De-identifying Facial Images using k-anonymity

Ori Brostovski

March 2, 2008

Outline

Photographs reveal information

- From family pictures, to news broadcasts to security cameras in a shop, we are photographed all the time, sometimes **without our agreement**.
- While the photographer may not have an interest in knowing our exact identity, his photos may reveal this information to others.
- The existence of image processing software allows a hostile side to automatically scan a picture for a specific person.

A solution

- The most commonly used physical feature for attach an identity to a person is the face.
- We can use image processing to aid us in protecting the anonymity of photographed people by hiding their faces, in fact this is already being done today. This is called **face de-identification**.

Why not blackout all faces?

- We wish to retain same facial information about the photographed people, for example a fashion shop interested in knowing how many of it's visitors are male or female.
- Sometimes, the photograph may be used psychologically and should therefore look more visually appealing (For a photo of a war victim, a blurred face would be much more effective in raising awareness then a black rectangle).
- Because we can achieve results as good as blackout but better looking.

The contribution presented

This presentation's contribution will be:

- Formalize the problem of face de-identification.
- Explore attacks against existing de-identification schemes.
- Devise a new de-identification scheme with stronger guarantees behind it.

Assumptions

This presentation will concentrate on a setting with the following assumptions:

- There is a canonical face representation.
- The canonical representations of the faces to be de-identified are in some set H which is public.
- Personal info can't be obtained from clothes or non-facial body parts.

Images in general

For our purposes, we will define images to be N-dimensional vectors whose coordinates are between 0 to 255.

Face Stills and Face Images

- A Face Still is an image which includes a single person's face, note that in most cases we can cut an image to a few small Face Stills.
- A Face Image is generated from a Face Still by cropping out the face, rotating it and normalizing it. A face image acts as the canonical representation for a face.

Distance

In many cases we would like to measure the similarity between two images I and J, to do this we will use the standard Euclidean metric (note, we assume that the images agree on dimensions, otherwise we can re-scale):

$$\text{euclid}(I, J) = \sqrt{\sum_i (I[i] - J[i])^2}$$

Face recognition software

- Let f be a function defined as:

$$f : \text{all possible images} \longrightarrow H$$

where

$$f(x) = \text{the image in } H \text{ that is closest to } x$$

- f is called face recognition software.
- For convenience, we will sometimes think of f as accepting a tuple of x-s and returning a tuple of faces in H.

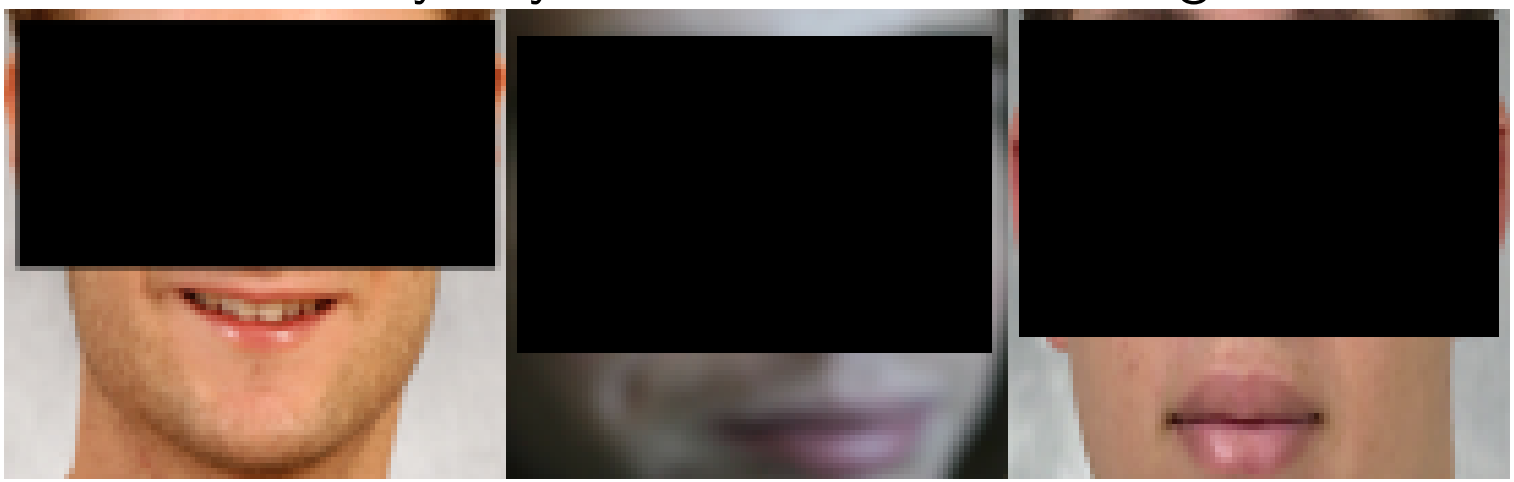
Outline

Popular De-identification algorithms

Here are a few examples for de-identification techniques used today:

Most of the popular algorithms are ineffective

In the following picture, there are three pictures on which one of the de-identification algorithms was used. One of those pictures is a picture of the nice person who wrote this presentation, with some effort many of you would be able to recognize him.



