Designing a face detection CAPTCHA

Adam C. Day
West Virginia University

DESIGNING A FACE DETECTION CAPTCHA

Adam C. Day

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Lane Department of Computer Science and Electrical Engineering

Morgantown, WV

2010

Keywords: CAPTCHA; Face Detection; Image Quality Metrics; Image Distortions

ABSTRACT

DESIGNING A FACE DETECTION CAPTCHA

Adam C. Day

Completely Automated Tests for Telling Computers and Humans Apart (CAPTCHAs) are quickly becoming a standard for security in every online interface that could be the subject to spam or other exploitation. The majority of today's CAPTCHA technologies rely on text-based images, which present the user with a string of distorted characters and asks the user to type out the characters. The problem with CAPTCHAs is that they are often difficult to solve and can generally be successfully defeated using techniques such as segmentation and optical character recognition. We introduce an image face recognition based CAPTCHA which presents the user with a series of distorted images and the question of deciding which of these images contain a human face. The user is required to click on all presented face images in order to successfully pass the CAPTCHA. The concept relies on the strength of the human ability to detect a face even amongst heavy distortion as well as the inaccuracies and short-comings of face recognition software. The CAPTCHA application was designed with a web interface and deployed on West Virginia University's Computer Science 101 attendance website. To test the success of the CAPTCHA, data for human success rates was compared alongside facial recognition software which attempted to solve the CAPTCHA. The results of the data gathered during testing not only prove the feasibility of face recognition based CAPTCHAs in general, but also provide valuable data regarding human versus computer recognition rates under varying types of image distortion.

ACKNOWLEDGEMENTS

I would like to first thank my advisor Dr. Afzel Noore for his advice and support. He has been unbelievably helpful with everything from my introductory undergraduate courses to the advising on research and my thesis. I greatly appreciate his support through the means of my teaching assistantships in digital logic. His expertise and guidance cannot be appreciated enough.

I would also like to thank Dr. Powsiri Klinkhachorn for being on my committee. In addition, his teaching and support through the teaching assistantships is also greatly appreciated. I would also like to thank Dr. Raymond Morehead for his help on my committee. I wish to acknowledge Dr. Matthew Valenti for his recommendations, support and his ability to make some of the most complicated topics easy to comprehend.

In addition, I would like to thank Brian Powell for his assistance with the research and publication and the use of the CS101 website for testing. The entire process was made so much easier with his help and knowledge.

I would also like to thank my family. My parents Wendy and James Bailey have been so caring and always supported me through my endeavors. My mom has always put me before herself and I wouldn't be where I am today without her love and support. I also thank my brothers and sisters Aron, Marne and Tad Davis. Lastly, I would like to thank my girlfriend Ashleigh James for her warmth, love and patience. Words cannot describe the help, love and support she has given me over the years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 CAPTCHAS

1.1.1 WHAT IS A CAPTCHA?

Although most people are not familiar with term CAPTCHA, many use them on weekly or even daily basis. CAPTCHAs are the security tests that are most often found on websites that require the use of registration. These security tests often consist of a series of distorted characters that the user must type in order to continue onto the desired location. The purpose of the CAPTCHA is to restrict access to a website or other form of resource to ensure that the access is being performed by a human rather than an automated computer system. In its simplest form, a CAPTCHA is a method for telling humans and computers apart. The term CAPTCHA stands for a "Completely Automated Public Turing Test to Tell Computers and Humans Apart" [1]. The term "Turing Test" refers to a game in which a human judge would ask a number of questions to a human and a computer player without knowing which player was the computer and which was human [2]. The judge would then have to determine which player was the human player and which was the computer player. In the case of a CAPTCHA, the judge is no longer a human, but is instead the computer server hosting the CAPTCHA; this computer must make the determination of whether the user is human or computer.

1.1.2 Common Uses for CAPTCHAs

The most common use of CAPTCHAs today is to protect websites from the automation of registration of accounts or other spam. CAPTCHAs are commonly found on websites that provide web-based e-mail services, forums, comment boxes or polls. In the case of e-mail services, the CAPTCHA prevents the automated registration of accounts, which could then be used to send junk e-mail or for other nefarious purposes. When CAPTCHAs are not put in place on forums and comment boxes, automated attacks can overrun these systems with unsolicited advertisements. The need for CAPTCHAs in polls is shown in the November 1999 poll on slashdot.com, which asked which was the best graduate school in computer science [1]. In this poll, both MIT and CMU created automated programs to vote for their own school [1].

While CAPTCHAs are used to prevent the generation of spam, they can also be used to prevent the receipt of spam. A common method of preventing internet spam is to only allow the receipt of e-mails by individuals on the recipient's contact list. The problem with this method is that it can often block legitimate e-mail as well. One approach to solving this issue is an application called Spam Arrest, which sends a reply e-mail to anyone who e-mails you that is not on your contact list [3]. In this reply e-mail, a CAPTCHA is sent, and upon successfully solving that CAPTCHA, the original sender is added to the original recipient's contact list. In this case, an automated spammer would generally ignore all incoming e-mail, while a legitimate user would reply to the e-mail.

Another common use for CAPTCHAs is to prevent the automated use of a wordlist to break usernames and passwords; this form of attack is often referred to as a dictionary attack [4]. On websites requiring the use of a username and password, if there is no means to slow down or prevent automated access, an attacker can simply iterate through a list of common usernames and password variations in an attempt to guess a user's login information. When adding a CAPTCHA to the login system, the user is often required to solve a CAPTCHA after the second or third failed login. While this may present a slight annoyance to a human user who cannot remember his password, it prevents the use of an automated attack.

1.1.3 Classifications of CAPTCHAs

While the most recognized type of CAPTCHA today is the distorted-text CAPTCHA, CAPTCHAs continue to evolve and change in an attempt to create a more secure design. CAPTCHAs not only involve the use of text, but commonly involve the use of images, audio and even video.

Text-based CAPTCHAs

Text-based CAPTCHAs are the most common and oldest form of CAPTCHA. In a text-based CAPTCHA, a sequence of characters is generated and then distorted in a manner that prevents the use of automated techniques. The user is then presented with this sequence of characters and asked to type out the characters. If the user types out the correct sequence of characters within a designated margin of error, then the CAPTCHA is considered to be correctly solved. The advantages of text-based CAPTCHAs are that optical character recognition (OCR) attacks are well understood, humans perform character recognition well, there are adequate character combinations, there are few localization issues, and the design is easily understood and can be generated quickly [5].

One of the first implementations of a text-based CAPTCHA was developed in 1997 by AltaVista, that needed a method for preventing the automatic adding of URLs to the website for indexing [6]. The original intent of the service was to provide website owners a method for submitting their site's URL so that the search engine could index their website Attackers soon realized that they could automate these submissions to skew the results of the search engine [6]. This early CAPTCHA introduced the basic concept of a text-based CAPTCHA, which involves the use of a random string of characters placed on a background and then rendered into an actual image. An example of the CAPTCHA is shown in Fig. 1.

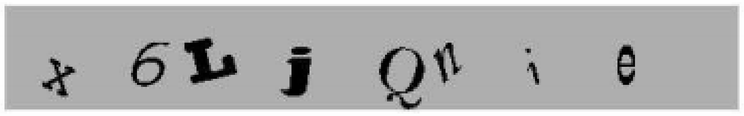


Fig. 1. Example AltaVista CAPTCHA

One of the first ongoing research efforts involved in CAPTCHAs was GIMPY, which was originally designed by researchers at CMU to assist Yahoo! in its problem of bots using their chatrooms to advertise websites [6]. This CAPTCHA involved the use of words selected from the English dictionary, which were then distorted and placed over a random background [6]. The user then had to type a certain number of these words. An example of the GIMPY CAPTCHA is shown in Fig. 2.

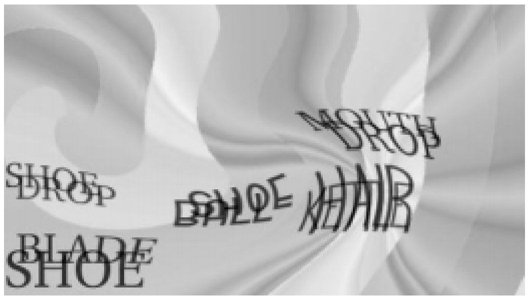


Fig. 2. Example GIMPY CAPTCHA

One of the most quickly growing CAPTCHAs today is reCAPTCHA [7]. reCAPTCHA works by taking words from scanned text from books and displaying them to the user to solve [7]. The system employs the use of a word that was not successfully recognized during scanning along with a control word [7]. If the user types the control word correctly, then they are assumed to have correctly solved the CAPTCHA. By presenting two words, the system is then able to gather an increasingly large number of answers for the unknown words until it can correctly assume the correct text for the unknown words. This system works to solve one of the key problems with text-based CAPTCHAs as the text used has already failed state-of-the-art OCR techniques. reCAPTCHA also works to put CAPTCHAs to good use by helping to digitize text that would otherwise not be digitized. An example of reCAPTCHA is shown in Fig. 3.

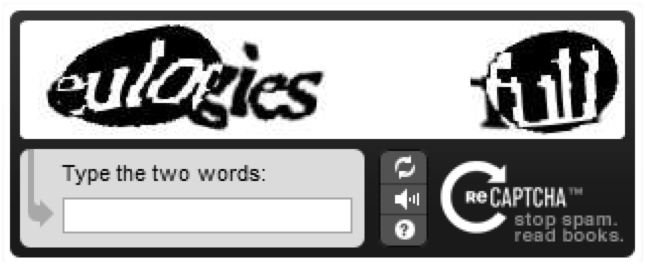


Fig. 3. Example reCAPTCHA Image

With the progression of text-based CAPTCHAs has also come the progression of CAPTCHA attacks. [8],[9],[10] all present techniques for breaking visual text-based CAPTCHAs that were once thought to be secure. One of the common techniques against text-based CAPTCHAs is the use of segmentation algorithms. These segmentation algorithms work by employing techniques such as histogram evaluation, image processing and machine learning to locate characters and then make a guess at the actual character being displayed [5]. The advancement of CAPTCHA attacks has led to the creation of non-traditional CAPTCHA types such as image, audio and video-based CAPTCHAs in an attempt to create a more secure CAPTCHA.

Image-based CAPTCHAs

Image-based CAPTCHAs generally rely on the concept of image classification, where the user must identify an image or group of images. In some forms of this CAPTCHA, the user is required to type out or select a label which matches the images presented. One example of this type of CAPTCHA is ESP-PIX, which presents the user with 4 images and asks the user to select a word from a drop-down menu that is common between the four images [11]. This technique presents a number of issues including object ambiguity, language barriers, an image database which requires the use of text-based labeling and a relatively small list of labels. An example of an ESP-PIX CAPTCHA is shown in Fig. 4.

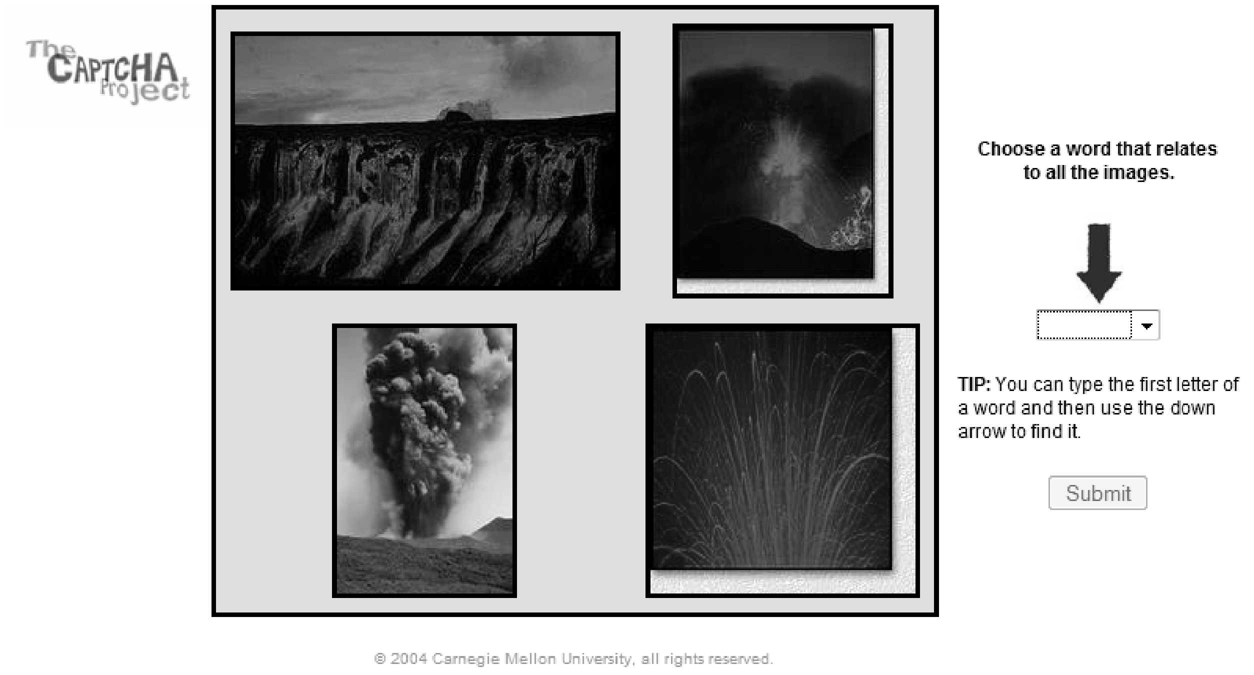

The CAPTCHA Project
Choose a word that relates to all the images.
TIP: You can type the first letter of a word and then use the down arrow to find it.
Submit
© 2004 Carnegie Mellon University, all rights reserved.

Fig. 4. Example ESP-PIX

In another form of image-based CAPTCHA the user is required to click or select all of the CAPTCHAs that match a particular label. One example of this type of CAPTCHA is Asirra, which stands for Animal Species Image Recognition for Restricting Access [12]. In this CAPTCHA, the user is required to select all of the pictures of cats among pictures of cats and dogs. Asirra overcomes a number of issues commonly found in other image-based CAPTCHAs by having a large enough database (over 3 million images from petfinder.com) that grows on a daily basis and by only presenting a small percentage of that database to the public at any given time [13]. However, it has been shown in [14], that a classifier has been designed that can tell cats and dogs apart with 82.7% accuracy which could then be used to defeat this CAPTCHA. An example of an Asirra CAPTCHA is shown in Fig. 5.

