Preserving Privacy by De-identifying Facial Images

Elaine Newton Latanya Sweeney Bradley Malin

March 2003
CMU-CS-03-119

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

In the context of sharing video surveillance data, a significant threat to privacy is face recognition software, which can automatically identify known people, such as from a database of drivers' license photos, and thereby track people regardless of suspicion. This paper introduces an algorithm to protect the privacy of individuals in video surveillance data by de-identifying faces such that many facial characteristics remain but the face cannot be reliably recognized. A trivial solution to de-identifying faces involves blacking out each face. This thwarts any possible face recognition, but because all facial details are obscured, the result is of limited use. Many ad hoc attempts, such as covering eyes or randomly perturbing image pixels, fail to thwart face recognition because of the robustness of face recognition methods. This paper presents a new privacy-enabling algorithm, named *k*-Same, that scientifically limits the ability of face recognition software to reliably recognize faces while maintaining facial details in the images. The algorithm determines similarity between faces based on a distance metric and creates new faces by averaging image components, which may be the original image pixels (*k*-Same-Pixel) or eigenvectors (*k*-Same-Eigen). Results are presented on a standard collection of real face images with varying *k*.

This research was supported in part by the Laboratory for International Data Privacy at Carnegie Mellon University and in part by the Defense Advanced Research Projects Agency and managed by the Naval Sea Systems Command under contract N00024-98-D-8124.

Keywords: *Video surveillance, privacy, de-identification, privacy-preserving data mining, k-anonymity*

1. Introduction

The objective of this work is to develop technology that reasonably preserves the anonymity of people whose images are recorded in public spaces and who are engaged in legal activities, while still enabling video recordings to be shared for other worthy purposes. The methods presented in this paper accomplish this by digitally modifying the images so that face recognition software cannot reliably relate people to their captured images.

There has been a tremendous proliferation of video surveillance cameras in public locations such as stores, ATMs, schools, buses, subway stations, and airports, in order to combat crime. For example, a recent survey of Times Square found 500 visible surveillance cameras in the area and 2500 total in New York City [6]. The existence of so many cameras and proposals for even more cameras have caused many citizens to protest their use in Tampa, Florida and Virginia Beach, Virginia where video cameras are being used in conjunction with face recognition software [7, 15], thereby enabling the eventual possibility of tracking almost all of the people most of the time.

In the United States, the legal grounds for law enforcement's use of video surveillance cameras to conduct a visual search in public places is based on the Fourth Amendment of the U.S. Constitution and case law, which has put forward the "reasonable expectation of privacy" test. While American society expects cameras in many public places, secondary sharing of video surveillance data and automatic face recognition is currently not expected. One could imagine in the near future the widespread use of video surveillance cameras, the subsequent sharing of that data, and the use of reliable face recognition software to explicitly identify and track people in real-time.

The goal of this work is to enable the sharing of video data with scientific assurances of privacy protection while keeping the data practically useful. What is needed is an algorithm to de-identify faces in video data such that many facial characteristics remain yet face recognition software cannot reliably identify subjects whose images are captured in the data. This work formally introduces the "preserved face de-identification" problem, in which face recognition software is restricted and details remaining in the face are minimally distorted. Sharing only de-identified data restores the current expectation of privacy so that society does not have to choose safety over privacy, but society can have both safety and privacy.

De-identifying video images involves altering the original face image. Preliminary ideas such as covering the eyes and nose, reducing the number of pixels in the face by averaging gray-scale values in a square block of pixels, and transforming gray-scale pixels to either a black or white value, were tested. We report herein on experiments that reveal that none of these kinds of solutions are an effective guard. Face recognition software can still accurately identify what remains.

The *k*-Same algorithm, introduced in this paper, is an effective solution. Faces that are similar to each other are averaged in such a way that many characteristics remain yet the resulting image is not reliably recognized by face recognition software. We report on experiments using the Army's FERET database, which includes multiple images of 1109 subjects. These images were effectively de-identified using *k*-Same.

2. Face De-identification Definitions

The principles of face de-identification are being first introduced in this work. It is imperative to precisely define terms and revisit common expressions from this vantage point. This section starts with a definition of a "face still" and progresses to definitions of "preserved face de-identification" and the "*k*-Same" problem. The next section carefully introduces the methods of face recognition software. Subsequent sections formally present heuristic-based *k*-Same algorithms and report on experimental results. This paper ends with discussion on steps needed to operationalize *k*-Same.

Images of faces, after a pre-processing phase, are the topic of this work. These are stored as a vector of values, but displayed visually as a matrix. See Definition 2.1 and Definition 2.2.

Definition 2.1 (Face Still) A **face still** is a column vector **P** of size N_{pic}. Each cell in **P** stores a value from 0 to 255, inclusive, which is the grayscale pixel value reporting intensity. The image displayed in a face still includes only one person's face.

Example 2.1 (Face Still) Figure 1(a) shows a face still. The size and orientation of the face is not assumed in a face still.

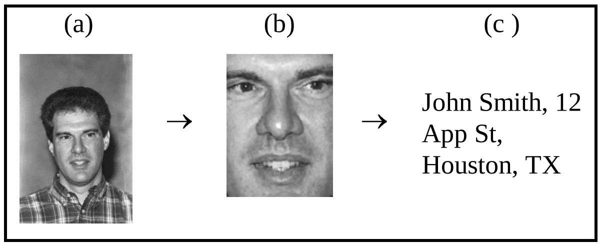


Figure 1. Face still (a) is normalized to face image (b) then identified to provide the name and address of the subject (c).

After a pre-processing phase, sometimes called "registration" in face recognition, a face still (definition 2.1) becomes a face image (definition 2.2). A face still is normalized, rotated, translated, and cropped, as needed, to provide a face image. The goal is to adjust for pose and make the location of eyes and their inter-pupil distance align in roughly the same position in each image, thereby making face images somewhat comparable to one another. The pre-processing needed to get a face image from a face still is one of the greatest current weaknesses of face recognition [4].

Definition 2.2 (Face Image) A **face image** (or simply "face" or "image") is a column vector Γ of size N. Each cell in Γ stores a value from 0 to 255, inclusive, which is the grayscale pixel value reporting intensity

Example 2.2 (Face Image) Figure 1(b) shows a face image normalized from Figure 1(a). The face image in this example has 13,266 pixels and is displayed graphically in 99 columns and 134 rows.

The detection of faces, the localization of face stills, and the registration of face stills into face images are all considered pre-processing to this work. Therefore, this work does not exploit any current weaknesses in detection and registration. This paper assumes well-formed face images are being processed. If the face images are not well formed, more privacy is achieved. For the remainder of this paper, unless otherwise noted, it is assumed that face images are well-formed.

Definition 2.3 (Face Set) A **face set** is a set of M face images, $\{\Gamma_i : |\Gamma_i| = N, i = 1,\ldots,M\}$. The result is analogous to a matrix of M columns and N rows. Each column is a face image. Each row corresponds to the same pixel location in each face image. Each element is a pixel.

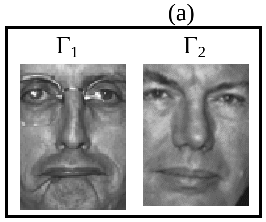
A face set, as defined in Definition 2.3, has M faces appearing as column vectors. There are N rows; each row represents a pixel location. It is often convenient in terms of privacy discussions to have a single face in a face set relate to one person. This is the idea of a "person-specific" face set described in Definition 2.4.

Definition 2.4 (Person-specific Face Set) Let **H** be a face set having M images, $\{\Gamma_1,\ldots, \Gamma_M\}$. **H** is **person-specific** if and only if each $\Gamma \in \mathbf{H}$ relates to only one person and no two images $\Gamma_1 \in \mathbf{H}$, $\Gamma_2 \in \mathbf{H}$ relate to the same person.

Example 2.3 (Person-specific Face Set) Figure 2 shows a person-specific face set having two face images. Each face is a grayscale image with 13,266 pixels. Each image relates to a distinct person and the two images do not relate to the same person. Figure 2(a) shows the face set graphically. Figure 2(b) shows the face set in a matrix configuration. Each column is a face image and each row is a pixel location.

Definition 2.5 (Face Identification) **<u>Face identification</u>** results when a face image is properly associated with explicit identifiers, such as name and address, of the person who is the subject of the face image.

(a)



(b)

	Γ_1	Γ_2
[1]	131	150
[2]	126	150
[3]	126	140
…	…	…
[13266]	73	26

Figure 2. Person-specific face set

Example 2.4 (Face Identification) Figure 1(c) shows the results of face identification. The face image in Figure 1(b) is identified as being that of “John Smith”.

Face identification, as presented in Definition 2.5, relates explicit identification to the subject whose face appears in the face image. Explicit identifiers, such as name and address, allow us to directly communicate with the subject [21]. “Face recognition,” as opposed to “face identification,” involves relating a face image to a known face image. Given a face set, face recognition software, as described in Definition 2.6, relates a face image to one in a given face set. Explicit identification of the faces in the face set may or may not be possible. Face recognition is the primary concern in this work.

Definition 2.6 (Face Recognition Software) Given a face set $\mathbf{H}$ and a face image Γ, **<u>face recognition software</u>** is a program that returns the image in $\mathbf{H}$ best matching Γ. Presumably, Γ is of a subject represented in $\mathbf{H}$.

In the next section, face recognition software is examined in detail. The goal of this work is to alter face images in such a way that automated face recognition cannot be reliably performed on the resulting images. Before claims of thwarting face recognition software can be made, more discussion is needed on altering face images. This is termed de-identification of face images, as described in Definition 2.7.

Definition 2.7 (Face De-identification) Let $\mathbf{H}$ and $\mathbf{H_d}$ be face sets where $\Gamma \in \mathbf{H}$ and $\Gamma_d \in \mathbf{H_d}$; f:$\mathbf{H} \rightarrow \mathbf{H_d}$ be a transformation function that attempts to conceal the identity of the subject of the original face image; and, $f(\Gamma) = \Gamma_d$. Γ and Γ_d have the same dimensions but $\Gamma \neq \Gamma_d$ (element-wise). f is termed **<u>face de-identification</u>** (or simply “de-identification”) Γ_d is a de-identified image of Γ. f is also termed a “de-identification function.”

