A Rank-Order Distance based Clustering Algorithm for Face Tagging

Chunhui Zhu*
Tsinghua University, Beijing, China
zhuchunhui2007@gmail.com

Fang Wen Jian Sun
Microsoft Research Asia, Beijing, China
{fangwen, jiansun}@microsoft.com

Abstract. We present a novel clustering algorithm for tagging a face dataset (e. g., a personal photo album). The core of the algorithm is a new dissimilarity, called Rank-Order distance, which measures the dissimilarity between two faces using their neighboring information in the dataset. The Rank-Order distance is motivated by an observation that faces of the same person usually share their top neighbors. Specifically, for each face, we generate a ranking order list by sorting all other faces in the dataset by absolute distance (e. g., L1 or L2 distance between extracted face recognition features). Then, the Rank-Order distance of two faces is calculated using their ranking orders.

Using the new distance, a Rank-Order distance based clustering algorithm is designed to iteratively group all faces into a small number of clusters for effective tagging. The proposed algorithm outperforms competitive clustering algorithms in term of both precision/recall and efficiency.

1. Introduction

The aim of face tagging is to help us to name faces in our desktop/online photo albums for better photo management. Today, most practical solutions [15, 19, 20, 23, 24] and commercial systems [2, 1] semi-automatically addresses this problem by integrating a friendly user interface and advanced vision technologies such as face detection/recognition. Among these works, clustering based methods [20, 23, 24] are arguably the most effective. In this kind of methods, the tagging is performed at the cluster level: automatically group faces into a number of clusters, and interactively give names to clusters. Therefore, the fundamental problem is how to accurately and efficiently group all faces of the same person into a small number of clusters (ideally a single cluster).

Because most photos in family album are taken under uncontrolled environments, any clustering algorithms are facing a few challenges or requirements:

1. Faces in an album usually form a few face clusters in high dimensional space with varying densities, sizes and shapes. This non-uniform distribution makes absolute distance (e. g., L1 or L2 distance between two face recognition features) easy to fail. As shown in Figure 1, the cluster of the boy is more sparse than the girl's cluster. If we use absolute distance on this example, the boy's face in the middle is closer to the girl's cluster than to the boy's.

2. Face detection often returns some faces of no interest or non-faces in the background. Usually, we do not want to tag these faces. The clustering algorithm should be able to handel these noises and outliers.

3. The running time of the algorithm should meet the requirement for quick user interaction.

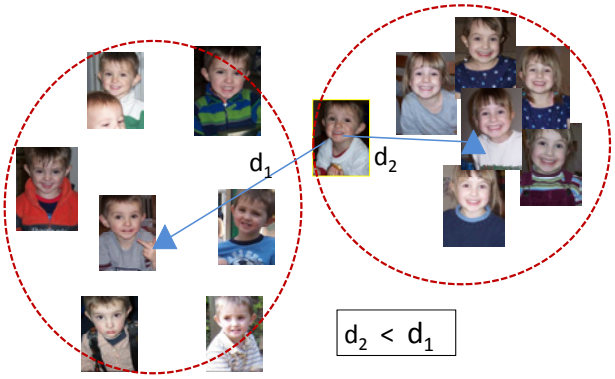


Figure 1. Non-uniform distribution in a face album (Photos are from Gallagher [10].). d_1 and d_2 are absolute distances from the boy in the middle to the center of the two clusters with difference densities. A clustering algorithm may mistakenly group the boy in the middle into right cluster because $d_2 < d_1$. In this paper, we show how to use neighboring structure of each face to address this difficult issue.

To cope with these issues, in this paper, we propose a new Rank-Order distance to better measure the dissimilarity between two faces. Different from the absolute distance, the new distance measures the dissimilarity between their neighborhood structures. It can well handle the non-uniform cluster distribution like varying densities, shapes,

*This work was done when Chunhui Zhu was a visiting student at Microsoft Research Asia.

sizes of face clusters. It is also robust to noises and outliers because the neighborhood structures of these "rare" faces can be easily revealed by the Rank-Order distance.

Due to the complex face distribution, all faces of the same person is usually formed by several *sub-clusters*. Since these sub-clusters are relatively tight, they can be robustly identified by the Rank-Order distance by simple thresholding. However, the connections between sub-clusters are usually weak and sparse due to disturbances from variations in illumination, pose, expression, etc. To tackle this issue, we present a Rank-Order distance based clustering algorithm to iteratively merge sub-clusters in an agglomerative way. The clustering algorithm combines a cluster-level Rank-Order distance and a cluster-level normalized distance. In each iteration step, any two face clusters with small Rank-Order distance and small normalized distance are merged. In such way, different sub-clusters from the same person are effectively connected.