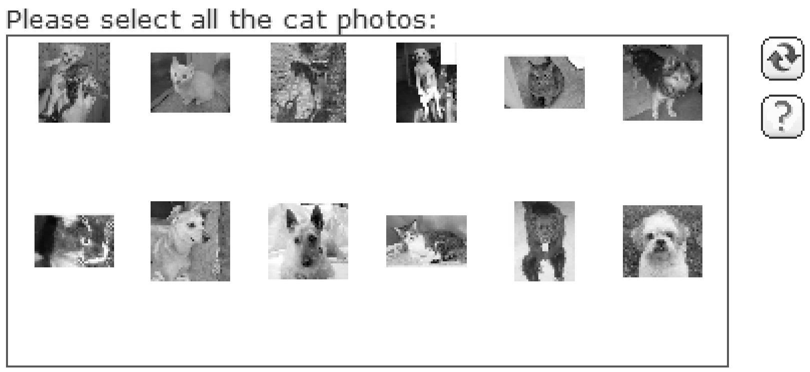


Fig. 5. Example Asirra CAPTCHA

Video-based CAPTCHAs

Another progression from text-based CAPTCHAs is the use of video in CAPTCHA design. In video-based CAPTCHAs, users are generally required to determine keywords or labels for a particular video. The most prevalent work involving the use of video-based CAPTCHAs was performed in [15], where users were asked to watch videos and label them with descriptive words. It was also shown in [15] that this type of CAPTCHA can compete with image-based CAPTCHAs based on human vs. computer success rates of 90% and 13%, respectively.

Audio-based CAPTCHAs

Although visual-based CAPTCHAs may be accessible to the majority of computer users, they remain inaccessible to those users who are visually impaired. To overcome this shortcoming, many websites have implemented audio CAPTCHAs which present the user with a number of words or characters in an audio file and ask the user to type back what was heard. Unfortunately, these audio CAPTCHAs have received the same treatment as more traditional CAPTCHAs and have been subject to numerous attacks [16],[17],[18]. For example, in [16] the audio CAPTCHAs on websites such as Google and Digg were attacked using popular techniques for speech extraction and achieved success rates of up to 71%.

1.2 MOTIVATION AND OUTLINE

Even with the great progression of CAPTCHAs in the last decade, new attacks are created which circumvent even some of the most advanced techniques. Attacks involving OCR and machine-learning segmentation methods have plagued what were once thought to be the most secure of CAPTCHA designs. Because of the growing number of issues with text-based CAPTCHAs there has been a progression of CAPTCHAs moving from text-based to image-based designs. There is still much room for growth in image-based CAPTCHA design. There is very little data regarding human vs. computer ability to solve image-based techniques. With the progression of image-based designs will follow the progression of image distortion techniques. In cases like Asirra, where classifiers have been designed to identify images, image distortion techniques can be applied lower the success rate of such attacks. There has also been little research involving the use of image quality metrics to analyze the effects of non-traditional distortions which could prove useful in the general design of CAPTCHAs.

Additionally, there has been little research regarding the use of face detection in CAPTCHA design. In [19], a face recognition CAPTCHA is designed which presents several techniques involving the use of faces. In the first presented scheme, the user is required to recognize the same image using multiple distortions. This scheme has issues regarding the use of the same base image multiple times in a single CAPTCHA, which could make for easier automated detection. In the second scheme, multiple photos of one person under various distortions are presented [19]. While both of these techniques involve the use of faces, no data is presented regarding the usability of different types of distortions or the success rates of such a CAPTCHA. The paper provides a general proof of concept, but provides little detail regarding the feasibility or efficiency of such a concept.

In this research we present a clickable, image-based CAPTCHA based on the concepts of face detection. In our scheme we present the user with a number of distorted face and non-face images and ask the user to click on all of the face images. The primary goal is to design a face-detection CAPTCHA and prove its ability to compete with other modern CAPTCHAs. We also wish to provide data regarding how image quality metrics respond to traditional and non-traditional distortion types and how this data can be used to better design image-based CAPTCHAs. Additionally, we wish to test our CAPTCHA design against computerized face-detection techniques and determine the success rates of a computer-based attack. Finally, we aim to provide some insights on the success rates of humans and computers based on the types of distortions used. This data can then be used in future research to design distortions that makes computer detection challenging, while allowing humans to provide correct responses.

The thesis is organized as follows:

- Chapters 2 and 3 present general background information for the understanding of image quality metrics and image distortions.
- Chapter 4 completely describes the face-detection CAPTCHA design and goes through each step of the CAPTCHA generation.
- Chapter 5 presents the results of the image quality metrics as well as the human and computer results.
- Chapter 6 presents the conclusions and potential for future work.

Chapter 2

Image Quality Metrics

2.1 Introduction to Quality Metrics

In order to apply an appropriate level of distortion to an image, it is necessary to determine the amount of perceivable distortion. To determine an acceptable level of perceptual distortion, the level of distortion must be verified. Visual image quality is determined using image quality metrics, which often involve the comparison of an original image and a modified image to determine the correlation between the two images. Quality metrics fall into two general categories, subjective metrics and objective metrics.

2.2 Subjective Metrics

Subjective image quality metrics involve the use of human testers to visually analyze the quality of a particular image. One such method is the MOS (Mean Opinion Score), which is commonly used in testing audio quality as well as image/video quality. The MOS involves asking human testers to rate the signal's quality on a scale of 1-5. Once the results from all testers are compiled, the mean of these scores is then taken to form the MOS. Since the subjective tests are based on human responses to image quality, the results of these metrics are the best in determining the true visual quality of an image. The downside of these subjective methods is that in order to achieve accurate results, a large sample of testers is required to analyze a large number of images. Because of the cost and time required to perform such tests, it is much more practical to use objective automated image quality metrics. In the next section we will provide an overview of commonly used objective image quality metrics.

2.3 OBJECTIVE METRICS

Objective image quality metrics are commonly broken down into conventional signal quality measures, which analyze the amount of energy in a particular signal and more modern metrics, which often involve the use of structural properties and/or HVS (Human Visual System) properties. The methods based on the Human Visual System properties can be further broken down into those which are based on the low-level properties of the HVS and those which are based more on high-level properties, such as the general structure of the image.

2.3.1 CONVENTIONAL STATISTICS ORIENTED METRICS

The most commonly known of the conventional signal quality measures is the MSE (Mean Squared Error), which measures the energy of the distortion. The MSE simply works by averaging the square of the differences in pixel intensity values between two images. The MSE for two-dimensional image can be written as follows:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\|I(i,j) - K(i,j)\|^2 \tag{2.1}$$

where *I(i,j)* and *K(i,j)* are the two *m* x *n* images. In the case of testing distortions, *I(i,j)* is the original image and *K(i,j)* is the modified image.

Another metric based on the energy of distortion is the PSNR (Peak Signal-to-Noise Ratio), which measures the ratio of the maximum intensity value to the amount of noise present in the image. The PSNR is based on the MSE and can be written as follows:

$$PSNR = 20 \cdot log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad (2.2)$$

where MAX_I represents the maximum possible pixel value. In most grayscale images, MAX_I is equal to 255.

Although it is one of the most commonly used methods in determining the quality of a signal, some studies have shown that that the MSE does not perform well in determining the loss of perceptual quality [20][21]. It is shown in [21] that the MSE can result in nearly identical values for images with widely varying degrees of image quality. In [22], it was shown that the PSNR performs well in tests involving the use of Gaussian noise, and has indistinguishable results in this category in comparison to advanced HVS-based metrics. Because of the weaknesses found in the MSE and corresponding PSNR, methods incorporating the use of the human visual system have been developed in an effort to improve the ability to correctly measure perceptual image degradation. Although these newer methods generally provide much greater results over a wide range of quality tests, the MSE and PSNR do perform well considering their simplicity in implementation.

2.3.2 Low-Level HVS Metrics

Low-level HVS metrics take into account how visually perceptive a distortion is through the use of models to characterize HVS properties such as contrast sensitivity and perceived image contrast. These metrics often take into account the visibility of distortion under multiple viewing conditions to determine the overall quality of the distorted image in relation to the source image. While it is advantageous that these metrics have the ability to be adapted for specific platforms, it also presents a downside in that they often have to be adjusted for each implementation or require the use of training data.

Noise Quality Measure

One algorithm based on low-level properties of the HVS is the noise quality measure (NQM). This quality metric performs modifications to the image in order to simulate the appearance to a human [23]. The NQM is based on the idea that the psychovisual effects of image filtering and noise-based distortion are separate [23]. The NQM works by first passing both the original and modified images through a contrast pyramid, based on [24], which computes the contrast of an image while taking into account various visual effects [23]. After being passed through the contrast pyramid, the NQM is then found by computing the signal-to-noise ratio of both the source and distorted images after they have been restored using a restoration algorithm [23]. In doing so, both images are analyzed as if they were viewed by a human. The NQM differs from many other methods in the fact that it involves the use of analyzing a modified original image.

The NQM can be written as follows:

$$NQM(dB) = 10log_{10}\left(\frac{\sum_x \sum_y O_s^2(x,y)}{\sum_x \sum_y \left(O_s(x,y) - I_s(x,y)\right)^2}\right) \tag{2.3}$$

where $O_s(x,y)$ represents the model restored image and $I_s(x,y)$, which represents the restored distorted image [23].

VSNR

The VSNR (Visual Signal-to-Noise Ratio) works by modeling the masking effects which cause difficulty in detecting distortions, the perceived contrast of distortions and the structural degradation of the image [25]. It is a wavelet-based method that takes advantage of both low and mid-level HVS properties. The VSNR works by first measuring the contrast thresholds for the detection of distortion in the image. The distortion is then measured and if the detected distortion is found to be below the determined threshold, the image is given a perfect VSNR rating. If the distortions are found to be above the threshold, then the low-level property of perceived contrast and mid-level property of global precedence are measured and modeled as Euclidean distances [25]. The linear sum of these two distances is then used to determine the VSNR. The advantages of the VSNR are a low computational complexity, low memory requirements, the use of a model based on luminance and the ability to compensate for a variety of viewing angles and conditions. It was shown in [25] that the VSNR is competitive with many of the other modern metrics, but that it is extremely sensitive to geometric distortion and transformation.

The VSNR can be written as follows:

$$VSNR = 10log_{10}(\frac{C^2(I)}{(VD)^2}) \tag{2.4}$$

$$\mathrm{VD} = \alpha \mathrm{d_{pc}} + (1-\alpha)\frac{\mathrm{d_{gp}}}{\sqrt{2}} \tag{2.5}$$

where C(I) represents the RMS contrast of the original image I, α represents the contribution of each Euclidean distance, d_{pc} is the amount of perceived contrast of the distortions and d_{gp} is the amount of disruption caused to the global precedence.

2.3.3 High-Level HVS Metrics

In general, algorithms based on high-level HVS properties perform quality assessment based on the overall structure of an image. Among these methods are the UQI (Universal Quality Index) [21], the SSIM (Structural SIMilarity) [20] and the VIF (Visual Information Fidelity) models [26].

UQI

Both the UQI and the SSIM have strengths that lie in their relationship to the Human Visual System and its analysis of the structural integrity of an image on a local level. These metrics do not specifically use a particular HVS-based model, but instead work by analyzing the high-level properties of an image. They work by taking into consideration the two images' differences in linear correlation, luminance and contrast [21]. The UQI for an original image, x, and a modified image, y, can be written as follows:

$$Q = \frac{\sigma_{xy}}{\sigma_x \sigma_Y} \cdot \frac{2\overline{x}\,\overline{y}}{(\overline{x})^2 + (\overline{y})^2} \cdot \frac{2\sigma_x \sigma_Y}{\sigma_x^2 + \sigma_y^2} \tag{2.6}$$

where:

$$\mathrm{x} = \{\mathrm{x_i} | \mathrm{i} = 1,2, \dots, \mathrm{N}\} \tag{1.1}$$

$$\mathrm{y} = \{\mathrm{y_i} | \mathrm{i} = 1,2, \dots, \mathrm{N}\} \tag{1.1}$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1.2}$$

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{1.3}$$

$$\sigma_\mathrm{x}^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{\mathrm{x}})^2 \tag{1.4}$$

$$\sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{1.5}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})\,(y_i - \bar{y}) \tag{1.6}$$

Each multiplicative term in the UQI represents one of the three components being analyzed to measure the differences between two images. The first component of the product term represents the linear correlation between x and y and takes on a range from -1 to +1 [21]. The second component of the product term represents the difference between the mean luminance of x and y and takes on a value between 0 and 1 [21]. The third component of the product term represents the difference in contrast and takes on a value between 0 and 1 [21]. The UQI is important because it forms the basis for the framework for the SSIM and other structural-based image quality metrics.

SSIM

The SSIM was introduced in [20] as a general form of the UQI. While both metrics are very similar in origin, the SSIM is preferred over the UQI due to the UQI's instability when $\sigma_x^2 + \sigma_y^2$ or $(\bar{x})^2 + (\bar{y})^2$ approaches 0. The SSIM can be written as:

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} \tag{1.7}$$

where

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu^2{}_x + \mu^2{}_y + C_1} \tag{1.8}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma^2{}_x + \sigma^2{}_y + C_2} \tag{1.9}$$

$$s(x,y) = \frac{2\sigma_{xy} + C_3}{\sigma_{xy} + C_3} \tag{1.10}$$

μ_x and μ_y are equivalent to $\bar{x}$ and $\bar{y}$ from the UQI. $C_1 = (K_1 L)^2$ represents a constant to prevent the previously mentioned instability. C_2 , C_3 can be represented as $C_2 = (K_2 L)^2$. L represents the range of pixel values for the image and $K_1 \ll 1$ and $K_2 \ll 1$ are generic constants. In [20], the weighting variables α, β and γ are set equal to 1 and $C_3 = C_2/2$ to simplify the expression, giving the resulting equation for the SSIM:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_x\sigma_y + C_2)}{({\mu^2}_x + {\mu^2}_y + C_1)({\sigma^2}_x + {\sigma^2}_y + C_2)} \tag{1.11}$$

The SSIM expression is then applied to the image using a sliding window in order to determine the local SSIM. The average of the SSIM values are then used to determine the overall image quality.

In general, the SSIM improves greatly on traditional methods such as the MSE, but does not do well in spatial related image modifications. For example, the SSIM yields better results for images with intensity modifications over images which have been spatially shifted, which is not consistent with human results [27].

