Intelligent Photo Gallery

Final Report

Kwan Hoi Shun (3035280972)

Supervised by Dr. Anthony Tam

April 2018
Abstract

People nowadays take a lot of photos and are in need of a tool for organizing their own collection of photos. Existing photo gallery application such as Google Photos uses face recognition and cloud storage to improve the photo management experience. However using cloud storage raises privacy concerns and performing face recognition on the local machine can be slow. This project aims to build a photo gallery application to provide an alternative to the existing solutions and allow users to manage the photos conveniently.

Since face recognition is a popular feature amongst modern gallery applications, we also performed different experiments on different approaches on the face recognition algorithms to find the most suitable face recognition solution to be implemented. This paper discusses the experiment results and the implementation of the facial recognition function along with the implementation of other photo management features. Moreover, this paper also mentioned the constraints faced in the development stages and ends with the future development of the project.
Acknowledgment

We would like to thank our supervisor, Dr. Anthony Tam for providing practical advices, valuable insight and patiently discussing the details of the project. We would also like to thank all of the teachers that have taught me in the previous years for equipping us for with adequate knowledge in algorithms and programming that allowed us to be able to develop the Intelligent Photo Gallery.
Table of Contents

List of Figures ....................................................................................................................... 5
List of Tables .......................................................................................................................... 6

Chapter 1.0 Introduction ......................................................................................................... 7
  1.1 Background ...................................................................................................................... 7
  1.2 Project Objective ............................................................................................................ 7
  1.3 Project Scope .................................................................................................................. 7
  1.4 Feature Priority List ....................................................................................................... 9
  1.5 Project Deliverable ......................................................................................................... 9

Chapter 2.0 Literature Review ............................................................................................... 10
  1.2 Existing Solutions ........................................................................................................ 10
     1.2.1 Google Photos ..................................................................................................... 10
     1.2.2 Facebook ............................................................................................................ 10

Chapter 3.0 Application Design .......................................................................................... 11
  2.1 Development Tools and Technologies ......................................................................... 11
      2.1.1 Application Language: Python ........................................................................... 11
      2.1.2 GUI Framework: Kivy ....................................................................................... 11
      2.1.3 Database Software ............................................................................................. 11
      2.1.4 Other Python Packages ...................................................................................... 11
  2.2 Database Design .......................................................................................................... 12
      2.2.1 Indexes for efficient searching ........................................................................... 13
      2.2.2 ‘update_count’ function ................................................................................ 13

Chapter 4.0 Face Recognition Methodology ........................................................................ 14
  4.1 Face Detection Stage ..................................................................................................... 14
  4.2 Feature Extraction Stage ............................................................................................. 14
      4.2.1 CNN Approach .................................................................................................. 15
      4.2.2 Facial Landmark Approach .............................................................................. 15
  4.3 Classification Stage ....................................................................................................... 16
      4.3.1 Support Vector Machine (SVM) ......................................................................... 16
      4.3.2 SVM with Probability ....................................................................................... 17
      4.3.3 K Nearest Neighbors (KNN) ............................................................................ 17
      4.3.4 Radius Neighbors Classifier ............................................................................ 17
      4.3.5 One-class SVM ................................................................................................ 17

Chapter 5.0 Face Recognition Experiment Settings ............................................................ 18
  5.1 Experiments with Feature Extraction Algorithms ....................................................... 18
      5.1.1 Experiment for the efficiency of the feature extraction algorithm ..................... 18
  5.2 Experiments with Classification Algorithms ............................................................. 19
      5.2.1 Experiment with Classification Algorithms ...................................................... 20
      5.2.2 Experiment with one-class SVM for Outlier Detection .................................... 20
      5.2.3 Experiment with the Modified Algorithms for Outlier Detection ..................... 21
      5.2.4 Measuring Metrics ............................................................................................ 22

Chapter 6.0 Face Recognition Experiment Results .............................................................. 23
  6.1 Result for the Feature Extraction Algorithm .............................................................. 23
      6.1.1 Efficiency of the algorithms ............................................................................. 23
      6.1.2 Accuracy of the algorithms .............................................................................. 23
Chapter 7.0 Project Methodology ................................................................................. 32
  7.1 Implementation of Import Photos ........................................................................ 32
  7.2 Implementation of Face Recognition .................................................................... 32
    7.2.1 Face recognition at Photo Import ................................................................. 33
    7.2.2 Labeling of Faces ......................................................................................... 34
    7.2.3 Initialization of the Face Recognition Models ............................................... 35
  7.3 Implementation of Import Filters ......................................................................... 36
  7.4 Implementation of Duplicate Photos Search ....................................................... 37

Chapter 8.0 Results ........................................................................................................ 39
  8.1 Basic Functions .................................................................................................... 39
    8.2.2 Search Image ............................................................................................... 39
    8.1.3 View and modify image ............................................................................... 40
  8.2 Face Recognition ................................................................................................ 41
    8.2.1 Unidentified Faces Screen ......................................................................... 41
    8.2.2 Similar Face Suggestion ............................................................................. 41
  8.3 Advanced Functions ............................................................................................. 42
    8.3.1 Import Filters and Apply Function ............................................................... 42
    8.3.2 Duplicate Photo Search ............................................................................... 42
    8.3.3 Configuring Storage location ...................................................................... 42
  8.4 Overall functionality ............................................................................................ 43

Chapter 9.0 Limitations ................................................................................................. 44
  9.1 Using pre-trained models for the CNN feature extraction approach .................... 44
  9.2 Python only supporting local storage ................................................................. 44

Chapter 10.0 Future Works ......................................................................................... 45
  10.1 Implementing Similar Photo Search ................................................................. 45
  10.2 Improving the loading time of the application .................................................. 45
  10.3 Split the storage of cropped facial image into multiple directory ...................... 45
  10.4 Implementing the sort function for screens with search bars ............................. 45
  10.5 Addition of Help Screen .................................................................................... 46

Chapter 11.0 Conclusion .............................................................................................. 47

References ................................................................................................................... 48

List of Figures

Figure 1: Class diagram of the database ...................................................................... 12
Figure 2: The stages of the face recognition process ................................................... 14
Figure 3: Illustration of the 68 facial landmark extracted by the landmark estimation algorithm ................................................................. 16
Figure 4: Sample faces in the LBW dataset ............................................................... 18
Figure 5: Example of face pair belonging to different person ........................................... 19
Figure 6: Example of face pair belonging to the same person ........................................... 19
Figure 7: Coarse grid search on distance threshold for the CNN approach ............................ 24
Figure 8: Fine grid search on distance threshold for the CNN approach .............................. 24
Figure 9: Sample images from the LBW dataset that contains more than 1 face ....................... 24
Figure 10: Coarse grid search on distance threshold for the facial landmark approach ............... 25
Figure 11: Fine grid search on distance threshold for the facial landmark approach ................. 25
Figure 12: Accuracy of different distance threshold for the facial landmark approach with only the original four ratios .................................................................................. 25
Figure 13: Accuracy of KNN with different k values ............................................................. 27
Figure 14: Accuracy scores for different ru values for one-class SVM (linear) ......................... 28
Figure 15: Accuracy scores for different ru values for one-class SVM (poly) .......................... 28
Figure 16: Accuracy scores for different ru values for one-class SVM (rbf) .......................... 28
Figure 17: Accuracy scores for different probability threshold in SVM (linear) ....................... 29
Figure 18: Accuracy scores for different probability threshold in SVM (poly) ......................... 29
Figure 19: Accuracy scores for different probability threshold in SVM (RBF) ......................... 29
Figure 20: Accuracy of the radius neighbour classifier with different radius values ............... 29
Figure 21: Flow of import photo .......................................................................................... 32
Figure 22: Face recognition at import photos ........................................................................ 33
Figure 23: Photo import screen ......................................................................................... 39
Figure 24: Search Screen ..................................................................................................... 40
Figure 25: Detailed image screen ......................................................................................... 40
Figure 26: Unidentified faces screen .................................................................................... 41
Figure 27: Similar face suggestion popup ............................................................................ 41
Figure 28: Import filters view .............................................................................................. 42