1.1. Related work

K-means [12] and spectral clustering [16, 18, 21] are the most widely used clustering algorithms. However, K-means is sensitive to the initialization and difficult to handle clusters with varying densities, sizes and shapes. Though spectral clustering can handle the non-uniform distribution well, its complexity is high and it usually performs poorly with the existence of noises and outliers. Moveover, both K-means and spectral clustering require specifying the cluster number from the user, which is inappropriate for face tagging tasks here.

Agglomerative hierarchical algorithms[3, 17, 14, 11] do not require pre-defining the cluster number. Starting with each sample as a cluster, agglomerative hierarchical clustering merges the closest pair of clusters that satisfies some distance/similarity criteria in each iteration, until no cluster pair satisfies the merge criteria. The only difference between these algorithms is their various distance/similarity criteria for merging. For example, we can merge clusters with the smallest minimum pairwise distance in each iteration, or clusters with the smallest maximum pairwise distance. Based on some more sophisticated merging criteria, algorithms such as DBSCAN[17], CURE[11] and Chameleon[14] are also proposed. These merging criteria are usually derived from observations from low dimensional data sets, and show satisfactory performance on these tasks, but usually fail to tackle the great challenge from high dimensional space[7].

Affinity Propagation[9] has been proposed to explore the intrinsic data structure by updating passing messages among data points. Though it can get clustering results with high accuracy on both low and high dimensional data sets, the convergence of the algorithm usually requires considerable time, and potential oscillation danger sometimes makes it hard to reach convergence.

Shared Nearest Neighbor method[7] is the closest work to us. This algorithm defines similarity based on the neighborhood two points share, and defines density based on such similarity. It then identifies representative/noise points in the data set through density thresholding. Clusters are finally formed by connecting representative/non-noise points, and noises are automatically located. The problem of this algorithm is its complicated and sensitive threshold parameter selection, which makes it unreliable for different data distribution.

Besides generic clustering algorithms, some specifically designed algorithms have also been proposed for face tagging. In [20], Tian *et al*. proposed to tag faces with partial clustering and iterative labeling. In order to achieve high accuracy, most faces are remained un-grouped, and the sizes of face clusters are usually small. Therefore, many user operations are still needed to label all the faces. In [13], Kapoor *et al*. suggested integrating match/non-match pairwise priori constraints into active learning, in order to give the best face tagging order. To achieve satisfactory performance, this method relies on the amount of available priori constraints, whose number is usually limited under real face tagging scenarios.

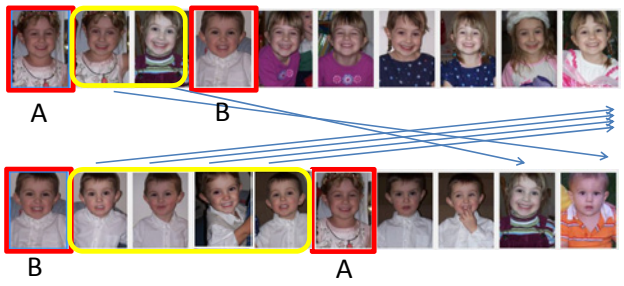
To compensate the imperfect property of face recognition feature, a few methods[15, 19, 24] choose to incorporate extra information such as social context, body, time, location, event, and torso identification. As long as such extra information is available and reliable, we should use them. But in this paper, we focus on face recognition feature.

2. Rank-Order Distance

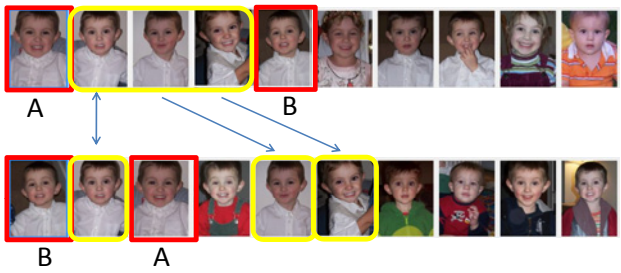
In this section, we introduce a new distance, called Rank-order distance, to measure the similarity of two faces. This distance is based on an interesting observation: two faces from a same person tend to have many shared top neighbors, but the neighbors of two faces from different persons usually differ greatly.