In order to improve upon the SSIM, several modified versions of the SSIM have been proposed in hopes of solving some of the SSIM's inadequacies. In [28] the MS-SSIM* is introduced in order to improve upon the previous fix to the limitation found in the UQI. [28] also analyzes the ability of image-quality metrics to predict the threshold capability of a particular image. To improve upon the areas of translation, scaling and rotation, the concepts of the SSIM were applied in the wavelet domain to create the Complex Wavelet-SSID (CW-SSIM) in [29]. Additionally, in [30] a Multi-Scale SSIM (MS-SSIM) is introduced, which improves the SSIM's ability to handle varying resolutions and viewing distance. The multi-scale aspect of the MS-SSIM works by taking both the original and the modified image, applying a low-pass filter and then down-sampling the resulting images by a factor of 2. This operation is performed M-1 times, with each operation representing a particular scale factor. At the jth scale, the previously mentioned c(x,y) and s(x,y) are each computed and written as $c_j(x,y)$ and $s_j(x,y)$. The luminance comparison is taken at the highest scale (M) and is written as $l_M(x,y)$. The new form of the SSIM is written as follows:

$$MS.SSIM(x,y) = [l_m(x,y)]^{\alpha_j} \cdot \prod_{j=1}^{M} [c_j(x,y)]^{\beta_j} [s_j(x,y)]^{\gamma_j} \qquad (1.12)$$

IFC

Although the IFC (Information Fidelity Criterion) metric is similar to conventional models in its analysis of images purely as signals, its strengths lie in its use of NSS (Natural Scene Statistics) models. Natural scene images generally are thought of as those images captured from cameras, camcorders and similar devices. Natural scene images do not include images such as paintings, computer-generated images and other images not found naturally in the world. The IFC metric works by modeling the image using NSS models and then analyzing the distortion of the image after it passes through a distortion channel. The IFC models the source images using the wavelet-based GSM (Gaussian Scale Mixtures) model [31]. The GSM model is described in detail in [32]. The wavelet-based distortion model used in [31] works by analyzing the amount of blur and additive noise found in the channel. The resultant IFC is then a measurement of the mutual information found between the source and received image. Some features of the IFC are that it does not involve the use of parameters or training data and works on all viewing platforms. A disadvantage of the IFC (and other metrics based on this model) is the computational complexity involved in performing the wavelet decomposition, which causes a significant increase in computation time when compared to previously discussed methods.

The IFC can be expressed as:

$$IFC = \sum_{k \in subbands} I\left(C^{N_k,k}; D^{N_k,k} | s^{N_k,k}\right) \tag{1.13}$$

where

$$I(C^N; D^N | s^N) = \frac{1}{2}\sum_{i=1}^{N} log_2\left(1 + \frac{g_i^2 s_i^2 \sigma_u^2}{\sigma_v^2}\right) \tag{1.14}$$

C represents a GSM modeling of the signal, D represents the distortion model and S represents the source model. $\sigma_u^2$is the variance of a zero-mean Gaussian Random Field used for modeling the source model and σ_v^2 is the variance of a zero-mean Gaussian Random Field used for modeling the distortion model. The IFC is further explained in the details of the VIF.

VIF

In [26], the VIF (Visual Information Fidelity) is introduced, which introduces some improvements to the IFC by means of normalization. The VIF essentially takes the result of the IFC and normalizes it by the reference information in order to adjust for variation in the amount of perceptual image distortion that a particular image may hold [26].

The source model of the VIF/IFC is expressed as:

$$C = S \cdot U\{S_i \cdot \overrightarrow{U_i}: i \in I\} \tag{1.15}$$

where S is a random field of positive scalars, U is a zero-mean Gaussian random field with covariance C_U and I is the image [26].

The distortion model of the VIF/IFC is expressed as:

$$D = GC + V = \{g_i \vec{C_i} + \vec{V_i}: i \in I\} \tag{1.16}$$

where C is a random field from the source signal, G is a deterministic scalar gain field, I is the image and V is a Gaussian noise field with a mean of zero and a covariance $C_V = \sigma_V^2 I$ [26].

The VIF can be expressed as:

$$VIF = \frac{\sum_{j \in subbands} I\left(\vec{C}^{N,j}; \vec{F}^{N,j} | s^{N,j}\right)}{\sum_{j \in subbands} I\left(\vec{C}^{N,j}; \vec{E}^{N,j} | s^{N,j}\right)} \tag{1.17}$$

where

$$I\left(\vec{C}^{N}; \vec{E}^{N} | s^{N}\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{M} log_2\left(1 + \frac{s_i^2 \lambda_k}{\sigma_n^2}\right) \tag{1.18}$$

$$I\left(\vec{C}^{N}; \vec{F}^{N} | s^{N}\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{M} log_2\left(1 + \frac{g_i^2 s_i^2 \lambda_k}{\sigma_v^2 + \sigma_n^2}\right) \tag{1.19}$$

The information in the numerator of the VIF represents the visual information that can be extracted from the received image, while the information in the denominator is the normalization factor, which represents the information that can be extracted from the source image. Tests in [26] showed that while the VIF has issues regarding computation time, it does perform on par with or better than the PSNR and SSIM and that it performs exceptionally well in tests involving multiple forms of distortion.

2.3.4 **METRIC PERFORMANCE**

In [22] a test of full reference image quality metrics was performed on 29 source images and a 779 resultant distorted images. The tests performed include JPEG2000/JPEG compression, white noise, Gaussian blur, and the simulation of distortion via a fast fading Rayleigh channel. The results of the test found that the SSIM-MS, IFC and VIF outperformed all other grayscale oriented algorithms. The test also showed that among the 10 different metrics tested, that the VIF yielded the most accurate results among the tests. Each of the metrics has its own strengths and weaknesses and respond differently to various testing conditions. While [22] tested several common distortion types, there are still far too many distortions that remain untested to justify the use of only a single image quality metric.

Chapter 3

IMAGE DISTORTIONS

This section will provide an analysis of commonly used types of image distortion. It will also provide example images of the actual visual impact of each distortion using multiple levels of distortion.

3.1 GEOMETRIC TRANSFORMATIONS

3.1.1 PROJECTIVE TRANSFORMATIONS

Projective transforms are transformations which represent a change in perspective or viewpoint. These transformations maintain the properties of incidence and cross-ratio, but do not maintain sizes or angles. In a projective transformation straight lines will remain straight.

For a general projective transformation, transformed coordinates of x and y are represented as u and v, where:

$$[up\ vp\ wp] = [x\ y\ w]T \tag{2.1}$$

$$u = \frac{up}{wp} \tag{2.2}$$

$$v = \frac{vp}{wp} \tag{2.3}$$

and T is a 3x3 matrix.

The following sections will show how modifying different values within T in different ways results in specific types of transformations.

Affine Transformations

Affine transformations are a subset of projective transformations; they preserve both ratio of distances along a line and the collinearity between points. In affine transformations the original length and angles within the image will not be preserved, and therefore the shapes of objects will be modified. Affine transformations generally consist of translation, scaling, shearing, rotation and combinations of these methods.

In affine transformations:

$$[u\ v\ ] = [x\ y\ 1]T \tag{2.4}$$

$$T = \begin{bmatrix} \text{x scale\_factor} & \text{y shearing\_coef} & 0 \\ \text{y shearing\_coef} & \text{y scale\_factor} & 0 \\ \text{x displacement} & \text{y displacement} & 1 \end{bmatrix} \tag{2.5}$$

Translation

The process of translation involves moving all pixel values by a constant value without any other form of modification. Translating an image effectively moves it from one location on the plane to another. Fig. 6 shows an image containing varying levels of x/y displacement.

The transformation matrix used for the displacement of pixel values is as follows:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \text{x displacement} & \text{y displacement} & 1 \end{bmatrix} \tag{2.6}$$

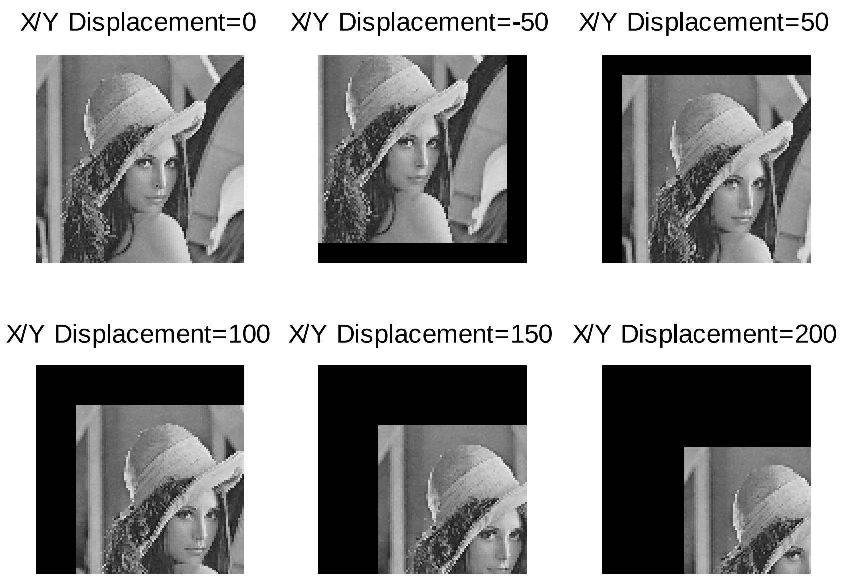


Fig. 6. Translation

Scale

Changing the scale of an image proportionally increases the size of the image along an axis by a particular scale factor. Generally the scale of an image is modified proportionally along both the x and y axes, but these values may be changed independently as well. Fig. 7 shows an image scaled on the X axis and Fig. 8 shows an image scaled on the Y axis.

The transformation matrix used for the modification of scale is as follows:

$$T = \begin{bmatrix} \text{x scale\_factor} & 0 & 0 \\ 0 & \text{y scale\_factor} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.7)$$

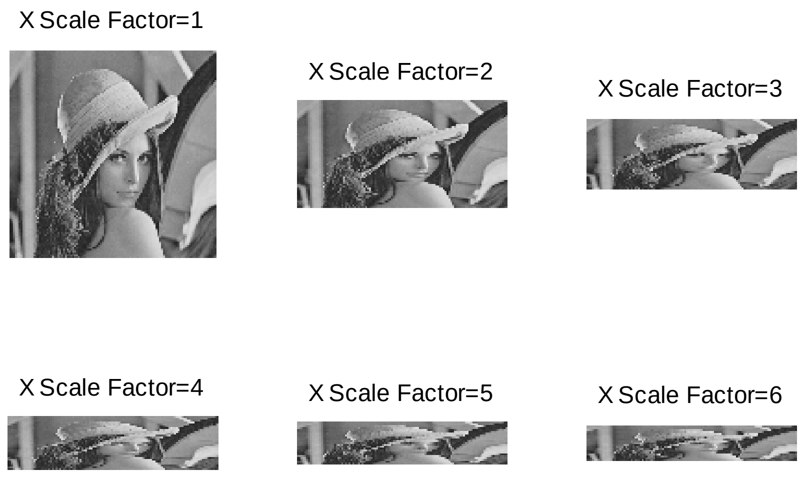


Fig. 7. Scaling X-axis

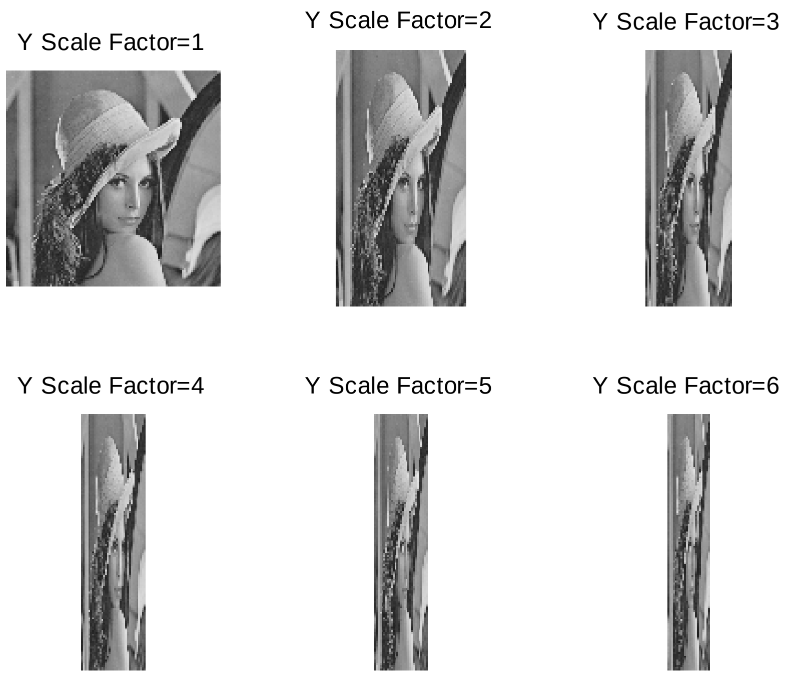


Fig. 8. Scaling Y-axis

Shear

The process of shearing involves holding all pixels on an arbitrary line are fixed, while other points are shifted parallel to that line by a distance proportional to the perpendicular distance from that fixed line [33]. Positive x coefficients cause the bottom of the image to move to the right, while negative x coefficients cause the top of the image to move to the right. Positive y coefficients cause the right side of the image to move down, while negative y coefficients move the left side of the image down. Fig. 9 shows the results of changing the X shearing coefficient and Fig. 10 shows the results of changing the Y shearing coefficient.