List of Tables

Table 1: he priority of each of the feature. The priority are ranked in the scale of 1-5, with 1 being the most important and 5 being least important. Features with priority larger or equal to 3 are considered core features for the application and features of lower priority than 3 are considered as tentative features ........................................................................................... 9
Table 2: List of features used in the facial landmark approach the first 4 features highlighted in light blue are features from the original approach ................................................................... 15
Table 3: Number of photos for the each of the selected subjects ........................................... 20
Table 4: Processing time of the feature extraction algorithms ................................................ 23
Table 5: Result of the feature extraction approaches, the better approach in each of measuring metric is indicated with bold text ................................................................................. 26
Table 6: Summary of the performance of the classification algorithms, the best model in each metric is indicated in bold ......................................................................................... 27
Table 7: Outlier score and inlier score of the best model of one-class SVM of different kernels ......................................................................................................................... 28
Table 8: Result of SVM with probability and radius neighbour using 50% test set and 10 known persons .................................................................................................................... 30
Table 9: Result of SVM with probability and radius neighbour using 20% test set and 10 known persons .................................................................................................................... 30
Table 10: Result of SVM with probability and radius neighbour using 50% test set and 2 known persons .................................................................................................................... 30
Table 11: Description of attributes of the 'FilterRule' object ................................................ 36
Chapter 1.0 Introduction

1.1 Background
Nowadays, the widespread of the use of smartphones allowed people to take a lot of photos to record memorable moments. Many people have accumulated a large collection of photo without proper organization on their computer. Using a good photo gallery application can allow them to organize their built up photo collection and quickly locate photos in their collection through the use of tags, albums and convenient photo management functionalities.

Currently there are some existing gallery application that can offer a lot of convenient photo management tools for organizing the photo collections. Face recognition is one of the common feature in existing application that provide additional information to the photos for people to locate them faster. However many of the existing solution uses cloud storage for storing the photos which raises privacy concerns, there are also limitation on the photo quality and the quantity of photos in free existing solutions. This projects is hence to provide an alternative photo gallery application that have no limitation on the photo quantity and photo quality and provide a quick and accurate face recognition feature in the photo gallery application.

1.2 Project Objective
This project aims to build a photo gallery application that protect users privacy and provide convenient photo management functionalities with no limitation on the quantity and quality of photos.

In addition to the development of a user friendly and convenient photo gallery application, this project also aims to find an efficient and accurate face recognition algorithm that will be suitable for a photo gallery application for implementing into the application.

1.3 Project Scope
This project will build a desktop photo gallery application that aims to enhance user’s photo management experience. The application will include basic photo gallery functionalities (i.e. import photos, search, tagging), more advanced photo management features(i.e. duplicate photo search, automatic photo filters) and an efficient face recognition model.
➢ **Face recognition for recognizing faces**
The application will have an efficient face recognition algorithm that can identify faces in photos accurately. The face recognition process will be carried out during the importation of the photos. Users will need to label unidentified photos and the system will provide similar face suggestion to the user for the user to validate the faces. The labeled faces will then be used for the training of the classification algorithm in the face recognition process.

➢ **Import photos and search photos**
The application will allow users to import photo and have a quick search functions that can help users to locate their photos quickly.

➢ **Duplicate search for searching duplicate photos**
A duplicate search function will be implemented into the application, the search will be automatically carried out on importation of the photos to help users to identify duplicated photos automatically.

➢ **Add and remove tags and albums to photos**
User will be able to use tags and albums to organize photos in groups. Tags and albums are used to organized photo into different groups and users can use tags and albums for searching. A key difference between tags and albums is that photos can have several different tags, while each photo can only be associated with a single album, hence users should use albums to group photos by the most important characteristic (i.e. important events) while tags should be used to provide less significant information (i.e. places, objects).

➢ **Apply actions to selected photos**
Users will be able to apply the same action(e.g. adding tags) to a selection of photos base on some criteria.

➢ **Import filters for labeling newly imported photos**
The import filters are very similar to email filters. Users will be able to specify conditions and the target album and tag such that all newly imported photos with matching conditions will be labeled with the specified album and tag.
Selecting photo storage location
Users will be able to configure the storage location of the photos.

User friendly Graphical User Interface (GUI)
The GUI of the application will be intuitive and users will be able to locate and use the functions of the application with ease.

1.4 Feature Priority List
Since there are quite a lot of features to be implemented in this project, a priority was assigned to each of the main features to indicate its importance (see Table 1). The project implemented the features in decreasing order of priority during the development process to ensure that all important features are implemented in the final deliverable.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add and remove tags and albums in photos</td>
<td>1</td>
</tr>
<tr>
<td>Apply same action to multiple selected photos</td>
<td>1</td>
</tr>
<tr>
<td>Auto tagging using events in Google Calendar</td>
<td>5</td>
</tr>
<tr>
<td>Import filters</td>
<td>2</td>
</tr>
<tr>
<td>Create, add and remove photos from albums</td>
<td>1</td>
</tr>
<tr>
<td>Face recognition</td>
<td>1</td>
</tr>
<tr>
<td>Face recognition with age progression</td>
<td>4</td>
</tr>
<tr>
<td>Import photos to the gallery</td>
<td>1</td>
</tr>
<tr>
<td>Search for duplicate photos</td>
<td>2</td>
</tr>
<tr>
<td>Searching with multiple criteria</td>
<td>1</td>
</tr>
<tr>
<td>Storing in local storage and NAS storage</td>
<td>3</td>
</tr>
<tr>
<td>Storing in NAS storage with personal cloud configuration</td>
<td>5</td>
</tr>
</tbody>
</table>

*Table 1: the priority of each of the feature. The priority are ranked in the scale of 1-5, with 1 being the most important and 5 being least important. Features with priority larger or equal to 3 are considered core features for the application and features of lower priority than 3 are considered as tentative features.*

1.5 Project Deliverable
This project develops a desktop gallery application that includes all the features of priority 3 and higher. The application is compatible on both Windows and Mac. It also deliver a report on the implementation of the desktop gallery application and the result of experiments on different face recognition approach.
Chapter 2.0 Literature Review

1.2 Existing Solutions

Currently, there are many applications that can help users store photos and organize them, this includes social network applications such as Facebook and Instagram that also allow users to share the photos with others. There are also online photo gallery applications like Google Photo that allow users to upload photos on a cloud storage.

1.2.1 Google Photos

Google Photos is a popular photo gallery application that stores user photos on a cloud storage and allows users to access their photos easily, it also has a quick face recognition feature that allows users to find photos using persons in the photos. However, users have to face a choice between limited storage space with uncompressed photos and unlimited storage with reduced photo quality if they choose to use the application for free. Moreover, by storing photos in a cloud storage, this means that the access of the photos is limited by the network connection speed. The users also lose total control on the photos once the photos are uploaded to the cloud storage, deleting of the photo in your collection may not mean for a deletion of the photo in the cloud server.

1.2.2 Facebook

Another popular choice for people to store their photos is Facebook. Facebook also provides a face recognition feature for identifying faces in the uploaded images. However, Facebook uses compression to store the user photos, reducing the quality of the stored photos. Facebook is also known to data mine user information, therefore it raises data privacy concerns whether it is safe to store photos on Facebook where Facebook may make use of the uploaded photos to analyze the user activity for suggesting advertisements.
Chapter 3.0 Application Design

In this chapter, we will introduce about the choice of development tools and technologies used in the development of this application, afterwards we will discuss about the design of the database of the application.

2.1 Development Tools and Technologies

2.1.1 Application Language: Python
- Python is chosen as the application language as the development team is more familiarized with Python. It is also chosen as it have a large collection of packages for dealing with different problems especially in deep learning, as this project will use machine learning algorithms for face recognition.