In Figure 2, we generate top neighbor lists for four faces by sorting absolute distance. Here, we use L1 norm between extracted face recognition features [4] as the absolute distance. In Figure 2(a), A and B are faces from different persons, and their top neighbors differs a lot even though absolute distance between them is quite small. On the other hand, A and B in Figure 2(b) are faces from a same person, and their top neighbors resemble a lot. Note that the absolute distance here cannot distinguish these two different cases, while their different neighborhood structures can. This motivates us to define a new metric to measure the dissimilarity between the neighborhood of two faces.

Formally, given two faces a and b, we first generate two order lists O_a and O_b by sorting faces in the dataset according to absolute distance, as shown in Figure 3. Next we



(a) A's top neighbors ≠ B's top neighbors



(b) A's top neighbors = B's top neighbors

Figure 2. Top neighborhood lists of two face pairs. (a) from different persons: although B is the second nearest neighbor of A, their Rank-Order distance is large (=226) since they do not share too many top neighbors. (b)from the same person: the Rank-Order distance between A and B is small (=8.5) because their top neighbors are well overlapped. Photos from Gallagher [10].

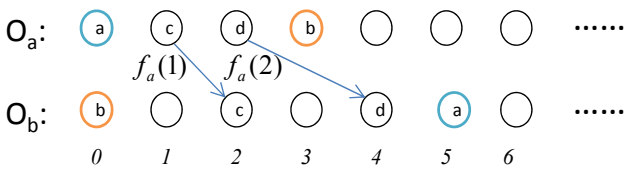


Figure 3. O_a and O_b are two order lists which are ranked using face a and b. The asymmetric distance $D(a,b) = O_b(f_a(0)) + O_b(f_a(1)) + O_b(f_a(2)) + O_b(f_a(3)) = O_b(a) + O_b(c) + O_b(d) + O_b(b) = 5 + 2 + 4 + 0 = 11$.

define an asymmetric Rank-Order distance $D(a,b)$ between a and b:

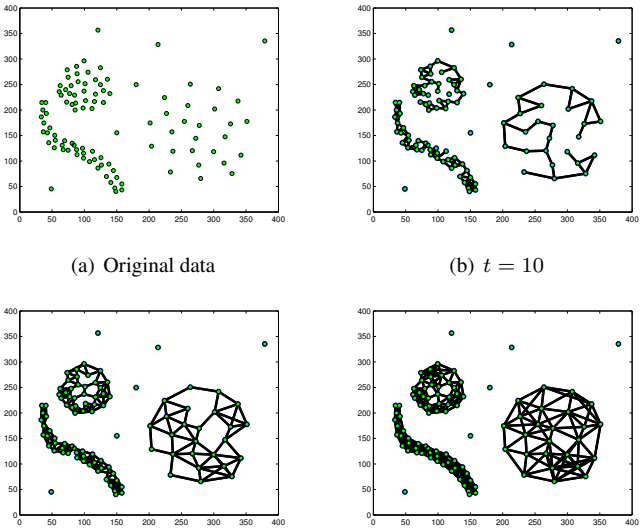
$$D(a,b) = \sum_{i=0}^{O_a(b)} O_b(f_a(i)), \tag{1}$$

where $f_a(i)$ returns the i^{th} face in the order list of a. For example, $f_a(1)$ refers to face c, the nearest one to face a in Figure 3. $O_b(f_a(i))$ returns the ranking order of the face $f_a(i)$ in b's order list. $O_a(b)$ is the order of face b in a's order list. Essentially, this distance is the summation of rank orders of a's top neighbors in b's order list, as illustrated in Figure 3. The small distance means many a's top neighbors are also b's top neighbors, and vice-versa.

We further normalize and symmetrize the distance in Equation (1) to obtain our final Rank-Order distance:

$$D^R(a,b) = \frac{D(a,b) + D(b,a)}{\min(O_a(b), O_b(a))}, \tag{2}$$

where $\min(O_a(b), O_b(a))$ is a normalize (average) factor to make the distance comparable. This normalization is important since $D(a,b)$ is biased towards penalizing large $O_a(b)$. $D(b,a)$ is defined by switching the roles of a and b. Note that $D^R(a,b)$ doesn't always satisfy triangle inequity for a strict distance definition. However, we name it Rank-Order distance here for convenience.