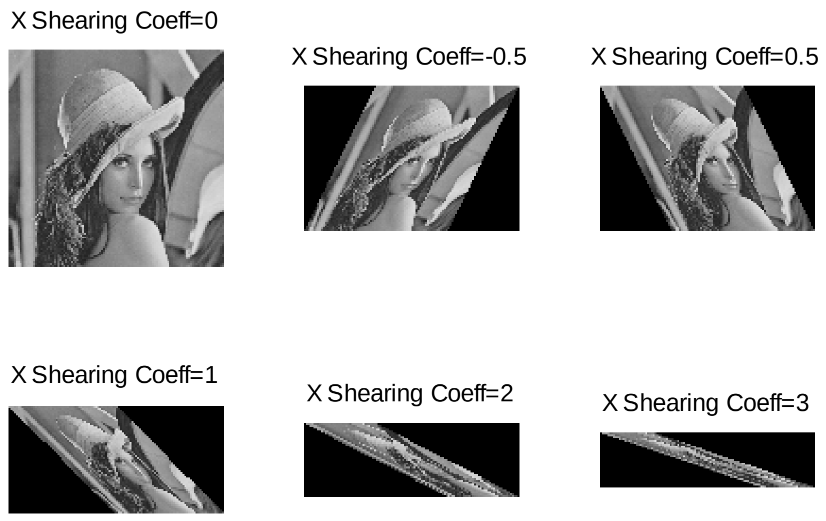


Fig. 9. Shearing X Coefficient

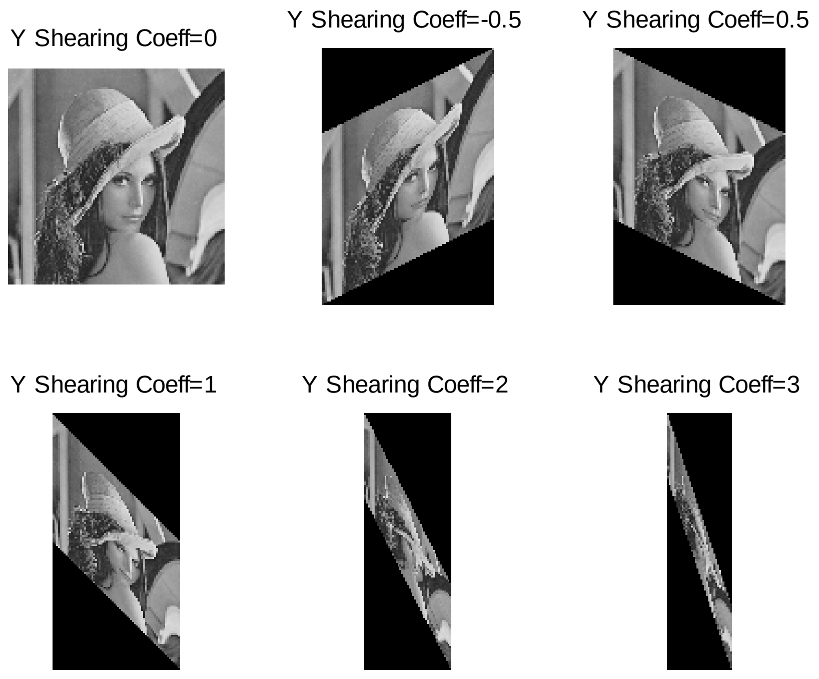


Fig. 10. Shearing Y Coefficient

The transformation matrix used for shearing is as follows:

$$T = \begin{bmatrix} 1 & \text{y shearing\_coef} & 0 \\ \text{x shearing\_coef} & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \tag{1.1}$$

Rotation

The process of rotation involves rotating an object about its center pixel location.

The transformation matrix used for rotation is as follows:

$$T = \begin{bmatrix} \cos(\text{angle}) & -\sin(\text{angle}) & 0 \\ \sin(\text{angle}) & \cos(\text{angle}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

The angle should be entered in the form of radians, where counterclockwise rotation correspond to positive radian values and clockwise rotation correspond to negative radian values. Fig. 11 shows an image rotation using six different settings.

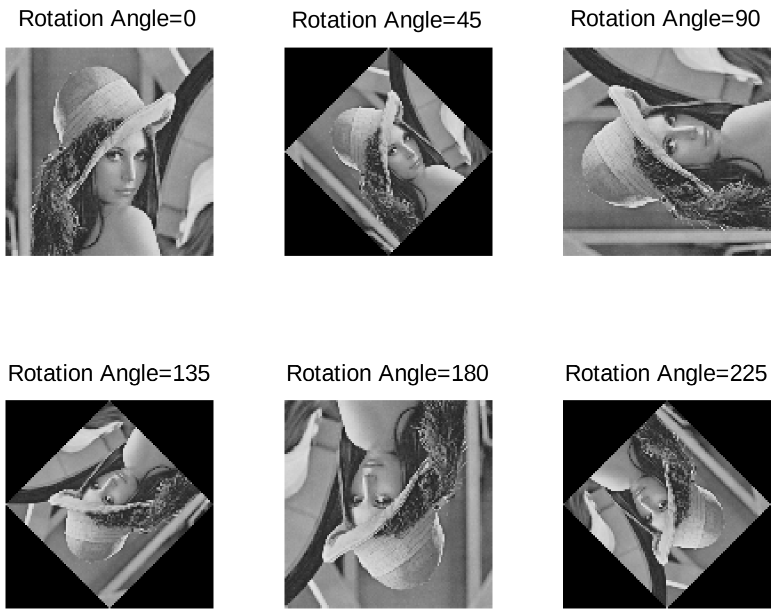


Fig. 11. Rotation

Resolution Reduction

Reducing the resolution of an image simply involves down-sampling the image by a particular scale factor and then up-sampling back to the original resolution. In the example used below, no interpolation is used when up-scaling the images; this method results in the greatest amount of visual distortion. Fig. 12 shows an image under increasing levels of resolution reduction.

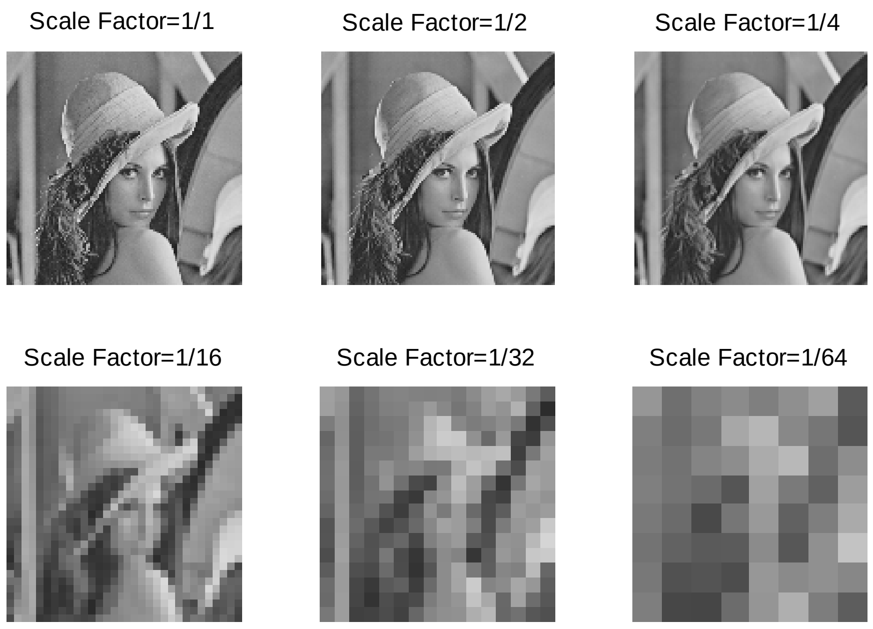


Fig. 12. Scale Factor Modification

3.1.2 PIECEWISE LINEAR TRANSFORMATIONS:

Piecewise linear transformations consist of applying different transformations to different parts of the image. This is effectively performed by splitting the image into multiple sub-images, performing modifications to each sub-image and then merging the sub-images together. Fig. 13 has been modified by leaving the left half of the image untouched and horizontally scaling the right half of the image by different scale factors.

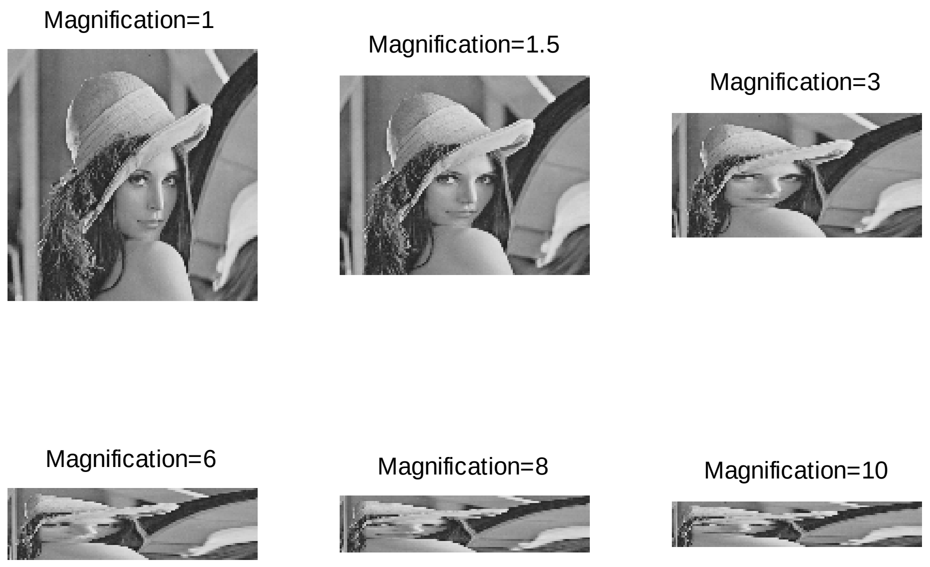


Fig. 13. Piecewise Linear Transformation

3.1.3 **RADIAL TRANSFORMATION**

Barrel Transformation:

In a barrel transformation, the magnification scaling factor of the image decreases in relation to the distance from the center axis. The transformation is performed by changing the x,y coordinates into polar coordinates and then scaling the r coordinates as follows:

$$r' = r + (a * r^3) \tag{1.3}$$

where a is a scaling factor and larger values of a result in a larger distortion effect. Fig. 14 shows the barrel distortion using six different values of a (amplitude). Fig. 14 shows an image under various levels of barrel distortion.

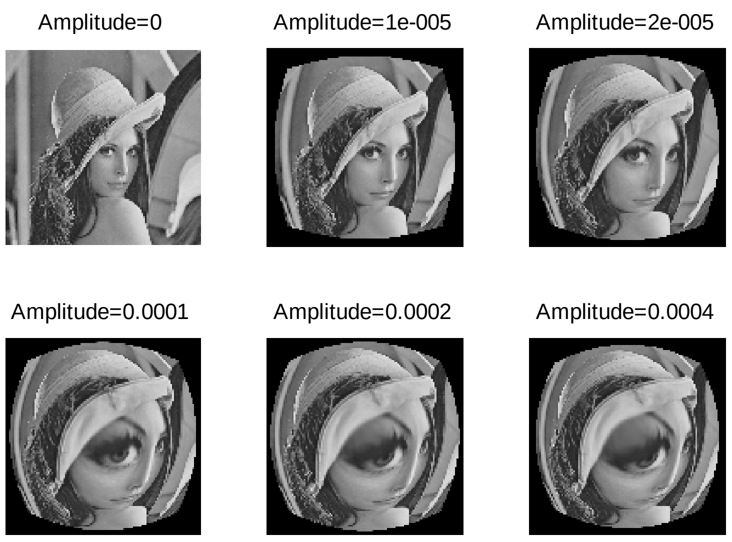


Fig. 14. Barrel Image Effect

Pin-Cushion Transformation:

In a pin-cushion transformation, the magnification scaling factor of the image increases in relation to the distance from the center axis. The transformation is performed by supplying negative values for the barrel transformation equation. Fig. 15 shows an image under various levels of pin-cushion distortion.

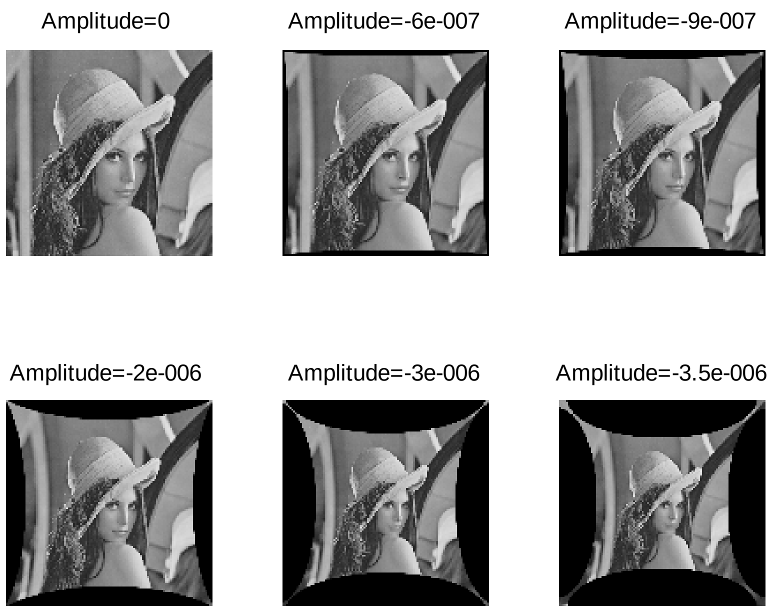


Fig. 15. Pin Cushion Effect

3.1.4 **Polynomial Transformation**

A polynomial transformation is a transformation involving the remapping of linear coordinates. These transformations are generally used to make corrections in size or general orientation. In cases where the image is in correct orientation, this method can also be used to distort the image. Unlike the first-order projective transformations, the weighting coefficients in the T matrix of second-order transformations have no physical counterparts, making the choosing of coefficients more difficult [34]. A second-order polynomial distortion can be seen in Fig. 16.

A 2nd-order polynomial transformation where u and v represent the modified x and y coordinates takes the following form:

$$[u\ v] = [1 \quad x \quad y \quad xy \quad x^2 \quad y^2]T \tag{1.4}$$

where T is the following 6x2 matrix with weighting coefficients a0-a5 and b0-b5:

$$T = \begin{bmatrix} a0 & b0 \\ a1 & b1 \\ a2 & b2 \\ a3 & b3 \\ a4 & b4 \\ a5 & b5 \end{bmatrix} \tag{1.5}$$

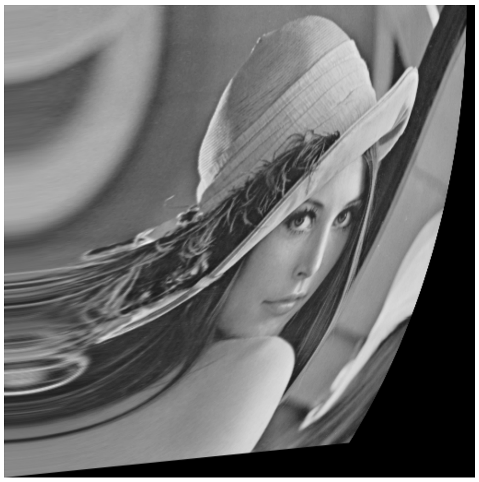


Fig. 16. 2nd-Order Polynomial Stretching

3.2 **NOISE GENERATION**

Noise can be considered to be a addition of a degradation signal to the original image signal. Noise generally takes on two forms, either periodic or random. MATLAB's built-in function, *imnoise*, is used to generate both additive and multiplicative noise.

3.2.1 **SALT & PEPPER NOISE**

The most basic form of random noise is salt & pepper noise, which simply replaces random pixels in the image with black and white pixels. MATLAB's implementation of salt and pepper noise takes on a single parameter, which indicates the percentage of the picture that is affected by the noise. An example of various levels of salt and pepper noise is shown in Fig. 17.

