5-2019

Different Approaches to Blurring Digital Images and Their Effect on Facial Detection

Erich-Matthew Pulfer

University of Arkansas

Different Approaches to Blurring Digital Images and Their Effect on Facial Detection

by

Erich-Matthew Pulfer

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Bachelor of Science in Computer Science with Honors

APPROVED, THESIS COMMITTEE

Qinghua Li, Ph.D.
Advisor and Committee Chair

John Gauch, Ph.D.
Committee Member

Dale Thompson, Ph.D.
Committee Member

FAYETTEVILLE, ARKANSAS
April 2019

Abstract

The purpose of this thesis is to analyze the usage of multiple image blurring techniques and determine their effectiveness in combatting facial detection algorithms. This type of analysis is anticipated to reveal potential flaws in the privacy expected from blurring images or, rather, portions of images. Three different blurring algorithms were designed and implemented: a box blurring method, a Gaussian blurring method, and a differential privacy-based pixilation method. Datasets of images were collected from multiple sources, including the AT&T Database of Faces. Each of these three methods were implemented via their own original method, but, because of how common they are, box blurring and Gaussian blurring were also implemented utilizing the OpenCV open-source library to conserve time. Extensive tests were run on each of these algorithms, including how the blurring acts on color and grayscale images, images with and without faces, and the effectiveness of each blurring algorithm in hiding faces from being detected via the popular open-source OpenCV library facial detection method. Of the chosen blurring techniques, the differential privacy blurring method appeared the most effective against mitigating facial detection.

1 Introduction

Today, individuals share personal data with the world in high capacity. One of the most-used mediums for sharing is digital images and videos, with social media giants such as Instagram boasting over 100 million photos and videos uploaded each day [5]. With this kind of scale and the general availability of this data, privacy is arguably a commodity that few are afforded once they present such data. While individuals are partially protected via privacy settings within applications and general security measures set in place by the companies who own the platform, their data is still vulnerable. Facial detection has grown steadily more and more widespread, with libraries such as OpenCV allowing practically anyone with access to a computer to implement a rudimentary facial detection screening on images. While isolating a face in an image is not necessarily a revealing step in itself, doing so allows attackers the opportunity to possibly engage in other attacks, such as running the detected face through a database for potential identity detection.

While it doesn't seem very intuitive, blurring images has been used as a way of preserving privacy within shared images. By blurring, viewers other than the intended party are generally rendered unable to determine the image's content. In practice, a medium of communication might blur the image using some form of cryptography scheme (i.e. the image is only unblurred once accessed by the account that has the appropriate key) in order to protect the image's integrity within transit. This scheme only works when parties are communicating, however. What happens when a user wants to share an image in a public environment that contains data about another individual who is unaware that their privacy is being jeopardized? Ideas such as that laid out in the [6] recognize the issue of privacy violations simply due to unawareness and seek to preserve affected parties. Steps such as this are steps in the right direction, the issue is making sure we are making the steps quickly enough to keep ahead of attackers and their innate ability to quickly determine exploits. For example, [6] outlines the application PrivacyCamera as giving users the opportunity to have their face blurred if they appear in other users' pictures taken with the application. However, an attacker and retrieved one of these images containing the blurred face and was hellbent on determining the identity of the person obscured, one of their first steps is going to be isolating the affected portion of the image. Many blurring algorithms utilized commonly are open-source and potentially mathematically reversible, albeit through brute force [7]. While developing an almost completely irreversible blurring algorithm such as the purpose was of [3] is an amazing step towards preserving individuals' privacy, what if we could slow down or even stop attackers significantly before they even reach the unblurring step. What if we prevented them from detecting the affected portion of the image at all, other than visually? Any step that preserves privacy is a step in the right direction and, though this may be a less daunting operation than developing blurring algorithms that are harder and harder to reverse, preventing attackers from isolating the affected portion of an image upon which they may begin running attacks would significantly stall them, thereby allowing the community the opportunity to take a few more steps in the right direction and keep ahead of those attempting to violate privacy.

2 Background

2.1 Image Processing Operations

2.1.1 High-Pass vs. Low-Pass Filtering

In image processing, images can be filtered with either a high-pass filter or a low-pass filter. In general, high-pass filters are associated with the sharpening of an image and low-pass filters are associated with the smoothing of an image. Sharpening is used to increase the contrast of an image, which entails the brightening of bright areas of images and the darkening of dark areas of images in an effort to make edges appear more defined. Smoothing an image is an effort to make an image appear less pixelated and reduce the noise present, generally by adjusting each pixel's value to be closer to an "average value," which is usually the average RGB value calculated from the image's entire set of pixels. Despite most people's natural propensity to contrast the two with one another, it is very important to recognize that the two operations are not proper inverses of one another. Despite the fact that smoothing can rudimentarily be used to lower contrast and, therefore, counter sharpening operations, sharpening cannot be used to recover lost details in images that are too blurry, dark, bright, etc. Blurring can be implemented in different ways, but low-pass filters are the most heavily used methods. Due to the nature of this thesis, it is safe to say that the blurring approaches used are forms of low-pass filters.

2.1.2 Kernels

In image processing, kernels are small matrixes used to alter an image in some way by convoluting between the image and the kernel. They are used for a wide variety of image processing operations and can only exist in odd number forms (3x3, 5x5, …). The convolution operation is the process of adding each element of the image to its neighbor, with the operation weighted by the kernel. Despite being denoted by a "*" symbol, this operation is not typical matrix multiplication. For example, imagine we have the two following 3x3 matrices:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}$$

The central element of each of these matrices would be [2,2]. If the first matrix is the kernel and the second is a piece of the image in question, then the value at the central element [2,2] of the image piece would become the value of $(9 * A) + (8 * B) + (7 * C) + (6 * D) + (5 * E) + (4 * F) + (3 * G) + (2 * H) + (1 * I)$. We would repeat this calculation for each of the other elements by centering the kernel over each element in question. This does, however, lead to an issue when dealing with edge points. In order to properly handle edges, the kernel will request data from pixels outside of the image's bounds, which is obviously not possible. Fortunately, there are multiple approaches to edge handling when it comes to kernels: (1) extend, (2) wrap, (3) mirror, (4) crop, and (5) kernel crop. Each of these methods are handled as follows:

(1) **Extend**: The kernel is extended back into the image region the required amount to make up for missing pixels. Corner points are extended back in a 90° angle, while other edge points are extended in lines.

(2) **Wrap**: The image is wrapped, meaning values are taken from the opposite edge or corner to fill in missing values.

(3) **Mirror**: The image is mirrored at edges.

(4) **Crop**: Any pixel in the resulting image that would require values from pixels that do not exist is ignored and, therefore, set to some other default value (usually 0). This form of edge handling usually results in the resulting image being slightly smaller than the original due to all edge pixels being converted to black forming a black border or simply being cropped off of the resulting image.

(5) **Kernel Crop**: Any pixel in the kernel that extends out of the bounds of the original image is not used. Normalization, which will be discussed next, is adjusted to compensate not using these kernel values.

Another important concept to understand when discussing kernels is the idea of normalization.

Normalization is an effort to make sure that average brightness/darkness of the average pixel in the resulting image is as close, if not equivalent, to that of the average pixel in the original image. This is achieved by dividing the value of each element within the kernel by the sum of all of the kernel's element values. Performing this operation ensures that the sum of all of the elements of a normalized kernel is always one.

2.1.2 Box Blurring Method

As alluded to above, box blurring is a form of low-pass filter for image processing. The kernel utilizes uniform values, meaning that every element within the kernel is equivalent. Because this uniformity is relatively simple to calculate compared to some other methods, box blurring is a fairly quick and common method for blurring images. The main parameter in box blurring is the size of kernel you would like to use. For example, imagine we wish to blur an image using a 3x3 matrix. Our kernel then would appear as such:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\end{bmatrix}$$

The $\frac{1}{9}$ preceding the first matrix is the normalization value for the kernel, which is how we reach the equivalent second matrix. It is important to note that the value of each element of the kernel is reduced to one if the kernel is normalized. For example, consider the 3x3 kernel below:

$$\begin{bmatrix}2 & 2 & 2\\2 & 2 & 2\\2 & 2 & 2\end{bmatrix}$$

This is, by definition, an appropriate kernel to utilize for box blurring because all of the values are uniform. It has not yet been normalized, but when we work to normalize it, we learn it is equivalent to the previous kernel we found:

$$\begin{bmatrix}2 & 2 & 2\\2 & 2 & 2\\2 & 2 & 2\end{bmatrix} * \frac{1}{sum\ of\ all\ elements} = \frac{1}{18}\begin{bmatrix}2 & 2 & 2\\2 & 2 & 2\\2 & 2 & 2\end{bmatrix} = \begin{bmatrix}\frac{2}{18} & \frac{2}{18} & \frac{2}{18}\\\frac{2}{18} & \frac{2}{18} & \frac{2}{18}\\\frac{2}{18} & \frac{2}{18} & \frac{2}{18}\end{bmatrix} = \begin{bmatrix}\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\\\frac{1}{9} & \frac{1}{9} & \frac{1}{9}\end{bmatrix}$$