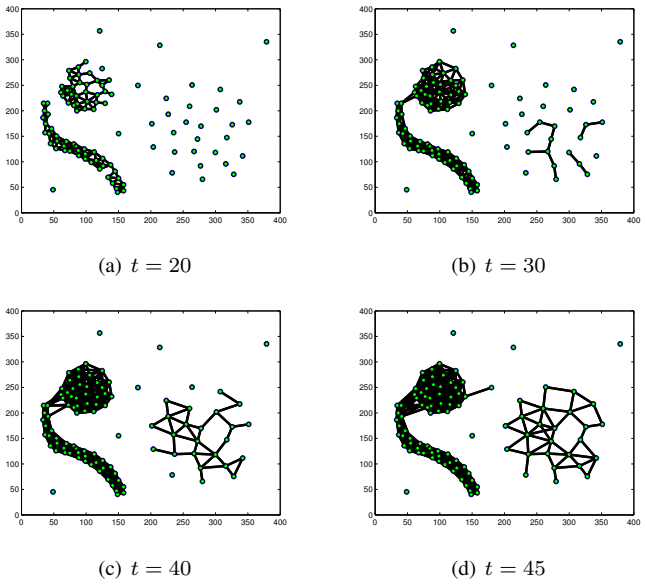
(a) Original data (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

Figure 4. 2D clustering by Rank-Order distance. We draw an edge between any two points whose Rank-Order distance is smaller than the threshold t. Three clusters are robustly discovered under varying threshold values.

Figure 4 shows a 2D example to demonstrate how the Rank-Order distance can robustly discover non-trivial clusters. In this example, there exist three natural clusters with varying densities, shapes and sizes, and noises (outliers). We plot all edges between points whose Rank-Order distance is smaller than a given threshold t. It can be seen that the Rank-Order distance can easily identify three clusters and is insensitive to threshold values. For comparison, we also do the same operations with the absolute (Euclidean) distance on this example in Figure 5. It shows that the absolute distance is not good for coping with varying density/shape/size clusters.

3. Rank-Order Distance based Clustering

The above simple clustering method, merging faces with small Rank-Order distance (below a certain threshold), works well for 2D examples we introduced. But, itself is



(a) $t = 20$ (b) $t = 30$

(c) $t = 40$ (d) $t = 45$

Figure 5. 2D clustering by Euclidean distance. We draw an edge between any two points whose Euclidean distance is smaller than the threshold t. Clusters with varying density/shape/size are challenging for the absolute distance.

insufficient to handle challenging face tagging problem. In a real face album, all faces of the same person usually consist of several "sub-clusters". Inside each "sub-cluster", the faces are tightly connected. Between sub-clusters, the connections are often weak and sparse due to variations in illumination, pose, expression, etc.

Simply merging faces with small Rank-Order distance will result in too many high-precision, tight sub-clusters. To cope with this difficulty, we present an iterative clustering algorithm to merge "sub-clusters", using the combination of a cluster-level Rank-Order distance and a cluster-level normalized distance.

Cluster-level Rank-Order distance. To compute the cluster-level Rank-Order distance, we first need to define cluster-level absolute distance. Although there are many sophisticated cluster (or called image set) distances [5, 22, 8], we find the closest distance between two clusters is simple and effective in our experiments:

$$d(C_i, C_j) = \min_{\forall a \in C_i, b \in C_j} d(a, b), \tag{3}$$

where a and b are faces in face clusters C_i, C_j respectively, and $d(a, b)$ is their absolute distance. With the distance $d(C_i, C_j)$, we then generate the order lists at the cluster level as described in previous section. Finally, the cluster-level Rank-Order distance is defined as:

$$D^R(C_i, C_j) = \frac{D(C_i, C_j) + D(C_j, C_i)}{\min(O_{C_i}(C_j), O_{C_j}(C_i))}, \tag{4}$$

where O_{C_i} and O_{C_j} return the ranking order at the cluster level, and $D(C_i, C_j)$ and $D(C_j, C_i)$ are computed with (1) at the cluster level.

Cluster-level normalized distance. When the number of clusters becomes smaller as the merging goes, the Rank-Order distance is less meaningful because there is not sufficient and reliable neighboring structure to explore. Therefore, we add an auxiliary distance to help the merging decision. Note that any global distance or threshold may lead to poor performance, as we have seen in Figure 1 and 5. To avoid using global distance, we define a cluster-level, locally normalized distance:

$$D^N(C_i, C_j) = \frac{1}{\phi(C_i, C_j)} \cdot d(C_i, C_j),$$
$$\phi(C_i, C_j) = \frac{1}{|C_i| + |C_j|} \sum_{a \in C_i \cup C_j} \frac{1}{K} \sum_{k=1}^{K} d(a, f_a(k)), \tag{5}$$

where $f_a(k)$ returns face a's k^{th} nearest neighbor, $|C_i|$ and $|C_j|$ are the numbers of faces in C_i and C_j, K is a constant parameter, and $\phi(C_i, C_j)$ is the average distance of faces in two clusters to their top K neighbors (in the whole dataset). $D^N(C_i, C_j)$ is locally normalized distance which considers the local density information. Thus, the distances in either dense or spare areas can be evaluated under an uniform scale.

