Framework for Active Clustering with Ensembles

Jeremiah R. Barr*, *Student Member, IEEE,* Kevin W. Bowyer, *Fellow, IEEE,* and Patrick J. Flynn, *Fellow, IEEE*

***Abstract*—Clustering approaches can alleviate the burden of tagging face identities in *ad hoc* video and image collections. We introduce a novel semi-supervised framework for clustering face patterns into identity groups using minimal human interaction. This technique combines concepts from ensemble clustering and active learning to improve clustering accuracy. The framework actively queries the user for a soft link constraint between each pair of neighboring faces that are ambiguously matched according to the ensemble. We demonstrate the efficacy of our approach with the broadest evaluation of active face clustering algorithms to date. Our evaluations focus on data that is appropriate for human-in-the-loop face recognition, including blurry point-and-shoot videos, images of women seen before and after the application of makeup, and photographs of twins. The results indicate that ensemble-based constrained clustering algorithms are generally more robust to noise than alternative approaches. These methods continue to improve the quality clustering as the set of constraints expands, even though the absolute number of constraint errors increases. Finally, we show that the proposed clustering algorithm is more accurate and parsimonious than the current state-of-the-art.**

I. Introduction

YOUTUBE ingests 100 hours of video footage every minute [1]. News organizations collect an equally daunting mass of imagery every day. Often in social media and consumer applications, the identities of the observed individuals can aid efforts to organize the data [2]. Likewise, the identities of people recorded by surveillance camera networks are valuable to the intelligence and forensics communities [3]. A human operator cannot always identify all of the people in such visual feeds using a database of known persons. Manually tagging all of the images of the same person poses a tremendous burden and is only practical for extremely high-profile cases.

We propose to address this problem through a combination of face matching tools and clustering techniques. Face matching provides a measure of similarity between a pair of faces. Clustering techniques facilitate exploratory analyses by grouping similar patterns into homogenous clusters [4]. Our primary objective is to partition the set of observed faces into identity clusters. Each identity cluster should correspond to a single individual and vice versa. This task is complicated by a number of challenges: Face matching remains an open problem in scenarios involving uncontrolled variations in pose, illumination, expression and other nuisance factors. And such scenarios are of course the ones of most interest. We thus argue that a minimal amount of information about which faces match should be interactively solicited from the user to guide the system to an accurate solution.

J. R. Barr, K. W. Bowyer and P. J. Flynn are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46656 USA

E-mail: jbarr1,kwb,flynn@nd.edu

EDICs: SUR-FORE, BIO-APPS, BIO-MODA-FAC, SUR-OTHS

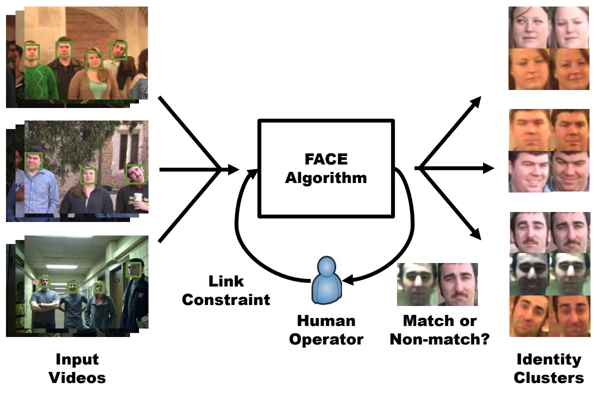


Fig. 1. The FACE algorithm forms identity clusters from faces observed in an *ad hoc* collection of images or, in this case, videos.

Previous work suggests that the face recognition capabilities of humans and computers are complementary. Face matching algorithms have advanced to the point where they can rival or surpass the recognition accuracy of people from the general public [5]. On the other hand, point-and-shoot cameras have been used to record a substantial proportion of the images and videos seen on social media websites. Face recognition algorithms often perform poorly on such data because motion blur and focus problems affect facial appearance [6]. Humans are surprisingly good at recognizing faces under degraded viewing conditions [7]. Humans also recognize faces covered by moderate amounts of makeup accurately [8], but commercial algorithms are relatively poor at matching pairs of faces in which one or both of the faces are laden with makeup [9]. Similarly, humans can outperform a commercial face matching algorithm on images of identical twins, especially when sufficient time is available for a careful analysis of skin marks [10]. Fusing the facial similarity estimates of computers and humans can lead to near-perfect matching performance [11].

This work introduces a general semi-supervised clustering scheme, the *Framework for Active Clustering with Ensembles (FACE)*, with an emphasis on clustering face patterns using minimal human interaction. The FACE technique combines concepts from ensemble clustering and active learning to improve clustering accuracy. In this application of ensemble clustering, two face patterns are assigned to the same cluster if the consensus of a diverse set of clusterings indicates they match. Diversity is achieved by integrating clustering algorithms with different failure modes, and by varying the parameterizations of the algorithms. The consensus of the ensemble is quantified by the proportion of clusterings that decide a pair of face patterns share a common identity. As the clusters are formed, FACE solicits human feedback in the form of a soft must-link or cannot-link constraint between each pair of neighboring face patterns with an ambiguous match consensus. The additional contributions of this paper are two-fold:

- We extend prior results from Davidson *et al.* [12] by showing that a clustering algorithm cannot efficiently apply soft constraints to converge on a perfectly accurate partitioning with a bounded number of clusters. We nevertheless exploit previous work on consensus clustering [13] to demonstrate that the ensemble-driven FACE algorithm can converge to the correct result.
- We present the broadest evaluation of constrained face clustering algorithms to date. Unlike earlier work from this area [14]–[16], we consider the effects of constraint noise, which will likely occur when a human user is tasked with matching unfamiliar faces [5]. We compare the FACE algorithm to the state-of-the-art in active face clustering [16], with a focus on applications that are amenable to human intervention. The results indicate that the FACE approach forms more accurate identity clusters than other approaches and is highly robust to constraint noise.

In the sections below, we review the related literature (Section II) and describe the FACE algorithm (Section III). The experimental results (Section IV) are then presented, followed by our conclusions (Section V).

II. Related Work

The application of clustering approaches to face data has received a significant amount of attention throughout the last decade. For instance, clustering methods have been used for organizing face images or sequences into identity groups. Recently developed algorithms have exploited pairwise constraints from external sources outside the (dis)similarity metric with the objective of improving accuracy. Constraints are generally gathered passively before clustering or obtained through active learning.

Recent work on clustering faces into identity groups includes that of Tao and Tan [17], who handled difficult pose variations by comparing face sequences in terms of frames with similar poses. Clustering is then performed with the affinity propagation algorithm. They present strong results from clustering experiments involving segments from three movies. A similar framework for automatically labeling the faces of characters in TV or movie videos was proposed by Sivic *et al.* [18]. Yu *et al.* [19] applied a parameter-free clustering algorithm to detect social groups and leaders in crowds recorded by a surveillance camera network.

Identity clusters can be exploited in people-counting applications. A meeting understanding system that clusters face sequences and then counts meeting attendees was introduced by Vallespi *et al.* [20]. This system consists of an adaptive subspace face tracker and a temporal subspace clustering method based on the normalized cuts clustering algorithm. Prince and Elder [21] combined clustering with a Bayesian approach to count the number of different people who appear in a collection of face images. The parameters of a generative probabilistic model describing the face manifold are learned during training. This model enables the computation of the posterior probability over possible clusterings, so that Bayesian model selection can be applied to compare partitionings of varying sizes.

Although the algorithms presented in [17]–[21] address key challenges such as head pose, illumination or facial expression, they do not provide a means to exploit external feedback. Such feedback is valuable when nuisance factors overwhelm the face matching algorithm. For instance, Wu *et al.* [14] employed pairwise must- and cannot-link constraints derived from the simple observations that all images from the same sequence must match and images of faces from co-occurring sequences cannot match. They incorporated these spatiotemporal constraints into a probabilistic clustering scheme based on hidden Markov random fields, thus enabling the incorporation of side information from outside the employed facial dissimilarity metric. The pairwise constraints improved the performance of the clustering algorithm.

Ideally, a sequence of face images would cover the entire interval of frames in which a given person appeared. Long, continuous sequences of faces support the creation of more spatiotemporal constraints than disconnected sub-sequences. Abrupt changes in appearance, occlusions and camera movements can cause tracking to fail and thereby prevent the collection of continuous face sequences in practice. Wu *et al.* [15] later extended the constrained clustering approach from [14] to simultaneously link sub-sequences and group faces based on appearance and motion cues.

Unlike the FACE algorithm, all of the constraints for these clustering schemes [14], [15] are spatiotemporal in nature and must be gathered prior to execution. The clustering algorithms do not actively select the pairs of faces for which constraints are retrieved. Some of the constraints may provide redundant match information that is strongly correlated with the outputs of the dissimilarity metric.

Cinbis *et al.* [22] drew upon the spatiotemporal relationships between tracked faces to compute actor-specific dissimilarity metrics via logistic discriminant metric learning. Throughout a series of experiments conducted on episodes of *Buffy the Vampire Slayer*, the clusterings computed with the actor-specific metrics were shown to be more accurate than partitionings obtained with generic metrics. The information provided by the clustering algorithm was not capitalized upon to improve performance, just as in [14], [15].

Similar to active learning in the context of supervised classification, active clustering techniques solicit a minimal number of constraints from a user with the aim of creating a higher quality clustering than can be obtained without user feedback [23]. A wide variety of active clustering algorithms have been devised, including pairwise constraint k-means (PCKmeans) [24] and active spectral clustering [25].

PCKMeans [24] queries for pairwise constraints in two initial phases, *Explore* and *Consolidate*. The Explore phase performs a farthest-first traversal of the data patterns while seeding an initial set of clusters. Actively queried link constraints determine the cluster to which each point is assigned, until k clusters are initialized or the query budget is exhausted. The Consolidate phase uses the initial clusters to group more patterns. By querying for $k-1$ constraints, the true cluster of a given pattern can be found. The set of must-link constraints is augmented by taking the transitive closure of the must-link constraints, and adding entailed cannot-link constraints.

PCKMeans suffers from two drawbacks. First, active learning takes place before clustering, so the query selections are not based on the confidence of the clustering algorithm. The Explore phase is also computationally intensive when the number of clusters is a large proportion of the number of patterns, which is often the case for face clustering tasks.

Active spectral clustering [25] iteratively takes a graph over the input patterns and the constraints that have already been solicited, and produces a cut of the graph which satisfies the constraints. Additional pairwise constraints are collected actively according to a selection procedure that maximally reduces the expected error between the clusters and the true class distributions. This algorithm is only suitable for data with two clusters. However, techniques for recovering an arbitrary number of clusters are briefly described in [25].