Between two different normalized kernels of different sizes, the larger of the two projects the greatest visual impact on the image. For example, consider the following images:

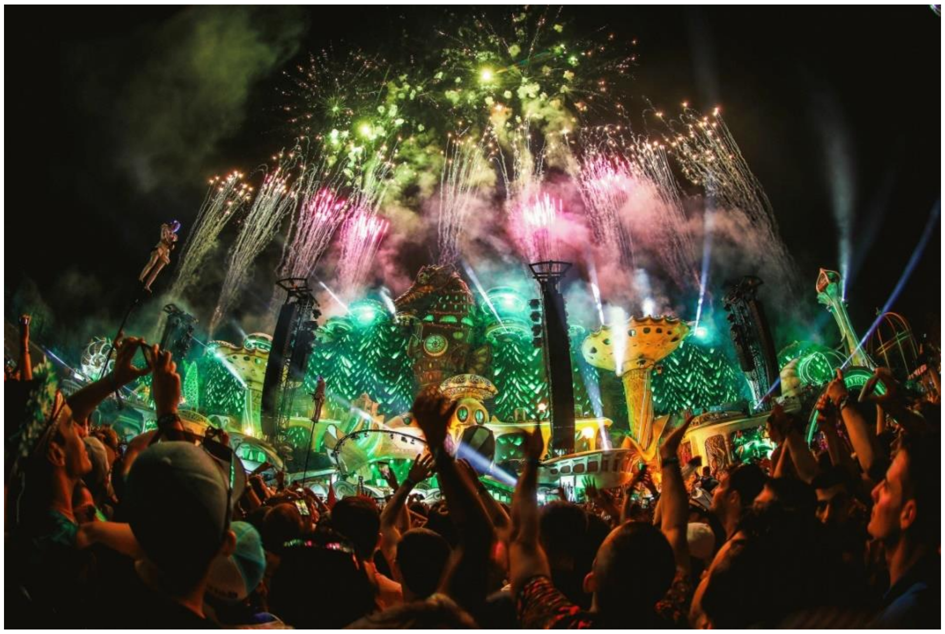


Figure 1: Original Image, selected due to high resolution (1200x800) and the amount of different colors present [9]

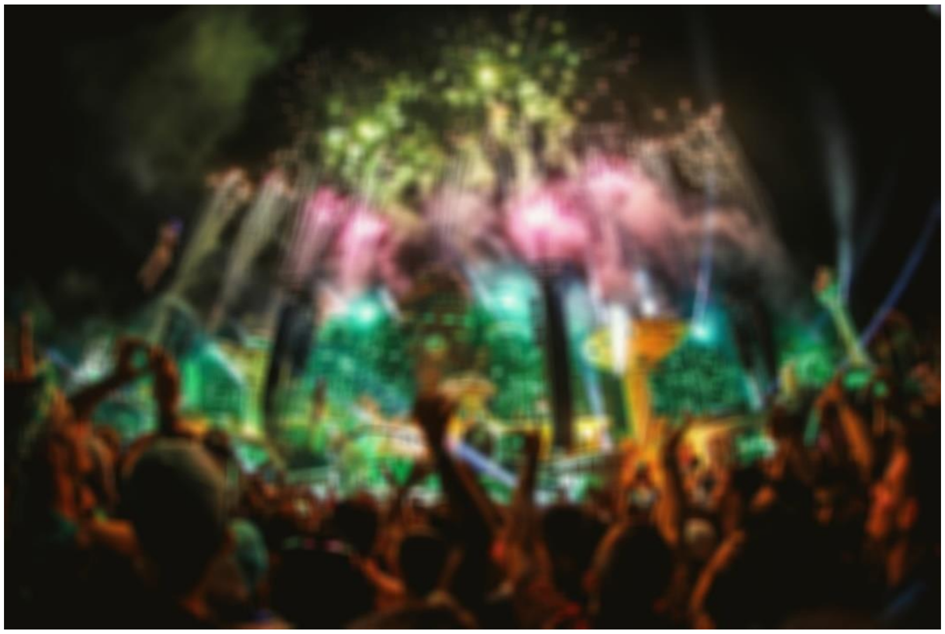


Figure 2: Image from Figure 1 box blurred using a 15x15 kernel

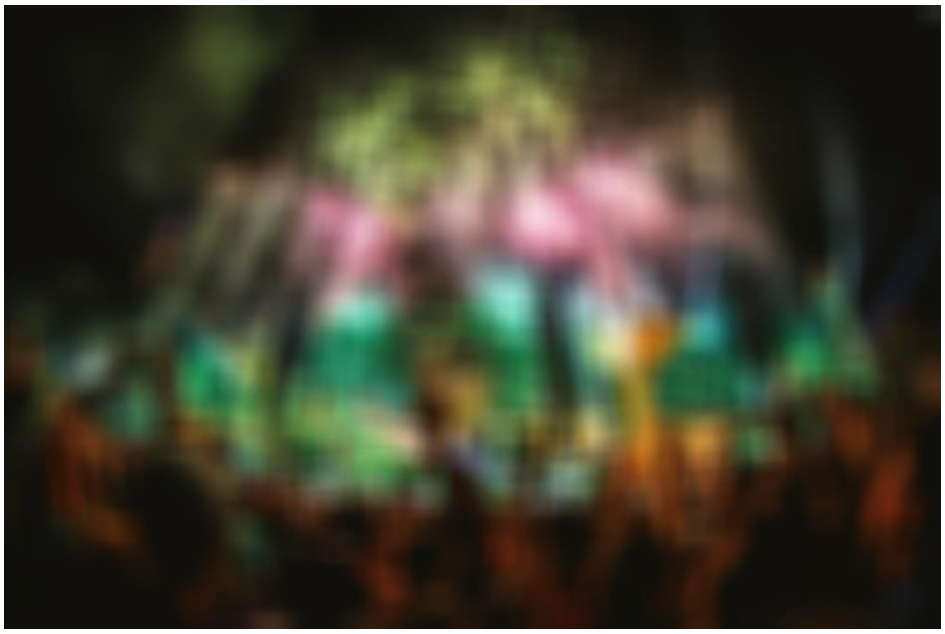


Figure 3: Image from Figure 1 box blurred using a 45x45 kernel

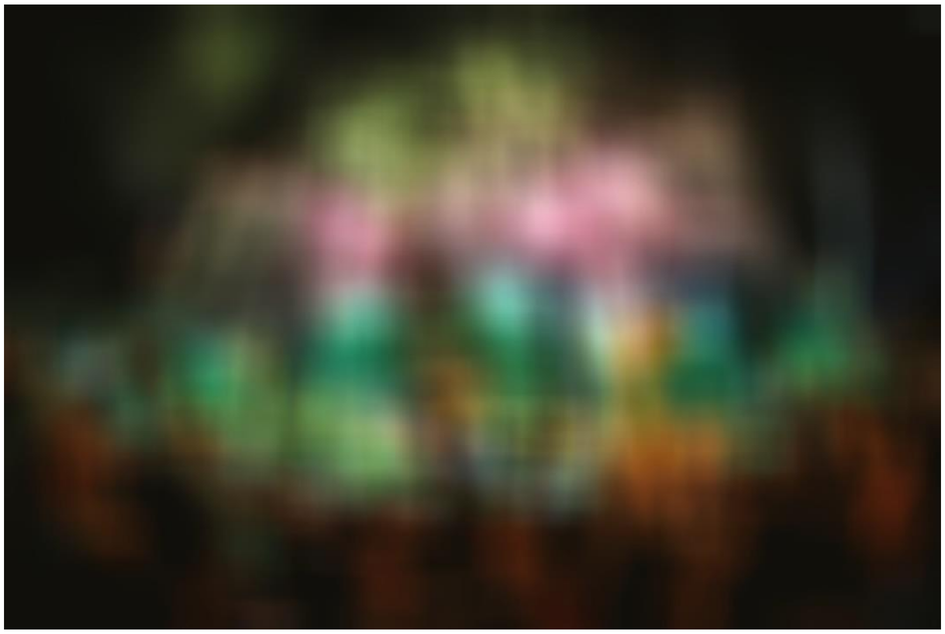


Figure 4: Image from Figure 1 box blurred using a 75x75 kernel

As you can see from the images, the visible blurriness increases as the size of the kernel used increases. This is due to the idea that, as kernel size increases, the number of pixels that affect a single pixel's value increases. We can also begin to make out box-like shapes in the image in the image blurred by a 75x75 kernel, reflective of the kernel and how its values are determined.