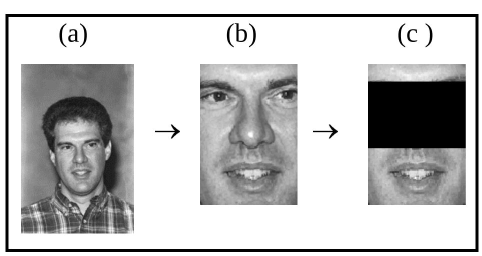


Figure 3. Face still (a) is normalized to a face image (b) and then de-identified to face image (c).

Example 2.5 (Face De-identification) Figure 3(c) is a de-identified image of Figure 3(b) in which the eyes are covered.

De-identifying a face image may provide some privacy protection, but the act of de-identification itself provides no assurances of anonymity. Other details may remain in the image that allow the subject to be re-identified; the process might be reversible; or there may exist other re-identification strategies that make identification possible. For example, different ad hoc efforts have attempted to mask identities of people during television interviews. At least one de-identification attempt failed to provide adequate protection and resulted in a lawsuit [24]. Later in this paper, it is shown that common ad hoc efforts at masking faces do not thwart face recognition software. De-identification alone is not sufficient; it has no qualifiers on what must be accomplished. A basis for making privacy assurances is needed.

The idea of "effective de-identification," as described in Definition 2.8, is to de-identify faces with some stated privacy assurances achieved by the de-identification. The goal is for de-identification to restrict face recognition (and/or face identification) to some provable extent.

Definition 2.8 (Effective De-Identification) Let $\mathbf{H}$ be a person-specific face set; $\mathbf{H_d}$ be a face set; f:$\mathbf{H}\rightarrow\mathbf{H_d}$ be the transformation function used in face de-identification, such that $f(\Gamma_1)=\Gamma_2$ where $\Gamma\in\mathbf{H}$ and $\Gamma_d\in\mathbf{H_d}$; g be a face identification relation g:$\mathbf{H_d}\rightarrow\mathbf{H}$; and, C be a claim about f's ability to restrict face identification (or face recognition) by g. The function f provides **effective de-identification** with respect to C. The goal is to determine the appropriate function f with a provable C. It is said that f is "effective." If f_1 and f_2 are effective with respect to the same C, then f_1 and f_2 are considered equally effective with respect to C.

Example 2.6 (Effective De-Identification) Let $\mathbf{H}$ be a person-specific face set and *BlackOut*() be the de-identification function defined as:

> Given person-specific face set $\mathbf{H}$ and face image $\Gamma\in\mathbf{H}$ having N rows,
> return $[0_1,\ldots,0_N]$.

The *BlackOut*() method returns a face image where each of the N cells has 0. In grayscale, 0 displays as the color black. Let f be *BlackOut*:$\mathbf{H}\rightarrow\mathbf{H_d}$, g be a relation such that g:$\mathbf{H_d}\rightarrow\mathbf{H}$, and C be:

> Given face set $\mathbf{H}$, $|\mathbf{H}|>1$, f:$\mathbf{H}\rightarrow\mathbf{H_d}$, and face image Γ_2, where $\Gamma_2 = f(\Gamma_1)$ for $\Gamma_1\in\mathbf{H}$,
> $\Gamma_2\in\mathbf{H_d}$
> Γ_1 cannot be uniquely determined.

When f is *BlackOut*(), the correctness of claim C is straightforward. Any relation g will relate to all members of $\mathbf{H}$ indistinctly. For example, sample runs of *BlackOut*() are shown in Figure 4. Let a, b, c and d be the face images shown in Figure 4a, 4b, 4c and 4d, respectively. *BlackOut*(a)=b and *BlackOut*(c)=d. Given face image e, which is either a copy of b or d, no human or machine can determine whether the subject of the image is a or c because b and d are identical. Correct face recognition is limited to guessing with probability $1/|\mathbf{H}|$. Face identification (putting a name to the face) depends on a further relation on $\mathbf{H}$.

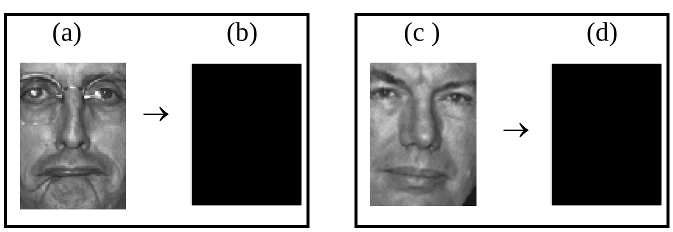


Figure 4. Effective De-identification using *BlackOut*()

De-identifications that have provable assurances of privacy protection are extremely powerful because they can be easily integrated into policy. Laws and regulations can dictate the nature of scientific assurances required for video de-identification in a particular setting. For example, *BlackOut*() is certainly an effective privacy guard in which no facial details are to be provided. But, *BlackOut*() may not be best for all policy settings. Some settings may require effective de-identification that maintains many facial details in the image. This is the idea behind "preserved face de-identification."

Let $F = \{f_i\}$ be a set of de-identification functions. By definition, a de-identification function f_i generates a distorted copy of the original face image. If $f_1 \in F$ is effective, then f_1 provides some provable privacy assurances. Of all $f_i \in F$ that are just as effective as f_1, preference is made for those that maintain the most detail in the image. The goal is the least distorted effective de-identification.

A metric named *loss*() is introduced to capture the information loss resulting from de-identification. *loss*() is a monotonic function that can be based on entropy, percentage of changed pixels, or some other scheme. *loss*() compares two face images, Γ_1 and Γ_2, and reports the amount of distortion between Γ_1 and Γ_2. The greater the value of $loss(\Gamma_1,\Gamma_2)$, the greater the information loss between Γ_1 and Γ_2. Minimum *loss*() is realized when the images are the same, such as $loss(\Gamma_1,\Gamma_1)$. Let f be a de-identification function, and $f(\Gamma_1)=\Gamma_2$, $loss(\Gamma_1,\Gamma_2) \geq loss(\Gamma_1,\Gamma_1)$. The *loss*() metric is used to describe "preserved face de-identification" in Definition 2.9.

Definition 2.9 (Preserved Face De-Identification) Let $\mathbf{H}$ be a person-specific face set; $\mathbf{H_d}$ be a face set; $F = \{f_i\}$ be a set of equally effective de-identification functions where each $f_i:\mathbf{H} \rightarrow \mathbf{H_d}$; $\Gamma \in \mathbf{H}$; and, *loss*() be a precision metric related to F. If for all $f_1 \in F$, there does not exist $f_2 \in F$ such that $loss(\Gamma,f_2(\Gamma)) < loss(\Gamma,f_1(\Gamma))$, then f_1 is a **preserved face de-identification** with respect to *loss*() and F. It is said that f_1 is "preservative." If $loss(\Gamma,f_1(\Gamma))=loss(\Gamma,f_2(\Gamma))$, then f_1 and f_2 are equally preservative with respect to *loss*() and F.

Example 2.7 (Preserved face De-Identification) Let $\mathbf{H}$ be a person-specific face set and *Average*() be the de-identification function defined as:

Given person-specific face set $\mathbf{H}$ and face image $\Gamma \in \mathbf{H}$ having N rows, return:

$$\psi = \left[\frac{\sum_{j=1}^{|\mathbf{H}|} \Gamma_j[1]}{|\mathbf{H}|} \quad , \ldots , \quad \frac{\sum_{j=1}^{|\mathbf{H}|} \Gamma_j[N]}{|\mathbf{H}|} \right] = \frac{1}{|\mathbf{H}|} \bullet \sum_{j=1}^{N} \Gamma_j \qquad (1)$$

The *Average*() method returns a face image (ψ) where each of the N cells has the average computed for each cell position in the images of $\mathbf{H}$. That is, each pixel in ψ is the sum of the value for that pixel in all the faces of $\mathbf{H}$ divided by the number of faces in $\mathbf{H}$. $Average:\mathbf{H} \rightarrow \mathbf{H_d}$. Let g be a relation such that $g:\mathbf{H_d} \rightarrow \mathbf{H}$ and C be the same C defined in Example 2.6. When f is *Average*(), the correctness of claim C is again straightforward. Any relation g will relate to all members of $\mathbf{H}$ indistinctly because each face is replaced with the average face ψ.

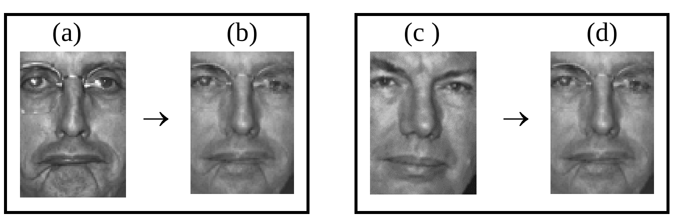


Figure 5. Preserved face De-identification using *Average*()

Sample runs of *Average*() are shown in Figure 5. Let a, b, c and d be the face images shown in Figure 5a, 5b, 5c and 5d, respectively. $\mathbf{H}=\{a, c\}$. $Average(\mathbf{H}, a)=b$ and $Average(\mathbf{H}, c)=d$. Given face image e, which is either a copy of b or d, no human or machine can determine whether the subject of the image is a or c because b and d are identical. Both b and d are ψ. As with *BlackOut*(), correct face recognition is limited to guessing with probability $1/|\mathbf{H}|$. So, both *BlackOut*() and *Average*() are equally effective. But in terms of details remaining in the image, *Average*() is better than *BlackOut*().