Neither [24] nor [25] focus on actively clustering face data. Biswas *et al.* [16] made one of the first ventures into this domain with the Active Hierarchical Agglomerative Constrained Clustering (AHACC) approach. The idea is to maximize the expected gain of the solicited link constraints throughout a sequence of iterations. The clustering algorithm is a simple variation of single-link hierarchical agglomerative clustering that satisfies the link constraints. Query selection is performed by simulating the effects of obtaining either a must- or a cannot-link constraint for all of the unconstrained pattern pairs. The pair that induces the greatest change in the clustering for a particular time step is chosen as the next query.

AHACC is limited because the constraints are assumed to be noise-free and self-consistent. This approach is also inefficient. Simulating the effects of a must- and a cannot-link for each pair of unconstrained patterns involves running the clustering algorithm a quadratic number of times in the number of patterns. These simulations occur once for each query. Further, the number of clusters must be specified prior to clustering. Resorting to optimization of the clustering with respect to the number of clusters will only amplify these inefficiencies. These issues restrict the applicability of AHACC.

Barr *et al.* [26] also investigated the application of active clustering to facial imagery. The authors proposed the Active Clustering with Ensembles (ACE) algorithm, a method that operates on an ensemble of partitionings to accurately cluster face patterns and select query face pairs. ACE creates the clustering ensemble by applying SCOP-KMEANS [27] on multiple representations of the data. If the ensemble yields an unclear consensus as to whether two faces reside within the same cluster, a link constraint is solicited from the user.

ACE and FACE are similar insofar as they leverage ensembles for the clustering and active learning tasks. These approaches nevertheless differ across a number of key dimensions. One salient difference is that ACE only uses a single clustering algorithm to create the ensemble, SCOP-KMEANS. FACE relies on algorithms with distinct biases to increase diversity within the ensemble. These algorithms include an approach that we introduce here, Soft Hierarchical Agglomerative Constrained Clustering, and a k-means based approach, LCVQE [32]. Moreover, FACE varies parameters such as the number of clusters in the ensemble members, whereas ACE generates ensemble members using the same parameters. Another point of distinction is that the ACE algorithm selects query pairs solely in terms of the ensemble consensus without regard to the distances between patterns. FACE accounts for the ensemble consensus yet draws from neighboring pairs that can further define the boundaries between clusters. Lastly, the analyses in [26] emphasize performance comparisons between actively and randomly selected queries solicited on a single data set. This work presents a more meaningful empirical evaluation involving multiple active clustering algorithms operating on data that are conducive to human feedback. We also clarify the theoretical underpinnings for the correctness of ensemble-based constrained clustering algorithms such as ACE and FACE.

III. Framework for Active Clustering with Ensembles

Our main focus is on a framework for clustering face observations according to their identities using feedback from a human user. The first step is to detect or track the faces to determine when and where they appear. This process results in a collection of cropped face images or sequences $\boldsymbol{F} = \{f_1, f_2, \ldots, f_{n_f}\}$. The ultimate goal is to attain an arbitrary identity labeling for the faces $\boldsymbol{L} : \boldsymbol{F} \rightarrow \mathbb{Z}$ that indicates which face observations correspond to the same person.

Clustering algorithms can either be instance- or metric-based. Whereas instance-based approaches operate on patterns represented by vectors, matrices or tensors, the metric-based approaches only require a matrix of (dis)similarity values for the pairs of patterns. A distance matrix can be readily computed from a collection of patterns using the Euclidean distance or more sophisticated metrics. We consequently assume that a matrix $\boldsymbol{X} \in \mathbb{R}^{n_f \times d}$ of d-dimensional face patterns is available prior to clustering. We sometimes interchange $x_i \in \boldsymbol{X}$ for f_i for simplicity of notation, *e.g.* we occasionally refer to $\boldsymbol{L}(x_i)$ instead of $\boldsymbol{L}(f_i)$. With this formulation, faces can be clustered with both instance-based and metric-based clustering algorithms.

The clustering inputs consist of $\boldsymbol{X}$ along with a set of must-link constraints $\mathcal{M} = \{m_{ij} : m_{ij} = (f_i, f_j)\}$ and a set of cannot-link constraints $\mathcal{C} = \{c_{lk} : c_{lk} = (f_l, f_k)\}$. When an external source of information suggests that a pair of faces should be connected by a must- or cannot-link constraint, they likely represent the same identity or different identities, respectively. This side information can enable the clustering algorithm to compensate for errors that occur if a (dis)similarity score suggests different faces match or observations of the same person do not match.

The set of link constraints $\mathcal{L} = \mathcal{M} \cup \mathcal{C}$ is iteratively enhanced by querying an oracle. The oracle may provide either noise-free or noisy responses. The constraints in $\mathcal{L}$ are *soft* in the sense that an algorithm is not required to satisfy all of them. As a result, noisy or inconsistent constraints can be ignored. We denote the set of violated must-link constraints with $\mathcal{V}_{\mathcal{M}} = \{m_{ij} \in \mathcal{M} : \boldsymbol{L}(f_i) \neq \boldsymbol{L}(f_j)\}$. Likewise, the violated cannot-link constraints $\mathcal{V}_{\mathcal{C}} = \{c_{lk} \in \mathcal{C} : \boldsymbol{L}(f_l) = \boldsymbol{L}(f_k)\}$.

Given the clustering inputs, we desire the labeling $\boldsymbol{L}$. The labels define identity cluster assignments: cluster $C_a = \{f_i : \boldsymbol{L}(f_i) = a\}$ and clustering $\boldsymbol{C} = \{C_a : a = 1 \ldots k\}$. The number of clusters k presumably is not known in advance. Instead, k is assumed to lie within $\{k_{\min}, \ldots, k_{\max}\}$, where $1 \leq k_{\min} \leq k_{\max} \leq n_f$. The labeling $\boldsymbol{L}$ is created through ensemble clustering with the constraints.

A. Clustering Faces

The FACE method combines ideas from ensemble clustering and active learning. In ensemble clustering, multiple partitionings are created with distinct clustering algorithms, through various algorithmic parameterizations or from different views of the data [13], [28]. A high quality clustering of the data can result from a consensus vote on which pairs of patterns belong to the same cluster. Moreover, an ensemble can be used to recover arbitrarily shaped clusters.

In supervised active learning, query-by-committee [29] is a well-known approach for creating queries for user labeling. This technique selects the training instance for which an ensemble of models produces the weakest consensus. We extend this idea to the semi-supervised clustering domain using partitioning ensembles.

The FACE algorithm iteratively clusters faces into identity-specific groups using a diverse ensemble of clusterings computed with distinct algorithms and parameterizations. Clusters are formed from pairs of faces that match according to the consensus of the ensemble.

Queries are generated from neighboring patterns for which the match consensus is unclear. The answers to these queries can potentially mitigate errors committed by the face matching algorithm. In this way, the FACE algorithm learns a similarity metric based on the consensus of the ensemble as it compensates for the limitations of automatic face recognition technology. This similarity metric is subsequently employed to produce a consensus clustering $\boldsymbol{C}_t$. Clustering completes after a user-defined query budget has been exhausted.

1) Clustering Ensemble: Ensemble clustering methods depend on a diverse collection of base partitionings with comparable quality to find a higher quality result than the best clustering in the ensemble [30], [31]. Diversity can be achieved in a number of different ways. In this work, we promote diversity using two base clustering algorithms with distinct biases, and apply a range of parameterizations to the base clustering algorithms.

The learning task is complicated by the fact that the pairwise constraints may have noise. Errors can be introduced by mistakes in human judgement. In contrast to hard constraints, which an algorithm must satisfy in order to attain a valid solution, soft constraints can be violated if they are inconsistent or otherwise reduce the optimality of the clustering. We therefore require base clustering algorithms that can accept soft constraints.

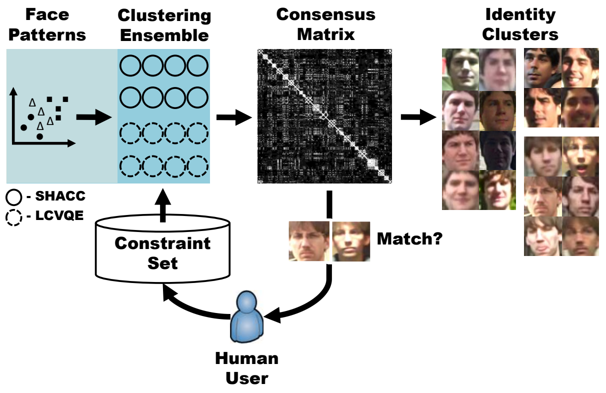


Fig. 2. An array of base clustering algorithm instances creates an ensemble of clusterings based on the input patterns, similarities and link constraints. The consensus of the ensemble about which faces match governs both query selection and the formation of the output clustering. Pairs of neighboring faces for which the consensus is unclear are submitted as queries to a human user.

One such technique, the Soft Hierarchical Agglomerative Clustering with Constraints (SHACC) algorithm, is a soft-constrained extension of the clustering technique at the core of the Active HACC algorithm (AHACC) [16]. We complement SHACC with the Linear Constrained Vector Quantization Error (LCVQE) [32] algorithm. LCVQE extends the classic k-means approach by incorporating information into the objective function and cluster updating procedure.

Similar to AHACC, the SHACC technique attempts to find a Minimum Spanning Forest (MSF) that best satisfies the constraints using a variant of Kruskal's algorithm [33]. The MSF is extracted from a graph $G = (V, E)$ with a set of vertices V corresponding to the patterns and a collection of edges E connecting each pair of vertices. Every edge is weighted with the *constrained distance* between its patterns. SHACC computes the MSF consisting of the minimum spanning trees for k connected components of G. The k connected components are treated as clusters in the AHACC and SHACC schemes. SHACC differs from AHACC insofar as it can alleviate the impact of noisy constraints as it finds the connected components.

The constrained distance effectively pulls the points connected by must-link constraints together and pushes patterns with cannot-link constraints apart. The link relationship between patterns x_i and x_j is quantified with σ_{ij}. If the patterns are connected by a cannot-link constraint, $\sigma_{ij} = \frac{1}{2}$; $\sigma_{ij} = 2$ if they share a must-link constraint; otherwise, $\sigma_{ij} = 1$. The constrained distance is given by

$$d_c(x_i, x_j) = d(x_i, x_j)^{\sigma_{ij}}, \qquad (1)$$

where $d(x_i, x_j)$ is the $\min\text{-}\max$ normalized Euclidean distance between the patterns.

Just as in Kruskal's algorithm [33], the MSF F is initially set to $(V, \emptyset)$. All of the weighted edges are ordered in a priority queue Q_E. While Q_E still contains more edges and F contains more than k connected components, SHACC iteratively removes the minimum weight edge from Q_E. The edge is added to F if it lies between vertices in separate connected components that share more must-link constraints than cannot-link constraints. This step essentially merges previously disconnected clusters as long as the constraints provide evidence in support of the merge.

