INDIAN INSTITUTE OF INFORMATION TECHNOLOGY ALLAHABAD

(A Centre of Excellence in Information Technology) Deoghat, Jhalwa, Allahabad – 211 012 (U.P.),
India Ph. 0532-2922008, 2922216, Fax: 0532-2430006,
Web: www.iiita.ac.in, E-mail: dr.e@iiita.ac.in

Semester VI – End Semester Project Report

Project Title

EFFECTIVE SUMMARIZATION OF LARGE COLLECTION OF PERSONAL IMAGES

Under the Guidance of:

Prof. U.S. Tiwari

Submission By:
Vaibhav Shrivastava (IIT2013027)
Praneel Rathore (IIT2013034)
Himanshu Tuteja (IIT2013038)
Aayush Garg (IIT2013061)
Anirudh Gupta (IIT2013117)

Table of Content

1. CERTIFICATION

This is to certify that the group of below mentioned students has successfully completed the project work titled **"EFFECTIVE SUMMARIZATION OF LARGE COLLECTION OF PERSONAL IMAGES"** as the VI semester mini-project prescribed by the Indian Institute of Information Technology, Allahabad.

This project is the record of authentic work carried out during the academic year (Jan – May) 2016.

Vaibhav Shrivastava (IIT2013027)
Praneel Rathore (IIT2013034)
Himanshu Tuteja (IIT2013038)
Aayush Garg (IIT2013061)
Anirudh Gupta (IIT2013117)

2. ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

I am highly indebted to Prof. U.S. Tiwari for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.
We would also like to express our gratitude towards Mr. Sudhakar Mishra for his kind co-operation which helped us in completion of this project.
Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

3. ABSTRACT

The amount of personal photos uploaded to social networks (e.g., Facebook, Myspace etc.) and photo sharing sites (e.g., Flickr, Picasa etc.) has been increasing rapidly. According to current estimates, three billion photos are uploaded on Facebook per month. Current photo hosting systems allow users to arrange their personal photos in albums. Any information need requires the user to drill down through the entire collection of photos, using the album / directory structure. This manual browsing may be tedious and inefficient. In this research, we propose a framework for generation of overview summaries from large personal photo collections.

These summaries are representative subsets of the larger corpus and try to capture the relevant information, given the size constraints. They will enable users to get an overview of the interesting information in the photo collections without skimming through the entire database. Personal photo summarization may be a very subjective process. We do not intend to create a unique summary subset from a photo corpus. Rather, our goal is to design a framework for automatic generation of informative overview of the collection.

4. Project Title

EFFECTIVE SUMMARIZATION OF LARGE COLLECTION OF PERSONAL IMAGES

5. Introduction

5.1 The Problem

With the ever increasing amount of photos uploaded on web through social networking sites and photo sharing sites, the need for their effective summarization is becoming more and more important. Also, as a matter of fact, most consumers cannot afford the time to annotate and organize their personal digital photos, they would face genuine problem of accessing them effectively and efficiently. Moreover the consumers wish to share a selective summary of their images with their relatives and friends, increasingly using remote social networking sites and photo sharing sites. Thus effective algorithms for image summarization are a need of the hour.

5.2 The Solution

We propose solution in this report for the problem in hand. We propose a system which uses heterogeneous information sources associated with digital photos and generates a summary. The algorithm adapts itself based on the type of event it is summarizing (Yearbook, Week or Single Day Event). We propose the summarization problem as a retrieval problem based on different types of queries (what/when/who) in this algorithm. The main idea of summarization is to find a representative subset of the data, which contains the information of the entire set.

6. Project Objectives

By doing this project, the followings are meant to achieve:

-Generating effective summarization of digital photos based on the user query on what/when/who.

-Using effective machine learning algorithms to achieve the above mentioned task.

7. Scope of Project Work

The scope of work will include the followings:

a) Study publications and articles about the concepts of image processing.

b) Study the language specifications of python and opencv.

c) Study the machine learning algorithms and choosing the most appropriate algorithms to apply on the problem in hand.

8. Literature Survey

The basic task that is required to accomplish the solution of effective summarization of images is to know about image processing. Any summarization algorithm will only be successful if it encapsulates image processing techniques with itself and based on the results obtained from image processing, can make the appropriate steps for accomplishing the task. Thus as a first step, we chose a tool for doing the image processing work. We chose opencv for this and chose python as our main language for performing tasks in opencv.