Clustering algorithm. With two defined distances, our clustering algorithm runs in the following way:

1. Let each face be a cluster.
2. Merge any cluster pair if their Rank-Order distance and normalized distance are under certain thresholds.
3. Stop if no cluster can be merged; otherwise update clusters and cluster distances, and go to 2.

The detailed description of the algorithm is shown in **Algorithm 1**. Besides forming some face clusters, the algorithm also outputs an "un-grouped" face cluster C_{un} which contains all individual faces that can not be merged.

Take the task in Figure 1 for instance, our algorithm can successfully find the desired two clusters in two steps. In the first step, the boy's and girl's clusters will be found, but the boy in the middle remains un-grouped due to big Rank-Order distance. In the second step, the normalized distance will favor merging the un-grouped boy into the the sparse cluster of boy rather than the dense cluster of girl.

For a practical implementation, we may not need to examine all pairs. We find that only considering the pairs between each cluster and its top neighbors (e. g., 20) is often sufficient. The results are not greatly affected by this method. Therefore, the main computational cost of the clustering algorithm is to compute the pairwise absolute dis-

Algorithm 1 Rank-Order distance based clustering

Input:

N faces, Rank-Order distance threshold t.

Output:

A cluster set $\mathbf{C}$ and an "un-grouped" cluster C_{un}.

1: Initialize clusters $\mathbf{C} = \{C_1, C_2, ..., C_N\}$ by letting each face be a single-element cluster.
2: **repeat**
3: **for all** pair C_j and C_i in $\mathbf{C}$ **do**
4: Compute distances $D^R(C_i,C_j)$ by (4) and $D^N(C_i, C_j)$ by (5).
5: **if** $D^R(C_i, C_j) < t$ and $D^N(C_i, C_j) < 1$ **then**
6: Denote $\langle C_i, C_j \rangle$ as a candidate merging pair.
7: **end if**
8: **end for**
9: Do "transitive" merge on all candidate merging pairs. (For example, C_i, C_j, C_k are merged if $\langle C_i, C_j \rangle$ and $\langle C_j, C_k \rangle$ are candidate merging pairs.)
10: Update $\mathbf{C}$ and absolute distances between clusters by (3).
11: **until** No merge happens
12: Move all single-element clusters in $\mathbf{C}$ into an "un-grouped" face cluster C_{un}.
13: **return** $\mathbf{C}$ and C_{un}.

tance and to sort each face's neighborhood. Its time complexity is $O(N^2)$.

4. Experimental Results

Figure 6. From top to bottom, five rows are example faces from MSRA-A, Easyalbum, Gallagher, MSRA-B and non-faces due to false face detection.

Database	#faces	#persons	#noises
MSRA-A	1,322	53	166
Easyalbum[6]	1,101	30	26
Gallagher[10]	829	27	20
MSRA-B	489	6	32

Table 1. Number of faces, persons and noises in the four albums. Noises refer to faces of no interest or non-faces due to false face detection. The number of persons excludes persons of no interest.

In this section, we evaluate our algorithm on four face albums and provide comparisons with other competitive clustering algorithms. Four face albums are: two public albums Easyalbum[6] and Gallagher[10], and two our own albums, MSRA-A and MSRA-B, contributed by our colleagues. MSRA-A contains daily photos of our colleague and a lot (> 50) of his friends; Easyalbum is a family album for a child growing from baby to kid, and his family and friends during those years; Gallagher is a family album containing photos from three children, other family members and their friends; MSRA-B contains traveling photos of our colleague and a few of his friends. Some face examples are shown in Figure 6 and statistics of each album are listed in Table 1. In all four test data sets, there exist faces of no interest (e.g. faces in the background) and non-faces due to false face detection, and we define them as noises in the data.

4.1. Evaluation metrics

A good clustering result should have the following properties: the majority of faces in each cluster should belong to a same person; each cluster should contain as many faces as possible; most noises including faces from persons of no interest and non-faces should remain un-grouped. Therefore, given a clustering result, a face cluster set $\mathbf{C} = \{C_1, C_2, ..., C_L\}$ and"un-grouped" cluster C_{un}, we evaluate it in the following way.