The edge restrictions can potentially prevent the formation of k components. A similar problem, referred to as a *dead-end*, can arise for hard constrained algorithms such as AHACC [34]. A dead-end occurs when additional cluster merges will result in a constraint violation, even though a smaller clustering that satisfies the constraints exists. The SHACC algorithm can retrieve a specified number of clusters because it operates on soft constraints. If there are $k_c > k$ components by the time Q_E is empty, F is filled with the $k_c - k$ edges connecting the components that would have the fewest constraint violations between their patterns. This step ensures that there are k components at the cost of increasing the number of violated constraints. Clusters are then created from the k connected components.

The SHACC method incorporates measures for addressing noise at the pattern and clusters levels. The constrained distance provides robustness to exceptionally harmful types of noise by changing the ordering of how components are connected. Noisy must-link constraints placed between widely separated patterns will cause a hard constrained algorithm such as AHACC to assign patterns from incomparable classes to the same cluster. The clusters that lie between the constrained pairs may subsequently be merged in error. When erroneous cannot-link constraints connect patterns located within the center of a cluster, a hard constrained algorithm will incorrectly split the cluster. The constrained distance enables SHACC to prioritize the cluster assignments of neighboring points over distant pairs of spuriously must-linked patterns. Likewise, neighboring patterns with incorrect cannot-link constraints can still be merged into the same cluster before SHACC terminates.

Comparing the numbers of must-link and cannot-link constraints connecting merge candidates can prevent inconsistent constraints from impacting accuracy. Correct constraints should occur more frequently than incorrect constraints if we expect to gain an advantage from human-in-the-loop processing. If this condition holds, the correct constraints will tend to provide evidence in support of accurate merging decisions while preventing problematic merges.

In contrast, LCVQE [32] builds on k-means by shifting cluster centers to accommodate violated constraints. The original k-means algorithm begins by selecting k cluster centers $\{\mu_a \text{ for } a = 1 \dots k\}$. Every pattern x_i is assigned to a cluster with the aim of minimizing the overall objective function, the *distortion*:

$$\sum_{a=1}^{k} \sum_{x_i \in C_a} \|x_i - \mu_a\|^2. \tag{2}$$

The centers are updated after the patterns are assigned to the clusters:

$$\mu_a = \frac{1}{|C_a|} \sum_{x_i \in C_a} x_i. \tag{3}$$

The assignment and update steps alternate iteratively until a stopping criterion is satisfied. For example, we may iterate T times. The final clustering constitutes the output of the algorithm.

LCVQE proceeds similarly, but constraint violations drive the cluster centers in directions that can increase the fidelity of the clustering. In the case of a violated must-link constraint, we have assigned patterns x_i and x_j to different clusters, C_a and C_b, yet they should belong to the same cluster. The cluster center updates can shift μ_a towards x_j and/or move μ_b towards x_i to improve the clustering. A cannot-link constraint violation entails the assignment of two faces to the same cluster C_a when they should be in separate clusters. This error is mitigated by finding the pattern $x \in \{x_i, x_j\}$ that is farthest from μ_a. The closest center to x besides μ_a is then moved towards x.

The LCVQE objective function measures the cost of unsatisfied constraints in terms of the distances between patterns. The cost of violating a must-link constraint is the total of the distances between x_i and μ_b and between x_j and μ_a. For an unsatisfied cannot-link constraint, the pattern that is the farthest from the center is determined. The distance between this pattern and its second closest cluster center serves as the violation cost. The objective function incorporates the overall distortion in addition to the violation costs.

Formally, the LCVQE objective function is $\sum_{a=1}^{k} J_a$, where

$$\begin{aligned} J_a = {} & \frac{1}{2} \sum_{x_i \in C_a} \|x_i - \mu_a\|^2 \\ & + \frac{1}{2} \sum_{m_{ij} \in \mathcal{V}_\mathcal{M}, \boldsymbol{L}_t(x_i)=a} \|x_j - \mu_a\|^2 \\ & + \frac{1}{2} \sum_{m_{ij} \in \mathcal{V}_\mathcal{M}, \boldsymbol{L}_t(x_j)=a} \|x_i - \mu_a\|^2 \\ & + \frac{1}{2} \sum_{c_{ij} \in \mathcal{V}_\mathcal{C}, N(c_{ij})=a} \|F_{\boldsymbol{L}_t(x_i)}(c_{ij}) - \mu_a\|^2, \end{aligned} \tag{4}$$

$$F_a(c_{ij}) = \begin{cases} x_i & \text{if } \|x_i - \mu_a\|^2 > \|x_j - \mu_a\|^2 \\ x_j & \text{otherwise, and} \end{cases} \tag{5}$$

$$N(c_{ij}) = \underset{a=1\dots k, a \neq \boldsymbol{L}_t(x_i)}{\operatorname{argmin}} \|F_{\boldsymbol{L}_t(x_i)}(c_{ij}) - \mu_a\|^2. \tag{6}$$

$\boldsymbol{L}_t$ is the cluster labeling at iteration t; $F_a(c_{ij})$ returns the pattern from c_{ij} that is furthest from center μ_a; and $N(c_{ij})$ outputs the label of the nearest cluster to $F_{\boldsymbol{L}_t(x_i)}(c_{ij})$ besides $\boldsymbol{L}_t(x_i)$.

The SHACC and LCVQE clustering algorithms both offer some level of robustness to constraint noise. They differ in the manners in which they process patterns. Whereas SHACC must perform a sequence of $n_f - k$ cluster merges to produce k clusters, LCVQE only needs to iterate once to create a fixed number of clusters. LCVQE can therefore update an existing clustering in response to new constraints. Conversely, LCVQE computes the distances between patterns and cluster centers every iteration. Its runtime is dependent on the dimensionality of the patterns as result. SHACC does not suffer from this drawback since it can employ a pre-computed distance matrix. SHACC can also continue merging a set of previously formed connected components when a collection of clusterings with difference sizes is required.

SHACC and LCVQE also have distinct biases with respect to the geometric characteristics of the clusters they retrieve. SHACC often produces elongated, chain-like clusters since it greedily merges nearby clusters without regard to the global effects on the clustering. As a generalization of k-means, LCVQE will tend to form compact, spherical clusters. These biases make each approach appropriate for different data distributions and cause the algorithms to form distinct clusterings from the same data.

2) Initialization: We bolster additional diversity within the ensemble through variations in the number of clusters, k, and by randomizing cluster initialization. FACE instantiates $n_b > 1$ base clustering algorithms at the outset. The algorithms are specified by the set $\boldsymbol{\beta} \subseteq \{\text{SHACC}, \text{LCVQE}\}$. The instances are divided evenly amongst the clustering algorithms listed in $\boldsymbol{\beta}$ into collections of $n_k = \frac{n_b}{|\boldsymbol{\beta}|}$ members. Additional ensemble diversity is induced by varying the number of clusters for the instances within $\{k_{\min}, \dots, k_{\max}\}$. Instance b_i is set to create $k_i = i\lfloor \frac{k_{\max}-k_{\min}}{n_k+1} \rfloor + k_{\min}$ clusters, where $i = 1 \dots n_k$.[1] Further, every LVQE instance is initialized with a different set of cluster centers selected randomly from the input data.

Additional initialization steps are performed to accelerate subsequent computations. The SHACC instances depend on a distance matrix $\boldsymbol{D}$. Entry d_{ij} is set to $d(x_i, x_j)$, *i.e.* the $\min$-$\max$ normalized Euclidean distance between x_i and x_j.

Our active learning method submits pairs of neighboring faces as queries to the user. The determination of which pairs of faces constitute mutual neighbors can be costly from a computational standpoint, especially since FACE submits multiple queries over a series of T iterations. We alleviate this issue by pre-computing an $n_f \times n_f - 1$ ranking matrix $\boldsymbol{R}$ to aid neighborhood lookups. For each pattern x_i, we rank the other patterns in increasing order of their respective distances to x_i. Rank 1 corresponds to the closest pattern, whereas rank $n_f - 1$ is associated with the farthest pattern. Component $r_{il} = j$ if pattern x_j has rank l for x_i.

3) Ensemble Clustering: The FACE algorithm invokes the base clustering algorithms over the course of T iterations. During iteration t, instances of LCVQE pass through the assignment and update stages one time. Instances of SHACC re-initialize and perform a complete sequence of merges. The partitionings produced by the base clustering algorithms are combined via an $n_f \times n_f$ consensus matrix $\boldsymbol{P}_t$. Element p_{ij} measures the proportion of the ensemble with faces f_i and f_j assigned to the same cluster. There is not a clear consensus amongst the clusterings when $p_{ij} = 0.5$, so the faces may or may not share a common identity. The faces should be assigned to the same cluster if p_{ij} is near one, and they should be assigned to different clusters if p_{ij} is near zero. Hence, $\boldsymbol{P}_t$ provides useful information for selecting query pairs and fusing the ensemble clusterings.

A single aggregated clustering $\boldsymbol{C}_t$ is created by performing classical single-link Hierarchical Agglomerative Clustering (HAC) on $\boldsymbol{P}_t$. This algorithm takes a similarity matrix such

[1]A single instance of SHACC can merge clusters from a partitioning with k_i components to form a clustering with k_{i-1} components, effectively acting as multiple instances for our purposes. We simplify the discussion by referring to SHACC as though it has multiple instances, except where otherwise noted.

```
Input:
  • Pattern matrix $\boldsymbol{X}$;
  • the set of base clustering algorithms $\boldsymbol{\beta}$;
  • the number of base clustering algorithm instances $n_b$;
  • the bounds on the number of clusters $k_{\min}$ and $k_{\max}$;
  • the number of queries per iteration $n_q$; and
  • the number of iterations $T$.
Output: Face labelling $\boldsymbol{L}_T$ with $n_f$ cluster assignments.
 1: $B = \emptyset$
 2: $n_k = \frac{n_b}{|\boldsymbol{\beta}|}$
 3: for all $\beta \in \boldsymbol{\beta}$ do
 4:     for $i = 1 \dots n_k$ do
 5:         $k = i\lfloor \frac{k_{\max}-k_{\min}}{n_k+1} \rfloor + k_{\min}$
 6:         Initialize $\beta$ instance $b$ to create $k$ clusters
 7:         Let $\boldsymbol{L}_b$ express the cluster assignments of $b$
 8:         Add $b$ to $B$
 9: Compute distance matrix $\boldsymbol{D}$
10: Compute ranking matrix $\boldsymbol{R}$
11: for $t = 1 \dots T$ do
12:     Let the consensus matrix $\boldsymbol{P}_t = 0$
13:     for all $b \in B$ do
14:         $\boldsymbol{L}_b = b(\boldsymbol{X}, \boldsymbol{D}, \mathcal{L}, \boldsymbol{L}_b)$
15:         for all $f_i$ and $f_j$ with $\boldsymbol{L}_b(f_i) = \boldsymbol{L}_b(f_j)$ do
16:             $p_{ij} = p_{ij} + \frac{1}{|B|}$
17:     Get current labels $\boldsymbol{L}_t$ and clusters $\boldsymbol{C}_t$ using HAC
18:     if $t < T$ then
19:         Let $Q_u$ be the set of unconstrained face pairs
20:         $Q = \text{GET-QUERY}(Q_u, \boldsymbol{P}_t, \boldsymbol{L}_t, \boldsymbol{C}_t, \boldsymbol{R}, n_q)$
21:         Update constraint set $\mathcal{L}$ with responses $Q$
22: return $\boldsymbol{L_T}$
```

Fig. 3. The FACE algorithm

as $\boldsymbol{P}_t$ as input and outputs a tree D called a dendrogram. The leaves of D represent singleton clusters consisting of individual faces, whereas the root is associated with a single cluster that contains all of the faces. Every level within D corresponds to one or more pairs of merged clusters. Clusters that are merged together near the leaves are more likely to represent the same person than clusters that are merged together near the root.