Let **H** be a face set and Γ_1 and Γ_2 be face images of size *N*. *euclid*() is a loss metric defined as the square root of the sum of the square of the differences in pixel values in the faces. *euclid*() is known as the Euclidian distance and is precisely defined as:

$$euclid(\Gamma_1, \Gamma_2) = \sqrt{\sum_{i=1}^{N} |\Gamma_1[i] - \Gamma_2[i]|^2} \tag{2}$$

Images *a* and *c* in Example 2.6 and Figure 4 are the same as images *a* and *c* in Example 2.7 and Figure 5. Let **H**=$\{a,c\}$ and $N=|a|=|b|$. Results from *BlackOut*() applied to **H** are in Example 2.6 and Figure 4 as images *4b* and *4d*; $4b=4d=[0_1,\ldots,0_N]$. *euclid*(*a*,*4b*)= 14,187 and *euclid*(*c*,*4d*)= 14,869. Results from *Average*() applied to **H** are in Example 2.7 and Figure 5 as images *5b* and *5d*. ψ is the average face computed from $\{a, c\}$ using Equation 1; $\psi=4b=4d$. *euclid*(a,ψ)= 2143 and *euclid*(c,ψ)= 2153. While *BlackOut*() and *Average*() are equally effective at de-identification, between the two of them, *Average*() is preservative. *Average*() leaves more detail in the resulting image.

A de-identification function can provide images that retain a lot of detail, but such functions may not be effective and therefore are not preservative. This is demonstrated in Examples 2.8 and 2.9.

Example 2.8 (Precise but not Preservative) Let *RandomOne*() be a de-identification function that given a face image Γ, provides a copy of Γ such that a randomly selected non-black pixel has its value changed to black. *RandomOne*() is a de-identification function that is not very effective. Despite all the precision that remains in the image, for example, *RandomOne*() does not respect claim *C* in Example 2.6, so *RandomOne*() is not as effective as *BlackOut*() and *Average*().

Given *euclid*() in Equation 2, as a loss metric, the maximum distortion a face image can undergo with *RandomOne*() is 255. That is, max(*euclid*(Γ, *RandomOne*(Γ)))=255, for any face image Γ. In comparison to the information loss measurements reported in Example 2.7, *RandomOne*() is more precise than *Average*() or *BlackOut*(), but it is not as effective, so it is not preservative.

Example 2.9 (Not Effective so Not Preservative) Figure 13 demonstrates an assortment of ad hoc de-identification functions. Except for *BlackOut*(), each of these de-identification functions will be shown by experimentation in a subsequent section to not be effective against face recognition software. Therefore, none of them are preservative.

The effectiveness of de-identification in both *BlackOut*() and *Average*() relies on a resulting image that relates ambiguously to all the members of the original person-specific face set (**H**). A substantive improvement involves having each de-identified image relate ambiguously to *k* members of the original face set, where $2 \le k \le |\mathbf{H}|$. In earlier work, *k*-anonymity was defined using field-structured data as providing privacy protection by having each record in the resulting data relate indistinctly to at least *k* individuals [20]. Definition 2.10 adapts the *k*-anonymity protection scheme to face images.

Definition 2.10 (*k*-anonymity on Face Images) Given a person-specific face set **H**, a set of de-identified face images $\mathbf{H_d}$ in which duplicates are maintained ($\mathbf{H_d}$ is a "multi-set"), $|\mathbf{H}|>1$, $|\mathbf{H}|=|\mathbf{H_d}|$, a de-identification function f: $\mathbf{H}\rightarrow\mathbf{H_d}$, and g:$\mathbf{H_d}\rightarrow\mathbf{H}$ a relation that is the inverse of *f*. If for each $\Gamma\in\mathbf{H}$ there exists $\Gamma_d\in\mathbf{H_d}$ where $f(\Gamma)=\Gamma_d$ and for each $\Gamma_d\in\mathbf{H_d}$, $|g(\Gamma_d)=\Gamma| \geq k$, then $\mathbf{H_d}$ adheres to **k-anonymity**. It is said that $\mathbf{H_d}$ is ***k*-anonymized** over **H**.

De-identifying each face in a person-specific face set $\mathbf{H_1}$, maintaining duplicate faces, provides a face set $\mathbf{H_2}$ that adheres to *k*-anonymity if and only if each face image appearing in $\mathbf{H_2}$ appears at least *k* times. Examples of *k*-anonymity on face images are provided in Examples 2.10, 2.11 and 2.12.

Example 2.10 (*k*-anonymity on Face Images) Let **H**=$\{a,c\}$ where images *a* and *c* are from Example 2.6 and Figure 4. $N=|a|=|b|$. Results from *BlackOut*() on the faces in **H** is $\mathbf{H}_3=\{[0_1,\ldots,0_N], [0_1,\ldots,0_N]\}$, as shown as Figure 4b and Figure 4d. $\mathbf{H}_3$ has $|\mathbf{H}|$=2 copies of the blackened image. $\mathbf{H}_3$ is *k*-anonymized, where *k* is 2.

Example 2.11 (*k*-anonymity on Face Images) Let $\mathbf{H}$ be a person-specific face set containing 100 images, $\mathbf{H}=\{\Gamma_1,\ldots,\Gamma_{100}\}$. *Average*(), defined in Example 2.7, is applied to each of the faces of $\mathbf{H}$ such that $\psi_{1,2}$ is the average face computed for Γ_1 and Γ_2 using Equation 1, $\psi_{3,4}$ is the average face computed for Γ_3 and Γ_4, …, $\psi_{99,100}$ is the average face computed for Γ_{99} and Γ_{100}. The resulting de-identified face set $\mathbf{H_4}=\{\psi_{1,2}, \psi_{1,2}, \ldots, \psi_{99,100}, \psi_{99,100}\}$. $\mathbf{H}_4$ has $|\mathbf{H}|$=100 images. $\mathbf{H}_4$ is *k*-anonymized, where *k* is 2.

Example 2.12 (*k*-anonymity on Face Images) Let $\mathbf{H}$ be the same set of 100 images from Example 2.11; $\mathbf{H}=\{\Gamma_1,\ldots,\Gamma_{100}\}$ where $|\Gamma_i|=N$ for i=1,…,100. *BlackOut*(), defined in Example 2.6, is applied to each of the faces in $\mathbf{H}$. The resulting de-identified face set $\mathbf{H_5}=\{[0_1,\ldots,0_N], \ldots, [0_1,\ldots,0_N]\}$. $\mathbf{H}_5$ has $|\mathbf{H}|$=100 images. $\mathbf{H}_5$ is *k*-anonymized, where *k* is 2, 3, …, 100.

Partitioning a person-specific face set into smaller face sets or "clusters" of at least *k* faces provides privacy protection because an aggregate face is published for the members of each cluster in lieu of their original images. Basing each aggregate face on a cluster of homogenous original faces minimizes information loss. One concern is finding optimal clusters – those having the *k* most homogenous faces. A distance measure is used to determine homogeneity or closeness.

In this work so far, a face image has been represented as a column vector having *N* cells. This same face image can also be represented as a point in *N*-dimensional space by viewing the vector as a tuple. A measure can then be used to compute distances between points (or faces) and to determine closest neighbors. An example is Euclidian distance, though other distance measures can be applied. Euclidian distance is defined as *euclid*() in Equation 2.

Once minimally distant clusters are found, a second concern emerges. Aggregate face images must be constructed in such a way as to minimize information loss. Many strategies are possible such as finding an equidistant point, or the point with the smallest squared distances. This work focuses on constructing a point (or face image) that represents the *N*-dimensional "average" of the cluster. In data mining, this is termed a centroid [10]. A centroid of a cluster is a face whose values are the pixel-wise average of the faces in the cluster. Equation 1 computes Ψ as the centroid of cluster $\mathbf{H}$. An example of a centroid is the averaged result in Example 2.7 and shown in Figure 5.

In summary, this work seeks preserved face de-identification that achieves *k*-anonymity privacy protection such that the de-identification is effective against face recognition software. The results of the de-identification must guarantee that face recognition software cannot reliably recognize the de-identified results. This is termed the *k*-Same problem, as described in Definition 2.11.

Definition 2.11 (*k*-Same) Given a person-specific face set $\mathbf{H}$; and, a face set $\mathbf{H_d}$ which is *k*-anonymized over $\mathbf{H}$ using a preserved face de-identification function f: $\mathbf{H} \rightarrow \mathbf{H_d}$, if f is effective with respect to the claim:

> Given any face image $\Gamma_d \in \mathbf{H_d}$, where $\Gamma_d = f(\Gamma)$ for $\Gamma \in \mathbf{H}$, there cannot exist any face recognition software for which the subject of Γ_d can be correctly recognized as Γ with better than $1/k$ probability.

then f is a <u>***k*-Same de-identification function**</u> and $\mathbf{H_d}$ is a <u>***k*-Same de-identification**</u>. The goal is to determine the appropriate function f with minimal information loss.

Partitioning a person-specific face set into *k*-sized clusters of faces achieves *k*-anonymity. Making each cluster's centroid minimally distant from each face in the cluster minimizes information loss. This is the approach for achieving *k*-Same solutions taken in this paper. It reduces to being the same problem as microaggregation in statistical disclosure control. Recent work [12] on microaggregation has shown that the optimal selection of minimally distant *N*-dimensional points, where $N>1$, is an NP-hard problem. A corollary is that no algorithm can be written that runs in reasonable time to find optimal groups of "closest" faces for *k*-Same clusters.

In this work two heuristic solutions *k*-Same-Pixel and *k*-Same-Eigen are presented. They both operate in real-time and achieve *k*-anonymity, but they do so in such a way that information loss may not be minimal. Both of these solutions achieve the *k*-Same claim of thwarting all face recognition software.

In the next section, the method and operating paradigm of face recognition software is introduced. Then, *k*-Same-Pixel and *k*-Same-Eigen are introduced as ways to thwart face recognition software. Following that are experimental results. This paper ends with a discussion on related work and a discussion on the general applicability of this work.