2.1.2 GUI Framework: Kivy
- Kivy is used as it is a popular GUI framework and is well documented. It is a free GUI framework and the GUI framework is more modern than the Python inbuilt GUI framework tkinter. Kivy is also a cross-platform GUI framework that gives a universal GUI appearance for both Windows and Mac, hence can save development time for tweaking the GUI twice for both platform.

2.1.3 Database Software
- Sqlite3 is chosen over MySQL as the database software as the intelligent photo gallery is a desktop application without any network server. Hence sqlite3 will be sufficient as the database and is simpler to set up than MySQL. Peewee is used as the ORM(Object-relational mapping) for this application, it allows a more orientated approach for handling the database objects through the use of class functions, which helps reduce the amount the code used to access and modify the database and also helps to better encapsulate the database objects.

2.1.4 Other Python Packages
We have also used several other python packages for implementing different parts of the project. This includes openCV for image operations, pickle for serializing lists for storage in the database, Dlib for face recognition algorithms and sklearn for classification algorithms.
2.2 Database Design

Below is a class diagram of the database design of this application (see Figure 1), illustrating what kinds of data in each of the classes.

The ‘Photos’ class is the most associated class in the database as the a photo is the main theme of a photo gallery application, each photo can be associated with at most 1 album and any number of tags and faces.

Each ‘Faces’ object can be associated with at most 1 ‘Persons’. The ‘feature’ attribute in the class is used to store a serialized list of 128 features describing the photos extracted by the face recognition algorithm. To retrieve the features the system will need to deserialize the content stored in ‘feature’. By storing the feature list in the blob field other than storing the feature list in a separate csv file, it reduces the overhead in reading and storing the feature list in a separate file.
In the ‘FilterRules’ class, the priority attribute is used to store the running priority of the filter rules, the attributes starting with ‘c_’ are attributes indicating the matching condition and the attributes starting with ‘a_’ are used to indicate the action to be taken.

2.2.1 Indexes for efficient searching
The ‘name’ field of the ‘Albums’, ‘Tags’ and ‘Persons’ object is specified with a unique index constraint. The constraint ensures that there will not be any object belonging to the classes to have another object to have the same name in the same class. The unique index also increases the efficiency of the searching operation, as the ‘name’ field of the ‘Albums’, ‘Tags’ and ‘Persons’ objects are frequently used in the search function, photo import function and searching the Tag screen, Album screen and Person screen.

The ‘hash’ field of the ‘Photos’ object is also specified to create an index to facilitate faster duplicate photo search.

2.2.2 ‘update_count’ function
The ‘Albums’, ‘Tags’ and ‘Persons’ object all have a class function of ‘update_count’, which updates the attribute of ‘photo_count’ of the object. the function is created to perform automatic cleanup of unused ‘Tags’ and ‘Persons’ objects, when the ‘photo_count’ is updated to 0 in the ‘update_count’ function in the 2 classes of objects, the object will then delete itself automatically, to free spaces in the database.
Chapter 4.0  Face Recognition Methodology

The face recognition approach used in this project can be divided into 3 stages, face detection stage for detecting the facial regions in the photos, feature extraction stage for extracting features from the cropped facial regions and lastly a classification stage to classify the faces into different persons, below is a brief illustration of the face recognition process (see Figure 2).

![Diagram of face recognition process]

Figure 2: The stages of the face recognition process

4.1 Face Detection Stage

The face recognition process begins with the face detection stage, the face detection algorithm detects the facial regions, the system then crops the facial region and pass it to the Feature Extraction Stage. The face detection algorithm is implemented using the ‘get_frontal_face_detector’ function from Dlib, which uses a sliding window with the Histogram of Oriented Gradient (HOG) algorithm to generate a HOG image of the window, the linear SVM algorithm is then used to compare the generated HOG image with the HOG pattern extracted by training with many faces to determine whether the window contains a face.

4.2 Feature Extraction Stage

In the feature extraction stage, facial features are extracted from the cropped facial region. Two approaches are tested for their efficiency and how well the extracted features describe the face. The first approach uses a convoluted neural network (CNN) to generate 128 features from the cropped region and the second approach uses a landmark estimation algorithm to locate the facial landmarks and calculate the ratios of the different landmarks as the features.
4.2.1 CNN Approach
The convoluted neural network approach uses a deep CNN to generate features from the cropped facial image. We have used a trained CNN model implemented in the Dlib package for this approach as training the CNN will take a lot of time and we did not have time to train the CNN model. The CNN model used is very similar to the facenet algorithm invented by Schroff, F., Kalenichenko, D., & Philbin, J. [1] The face recognition algorithm will find the best 128 features to distinguish different persons. A deep convolutional neural network is used to select the facial points, and in each training iteration, a triplet of photos is used (two photos from the same person and a third photo from a different person). The network will then be tweaked such that the feature points extracted from the same person will be more similar and vice versa. The resulted CNN is then used to generate 128 features from the cropped facial region.

4.2.2 Facial Landmark Approach
Ramesha, Raja, Venugopal and Patnaik have used the shape and texture of the face to perform face matching and have achieved a high matching percentage of 100%. [2] However extracting and computing texture regions for the face can be time consuming, hence we have modified the approach to use only the ratios and colours to describe the face. In additional to the original four ratios describing the shape of the face, we have added 6 more features that describe the face colour and the 3 other ratios of distances between facial landmarks. Below is a table that describe all the features involved in this approach. (see Table 2)

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>Ratio between the distance of the eyes and the eye-nose distance</td>
</tr>
<tr>
<td>f_2</td>
<td>Ratio between the distance of the eye and the eye-lip distance</td>
</tr>
<tr>
<td>f_3</td>
<td>Ratio between the eye-nose distance and the eye-chin distance</td>
</tr>
<tr>
<td>f_4</td>
<td>Ratio between the eye-nose distance and the eye-lip distance</td>
</tr>
<tr>
<td>f_5</td>
<td>Ratio between the eye-nose distance and the width of the nose</td>
</tr>
<tr>
<td>f_6</td>
<td>Ratio between the width of the face and the width of the nose</td>
</tr>
<tr>
<td>f_7</td>
<td>Ratio between the width of the face and the eye-lip distance</td>
</tr>
<tr>
<td>f_8, f_9, f_10</td>
<td>The r, g, b descriptor of the average color of the landmark points on the nose</td>
</tr>
</tbody>
</table>

Table 2: List of features used in the facial landmark approach the first 4 features highlighted in light blue are features from the original approach
The additional features are selected in a way that the ratio will not be significantly affected by facial movements, i.e., the width of the nose and the width of the face used in feature \( f_5 \), \( f_6 \) and \( f_7 \). Therefore, the eye colour and the lip width are not selected as part of any additional feature as they are easily affected by the blink of the eye and lip movement.

To locate the landmarks on the face, we used a trained facial landmark estimation model to locate the 68 different landmarks on the face (see Figure 3). This is a replacement of the canny edge detector in the original approach, increasing the speed of locating the facial landmarks.

4.3 Classification Stage

In the classification stage, the features extracted in the feature extraction stage are used to classify the faces into different persons. The radius neighbor classifier is selected as the classifier for the classification stage. This section will talk about all of the classification algorithms that have been tested in the experiments.

4.3.1 Support Vector Machine (SVM)

A regular SVM works by finding a hyperplane that separates the 2 classes as wide as possible, in this project, SVM is used for multiclass classification. The multiclass SVM classification uses multiple SVM, with 1 SVM for each class, each classifying the corresponding class and the rest of the persons.
4.3.2 SVM with Probability
To make the SVM to be able to identify the outliers, we modified the SVM to make it return probabilities of the face belonging to different classes instead. The probability is calculated by using logistic regression. By using probabilistic output, we can find the highest class probability and compare to a probability threshold, if the highest probability is lower than the probability threshold, we consider the face as an outlier, else we just classify the face to the class of person with the highest probability.