A single clustering $\boldsymbol{C}'_l$ can be retrieved by cutting D at level l. A cluster is formed from the leaves of each tree in the forest resulting from the cut. We choose a clustering under the assumption that $k_{\min}$ and $k_{\max}$ are rough estimates. If $k_{\max}$ is far larger than the true number of clusters in the data, a partitioning of size $k_{\max}$ will arbitrarily divide intrinsic clusters. A partitioning of size $k_{\min}$ will merge intrinsic clusters together if $k_{\min}$ is too small. Another complication is that comparing the qualities of clusterings can be computationally demanding. We consequently treat clusterings with k near either of the extremes as outliers, and take the simple approach of choosing the clustering of median complexity:

$$\boldsymbol{C}_t = \underset{\boldsymbol{C}'_l \in D : k_{\min} \le |\boldsymbol{C}'_l| \le k_{\max}}{\text{argmedian}} |\boldsymbol{C}'_l| \tag{7}$$

Input:

- The set of unconstrained face pairs Q_u;
- consensus value matrix $\boldsymbol{P}$;
- face labelling $\boldsymbol{L}$;
- clustering $\boldsymbol{C}$;
- ranking matrix $\boldsymbol{R}$; and
- the number of queries per iteration n_q.

Output: List Q of face pairs selected as queries

```
1:  Let $Q_b$ be an empty set of ambiguity value bins
2:  for all $(f_i, f_j) \in Q_u$ do
3:      Insert $(f_i, f_j)$ into $Q_b(|p_{ij} - 0.5|)$
4:  Let $A$ be the ambiguity values for $Q_b$
5:  Let the mutual neighbor matrix $\boldsymbol{K} = 0$
6:  for $i = 1 \dots n_f$ do
7:      Let $\boldsymbol{N}$ be a $|\boldsymbol{C}| \times |\boldsymbol{C}|$ matrix of zeros
8:      Let $a = \boldsymbol{L}(f_i)$
9:      for $l = 1 \dots n_f - 1$ do
10:         Let $j = r_{il}$
11:         Let $b = \boldsymbol{L}(f_j)$
12:         $n_{ab} = n_{ab} + 1$
13:         if $n_{ab} \leq \sqrt{|C_a \cup C_b|}$ then
14:             $k_{ij} = 1$
15: for $i = 1 \dots n_f$ do
16:     for $j = i + 1 \dots n_f$ do
17:         $k_{ij} = k_{ji} = \left\lfloor \frac{k_{ij}+k_{ji}}{2} \right\rfloor$
18: for all $\alpha \in A$ do
19:     for all $(f_i, f_j) \in Q_b(\alpha)$ do
20:         if $k_{ij} = 1$ then
21:             Insert $(f_i, f_j)$ into $Q$
22:             if $|Q| = n_q$ then
23:                 return $Q$
24: return $Q$
```

Fig. 4. The GET-QUERY algorithm.

In the event that the number of clusters k is known prior to execution, the dendrogram is simply cut at the level with k clusters.[2] The final face labeling is produced from $\boldsymbol{C}_T$.

4) Query Selection: Useful queries should complement the information from the facial dissimilarity metric. We require human feedback when a pair of faces is ambiguously matched according to the ensemble. The constraints should also pertain to neighboring points that can define cluster boundaries. As a result, the GET-QUERY algorithm chooses n_q pairs of neighboring faces that will reduce the overall uncertainty about the cluster assignments. Figure 5 motivates this selection procedure.

The match/non-match decision for a pair of faces can be modelled as a Bernoulli trial. Entry p_{ij} from the consensus matrix $\boldsymbol{P}_t$ can be interpreted as an estimate of the probability that faces f_i and f_j have the same identity, and $1 - p_{ij}$ can be treated as an estimate of the probability that the faces don't match. The uncertainty regarding the match/non-match decision can be measured in terms of *entropy*. The entropy of a Bernoulli trial with probability p is

$$H(p) = p \log_2(p) + (1-p)\log_2(1-p). \quad (8)$$

The maximum value of $H(p)$ is reached at $p = 0.5$, so the uncertainty about the match/non-match decision is greatest when $p_{ij} = 0.5$. The uncertainty increases as p_{ij} approaches 0.5 from the left or right. Hence, $|p_{ij} - 0.5|$ can serve as a measure of the ambiguity in the match/non-match decision.

GET-QUERY uses the ambiguity to rank candidate query pairs. Given a clustering of the faces that was constructed using the previously collected constraints, GET-QUERY bins the unconstrained face pairs according to the ambiguity in their match/non-match decisions. Every unconstrained pair of faces f_i and f_j is assigned to the bin for value $|p_{ij} - 0.5|$.

All of the pairs are then analyzed to recover pairs of faces that are *mutual neighbors*. We account for the distribution of the patterns within the feature space by defining mutual neighbors in terms of clusters: A candidate pair of faces f_i and f_j either belong to the same cluster or they belong to different clusters. In the case where both faces are assigned to cluster C_a, they are mutual neighbors if f_i is among the $\sqrt{|C_a|}$ nearest neighbors of f_j from cluster C_a, and vice versa. When f_i is assigned to cluster C_a and f_j is assigned to cluster C_b, f_i must be among the $\sqrt{|C_a| + |C_b|}$ nearest neighbors of f_j from both clusters, and vice versa.

We find mutual neighbors by scanning the rows of the ranking matrix $\boldsymbol{R}$ one-by-one. All the while, we track the number of patterns we've encountered from each pair of clusters in matrix $\boldsymbol{N}$. Element r_{il} of $\boldsymbol{R}$ provides the index j of the l^{th} closest face to f_i overall. Suppose $f_i \in C_a$ and $f_j \in C_b$, where a may or may not match b. We use

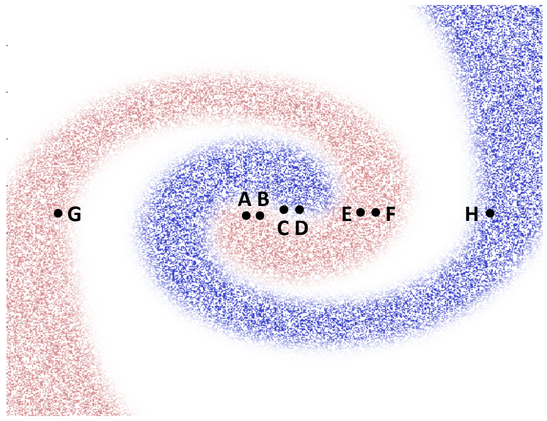


Fig. 5. Potential queries for two clusters. We argue that the constraints should connect neighboring points with ambiguous consensus values. Enforcing constraints between neighboring points A, B, C and D would further define the boundaries between the clusters. The certainty about the cluster assignments for these points would also be weak, so constraints that link these patterns would be informative. By comparison, the ensemble would likely reach the consensus that points E and F belong to the same cluster since they lie at the center of a densely populated region. Noisy constraints drawn between the interior patterns and faraway points G and H would potentially cause the clustering algorithm to merge or split the true clusters in error.

[2]There are n_b possible values for each element in $\boldsymbol{P}_t$, so there are no more than n_b dendrogram levels. Multiple cluster merges would need to be performed in the level for some consensus value if $n_b < n_f$. A level consisting of n_m merges would have 2^{n_m} associated clusterings of various sizes because we can ignore some merges. When k is provided and the dendrogram must be cut at such a level, we randomly select a clustering of size k from that level for the sake of efficiency.

n_{ab} (n_{ba}) for determining whether f_j (f_i) is no farther than the $\lfloor\sqrt{|C_a \cup C_b|}$ closest patterns to f_i (f_j) from C_a and C_b. The faces are mutual neighbors if both of these conditions hold. Matrix $\boldsymbol{K} \in \mathbb{R}^{n_f \times n_f}$ ultimately indicates which faces are mutual neighbors.

The ambiguity bins are filtered for pairs of points that are mutual neighbors, starting with the bin for the most ambiguously matched pairs and proceeding toward the bin containing the least ambiguous pairs. The querying process ceases when n_q pairs are gathered or when all of the remaining unconstrained pairs have been considered.

B. Computational Complexity

The FACE scheme (Figure 3) passes through an initialization phase (Section III-A2), and then repeatedly iterates between the clustering (Section III-A3) and query selection (Section III-A4) algorithms. We have n_k LCVQE instances. Every LCVQE instance requires $\mathcal{O}(k_i)$ steps to select the cluster centers, where $k_i = i\lfloor\frac{k_{\max}-k_{\min}}{n_k+1}\rfloor + k_{\min}$ and $i \in \{1 \ldots n_k\}$. The combined time cost associated is thus $O(n_k * (k_{\min} + k_{\max}))$. This cost is no worse than $\mathcal{O}(n_f)$ if we assume $n_k \ll n_f$ and $k_{\min}, k_{\max} = \mathcal{O}(n_f)$. On the other hand, SHACC only accepts parameters during initialization, so the SHACC contribution is negligible. The overall initialization runtime is dominated by the creation of the distance and ranking matrices, $\boldsymbol{D}$ and $\boldsymbol{R}$, which spans $\Theta(n_f^2 * d + n_f^2 * \log n_f)$ operations.