2.1.3 Gaussian Blurring Method

Gaussian blurring, like box blurring, is also a low-pass filter method for blurring images. It also utilizes a kernel to alter the image, but, unlike box blurring, the kernel's elements are not all a uniform value. Instead, the values of each element of a Gaussian kernel are calculated by a Gaussian function, meaning the resulting image is the convolution of the original image with a Gaussian function. The function utilized for the calculation of the kernel's elements is as follows:

$$Gaussian(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

The *x* and *y* correspond to an (*x*, *y*) coordinate system starting at the origin of the image (the top leftmost pixel), while σ is the standard deviation. This is the Gaussian function carried out in 2 dimensions, but it is important to note that a lot of times the Gaussian filter is carried out in one direction (usually horizontal first), followed by another run that runs in the other direction. In this case, the Gaussian function used only considers one direction:

$$Gaussian(a) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{a^2}{2\sigma^2}}$$

The a is whichever direction the run is being made at the time (*x* when running horizontal, *y* when running vertical). This two-pass method of Gaussian filtering requires fewer calculations and, therefore, is generally preferable in implementation. Consider the following 3x3 Gaussian kernel:

$$\begin{bmatrix} 0.077847 & 0.123317 & 0.077847 \\ 0.123317 & 0.195346 & 0.123317 \\ 0.077847 & 0.123317 & 0.077847 \end{bmatrix}$$

As you can see, there are actually only 3 unique values in this kernel: 0.077847, 0.123317, and 0.195346. This shows us that the kernel values decrease symmetrically as we move away from the center element. If we move away from the center element of any Gaussian kernel by moving only in cardinal directions (i.e. not up, down, left, or right), then all elements that are an equivalent distance from the center element are the same value. For example:

$$\begin{bmatrix} 4 & 3 & 2 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 2 & 3 \\ 4 & 3 & 2 & 3 & 4 \end{bmatrix}$$

The element denoted 0 is the central element and all elements denoted by the same number would be equivalent in value to one another (i.e. all 1's have the same value, all 2's have the same value, etc.). As with box blurring, the visual impact on an image increases as the kernel size increases. To display this, consider the following examples of Gaussian blurring applied to the original image depicted in Figure 1:

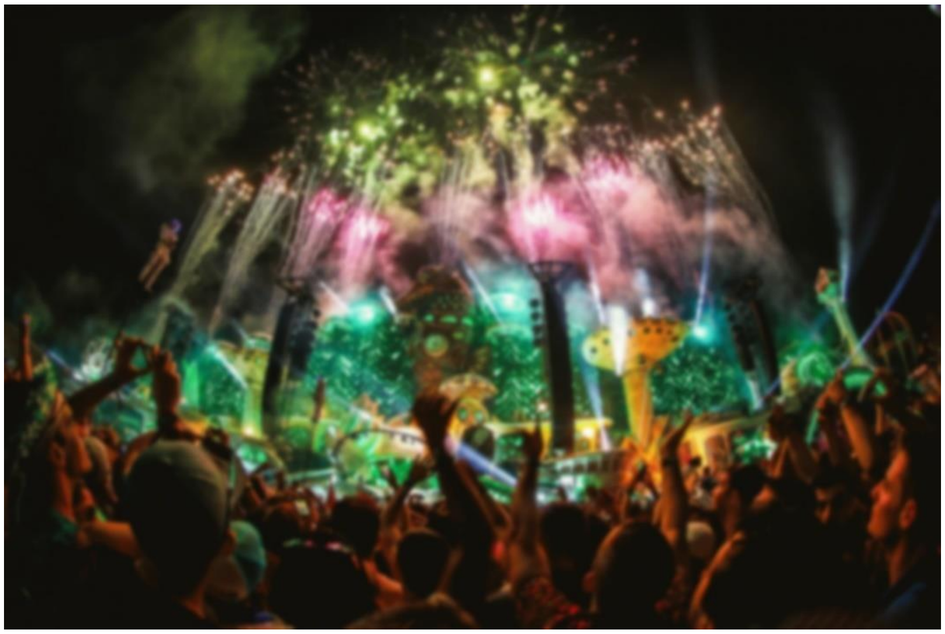


Figure 5: Image from Figure 1 Gaussian blurred using a 15x15 kernel

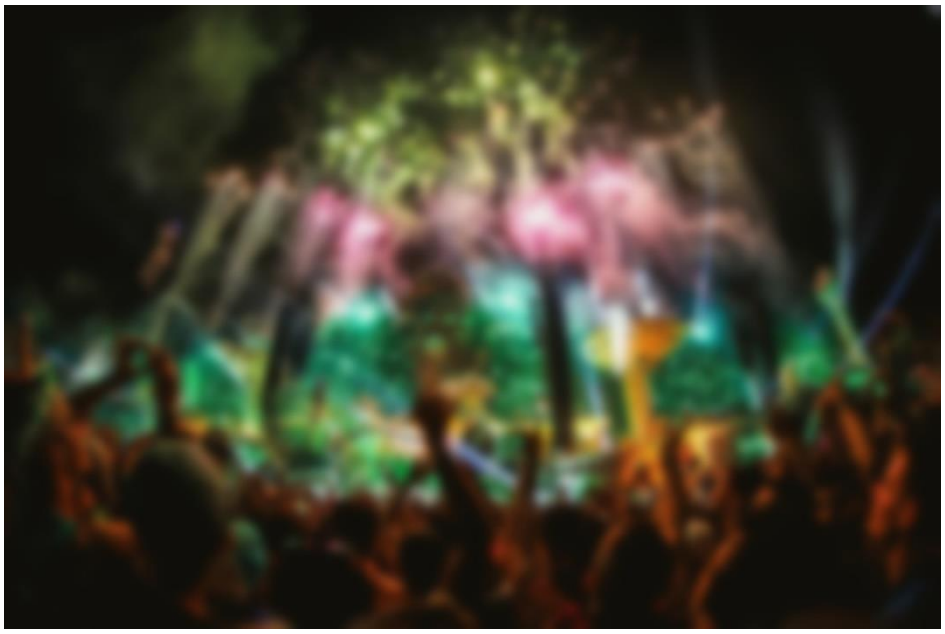


Figure 6: Image from Figure] Gaussian blurred using a 45x45 kernel

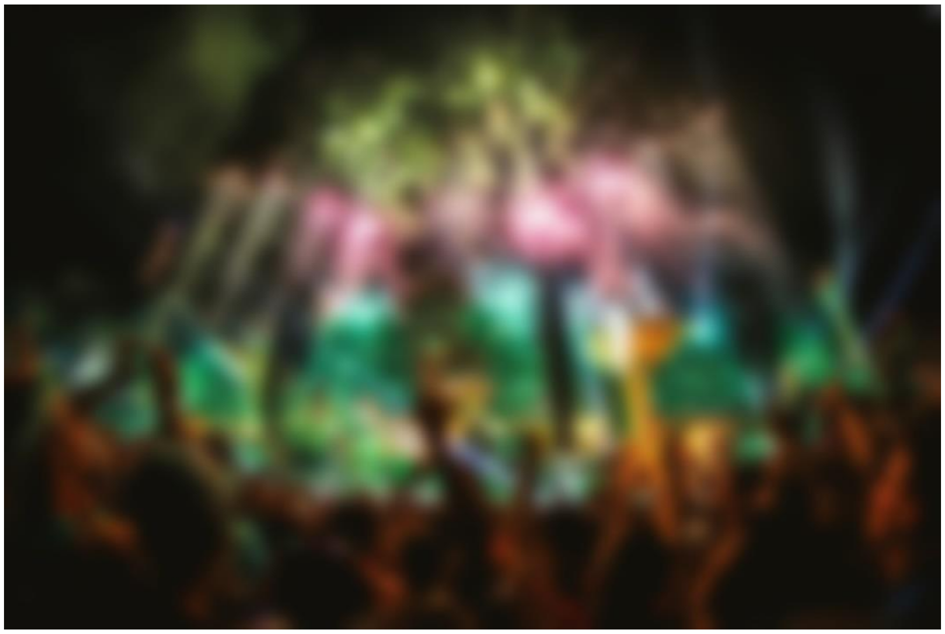


Figure 7: Image from Figure 1 Gaussian blurred using a 75x75 kernel

This blur appears much more radial visually as compared to the box blurring method, which correlates with the kernel values decreasing radially from the center point.

2.1.4 Differential Privacy Blurring Method

The differential privacy method is described further in the related work section due to the fact that it is not a widely established blurring method and is outlined in a very recent paper.

2.2 Related Work

Image blurring is not a new concept by any means, nor is the idea of utilizing it for means of privacy [3]. It has often been used in the obfuscation of information, though this has recently been considered to be insecure [7]. Box blurring and Gaussian blurring are two of the most commonly used blurring algorithms utilized in image processing, even implemented within image editing software such as Adobe® Photoshop®. The differential privacy blurring scheme used for this thesis was outlined in a recent paper from the University of Albany [3].