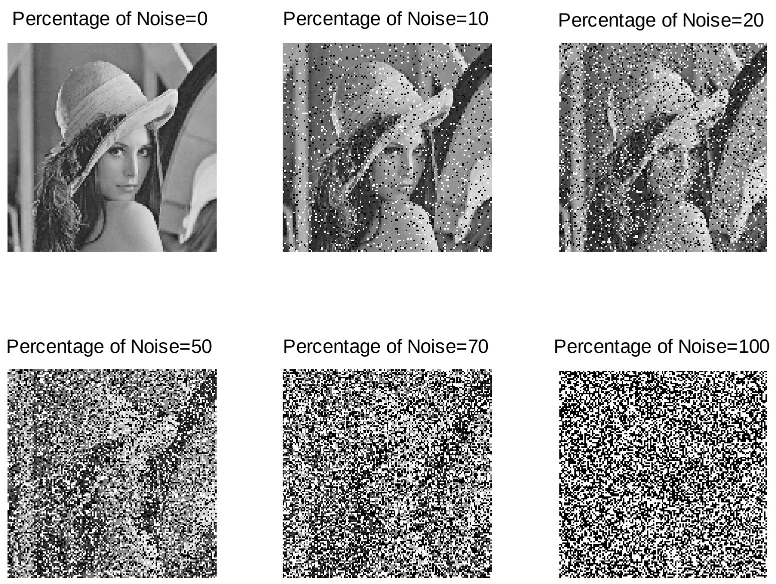


Fig. 17. Salt & Pepper Noise

3.2.2 **GAUSSIAN NOISE**

Gaussian noise is a form of additive white noise which is given a normal distribution. Both the mean and variance of the noise can be modified to change the intensity of the noise. Fig. 18 shows an image modified by zero-mean Gaussian noise using different variance values.

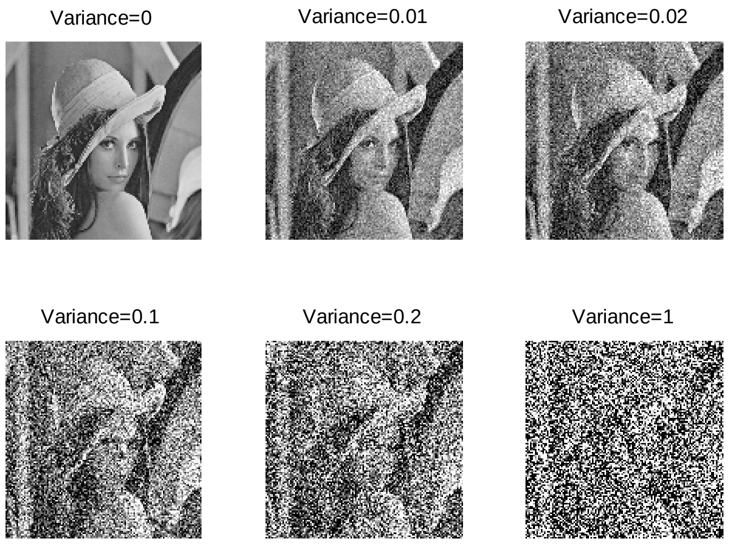


Fig. 18. Gaussian Noise - Zero Mean

3.2.3 **SPECKLE NOISE**

Speckle noise is a form of multiplicative noise. In multiplicative noise, the random noise values are multiplied by the pixel values of the images rather than being added to the pixel values. MATLAB's implementation produces a zero-mean, normally distributed speckle noise with a variance parameter. Fig. 19 shows an image distorted by speckle noise using different variance values.

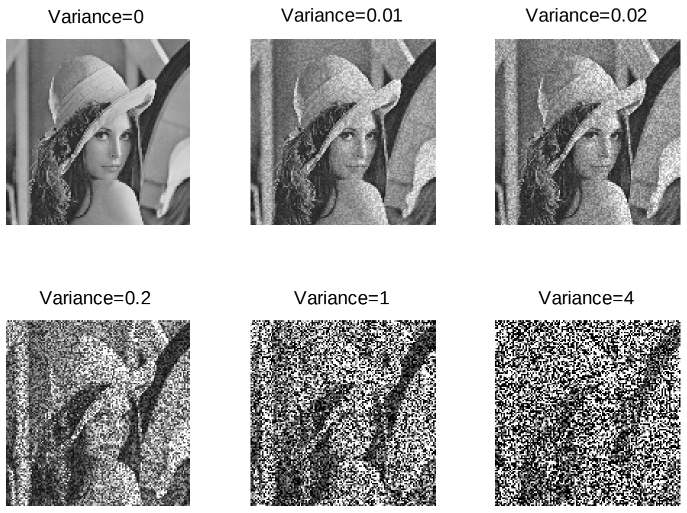


Fig. 19. Speckle Noise

3.2.4 PERIODIC NOISE

Unlike most forms of noise, which are based on a random pattern, periodic noise takes on a predictable pattern. Periodic noise is a global effect to the image and cannot be removed by modeling local degradations [35]. Fig. 20 replicates the appearance of scan-lines.

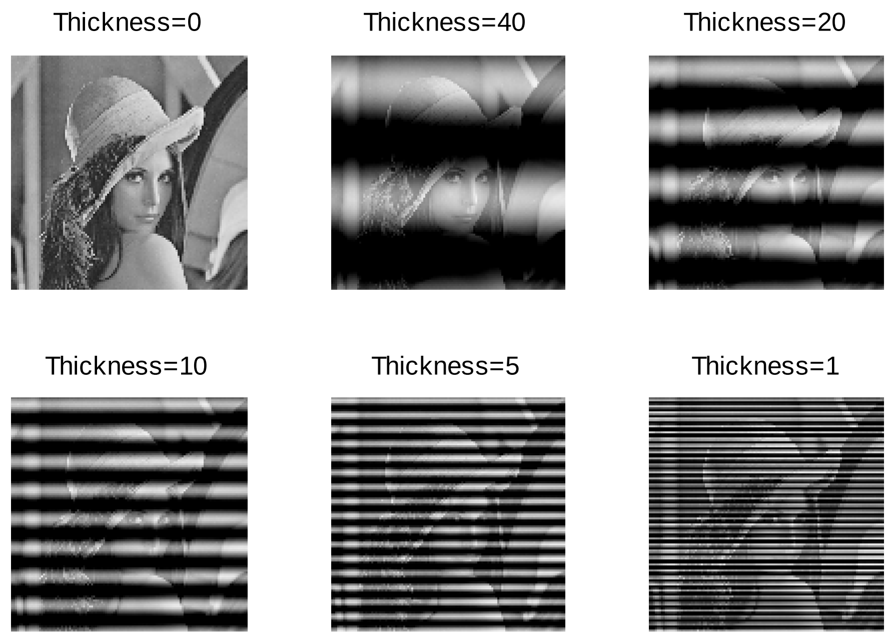


Fig. 20. Periodic Noise - Horizontal

Periodic noise can be produced by producing a noise (n) based on a periodic function (such as sin or cos) and adding it to the original image.

3.3 MATHEMATICAL MORPHOLOGY

Mathematical Morphology is a technique for operating on an image using concepts oriented around set theory. In a simple binary image, white pixels generally represent the foreground and black pixels represent the background. In a grayscale image, pixels are mapped to a three-dimensional space based on the intensity of each pixel values. In this case, a three-dimensional structuring element is also used. The two basic transformations of mathematical morphing are dilation and erosion.

3.3.1 **DILATION**

In order to perform a morphological operation the image and an array (often referred to as the kernel) are both required. The kernel is a structuring element that determines how the dilation operates. In order to perform the dilation operation, the kernel is placed over each background pixel. If any of the pixels in the kernel overlap with a foreground pixel, then the selected background pixel is turned into a foreground pixel. This effectively causes the pixels to grow and expand into one another.

Flat Dilation is defined as:

$$A \oplus B = \bigcup_{b \in B} A_b \tag{1.6}$$

Non-Flat (gray-scale) Dilation is defined in [36] as:

$$f \oplus b(x,y) = max\{f(x - x', y - y') + b(x', y') | (x', y') \in D_b\} \tag{1.7}$$

where D_b is the domain of b.

Fig. 21 is an example of a non-flat dilation performed using a "sphere-shaped" structuring element with a height of 5 and a varying radius.

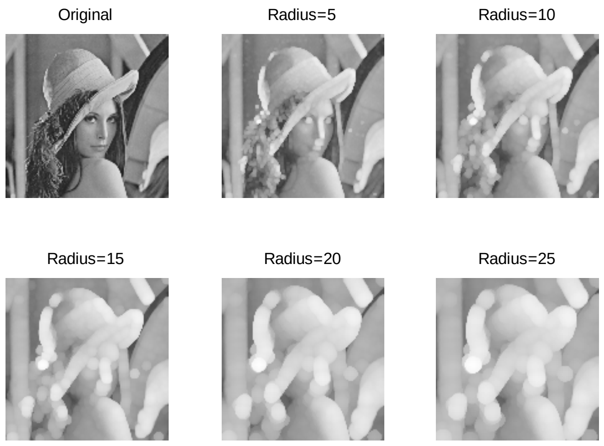


Fig. 21. Non-Flat Dilation - Sphere Shaped Structure

3.3.2 Erosion

The erosion operation works in an opposite manner to that of the dilation. Instead of causing pixels to expand into one another, erosion causes areas of pixels to shrink. To perform this operation, the kernel is placed over each foreground pixel. If any of the pixels in the kernel overlap with a background pixel, then the selected pixel is turned into a background pixel.

Flat Erosion is defined as:

$$A \ominus B = \bigcap_{b \in B} A_{-b} \tag{1.8}$$

Non-Flat Dilation is defined in [36] as:

$$f \ominus b(x, y) = min\{f(x + x', y + y') - b(x', y') | (x', y') \in D_b\} \tag{1.9}$$

where D_b is the domain of b.

Fig. 22 is an example of a non-flat erosion performed using a "sphere-shaped" structuring element with a height of 5 and a varying radius.

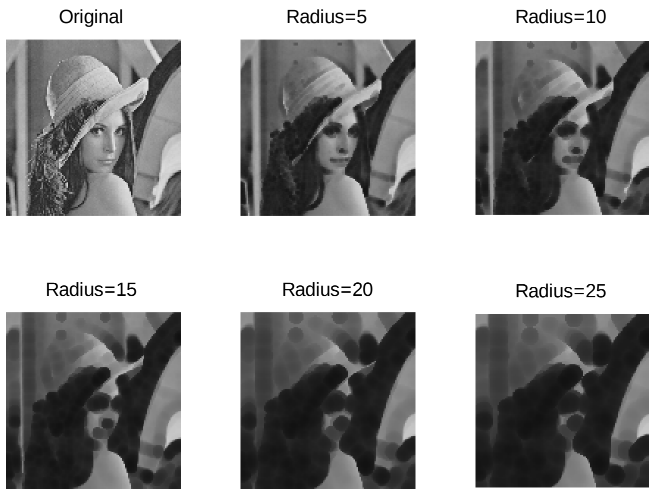


Fig. 22. Non-Flat Erosion - Sphere Shaped Structure

3.3.3 **Opening**

The opening operation consists of performing an erosion operation and then a dilation operation, while using the same kernel for both operations. The effect of these two operations is that sharp edges are removed from the image.

Flat and non-flat opening is defined as:

$$A \circ B = (A \ominus B) \oplus B \tag{1.10}$$

Fig. 23 is an example of a non-flat opening performed using a "sphere-shaped" structuring element with a height of 5 and a varying radius.

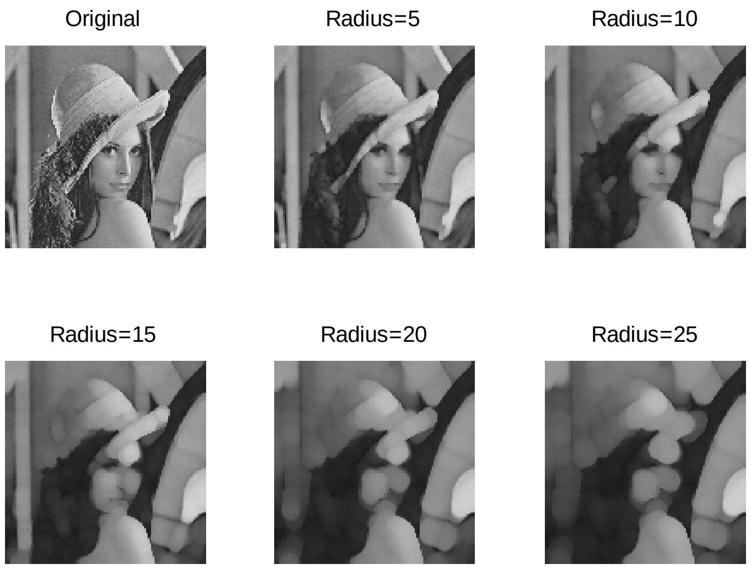


Fig. 23. Non-Flat Opening - Sphere Shaped Structure

3.3.4 **Closing**

The closing operation is the opposite of an opening operation and consists of performing a dilation operation and then an erosion operation, while using the same kernel for both operations. The effect of the two operations is that small holes and gaps in the image are filled.

Flat and non-flat closing is defined as:

$$A \bullet B = (A \oplus B) \ominus B \tag{1.11}$$

Fig. 24 is an example of a non-flat closing performed using a "sphere-shaped" structuring element with a height of 5 and a varying radius.

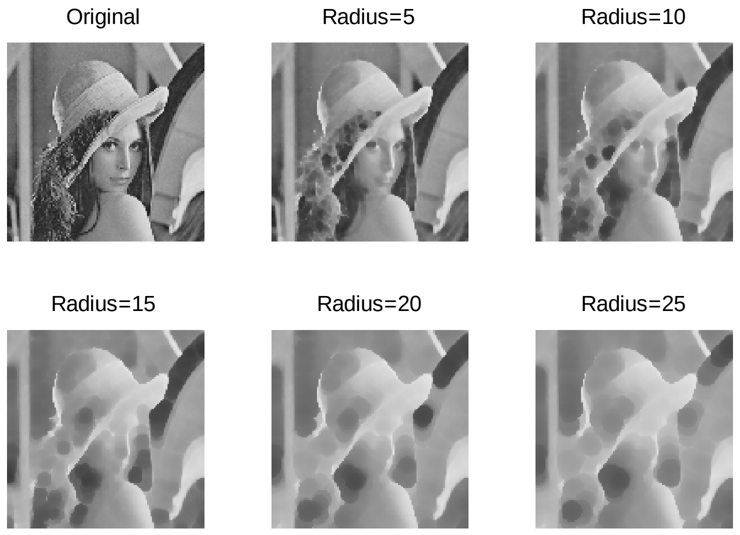


Fig. 24. Non-Flat Closing - Sphere Shaped Structure

3.4 FREQUENCY-BASED MODIFICATIONS

Performing certain operations in the frequency domain can be much less computationally intensive than in the spatial domain. For example, performing basic filtering operations is much simpler in the frequency than it is in the spatial domain. In order to perform filtering in the frequency domain, the FFT is first performed on the image, the filter is then multiplied by the transformed image and then the inverse FFT is performed.