The precision of each face cluster is important because we do not want to let the user pay much attention to correct any mis-grouped faces in every cluster. We define a global *Precision* of all clusters as:

$$Precision = 1 - \frac{\sum_{i=1}^{L} |\#M_i|}{\sum_{i=1}^{L} |C_i|}, \tag{6}$$

where $|\#M_i|$ is number of mis-grouped faces in C_i.

We also want to include as many non-noise faces as possible in the face clusters but not in the un-grouped cluster. We define *Recall* to measure how many non-noise faces are grouped:

$$Recall = \frac{\sum_{i=1}^{L} (|C_i| - |n_i|)}{|F| - |\#noise|}, \tag{7}$$

where $|n_i|$ is the number of noise faces in C_i, $|\#noise|$ is

Database	Affinity Propagation									SNN			Rank-Order		
	$p = 0.9$			$p = 1.1$			$p = 1.3$								
	P	R	CR	P	R	CR	P	R	CR	P	R	CR	P	R	CR
MSRA-A	.97	.85	3.07	.84	.98	5.61	.77	.98	7.81	.98	.32	4.20	.98	**.87**	**7.02**
Easyalbum	.98	.87	3.07	.88	.99	5.77	.83	.99	8.51	.98	.22	4.57	.98	**.89**	**7.56**
Gallagher	.97	.85	3.41	.81	.99	8.97	.78	.99	13.8	.97	.18	5.96	.97	**.86**	**9.12**
MSRA-B	.99	.87	3.50	.91	.99	6.64	.88	.99	12.5	.95	.15	3.75	.99	**.93**	**19.8**

Table 2. Comparison of Affinity Propagation, Shared Nearest Neighbor, and our algorithm. At the same *Precision* (P) level (around 98%), our algorithm is clearly superior in both *Recall* (R) and *Compression Ratio*(CR).

the number of noises in the album, and $|F|$ is the number of all faces(including noise faces) in the album.

Since the size of each cluster directly determines how many faces can be tagged by one user operation, we define a *Compression Ratio* to measure the average size of all clusters:

$$Compression\ Ratio = \frac{1}{L}\sum_{i=1}^{L}|C_i|. \tag{8}$$

The higher *Compression Ratio* is, the less user interaction is required.

For completeness, we also use general clustering quality metric, Normalized Mutual Information (NMI), to evaluate different clustering algorithms. NMI takes *Precision*, *Recall* and *Compression Ratio* all into account and is calculated as:

$$NMI(\mathbf{\Omega}, \mathbf{C}) = \frac{I(\mathbf{\Omega}, \mathbf{C})}{\sqrt{H(\mathbf{\Omega})H(\mathbf{C})}}, \tag{9}$$

where $\mathbf{\Omega}$ is the ground truth cluster set, $H(\mathbf{\Omega})$ and $H(\mathbf{C})$ are the entropies for cluster sets $\mathbf{\Omega}$ and $\mathbf{C}$, and $I(\mathbf{\Omega}, \mathbf{C})$ is the mutual information between $\mathbf{\Omega}$ and $\mathbf{C}$.

4.2. Results

Our clustering algorithm contains two parameters, Rank-Order distance threshold t, and the number of top neighbors K in Equation (5). We set $K = 9$ and $t = 14$ in all experiments. In Figure 7, some face clusters and the un-grouped cluster generated by our algorithm on Gallagher album are shown. As we can see, faces from a same person with large variations can be correctly clustered, and some noises and outliers are automatically put in the un-grouped cluster.

We compare our algorithm with Affinity Propagation(AP) [9], Spectral Clustering(SC)[16], Shared Nearest Neighbor(SNN)[7] and K-means(KM)[12]. For all clustering algorithms, we use the same learning-based descriptor [4] for face representation and the same L1 distance between face representations as the absolute distance.

4.2.1 Comparison with AP and SNN

For AP, we set its preference parameter to $p \times median(S)$, where p is scale factor and $median(S)$ is the median of all pairwise similarities. By changing p, we report the result of AP in Table 2. In SNN[1], there are five parameters: neighborhood size M, weak-link threshold t_w, representative point threshold t_r, noise threshold t_n, and merge threshold t_m. We tune these parameters to obtain the best

Figure 7. Some face clusters and un-grouped cluster generated by our algorithm on Gallagher album.

[1] Code available at http://www-users.cs.umn.edu/ ertoz/snn/.