2.2.1 Differential Privacy Blurring Method

The differential privacy method as utilized in this thesis is a new form of blurring technique described in Liyue Fan’s paper “Image Pixelization with Differential Privacy” [3]. Because this method is relatively new, its implementation is questionable. The concept as outlined builds off of the idea of pixelization. Pixelization is the idea of reducing an image’s resolution in an effort to deliberately induce pixelation, or the display of a bitmap at such a high size that its individual pixels are visible and distinguishable to the naked eye. This technique essentially gets the color stored within a pixel, duplicates it, and pastes it into surrounding pixels until reaching a desired size. This idea then results in all colors within the image being represented in blocks. By doing so, we receive images such as the following:

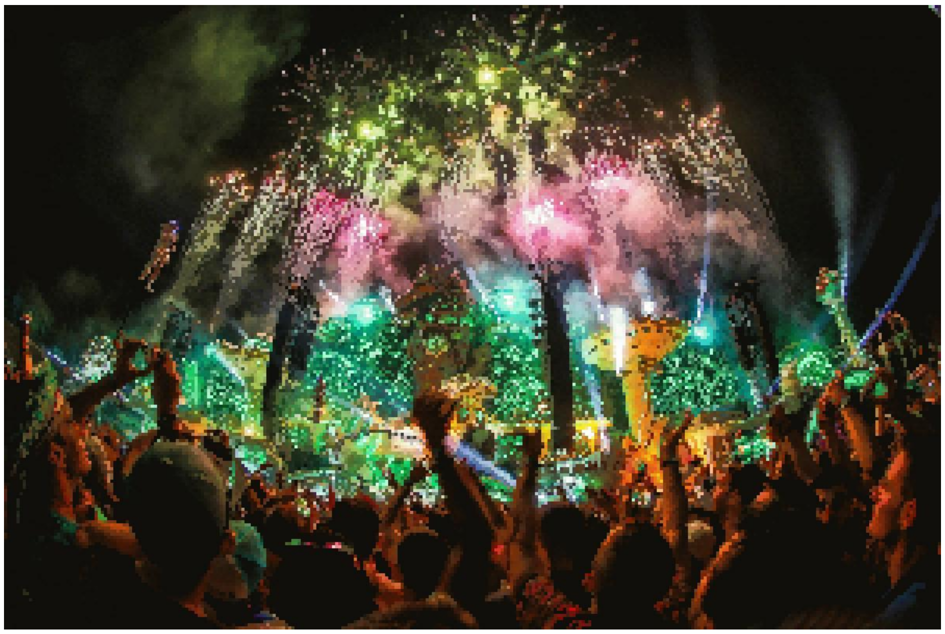


Figure 8: Image from Figure 1 pixelized with a pixel block size of 4

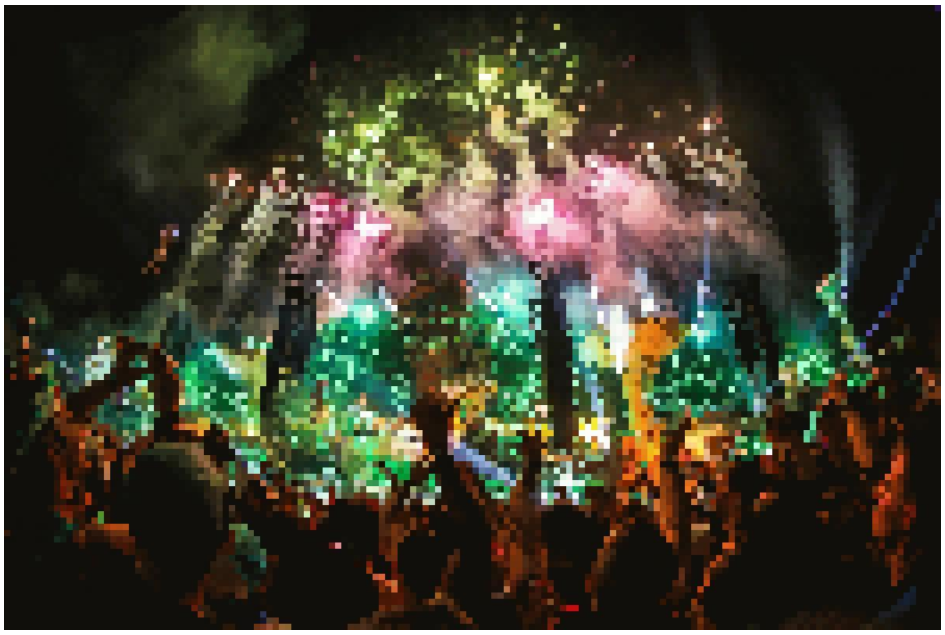


Figure 9: Image from Figure 1 pixelized with a pixel block size of 8

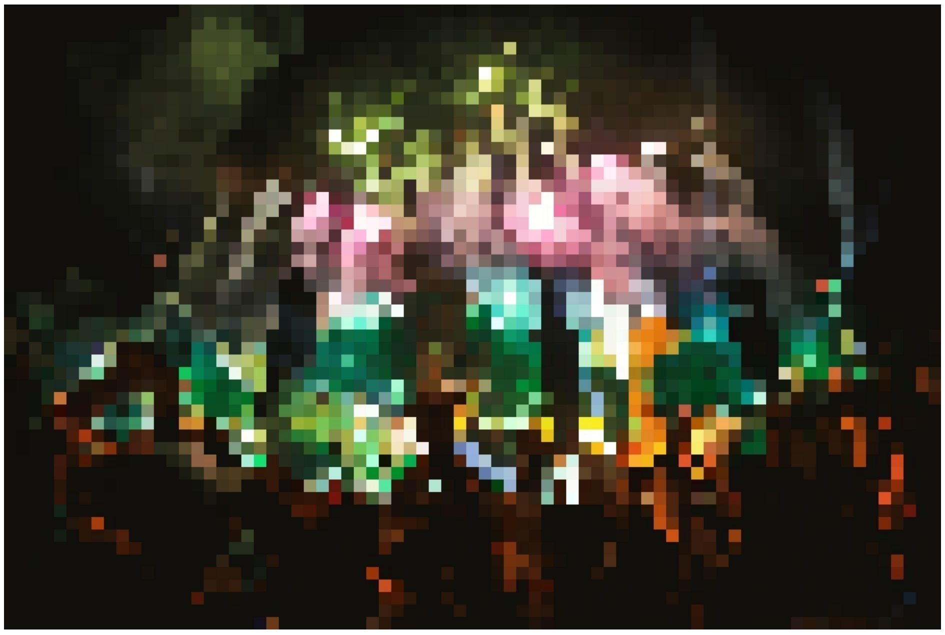


Figure 10: Image from Figure 1 pixelized with a pixel block size of 16

As the pixel block size increases, the image appears to be made of fewer total pixels, therefore appearing to be of a lower resolution. This is basic pixelization at work. The differential privacy scheme, however, adds a new element: noise is added to the dominant RGB value acquired in each pixel. This noise is calculated as a random value selected from a Laplace distribution. A

Laplace distribution is a continuous probability distribution, which, therefore, allows us infinite possible values to select as our noise value. However, due to the nature of RGB values only ranging from 0 to 255, we must use those values as our bounds. As laid out in [3], the Laplace distribution is calculated with a mean of 0 and a scale of $\frac{255m}{b^2\varepsilon}$ where *b* is our pixel block size, *m* is a selected number of pixels that differ between any neighboring images, and ε is our privacy parameter. As described, the default values are *b*= 16, *m* = 16, and ε = 0.5. [3] also explains the impact each of these variables has, as explained more rudimentarily below:

(1) ***b***: As *b* increases, the produced image more closely resembles a pixelized image developed without the added noise (i.e. if *b* increases, the range of potential noise values is lowered).

(2) ***m***: As *m* increases, so too does the strength of the degree of privacy achieved.

(3) **ε**: Lower ε values ensure a stronger degree of privacy achieved.

While the focus of this thesis was more focused on the concept of privacy achieved through the inability to detect faces, it is important to note that this technique of image processing is the most secure of the provided methods.

3 Design

The goal of this thesis is to determine which image processing technique most effectively reduces the possibility of facial detection. In the case of this thesis, the technique providing the most security is the technique that sees the greatest average decrease in the number of faces detected after utilization of that method to blur initially detected faces in the original image. These three techniques are by no means the only three available, but simply are meant to act as a base set for determination to be built upon later through future research.

The initial thought behind this thesis was this: blurring algorithms are effective, but in certain regards the content can still be determined, especially through the naked eye in large-scale cases. For example, I may detect and blur a face within an image, present it to an individual asking them what has been blurred, and, through context, they may determine that it is, in fact, a person's face that is blurred. While they may not be able to determine identity, they were still able to determine the subject matter. The question then arose: could the same be said for facial detection algorithms? This question cannot be answered completely in one definitive "Technique A is better than all other techniques" format simply because each of the techniques implemented for this thesis rely on their own parameters that alter their effectiveness. In order to effectively test each technique, this thesis was designed to implement each technique, run each of them with multiple parameter values, and to compare the number of faces detected in the original image to the number of faces detected in each produced image.