8.1 OPENCV [1]:-

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision, originally developed by Intel's research center in Nizhny Novgorod (Russia), later supported by Willow Garage and now maintained by Itseez. The library is cross-platform and free for use under the open-source BSD license. OpenCV is written in C++ and its primary interface is in C++. There are bindings in Python, Java and MATLAB/OCTAVE. Wrappers in other languages such as C#, Perl, and Ruby have also been developed. All of the new developments and algorithms in OpenCV are now developed in the C++ interface

8.2 PYTHON [2]:-

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

8.3 ORB (Feature Descriptor)

ORB (Oriented FAST and Rotated BRIEF) is a fast robust local feature detector that can be used in computer vision tasks like object recognition or 3D reconstruction. Feature matching is at the base of many computer vision problems, such as object recognition or structure from motion. Current methods rely on costly descriptors for detection and matching. A binary descriptor based on BRIEF, ORB is comparatively scale and rotation invariant while still employing the very efficient Hamming distance metric for matching. As such, it is preferred for real-time applications.

8.4 Logistic Regression [3]

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Thus, it treats the same set of problems as probit regression using similar techniques, with the latter using a cumulative normal distribution curve instead. Equivalently, in the latent variable interpretations of these two methods, the first assumes a standard logistic distribution of errors and the second a standard normal distribution of errors.

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}(1-y^{(i)})\log(1-h_\theta(x^{(i)})) + y^{(i)}\log h_\theta(x^{(i)})\right]$$

8.5 Haar-cascade Detection [4]

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Thus, it treats the same set of problems as probit regression using similar techniques, with the latter using a cumulative normal distribution curve instead. Equivalently, in the latent variable interpretations of these two methods, the first assumes a standard logistic distribution of errors and the second a standard normal distribution of errors.

8.6 PCA [5]

The official definition of PCA from Wikipedia is "***Principal component analysis (PCA)*** is a statistical procedure that uses orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components."

From an engineering point of view, PCA is defined as a dimensionality reduction algorithm. Two straightforward examples of where the dimension might need to be reduced are data visualization and data compression. For example in data visualization, spreadsheets with a lot of columns are difficult to visualize. By reducing the data to 3 or 2 dimensions, you can chart/graph it. An example of data compression is an image compression. Processing large images is computationally expensive. By reducing a large image and still preserving the critical data in the image that is required for processing, you can speed up the processing and use less resources.

The PCA eigenvector decomposition algorithm:

- Mean normalize the data (feature scale if needed)
- Calculate the covariance matrix
- Find the eigenvectors in the covariance matrix.
- Sort the eigenvectors from largest to smallest
- Use the largest ones as the principal components
- Multiply the data by the largest components to transform the data

9. Other Related Work

9.1 Effective Summarization of Large Collection of Personal Photos [6] [7]

In this paper, the authors have accomplished the same task with another different approach. They define the metric Quality which determines the aggregate interestingness of a summary as:

Diversity of a summary ensured that it contained minimum redundant information. Diversity of the summary was modelled as an aggregation of the mutual distances of the photo pairs: Div(S) = Min (Dist j. A summary should be a good representative of the larger corpus it is created from. Coverage of a summary was computed by the aggregating the Represent values of each of its photos. $Cov(S, P) = \bigcup \{P \in S\} |Represent(p,P)|$. They modelled summarization as a multi objective optimization function F which jointly maximizes the above properties.

$$S^* = \arg\max F(Qual(S*), Div(S*), Cov(S*, P))$$

for $S^* \subseteq P$ where $|S^*| = M$. F combines the individual properties to generate a single effectiveness metric.

9.2 BIRCH – An effective data clustering method for very large datasets [8]

This paper presents a data clustering method named BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), and demonstrates that it is especially suitable for very large databases. BIRCH incrementally and dynamically clusters incoming multi-dimensional metric data points to try to produce the best quality clustering with the available resources (i.e., available memory and time constraints). BIRCH can typically find a good clustering with a single scan of the data, and improve the quality further with a few additional scans.

10. Methodology

10.1 User Input Query

Our database consists of around 2500 digital photos combined from various trips in different time intervals. User inputs a what/when/who query to give a hint about what type of summarization does the user desire. The subparts of query are defined as follows:

What

This query described the type of summary the user wants. Whether he/she desires a yearbook collection, a weekly collection or summary of some particular trip.