If a human can bypass the de-identification so easily, what about a machine?

Principal component analysis

- PCA is a common technique used for identifying faces.
- The main problem with face images is that they are high dimension vectors.
- PCA allows us to project these vectors to a lower dimension space whose basis is composed of vectors which are more **variant** between different faces.

Principal component analysis

- Let $X = (x_1, x_2, \dots)$ be a matrix whose columns are the high dimensional vectors we want to project. We will assume that $\sum_i x_i = 0$.
- We define a column vector w_1 as

$$w_1 := \underset{\|w\|=1}{\operatorname{argmax}} \operatorname{var}(w^T X) =$$

$$\underset{\|w\|=1}{\operatorname{argmax}} E((w^T X) .* (w^T X))) =$$

$$\underset{\|w\|=1}{\operatorname{argmax}} \sum_i \langle w, x_i \rangle^2$$

One can think of w_1 as the vector in whose direction the variance of the x_i-s is the highest.

Principal component analysis

- We now define $\hat{X}_1$ as

$$\hat{X}_1 = X - w_1 w_1^T X =$$
$$X - w_1 \begin{pmatrix} \langle w_1, x_1 \rangle & \langle w_1, x_2 \rangle & \dots \end{pmatrix} =$$
$$X - \begin{pmatrix} \langle w_1, x_1 \rangle w_1 & \langle w_1, x_2 \rangle w_1 & \dots \end{pmatrix} =$$
$$\begin{pmatrix} x_1 - \langle w_1, x_1 \rangle w_1 & x_2 - \langle w_1, x_2 \rangle w_1 & \dots \end{pmatrix}$$

- $\hat{X}_1$ is the orthogonal complement of w_1 since

$$\forall i : \langle w_1, x_i - \langle w_1, x_i \rangle w_1 \rangle = \langle w_1, x_i \rangle - \langle w_1, x_i \rangle \langle w_1, w_1 \rangle = 0$$

- We can now repeat the process to extract a w_2 for $\hat{X}_1$ and create a $\hat{X}_2$ and so on... until we have enough w_i-s.
- The w_i-s are called **Principal components**.

The eigenfaces algorithm

The eigenfaces algorithm gets three parameters:

- **training** - This is the set of face images from which principal components are extracted. Note that the mean of these faces is not necessarily zero, in which case we will call the mean ψ and reduce it from each face image in **training**.
- **gallery** - These are all the possible faces against which we can find a match.
- **probe** - This is a set of face images that we want to match against one of the faces in **gallery**. For simplicity purposes, we will assume $|\textbf{probe}| = 1$.

The Eigenfaces algorithm

The algorithm works by as following:

1. To force the mean of the faces is zero, we define ψ to be the mean of **training** and reduce it from every image in **training**.
2. We extract principal components from **training**, we call the space created by the principal components the **eigenfaces space**.

The Eigenfaces algorithm

The algorithm works by as following:

1. To force the mean of the faces is zero, we define ψ to be the mean of **training** and reduce it from every image in **training**.
2. We extract principal components from **training**, we call the space created by the principal components the **eigenfaces space**.
3. For each face image I_i in **gallery**, we create an image J_i by reducing ψ from it and projecting it to the eigenfaces space.
4. We can now take the face image **probe** and reduce ψ from it and project it as well. After this is done, we can say that the closest face image to **probe** is the J_i whose distance from it is the lowest, the 2nd closest face image to **probe** is the J_i whose distance from it is the second lowest, and so on.

Using Eigenfaces to attack the popular de-identification algorithms

- We will define H_d to be a collection composed of de-identified versions of the images in H.
- The challenge of the attacker is to find out which face image corresponds to which de-identified face image.

Attacks using Eigenfaces against the popular de-identification algorithms

- **Naive recognition** - The de-identified images are used as **probe** and the original images are used as **gallery**.
- **Reverse recognition** - The de-identified images are used as **gallery** and the original images are used as **probe**.
- **Parrot recognition** - The attacker is assumed to know the de-identification function by himself. He can then use the same technique as naive recognition but activate the de-identification function on **training** and **gallery** first.

The power of Parrot recognition

- A test was made with H being composed of 200 random faces.
- Against pixelation, bar mask (hiding the eyes with a black rectangle) and T mask (hiding the eyes and the nose with two black rectangles), Parrot recognition gave 100% correct results.
- Intuitively, this is because these de-identification functions are 1-1 for most H-s.