3. Face Recognition Software

Face recognition software is a relatively new study, but progress has recently been fueled by the availability of improved, inexpensive cameras and recent terrorist events in the USA. The first face recognition algorithm was developed in the late 1970's [8]. The current baseline algorithm for face recognition algorithm, "Eigenfaces" also known as Principal Components Analysis or PCA, was introduced in 1991 [22]. Because Eigenfaces is the fundamental face recognition technique against which others are measured and because multiple institutions openly provide its code, Eigenfaces is used in the methods explored in this paper. A description of how Eigenfaces works is provided below, but first, some of the basic terms are described.

Three face sets are used in face recognition software: the training set, gallery, and probe. The **training set** (**training**) is a face set used to initially learn parameters a face recognition algorithm may use to classify faces. A training set may be used to generate a representative face in algorithms where recognition is based on measurements of how face images differ from the representative face.

A **gallery** (**gallery**) is a face set of known faces. During recognition, face images are compared to those in **gallery**. It is not required that **training** and **gallery** be the same face set. A **probe** (**probe**) is a face set of faces to be "recognized" by selecting best matches in the gallery [11]. Subjects in **probe** are not necessarily also in **gallery**.

Recognition of a face in **probe** is a rank ordering of the faces in **gallery** where the "closest" face appears first and the least similar face appears last. The **performance** of a face recognition program is defined as the percentage of faces in **probe** correctly matching their e closest **gallery** images [22], where $e \geq 1$. When e is 1, only the single "best matched" result is used.

```
Algorithm: EigenFaces(probe, gallery, training)
Input:   Face sets probe, gallery, training
Output:  Displays each face in probe along with its best
         matching face in gallery
Steps
1      Setup(gallery, training)
2      for each Γ∈probe
3          match = Recognize(Γ)
4          print  Γ, match[1][1]    // display face & best match
```

Figure 6. Eigenfaces pseudo-code.

The Eigenfaces algorithm appears in Figure 6, with supporting methods in Figures 7, 8, and 9. The basic idea of *Eigenfaces*() is to map all the images in **gallery** into a face space characterized by the faces in **training**; see line 1 in Figure 6. Then, for each face in **probe**, display the image in **gallery** that best matches it in the previously defined face space; see lines 2,3 and 4 in Figure 6. In the next paragraphs, the details of these steps are provided.

Algorithm: *Setup*(gallery, training)

Input: Face sets **gallery** and **training**.

Output: **facespace,** a 2-dimensional matrix. Each row has a pair of face images. Column [1] is a face from **gallery** and column [2] is that face projected in "face space". Each face in **gallery** is present once in the matrix, so there are |**gallery**| rows.

Steps

```
1   let M = |training|
2   Ψ = Average(training)                       // defined in Equation 1
3   for each Γi∈training
4       Φi = Γi - Ψ                              // difference of average face
5   let A = [Φ1, …, ΦM]                         // matrix of difference vectors
6   let C = A^T A                               // variation of covariance matrix
7   L[i]= λi such that |C-λiI|=0 for i = 1,…, M  // eigen values
8   V[i]= vi such that Cvi = λivi for i = 1,…, M // eigen vectors
9   for i=1 to |gallery| do:
10      let Γ∈gallery                            // select a face from gallery
11      gallery = gallery - {Γ}
12      facespace[i][1]= Γ                       // store face
13      facespace[i][2]=Project(Γ)               // projected face
14  return facespace
```

Figure 7. Pseudo-code for Eigenfaces initialization.

Throughout these methods, let each face image in **probe**, **gallery** and **training** have N pixels and the number of face images in **training** be M.

Figure 7 contains pseudo-code for the *Setup*() method, which creates an Eigen "face space" based on the faces in **training** and projects images from **gallery** into the face space. Here is a description of how *Setup*() works. An "average face," Ψ, is computed across the M face images in **training**, as shown in line 2. The "average face," Ψ, is subsequently taken away from each image, Γ_ι in **training**, yielding difference faces, Φ_i, in lines 3 and 4. The set of these "difference" faces is an N by M matrix, **A**, as shown in line 5.

A covariance (or scatter) matrix is the product of the transpose of **A** with itself $\mathbf{AA^T}$, but due to the structure of face recognition data, namely, $M<<N$, the variable **C** is computed using $\mathbf{A^TA}$, yielding a square matrix of dimension M. The eigenvectors (or "eigenfaces") of **C** are determined by satisfying the characteristic equation, where the determinant of the matrix less the eigenvalues along the diagonal elements is equal to zero (using the method of Lagrange multipliers). See lines 7 and 8 of Figure 7. This approach is called the Snapshot method [19].

Algorithm: *Project*(Γ)

Input: Face image Γ

Output: a face image, which is Γ in the "face space" characterized by the previously derived eigenvectors and average face from the training set.

Uses: Eigenvectors **V** and average face Ψ

Steps

```
Ω[i] = V[i]^T(Γ-Ψ)  for i = 1,…, M      // store weights using only top eigenvectors
return Ω
```

Figure 8. Projecting a face image into Eigenfaces' face space.

"Eigenfaces" is the resultant multidimensional face-space characterized by the Eigen vectors and the average face. All images in **gallery** are projected into the face space and their original and projected images saved in a matrix, **facespace**; see lines 9 through 14 in Figure 7. Figure 8 shows the expression for projecting an image into eigenfaces.

Algorithm: *Recognize*(Γ)

Input: Face image Γ

Output: a copy of **facespace** where the rows are sorted in order of closeness to Γ's projected face. The most similar face appears in the first row and the least similar face appears in the last row.

Uses: **facespace** and a distance function *dist*()

Steps

```
1  O = Project(Γ)                // project into face space
2  let match = facespace         // make a copy
3  Sort match by row based on dist(O, match[][2])
                                 // sort projected images
4  return match
```

Figure 9. Recognizing a face image using Eigenfaces.

The result of the initialization process is **facespace**, the average face, Ψ, and the eigen vectors **V**. These are used by *Recognize*(), in Figure 9, to match faces as follows. Face images in **probe** are projected through the eigenfaces in line 1. The distance between the projected probe image and each projected gallery image is computed, typically using Euclidean distance or Mahalanobis distance. A face, $\Gamma \in$ **probe**, is recognized as the face in **gallery** whose projected image is closest to Γ's projected image. *Recognize*() returns all the images in **gallery** sorted by closeness of its projected images to Γ's projected image; see line 3. The resulting vector provides a rank order of best matches.

4. *k*-Same-Pixel and *k*-Same-Eigen

Eigenfaces() and its supporting methods are used by the algorithms *k*-Same-Pixel and *k*-Same-Eigen, which are described in this section. These *k*-Same algorithms are effective at limiting the ability of face recognition software to reliably recognize faces, even when Eigenfaces is not the face recognition software used. Enforcing *k*-anonymity provides this assurance. These algorithms are not optimal *k*-Same solutions because there may exist other equally effective de-identification functions that maintain more detail in the de-identified face images.

4.1 *k-Same-Pixel*() and *k-Same-Eigen*()

Figure 10 presents the *k-Same-Pixel*() algorithm. Given an original person-specific face set, **H**, and a value *k*, the algorithm returns a *k*-Same face set over **H** with respect to *k*. It begins by using Eigenfaces' *Setup*() routine in line 1. **H** is used to generate an overall average face and the projected gallery images against which faces will be recognized.

Each face in **H** is de-identified in lines 4 through 10 as follows. For a given face, *Recognize*() provides a matrix containing all the face images, not yet de-identified, sorted by closeness to the given face. The top *k* rows of the matrix, representing the *k* closest faces, are averaged together in step 7. *k* copies of this average face are added to the solution set, $\mathbf{H_d}$, in line 8. The *k* faces are then removed from **H** (line 9) and from **facespace** (line 10) because they have all been de-identified. Upon each iteration of the loop, the number of faces to de-identify is reduced by *k*. In case the number of original face images in **H** is not evenly divisible by *k*, an adjustment to *k* is made in line 5 on the last iteration, to group up to 2*k*-1 faces together. When execution concludes, the resulting face set, $\mathbf{H_d}$, has at least *k* occurrences of each face image contained within it. Each image in $\mathbf{H_d}$ is the average of *k* closest faces.

```
Algorithm: k-Same-Pixel(H, k)
Input:   Person-specific face set named H and a k
         privacy constraint
Output:  a k-Same face set named H_d
Assumes: |H| ≥ k
Uses:    Eigenfaces pseudo-code and its facespace variable
Steps
1     Setup(H, H)                                    // H is gallery and training
2     let H_d be an empty multi-set
3     for Γ∈H do:
4         match = Recognize(Γ)
5         if |H| < 2k then k = |H|                   // group al
6         let closest be the face set containing match[1][1],…,match[k][1]
7         Ψ = Average(closest)
8         Add k copies of Ψ to H_d
9         Remove faces match[1][1], …, match[k][1]  from H
10        Remove rows match[1][], …, match[k][] from facespace
11    return H_d
```

Figure 10. De-identification algorithm *k*-Same-Pixel averages *k* closest gallery images.

In *k-Same-Pixel*(), Figure 10 lines 6 and 7, the de-identified image for *k* closest faces is the pixel-wise average of the original face images. A variation is to use the projected images as the basis for averaging. This is done in *k*-Same-Eigen. Line 6 is replaced in Figure 10 with the line appearing in Figure 11. That is the only difference between *k-Same-Pixel*() and *k-Same-Eigen*(). Images provided by *k-Same- Eigen* () have a blurred effect compared to those from *k-Same-Pixel*() because the projected images used in *k-Same-Eigen*() retain only the most important characteristics. Less important components have been removed.

```
6         let closest be the face set containing match[1][2],…,match[k][2]
```

Figure 11. De-identification algorithm *k*-Same-Eigen is same as *k*-Same-Pixel in Figure 10 except for this replacement of line 6, which averages *k* closest projected images.