When

This query describes whether the user wants collection of day light images or night images or both.

Who

Our algorithm also lets the user to decide with whom he wants his photos in his summary. He selects the person's name with whom he wants his photos in the summary.

The layout of how the user inputs this query is provided below-

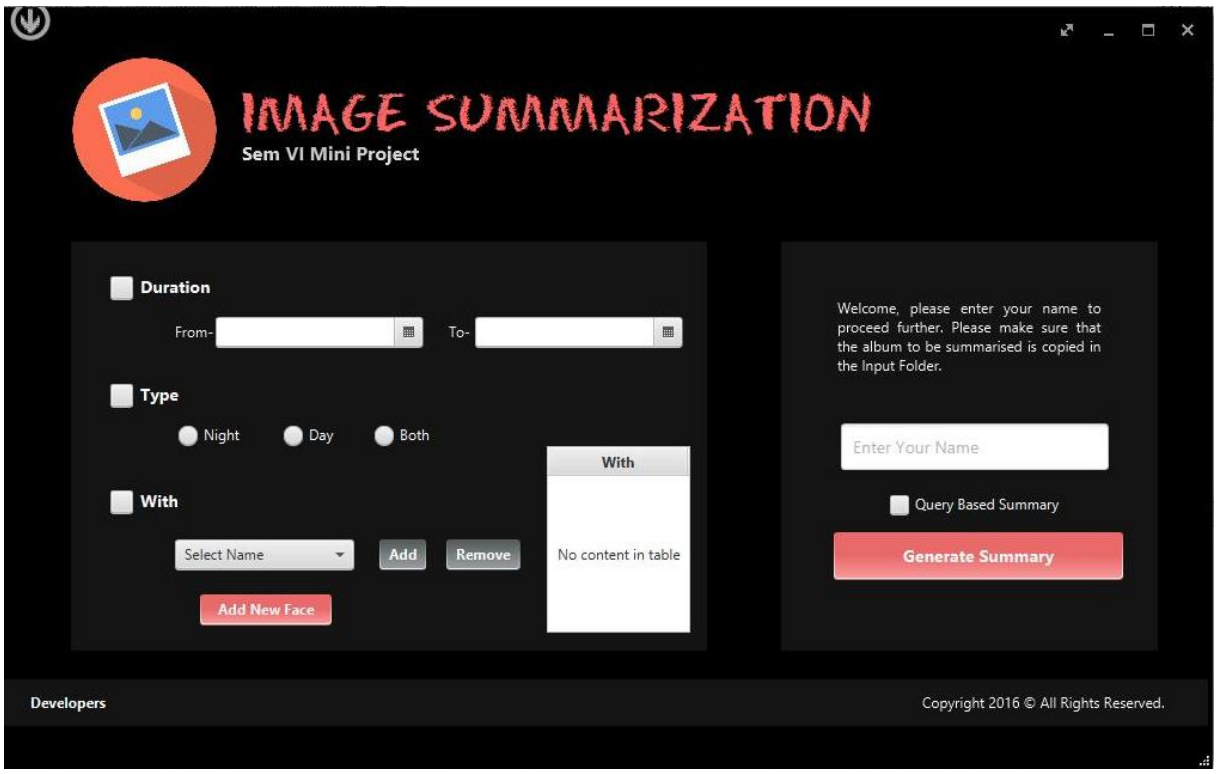


Fig 1.

The user also has a choice to use this query or not by clicking on the check box on the right above *Generate Summary* button. Check boxes on the left ensure whether that particular query associated with that check box is supposed to be used or not.

10.2 Selecting images on the basis of EXIF-Date and EXIF-Time

Once the user has completed providing the query with the type of summarization he/she desires, we sort our images in the database according to the what query provided by the user. For that we use the EXIF parameters of digital images and extract the date and time of creation of the photos. Let's dig a little more into what these EXIF parameters signify.