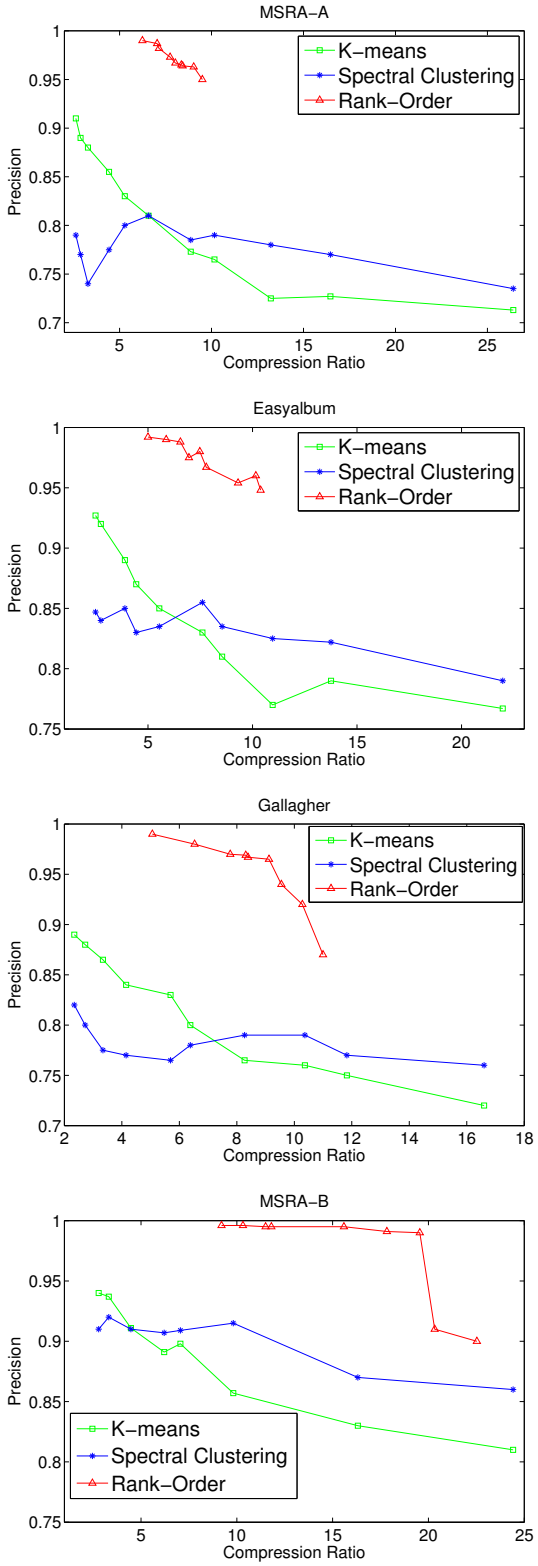


Figure 8. Results for *Compression Ratio* vs. *Precision* on the four albums.

result we can get. The results of SNN around 98% *Precision* level are also given in Table 2.

By viewing the result, we can see that under the same *Precision*(around 98%) level, our algorithm outperforms AP a little in *Recall*, and significantly in *Compression Ratio*. Although the *Compression Ratio* of AP can reach the same level when we use a large $p = 1.3$, the *Precision* will become too low to be acceptable in the real application. For SNN, in order to achieve high *Precision*, we must set a very strict merging threshold, which leads to low *Recall* and *Compression Ratio*.

4.2.2 Comparison with KM and SC

Because the majority of clusters produced by K-Means (KM) and spectral clustering (SC) often contain more than one face, the *Recall* of KM and SC is nearly 1 especially when the cluster number is small. Therefor, we only compare *Precision* vs. *Compression Ratio*. We plot *Compression Ratio* against *Precision* curves in Figure 8 on all the albums for KM, SC, and our algorithm. Given different cluster numbers on each album, in KM, we select best one from 100 runs; in SC, we select the best result under different scale parameter σ [18] in calculating the pairwise similarity. For our algorithm, we plot results by varying $t \in [4, 20]$ with interval of 2 when $K = 9$.

Viewing Figure 8, while KM and SC can achieve acceptable *Precision* when cluster number is large, the *Compression Ratio* is low. If we decrease the cluster number for SC and KM, its *Precision* will fall due to inability of handling noises and outliers which should be un-grouped but in fact are mistakenly merged with other faces due to limited number of cluster. Meanwhile, our algorithm achieves a good balance between *Precision* and *Compression Ratio*.

4.2.3 NMI evaluation

Database	KM	SC	RO	AP	SNN
MSRA-A	.675	.678	**.787**	.693	.750
Easyalbum	.545	.525	**.705**	.573	.559
Gallagher	.565	.535	**.744**	.599	.573
MSRA-B	.673	.677	**.827**	.628	.555

Table 3. NMI for different algorithms on four albums. RO stands for our Rank-Order distance based clustering algorithm.