Example 4.1 (*k*-Same-Pixel and *k*-Same-Eigen Face Images) Figure 17 shows face images de-identified by *k-Same-Pixel*() and *k-Same-Eigen*() over the same subjects for k=2, 3, 5, 10, 50 and 100. The number of faces in the original face set is 200. The *k-Same-Pixel*() images are less blurred.

4.2 Correctness of *k-Same-Pixel*() and *k-Same-Eigen*()

Let **H** be a person-specific face set, $|\mathbf{H}|>1$, k be a privacy constraint, $k>1$, $|\mathbf{H}|\geq k$, and $\mathbf{H_d}$ be the result from *k-Same-Pixel*(**H**,*k*), or alternatively from *k-Same-Eigen*(**H**,*k*). Theorem 1 shows that $\mathbf{H_d}$ is *k*-anonymized over **H**. Theorem 2 shows that *k-Same-Pixel*() is effective at thwarting face recognition software. There may exist some other *k*-same de-identification function, other than *k-Same-Pixel*(), however, that can have less information loss. These points are discussed in this subsection.

Theorem 1. If **H** is a person-specific face set, $|\mathbf{H}|>1$, k is a privacy constraint, $k>1$, $|\mathbf{H}|\geq k$, and $\mathbf{H_d}$=*k-Same-Pixel*(**H**,*k*), then $\mathbf{H_d}$ is *k*-anonymized over **H**. [1]

Proof. Figure 10 contains pseudo-code for *k-Same-Pixel*(). In each iteration of the loop contained in lines 3 through 10, k faces are added to $\mathbf{H_d}$ (see line 8), and k faces are removed from **H** (see line 9). On the last iteration, k is adjusted from its original value, k_0, if needed, to be the number of faces remaining

[1] Theorem 1 also holds for *k-Same-Eigen*(**H**,*k*).

(line 5) in **H**. On the last iteration, $k_0 \le k < 2k_0$. When the loop ends, the number of faces originally in **H** is the number of faces in $\mathbf{H_d}$. On each iteration, k copies of the same averaged face (Ψ in line 7) are added to $\mathbf{H_d}$ (line 8), so the k copies are indistinguishable. The k copies of the average face (Ψ) have a one-to-one correspondence to the original k face images in **H** (see lines 6 and 7) that composed Ψ. In summary, (1) $|\mathbf{H}|=|\mathbf{H_d}|$; (2) for each $\Gamma \in \mathbf{H}$ there exists $\Psi \in \mathbf{H_d}$; and, (3) for each $\Psi \in \mathbf{H_d}$, there are k original faces on which it was based. □

Theorem 2. If **H** is a person-specific face set, $|\mathbf{H}|>1$, k is a privacy constraint, $k>1$, $|\mathbf{H}| \ge k$, $\mathbf{H_d}$=*k-Same-Pixel*($\mathbf{H}$,k), and $\Gamma_d \in \mathbf{H_d}$, there cannot exist any face recognition software for which the subject of Γ_d can be correctly recognized in **H** with better than $1/k$ probability. [2]

Proof. The proof is straightforward, but is expounded to demonstrate characteristics of the privacy protection provided. Let $g:\mathbf{H_d} \rightarrow \mathbf{H}$ be a software recognition program, and $\Psi_d \in \mathbf{H_d}$. Let **P** be a face set containing the subjects of Ψ_d; that is, $\mathbf{P}=\{\Gamma_i \mid \Gamma_i \in \mathbf{H}$ and *k-Same-Pixel*$(\Gamma_i)=\Psi_d$, for $i=1,\ldots,k\}$. There are 3 cases to consider.

Case 1 (k best choices are all onto **P**): each $g(\Psi_d)$ is a vector whose best matching k faces are the same (and only the same) as the faces in **P**. The k faces in **P** are the subjects of the k occurrences of Ψ_d. The Ψ_d's are indistinguishable, so correctly assigning ("recognizing") a Ψ_d to the face in **P** that was its subject has probability $1/k$.

Case 2 (k best choices are not all onto **P**): Let **A** be the face set containing the k best matches of $g(\Psi_d)$. **A** includes some faces in **P** and some not in **P**. Assigning ("recognizing") a Ψ_d to the face in **A** that was its subject is probability 0 if the subject is not present in **A** and $1/k$ otherwise.

Case 3 (matching each face in **P**): if **P** is known and $g(\Psi_d)$ claims its best match to be Γ, $\Gamma \in \mathbf{P}$, then for each of the k faces $\Gamma_i \in \mathbf{P}$, declaring Ψ_d to be recognized as each Γ_i will only be correct once, so the probability of a correct recognition is $1/k$. If $\Gamma \notin \mathbf{P}$, then the probability of a correct recognition is 0.

In all cases, the probability of correctly recognizing a face was no better than $1/k$. □

Theorem 2 shows that even though methods of *Eigenfaces*() is used by *k-Same-Pixel*() and *k-Same-Eigen*(), their ability to provide k-anonymized solutions is invariant to the face recognition software that attempts to recognize faces from their results. The de-identified faces are not just protected against matrix decomposition-based face recognition technology. *k-Same-Pixel*() and *k-Same-Eigen*() can protect against any face recognition system, including technologies that have yet to be designed. Let **H** be a person-specific face set that is the subject of de-identification. A k-anonymized solution over **H** reduces the number of distinct faces to $|\mathbf{H}|/k$ prior to a recognition attempt. It is impossible to distinguish between at least k faces in **H** for any face in the k-anonymized solution over **H**.

k-Same-Pixel() and *k-Same-Eigen*() are k-Same de-identification functions. However, they are not necessarily optimal in terms of minimal information loss. Let **H** be a person-specific face set that is the subject of k-Same de-identification by *k-Same-Pixel*(). *BlackOut*() maintains no facial details whatsoever and *Average*() aggregates over **H** whereas *k-Same-Pixel*() aggregates over homogeneous clusters of faces in **H**, so *k-Same-Pixel*() is preservative over {*BlackOut*(), *Average*()}. However, there may exist another equally effective de-identification function that has less information loss.

[2] Theorem 2 also holds for *k-Same-Eigen*($\mathbf{H}$,k).

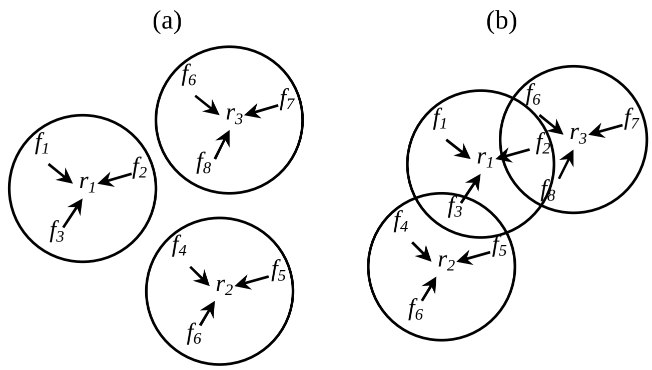


Figure 12. Two scenarios for original faces *f* being partitioned into *k*-sized clusters with centroids *r*, where *k*=3: (a) isolated clusters; (b) overlapping clusters.

Optimal clusters are not necessarily found in *k-Same-Pixel*() or *k-Same-Eigen*(). If each face in a cluster, selected the same, and only the same, set of *k* faces as its closest faces, then *k-Same-Pixel*() would provide an optimal solution. See Figure 12(a). But this partitioning is a special case. Having all clusters in a face set naturally isolated is rare. More often clusters overlap, making the best choice of minimally distant clusters difficult to determine. See Figure 12(b). *k-Same-Pixel*() does not attempt to disambiguate these nested cluster preferences. Instead, one face is randomly selected and its *k* closest faces are grouped together. A different selection of faces typically reveals different clustering choices over the same face set. For these reasons, *k-Same-Pixel*() can provide results with more information loss than is minimal, and *k-Same-Eigen*() can have worse recognition results because of its additional loss of facial details.

4.3 Complexity of *k-Same-Pixel*() and *k-Same-Eigen*()

Earlier this paper reported that the optimal partitioning of faces in a face set into clusters that are minimally distant is NP-hard, and as a result, no such algorithms would run in reasonable time. *k-Same-Pixel*() and *k-Same-Eigen*() do not always make optimal cluster selections, as just discussed, but they do run quickly, in linear time. The composition of *k-Same-Pixel*() and related algorithms in this paper focused on clarity. Some operational efficiency is possible, but as written, these algorithms demonstrate linear time execution.

Let *N* be the number of pixels in the face images and *M* be the number of images in the original face set, *N*>>*M*. The algorithm *k-Same-Pixel*() begins in Figure 10 line 1 with *Setup*(), whose pseudo-code appears in Figure 7. The averaging in line 2 of *Setup*(), takes *NM* steps. The transpose of the matrix and computation of the covariance matrix in line 6 takes M^2 steps. The loop in lines 10 through 13 of *Setup*() executes *M* times, projecting a **gallery** face on each iteration. *Project*() in Figure 8 takes *NM* steps. So, the loop in *Setup*() takes *NM* steps; *Setup*() is characterized by *N*.

A loop constitutes the remainder of *k-Same-Pixel*(), see Figure 10 lines 3 through 10. The loop iterates *M*/*k* times. In each iteration, *Recognize*(), whose pseudo-code appears in Figure 9, executes. Line 1 of *Recognize*() executes *Project*(), which takes *NM* steps. Copying the matrix, in line 2, takes 2*NM* steps. The sort in line 3 takes *M*log*M* steps, but determining the distances between a given face and every other face using Euclidian distance (defined equation 2) takes *N* steps. So, *Recognize*() is characterized as taking *N* steps.

Averaging *k* faces, in line 7 of *k-Same-Pixel*(), takes *Nk* steps. Removal of *k* faces from **H** (line 9) and from **facespace** (line 10) each takes no more than *NM* steps. Overall, because *N*>>*M*, the complexity of *k-Same-Pixel*() and *k-Same-Eigen*() is O(*N*).