EXIF - Exchangeable image file format

EXIF is a standard that specifies the formats for images, sound, and ancillary tags used by digital cameras (including smartphones), scanners and other systems handling image and sound files recorded by digital cameras. The specification uses the following existing file formats with the addition of specific metadata tags: JPEG discrete cosine transform (DCT) for compressed image files, TIFF Rev. 6.0 for uncompressed image files, and RIFF WAV for audio files (Linear PCM or ITU-T G.711 μ-Law PCM for uncompressed audio data, and IMA-ADPCM for compressed audio data)

According to the WHAT query of user, the selected candidates of images from our dataset that satisfy the query are used for further selection process. Snippet of the code is provided below –

```
def get_exif(fn):
    ret = {}
    i = Image.open( fn)
    info = i._getexif()
    not_avl = "00 00"
    if info == None:
        print not_avl
        return not_avl
    for tag, value in info.items():
        decoded = TAGS.get(tag, tag)
        #ctime = value[0x9003]
        ret[decoded] = value
        if decoded == "DateTimeOriginal" :
            return value


    return not_avl
```

Fig 2.

```
#SELECT PHOTOS IN THE GIVEN TIME PERIOD
for image in files:
    print "call"
    feature = get_exif(image)
    str = feature.split(' ', 1)
    date[ctr] = str[0]
    time[ctr] = str[1]
    print date[ctr] , " " , time[ctr]
    ctr = ctr + 1

for i in range(0,ctr):
    if date[i] == "00" :
        os.remove(files[i])
        print files[i] , "Removed"
    else :
        print date[i]
        if date[i] < date1 or date[i] > date2:
            os.remove(files[i])
            print files[i] , "Removed"
```

Fig 3.

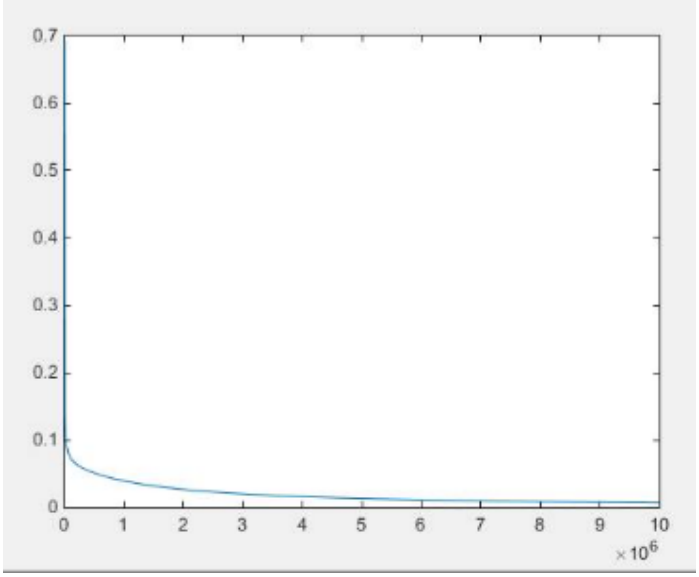
10.3 Selection of Day/Night Images

The new selected database of pictures (selected according to the WHAT query described above) is then further scrutinized to select what type (Day/Night) images our user desires. To accomplish this task, machine algorithm namely, logistic regression is used. Firstly, the classifier was trained using 498 labelled photos consisting both day and night images. The code snippet is provided below-

```
def predict(fn) :
    feature = [0] * blocks * blocks * blocks
    image = Image.open(fn)
    image = resizeimage.resize_cover(image, [300, 300])
    pixel_count = 0

    for pixel in image.getdata():

        ridx = int(pixel[0]/(256/blocks))
        gidx = int(pixel[1]/(256/blocks))
        bidx = int(pixel[2]/(256/blocks))
        idx = ridx + gidx * blocks + bidx * blocks * blocks
        feature[idx] += 1
        pixel_count += 1

    features = [float(x)/float(pixel_count) for x in feature]
    features.insert(0,1)
    ans = 0.0
    l = len(features)

    for i in range(0, l):
        #print features[i]
        ans = (ans + (theta[i] * features[i]));

    print ans
    #print pixel_count
    if ans < 0:
        return 0
    return 1
```

Fig 4. Around 498 photos were used to train the model for prediction of day/night images. To accomplish this task, **Logistic Regression** is used. Feature set for the same is the set of RGB components for every picture.

Fig 5. The image provided on the right is plot of cost function values calculated in the training of logistic regression model used for predicting day/night images.



After selecting the dataset after implementing the WHEN query, the selected images are sent further to decide who all the user wants with himself, in the final summarization.

10.4 Face detection and Recognition

The "who" step in our summarization algorithm consists of face detection and recognition. For face detection we have used Haar like features which are digital image features used for object recognition.

A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. For example, let us say we have an image database with human faces. It is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore a common haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object (the face in this case).