4.3.3 K Nearest Neighbors (KNN)
KNN is a simple classification algorithm that perform classification using votes from the k nearest neighbors. In this project, KNN used the Euclidean distance as the distance metric and weight the votes of the k nearest neighbors with the inverse of the distance such that closer neighbors will have a higher voting power than more distant neighbors.

4.3.4 Radius Neighbors Classifier
The radius neighbors classifier is similar to KNN algorithm. However instead of using k neighbors, it uses every neighbors within the specified radius of the classifier, this allows the algorithm to distinguish outliers by marking faces without any neighbor of faces that belongs to known persons as unidentified faces (outliers). The voting weight function and distance metric of radius neighbors classifier are the same as KNN in this project.

4.3.5 One-class SVM
The one-class SVM is an outlier detection algorithm, it works similarly to the regular SVM, however instead of trying to maximizing the separation of the 2 classes. It tries to maximize the distance of between the hyperplane and the origin, the distance is restricted by the nu parameter, which limits the upper bound on the fraction of training errors (points outside of the hyperplane) and the lower bound of the fraction of support vectors. It was proposed to use this algorithm before the SVM or KNN algorithm to filter out the outliers so that the SVM and KNN classifier will not misclassify the outliers.
Chapter 5.0  Face Recognition Experiment Settings

We have conducted several experiments to determine which algorithm is best for the feature extraction stage and the classification stage of the face recognition process. For the feature extraction algorithms, we test for their ability to determine faces belongs to the same person or not. And for the classification algorithm, we test for how well the algorithm classifies the faces using the LBW dataset.

5.1 Experiments with Feature Extraction Algorithms

5.1.1 Experiment for the efficiency of the feature extraction algorithm

In this experiment, we have selected 1000 image from the LFW dataset [4, 5], each image contains a single face, the figure below showed for some of the image used in this test. (see Figure 4) We then measure the time for the feature extraction algorithms to process all of the images. The total time of the face detection stage and loading the image from the file is also measured, so that we can combine the time in the 2 stages to make a better comparison, i.e. the overhead in loading the image maybe long enough to make the time difference in the feature extraction stage less significant.

![Figure 4: Sample faces in the LBW dataset](image)

5.1.2 Experiment for the Ability to Extract Facial Feature

We chose to use LFW dataset for this experiment as it is a widely used benchmark test for face recognition algorithms, the dataset consists of many face pairs and uses the face pairs to determine whether the faces in the pair belongs to the same person or not. Below are some example of the face pairs in the dataset. (see Figure 5 and Figure 6)
As suggested by the dataset, we have used view 2 for the benchmark for comparing the algorithms, the view 2 of the dataset contains 6000 face pairs with 3000 pairs belonging to the same person and the others belonging to different persons.

To determine whether the faces belong to the same person using the extracted features, we first calculate the Euclidean distance of the face pair given by (1):

$$\text{dist} = f_1^a - f_1^b + f_2^a - f_2^b + \ldots + f_n^a + f_n^b$$

with: $f_n^a$ as feature n of face a / b in the face pair

The Euclidean distance ($\text{dist}_e$) is then compared to a distance threshold ($\text{dist}_t$), if $\text{dist}_e \leq \text{dist}_t$ the face pair is regarded as belonging to the same person, otherwise they are regarded as faces belonging to different persons. The threshold value will be tuned for each approach to find the threshold value that gives the best result for each approach.

Since the usage of Euclidean distance assumes the same weight for different features, the features extracted from the facial landmark approach requires normalization across the column as the ranges of the facial landmark ratios are not the same. The features from the CNN approach does not require normalization as all of the features are in the range of [0, 1].

5.2 Experiments with Classification Algorithms

From the result of the experiment for the feature extraction stage (section 6.1), it has been found that the CNN approach extracts better features and hence we continued to experiment on the algorithms for the classification stage using features extracted by the CNN approach. We used the VGGFace2 as the dataset for the following experiments, which is a large scale image dataset for face recognition. The VGGFace2 dataset is selected as the dataset have a large amount of photos of different persons, on average there are 362 images per subject.[3]
5.2.1 Experiment with Classification Algorithms

**Training and Testing dataset:**

10 subjects are selected for the training and testing dataset. All of the photos of the 10 subjects is split in a 50% train-test split using a stratified split to make sure that the train-test split is split equally amongst the subjects. Below is a table of the number of photos of each of the selected subjects. (see Table 3)

<table>
<thead>
<tr>
<th>Subject</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of photos</td>
<td>424</td>
<td>136</td>
<td>454</td>
<td>180</td>
<td>458</td>
<td>441</td>
<td>279</td>
<td>585</td>
<td>367</td>
<td>522</td>
<td>3846</td>
</tr>
</tbody>
</table>

*Table 3: Number of photos for the each of the selected subjects*

It is noted that the number of photos of each subject are highly varied (ranging from 136 photos to 585 photos), this is to simulate the real life situation of a gallery application, where different persons have different frequency of appearing in the photos, i.e. family members will appear more frequently than friends in the photo gallery.

**Experiment Procedure:**

We performed a grid search on the different kernels and hyper parameters of the SVM classifier to find the best set of parameter. Similarly, we have also tuned the KNN classifier to find the best number of neighbors. The result of the best model for each of the algorithm were then verified using a 5-fold cross validation.

5.2.2 Experiment with one-class SVM for Outlier Detection

Since in real life, there will be photos taken with strangers and friends that meet infrequently. the faces of these strangers and friends most probably are not labeled in the gallery application. Hence it is important that the classification algorithm can properly deal with outliers.

However, the classifiers tested in the section 5.2.1 are not able to identify the outliers, therefore it is proposed to use a one-class SVM to filter the faces before using the SVM and KNN to remove the outliers before passing the faces for SVM and KNN to classify. In this
experiment we tested the ability of the one-class SVM to identify the outliers to explore the possibility of this proposed implementation.

**a) Training and Testing dataset:**
The set of 10 individuals representing 10 known person in the gallery application and another set of photos representing unknown persons (outliers) by selecting 10 photos from 441 different persons.

**b) Experiment procedures**
We tune the ‘nu’ parameter of the one-class SVM to see how well does the best model classify the faces of the known person from faces from the faces belonging to the outliers.

**5.2.3 Experiment with the Modified Algorithms for Outlier Detection**
According to the result in section 6.2.2, the one-class classifier does not perform well enough to identify the outliers, therefore in this experiment we will test the radius neighbors classifier and the SVM classifier with probability and see how well they distinguish the outliers.

**a) Training and Testing dataset:**
The set of 10 individuals is continued to be used in this experiment to represent the photos of known persons. Another set is created to represent the unknown person (outliers) using 10 photos from each of the other 441 persons used to represent photos of unknown persons.

**b) Experiment procedures**
For this test, the SVM with probability and the radius neighbor were tuned with different parameter to find the best model for each of the algorithms using the set of 10 individuals. The radius neighbors classifiers will also be tuned to find the best radius.

We will then repeat the experiment using a 20% test set with the same dataset, this test is to simulate the accumulation of photos in the photo gallery application, and measure how well does the classifier performs with an increased training data size.

We will repeat the experiment again using a 50% test set with photos only belongs to 2 selected persons, this test measures how well does the model handles different amount of
known persons, this is to simulate the changing number of labeled persons in the gallery application.

