Initial Project Description and Work Plan

Student Number: 40261335

Student Name: Adam Cregan

Supervisor: Paul Miller

Table of Contents

1.0 Problem Description

In this project I plan to create an image search engine that will take an image from a user's input and compare its characteristics against a gallery of locally stored images to determine which are the most similar. The images which are deemed to be most similar will be displayed to the user. From here the user can close the application or they can choose to clear the data and choose another image to search. A possible user interface is shown below in Fig. 1. This search function should use distance measurements to calculate which images are the most similar based on the colour and feature extraction. The user should only be allowed to select images and there should be more error handling that should ensure that the user has selected an image when they press the search button, and the application should inform the user there are no images in the database if the system registers there are no images.

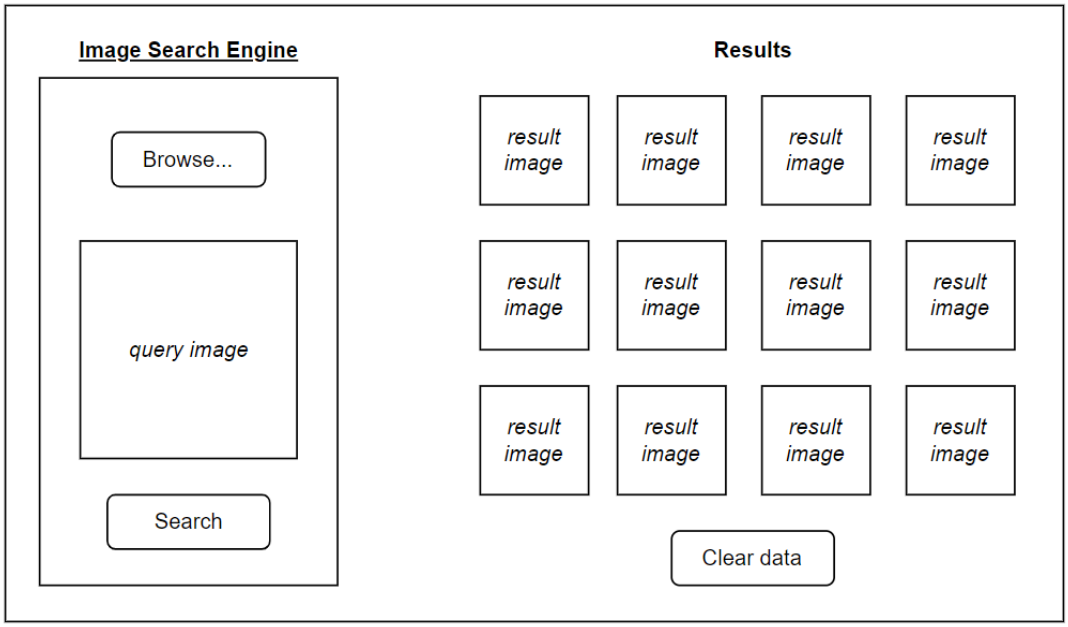

Fig. 1 *Possible GUI*

2.0 Software

Eclipse:

I plan to use the latest version of Eclipse (Eclipse IDE for Java Developers - v2022-12(4.26.0)) to develop this application. I have chosen to program in Java in Eclipse because and I have prior experience using it and I can achieve all that I need to for this project only using this tool. I also plan on using the OpenCV library to some degree during initial testing for this application as there are functions which will help the development of this application i.e. matrices and histogram.

3.0 Marking Criteria/ Requirements (end-user)

- Allows user to select a query image and show it in GUI
- Query image can exist inside/outside image database. Selected image will be resized to 512x512 (like all the images in the database)
- Apply search functions to query image when requested
- Iterate through images in gallery and apply functions to each image
- Returns multiple images from gallery that are similar based on distance measurement (top 10 images that are closest)(perhaps less if some are closest but still not close)
- Create GUI with buttons for image selection, search, and clear data
- The application should return all the best images in less than 15 seconds

- Implement effective distance measurement for colour, texture, and shape
    - Compare results for accuracy for histogram correlation, chi-square, intersection, and Bhattacharyya and implement the best method(s)
    - Implement the best performing function(s) manually without the use of OpenCV functions
    - Test using thresholding and edge detection for feature extraction
    - Test using the Local Binary Patterns (LBP) descriptor, to extract texture features

- Implement Precision-Recall curves to analyse performance
- Improve scalability by calculating features offline for image database

- Implement error handling (pop-up message) for if:
    - The user clicks the “Search” button when no query image has been selected
    - There are no images in the database when the user tries to search
    - The user selects an invalid file type i.e. not jpg

4.0 Gantt Chart

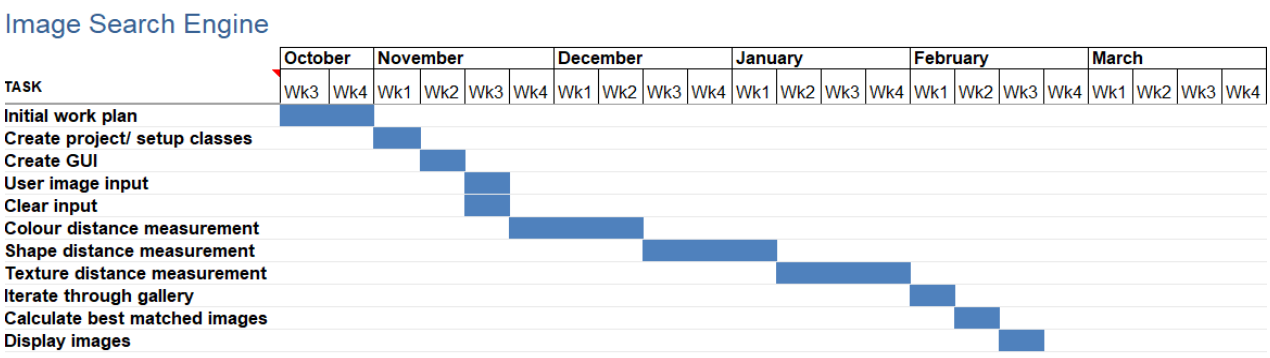

Fig. 2 *Gantt Chart*

5.0 Goals

Colour:

- colour histogram (*primary)*
- using histogram correlation, chi-square, intersection, and Bhattacharyya
- segment colour proportion by region and by spatial relationship among several colour regions (*secondary*) *optional addition, time/ result dependent*

Shape: Segmentation/thresholding or edge detection (compare results using Euclidean distance and/or chi-squared distance).

Texture: Two-dimensional grey level variation, using Local Binary Patterns (LBP) (compare results using Euclidean distance and/or chi-squared distance).

For testing I will split the image database into 20% test images and 80% image gallery (database). This means that the query image will not appear in the results and the performance analysis will be more accurate.

6.0 Basic System

- Returns best matched images based on a single method of either correlation, chi-square, intersection, and Bhattacharyya. This will be hardcoded in the backend but can be easily changed at any time. The method chosen for this system will be determined by some accuracy tests to determine which measurement is best.

6.1 Intermediate System

- Implement the best performing model from the basic system without the use of the OpenCV functions to demonstrate understanding. In addition, precision-recall curves should be generated to analyse performance

6.2 Advanced System

- Improve runtime and scalability of intermediate system by storing the features of the images in a database. Thresholding and/or edge detection, and Local Binary Patterns (LBP) should also be implemented with the initial methods to best enhance the performance of the final model

7.0 GitHub

Below is a link to the git repository which I use to store all my Java code and all of the report and testing documentation:

https://github.com/acregan04/ImageSearchEngine.git