In the detection phase, a window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects. Because such a Haar-like feature is only a weak learner or classifier (its detection quality is slightly better than random guessing) a large number of Haar-like features are necessary to describe an object with sufficient accuracy.

Here for simplicity we have used haar cascades that were available with opencv package in python. We have integrated our face detection scripts with haar_cascade_frontal.xml that was available in opencv as a default haar-like feature detector for faces.

The face recognition part was a bit tricky with the available tools and the problem in hand. As we were training our recognizer on the images of personal photos, it was difficult to align faces of a person for construction of a proper dataset. Nevertheless we tried to achieve a significant dataset for training by removing background from the image of a face and then applying histogram equalization on the images. With this work around, we were able to achieve a considerable amount of accuracy for face recognition without diving into the vast ocean of face recognition with deep learning.

On the above achieved dataset, we applied PCA(Principal Component Analysis) to represent the images in a reduced dimension that can be used for comparing faces. Now for every image in our database, we computed the Euclidean distances between the eigen faces for the trained data images and the given image transformed into its respective eigen face. If we find a mapping for a trained data image for which Euclidean distance with the given image to be less than a threshold value, then the given image is recognised correctly as that image otherwise denotes an imposter or unknown person.

10.5 Clustering/Exemplar Selection

The final dataset selected after time/date selection, day/night selection and the face recognition process, it's time to finally cluster the then selected images into clusters with similar photos falling into the same cluster.

The similarity between the pictures is calculated using two factors, namely, mean RGB values and ORB feature matching.

10.5.1 Mean RGB Feature set

The code snippet for feature collection is provided below-

```
23
24  #extract feature vector for each image
25  print "extracting feature vector for each image..."
26  print
27  tstart = datetime.now()
28  features = zeros((n,3))
29  for i in range(n):
30      im = array(Image.open(imlist[i]))
31      R = mean(im[:,:,0].flatten())
32      G = mean(im[:,:,1].flatten())
33      B = mean(im[:,:,2].flatten())
34      features[i] = array([R,G,B])
35
36  print ("features extracted")
37  print
38  print "Time Used: ",
39  print datetime.now() - tstart
```

Fig 6.

For all the images provided in the working directory, R, G, B feature sets with mean values for every picture is calculated. Thereby, creating the set *features* which contains an array of mean RGB values for every image.

10.5.2 ORB Feature Matching

Brute-Force matcher is simple. It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned. But using ORB Feature mapping, following result is obtained.

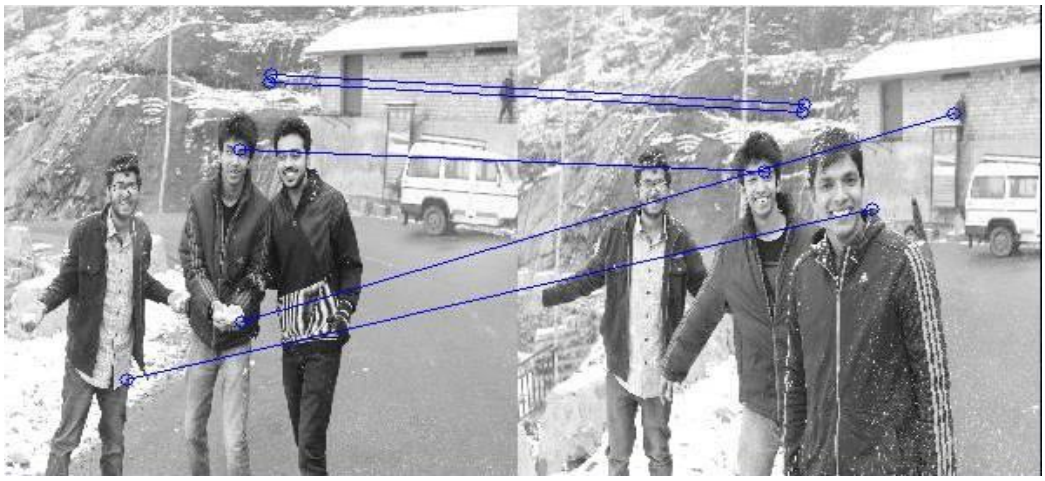


Fig 7. Feature matching on two different yet similar photos.
Common features are mapped perfectly.