5.2.4 Measuring Metrics

(1) The prediction accuracy of the model

(2) The percentage of correct prediction of known individuals (inlier score), given by

\[ s_{\text{inlier}} = \frac{\text{number of correctly predicted inliers}}{\text{total number of inliers}} \]  

(3) The percentage of correct prediction of the unknown individuals (outlier score), given by

\[ s_{\text{outlier}} = \frac{\text{number of correctly predicted outliers}}{\text{total number of outliers}} \]  

(4) The training time of the model

(5) The prediction time of the model
Chapter 6.0  Face Recognition Experiment Results

6.1 Result for the Feature Extraction Algorithm

6.1.1 Efficiency of the algorithms

The total time and average time for processing 1000 photos with the feature extraction approaches are presented in the following table (see Table 4). The face detection time and loading time is also measured to provide a comparison with the processing time of the feature extraction algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total Processing Time</th>
<th>Processing time per photo</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN approach</td>
<td>24.882s</td>
<td>0.02488s</td>
</tr>
<tr>
<td>Facial Landmark approach (4 ratios + additional features)</td>
<td>2.770s</td>
<td>0.00278s</td>
</tr>
<tr>
<td>Facial Landmark approach (4 ratios)</td>
<td>2.645s</td>
<td>0.00265s</td>
</tr>
<tr>
<td>Face Detection Time*</td>
<td>37.261s</td>
<td>0.03726s</td>
</tr>
</tbody>
</table>

*The feature detection time also include the time for loading the image

The experimental result of this test has shown that both the CNN approach and the landmark approaches. It has also shown that the computation of additional information of the landmark approach does not increase the processing time significantly and that the landmark estimation algorithm contributed to most of the processing time in the landmark approaches. However, when we take the face detection time into account, the difference in the processing time become less significant, using the Landmark approaches (0.040 s) is only faster than the CNN approach (0.062 s) by 35% and both approach are quick at extracting the facial features, taking less than 0.1 second in processing the image and extracting the features.

6.1.2 Accuracy of the algorithms

We used correct score, precision score and recall score to measure the accuracy of the algorithms. The correct score measures the percentage of the correct result, the precision score measures, the precision score measures the percentage of face pairs predicted as belonging to the same person that are true and the precision score measures the percentage of face pairs belonging to the same person that are correctly identified.
a) CNN Approach

We first conducted a coarse grid search on the distance threshold in the range of [0.4, 0.75] with each value 0.05 apart, the result of the coarse grid search is shown in Figure 7, we can observe that the best distance threshold lies between the range of [0.6, 0.7] where the correct score peaks and the precision and recall score intersects in this region. A fine grid search is then performed to locate a more precise distance threshold, the result of the fine grid search is shown in Figure 8. From the graph we can see that the precision score, recall score and the correct score all intersect at distance threshold = 0.66. Therefore we have determined that the best distance threshold for the CNN approach to be 0.66 with the correct score, precision score and recall score = 0.975.

It is also noted that the actual accuracy of the CNN approach should be higher than the experimental score of 0.975. As after the experiment we have found out that a very small portion of images from the LBW dataset contains more than 1 face. (see Figure 9) As the experiment setup only store a feature list describing only 1 face per image, the face it store may not be the person marked at the image, hence contributing slightly lowered score.

Figure 7: Coarse grid search on distance threshold for the CNN approach

Figure 8: Fine grid search on distance threshold for the CNN approach

Figure 9: Sample images from the LBW dataset that contains more than 1 face
b) Facial Landmark Approach

We first conducted a coarse grid search on the distance threshold in the range of [0.25, 0.65] with each value 0.05 apart, the result of the coarse grid search is shown in Figure 10, we can perform a finer grid search in the region of intersection of [0.4, 0.5]. From Figure 11 we can clearly see that the correct score, precision score and recall score intersect each other at distance threshold = 0.45. Therefore we have determined that the best distance threshold for the CNN approach to be 0.45 with the correct score, precision score and recall score = 0.623.

The low correct score of 0.623 indicates that the features extracted by the facial landmark approach is not good enough to distinguish whether the face pair belongs to the same person, with the corrects score only slightly better the guessing (correct score of 0.5).
We have also compared the result with the version of facial landmark approach with only the four ratios used in the original approach (see Figure 12) and we have found that the inclusion of additional features only provided a very slight improvement on the original correct score of 0.614.

### 6.1.3 Summary on the Feature Extraction Experiments

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Correctness</th>
<th>Precision</th>
<th>Recall</th>
<th>Processing time</th>
<th>Processing time with overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN approach</td>
<td>0.975</td>
<td>0.975</td>
<td>0.975</td>
<td>0.025s</td>
<td>0.062s</td>
</tr>
<tr>
<td>Facial landmark approach</td>
<td>0.623</td>
<td>0.623</td>
<td>0.623</td>
<td>0.03s</td>
<td>0.040s</td>
</tr>
</tbody>
</table>

*Table 5: Result of the feature extraction approaches, the better approach in each of measuring metric is indicated with bold text*

The result of the feature extraction algorithms are summarized in the above table (see Table 5). We have found that the CNN approach for feature extraction significantly more time to process than the facial landmark approach, but when we take the overhead of loading the photo and face detection into account, the difference become more tolerable. In terms of accuracy, the CNN approach is much better at recognizing faces than the facial landmark approach, without the extracting the facial texture map, the facial landmark approach does not correctly determine whether the face pair belongs to the same person. Therefore the CNN approach is selected as the feature extraction algorithm for this project because of its accuracy of identifying faces.

### 6.2 Result for the Classification Algorithms

#### 6.2.1 Performance of the Classification Algorithms

**a) Support Vector Machine**

The hyper parameters of SVM for each kernel (‘linear’, ‘polynomial’ and ‘RBF’) are tuned such that it optimized for the 50% train-split dataset of the 10 selected persons.

The best set of parameter for each of the kernel type are:

1. Linear kernel: $C=100$, $\gamma=0.001$ Accuracy: 0.986
2. Polynomial kernel:$C=1000000$, $\gamma=0.001$, degree = 2 Accuracy: 0.987
3. RBF kernel: $C=100000$, $\gamma=1$ Accuracy: 0.988
b) KNN

![Accuracy of KNN with different k values](image)

*Figure 13: Accuracy of KNN with different k values*

We have tested KNN using different k values (see Figure 13), we have found that the KNN algorithm performs the best when k=3 with an accuracy of 0.987.

### Summary of results of SVM and KNN

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy (50% split)</th>
<th>Cross validated score (5 fold)</th>
<th>Training time</th>
<th>Prediction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (linear)</td>
<td>0.9865</td>
<td>0.9823</td>
<td>0.0914s</td>
<td>0.1055s</td>
</tr>
<tr>
<td>SVM (poly)</td>
<td>0.9870</td>
<td><strong>0.9846</strong></td>
<td>0.1116s</td>
<td>0.1177s</td>
</tr>
<tr>
<td>SVM (RBF)</td>
<td><strong>0.9875</strong></td>
<td>0.9836</td>
<td>0.1305s</td>
<td>0.1283s</td>
</tr>
<tr>
<td>KNN</td>
<td>0.9870</td>
<td>0.9817</td>
<td>0.0151s</td>
<td>0.8059s</td>
</tr>
</tbody>
</table>

*Table 6: Summary of the performance of the classification algorithms, the best model in each metric is indicated in bold*

The table above summarizes the performance of the SVM and KNN classifier. (see Table 6) All of the SVM classifier with different kernel and KNN classifier have similar accuracy of 0.98. However there is a major difference in the training time and prediction time between the SVM models and the KNN model, the SVM models have a longer time training the model than KNN as it needs to compute the hyperplane instead of simply storing the faces in KNN. The KNN model took a much longer time to predict the result than the SVM models as it needs to locate the nearest neighbors whereas in SVM the model only need to determine which side the face belongs to classify the face.

The result has shown that SVM and KNN is suitable algorithms for classifying the faces using features extracted by the CNN approach.
### 6.2.2 Performance of the one-class SVM

Figure 14, 15 and 16 shows the accuracy scores for different ru values for one-class SVM of different kernel, by locating the intersection point of the intersection points in each of the figure, we have the following table.

