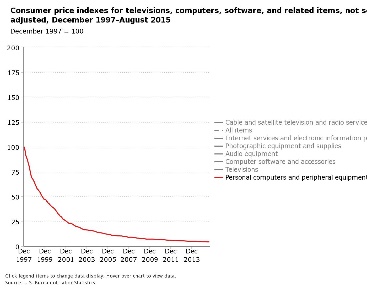
**Face Image Clustering**

Eldin Sahbaz

Syracuse University

esahbaz@syr.edu



**0 Abstract**

In the space of face image clustering, there is no universally agreed upon underlying face representation, distance metric, or clustering scheme. Therefore, I developed a lightweight algorithm that can locate and extract faces from images and organize the extracted faces into groups by individual identity. Our analysis includes using the Yale B dataset to study the accuracy of grouping face images by identity. Our approach was to use the Haar Cascades classifier to extract faces from images, eigenfaces algorithm to extract facial features, and affinity propagation to perform clustering.

Figure 1. Consumer Price Indices for Computers

**1 Introduction**

In recent history, technology has become more powerful and less expensive.[[1]](#footnote-1) This interesting phenomenon is giving a greater portion of the population access to technology they did not previously have. Because of this, we are witnessing what is known as the ‘big data revolution.’ The meaning of big data follows logically from its name. It’s simply, large amounts of data. But what fields are affected by big data? Healthcare, law enforcement, technology, etc. Seeing as every industry is affected by big data, there are many types of data that have been and are being collected. Examples of this is stock market data, heights of ocean tides, healthcare data, etc. However, the data that will be the focus of this paper is image data, specifically images of people.

With the wide spread use of social media such as Instagram, Facebook, and Twitter, there is more data available than ever before. Facebook reports that 350 million images are published on their social network per day.[[2]](#footnote-2) Although social media platforms such as Facebook allow users to tag/link individuals in images with their Facebook accounts, this data is inconsistent. Image tagging is not done autonomously, therefore it is left up to Facebook users to tag individuals in images. Facebook users may tag individuals into images even if there are no people in the picture. Likewise, any Facebook user may tag an image, even if it does not belong to them. Also, any individual may be tagged into the image, even if he or she is not in the image. Furthermore, individuals that have been tagged into images may be untagged. Therefore, tag data should not be incorporated when labelling individuals in images since this data is not validated by social media platforms and may easily be tampered with, inaccurate, or missing altogether.

Having the ability to effectively group individuals in images is very likely to give new insights into how social networks function and may influence social media mining. This sort of data may be used in marketing strategies, in applications that attempt to find the most influential individual in a network, or in applications that attempt to study information dispersal.

Aside from social media mining, the demand for this sort of analysis is present in the law enforcement industry. Given a crime scene camera feed, it is extremely valuable to be able to identify all the different actors in the provided footage. This analysis may not only provide new insight into what perpetrators or victims were present in the footage and present a concise profile. Likewise, the presence of certain faces may direct the investigators to new leads.[[3]](#footnote-3)

Regardless of the application, we may not make any assumptions as to how many individuals may be present in the data or how many images may be present for each unique individual. It is very likely that data may be more densely populated with one individual and not as populated with another individual. This phenomenon poses challenges for many clustering algorithms. Many clustering algorithms, such as DBSCAN, attempt to find clusters of equal density while other clustering schemes, such as K-Means, attempts to form similar sized clusters (DBSCAN is a density based algorithm while K-Means is a distance based algorithm).

In this paper, we explore using eigenfaces as the underlying face representation along with affinity propagation as the clustering scheme. We, like several other published works, test our algorithm on the Yale B dataset since this dataset has a good variance in expressions and facial features. This dataset poses some classical challenges that are faced in traditional facial recognition, while maintaining an identity to total number of images ratio that is comparable to other datasets used by other researchers in this research topic.

The contributions from this paper are:

1. A lightweight face clustering algorithm that requires no labels and uses eigenfaces
2. Face image outlier detection and suppression (i.e. images clustered incorrectly/non-cohesive clusters)

**2 Related Work**

The machine learning problem of clustering (a subset of unsupervised learning) has been very well studied in the past and will likely continue to be well studied. There have been many algorithms and papers published in this area presenting noteworthy results. [1] Likewise, the problem of face recognition has also been well studied in the past. There have been many noteworthy algorithms and papers in this area of research as well – especially with the recent reemergence of deep learning. [2] However, the problem that has not gotten as much attention is face image clustering – a combination of the two.