3.1 Data Sources

In order for each of the techniques to be tested thoroughly, an appropriate dataset needed to be developed. Images can be taken of anything and anyone and so the dataset needed to be as large as possible in order to cover the most amount of possibilities as possible. Ultimately, the decision came to utilize two main datasets: the AT&T Database of Faces and an image dataset collected by another student at the University of Arkansas who is also researching the in the field of image privacy. All of the images in both datasets are of .jpg format, making them easily handled by many existing Java libraries.

3.1.1 AT&T Database of Faces

AT&T, one of the leading telecommunications companies in the world, released a large collection of grayscale images entitled "The Database of Faces" in 2002 [8]. This repository contains 10 images each of 40 different subjects, comprising a dataset of 400 images. Each of the 10 images of a subject are profile shots, meaning that the image's sole subject is the individual's face. The 10 images are taken of the face at different angles and of the individual making different facial expressions in an effort to provide a diverse dataset both for each individual and for the entire collection of individuals overall. This database is easily accessible and was also utilized in [3], making it a very appealing dataset, especially given its focus on faces. This dataset was solely used to test the implementation of each of the image processing techniques.

3.1.2 Personally Collected Dataset

While the face images provided by the AT&T Database of Faces are excellent testing material, they are unfortunately idealized images. Very rarely are the images that individuals share online grayscale images that contain only a face in front of a neutral background. Images may contain 1 or multiple subjects from a variety of angles and distances in various settings. In an effort to utilize a more realistic dataset, this thesis utilized a dataset already collected by another student performing research on facial detection as given permission by the advisor. This dataset was compiled of 213 images, each image drastically unique from the others. These images are more reflective of real images shared regularly online every day, consisting of 1 or more subjects in different lighting, environments, positions, etc. This dataset was solely used to test the effectiveness of each of the techniques in beating a facial detection algorithm.

3.2 Facial Detection

Facial detection has become increasingly popular, with even common smartphones now utilizing for security purposes. Because of this, there are an increasing number of face detection algorithms being made available to the public for developmental purposes. Because of this and in the interest of avoiding some type of developmental bias, the design decision was made to utilize a commonly used facial detection algorithm for the purposes of testing. This choice is further explained in section 4.

4 Implementation

4.1 Box Blurring and Gaussian Blurring

Because box blurring and Gaussian blurring are commonly used blurring algorithms, they were easily implemented via the commonly used open-source image processing Java library OpenCV. To operate each algorithm, the image in question is converted to a Mat object. Mat objects are also defined within the OpenCV library as n-dimensional dense numerical arrays that can be either single-channel or multi-channel [2]. Because it can store matrices, it has the ability to store grayscale and color images (the RGB values are stored in a matrix). This allows us to alter the values via matrix operations, as described involving kernels in section 2. Both the box blurring and the Gaussian blurring function accept a source Mat object, a destination Mat object which allows us to save the alterations made to an entirely new image rather than overwriting, and then a Size object used to develop the kernel. The Gaussian function also requires a sigma value, which was set to 1 for the purpose of this thesis since this is widely accepted as the default sigma value for Gaussian filtering [4].

4.2 Differential Privacy Blurring

To implement the differential privacy blurring scheme as outlined in [3], the OpenCV library and the Apache library were both used. Pixelization was implemented to operate on a Mat object following the example of the box blurring and Gaussian blurring methods. The Apache library's math package included a Laplace distribution function greatly simplifying the whole process of getting the value to use as noise for this scheme. Depending on the pixel size specified (the *b* value for this scheme), a subimage of the appropriate size is created, the dominant color from said subimage is collected, the noise is added to this collected RGB value, and then each pixel in the subimage is set to the corresponding value. This results in a pixelated, potentially discolored image such as the figure below:

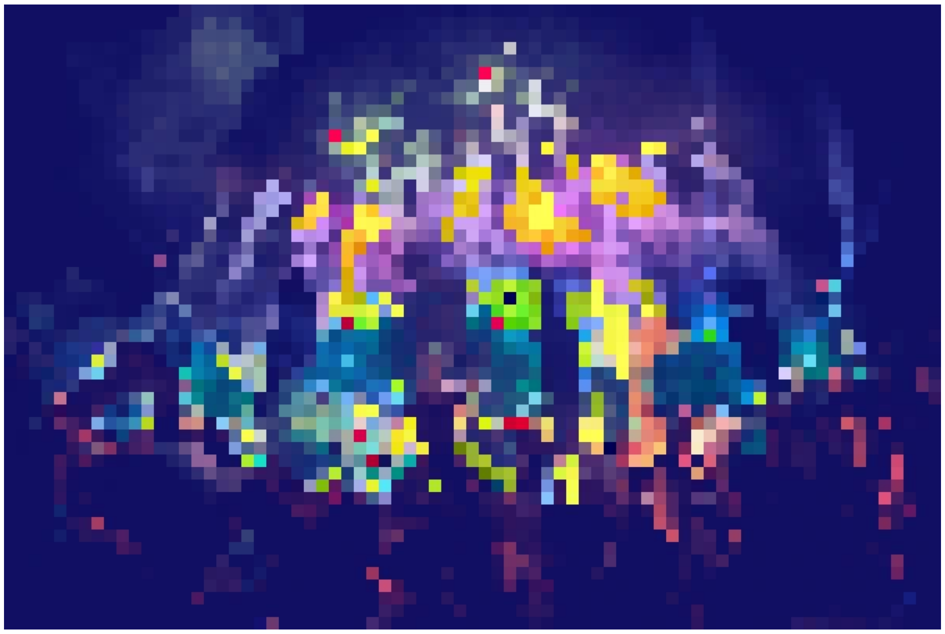


Figure 11: Image from Figure 1 blurred via differential privacy method with default values of b = 16, m = 16, ε = 0.5 as outlined in [6]

The discoloration comes from the addition of noise to the fetched dominant RGB value.

4.3 Facial Detection

As mentioned previously, facial detection has become much more accessible and popular in the past few years. Due to the popularity and simplicity of the OpenCV, it was also selected to implement facial detection. OpenCV facial detection utilizes what is referred to as a "CascadeClassifier," which is just a class used to load a classifier for object detection. Because of the idea that faces present in images can be of various sizes and in various positions, OpenCV's multi-scale detection seemed to be wise choice. Upon loading the classifier, the classifier is then fed the image in question and returns a MatOfRect object, which is essentially an array of Rect objects. Each Rect contained within the MatOfRect object depicts the outline around the detected object or faces in this case.

4.4 Data Collection

For data collection, it was important to run all of the original images within the manually collected dataset through facial detection multiple times: once on the original images and then once on each altered image developed from the utilization of the various schemes. In total, facial detection was run 56 times for every 1 image in the dataset. As our dataset was 213 images, this results in 11,928 different facial detection tests run total. The number of faces detected from each of these facial detection runs was then record for evaluation.

4.5 Evaluation

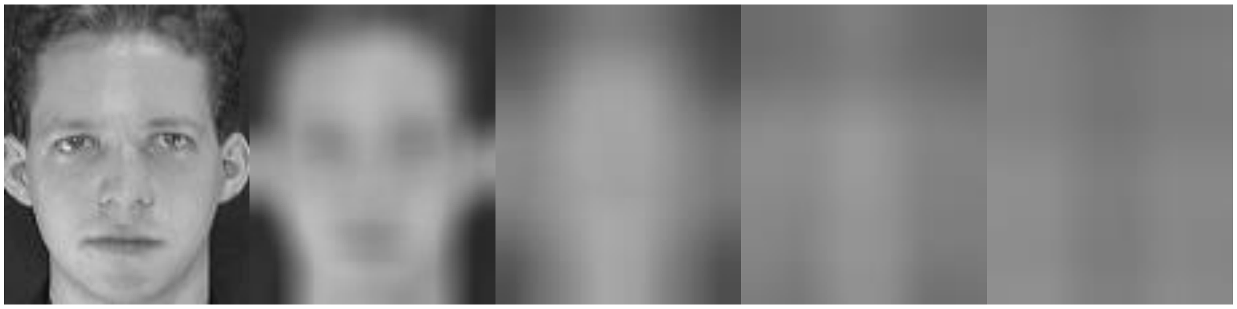
Upon completion of all tests, the collected number of detected faces for each individual altered image was compared to the number of faces detected in their corresponding original image. A lower number of faces detected in an altered image when compared to the original was considered an improvement in privacy, an equivalent number of faces detected in the altered image when compared to the original was considered to be no improvement in privacy and the alteration was rendered unnecessary, and a higher number of faces detected in an altered image when compared to the original was considered a significant error.