Outline

Effective de-identification

- Let f be a de-identification function.
- Let C be some claim regarding the ability of f to hinder face detection.
- A function f will be called effective with respect to C in the case where given only a set of de-identified face images, whatever is claimed in C is indeed true.

Effective de-identification - an example

- Define f such that $f :=$ blackout.
- We can define C to be the claim: *Given a set of face images H and a set of de-identified face images H_d, for any de-identified face image in H_d, it is impossible to uniquely determine it's corresponding original face image.*
- Since f sends all elements of H to zero, given an output of $f(x)$ it is impossible to guess x with probability higher than $\frac{1}{|H|}$.

Preserved face de-identification

Let's define a notion to tell us that a function doesn't cause too much information loss.

- Let *loss* be some distance metric (usually Euclidean, but not necessarily).
- Let f be some function and g be another function.
- We define f to be preserved over g with respect to *loss* if

$$\forall \Gamma \in H : loss(\Gamma, f(\Gamma)) < loss(\Gamma, g(\Gamma))$$

Preserved face de-identification - an example

- Define $f_1 :=$ blackout.
- Define f_2 as $f_2(x) = \frac{\sum_{y \in H} y}{|H|}$. f_2 is called the mean function.
- Let C be the same claim from the previous example.
- Let *loss* be Euclidean distance.

Preserved face de-identification - an example

If we assume that:

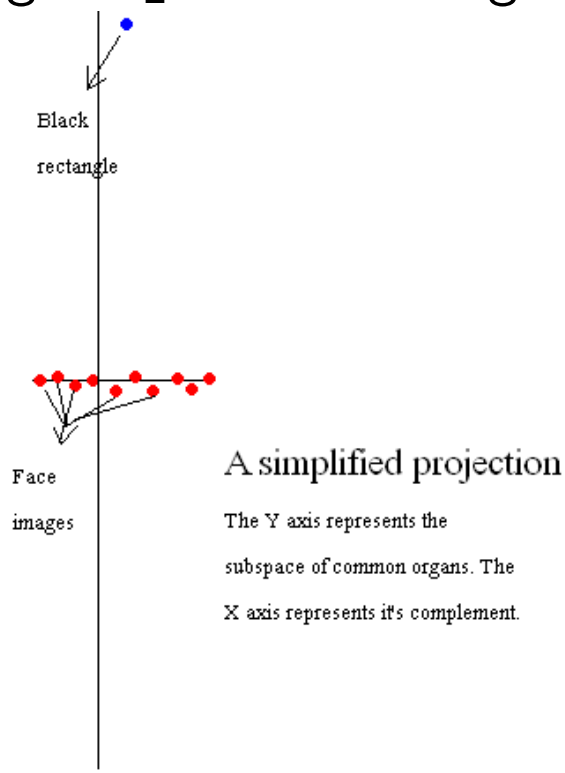
- All people have common facial organs (eyes, mouths etc.).
- The algorithm to generate a face image from a face still manages to place these organs on top of each other
- The percent of facial organs in a face image is high enough.

We can say that:

- f_2 is preserved over f_1 with respect to *loss*.

Preserved face de-identification - an example

- The reason we need the assumptions is that in order to be preserved, the distance $dist(f_2(\Gamma), \Gamma)$ must be less than $dist(f_1(\Gamma), \Gamma)$ for **all** Γ-s.
- For this to happen, we need the images to be similar in a subspace big enough to give f_2 the advantage over f_1.



A more relaxed claim - k-anonymity

- Let k be a positive integer.
- Let f be a de-identification function.
- Let C be the k-anonymity claim.
- We say that f is effective with respect to C if for every image in $y \in f(H)$, we have at least k sources in H.

An example of a de-identification effective with respect to k-anonymity

- Let us define Γ_i-s such that

$$H = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$$

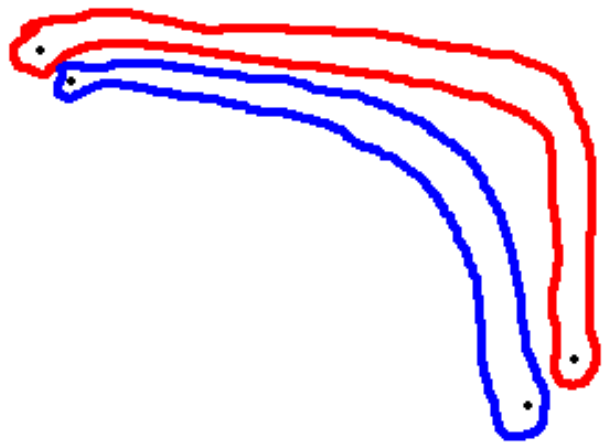
- Let f be the de-identification function defined as:

$$f(x) = \begin{cases} \frac{\Gamma_1+\Gamma_2}{2} & \text{if } x \in \{\Gamma_1, \Gamma_2\} \\ \frac{\Gamma_3+\Gamma_4}{2} & \text{if } x \in \{\Gamma_3, \Gamma_4\} \\ \frac{\Gamma_5+\Gamma_6}{2} & \text{if } x \in \{\Gamma_5, \Gamma_6\} \\ \vdots & \end{cases}$$

- f is effective with respect to 2-anonymity.