3.4.1 LOW-PASS FILTERING

The blurring operation is performed through the use of a low-pass filter. The Gaussian blur works by assigning each pixel a weighted average value based on the pixel in that pixel's neighborhood. The selected pixel's value receives the greatest weighting constant and nearby pixels receive increasing smaller weights in proportion to their distance from the selected pixel.

The 2D Gaussian blur is defined as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1.12}$$

where x is the horizontal distance from the origin, y is the vertical distance from the origin and σ is the standard deviation.

We apply a rotationally symmetric Gaussian low-pass filter of a set size over the image with varying degrees of standard deviation. When keeping the filter size fixed, increasing the standard deviation increases the blur of the image. Fig. 25 uses a fixed 20x20 window size.

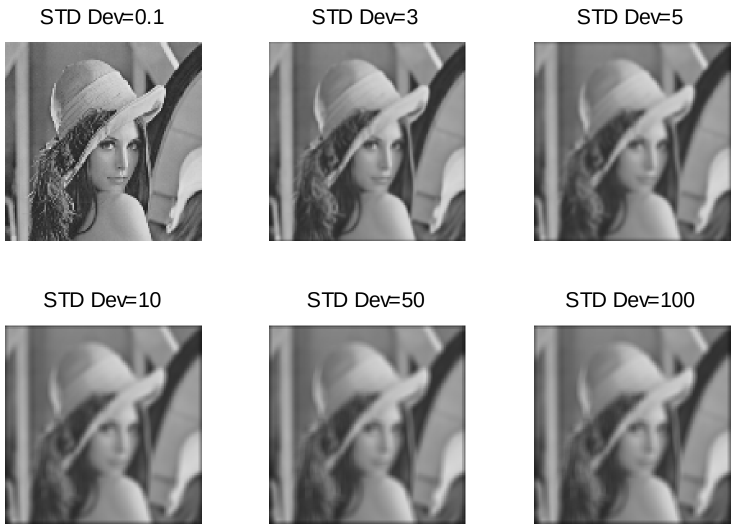


Fig. 25. Gaussian Blur

3.4.2 HIGH-PASS FILTERING

Unlike low-pass filters, which blur the image, high-pass filters do the opposite and act to sharpen the image. For example, the Laplacian filter is a 2nd derivative high-pass filter which highlights regions of rapid intensity change. This type of filter also works well in edge detection, but due to its high sensitivity to minor amounts of noise, the image must often be smoothed or blurred before detection is performed. Changing α changes the shape of the Laplacian filter. Several examples of the Laplacian filter are shown in Fig. 26.

The Laplacian filter can be written as follows:

$$\nabla^2 = \frac{4}{(\alpha + 1)} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix} \tag{1.13}$$

Fig. 26. Laplacian Filtering

3.5 CONTRAST ADJUSTMENT

3.5.1 CONTRAST STRETCHING

Contrast stretching involves modifying the contrast in an image by modifying the range of intensity values it contains. When darkening an image, the upper part of the histogram is mapped over the entire range of the output histogram. In the case of lightening an image, the lower portion of the histogram, representing the lighter portions of the image, are mapped over the entire range of the output histogram. To darken an image, the upper portion of the histogram is selected and remapped over the entire histogram range. To lighten an image, the lower portion of the histogram is selected and remapped over the entire histogram range. The smaller the range of the selected original histogram, the greater the difference in contrast. Fig. 27 shows contrast stretching which results in a darker image and Fig. 28 shows contrast stretching which results in a lighter image.

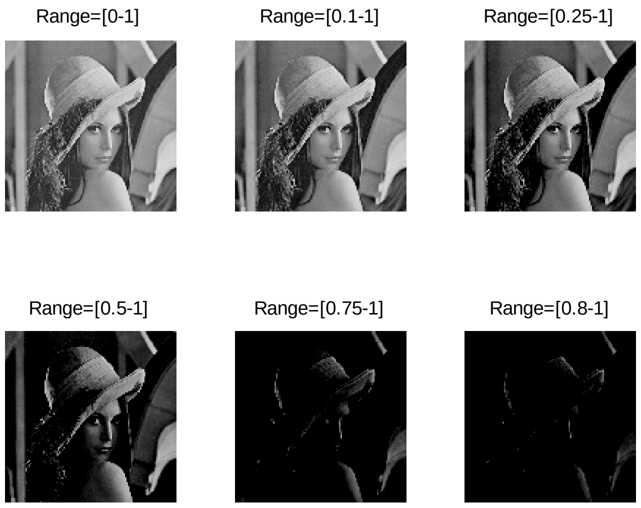


Fig. 27. Contrast Stretching - [0.5 1] -> [0 1]

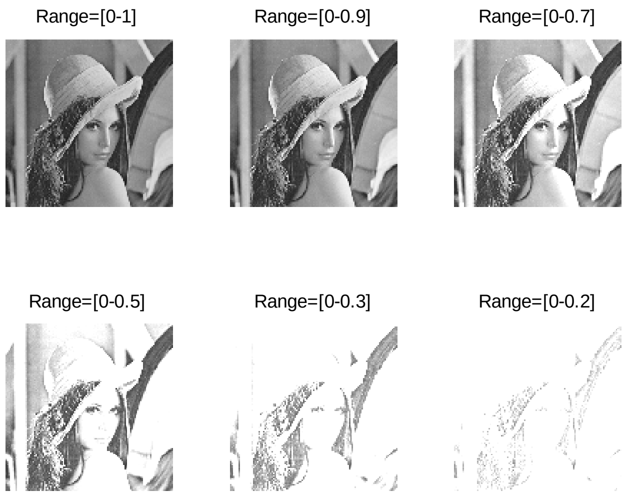


Fig. 28. Contrast Stretching - [0 0.5] -> [0 1]

3.6 **OTHER TRANSFORMATIONS**

3.6.1 **OIL-PAINT TRANSFORMATION**

The oil-painting transformation involves calculating the histogram for each pixel and its surrounding pixels and then assigning the most commonly occurring brightness value to that pixel [37]. Increasing the window size when determining brightness values results in larger "oil-droplet" sizes, giving the image a more distorted appearance. An example of the oil-painting transformation is shown in Fig. 29.

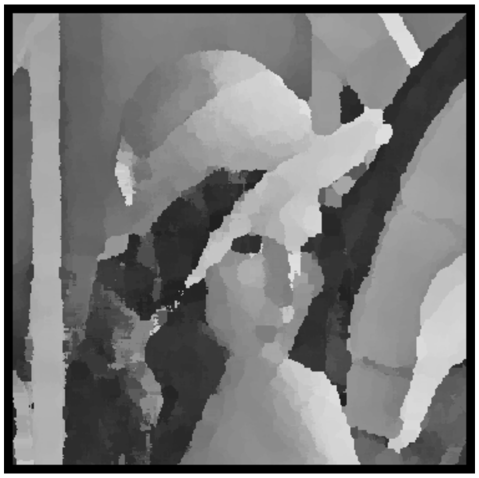


Fig. 29. Oil Painting Effect

Chapter 4

CAPTCHA SYSTEM DESIGN

4.1 GENERATION OF BACKGROUND

The first stage in building the CAPTCHA image is to generate a suitable background on which to place the face and non-face images. The background that was chosen was a random background of 75 to 125 pixel rectangles. This background was chosen in order to help deter computerized CAPTCHA detection. Each rectangle's color is a random grayscale value. The concept behind the random background is that the randomized rectangles help mask the location of the placed face and non-face images by preventing techniques such as edge detection. Additionally, further noise and/or patterns could be added to each rectangle in order to further enhance security. The generated background images are 500x300 pixels in size. An example of one of the sample background images is shown in Fig. 30.

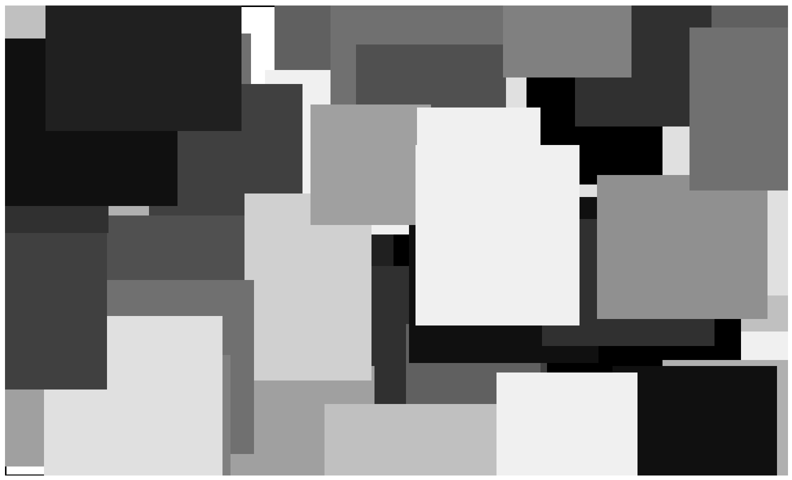


Fig. 30. CAPTCHA Background Image

4.2 IMAGE COLLECTION

4.2.1 OBTAINING THE UNMODIFIED FACE IMAGES

The images chosen for the face images are from the Carnegie Melon University frontal_images face database [38]. The database consists of 180 grayscale face images of varying sizes and rotations. From this dataset, images were chosen to be used in the CAPTCHAs. In addition to the CMU images, the image 'Lena' was also added to the test database for a total of 80 images. The images that were removed from the original dataset included multiple faces, contained a face that was too small or had a low resolution. The final 80 images were converted to a PNG format for consistency. An example of an image from the CMU face database is shown in Fig. 31.

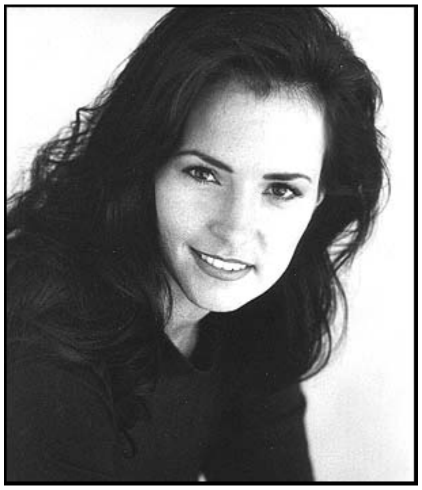


Fig. 31. Face Image from CMU Database

4.2.2 OBTAINING THE UNMODIFIED NON-FACE IMAGES

The non-face image database was generated by downloading a set of random images from the website Flickr. Flickr is a website where users may upload, view and organize their images and videos. The advantage of using Flickr to generate random test images is that the website provides an excellent search API for finding and downloading the images and because users are able to indicate whether or not they wish to distribute their work under the Creative Commons license.

Under specific Creative Common's licenses, users are allowed to redistribute the works with modifications, which was required as each non-face image was to be distorted before being added to it's respective CAPTCHA image [39]. The images were randomly generated using search keywords of animal species such as 'dog', 'frog' or 'horse'. From the random images downloaded, 65 were chosen to be used for the CAPTCHA database. Once selected, these images were converted to grayscale PNG files to allow for consistency with the face database. An example of one of the images from Flickr, which was used in the test database is shown in Fig. 32.

Fig. 32. Non-Face Image from Flickr

4.3 IMAGE DISTORTION

4.3.1 SELECTION OF DISTORTIONS

During the process of selecting distortions, 24 distortions were tested and compared. Of these distortions, 16 were used in the generation of the CAPTCHA images. Table 1 lists all of the distortions used during CAPTCHA generation. Table 2 lists the distortions that were tested, but eventually omitted due to similarities to other distortions or not having enough effect on the images.

Table 1. Distortions Used in CAPTCHAs

Distortions used in CAPTCHAs	
1. Barrel	2. Opening
3. Blur	4. Periodic Noise
5. Closing	6. Piecewise Scaling
7. Darkening	8. Resolution Modification
9. Dilation	10. Rotation
11. Erosion	12. Scaling x-coordinates
13. Laplacian Filtering	14. Scaling y-coordinates
15. Lightening	16. Speckle Noise

Table 2. Omitted Distortions

Omitted Distortions	Reason for Omission
Translation	Did not yield any desirable effect
X/Y Proportionate Scaling	Effect already performed when adding image to CAPTCHAs
Shearing	Until undesirable levels of shearing are reached the results are very similar to rotation
Pin Cushion	Did not provide enough distortion to center of image
Polynomial	Too many parameters to achieve quantifiable different amounts of distortion
Salt & Pepper Noise	Not sufficiently different than speckle noise
Gaussian Noise	Not sufficiently different than speckle noise
Oil Paint Distortion	Similar to morphologic operation results

4.3.2 DISTORTING THE FACE AND NON-FACE IMAGES

Before merging the database images with the randomly generated background images, each image from the database was modified using each of 16 different types of image distortion. The distortions used were: barrel, blur, closing, darken, dilation, erosion, Laplacian filtering, lightening, opening, periodic noise, piecewise scaling, resolution modification, rotation, scaling the x coordinates, scaling the y coordinates, and speckle noise. For the chosen distortions, three levels of severity for each distortion were chosen. These levels of severity are labeled as 1, 2 and 3 in the database, with 1 being a small level of distortion, 2 being a medium level of distortion and 3 being a severe level of distortion. The settings for each level were initially chosen randomly, and then adjusted as determined necessary.

Fig. 33 and Fig. 34 show distorted face and non-face images with level 2 speckle distortion. Table 3 shows the distortions and parameters used during distortion.