5 Results

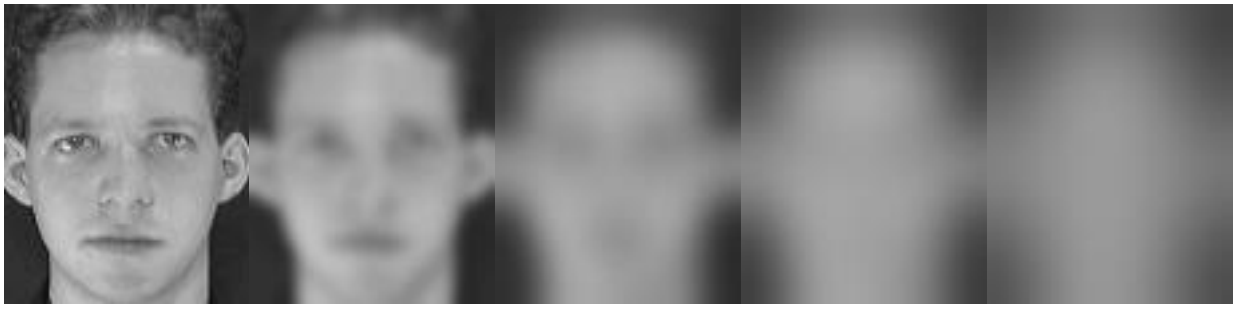
After all of the tests were run, all of the altered images were written to new images and saved, while the values regarding number of faces detected before and after alterations were recorded and kept in a spreadsheet for evaluation.

5.1 Images

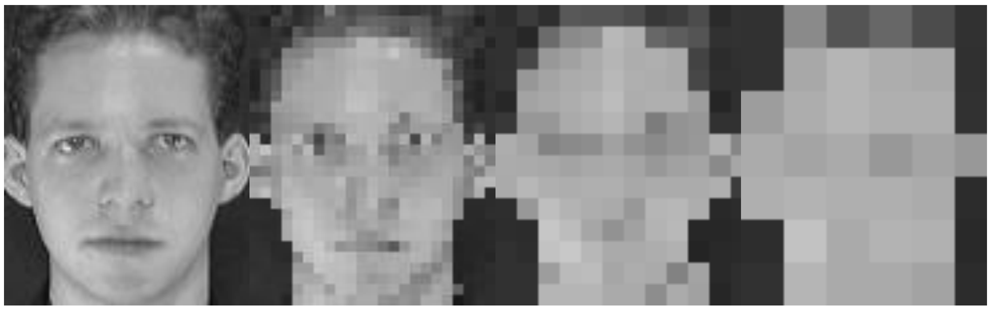
Each alteration performed on the original images from the datasets were kept as reference for the effects the alteration had and for later potential use. As there are way too many images to display them all here, a set of them has been selected to display each of the types of alterations made on images from both datasets:



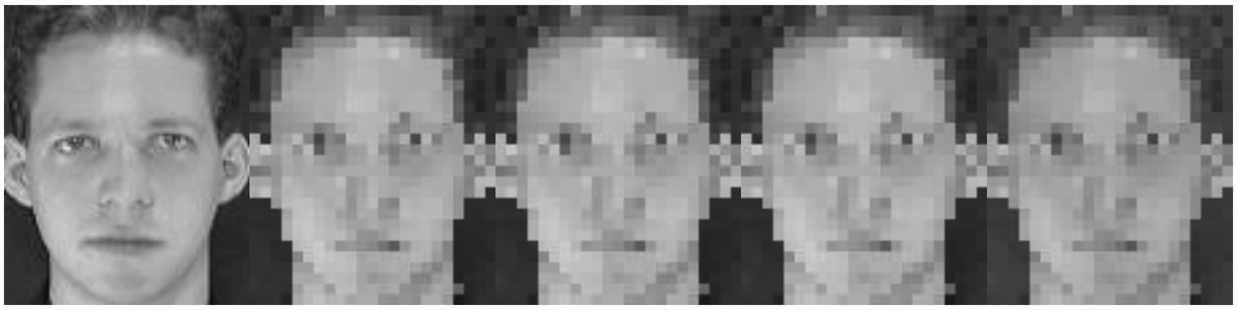
Figures 12 - 16: Box blurred images, from left to right: Original image of subject s1, kernel size 15, kernel size 45, kernel size 75, kernel size 105 [8]



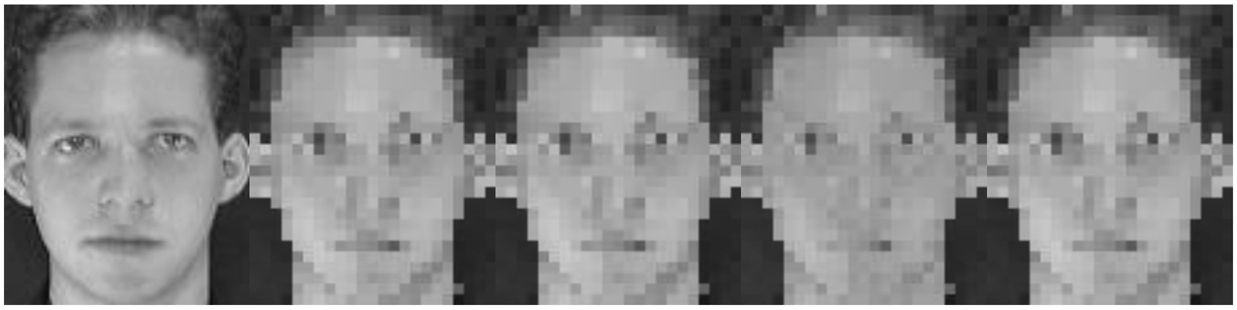
Figures 12, 17 - 20: Gaussian blurred images, from left to right: Original image of subject s1, kernel size 15, kernel size 45, kernel size 75, kernel size 105 [8]



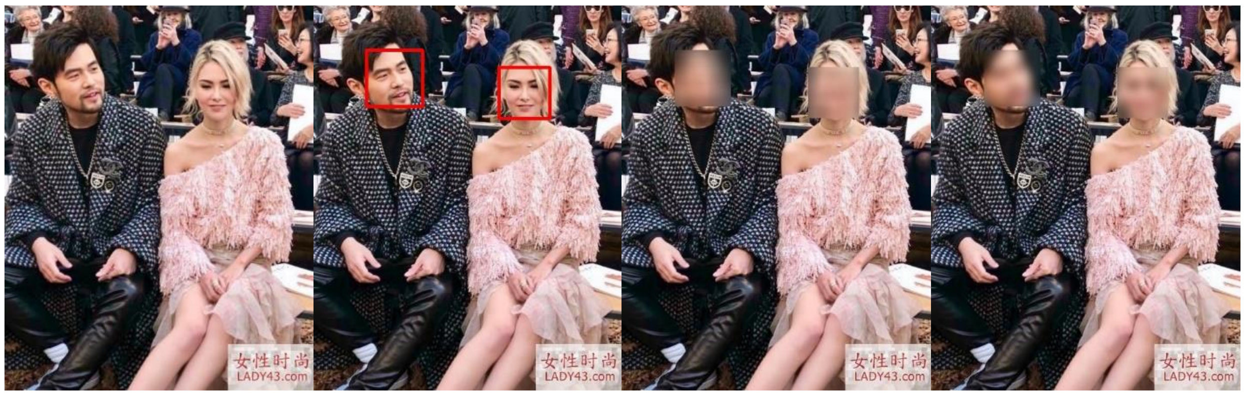
Figures 12, 21 - 23: Images blurred via differential privacy method, from left to right: Original image of subject s1, b value 4, b value 8, b value 16, all images have an m value of 16 and an ε value of 0.5 [8]



Figures 12, 24 - 27: Images blurred via differential privacy method, from left to right: Original image of subject s1, m value 16, m value 32, m value 48, m value 64, all images have a b value of 16 and an ε value of 0.5 [8]



Figures 12, 28 - 31: Images blurred via differential privacy method, from left to right: Original image of subject s1, ε value 0.25, ε value 0.5, ε value 0.75, ε value 1.0, all images have a b value of 16 and an m value of 16 [8]



Figures 32-35: From left to right: Original image, faces detected and boxed in, faces box blurred w/ kernel size 45, faces gaussian blurred with kernel size 45[1]

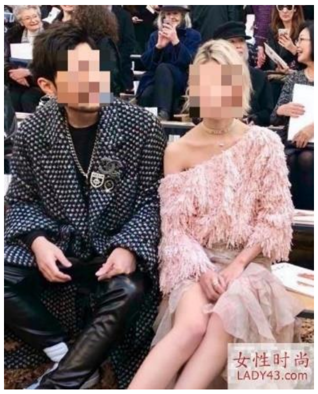


Figure 36: Faces from Figure 33 blurred via differential privacy method [1]

For all 613 original images collected across both datasets, there exist 56 altered images. This results in 34,384 images in total. Of those 34,328 images, 29,424 of them are alterations via the differential privacy method using multiple *b*, *m*, and *ε* values. The remaining 4,904 images are images created via the box blurring and Gaussian blurring methods.