In a paper published by Menon et al., the proposed method for face image clustering was to use a Hilbert filling curve to reduce 2D image data to 1D. Once they reduced the image to one dimension, a 1D Haar wavelet function is used to decompose the data into a set of low frequency coefficients that were used as features. The paper also proposed an alternative to this approach. In the alternative approach, they would subsample the image to reduce the feature space, then perform a raster scan to extract a feature vector. The real strength from this paper is that they incorporated an incremental learning algorithm for clustering. In their clustering approach, if the distance between samples fell below a correlation threshold, a new cluster would be created. If the distance happened to be greater than this threshold, then the sample would be added to the best-match cluster. [3]

In a paper published by Ho et al., a method was proposed that would work under the conditions of variable lighting. Rather than building a distance matrix, their approach was to build an affinity matrix with conic affinity or gradient affinity as their affinity metric. To build this affinity matrix using conic affinity, they assumed that the object being clustered was a Lambertian object with fixed positioning. This affinity matrix was then fed to a standard spectral clustering algorithm. [4]

Figure 2. Face with Varying Light Source from Ho et al.

In a paper published by Cui et al., an interactive photo annotation system was proposed. This interactive photo annotation system is based on face clustering (the problem we’re trying to solve here). This paper presents the idea of using LBP features extracted from detected faces as well as texture features from an individual’s body for clustering. [5]

Tian et al., built upon the approach published by Cui et al. and allowed for a class that incorporated images that did not cluster strongly enough with any of the other classes. This allows the algorithm to discard samples altogether. [6]

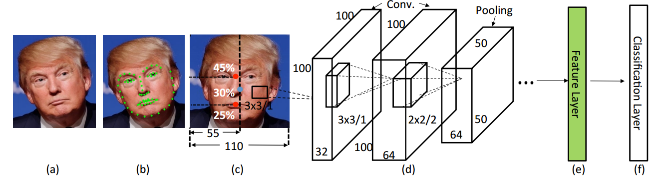
One publication by Otto, et al., proposed using the STASM library to extract key points from an individual’s face. With these points extracted, the combined feature vectors from LBP and HOG was used for PCA and LDA. [7]

Figure 3. Face Classification Process by Otto et al.

Another publication by Otto et al., passing normalized faces through a 10-layer Convolutional Neural Network with a 3x3 filter, constructing a K-Neighbor graph, and clustering the images via rank-order clustering. [8]

Zhu et al., proposed using a new distance measure for clustering face images. The new distance measure, Rankorder distance, is used to measure the similarity of two faces. They based this distance measure on two interesting observations:

1. Two faces of the same person have many top neighbors in common
2. Neighbors of two faces from different individuals generally will greatly differ

Using this new distance metric, they initially let each face be its own cluster, then merge clusters until there are no more clusters to merge (give the distance is below a certain threshold). [9]

Zhao et al., proposed using a 2D Hidden Markov Model to calculate the probability that a certain individual will appear in an image. With this calculated probability, they utilized hierarchical clustering to group images by identity. [10]

Liu et al., proposed using Harr wavelets as well as Menon et al. However, the difference here is once the Haar wavelet features were extracted from images, an approximate nearest neighbor graph was built. This K Neighbors graph was used to find an initial clustering. A union-find algorithm was applied to this initial clustering to find the final clustering. This approach was used to cluster images in general – which could be applied to face image clustering.

Kapoor et al., proposed using active learning with constraints to produce the best face tags. [11]

**3 Proposed Approach**

Face image clustering can effectively be split into three stages:

1. isolating faces from images and normalizing them
2. Clustering the face images by identity
3. Filtering the clusters to maximize homogeneity

**3.1 face extraction and normalization**

Given a set of images, we iteratively visit each image and use Haar Cascades to identify faces present in the image and extract them into a list. Once each detected face is extracted into a list, we normalize the face sizes to a 100x100 matrix. If an image is larger than 100x100, we use cubic interpolation. If an image is smaller than 100x100, we use area interpolation.

Once each face is extracted and resized to 100x100, we normalize the images by converting them from RGB to gray scale and performing histogram equalization to enhance image contrast.

**3.2 image clustering**

With all the faces extracted and normalized, we now transform the faces to the underlying face representation we will be using for clustering.

In our approach, we decided to make use of Turk and Pentland’s eigenfaces algorithm for face recognition. [12] In general, we would be given a train dataset and a test dataset. However, since we are attempting to cluster one set of images, we only use a portion of the original algorithm. The portion of the eigenfaces algorithm that we use is converting each face image into a column vector in an NxM matrix, where N is the number of faces and M is the number of pixels in each face image. We then find the eigenvectors of the NxN vector, then project the faces onto the top K eigenvectors, where K is a positive integer. We then compute the weights for these faces by multiplying the matrix we just computed by the face images.