The clustering process subsumes multiple high-level stages that are repeated over the course of T iterations. The contributions of the LCVQE and SHACC instances vary significantly. A single LCVQE instance executes $\mathcal{O}(k * d * (n_f + |\mathcal{L}|))$ steps per iteration [35]. The LCVQE instances perform $\mathcal{O}(n_f * d * (n_f + |\mathcal{L}|))$ operations altogether at time t. The total LCVQE runtime can degrade to $\mathcal{O}(n_f^3 * d)$ in pathological cases where $|\mathcal{L}| = \mathcal{O}(n_f^2)$. SHACC extends Kruskal's algorithm [33], a technique that incurs an $\mathcal{O}(n_f^2 * \log n_f)$ cost that is dominated by queueing operations. A single SHACC instance can produce clusterings of the required sizes by successively merging the clusters of the largest partitioning. As a result, SHACC only needs to execute $\mathcal{O}(n_f^2 * \log n_f)$ operations every iteration. Outside of the base clustering algorithms, FACE performs $\mathcal{O}(n_b * n_f^2)$ additions while maintaining the consensus matrix $\boldsymbol{P}_t$ along with $\mathcal{O}(n_f^2)$ steps for aggregating the ensemble matrices via single-link HAC [36]. The aggregate clustering runtime is hence $\mathcal{O}(T * (n_f * d * [n_f + |\mathcal{L}|] + n_f^2 * \log n_f))$.

Query selection (Figure 4) also occurs every iteration. Binning the face pairs will take $\mathcal{O}(n_f^2)$ steps in the worst case; the mutual neighbor matrix $\boldsymbol{K}$ can be computed in $\Theta(n_f^2)$ steps; and the selected pairs can be aggregated in $\mathcal{O}(n_f^2)$ time. Query selection therefore has an $\mathcal{O}(T * n_f^2)$ time complexity. The overall computational complexity of FACE, $\mathcal{O}(T * (n_f * d * [n_f + |\mathcal{L}|] + n_f^2 * \log n_f))$, is largely attributable to the base clustering algorithms. The computational complexity increases to $\mathcal{O}(T * (n_f^3 * d))$ when the constraint set approaches its maximum size yet reduces to $\mathcal{O}(T * n_f^2 * (d + \log n_f))$ when $|\mathcal{L}| = \mathcal{O}(n_f)$.

C. Correctness

From a statistical standpoint, the correctness of a soft constrained clustering algorithm concerns its ability to converge on a clustering that satisfies as many constraints as possible. This notion of correctness is at odds with earlier analytical results introduced by Davidson *et al.* [12]. Unlike soft constraints, which we employ in the FACE algorithm, hard constraints cannot be violated. Davidson *et al.* showed that it is not computationally feasible for a clustering algorithm to determine whether a given set of hard constraints can be satisfied when k is bounded. We build on this result by demonstrating that the globally optimal value for the number of violated soft constraints cannot be determined efficiently if k is restricted to lie within $(k_{\min}, \ldots, k_{\max})$. As a result, a constrained clustering algorithm cannot efficiently apply a general set of soft constraints to converge to the best clustering.

The *hard constraint feasibility problem* can be stated as follows: For pattern set $\boldsymbol{X}$ and hard link constraints $\mathcal{L}$, does there exist a partitioning of $\boldsymbol{X}$ into k groups such that $k \in (k_{\min}, \ldots, k_{\max})$ and all constraints are satisfied [12]? Davidson *et al.* proved that the hard cannot-link constraint feasibility problem is **NP**-complete by reduction from the well-known graph coloring problem. The hard constraint feasibility problem, which can involve a mix of must- and cannot-link constraints, is **NP**-complete by extension.

The *soft constraint feasibility problem* is defined similarly. We are given a pattern set $\boldsymbol{X}$, link constraints $\mathcal{L} = \mathcal{M} \cup \mathcal{C}$, and the total number of constraints that can be violated V. We wish to determine whether $\boldsymbol{X}$ can be partitioned into $k \in (k_{\min}, \ldots, k_{\max})$ groups such that $V = |\mathcal{V}_{\mathcal{M}}| + |\mathcal{V}_{\mathcal{C}}|$. This problem is just as computationally difficult as its hard constraint counterpart.

Theorem 1. *The soft constraint feasibility problem is* ***NP****-complete.*

Consult Appendix A for the proof.

The associated *soft constraint optimization* problem of finding the global minima for V is **NP**-hard, since we would solve multiple instances of the soft constraint feasibility problem as we searched for the optima. This means that clustering with soft constraints is computationally intractable, because we cannot find the most accurate clustering with the minimum number of violations if we cannot find a single clustering with the minimum number of violations. We also can't expect to efficiently select a set of potential constraints that will result in minimal violation cost on arbitrary data. These limitations preclude the absolute correctness of SHACC, LCVQE and any other soft-constrained clustering algorithm.

Constraint violations notwithstanding, the FACE algorithm is still correct insofar as it can converge to the ideal identity clustering. The ultimate output of the FACE method is created with a constraint-agnostic algorithm, single-link HAC, that is not susceptible to feasibility issues. More importantly, Topchy [13] proved rigorously that the accuracy of a consensus clustering improves with increasing ensemble size n_b, provided each ensemble member performs better than random.

An accurate ensemble must have members with uncorrelated errors. Otherwise, there is no benefit to increasing its size.

An ensemble of infinite size will naturally have great diversity amongst its members, but we wish to achieve sufficient accuracy and robustness in realistic scenarios. The empirical analyses in the next section highlight the performance characteristics of clustering ensembles acting on carefully chosen constraints.

IV. Empirical Results

We now consider how various parameters affect performance, and how well the FACE technique compares to the current state-of-art in active face clustering. We first show the effects of varying the size of the ensemble and selecting alternative combinations of base clustering algorithms. We then evaluate the FACE framework alongside active clustering algorithms that have been previously applied to face data, including the state-of-the-art AHACC algorithm [16]. Our assessments pertain to data that are difficult for face matching algorithms yet appropriate for the unique recognition capabilities of humans.

A. Data Sets

We performed our experiments on a point-and-shoot video set, *SN-Flip*; a collection of images showing women before and after makeup is applied, *YouTube Makeup* (YMU); and the *Twins* data set. See Table I for a summary of these data sets.

SN-Flip [26] consists of 28 point-and-shoot videos of crowds interacting in distinct scenes. 19 of the videos were acquired outdoors in sunny, snowy or overcast conditions. The interocular distances of the located faces ranged from 4 to 70 pixels, with a median distance of 19 pixels. Video quality varies across the set because the sequences were acquired with two sensors. Both cameras were prone to focus issues, especially in poorly lit scenes. The quality levels of some face sequences are also affected by crowd occlusions and variations in illumination, facial expression and pose.

The YMU [9], [37] images were extracted from YouTube tutorials on applying makeup. There are four images per subject, with two taken before and two taken after makeup is applied. The makeup patterns range from light to heavy. Although all of the individuals held a frontal head pose, the illumination conditions and facial expressions differed between most pairs of images. The primary difficulty for this data is to assign the makeup covered and makeup-free images of each subject to the same cluster.

Lastly, the Twins data set [38] contains images of nearly 100 pairs of identical twins photographed at the 2009 and 2010 Twins Days Festival in Twinsburg, Ohio. The images used for the proposed experiments show frontal faces. The illumination conditions were carefully controlled. Further, the majority of the subjects held a neutral facial expression. The key challenge for this data is to avoid placing images of identical twins into the same cluster.

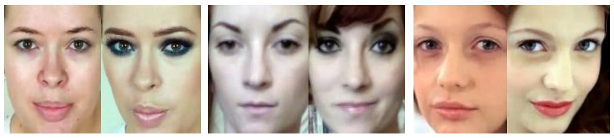
B. Data Pre-processing

A series of pre-processing steps must be completed prior to face clustering. The faces are located and cropped from the images or video frames, and then matched, yielding a similarity score matrix. The algorithms for locating and matching faces were treated as black boxes.

TABLE I
Data set summary. The number of true classes

Data set	Subjects	Size	Templates	True classes
SN Flip	190	28 videos	777	183
YMU	151	604 images	444	146
Twins	187	577 images	576	187

(a) SN-Flip - Video frames with focus and illumination issues


(b) YMU - Each pair shows the same person with and without makeup

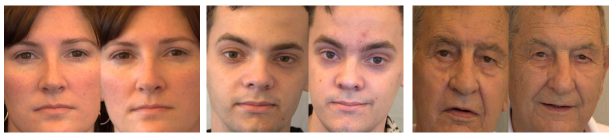

(c) Twins - Each pair shows images of identical twins

Fig. 6. Sample images from the experimental data sets.

In the case of the SN-Flip video data set, we employed a variety of algorithms from a Commercial Off-The-Shelf (COTS) face recognition system for pre-processing. The COTS face detector was used in tandem with an optical flow technique to extract sequences of face images or *face tracks*. Face tracks were formed by detecting faces in each video frame and connecting them with previously observed faces that share common features. Feature points lying within the face detection boundaries were followed across frames with the Kanade-Lucas-Tomasi tracker [39]. Since adjacent frames of video tend to be highly correlated, we sampled one frame from every half second of video prior to matching. The COTS face matcher was then employed to compute the maximum similarity score between the remaining frames of each pair of face tracks.

For each video in the SN-Flip data set, all subjects were tracked in at least one interval of frames. 817 face tracks were extracted from the video collection. Feature templates were successfully extracted from 777 of the tracks by the COTS software. The remaining tracks were not included in subsequent performance analyses because their templates were not available for matching. The removal of these tracks resulted in the reduction of the size of the subject pool from 190 to 183 people.

We ran the COTS face detector directly on the YMU and Twins images. All of the faces from these data sets were detected correctly. The COTS template extraction algorithm

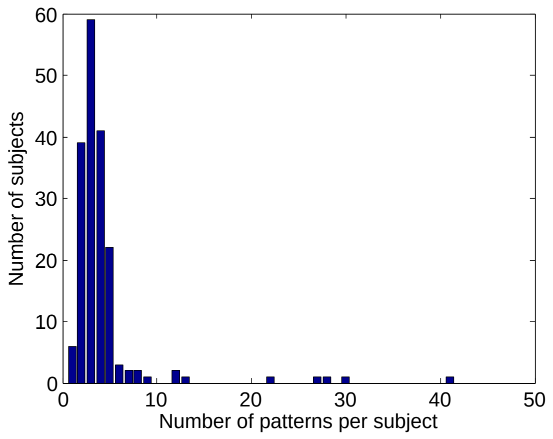


Fig. 7. SN-Flip: The distribution of the number of patterns per subject.

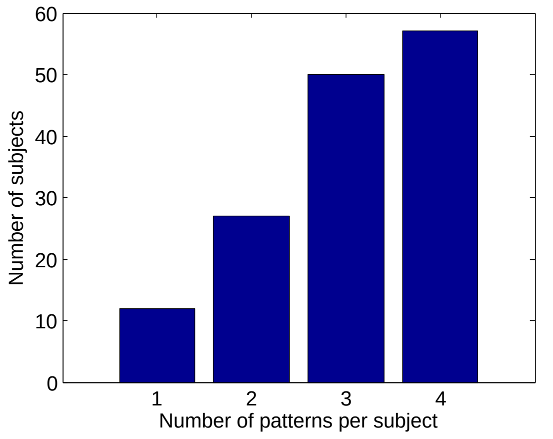


Fig. 8. YMU: The distribution of the number of patterns per subject.

was particularly sensitive to heavy makeup, as only 444 templates were created on the YMU collection. The images with templates spanned 146 subjects. A template was created for 576 of the Twins photographs. Every Twins subject had at least one template.