5.1 Algorithm Analysis

Upon completion of all tests, the number of faces detected before and after each alteration on the 213 images in dataset 2 was recorded in order to determine which blurring algorithm was the most effective at mitigating facial detection. Success in mitigation was measured in the difference between the number of faces detected in the original image and the number of faces detected in the altered image, which we will refer to as the DIFF. The higher the DIFF, the higher the number of faces from the original image that were obscured. A lower DIFF means that more of the faces from the original images are still detectable, with a negative DIFF meaning the alteration produced a new section (or sections) within the altered image that the facial detection feature recognized as a "face." This means we ultimately want the method that produces the highest DIFFs on average. This value was calculated for every original image and each of its altered images, then averaged for each method at its respective parameters. Because box blurring and Gaussian blurring are the two more common blurring methods are utilized the same kernel sizes, we will compare them first:

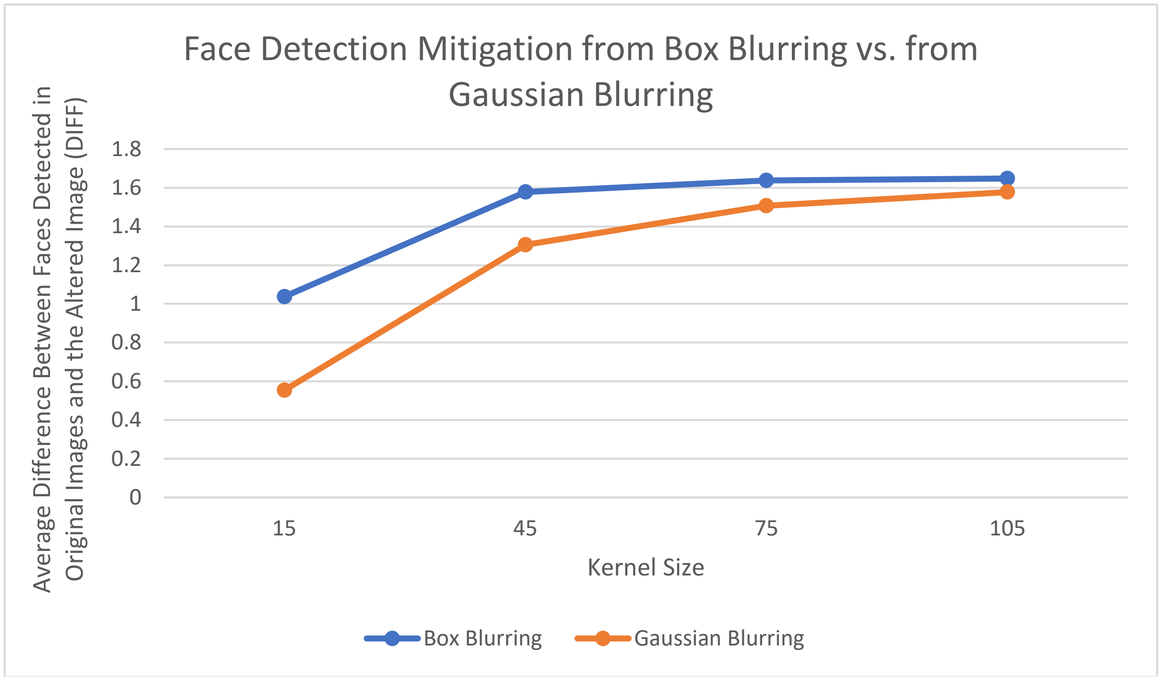


Figure [37]: Box blurring vs. Gaussian blurring at equivalent kernel sizes

As the chart displays, box blurring continuously had a higher average DIFF than Gaussian blurring did, suggesting it was the blurring method that was more effective at facial detection mitigation than the Gaussian blurring method was. While this thesis does not delve into why that is exactly, Gaussian blurring does produce a radial filter which might be more easily recognized by the facial detection method as "face-like" since face shape is ovular. Because of the vast differences between the types of blurring exhibited (box and Gaussian blurring are a form of filtering, while the differential privacy method is pixelization), it is not exactly fair to compare them to one another. However, because of the sheer amount of differential privacy tests that were run, we can compare this method to itself when we change its parameters.

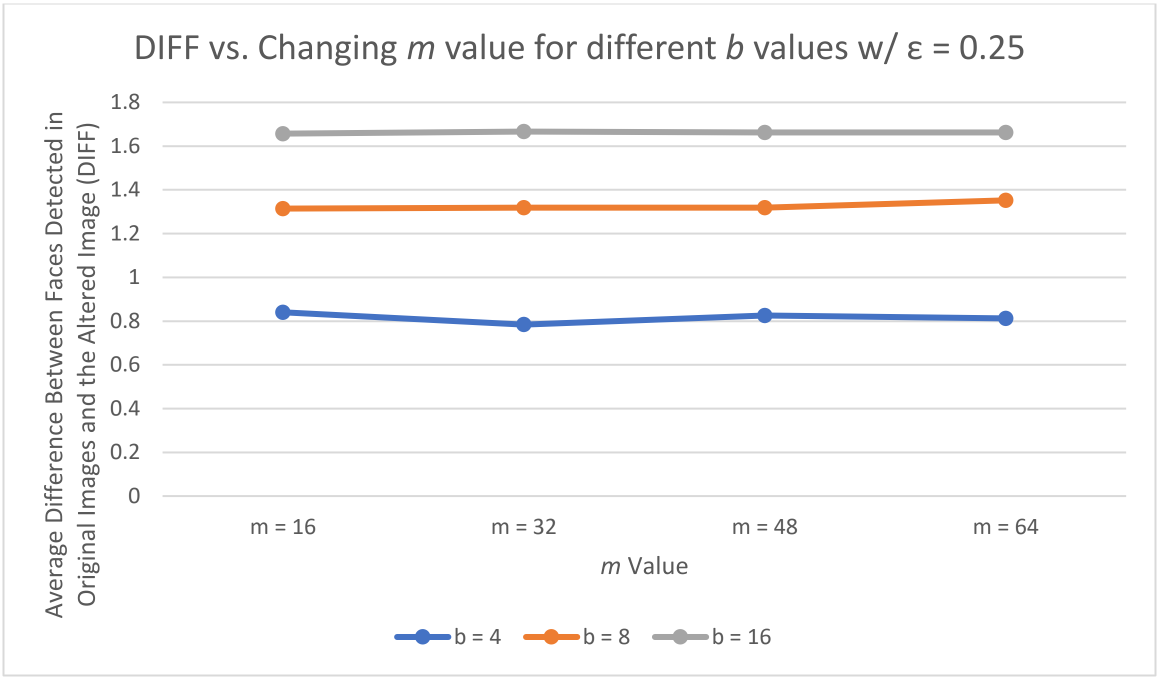


Figure [38]: Changing b value vs. changing m value

This chart reveals that changes in *b* value used with the differential privacy method have a greater impact on the DIFF than changes in *m* do. This suggests that the higher the *b* value, the higher the DIFF. Next, we can compare a constant *b* value to the changes in *m* and ε.

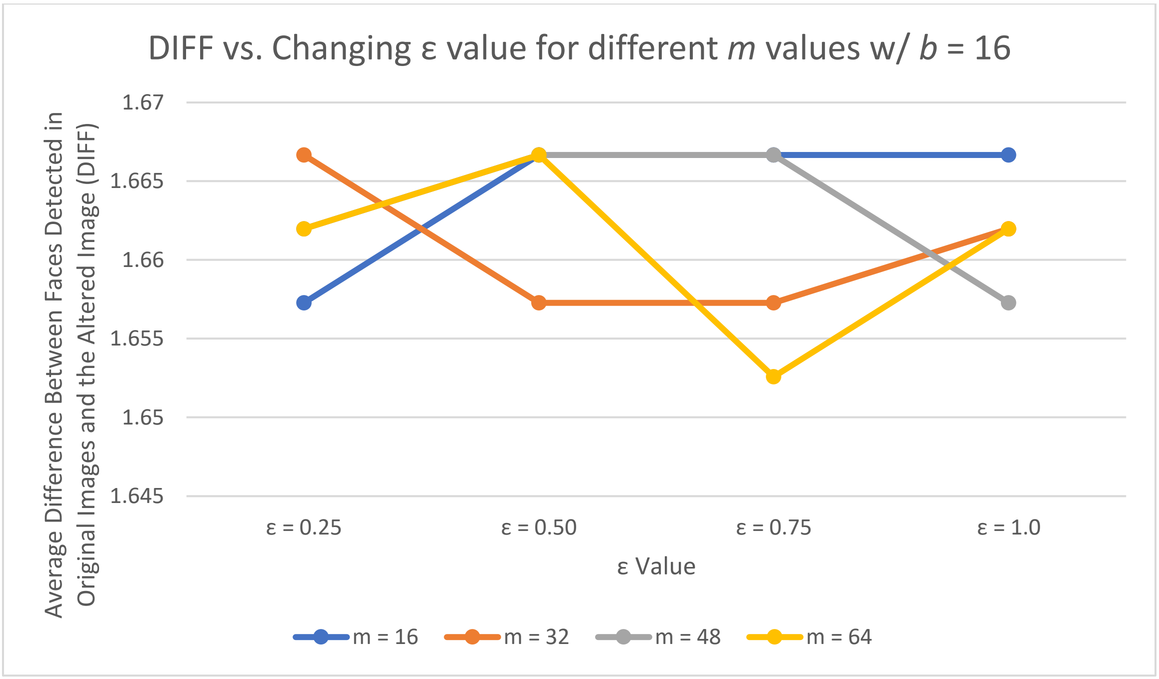


Figure [39]: Changing m value vs. changing ε value

This chart reveals that the adjust me of the ε value does not appear to have a big impact on the DIFF, nor does a change in *m* value. From this and the previous display, we can make our final comparison between *b* and ε:

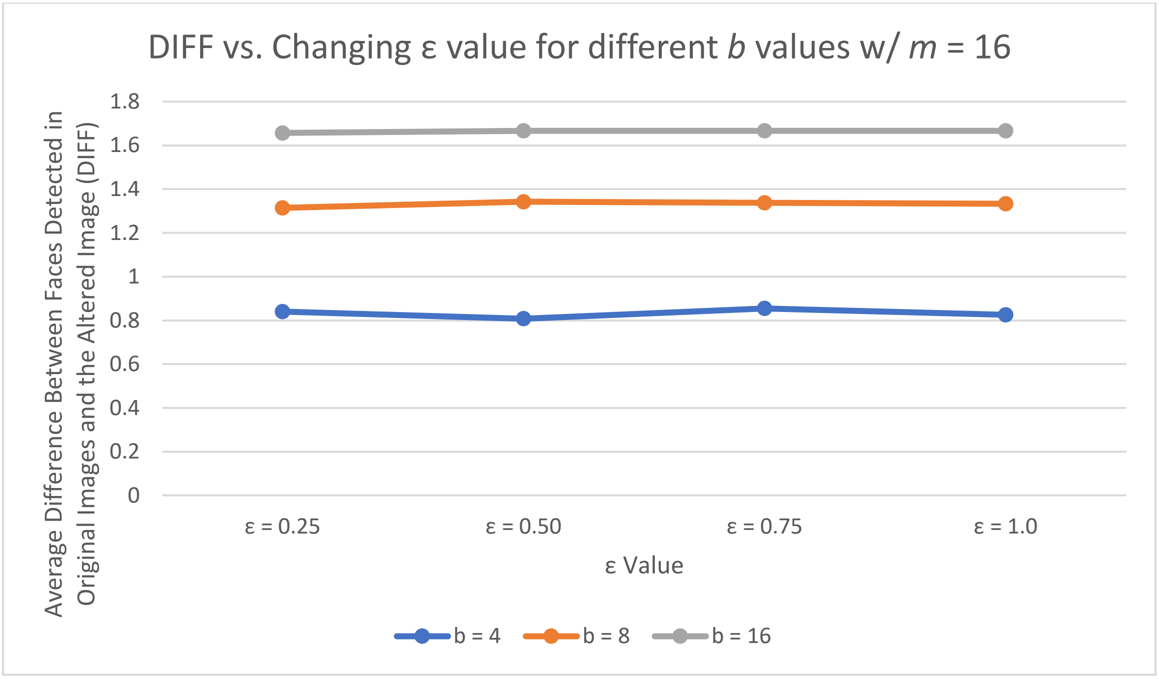


Figure [39]: Changing b value vs. changing ε value

This chart provides a similar conclusion to that of the first one dealing with the differential privacy method: changes in *b* value have a much greater impact on the DIFF than changes in the ε value do. After all of these comparisons, it is reasonable to conclude that the higher the *b* value, the more effectively the differential privacy method prevents facial detection. The *m* value and ε value have little impact on the effectiveness. Overall, the differential privacy method provided the highest DIFF out of all the methods: 1.666666667 on multiple occasions. Gaussian blurring provided the lowest DIFF out of all the methods: 0.55399061 at kernel size 15.

6 Conclusion

6.1 Summary

The purpose of this thesis was to set the groundwork for the potential of utilizing image blurring to prevent facial detection within images in an effort to mitigate or delay attackers from utilizing such a tool to violate individuals' privacy. Three image blurring methods were implemented in order to test the effectiveness of each in the prevention of successful facial detection: box blurring, Gaussian blurring, and a new differential privacy method of pixelization. Two datasets were collected for testing purposes, one being the AT&T Database of Faces, and the other being an image dataset compiled from various sources and provided by a fellow student also performing research at the University of Arkansas. Upon the collection of these datasets, tests were developed to determine the effectiveness of each method on the images within the dataset. These tests included box blurring at a kernel size of 15, 45, 75, and 105, Gaussian blurring at a kernel size of 15, 45, 75, and 105, and the usage of the differential privacy pixelization method a every possible combination of *b* values 4, 8, and 16, *m* values of 16, 32, 48, and 64, and ε values of 0.25, 0.50, 0.75, and 1.0. Facial detection was run on the original image, as well as the resulting image created by each of the tests, with the number of faces detected in each case recorded for later analysis.

Out of the three selected blurring methods, the differential privacy pixelization method provided the most effective negation of facial detection, with an average reduction of 1.666666667 in the number of faces detected. Box blurring proved to be more effective in preventing successful facial detection than Gaussian blurring, with Gaussian blurring providing the lowest effectiveness of all. The positive relationship between an increase in *b* value used and the effectiveness of preventing successful facial detection through use of the differential privacy pixelization method suggests that this method could be developed into an even greater tool for privacy protection.

6.2 Drawbacks and Future Work

This thesis experienced its fair share of drawbacks, the most gleaming of which was the fact that the facial detection method implemented via OpenCV was not 100% in its detections. Certain images had faces present that remained undetected by the algorithm, some had “faces” detected within the image that were not actual human faces, while others experienced both of these issues. Because the focus of this thesis was the effectiveness of blurring in preventing faces from being detected in general, this issue was allowed as the blurring was enacted on any and all faces detected. However, if the information collected in this thesis were to progress in an effort to provide greater digital privacy to the world, a more accurate facial detection method would need to be developed or implemented if it already exists. Blurring in an attempt to prevent successful facial detection has no purpose if faces that are meant to be blurred aren’t detected in the first place.

Overall, this concept should be broadened to accept a much larger number of image processing techniques and compare their relative effectiveness. Blurring is by no means the only way to alter an image as to hide content from the naked eye and any such alteration should be included in this study to determine the most effective method currently available.

It is also still important to note that the privacy sought by this thesis does not prevent attacks, but rather mitigates them and stalls attackers. In today’s digital world, it’s practically impossible to do anything more than slow attackers down. Any time that can be bought for security professionals to implement further methods to mitigate new, fresh attacks is a priceless commodity.

Acknowledgements

This thesis was advised by Dr. Qinghua Li and received constructive feedback from Dr. John Gauch and Dr. Dale Thompson, all professors at the University of Arkansas. Part of the dataset utilized in this thesis is due to the credit of AT&T Laboratories Cambridge. A large amount of functionality of this thesis was able to be implemented thanks to the developers of the open-source OpenCV image processing library for Java.

References

[1] "5a9f6836e4c67.Jpg." *lady43.Com*, 2018, www.lady43.com/Article/detail/id/24286.html.

[2] "Class Mat." *Mat (OpenCV 2.4.2 Java API)*, 3 July 2012, docs.opencv.org/java/2.4.2/org/opencv/core/Mat.html.

[3] Fan, Liyue. "Image Pixelization with Differential Privacy." *University at Albany*, 2018.

[4] Fisher, R., et al. "Gaussian Smoothing." *Spatial Filters - Gaussian Smoothing*, 2003, homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

[5] "Instagram by the Numbers (2019): Stats, Demographics & Fun Facts." *Instagram by the Numbers (2019): Stats, Demographics & Fun Facts*, Omnicore Agency, 6 Jan. 2019, www.omnicoreagency.com/instagram-statistics/.

[6] Li, Ang, et al. "PrivacyCamera: Cooperative Privacy-Aware Photographing with Mobile Phones." *University of Arkansas, University of Tennessee*.

[7] McPherson, Richard, et al. "Defeating Image Obfuscation with Deep Learning." *ArXiv.org*, 6 Sept. 2016, arxiv.org/abs/1609.00408v2.

[8] "The Database of Faces." *The Database of Faces*, AT&T, 2002, www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[9] "Tomorrowland 2018 - Weekend 1." *Albums :: Tomorrowland Media*, 365.tomorrowland.com/album/view/aid/20.