors (embeddings) into a CSV file. This utility is typically run once to generate data for the main applications.

Project Report

A detailed report on the project, including results and analysis, can be found in REPORT.md.

Methods Overview

Task	Feature Vector	Distance Metric
1. Baseline	7x7 center pixel patch	Sum of Squared Differences (SSD)
2. Histogram	rg chromaticity histogram (16x16 bins)	Histogram Intersection
3. Multi-histogram	Top/bottom half RGB histograms (8 bins each)	Weighted avg of histogram intersection
4. Texture-Color	Whole-image RGB histogram + Sobel magnitude histogram (8 bins)	50/50 weighted histogram intersection
5. DNN Embeddings	512-dim ResNet18 features from CSV	Cosine distance (or SSD)
6. Custom DNN	512-dim ResNet18 features from custom-generated CSV	Cosine distance (or SSD)
7. Custom	HSV histograms (whole, top 1/3, bottom 1/3) + Sobel texture + edge density	Weighted combination (30/20/20/15/15)

Extensions

1. An interactive GUI was developed for the project using the ImGui library. This GUI provides a more user-friendly way to interact with the CBIR system, allowing users to:
    - Browse and select a target image using a native file dialog.
    - Choose the matching algorithm and number of results from dropdown menus.
    - View the target image and the returned matches visually in the same window.
2. **Blue Trash Can Finder:** This method ranks images by the percentage of a specific shade of blue, designed to find the blue trash cans present in the dataset.
3. **Banana Finder:** A simple color-based detector that ranks images by the percentage of yellow pixels they contain. This is effective for finding images with prominent yellow objects.
4. **Face Detector:** It ranks images based on the number of faces detected compared to the target image.

5. **Gabor Filter (Texture):** This method uses a bank of Gabor filters with different orientations and frequencies to create a feature vector describing the texture of an image. It is effective for retrieving images with similar textural patterns.
6. **Custom DNN (ResNet18):** This extension uses a utility program (`generate_embeddings`) to process the entire image database through a local ONNX ResNet18 model, creating a new `Custom_ResNet18_olym.csv` file. The GUI can then use this custom-generated set of embeddings for matching, allowing for direct comparison with other pre-computed feature sets.

Submission Time

Within Late deadline. We have worked very hard and implemented every extension. Kindly grade it. Raj is registered with DAS and has DAS accommodations. He is having a very tough time, yet he contributed equally, whenever he could.

Videos

None.

Acknowledgements

- OpenCV documentation for histogram and Sobel filter references
- Course materials and sample code provided by Prof. Bruce Maxwell
- Shapiro and Stockman, Chapter 8 for histogram matching concepts
- An AI assistant (Gemini) was used to help write and debug code, and for project documentation.