<table>
<thead>
<tr>
<th>Model</th>
<th>Best ru value</th>
<th>Outlier score</th>
<th>Inlier score</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-class SVM (linear)</td>
<td>0.4</td>
<td>0.495</td>
<td>0.504</td>
</tr>
<tr>
<td>One-class SVM (poly)</td>
<td>0.5</td>
<td>0.498</td>
<td>0.507</td>
</tr>
<tr>
<td>One-class SVM (RBF)</td>
<td>0.15</td>
<td>0.671</td>
<td>0.666</td>
</tr>
</tbody>
</table>

*Table 7: Outlier score and inlier score of the best model of one-class SVM of different kernels*

The result in Table 7 has shown that the one-class SVM is not good for determining faces belonging to unknown persons (outlier). The best model, one-class SVM using RBF kernel is only capable of correctly identifying 67.1% of the outliers and 66.6% of the inliers. This accuracy is not enough for using one-class SVM for reliably filtering outliers and pass the filtered result for the SVM or KNN classifier for classification.

### 6.2.3 Performance of the modified algorithms in outlier detection

#### a) SVM with Probability

The hyper parameters of SVM for each kernel (‘linear’, ‘polynomial’ and ‘RBF’) are tuned such that it optimized for the 50% train-split dataset of the 10 selected persons.

Since the SVM is probability is essentially the same as the regular SVM, we start with the best model found in the previous experiment.

(1) Linear kernel: \( C=100 \), \( \text{gamma}=0.001 \)  

Accuracy: 0.986
(2) Polynomial kernel: C=1000000, gamma=0.001, degree = 2  Accuracy: 0.987
(3) RBF kernel: C=100000, gamma=1  Accuracy: 0.988

The graphs above show the inlier scores and outlier scores of the SVM models with different probability threshold. (see Figure 17, 18 and 19) To find the best probability threshold we for the models, we used the intersection point of the inlier score and outlier score in each of the kernel, which are all located at around probability threshold = 0.85, hence the value of 0.85 is chosen as the best probability threshold for all of the SVM models.

b) Radius Neighbor classifier
We have tested the radius neighbor classifier with different radius value to find the radius value for the best model. From the following graph (see Figure 20), the accuracy of the radius neighbor classifier peaks at around radius =0.48. Hence it is chosen as the radius for the best model.
Summary of the result of SVM with probability and radius neighbor classifier

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Inlier score</th>
<th>Outlier score</th>
<th>Training time</th>
<th>Prediction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (linear)</td>
<td>0.9171</td>
<td>0.9298</td>
<td>0.9192</td>
<td>0.4309s</td>
<td>0.1139s</td>
</tr>
<tr>
<td>SVM (poly)</td>
<td>0.9268</td>
<td>0.9390</td>
<td>0.9293</td>
<td>0.4772s</td>
<td>0.1210s</td>
</tr>
<tr>
<td>SVM (RBF)</td>
<td>0.9382</td>
<td>0.9501</td>
<td>0.9433</td>
<td>0.6056s</td>
<td>0.1447s</td>
</tr>
<tr>
<td>Radius neighbor</td>
<td>0.9683</td>
<td>0.9709</td>
<td>0.9277</td>
<td>0.0119s</td>
<td>2.3000s</td>
</tr>
</tbody>
</table>

Table 8: Result of SVM with probability and radius neighbour using 50% test set and 10 known persons

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Inlier score</th>
<th>Outlier score</th>
<th>Training time</th>
<th>Prediction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (linear)</td>
<td>0.9370</td>
<td>0.9481</td>
<td>0.8814</td>
<td>0.8028s</td>
<td>0.0522s</td>
</tr>
<tr>
<td>SVM (poly)</td>
<td>0.9345</td>
<td>0.9481</td>
<td>0.9111</td>
<td>0.8395s</td>
<td>0.0756s</td>
</tr>
<tr>
<td>SVM (RBF)</td>
<td>0.9549</td>
<td>0.9636</td>
<td>0.9136</td>
<td>1.3450s</td>
<td>0.0840s</td>
</tr>
<tr>
<td>Radius neighbor</td>
<td>0.9662</td>
<td>0.9740</td>
<td>0.9206</td>
<td>0.0206s</td>
<td>1.4351s</td>
</tr>
</tbody>
</table>

Table 9: Result of SVM with probability and radius neighbour using 20% test set and 10 known persons

Table 8 shows the result of the classifiers using 50% test set and 10 known persons all of the classifier have a good accuracy, inlier score and outlier score with the radius neighbor having a much longer prediction time of 2.3s than the SVM classifiers. We than compare the result of Table 9 which shows the result of the classifiers using a 20% test set with 10 classifiers, comparing the result to Table 7, we can observe that all of the outlier score slightly decreased and the inlier scores increased slightly. This shows that putting more photos into the gallery will only affect the outlier detection of the classifiers slightly.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Inlier score</th>
<th>Outlier score</th>
<th>Training time</th>
<th>Prediction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (linear)</td>
<td>0.9481</td>
<td>0.9481</td>
<td>0.2804</td>
<td>0.0096s</td>
<td>0.0033s</td>
</tr>
<tr>
<td>SVM (poly)</td>
<td>0.9893</td>
<td>0.9893</td>
<td>0.2714</td>
<td>0.0108s</td>
<td>0.0035s</td>
</tr>
<tr>
<td>SVM (RBF)</td>
<td>0.9893</td>
<td>0.9893</td>
<td>0.3124</td>
<td>0.0112s</td>
<td>0.0038s</td>
</tr>
<tr>
<td>Radius neighbor</td>
<td>0.9571</td>
<td>0.9571</td>
<td>0.9420</td>
<td>0.0047s</td>
<td>0.0860s</td>
</tr>
</tbody>
</table>

Table 10: Result of SVM with probability and radius neighbour using 50% test set and 2 known persons
Table 10 shows the result of the classifiers using a 50% test split with 2 known persons we can observe that the inlier score of the SVM models have a slight increase when compare to the result in Table 7. For the outlier score, we can observe that the SVM models have a significant drop in the outlier scores while the outlier score of the radius model increased by a small amount. This means that changing the number of persons in the gallery can cause the SVM classifiers to misclassify a majority of photos belonging to unknown persons as known person, therefore the performance of the SVM classifiers in this test is unacceptable.

Since the SVM model performed poorly in the test with 2 known person, misclassifying the majority of the outliers in the test, we have decided to use the radius neighbor as the classification algorithm for the classification stage as it performed steadily in tests with changes in the number of photos and number of known persons. Despite having a longer prediction time than the SVM models, the prediction time of the radius neighbor classifier is still in an acceptable range as the 2.3 seconds in Table 7 predicted the 1923 photos, which on average only takes 0.0012 seconds to process the classification of a single photo.
Chapter 7.0  Project Methodology

In this section, we will talk about the implementation of the more complicated features in this project, namely photo import, face recognition, import filters and duplicate photos search. We will then discuss about the choices made in the implementation and theory behind. For the database objects mentioned in this section please refer to the class diagram (Figure 1) for reference.

7.1 Implementation of Import Photos

Most of the automated features happens on the importation of photos, below is a brief illustration of the events that happens at the importation of the photo.

![Flow of import photo](image)

After user selected the photos to import into the gallery application, a new file name will be assigned to each of the photo using the universally unique identifier version 4 (UUID4), it randomly generate a number of 122 bits, hence it is practically impossible to have file name crashes in the application. The system then passes the specified tags, album, new file name and the original file path to the class function ‘create_photo’ from ‘Photos’ to create a photo.

After the creation of the ‘Photos’ object, the system will then select the newly imported photos using the attribute ‘new’ == TRUE to process them with the face recognition, import filter and duplicate photo search procedures, see section 7.2 to section 7.4 for the details.

7.2 Implementation of Face Recognition

Following the result of the experiments in chapter 6.0. We have implemented the face recognition using the CNN approach for the feature extraction stage and the radius neighbor classifier for the classification stage. This section will mention how face recognition works in the application.
7.2.1 Face recognition at Photo Import

As mentioned in section 7.1, the system will run a face recognition function on all newly imported photo, below is a more detailed flow (see Figure 22) at the ‘3 face recognition’ during importation of the photos shown in Figure 21.