Finally, we use NMI as in (9) to evaluate the performance of all algorithms at the same time. NMIs in Table 3 are computed with the best clustering results of each algorithm. Our algorithm can achieve consistent superior performance on all albums with highest NMI.

4.3. Runtime

As mentioned in Section 3, we need not examine all pairs for $D^R(C_i, C_j)$ and $D^N(C_i, C_j)$, and we only consider pairs between each cluster and its top neighbors(e.g., 20). The time complexity of our algorithm is $O(N^2)$. In Figure 9, we plot the runtime of all algorithms under different data size. Every plotted point is the average of 20 runs. The pairwise absolute distance computation time is not added in the result.

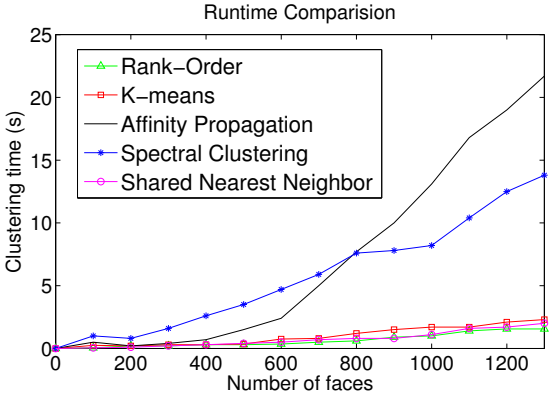


Figure 9. Runtime comparison.

5. Conclusion

We present a novel Rank-Order distance and an iterative clustering algorithm built on the distance. The Rank-Order distance is better for handling data distribution with varying densities/shapes/sizes, and noise/oulier. On the face tagging problem, we see the superior performance by this algorithm. Since the core distance and algorithm are generic, we are planning to study their values in more computer vision problems.

References

[1] Picasa web album. http://picasa.google.com/.

[2] Windows live photo gallery. http://photogallery.live.com/.

[3] A.K.Jain and R.C.Dubes. Algorithms for Clustering Data. Prentice Hall, 1988.

[4] Z. Cao, Q. Yin, X. Tang, and J. Sun. Face recognition with learning-based descriptor. In *Proc. CVPR*, 2010.

[5] H. Cevikalp and B. Triggs. Face recognition based on image sets. In *Proc. CVPR*, 2010.

[6] 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*, pages 367–376, 2007.

[7] L. Ertöz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *SIAM international conference on data mining*, volume 47, 2003.

[8] W. Fan and D. Yeung. Locally Linear Models on Face Appearance Manifolds with Application to Dual-Subspace Based Classification. In *Proc. CVPR*, 2006.

[9] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972, 2007.

[10] A. Gallagher and T. Chen. Clothing cosegmentation for recognizing people. In *Proc. CVPR*, 2008.

[11] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases* 1. *Information Systems*, 26(1):35–58, 2001.

[12] J. Hartigan and M. Wong. A k-means clustering algorithm. *JR Stat. Soc., Ser. C*, 28:100–108, 1979.

[13] A. Kapoor, G. Hua, A. Akbarzadeh, and S. Baker. Which faces to tag: Adding prior constraints into active learning. In *Proc. ICCV*, 2009.

[14] G. Karypis, E. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 2002.

[15] M. Naaman, R. Yeh, H. Garcia-Molina, and A. Paepcke. Leveraging context to resolve identity in photo albums. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 178–187, 2005.

[16] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2(14), 2001.

[17] J. Sander, M. Ester, H. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2002.

[19] B. Suh and B. Bederson. Semi-automatic image annotation using event and torso identification. *Human Computer Interaction Laboratory, University of Maryland, College Park, Maryland, USA*, 2004.

[20] Y. Tian, W. Liu, R. Xiao, F. Wen, and X. Tang. A face annotation framework with partial clustering and interactive labeling. In *Proc. CVPR*, 2007.

[21] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[22] R. Wang, S. Shan, X. Chen, and W. Gao. Manifold-Manifold Distance with application to face recognition based on image set. In *Proc. CVPR*, 2008.

[23] L. Zhang, L. Chen, M. Li, and H. Zhang. Automated annotation of human faces in family albums. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 355–358, 2003.

[24] M. Zhao, Y. Teo, S. Liu, T. Chua, and R. Jain. Automatic person annotation of family photo album. *Image and Video Retrieval*, pages 163–172, 2006.