Table 3. Parameters of Distortions

Distortion	Variable Adjusted	1	2	3
Barrel Distortion	Amplitude of Cubic Term	0.00008	0.0002	.0005
Blur	Std Dev	4	8	20
Closing	Radius	3	5	8

Darkening	Histogram Min-range	0.5	0.6	0.7
Dilation	Radius	3	5	8
Erosion	Radius	3	5	8
Laplacian Filtering	A	20	10	5
Lightening	Histogram Max-range	0.6	0.5	0.3
Opening	Radius	3	5	8
Periodic Noise	% of image removed	2/3	4/5	6/7
Piecewise Scaling	Scale Factor	2:1	3:1	6:1
Resolution Resizing	Scale Factor	1:4	1:8	1:10
Rotation	Degrees Rotated	45	90	180
Scaling X	Scale Factor	4	5	6
Scaling Y	Scale Factor	2.5	3	4
Speckle Noise	Variance	0.2	1	3

Fig. 33. Face image Modified with Speckle Noise

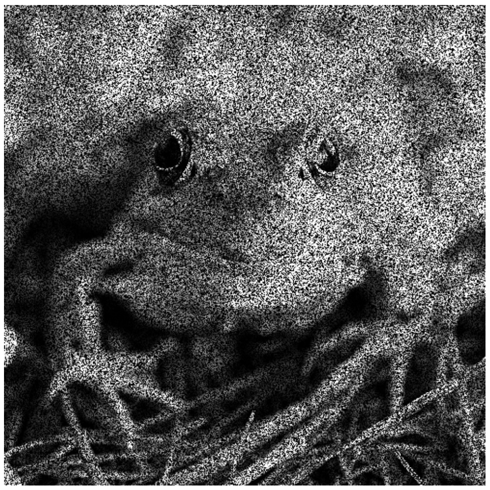


Fig. 34. Non-Face Image Modified with Speckle Noise

4.4 LOCATING FACES

Once the original and modified face databases were generated, a new database was created to store the bounding coordinates of the face images. These coordinates were determined manually and represented the area within the face image that contained the actual face. The coordinates that were saved were the upper-left x and y values and the lower-right x and y values. In the finalized CAPTCHA, the user is required to click within these coordinates in order to correctly solve the CATPCHA. Clicking on the image itself, but outside of these coordinates represents a failure to successfully solve the CAPTCHA. The first step in the generation of the database involved obtaining the coordinates for the original face images. Once these coordinates were determined, the modified face coordinate database was generated by directly copying the original coordinates for all modifications that did not involve the modification of the pixel locations. In the cases of the barrel, rotation, piecewise scaling, and x and y scaling modifications, the located face coordinates were transformed accordingly to match the appropriate distortion.

Fig. 35 shows an example image from the test database with a bounding box drawn to represent the location of the actual face on the face image.

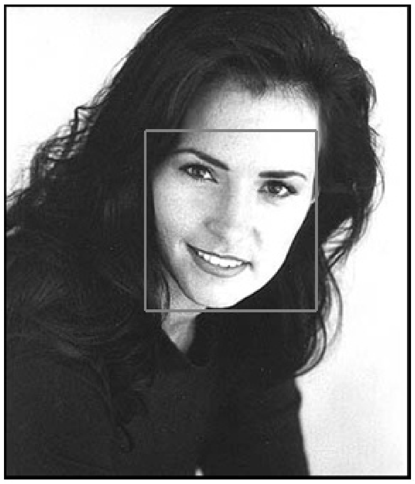


Fig. 35. Example Image With Bounding Coordinates Drawn Around Face

4.5 Generating CAPTCHAs

After the distorted face and non-face image databases were generated, these images were used to generate the final CAPTCHA images. The CAPTCHA images were created by taking the background images and overlaying several distorted images. Before being placed on top of the background, the face and non-face images were resized so that neither the length nor the width exceeded a randomly generated value between 75 and 125 pixels. A total of four to five total images were placed on each background image. Of these four to five images, one to the total number of images minus 1 were face images, with the remaining being non-face images. Two base CAPTCHAs for each combination of face/non-face images was generated resulting in a total of 14 base CAPTCHA images during testing. From these 14 base CAPTCHAs, one variance for each of the 16 distortions and 3 levels of distortion was generated resulting in 672 distorted CAPTCHAs.

An example of a completed CAPTCHA image can be found in Fig. 36. Along with the generated CAPTCHAs, a log was generated which contained the data shown in

Table 4. The table contains data regarding which images were stored in a particular CAPTCHA and the relative face coordinates of each placed face image. This log was used to generate a MySQL database, which was used in determining whether or not the user clicked in the correct location(s) within the CAPTCHA image.

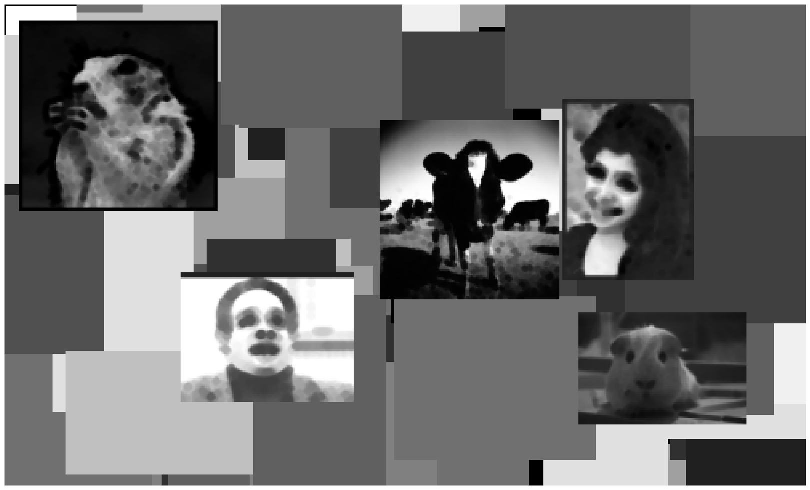


Fig. 36. Example Completed CAPTCHA Image

Table 4. Generated CAPTCHA Log

File Name	Distortion Type	Distortion Level	Height	Width	Total Images	Face Images	Faces Embedded	Non-faces Embedded	Face Coordinates	Placed Image Coords	Face Size	Non-face Size

4.6 SERVER & DATABASE

Two primary databases are used for determining success and statistics regarding success rates for the implemented CAPTCHAs. The generated CAPTCHA database, which was created from the previously mentioned log file, contains information regarding which images are located in which CAPTCHAs, the level of distortion for the particular CAPTCHA, the size of the CAPTCHA image and the locations of the face coordinates. The fields from this database are shown in Table 5. When a CAPTCHA attempt is submitted on the website, the clicked face coordinates are verified against the actual face coordinates stored in the generated CAPTCHA database. If the clicked coordinates fall within those coordinates stored in the face coordinates field, then the CAPTCHA is considered to be successfully solved.

The second database, the CAPTCHA log, is used primarily in determining statistics for the CAPTCHAs including storing success rates, user attempts, time taken to solve and clicked coordinates. Whenever a CAPTCHA is attempted, information regarding that attempt is written to the database, which is shown in Table 6. This database stored all CAPTCHA results solved by humans and was the basis for all of statistical analysis of the human-based success rates and time taken to solve.

Table 5. Generated CAPTCHA Database Fields

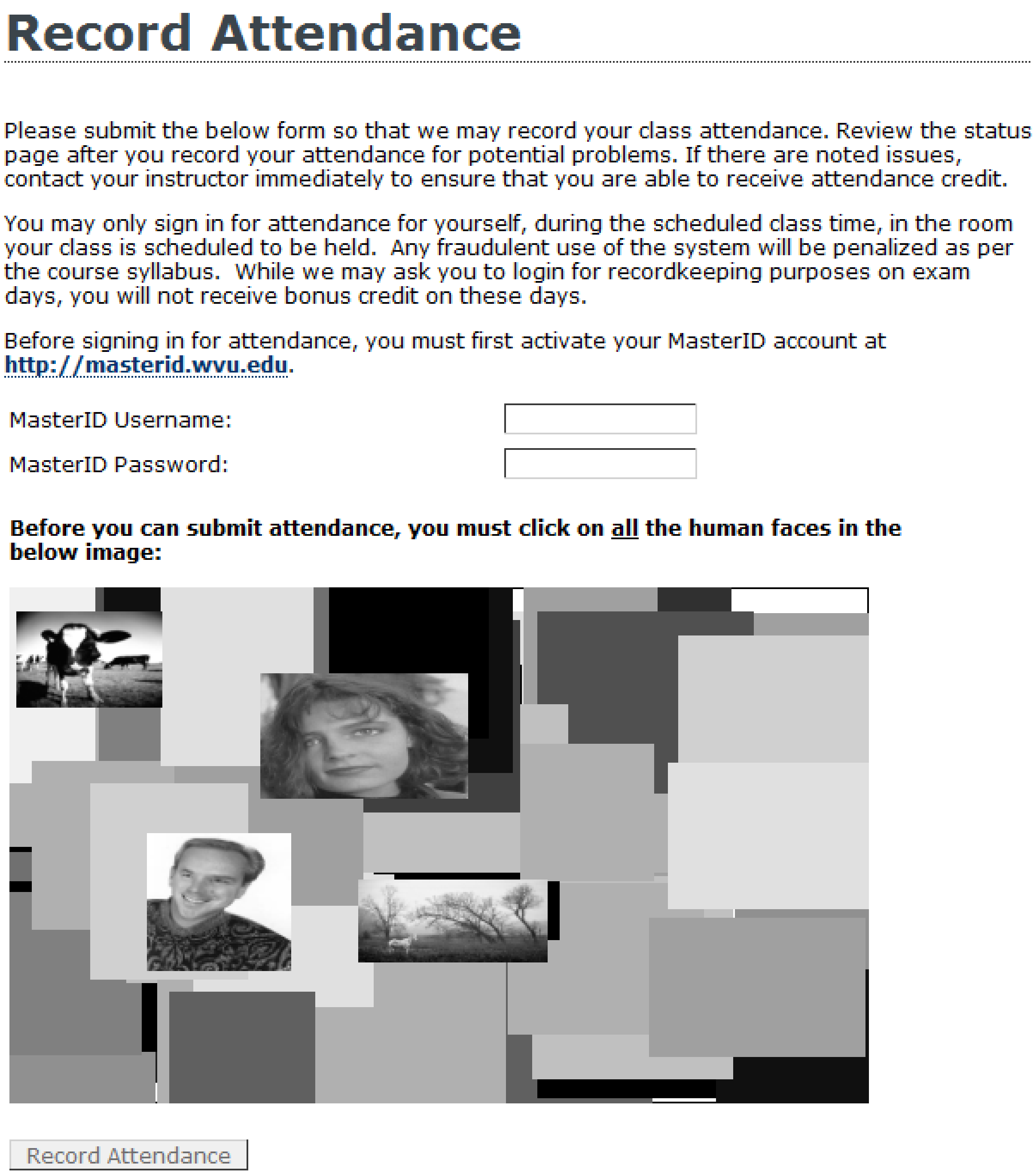
id	file	distortion type	distortion amount	count total images	count face images	image height	image width	face coords	original face images	original other images

Table 6. CAPTCHA Log Database Fields

id	captchaid	attempt	success	ip	username	generatedtime	firstclicktime	submittime	coords

4.7 USER INTERFACE

Due to the large number of generated CAPTCHA images, in order to determine reliable statistics regarding the success rates of the CAPTCHAs, each CAPTCHA image needed to be tested by a large number of users and needed to be tested multiple times. In order to achieve this, the CAPTCHA was implemented on the WVU Computer Science 101 website, which is used by hundreds of students on a daily basis for the purpose of recording attendance. In order to record attendance, the students must login to the website using their WVU MasterID username and password. The CAPTCHA was added to this login page and was set up so that students were required to solve the CAPTCHA each time they wished to record their attendance. In order to successfully solve the CAPTCHA, the user must click on each of the faces located within the CAPTCHA image. In the event that the CAPTCHA is solved incorrectly, an error message is displayed and the user is required to once again fill out the login information as well as solve a different CAPTCHA image. A screen capture of the CS101 attendance page, along with the implemented CAPTCHA is shown in Fig. 37.



Record Attendance

Please submit the below form so that we may record your class attendance. Review the status page after you record your attendance for potential problems. If there are noted issues, contact your instructor immediately to ensure that you are able to receive attendance credit.

You may only sign in for attendance for yourself, during the scheduled class time, in the room your class is scheduled to be held. Any fraudulent use of the system will be penalized as per the course syllabus. While we may ask you to login for recordkeeping purposes on exam days, you will not receive bonus credit on these days.

Before signing in for attendance, you must first activate your MasterID account at http://masterid.wvu.edu.

MasterID Username:	
MasterID Password:	

Before you can submit attendance, you must click on all the human faces in the below image:

Fig. 37. Screen-Capture of CAPTCHA Implemented on CS101 Website

Chapter 5

RESULTS

5.1 IMAGE QUALITY METRICS

In order to determine the severity of the distortions applied to the images from an analytical perspective, nine image quality metrics were used to determine the variance in quality between each of the undistorted CAPTCHAs and the distorted variants of those CAPTCHAs.

For the 16 distortion types, each distorted image of a particular type was compared with its corresponding original image. The 3 levels of distortion for each distortion type, 14 base CAPTCHA images, and 16 tested distortions, resulted in 672 comparisons for each of the 10 metrics. The tests were performed using MATLAB along with the the Metrix MuX package [40], which is a collection of image quality metric algorithms combined under a common interface along with run_metrics.m, which is wrapper code to perform all of the Metrix Mux operations. The results of each test are categorized by distortion type, degree of distortion and metric and are shown in Table 7. The numbers in the table represent an average of 14 tests run for each level of distortion and distortion type.