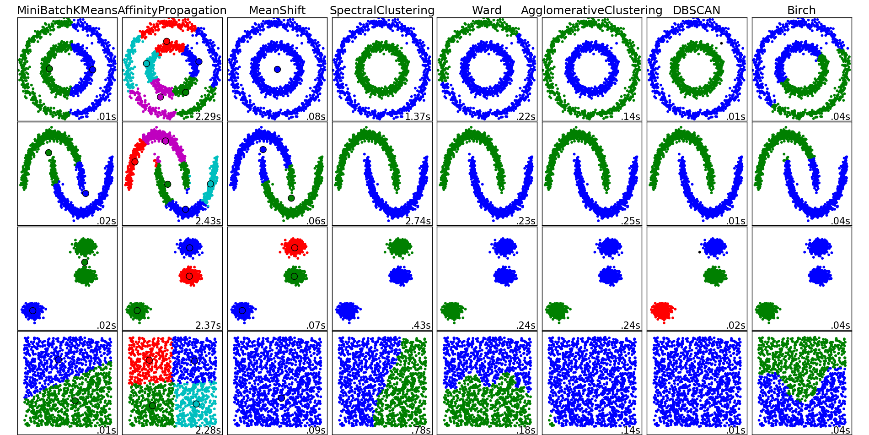
Once we’ve obtained the weights corresponding to each face, we create an NxN distance matrix by using cosine distance. We then normalize this distance maxtrix via Min-Max normalization. This bounds all values between zero and one. Having all values between zero and one makes it easy to convert this distance matrix into an affinity matrix. To convert the distance matrix to an affinity matrix, we simply subtract each entry from one to get the affinity score.

Figure 4. Scikit-Learn’s Clustering Methods

The next step in the process is clustering the affinity matrix. To do this, we passed the similarity matrix as an argument to the affinity propagation clustering algorithm. From the set of clustering algorithms pre-implemented in Scikit-Learn, affinity propagation seemed to best divide the data into clusters.

**3.3 cluster filtering**

The third and final step in our algorithm is to filter bad clusters and misclustered faces from our data. We only want to keep data that is correctly clustered and throw away everything else. We do this in two steps: the first step is to perform coarse-filtering (i.e. filtering at the cluster level), and the second step is to perform fine-filtering (i.e. filtering at the image level). In addition to filtering at two levels, our goal for this stage is to maximize purity. We care more about only getting clean data instead of getting clean data with some bad data.

At the coarse level filtering, we stored the z-scores of the normalized sum of all the distances within a given cluster into a list.

We then remove any cluster whose z-score is greater than one or less than negative one. Generally, outlier detection is done by using three and negative three as the cutoff values. The problem with using these values is that they did not capture any of the bad clusters. Even though we may be capturing more data than we intended, we still achieve our goal of maximizing purity.

The next step is to filter the individual clusters at a finer level. Here, we compute the silhouette coefficients for each of the remaining samples. Each face image receives a value ranging from negative one

****

Figure 5. Cluster Before Filtering



Figure 6. Cluster After Filtering



Figure 7. Removed Cluster

to positive one (since that’s the range of the silhouette coefficient). Since all the silhouette coefficients of our data had only minimal differences, we took the z-scores of these values as well and removed all samples whose z-scores were below negative one or greater than positive one.

**4 Experimental Results**

To evaluate our algorithm’s performance, we tested it against the Yale B database. The Yale B dataset originally has 165 images. Of these 165 images (each containing only one individual), the Haar Cascades classifier was only able to detect 125 faces from these images. After inspecting our initial clustering, our algorithm yielded 16 clusters. This is one more cluster than individuals in the original dataset. Inspecting our clustering after performing our coarse and fine filtering, we ended up with a final count of 11 clusters. While the number of clusters is further from the target number of clusters after filtering than before the filtering, on average the filtered clusters are purer since they’re generally more homogeneous. We say that the clusters after filtering are generally more homogeneous because one of bad clusters from the initial clustering remained. However, all other bad clusters were removed and all other clusters are completely homogeneous.

**5 Future Work**

There is still much work to be done to further improve this algorithm. Since this project can effectively be split into three sub-steps, it follows naturally to analyze the improvements that can be achieved in each of these sub-steps.