Order affects quality

- Let us look at an extension g of the previous function for some arbitrary k.
- When defining g above, we assume some kind of an ordering on the Γ_i-s within H. If a random ordering is used, we end up with mean images that may not be similar to the sources.
- This will cause more data loss.



The k-Same-Pixel algorithm - initialization

For this algorithm, we will use the notion of a return table, which specifies for each function input what should it's output be. The algorithm initializes as following

- Let k be a privacy constant.
- Train eigenfaces on H.
- Define $S := H$.

The k-Same-Pixel algorithm - loop

For each $\Gamma \in S$, we:

- Check if $|S| \geq 2k$:
  - If $|S| \geq 2k$, use eigenfaces to find the k-closest projections to Γ, call them K.
  - If $|S| < 2k$ use eigenfaces to find the $|S|$-closest projections to Γ, call them K.
- Set $m :=$ mean of elements of K.
- Set our return table to say that $\forall \Gamma' \in K : f(\Gamma') = m$.
- Set $S := S \setminus K$.

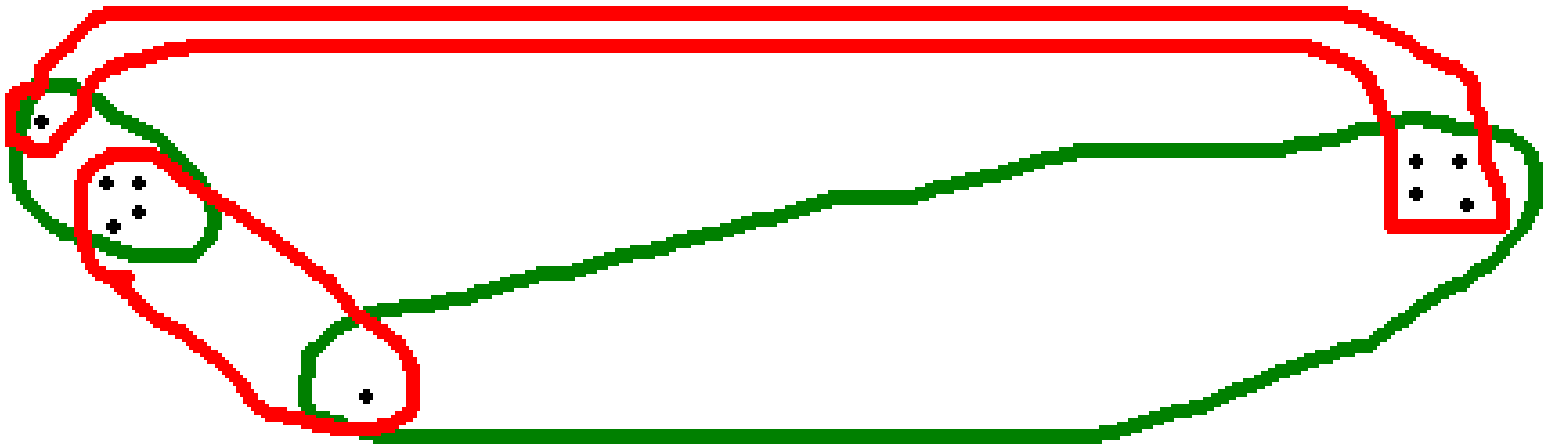
k-Same-Pixel looks good

As shown in the following example, k-Same-Pixel gives good looking results:

$|H|=805$

k-Same-Pixel is not perfect

In the description of the algorithm, one thing is left random, which Γ is selected. This can affect in cases like the following:



k-Same-Pixel is not preserved over the mean function

k-Same-Pixel is **not** preserved over the mean function with respect to Euclidean distance.

k-Same-Pixel is not preserved over the mean function

Here is an example of the previous claim:

- black dots - face images (equal to their projections).
- red dots (and dots marked as red) - k-Same-Pixel means.
- dot marked as blue - mean of all face images.

Summary

In this presentation we have:

- Defined a more formal model for image de-identification.
- Demonstrated problems with existing de-identification algorithms since they are mostly 1-1.
- Showed a new method that gives stronger guarantees to face image anonymity.

Biography

- E. Newton, L. Sweeney, and B. Malin. Preserving Privacy by De-identifying Facial Images.
- Wikipedia's "Principal Component Analysis" entry.