A pattern matrix $\boldsymbol{X}$ was computed for every data set by applying multi-dimensional scaling to the matrix of COTS similarity scores. We set $d = 0.05 * n_f$ in all cases. In this way, face images and tracks can both be represented as vectors. The pattern matrices can be retrieved from http://www3.nd.edu/~cvrl/CVRL/Data_Sets.html. To ensure fairness in the comparisons between instance- and metric-based clustering algorithms, we computed a dissimilarity matrix for the metric-based methods using the Euclidean distances between the patterns.

In general, the pre-processed data sets were not well balanced because of failures that occurred when the templates were created. Some subjects also were observed more frequently than others in the SN-Flip and Twins imagery. Figures 7 through 9 show the distributions of the number of patterns per subject. For SN-Flip, one subject holds 5% of

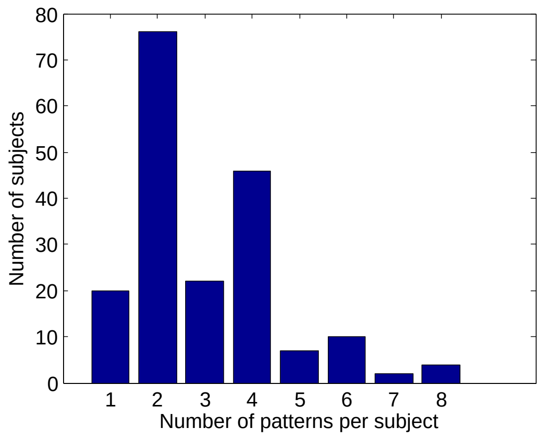


Fig. 9. Twins: The distribution of the number of patterns per subject.

the patterns and ten of the 183 classes contain more than 25% of the patterns. Six of the SN-Flip classes each only have one pattern. Ten of the 187 of the Twins classes comprise more than 15% of the patterns and 20 of the classes consist of one pattern apiece. The YMU data is better balanced since it includes four images of each subject. Nevertheless, the commercial face recognition software failed more frequently on some subjects than on others, so 12 of the 151 classes are singletons. These variations in the number of patterns per class increased the difficulty of the clustering problem.

C. *Experimental Protocols*

We treat clustering as an information retrieval task, wherein each cluster acts as a retrieval result for the patterns from some face identity class. We desire sets of clusters that have a one-to-one correspondence with the identity classes. $Precision$ and $recall$ are two common information retrieval performance metrics that, in combination, can measure the extent to which a clustering adheres to this correspondence. We can express $precision$ as the proportion of patterns from a cluster with a particular identity. Then $recall$ denotes the proportion of patterns for a given identity that were retrieved in a cluster. The *F-measure* acts as a summary statistic by aggregating the $precision$ and $recall$ measures.

We assess face clusterings in terms of a reference partitioning generated from the ground truth subject identifiers, and assume that each cluster best retrieves the patterns from a single identity class. Let $\{C_b^* \; for \; b \in 1..k^*\}$ represent the identity classes formed from the manually created identifiers for $\boldsymbol{F}$. For cluster C_a and class C_b^*, the precision and recall are defined as

$$precision(a, b) = \frac{|C_a \cap C_b^*|}{|C_a|}, \text{ and} \tag{9}$$

$$recall(a, b) = \frac{|C_a \cap C_b^*|}{|C_b^*|}. \tag{10}$$

The *F-measure* $F(a, b)$ for cluster C_a and class C_b^* is the harmonic mean of the precision and recall scores:

$$F(a,b) = 2 * \frac{precision(a,b) * recall(a,b)}{precision(a,b) + recall(a,b)}. \quad (11)$$

The overall *F-measure* for the entire clustering is expressed as

$$F = \sum_{b=1}^{k^*} \frac{|C_b^*|}{|\boldsymbol{F}|} \max_{a=1\ldots k} F(a,b). \quad (12)$$

F, $F(a,b)$, $precision(a,b)$, $recall(a,b)$ all vary with $[0,1]$, where 1 corresponds to ideal performance. Our evaluations center on the extent to which the F-measure varies as more constraints are gathered.

Some clustering algorithms automatically select the size of the partitioning when given a range of clustering sizes. Other algorithms require the user to provide a single value for the number of clusters as a parameter. Inferring the size of the clustering introduces additional challenges to the problem at hand. We employed two experimental protocols to ensure the performance comparisons are clear and fair:

1) Fixed Clustering Size (FCS) - the algorithms evaluated according to this protocol accepted a single value for the number of clusters, k. We set k equal to the number ground truth classes remaining after the imagery from each data set was pre-processed (consult Table I for the number of ground truth classes within the data sets).
2) Variable Clustering Size (VCS) - this protocol requires an algorithm to accept a range of sizes within $[k_{\min}, \ldots, k_{\max}]$, where $k_{\min} = 0.2 * n_f$ and $k_{\max} = 0.7 * n_f$. In all cases, this range included the true number of identity classes that remained after pre-processing.

The evaluation protocols address the scenarios where all of the constraints are correct and where randomly selected constraints are incorrect, as human users typically make mistakes. Noise was introduced by flipping cannot-link constraints to must-link constraints and vice versa with a probability of 0.10. Each experimental configuration involving constraint noise and/or a non-deterministic algorithm was evaluated 25 times. We report the average F-measure over the 25 trials and present error bars representing one standard deviation about the mean.

D. Parameterization

The behavior of the FACE framework can be controlled by altering a number of parameters. The number of queries per iteration n_q was set to 10 and the number of iterations T was fixed at 100. No more than, say, 500 active constraints would be gathered after 50 iterations with these settings. Another set of important parameters governs the complexities of the clusterings produced for the ensemble or as output. For the VCS protocol, the sizes of the output clusterings were selected using Equation 7 from Section III-A3. For the FCS protocol, we cut the dendrogram containing the fused clusterings at the level with k clusters instead of applying equation Equation 7. For both evaluation protocols, we set $k_{\min} = 0.2 * n_f$ and $k_{\max} = 0.7 * n_f$ for the clusterings within the ensemble to promote diversity and maintain parity with the VCS results.

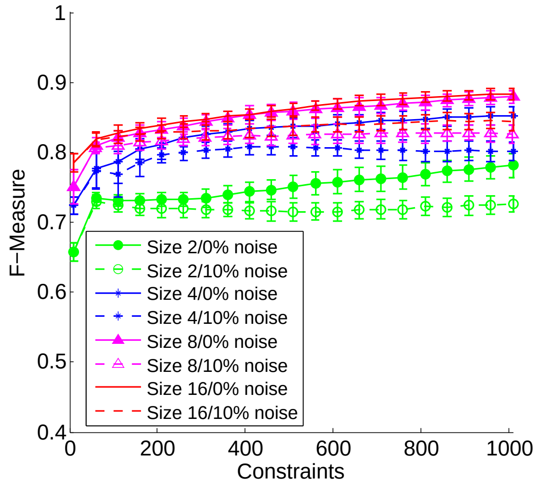


Fig. 10. LCVQE performance on SN-Flip

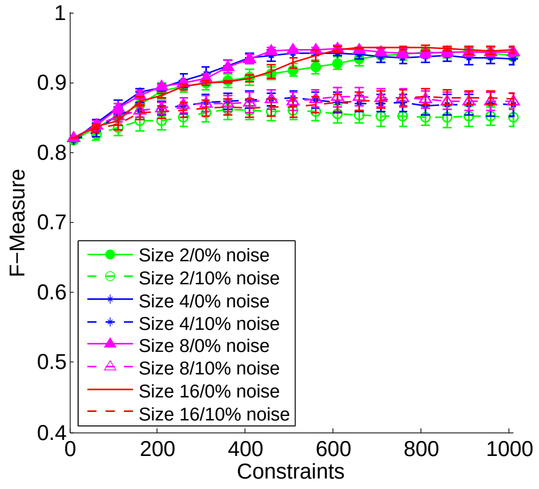


Fig. 11. SHACC performance on SN-Flip

In the next section, we analyze the effects of altering the ensemble size n_b and the collection $\boldsymbol{\beta}$ of base clustering algorithms used for creating the ensemble.

E. Varying Ensemble Sizes and Base Clustering Algorithms

We varied the ensemble size within $\{2, 4, 8, 16\}$ as we compared combinations of base clustering algorithms on the SN-Flip data set. Figures 10, 11 and 12 show the results for the VCS protocol; the trends for the FCS protocol are nearly identical. Doubling the size of the ensemble from two to four members increased performance for all of the configurations. Increasing the ensemble size to eight members resulted in significant improvements for the LCVQE and LCVQE+SHACC based ensembles. The performance of the LCVQE+SHACC combination rose slightly at some points when the ensemble reached 16 members.

In all other cases, performance either plateaued or declined when the ensemble grew in size. The deterministic SHACC algorithm was particularly poor at creating sufficiently diverse clusterings for the larger ensembles to yield higher quality

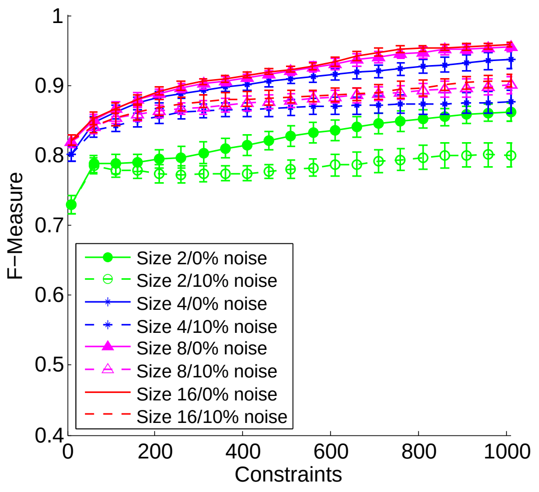


Fig. 12. LCVQE+SHACC performance on SN-Flip

partitionings. By comparison, LCVQE clusterings vary in response to random changes in the initial cluster centers. The LCVQE+SHACC scheme was able to benefit from larger ensembles because of this additional source of diversity incorporated into the LCVQE algorithm, along with the fact that the LVQE and SHACC algorithms create clusterings with disparate structures.

The impact of noise was not as significant for LCVQE as it was for SHACC, even though SHACC performed significantly better in general. SHACC accommodates the pairwise constraints locally, as it focuses on pairwise clustering assignments as opposed to the global structure of the data. LCVQE responds to constraints by shifting clustering centers in directions that will eventually lead to fewer constraint violations. SHACC consequently surpassed LCVQE when the constraints were noise-free since it was more responsive. LCVQE was less sensitive to noise though.