The first phase is to perform face extraction and normalization. As mentioned in section three, we used a Haar Cascading classifier to detect and extract faces from images. However, the Haar Cascading classifier is not the most effective approach to this problem. In the future, we would like to implement alternative solutions, such as DLIB and STASM, that have higher accuracy than the Haar Cascading classifier.

As we described in section three, we normalize the size and intensities of the face images, however, there is still much more that can be done to more effectively normalize the face images. In a paper proposed by Otto et al., they rotate detected faces in the image plane to make it upright, find the central point on the face and center the face in the x-axis based on this central point. [13]

The next aspect of our algorithm that we considered working on is testing different clustering schemes. While affinity propagation worked very well in finding accurate clusters in the data, it may be beneficial to use the neural networks framework for clustering. Specifically, the restricted Boltzmann machine and the self-organizing map algorithms are well known and could be used to achieve our goal.

When clustering, it’s very difficult to get perfect clusters, because of this we performed post-clustering filtering. In the future, we would like to implement additional outlier detection and suppression techniques to experiment with and see which approaches most effectively serve our purposes of removing incorrectly clustered data. One possible approach is to use sum of squared errors with the median face instead of the mean face. Likewise, we could incorporate Scikit-Learn’s novelty and outlier detection algorithm to identify images that don’t belong in their respective clusters.

Lastly, we would like to test our algorithm on multiple data sets to see how it behaves in various environments. There are several datasets, that other researchers in this area have used. Each data set works under different assumptions, therefore we would be able to observe the limits of our algorithm. Datasets that others researching face image clustering have used include:

1. Labeled Faces in the Wild
2. YouTube Faces
3. Webfaces
4. CASIA-webfaces

**6 Conclusion**

In this paper, we propose a novel algorithm to extract faces from a set of images and cluster them by identity. The two main contributions from this paper. The first contribution is the face image clustering algorithm we designed using eigenfaces as the underlying face representation and affinity propagation as the clustering scheme. The second contribution from this paper is the face image outlier detection and suppression algorithm that aims to maximize cluster purity by removing any bad clusters present in the clustering and remove all images that are incorrectly clustered. All things considered, our algorithm’s performance meets the expectations we had at the set out of this project.

**7 References**

1. R. Xu, D. Wunsch, “Survey of Clustering Algorithms”, IEEE, 2005.
2. O. Parkhi, A. Vedaldi, A. Zisserman, “Deep Face Recognition”, 2015.
3. M. Menon, E. Boudreau, “An Automatic Face Recognition System Using the Adaptive Clustering Network”.
4. J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, “Clustering appearances of objects under varying illumination conditions,” in Proc. CVPR. IEEE, 2003.
5. J. Cui, F. Wen, R. Xiao, Y. Tian, and X. Tang, “Easyalbum: an interactive photo annotation system based on face clustering and re-ranking,” in Proc. of the SIGCHI conference on Human factors in computing systems. ACM, 2007, pp. 367–376.
6. Y. Tian, W. Liu, R. Xiao, F. Wen, and X. Tang, “A face annotation framework with partial clustering and interactive labeling,” in Proc. CVPR. IEEE, 2007.
7. C. Otto, B. Klare, A. Jain, “An Efficient Approach for Clustering Face Images,” in Proc. ICB. IEEE, 2015.
8. C. Otto, D, Wang, A. Jain, “Clustering Millions of Faces by Identity”, arXiv:1604.00989v1, 2016.
9. C. Zhu, F. Wen, and J. Sun, “A rank-order distance based clustering algorithm for face tagging,” in Proc. CVPR. IEEE, 2011, pp. 481–488.
10. M. Zha, Y. Teo, S. Liu, T. Chua, and R. Jain, “Automatic person annotation of family photo album,” in Image and Video Retrieval. Springer, 2006, pp. 163–172.
11. A. Kapoor, G. Hua, A. Akbarzadeh, and S. Baker. Which faces to tag: Adding prior constraints into active learning. In Proc. ICCV, 2009
12. M. Turk, A. Pentland, “Eigenfaces for Recognition”.
13. D. Wang, C. Otto, and A. K. Jain, “Face search at scale: 80 million gallery,” arXiv preprint arXiv:1507.07242, 2015.

1. <https://www.bls.gov/opub/ted/2015/long-term-price-trends-for-computers-tvs-and-related-items.htm> [↑](#footnote-ref-1)
2. <http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9> [↑](#footnote-ref-2)
3. <https://www.nist.gov/programs-projects/chexia-face-recognition> [↑](#footnote-ref-3)