The code snippet for feature collection is provided below-

```
59  os.chdir(path)
60  path ="C:/Python2711/codes/orb/images3/"
61  img1 = cv2.imread("0.jpg",0)
62  vec=[]
63  dic = dict()
64  exifdic=dict()
65  for i in range(1,15):
66      img2 = cv2.imread(str(i)+".jpg",0)
67      orb = cv2.ORB(1000,1.2)
68
69      kp1, des1 = orb.detectAndCompute(img1,None)
70      kp2, des2 = orb.detectAndCompute(img2,None)
71
72          # create BFMatcher object
73      bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
74
75          # Matching descriptors.
76      matches = bf.match(des1,des2)
77
78          # Sorting in the order of their distance.
79      matches = sorted(matches, key = lambda x:x.distance)
80          #print len(matches)
81          #vec.append(len(matches))
82      imagename=str(i)
83      dic[imagename]=len(matches)
84
```

Fig 8.

The output of this code (applied for one image namely, 0.jpg) is as follows-

```
Finding features matched of
0.jpg with other files in the directory


Image            Num of features matched
1                130
10               451
4                460
11               465
14               472
7                474
3                479
5                494
9                498
2                503
6                503
13               505
8                508
12               520
```

Fig 9.

The above output shows the number of features that are matched between 0.jpg and the files mentioned above in increasing order. We use these values depicting the number of features matched between pairs of images to cluster the images more efficiently.

Clustering Algorithm – Hierarchical Clustering [9]

The algorithm creates a hierarchical structure which splits and merges images with most similarity into one subtree. In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendogram.

The code snippet for the above algorithm is provided below-

```
30      while len(clust)>1:
31          lowestpair=(0,1)
32          closest=distance(clust[0].vec,clust[1].vec)
33
34          # loop through every pair looking for the smallest distance
35          for i in range(len(clust)):
36              for j in range(i+1,len(clust)):
37                  # distances is the cache of distance calculations
38                  if (clust[i].id,clust[j].id) not in distances:
39                      distances[(clust[i].id,clust[j].id)]=distance(clust[i].vec,clust[j].vec)
40
41                  d=distances[(clust[i].id,clust[j].id)]
42
43                  if d<closest:
44                      closest=d
45                      lowestpair=(i,j)
46
47          # calculate the average of the two clusters
48          mergevec=[(clust[lowestpair[0]].vec[i]+clust[lowestpair[1]].vec[i])/2.0 \
49              for i in range(len(clust[0].vec))]
50
51          # create the new cluster
52          newcluster=cluster_node(array(mergevec),left=clust[lowestpair[0]],
53                                  right=clust[lowestpair[1]],
54                                  distance=closest,id=currentclustid)
55
56          # cluster ids that weren't in the original set are negative
57          currentclustid-=1
58          del clust[lowestpair[1]]
59          del clust[lowestpair[0]]
60          clust.append(newcluster)
```

Fig 10.

A distance vector is maintained for each pair of images in our dataset which is calculated on the basis of 2 features discussed above namely, mean RGB feature set and ORB matching set. According to this distance vector these images are classified into a hierarchical tree structure with similar images lying in the same subtree.

When this clustering was applied on smaller set of images, the dendogram created was as follows-

Fig 11.

11. Results

Now, required images from each cluster are picked randomly and defined in another folder which contains a representative from every cluster that represents all other similar images present in its respective cluster. The final folder contains the required image summary of large collection of dataset.

Fig 12. The summary created by selecting a representative from every cluster

12. Accuracy

Accuracy of individual steps in our entire algorithm is as follows-

- Day/Night Classification is trained with **500** photos and when tested on test data of **100** different images, the accuracy obtained was approximately **88%**.
- Face Detection applied on our dataset resulted in an accuracy of **87%**, which means that around **87%** of faces get detected in our algorithm.
- Face recognition applied using PCA gave an accuracy of **70%** with frontal face images and approximately **46%** on side profile images.

13. References

[1] http://opencv.org/documentation.html
[2] https://docs.python.org/3/
[3] https://en.wikipedia.org/wiki/Logistic_regression
[4] http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0
[5] http://blog.translucentcomputing.com/2014/03/principal-component-analysis-for.html
[6] http://delivery.acm.org/10.1145/2070000/2069208/p9sinha.pdf
[7] http://www.ra.ethz.ch/CDStore/www2011/companion/p127.pdf
[8] http://www.cs.sfu.ca/CourseCentral/459/han/papers/zhang96.pdf
[9] https://en.wikipedia.org/wiki/Hierarchical_clustering