Using LCVQE and SHACC in tandem produced the best results in both the noise-free and noisy regimes. The errors committed by the base clustering algorithms tended to be offsetting, which allowed the LCVQE+SHACC combination to achieve the best results. The LCVQE+SHACC configuration with 16 clusterings reached respective F-measures of 0.95 and 0.90 in the noise-free and noisy cases after querying 777 constraints, which is the operating point where there is no more than one constraint per face on average. We refer to this parameterization as the FACE algorithm for simplicity.

The clusterings from all of the configurations typically improved as more constraints were gathered, whether or not the responses to the queries were noisy. The changes induced by 100 constraints consistently increased the quality of the output clustering. The performance degradations resulting from noise were similar for most of the ensemble sizes and combinations of base algorithms. Although the effects of noise were largely negative, the ensembles tended to bolster the formation of higher fidelity clusterings as the set of constraints expanded.

F. Active Clustering Algorithms

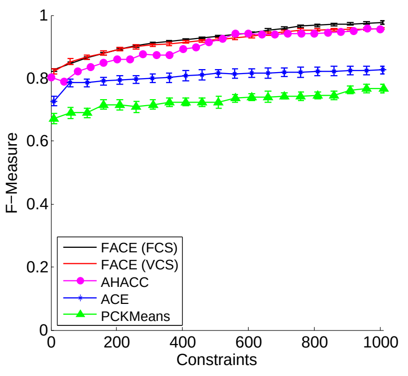
We evaluated the FACE framework on the SN-Flip, YMU and Twins data sets alongside active clustering algorithms that have been previously applied to face data. These algorithms included the PCKMeans method [24], the ACE technique [26] and the state-of-the-art AHACC algorithm [16]. Each of these approaches has a readily available implementation. PCKMeans, ACE and AHACC do not incorporate a means of automatically selecting the number of clusters, so they were assessed using the FCS protocol alone while FACE was evaluated using the FCS and VCS protocols. Consult Figures 13 and 14 for the results.

The performance trends are similar across data sets. All of the approaches were able to benefit from increasingly large constraint sets in the noise-free case. The algorithms that exploit pairwise constraints generally outperformed constraint-agnostic algorithms when all of the feedback was noise-free: PCKMeans without constraints is equivalent to traditional k-means, and the performance of AHACC at the zero-constraint mark is representative of single-like HAC algorithms.

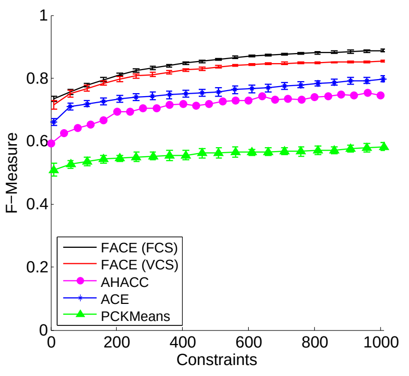
PCKMeans performed poorly. As the number of constraints grew, PCKMeans failed to increase the fidelity of the clustering at a rate as high as the other algorithms. The ACE, AHACC and FACE schemes inform the selection of active queries with outputs from the clustering algorithm(s). The PCKMeans constraints are not necessarily correlated with the pairs the clustering algorithm finds most difficult because all of the constraints are gathered before clustering begins. Noise is particularly harmful since PCKMeans enlarges an initial set of constraints by applying the transitive closure over the must-link set. Additional cannot-link constraints are added through entailment: if a must-link constraint m_{ij} and a cannot-link constraint c_{jk} reside in the constraint pool, a cannot-link c_{ik} is created. These steps propagate noise, causing the quality of the PCKMeans clusterings to degrade below the quality of the partitionings produced by k-means and the other active clustering algorithms.

The relative performance of the ACE method differed between data sets. The performance degradation resulting from constraint noise was small compared to all of the other algorithms. It was the third most accurate technique on the YMU data set, but the second least accurate algorithm on the SN-Flip and Twins data sets. Similar to the FACE framework, ACE performs consensus clustering. The application of ensemble methods mitigated the impact of noise for both of these algorithms. However, ACE only uses a single base clustering algorithm, but FACE draws on disparate algorithms to increase diversity within the ensemble. Another key difference is that ACE selects query pairs solely in terms of the ensemble consensus. FACE accounts for the ensemble consensus as it finds neighboring pairs that may define the boundaries between clusters. These differences lead the FACE algorithm to create more accurate clusterings and solicit higher impact constraints.

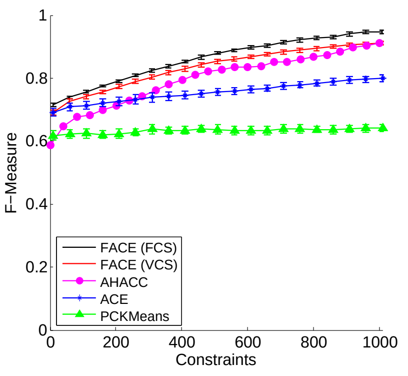
AHACC reached similar levels of performance to the FACE framework in the noise-free regime on the SN-Flip and YMU data sets. AHACC also produced the most stable results on noise-free constraints due to its deterministic nature. In the



(a) SN-Flip

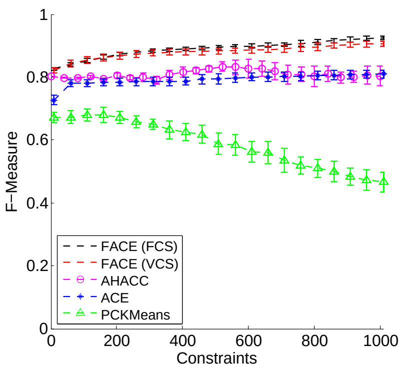


(b) YMU

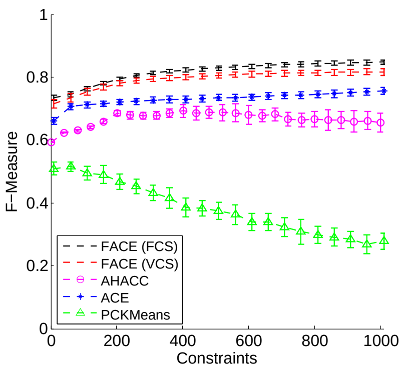


(c) Twins

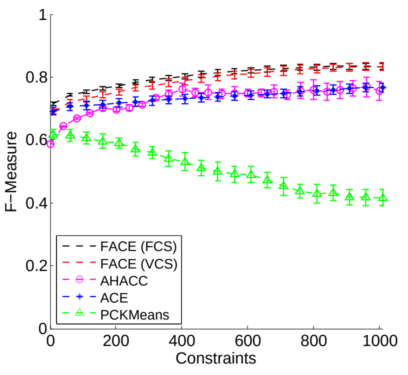
Fig. 13. Results with noise-free constraints.



(a) SN-Flip



(b) YMU



(c) Twins

Fig. 14. Results with 10% noise in the constraints.

noisy regime, AHACC was able to improve the clustering as more constraints were gathered, until the number of erroneous constraints passed a threshold. This behavior is similar to that of the related SHACC algorithm (see Section IV-E). SHACC, however, was less sensitive to noise on the SN-Flip data set since it operated in an ensemble setting.

AHACC is particularly responsive to constraints. The addition of some constraints can produce large changes in the clustering. Moreover, the AHACC querying scheme explicitly searches for the pair of faces that would produce the greatest expected change. These characteristics can significantly amplify the effects of noise.

The FACE framework generally was the most accurate and robust approach. FACE required a relatively small number of constraints to achieve each level of performance, regardless of whether the size of the clustering was selected automatically. Providing the number of identity clusters as input, as in the FCS protocol, frequently enabled FACE to outperform the other algorithms by a wider margin. A clear trend for each of the ensemble methods, FACE and ACE, is the general tendency for the aggregated clustering to continue improving as the number of noisy constraints increases. This robustness was consistently exhibited by the clustering ensembles across all three data sets. The FACE algorithm is thus well-suited for typical face clustering applications, because it can successfully exploit valuable yet noisy human feedback to offset the failures of automatic face matching algorithms.

V. Conclusions

Active face clustering is a useful means of organizing collections of face images, despite the current limitations of face recognition technology. A clustering algorithm must be responsive to user feedback yet robust to noise in order to be of any practical use. The number of constraints required to achieve a given level of performance must also be minimal.

This paper presents a number of key findings:

1) For bounded k, determining whether a given set of soft constraints can be satisfied with a minimum number of violations is computationally infeasible. An efficient algorithm cannot find a single clustering that minimizes the number of constraint violations, much less the best clustering.
2) The results indicate that ensemble-based constrained clustering algorithms are generally more robust to noise than alternative approaches. These methods continue to improve the quality clustering as the set of constraints expands, even though the absolute number of constraint errors increases.
3) Taking the transitive closure of the must-link constraints and inferring cannot-link constraints is only useful in

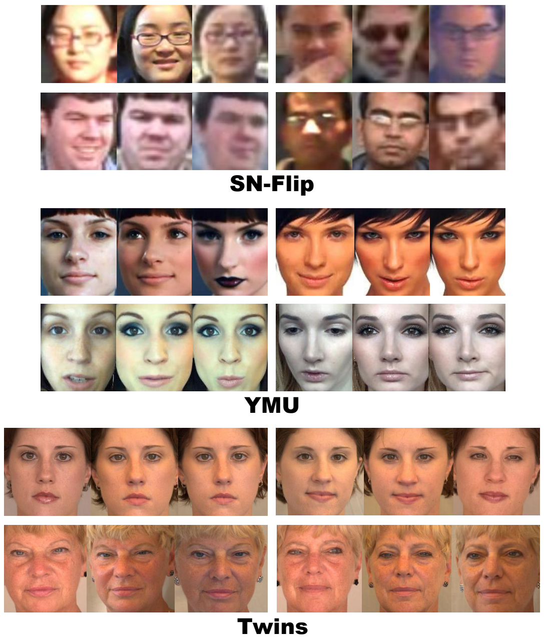


Fig. 15. Examples of correct clusters formed by the FACE algorithm.

contexts where the constraints are guaranteed to be completely accurate. This observation is based on the strong impact noise has on PCKMeans, a soft-constrained algorithm that applies these techniques to increase the number of constraints. This result is in agreement with a thorough empirical study that was recently conducted by Covões *et al.* [35], who found that performing the transitive closure with noisy constraints was detrimental to the constrained vector quantization error [12], LCVQE and the metric PCKMeans [40] algorithms. Arbitrarily increasing the number of noise-free constraints through imputation is not guaranteed to improve performance either [41].

4) The proposed FACE algorithm is accurate, robust and parsimonious compared to the state-of-the-art. Pairs of neighboring face patterns that are ambiguously matched in terms of the ensemble consensus were shown to make effective queries.
5) Human-in-the-loop face recognition can improve clustering performance on challenging data sets. The FACE algorithm successfully exploited noisy constraints on point-and-shoot videos, images of women wearing heavy makeup, and photographs of identical twins.