5. Experiments

5.1 Materials

Experiments were run on facial images from the U.S. Army's Face Recognition Technology (FERET) database [14]. FERET contains photographs of subjects who volunteered to be photographed over multiple sessions in various locations (but setup the same way to have the same background and lighting) between 1993 and 1996. The FERET database is the largest publicly-available database of facial images collected in a systematic manner and is widely used by face recognition researchers and developers.

The database has 14,126 face stills of 1199 people. Coordinates for the center of the eyes and tip of the nose are provided for each face still. Face images were cropped,rotated, translated, and scaled from the frontal face stills using this information. Each resulting face image has 13,266 pixels, displayed graphically as 99 columns and 134 rows.

For the purposes of this paper, it is necessary to explain two collections of face stills within the FERET database, "fa" and "fb". These refer to two frontal views taken of each subject within five minutes of each other in the same lighting; a different facial expression was requested for the second frontal image. 1005 distinct subjects appear in both fa and fb. Using a single best match result, Eigenfaces recognizes fa faces to fb faces, or vice versa, with greater than 70% correctness (or accuracy), and recognizes fa faces to fa faces, or fb faces to fb faces, with 100% correctness [13].

The code for Eigenfaces is publicly available in several programming languages. This work used MatLab (versions 6 and 6.5) to run experiments, with "calcpca.m" version 2.4 written by Iain Matthews with some additional coding for processing experiments written by Ralph Gross. Both are researchers in the School of Computer Science at Carnegie Mellon University.

5.2 Test Design

In all experiments, training, gallery and probe face sets are person-specific. There is a one-to-one relationship between the subjects in the gallery and probe face sets. Each subject appearing in gallery also appears in probe, and vice versa. Unless otherwise noted, 200 face images were randomly chosen for each experiment; let **H** represent this resulting face set. The probability of correctly recognizing a face image in **H** to a face image in **H** by guessing is 1/|**H**|= 0.005.

Current face recognition algorithms have numerous weaknesses; these include: detecting faces, producing face images from face stills, and, recognizing the same person after a lapse in time. These are all issues independent of the de-identification task. It is crucial to test de-identification functions in such a way as to not exploit these weaknesses in face recognition software. Therefore, all experiments reported herein test recognition using only fa face images to fa face images (or fb to fb). Because Eigenfaces always yields 100% recognition for this setup, these experiments will show how this performance degrades due to de-identification efforts.

The performance of Eigenfaces is: (a) reported by the percentage of correct recognitions having a single best match; and, (b) plotted as the percentage of correct matches in the top e matches, as e goes from 1 (best match) to 200 (total number of images in probe). Figure 14 provides an example. The horizontal axis is e and the vertical axis is the percentage of correct matches. The percentage of correct recognition is about 2%, plotted as the leftmost point on the curve.

To test de-identification techniques, consider the options available to an attacker, who attempts to re-identify (by face recognition software) a de-identified face set. There are three different kinds of attacks, corresponding to three different arrangements of recognizing gallery images to probe images, available. These involve matching: (1) original images to altered images; (2) altered images to original images; and, (3) altered images to altered images. Descriptions of these attacks are provided below.

The first type of attack, in which original images are matched to altered images, is termed **naïve recognition** in the context of these experiments. No action is taken by the attacker to account for the de-identification affect. The de-identified images are just run through the face recognition software. For experimental soundness herein, the gallery in naïve recognition tests, will be only the original face set of the images subject to de-identification. In the general case, outside of this experimentation, the gallery would presumably include faces beyond those de-identified thereby possibly providing even better de-identification results.

The second type of attack, in which altered images (as gallery) are matched to original images (as probe), is termed **reverse recognition** in the context of these experiments. Reverse recognition assumes the attacker already has a face set containing the original images that were the subjects of de-identification. The goal is to determine a correct one-to-one correspondence. By using the altered images as the gallery, the alterations due to de-identification may decompose and become dispersed through some number of principle components, thereby limiting the affects of the alterations when matching faces.

The third type of attack, in which altered images are matched to altered images, is termed **parrot recognition** in the context of these experiments. Many de-identification techniques can be replicated. For example, placing a black band over the eyes can easily be duplicated. The attacker can invoke the same de-identification technique on his face sets, thereby de-identifying them. The training set and gallery of the attacker are then de-identified also, and recognition matches a de-identified gallery to a de-identified probe. For experimental soundness herein, the gallery in parrot recognition tests, will contain only those images appearing in the probe. In the general case, outside of this experimentation, the gallery would presumably include faces beyond those de-identified in the probe, thereby possibly providing even better de-identification results.

5.3 Ad Hoc De-identification Methods do not Thwart Face Recognition Software

There exist a number of ad hoc de-identification techniques that attempt to mask the identity of a subject in a face image. To the human eye a masked face image may look sufficiently de-identified. This subsection reports how several ad hoc de-identification techniques affect the recognition abilities of face recognition software. Other than blocking out the entire image, as done by *BlackOut*(), these experiments show that ad hoc attempts do not thwart face recognition software.

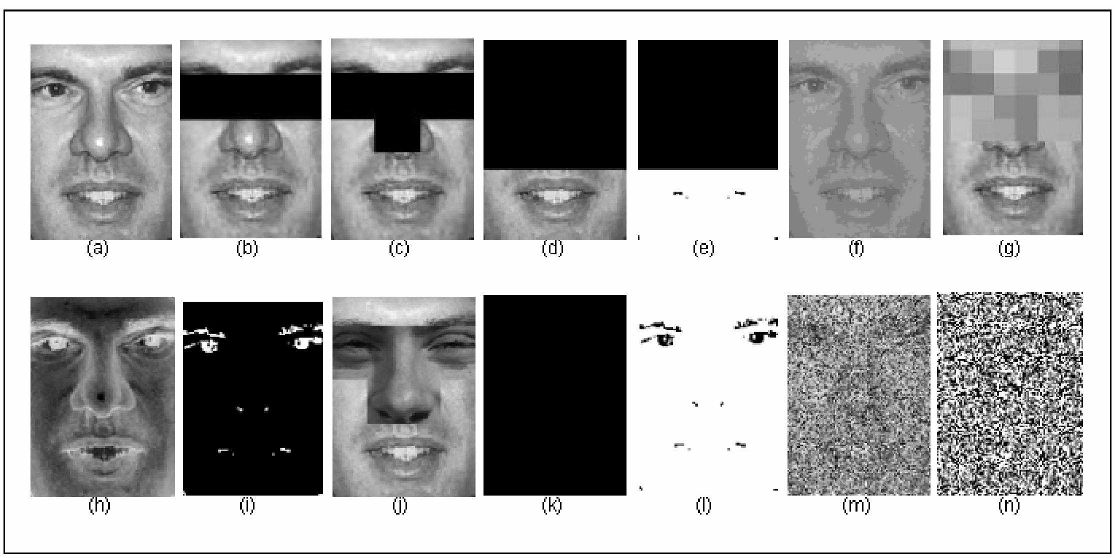


Figure 13. Examples of (a) an original face image and each ad hoc de-identification method: (b) bar mask, (c) T mask, (d) mouth-only in gray scale (e) mouth-only in black and white, (f) ordinal, (g) pixelation, (h) negative in gray scale, (i) negative in black and white, (j) Mr. Potato face, (k) blackout, (l) threshold, (m) random in gray scale, and (n) random in black and white.

This section is a contribution of experiments using face image sets altered by common de-identification and perturbation techniques to the compendium of face recognition research. With the exception of some partial occlusion experiments, where an object randomly occludes part of the face [5], and experiments in facial symmetry [9], ad hoc de-identification methods have not been previously tested against automated face recognition prior to this work. Below are descriptions of the ad hoc de-identification techniques tested and following that are experimental results.

The first four de-identification functions use different types of occlusion, or masking techniques.

Mask. *BlackOut*: colors the entire face image with either a single color or pattern. *Bar masks*: covers the eyes with a single colored band. *T masks*: cover the eyes and nose with a single color. *Mouth only:* leaves only the mouth and chin exposed, and was tested in gray-scale and in black/white.

Pixelation. Reduces the number of distinct pixel values in a face image by replacing a square block of pixel values with their averaged value. Experiments were conducted on square pixel blocks of 15, 20, and 30 pixels across.

Negative. For grayscale, change each pixel value *x* to the value 255-*x*, which flips a value to its equivalent on the opposite end of gray-scale. For black/white, each black pixel (0) becomes white (255) and vice versa.

Mr. Potato Face. Replace regions of the face image namely, eyes, nose, and mouth areas, with those from other faces in gallery. Also known as mix and match.

Random Noise. In black/white images, choose a random pixel position to flip. In gray-scale images, a random value between 0 and 255 replaces a set of randomly chosen pixel positions. The same set of pixels is randomly perturbed in each image.

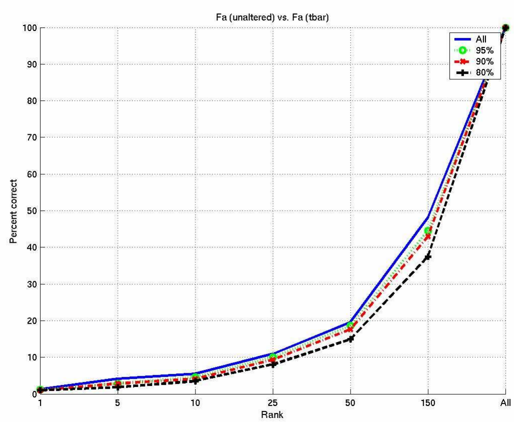


Figure 14. Performance of naïve recognition, unaltered to T-mask images, using Eigenfaces. Best match recognition is 1%.

The next two de-identification functions remove variation within pixel values by changing the rank or ordinal numbers.

Ordinal. Removes the variation within a face image by converting the value of each pixel to one of five bin values. This creates face images, whose number of color is between two (black/white), and gray-scale (256 values).