The ‘face extraction stage’ is performed at the import of the photo to make sure that every photo imported into the gallery application would be processed through the feature extraction stage. ‘Faces’ objects are created at this stage and to be associated back to the photo the face belongs, below is a pseudocode for processing a photo in the feature extraction stage

```python
1  img = read_image(img_path)
2  faces = face_detection(img)
   # function to detect the facial regions, returns a list of rectangles containing cropped facial regions
3  face_list = []
4  for face in faces:
5      face_feature_ls = get_feature_list(face)
      # function to compute the feature list from the cropped facial image
6      cropped_img = crop(img, face)
      # function to crop the face image
7      crop_img_path = settings.Gallery_new_face_dir + uuid4()
      # compute the path to store the image using uuid4 and a defined directory stored in settings.py in this directory
```
face_list.append((crop_img_path, feature_ls))
for path, features in face_list:
    new_face = Faces.create(identified = False, img_path=path, features = serialize(features), face_photo = photoID)

The system performs the face detection and feature extraction at the start of the ‘feature extraction’ stage of importing photos. It then crops the facial region of the face and save it, this is to reduce the loading time of showing the face in various screen in the application, as the cropped faces is used in several screen in the application. In particular the ‘Person screen’ and ‘Unidentified faces screen’ can shows hundreds to thousands of photos on the screen, it would be very time consuming if the face image needs to be cropped one by one in these screens. The cropped img is assigned with a file name using uuid4 similar to the photo file to prevent name crashing.

After the ‘feature extraction’ stage, the system then determines eligibility for continuing into the classification stage. The eligibility test if there are 2 or more persons that have 50+ associated faces. The constraint of 50+ associated faces is important as radius neighbor, the classification algorithm used in the face recognition, works by using a weighted voting for all of the neighbors within the specified radius (0.48), the 50+ faces requirement ensures each ‘recognized person’ (persons with 50+ associated faces) have sufficient voting power such that persons with similar faces (faces with Euclidean distance smaller than the specified radius) will be correctly identified.

Person with 50+ faces are regarded as ‘recognized person’ in the application. This is represented by the boolean attribute ‘recognized’ from the ‘Persons’ object stored in the database. The ‘recognized’ attributes also plays a role in the training of the radius neighbors classifiers, only persons with the ‘recognized’ attribute marked with TRUE will be trained in the classification model. Therefore, only ‘recognized persons’ would be automatically recognized in the application..

7.2.2 Labeling of Faces
We have mentioned that the system requires at least 2 recognized persons (person with 50+ identified face) to start the classification process. This section will talk about the manual face labeling process for labeling the first 50+ image for each of the person and the automatic labeling of faces (classification stage).
Manual Face Labeling
Initially, all faces are set to unidentified. The user needs to select an unidentified face and inputs the person the face belongs to. The system will then suggest at most 64 similar faces in return, by comparing the Euclidean distance between the selected face and every other ‘Faces’ object stored in the database and report similar face for faces that have distance below a threshold of 0.55. The value of 0.55 is selected rather than the previously determined 0.66 in section 6.1.2 as we wanted the application to provide more relevant suggestion to the user. Since the threshold value of 0.55 have a precision score of 0.998, it means that the faces suggested by the system should be correct 99.8% of the time, this is much higher than the 97.5% of the value of 0.66.

To speed up the similar face query, 2 optimization is implemented, the first is to terminate the search once 64 similar face images are found and the second optimization is to use the unsquared rooted Euclidean distance for comparison. In the application the value of $0.55^2 = 0.3025$ is used instead of 0.55 for direct comparison with the unsquared rooted value of the Euclidean distance. This saves the computation time in performing the square root procedure.

Automatic Face Labeling (Classification Stage)
The face labeling event occurs at 3 different occasions, provided that the labeled faces passes the minimum requirement of performing face recognition (2+ recognized persons).
(1) Labeling of faces in new photos during photo import
(2) Labeling of unidentified faces during the start of the application
(3) Labeling of unidentified faces when a new person becomes eligible for classification

7.2.3 Initialization of the Face Recognition Models
The face recognition models are loaded at the start of the application with the exception of the classification model, which is trained at the start of the application and when there is a new person becomes eligible for recognition, i.e. the ‘recognized’ attribute in ‘Persons’ changes from False to True. By training the model at the start of the application we can make sure that the classification model is updated for classify the faces using all of the information provided by the user and the previously classified results. The training of the classification
model uses all identified faces in the database to make sure that the classification result is always consistent.

### 7.3 Implementation of Import Filters

The import filters are stored as ‘FilterRule’ objects in the database. The ‘FilterRule’ object contains the following attributes: (see Table 11)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter_id</td>
<td>Primary Key</td>
<td>A filter id for referring to the ‘FilterRule’ object</td>
</tr>
<tr>
<td>priority</td>
<td>INT</td>
<td>An integer for execution order of the ‘FilterRule’ object, the rules are executed in ascending priority</td>
</tr>
<tr>
<td>c_album</td>
<td>STRING</td>
<td>Condition: the name of the album</td>
</tr>
<tr>
<td>c_person_ls</td>
<td>BLOB</td>
<td>Condition: a serialized list of persons</td>
</tr>
<tr>
<td>c_tag_ls</td>
<td>BLOB</td>
<td>Condition: a serialized list of tags</td>
</tr>
<tr>
<td>c_create_date_bf</td>
<td>DATETIME</td>
<td>Condition: the date before constraint</td>
</tr>
<tr>
<td>c_create_data_af</td>
<td>DATETIME</td>
<td>Condition: the data after constraint</td>
</tr>
<tr>
<td>a_album</td>
<td>TEXT</td>
<td>Action: the name of the album to be added</td>
</tr>
<tr>
<td>a_tag</td>
<td>TEXT</td>
<td>Action: the name of the tag to be added</td>
</tr>
</tbody>
</table>

Table 11: Description of attributes of the ‘FilterRule’ object

The import filters work similar to an email filter, you specify the conditions and the actions of the filter, the import filter will automatically perform the specified action on matching photos. The user can specify the range of the date the photos is taken together with the album, person, tags associated with the photos as conditions, these conditions are stored in the condition attributes starting with ‘c_’ in the ‘FilterObject’. For the action, the user can specify the name of the album and tag to be associated with the photo, the action information are stored in the attributes starting with ‘a_’.

At the ‘Import Filter’ stage in the import photo process, a query all the ‘FilterRule’ objects in ascending order of its priority is executed. Afterwards, the class function ‘apply_filter’ is called on each of the ‘FilterRule’ object to apply the filter to all new photos. The ‘apply_filter’ then start the query for matching photo with the filter ‘Photo.new = TRUE’ to filter out most of the irrelevant image. It then continue to match the condition one by one.
A few modifications is made to optimize the speed of the query, this includes querying for the album name, tag name and person name directly on the ‘Albums’, ‘Tags’, and ‘Persons’ objects and if the any of the query returns None, the ‘apply_filter’ function is returned immediately without further evaluation of other attributes. Another modification is to query for the id for the a_album and a_tag before associating for the album and tag for the matching photo, such that we will only need to retrieve the tag_id and album_id for associating each of the photo object. Finally the ‘update_count’ function for the target ‘Albums’ and ‘Tags’ are executed att the end of the apply_filter instead of updating after associating each photo to speed up the apply filter process.

7.4 Implementation of Duplicate Photos Search

The duplicate photo search in this photo gallery application does not only include searching for exact duplicated photos but also duplicated photo of different resolutions. We used the average hash algorithm for finding duplicate photos. Below is a pseudocode of the average hash algorithm

```python
1  def AverageHash(img):
2      small_img = resize_image(img, 8, 8)  # Resize the image to an 8 by 8 image
3          small_img = greyscale(small_img)
4          average_color = average color of small_img
5          img_bits = ‘’
6          for pixel in small_img:
7              if pixel > average_color:
8                  img_bits += ‘1’
9              else:
10                 img_bits += ‘0’
11          return img_bits  # Returns the string of 64 bits
```

The average hash is computed at the creation of the ‘Photos’ object. However, the duplicate photo search is performed at the last stage of the photo importation process. During the ‘duplicate photos search’ stage, the average hash string is queried into the database to find if any of the existing ‘Photos’ object have the same hash, if such hash is found the system will regard the 2 ‘Photos’ object as a duplicate and delete the duplicate.