Our future work will expand on these results. An open research problem is to determine whether or not a particular constraint is erroneous. The "consensus of the crowd" could serve as a tool for detecting noise, as could comparisons with clusterings from the ensemble. The affiliation network connecting the clusters of people who appear together frequently could act as another source of information for validating constraints. In the context of constraint sets that are enriched through inference, this goal could be accomplished by comparing the queried and imputed constraints to determine which ones are likely to be incorrect. Further research will center on increasing the efficiency of the framework through the application of distributed computing and approximation methods.

Appendix A
Soft Constraint Feasibility Problem

The computational intractability of the soft constraint feasibility (SCF) problem can be demonstrated by reduction from the hard constraint feasibility (HCF) problem described in [12].

Theorem 2. *The soft constraint feasibility problem is* ***NP****-complete.*

Proof. The verification of a solution to an SCF instance can be done in polynomial time by checking the number of clusters and the number of violated constraints. The problem is thus in **NP**. We prove that the SCF problem is **NP**-hard by reduction from the HCF problem. Let I_h be an HCF instance with pattern matrix $\boldsymbol{X}$, hard constraints $\mathcal{L} = \mathcal{M} \cup \mathcal{C}$, and clustering bounds $k_{\min}$ and $k_{\max}$. $\boldsymbol{X}$, $\mathcal{L}$, $k_{\min}$ and $k_{\max}$ can be used as-is in the corresponding SCF instance I_s. Finally, we set $V = 0$. This transformation can be carried out in constant time. It is also clear that we have a solution to I_h if and only if we have a solution to I_s. □

References

[1] "YouTube statistics," http://www.youtube.com/yt/press/statistics.html, retrieved April 14, 2014.
[2] R. Chellappa, P. Sinha, and P. J. Phillips, "Face recognition by computers and humans," *Computer*, vol. 43, no. 2, pp. 46–55, Feb. 2010.
[3] N. A. Spaun, "Forensic biometrics from images and video at the Federal Bureau of Investigation," *Proc. IEEE International Conference on Biometrics: Theory Applications and Systems*, 2007.
[4] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, Sep. 1999.
[5] P. J. Phillips and A. J. O'Toole, "Comparison of human and computer performance across face recognition experiments," *Image and Vision Computing*, vol. 32, no. 1, pp. 74–85, Jan. 2014.
[6] J. R. Beveridge, P. J. Phillips, D. S. Bolme, B. A. Draper, G. H. Given, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, K. W. Bowyer, P. J. Flynn, and S. Cheng, "The challenge of face recognition from digital point-and-shoot cameras," *Proc. IEEE International Conference on Biometrics: Theory, Applications and Systems*, 2013.
[7] P. Sinha, B. Balas, Y. Ostrovsky, and R. Russell, "Face recognition by humans: 19 results all computer vision researchers should know about," *Proc. IEEE*, vol. 94, no. 11, pp. 1948–1962, 2006.
[8] S. Ueda and T. Koyama, "Influence of make-up on facial recognition," *Perception*, vol. 39, no. 2, pp. 260–264, 2010.
[9] A. Dantcheva, C. C. Chen, and A. Ross, "Can facial cosmetics affect the matching accuracy of face recognition systems?" *Proc. IEEE International Conference on Biometrics: Theory, Applications and Systems*, pp. 391–398, 2012.
[10] S. Biswas, K. W. Bowyer, and P. J. Flynn, "A study of face recognition of identical twins by humans," *Proc. IEEE International Workshop on Information Forensics and Security (WIFS)*, 2011.
[11] A. J. O'Toole, H. Abdi, F. Jiang, and P. J. Phillips, "Fusing face-verification algorithms and humans," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 5, pp. 1149–1155, Oct. 2007.
[12] I. Davidson and S. S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm," *Proc. SIAM International Conference on Data Mining*, 2005.
[13] A. P. Topchy, M. H. C. Law, A. K. Jain, and A. L. Fred, "Analysis of consensus partition in cluster ensemble," *Proc. IEEE International Conference on Data Mining*, pp. 225–232, 2004.
[14] B. Wu, Y. Zhang, B. Hu, and Q. Ji, "Constrained clustering and its application to face clustering in videos," *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

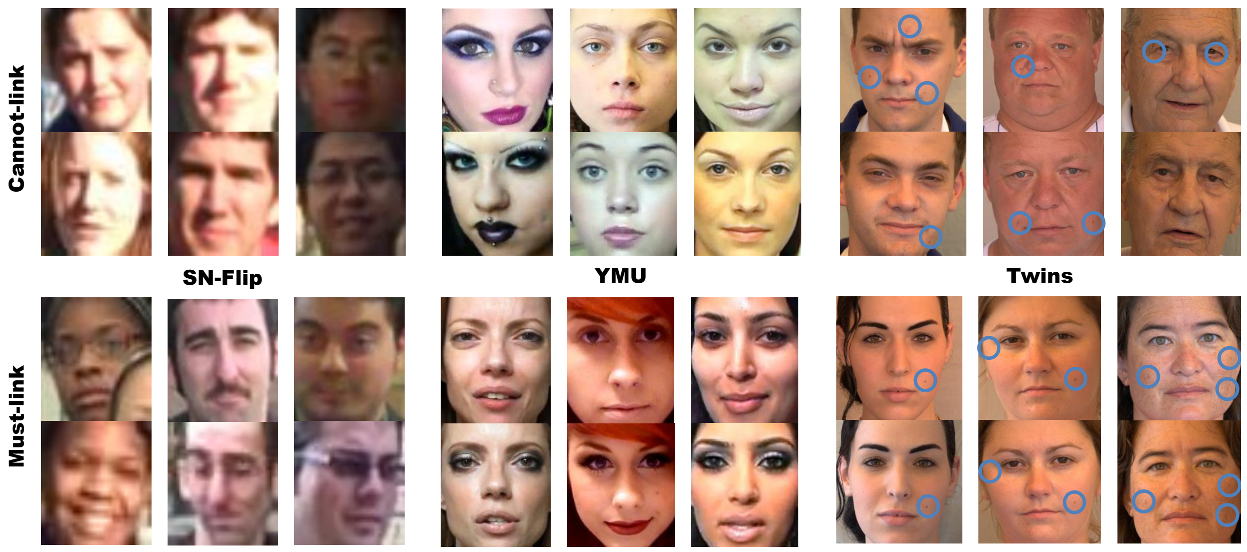


Fig. 16. Constraint samples from the FACE algorithm. Each vertically grouped pair is associated with a cannot-link (top row) or a must-link (bottom row). Circles mark identifying facial features in the Twins images.

[15] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji, "Simultaneous clustering and tracklet linking for multi-face tracking in videos," *Proc. IEEE International Conference on Computer Vision*, pp. 2856–2863, 2013.

[16] A. Biswas and D. Jacobs, "Active image clustering with pairwise constraints from humans," *International Journal of Computer Vision*, pp. 1–15, 2013.

[17] J. Tao and Y.-P. Tan, "Efficient clustering of face sequences with application to character-based movie browsing," *Proc. IEEE International Conference on Image Processing*, pp. 1708–1711, 2008.

[18] J. Sivic, M. Everingham, and A. Zisserman, "'Who are you?' - Learning person specific classifiers from video," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[19] T. Yu, S.-N. Lim, K. Patwardhan, and N. Krahnstoever, "Monitoring, recognizing and discovering social networks," *Proc. Conference on Computer Vision and Pattern Recognition*, pp. 1462–1469, 2009.

[20] C. Vallespi, F. D. la Torre, M. Veloso, and T. Kanade, "Automatic clustering of faces in meetings," *Proc. IEEE International Conference on Image Processing*, pp. 1841–1844, 2006.

[21] S. Prince and J. Elder, "Bayesian identity clustering," *Proc. Canadian Conference on Computer and Robot Vision*, pp. 32–39, 2010.

[22] R. G. Cinbis, J. Verbeek, and C. Schmid, "Unsupervised metric learning for face identification in TV video," *Proc. IEEE International Conference on Computer Vision*, pp. 1559–1566, 2011.

[23] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison Tech. Report*, 2010.

[24] S. Basu, A. Banerjee, and R. J. Mooney, "Active semi-supervision for pairwise constrained clustering," *Proc. SIAM International Conference on Data Mining*, pp. 333–344, 2004.

[25] X. Wang and I. Davidson, "Active spectral clustering," *Proc. IEEE International Conference on Data Mining*, pp. 561–568, 2010.

[26] J. R. Barr, L. A. Cament, K. W. Bowyer, and P. J. Flynn, "Active clustering with ensembles for social structure extraction," *Proc. IEEE Winter Conference on Applications of Computer Vision*, 2014.

[27] K. Wagstaff, "Intelligent clustering with instance-level constraints," *Cornell University PhD Thesis*, 2002.

[28] A. Fred and A. Jain, "Data clustering using evidence accumulation," *Proc. International Conference on Pattern Recognition*, vol. 4, pp. 276–280, 2002.

[29] Y. Freund, H. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine Learning*, vol. 28, no. 2-3, pp. 133–168, Aug. 1997.

[30] A. Strehl and J. Ghosh, "Cluster ensembles - A knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[31] X. Z. Fern and W. Lin, "Cluster ensemble selection," *Statistical Analysis and Data Mining*, vol. 1, no. 3, pp. 128–141, Nov. 2008.

[32] D. Pelleg and D. Baras, "K-means with large and noisy constraint sets," *Proc. European Conference on Machine Learning*, pp. 674–682, 2007.

[33] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[34] I. Davidson and S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," *Proc. European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 59–70, 2005.

[35] T. F. C. oes, E. R. Hruschka, and J. Ghosh, "A study of k-means-based algorithms for constrained clustering," *Intelligent Data Analysis*, vol. 17, no. 3, pp. 485–505, 2013.

[36] R. Sibson, "SLINK: an optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, no. 1, pp. 30–34, 1973.

[37] C. Chen, A. Dantcheva, and A. Ross, "Automatic facial makeup detection with application in face recognition," *Proc. International Conference on Biometrics (ICB)*, 2013.

[38] N. Srinivas, G. Aggarwal, P. F. Flynn, and R. W. V. Bruegge, "Analysis of facial marks to distinguish between identical twins," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1536–1550, Oct. 2012.

[39] C. Tomasi and T. Kanade, "Detection and tracking of point features," *Carnegie Mellon University Technical Report CMU-CS-91-132*, 1991.

[40] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," *Proc. International Conference on Machine Learning*, 2004.

[41] I. Davidson, K. L. Wagstaff, and S. Basu, "Measuring constraint-set utility for partitional clustering algorithms," *Proc. European Conference on Principle and Practice of Knowledge Discovery in Databases*, pp. 115–126, 2006.