Threshold. Transforms all gray-scale pixels to either a black or white value (0 or 255, respectively) depending upon a threshold value chosen in the range of 0 to 255. In additional experiments based on black/white techniques, a threshold value of 65 achieved 100% recognition; see Figure 15.

Many ad hoc de-identification methods are capable of defeating naïve recognition, which matches original (gallery) to altered (probe) images. Figure 14 shows the performance of Eigenfaces in recognizing face images de-identified by T mask. Table 1 shows best match results using naïve recognition on images from different ad hoc techniques. Recall, 100% correct recognition was realized when matching original-to-original images. Masking deteriorates this performance. Only 2% correct recognition was realized when matching original to bar masked images. Figure 14 shows only 1% correct recognition when matching original to T-masked images.

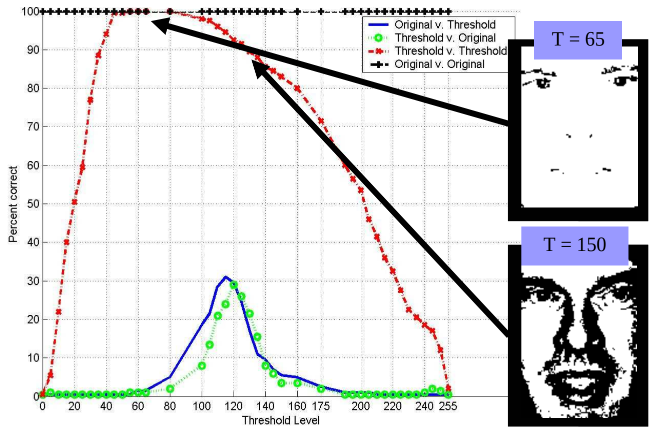


Figure 15. Eigenfaces recognition based on best matches using threshold images.

Pixelation is an exception: 99% correct recognition was realized when matching original to pixelated images; see Table 1. Despite looking somewhat de-identified to humans, see Figure 13(g), and despite pixelation's common use on television to hide faces during interviews, pixelation has virtually no effect on thwarting naïve face recognition software. Pixelation is not the only exception. Figure 15 shows a maximum of 30% correct recognition when matching original to threshold images with a threshold of 118.

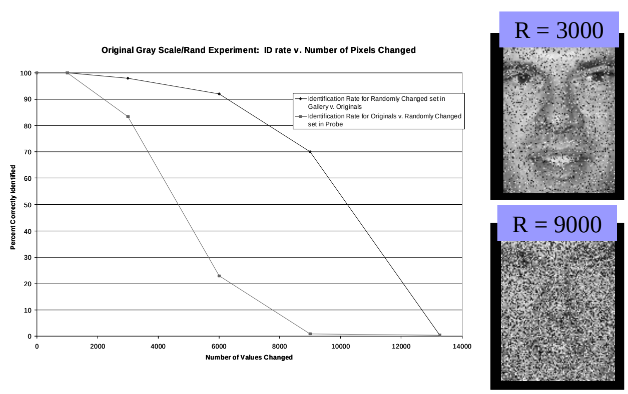


Figure 16. Eigenfaces recognition based on best matches using images perturbed by adding random noise.

Figure 16 shows the results of naïve recognition on images in which random noise was added. The bottom curve shows the recognition rate when matching original to noisy images. When 9000 of 13,266 (or 68%) of the pixels were made noisy, the recognition plummeted to 1%, but at 6000 pixels perturbed, correct recognition was 23%. Random noise in black/white images decreases recognition after approximately half of the pixel values were perturbed.

Reverse recognition, which matches altered (gallery) to original (probe) images, performs the same as naïve recognition on most of the ad hoc de-identification techniques tested. There are exceptions. In threshold de-identification, Figure 15 shows a maximum of 30% correct recognition at a threshold of 118 using naïve recognition. Reverse recognition shifts the curve to the right and slightly down. A maximum of 29% correct recognition resulted at a threshold of 120 using reverse recognition. More noticeable differences appear with additive random noise. The bottom curve in Figure 16 shows the performance of naïve recognition, and the top curve shows the performance of reverse recognition. With 68% of the pixels made noisy, naïve recognition was only 1%, but reverse recognition was 70%.

In parrot recognition, Eigenfaces was re-trained on face images having the same alteration and the gallery was also de-identified. Parrot recognition circumvents most of the ad hoc de-identification techniques tested. More than 99.8% correct recognition was realized when matching altered to altered images. See Table 1.

Table 1. Percentage of correct best match recognitions for various ad hoc de-identification methods tested with: original to altered images, modeling naïve recognition attempts; and, altered to altered images, modeling parrot recognition attempts.

Altered Face Image	Orig. to Alter. (naïve)	Alter. to Alter. (parrot)
Original (unaltered)	100%	100%
Bar mask	2%	100%
T mask	1%	100%
Black out	0%	0%
Pixelation	99%	100%

There are exceptions. BlackOut, which provides no information, retains results no better than guessing with parrot recognition. For threshold and additive random noise, results for various levels are shown in Figures 15 and 16 (for gray-scale), respectively. In Figure 15, the threshold experiments of altered to altered (the highest curve in Figure 15 marked with x's) show that parrott recognition remains above 10% within the threshold values of 5 and 250, which are the extremities of the gray-scale.

These experiments demonstrate that many ad hoc de-identification techniques may look convincing to human eyes, but in general, they provide little or no protection from face recognition software. With the exception of BlackOut, significant numbers of images were correctly recognized by one or more of the attacks described.

5.4 *k*-Same Thwarts Face Recognition Software

Figure 17 displays visual results of *k*-Same-Pixel (top row) and *k*-Same-Eigen (bottom row) for k= 1, 2,3,5,10,50, and 100 from left to right.

Figure 17. Face images from *k*-Same-Pixel (top row) and k-Same-Eigen (bottom row) for *k*= 1, 2, 3, 5, 10, 50, and 100 from left to right.

The performance of naïve recognition, which matches original (gallery) to altered faces de-identified by *k*-Same-Pixel and *k*-Same-Eigen (probe), is provided in Figure 18. Both algorithms were tested for *k*=2, 3, 5, 10, 50, and 100, with person-specific face sets of 805 face images. The averaged results are plotted along with the expected value at each *k*. Correct recognition results from *k*-Same-Pixel and *k*-Same-Eigen experiments appear below the ideal value of 1/*k*. The expected value, 1/*k*, and the actual performance of *k*-Same-Pixel and *k*-Same-Eigen have an average difference of -1.6% (each) at *k*=2 and -63% and -81%, respectively, at *k*=100. The *k*-Same-Pixel and *k*-Same-Eigen results have an average absolute difference of 0.0017, compared on the same data set for 20 runs. The average absolute difference across the two different data sets is found to be 0.0050, which is larger than that between the two algorithms. Similar recognition results of less than 1/*k* were found with reverse recognition and parrot recognition.

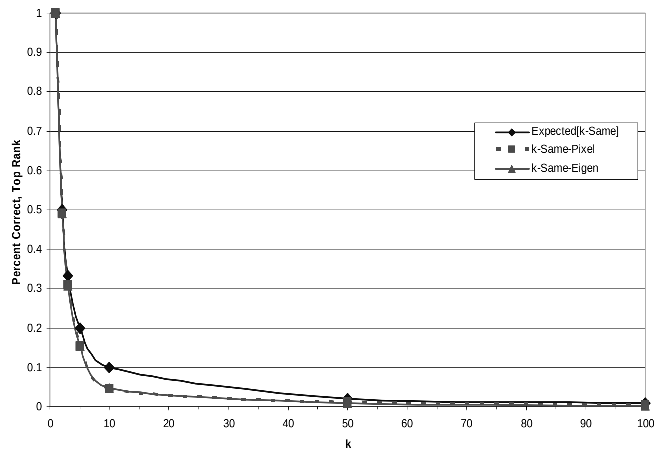


Figure 18. Performance of naive recognition on faces de-identified by *k*-Same-Pixel and *k*-Same-Eigen. Results based on best match ("top rank"). The ideal upper bound recognition rate of 1/*k* is also plotted.

These experimental results demonstrate *k*-Same-Pixel and *k*-Same-Eigen's ability to provide k-anonymity protection, and in so doing, thwart face recognition software from reliably recognizing their de-identified faces.

6. Related Work in Privacy Preserving Data Mining

This work can be viewed as privacy preserving data mining in high-dimensional data. For the past several years, research in privacy preserving data mining techniques has been split into several areas of research: secure multiparty computation, rule hiding, and perturbation techniques. Here is how face de-identification fits in. Face de-identification algorithms seek to protect data that must be shared outside the original collecting institution, so secure multi-party computation has no bearing.

The goal in rule hiding [3, 17, 18] is to prevent sensitive rules from being revealed in shared data. One way to determine sensitive information for rule hiding in facial images might be to consider the principle components of the eigenface decomposition. A dimension-reducing algorithm for face recognition, such as eigenfaces, does not permit the generation of easily understandable rules. In the decomposition process, sensitive features can become dispersed through a considerable number of the principle components. If rules are considered as the eigenvectors of decomposition, then controlling for certain eigenvectors, through a technique such as suppressing them, may protect the identity of certain faces. However, different principle components, or a combination of such, can be sensitive for different faces and one cannot suppress all principle components. While there is room for research, there appears no simple way to classify the features of an image or its decomposition into sensitive and non-sensitive rules for rule hiding techniques. This remains a challenge to the research community.

Perturbation may provide an applicable privacy preserving method for de-identifying faces. Perturbation approaches to privacy preserving data mining have been addressed in previous research, namely [1, 2, 16]. The basic design and premise of a perturbation technique is as follows. The original data collecting institution has a distribution of data *X*. From this data, the institution will produce a new distribution $Y = X + Z(\Theta)$, where $Z(\Theta)$ is some known controlled perturbing distribution, such as a guassian distribution with mean zero. The institution releases the new distribution *Y* and the distribution used for the perturbation process $Z(\Theta)$, instead of the original dataset *X*. The goal for individuals, or institutions outside of the original institution, is to learn rules, patterns, or classifiers similar to those that could be learned from the unperturbed data, usually through some expectation maximization technique.