To speed up the duplicate photo search process, an index is added to the attribute of ‘hash’ of the ‘Photos’ object. By using the average hash algorithm, it allows the implementation of
similar photo search, which search similar photo using the hamming distance between the hash value of the photos, image with hamming distance less than a certain threshold is then regarded as similar images.
Chapter 8.0 Results

8.1 Basic Functions

8.1.1 Import Photos

Users can upload photos into the application by dragging photos into the import page of the application. A preview of the photos dragged will be provided in the left side of the page and user can specify the album and tags to be associated with the uploaded photos and confirm the import action by clicking the ‘import’ button. (See Figure 3)

![Figure 3: Photo import screen](image)

8.2.2 Search Image

The search page of the prototype provide a grid of matching photos to the user. (see Figure 4) The user can specify the keywords to be matched in the search bar and search for desired photos. User can also add ‘t;’ before a keyword to specify photos to have the exact tag name as the keyword and add ‘a:’ to specify for photos of the specific album. The user can also select different image sizes for viewing the search result.

For example, to search for photos with tags of name ‘abc’ and album of ‘123’, the user may type in ‘t:abc a:123’ as the search phrase.
8.1.3 View and modify image

By clicking on any photo in the search grid. The user can access the detailed image page of the photo.(See Figure 5) Users can also edit details about the image by adding tags, removing tags and changing the album of the photo by clicking on the ‘modify info’ button. The user can also search for photos with any of the tags that is associated with the current photo by clicking on any tags in the bottom right corner of the detailed image screen.

Figure 24: Search Screen

Figure 25: Detailed image screen
8.2 Face Recognition

8.2.1 Unidentified Faces Screen
All of the unidentified faces will be loaded in this screen, the user can then click on any of the face image to label the selected face.

![Unidentified faces screen](image)

Figure 26: Unidentified faces screen

8.2.2 Similar Face Suggestion
The system presents similar faces after the user successfully labeled a face, the user can remove faces that does not belong to the same person by clicking on the irrelevant face image. Once the user confirms all faces belongs to the same person, the user can press the ‘confirm’ button.

![Similar face suggestion popup](image)

Figure 27: Similar face suggestion popup
8.3 Advanced Functions

8.3.1 Import Filters and Apply Function

Users can view the existing filters in the upload filter screen, the priority of the filters can be changed by using the ‘UP’ and ‘DOWN’ button on the right. To create new import filter, the user can click the ‘Add New Filter’ button on the top right corner.

In addition to the automatic import photo filtering, the user can also uses the import filter to apply specified action (add tag / add album) to all photos in the gallery with matching criteria. To do so, the user can create a photo filter with the specified conditions and action and click on the ‘apply’ button on the corresponding filter. For example the user can specify a condition of album == ‘album1’ and set the action to add ‘album2’ and click apply to move all of the image from ‘album1’ to ‘album2’.

8.3.2 Duplicate Photo Search

The system automatically detects for duplicate photo on import of new photos, if a duplicate is detected, the photo is not uploaded into the application.

8.3.3 Configuring Storage location

Users can configure the storage location of the photos in the initial setup of the application by changing the value of the gallery home directory stored in settings.py, the python file responsible for storing the storage paths.
8.4 Overall functionality

The functionality of the whole application is tested, we have tested that the face recognition, upload filter and the duplicate photos search and the functions worked as intended. All of the buttons have been clicked to make sure that they perform the correct action.
Chapter 9.0  Limitations

Due to the limited development time, this photo gallery application developed in this project still have a limited amount of functionalities.

9.1 Using pre-trained models for the CNN feature extraction approach

In this project, we have used a pre-trained model for the feature extraction algorithm in the CNN approach. This is because the training of the deep CNN would require millions of images and thousands of hours of training time. Due to limited computation resources we have, we have to use a pre-trained model for the algorithm.

9.2 Python only supporting local storage

It was proposed to have a storage option between the local storage and network storage (i.e. Network Attached Storage) for this photo gallery application. However python can only handle local path and implementing the network storage in the application would require a complete change of the structure of all file open and write operations with new packages to handle the network storage connection. Due to the limited development time, the network storage options is not implemented into the application. However, user can uses other solution to change the file storage location of the application, for example user can use a SMB file plugin to mount the network storage to a mount point such that the network storage location can be referred with a local path.
Chapter 10.0  Future Works

As mentioned in the previous chapter, the application still have a limited amount of functionalities due to the limited development time. We planned to add new features and further optimize the efficiency and the user-friendliness of the application.

10.1 Implementing Similar Photo Search

As mentioned in section 7.4 the current application already used the average hash for determining finding duplicated images, the average hash stored in the ‘Photos’ object can also be used to implement similar photo search for the user. The similar photo will use hamming distance (the number of different bits between the hashes) between the hashes of different photos together with a distance threshold to determine whether the images are similar.

10.2 Improving the loading time of the application

In the current application, loading the search screen and album screen takes a few seconds when there is a lot of image in the gallery / the photo album. This is partly because of the loading of the full image and resize to fit in the grid on the screen. It is proposed to compute a photo thumbnail similar to the cropped facial images that is resized and stored in a separate directory. The thumbnail of the photos will be used instead of the full image file for displaying in the image grid of the search screen and album screen.

10.3 Split the storage of cropped facial image into multiple directory

In the current application, all of the cropped facial images are stored into the same directory, this can lead to slowdown in reading of the cropped facial images when the number of cropped facial images becomes very large. It is proposed to create 64 different directories ranging from ‘00’ to ‘ee’ to store the cropped facial images into the corresponding directory using the first 2 character of the hashed file name.

10.4 Implementing the sort function for screens with search bars

The current application already collect information of the number of views and the last view date of the photos and the all of the existing search functions for executing search queries have already included an ‘order_by’ parameter, therefore the implementation of the sort
function for the search screen and other screens will not be a difficult task and will help the user to locate the photos more easily.

### 10.5 Addition of Help Screen

Some of the functionalities in the application are not straightforward enough for users to understand immediately. For example, the user may have questions on how the face recognition process works in the application. Therefore, it is suggested to add a help screen that displays premade images of storyboards about the face recognition flow in the application and illustrations of what each button does in every screen. This will allow new users to learn about the user interface more easily and improve the user-friendliness of the application.
Chapter 11.0 Conclusion

We have created a photo gallery application that provides a number of convenient photo management features to help users to better manage their photo collection. This includes the face recognition function, import filters, duplicate photo search and other basic photo gallery functionalities.

This paper described the implementation of Intelligent Photo Gallery, and the experiments performed to find the most suitable face recognition algorithm to be implemented into the gallery application. The experiment result has shown that using a CNN to extract the facial features is better than using ratios between facial landmarks. It is also found out that radius neighbors classifier is a more suitable classification algorithm than SVM to classify faces in a photo gallery application as the classifier performed consistently well with increasing number of photos in the application and changing number of identified persons.

An important task to do in the future is to improve the user-friendliness of the application and optimizing the performance of the gallery application. The performance of the current application has not been fully optimized for a massive photo storage, and there are noticeable delays of a few seconds between some of the screen changes. Perhaps precomputing the thumbnail of every photo during the photo import could help reduce the delay of loading large image files. The current application also lacks tutorials and documentation in the application to allow new users to learn to use the application with ease. Implementing a help screen with storyboards explaining the more complex mechanics of the application and illustrations of the UI layout may help improve the user friendliness of the application.
References