In de-identifying faces, one could consider the set of original faces as *X*, and the set of perturbed faces as *Y*. The problem with such an approach in facial images is that modern face recognition software is highly insensitive to noise in the images. This was demonstrated in the experimental results reported in the previous section by taking $Z(\Theta)$ to be random noise on individual pixels of the original face images. For black and white images, perturbing noise is the flip of a color from black to white and vice versa. For grayscale images, perturbing noise is any value on the gray scale [0, …, 255]. As demonstrated in the experiments above, random noise has little effect on the ability to dampen the recognition of de-identified faces by Eigenfaces, until roughly half of the pixel values have been flipped in black and white images or three-quarters of the pixels are changed to random values in gray-scale images. Facial details remaining in the de-identified images may appear horribly obscured due to the amount of perturbation necessary to sufficiently thwart recognition.

There may exist different types of perturbation beyond additive random noise that would provide reasonable face de-identification. Using more complex perturbation models for face de-identification is an open research question.

k-same algorithms offer a new kind of privacy preserving technique for dimension-reducing data mining algorithms. More work is needed to determine how variants of *k*-Same algorithms generalize to privacy preservation in high-dimensional data other than face images.

7. Related Work on Formal Protection Models

Formal protection models provide a means of characterizing scientific assurances for a privacy protection schema. In foundational work on computational disclosure control (or "data privacy") [21], definitions for several formal protection models are presented. Two models, *wrong*-map and *k*-anonymity, are of relevance to this discussion. As shown earlier in Theorem 1 and in the proof of Theorem 2, *k*-Same algorithms derive their privacy protection by adherence to *k*-anonymity, which guarantees that each de-identified face relates ambiguously to at least *k* original faces. When faces are optimally clustered for constructing *k*-samed faces, then a *k*-same algorithm provides an optimal solution such that the maximal amount of facial detail remains in the de-identified image. If the set of *k*-samed faces are not constructed from an optimal clustering of faces, then a computer will need more than *k* choices to find the faces that make up a *k*-same face. See Figure 12 for depictions of clustering scenarios.

An alternative to *k*-anonymity is *wrong*-map, which with respect to face recognition, can be described as follows. Every de-identified face is guaranteed a 0 probability of correct recognition. One way to achieve wrong-map protection using the *k*-Same clustering approach is to exploit overlapping clusters so that the de-identified face maximizes the number of possible overlapping clusters that could compose the face. The resulting face replaces a face not in the cluster that composed it but one near that cluster. Achieving wrong-map protection while maximizing the facial details that remain is an open research question.

8. Privacy Implications

This paper concludes in this section by addressing: (1) what is needed for this work to de-identify video clips; (2) what practices are needed to deploy this kind of work; and, (3) how this work relates to U.S. policy on data sharing.

Consider the real-world task of de-identifying a surveillance video clip. *k*-Same algorithms provide a key ingredient to accomplish this, but to construct an operational system, more work is needed. Consider *k*-Same algorithms to be a de-identification tool, which has as input, a person-specific face set, and as output, a *k*-same de-identified face set. There is a pre-processing step needed to convert facial representations in the video clip to a person-specific face set. There is also a post-processing step needed to place de-identified faces into a video clip so that the behaviors and actions of the people remain from the original video clip, but the de-identified faces replace the original facial representations. For example, a person coughing in the original video clip should be a de-identified person coughing in the final video clip. If needed, post-processing can also apply encrypted identifiers to the de-identified faces, so that given a proper key, the original face for a person is revealed. Both the pre-processing and post-processing steps contain active areas of research in computer vision, computer graphics and face recognition.

The basis of privacy protection of *k*-same algorithms is *k*-anonymity. There are numerous possible attacks on *k*-anonymity with known solutions [20]. One of the biggest problems is determining the proper privacy constraint *k*. The selection of *k* depends in part on whether protection must also prohibit the ability for all *k* individuals to be known. For example, for a *k*-samed face, all *k* subjects may be identified by name. Even though determining which named person appears in which frames in the de-identified video clip cannot be done exactly, the *k* people can all be identified and acted upon as a group. With other kinds of data, *k* is prescribed by policy, but even still, care must be taken.

De-identification, as described in this work, is not a guarantee of anonymity. This work seeks to thwart face recognition for the purpose of limiting automatic persistent recognition of populations whose images are captured on video but who have done nothing suspicious. While these techniques thwart face recognition software, for example, correct identification is still possible by recognizing clothes, behavior, or co-occurrences with other people.

To conclude, the broader policy setting is examined. U.S. policy typically treats the privacy of person-specific data in a binary fashion: either the data collector holds data exclusively (or almost exclusively); or data are shared with few restrictions. Data de-identification, in which an individual's identity cannot be determined in the data, enables a spectrum of in-between sharing policies. Using de-identification, data can be shared with scientific assurances of anonymity while the resulting data remains practically useful. The Privacy Act of 1974 and the U.S. Supreme Court decision *Whalen v. Roe* (1977) appear to support scientific de-identification [23, 25]. In the case of the latter, this is presumably because de-identification reduces risk while improving utility.

In terms of video surveillance data, having society choose between either sharing video data as it is collected or prohibit its sharing altogether, can lead to polarized policies. Recently, the Virginia State House passed a bill requiring law-enforcement agencies to apply for a highly specified court order prior to employing face recognition software. (The Virginia State Senate has not yet voted on the bill.) De-identification technology allows data to be shared more freely while providing privacy protections that such legislation seeks.

9. Acknowledgements

The authors thank Ralph Gross, Granger Morgan, and Yiheng Li for insightful discussions. Work by Luis Ahn and Manuel Blum on programs that tell computers and humans apart inspired the authors' earliest approaches to this work. The authors are grateful for support from Sherice Livingston, Joe Lombardo and Julie Pavlin and thank the ESSENCE II System and the U.S. Department of Defense Global Emerging Infections System for motivation. This research was sponsored in part by the Defense Advanced Research Projects Agency and managed by the Naval Sea Systems Command under contract N00024-98-D-8124.

References

[1] R. Agrawal and S. Ramakrishnan. Privacy-preserving data mining. In *ACM SIGMOD*, Dallas, pp. 439-450, May 14-19, 2000.

[2] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *ACM Symposium on Principles of Database Systems*, pp 247-255, Santa

Barbara, May 21-23 2001.
[3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proc of IEEE Knowledge and Data Engineering Workshop Workshop (KDEX)*, pp 45-72, Chicago, Nov 1999.
[4] [1] D. Blackburn, J. Bone, and P. Phillips. *Face Recognition Vendor Test* (*FRVT) 2000 Evaluation Report*. NIST Technical Report, 2001. Available online at:
[2] http://www.frvt.org/FRVT2000/documents.htm
[5] R. Gross, J. Cohn, and J. Shi. Quo Vadis Face Recognition. Third Workshop on Empirical Evaluation Methods in Computer Vision, *IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, 2001.
[6] Institute for Applied Autonomy (IAA). iSee Project: survey and web-based application charting the locations of closed-circuit television (CCTV) surveillance cameras in urban environments. 2003. Available at: http://www.appliedautonomy.com/isee/info2.html.
[7] M. Jones. All eyes are on Oceanfront's new surveillance system. *The Virginian-Pilot*. September 10, 2002.
[8] T. Kanade. *Computer Recognition of Human Faces.* Birkhauser, Basel and Stuttgart, 1977.
[9] Y. Liu, K. Schmidt, J. Cohn, and R. Weaver. Facial Asymmetry Quantification for Expression Invariant Human Identification', *International Conference on Automatic Face and Gesture Recognition*. Washington D.C., 2002.
[10] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press. Berkeley: Vol. 1, pp. 281-297.
[11] H. Moon and P. Phillips. "Computational and performance aspects of PCA-based face recognition algorithms," *Perception*, 2001, Vol. 30, No. 3, pp. 303 - 321.
[12] A. Oganian and J. Domingo-Ferrer. On the complexity of microaggregation. In *Proc Joint Economic Commission for Europe, Work Session on Statistics Data Confidentiality*. Skopje. March 2001.
[13] P. Phillips, H. Moon, S. Rizvi, and P. Rauss. The FERET Evaluation methodology for face-recognition algorithms. *IEEE Trans. PAMI*, 22 (10), 2000.
[14] P. Phillips, H. Wechsler, J. Huang, and P. Rauss. The FERET database and evaluation procedure for face recognition algorithms. *Image and Vision Computing Journal*, 16(5): 295-306, 1998.
[15] J. Reeves. Tampa Gets Ready For Its Closeup. *Time*. 16 July 2001.
[16] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *VLDB*, Hong Kong, July 23-26, 2002.
[17] Y. Saygin, V. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. In *SIGMOD Record*. 30 (4), 2001.
[18] Y. Saygin, V. Verykios, and A. Elmagarmid. Privacy preserving association rule mining. In *Proc 12th Int'l Workshop on Research Issues in Data Engineering (RIDE)*, Feb 2002.
[19] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Optical Society of America* 4, 1987, 519–524.
[20] L. Sweeney. *k*-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems,* 10 (5), 2002; 557-570.
[21] L. Sweeney. Computational disclosure control: A primer on data privacy protection. Ph.D. Thesis, M.I.T. Cambridge, MA. 2001.
[22] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*. 3 (1): 71-86, 1991.
[23] Turkington, R. and Allen, A. *Privacy Law*. West Group: St. Paul, 1999.
[24] B. Wall. Digitizing facial features fails to prevent identification of plaintiff. *USA Today*. March 10, 1996.
[25] J. Woodward, K. Webb, E. Newton, M. Bradley, D. Rubenson, et al. *Army Biometric*

Applications: Identifying and Addressing Sociocultural Concerns. MR-1237-A. RAND: Santa Monica, CA, 2000.