Table 7. Metrics for Distortions

Distortion	Level	IFC	NQM	PSNR	MSE	SNR	SSIM	UQI	VIF	VIFP	VSNR
barrel	1	12.74666	4.421135	14.63399	2372.972	9.563371	0.762702	0.729507	0.332562	0.449834	7.954591
barrel	2	12.74606	4.57092	14.73617	2319.893	9.665551	0.764661	0.73215	0.332376	0.449153	8.044797
barrel	3	12.73014	4.565882	14.71801	2326.855	9.647391	0.76025	0.727125	0.328839	0.446473	7.973947
blur	1	14.90893	28.78776	30.987	61.30213	25.91638	0.960041	0.951726	0.683272	0.735641	29.67261
blur	2	14.40342	25.54581	28.63166	102.7533	23.56104	0.935562	0.926543	0.607477	0.688435	25.8292
blur	3	14.23992	24.42416	27.87122	121.7605	22.8006	0.925375	0.916256	0.581041	0.672903	24.66467
closing	1	15.66718	22.31213	29.85903	85.34334	24.78841	0.9741	0.966577	0.736364	0.802425	24.13617
closing	2	15.20528	19.8976	27.61969	136.6949	22.54907	0.961271	0.951544	0.677841	0.763948	21.03724
closing	3	14.85883	17.7647	25.63734	208.5782	20.56672	0.945173	0.933281	0.623006	0.725889	18.38579
darken	1	14.23332	9.164081	18.12963	1069.715	13.05901	0.845899	0.847259	0.573693	0.67275	16.37755
darken	2	13.9576	7.786826	16.75153	1458.283	11.68091	0.822612	0.823918	0.525816	0.63419	15.62855
darken	3	13.72664	6.497948	15.47522	1933.233	10.4046	0.800266	0.80161	0.483804	0.599632	15.17146
dilation	1	14.30347	15.42151	23.20019	353.4115	18.12957	0.927341	0.917725	0.572924	0.675087	15.88895
dilation	2	13.97663	13.24104	21.26045	540.7473	16.18983	0.900716	0.889824	0.517878	0.635663	13.49241
dilation	3	13.72832	11.3808	19.6887	764.6583	14.61808	0.875071	0.862181	0.474869	0.605232	11.65753
erosion	1	14.38	15.97381	23.38371	338.202	18.31309	0.930347	0.918211	0.579915	0.680666	22.29343
erosion	2	14.03676	13.58629	21.1752	554.1346	16.10458	0.899488	0.886212	0.523686	0.639361	20.27136
erosion	3	13.78842	11.62822	19.50565	806.0411	14.43503	0.871304	0.856706	0.48091	0.608597	18.78116
laplacian	1	13.09356	2.866438	12.05816	4420.367	6.987536	0.732072	0.729765	0.37396	0.506137	14.76553
laplacian	2	13.07089	3.357798	12.55426	4008.336	7.48364	0.732053	0.725416	0.372589	0.505481	14.81895
laplacian	3	13.02613	4.06011	13.29507	3457.31	8.224452	0.726631	0.715579	0.368457	0.50119	14.44657
lighten	1	14.83242	12.18835	20.03647	675.5618	14.96585	0.931098	0.908805	0.659593	0.735759	11.64302
lighten	2	14.41462	10.05768	18.28019	1012.554	13.20957	0.90735	0.882312	0.599512	0.686351	9.411796
lighten	3	13.79511	6.378439	15.24904	2028.818	10.17842	0.855577	0.826124	0.498266	0.601017	6.345327
opening	1	15.75677	22.94271	30.80787	68.88375	25.73725	0.976099	0.964062	0.750299	0.817193	29.64863
opening	2	15.26959	20.41094	28.23205	119.8743	23.16143	0.963106	0.948828	0.69107	0.778301	26.76061
opening	3	14.84279	18.13706	25.97536	200.5996	20.90474	0.946942	0.930361	0.634181	0.738099	24.35814
periodic	1	14.53007	7.126341	14.91652	2251.388	9.845903	0.816076	0.801747	0.557411	0.61312	15.28423
periodic	2	13.98096	4.955597	13.32222	3255.221	8.251596	0.768245	0.758513	0.482031	0.559188	14.86991

periodic	3	13.78058	4.117636	12.71862	3740.456	7.648001	0.75081	0.745004	0.452622	0.535669	14.73752
piecewise	1	12.71513	5.317149	15.73597	1846.75	10.66535	0.771795	0.739838	0.377081	0.503221	9.44514
piecewise	2	12.39884	4.226259	14.70946	2332.315	9.638839	0.750003	0.716722	0.338654	0.468826	8.375987
piecewise	3	12.46997	3.794287	14.11109	2683.203	9.040472	0.74896	0.716046	0.330219	0.456914	7.871656
resolution	1	16.75761	32.05313	34.35378	37.83564	29.28316	0.983952	0.980145	0.831798	0.83714	36.40972
resolution	2	15.29323	26.45745	29.38656	94.5395	24.31594	0.954572	0.948757	0.68804	0.739939	27.69022
resolution	3	15.06325	26.38778	28.68442	107.9059	23.6138	0.943893	0.937607	0.663079	0.721109	26.31165
rotate	1	12.07874	3.78331	14.38876	2496.746	9.318139	0.727952	0.693271	0.311114	0.427348	7.724963
rotate	2	11.8164	3.277151	13.94964	2760.486	8.879017	0.721544	0.685528	0.319231	0.440838	7.458746
rotate	3	13.02173	4.3657	14.98736	2157.746	9.916736	0.772125	0.748791	0.367393	0.506312	8.390551
scale_x	1	12.55147	3.855579	14.14066	2667.542	9.07004	0.752726	0.719422	0.334101	0.459668	7.909268
scale_x	2	12.55746	3.804976	14.03971	2729.707	8.969092	0.752513	0.718697	0.331255	0.456254	7.797343
scale_x	3	12.56008	3.80353	14.00219	2756.203	8.931569	0.75313	0.718837	0.329736	0.454049	7.752814
scale_y	1	12.51568	4.150942	14.57695	2409.597	9.506328	0.753984	0.721899	0.341221	0.469421	7.931818
scale_y	2	12.50991	4.104293	14.44561	2485.128	9.374991	0.754101	0.72218	0.337965	0.465719	7.820834
scale_y	3	12.51819	4.073539	14.29416	2578.313	9.223545	0.75388	0.721674	0.333652	0.461185	7.704628
speckle	1	15.10733	24.03985	28.8201	105.8226	23.74948	0.941226	0.937499	0.701264	0.739581	28.99561
speckle	2	14.18007	17.20641	22.79204	408.7769	17.72142	0.887974	0.886511	0.567123	0.639346	22.11757
speckle	3	13.76343	14.58099	21.02494	598.7517	15.95432	0.857178	0.85516	0.504789	0.59367	19.55071

Once all the distortion data was compiled, the metrics were placed into one of the following categories of image quality metrics: statistical, high-level HVS and low-level HVS. The statistical methods include the MSE, SNR, PSNR. The low-level HVS methods include the NQM and VSNR. The high-level HVS methods include the IFC, UQI, SSIM, VIF and VIFP. Once the metrics were sorted, the correlation between each of the metrics within each category was taken in order to determine similarity between metrics and reduce the amount of overall data to be examined. The correlations are shown in Table 8, Table 9 and Table 10.

Table 8. Correlation of Statistical Metrics

	MSE	**SNR**	**PSNR**
MSE		0.919315	0.919315
SNR	0.919315		1
PSNR	0.919315	1	

Table 9. Correlation of Low-Level HVS Metrics

	NQM	VSNR
NQM		0.920046
VSNR	0.920046	

Table 10. Correlation of High-level HVS Metrics

	IFC	**SSIM**	**UQI**	**VIF**	**VIFP**
IFC		0.921187	0.941969	0.988929	0.977565
SSIM	0.921187		0.994294	0.949712	0.960603
UQI	0.941969	0.994294		0.965707	0.97526
VIF	0.988929	0.949712	0.965707		0.991932
VIFP	0.977565	0.960603	0.97526	0.991932	

For the statistically based metrics, the MSE, SNR and PSNR showed a very strong correlation and also showed identical data for the ranking of severity of each distortion which reduces the necessity to present further data for all three metrics. Of the statistical based methods, the PSNR was selected for further analysis. For the low-level HVS based methods, the NQM and showed a strong correlation, but upon further analysis of the data, there were many differences in the ordering of severity of different distortion types. Because of these differences and the knowledge that the NQM and VSNR are based on quite different methodologies, both the NQM and VSNR were chosen for further analysis. There were a number of strong correlations found within the high-level HVS metrics. Of the VIF, VIFP and IFC, only the VIF was chosen for further analysis due to the VIFP being simply a computationally simpler multi-scale version of VIF and the IFC being the basis of the work for the improved VIF. The strong correlation between the SSIM and UQI is due to the SSIM being a general form of the UQI and because of this, only the SSIM was chosen for further analysis.

5.1.1 **STATISTICAL METRIC ANALYSIS**

The PSNR showed results that were as expected for a statistical based analysis. These results are shown in Fig. 38. Distortion methods which did not involve the moving of pixel locations such as resolution reduction and blurring yielded the greatest PSNR, while those that involved moving pixel locations such as the barrel distortion, rotation and scaling yielded some of the lowest PSNRs. Although the periodic function does not involve the moving of pixels, it also yielded a very low PSNR; this is most likely due to the large amount of data removed during the distortion. The Laplacian filtering performed the worst due to the operation removing much of the image data and only leaving the image's edges.

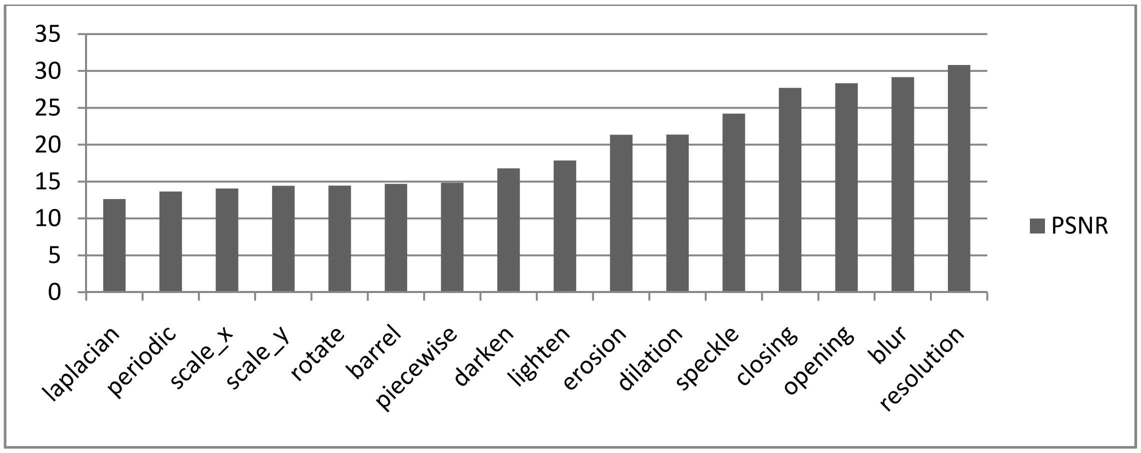


Fig. 38. PSNR Metrics

5.1.2 Low-Level HVS Metric Analysis

The low-level HVS metrics yielded similar results to the PSNR, especially in the case of the NQM. The NQM ranked the traditional distortions among the highest and ranked the pixel-location based distortions among the worst. Its results are quite similar to the PSNR's results due to the NQM metric's reliance on a weighted signal-to-noise ratio. The results for the NQM are shown in Fig. 39. The VSNR's dependence on geometric pixel locations is shown to be even greater than that of the PSNR and NQM. The 5 distortions that involved modifying pixel locations ranked the worst on the VSNR scale. The results for the VSNR are shown in Fig. 40.

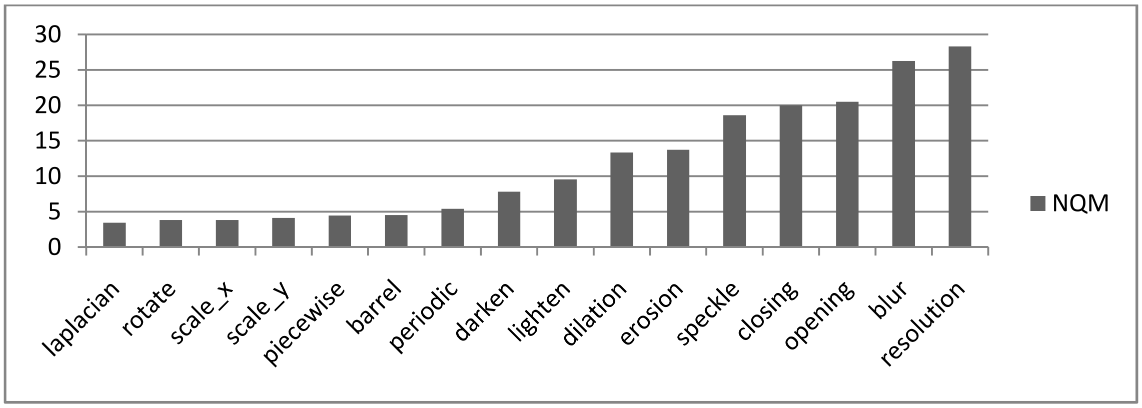


Fig. 39. NQM Metrics

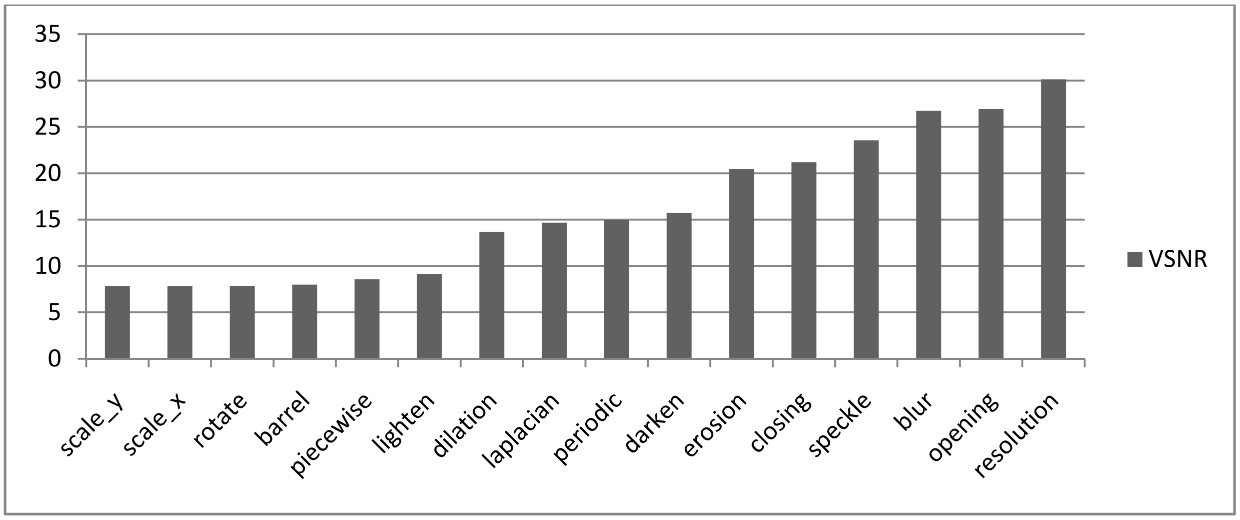


Fig. 40. VSNR METRICS

5.1.3 **HIGH-LEVEL HVS ANALYSIS**

The SSIM metric yielded very strong results across all distortion types, with geometric distortions yielding a low correlation between original and modified CAPTCHAs. Like the NQM and PSNR, the Laplacian filtering yielded the lowest overall correlation. In general, the results correspond with the data from [27], which shows the SSIM yielding better results for intensity distortions than for spatial shifting. The most interesting point to note about the SSIM data is that on a 0 to 1.0 scale, the lowest similarity index was 0.7303, which still indicates a strong similarity between original and modified images. The SSIM appears to take the unchanged background data into account more than the other metrics The results for the SSIM are shown in Fig. 41.

The VIF, which has a strong reliance on blur and additive noise yielded similar results to the other metrics with the spatial distortions all yielding values in the 0.40 to 0.50 range. The highest ranking of the results was the resolution distortion, which still only yielded a VIF of 0.7276 on a 0 to 1.0 scale. The results for the VIF are shown in Fig. 42.

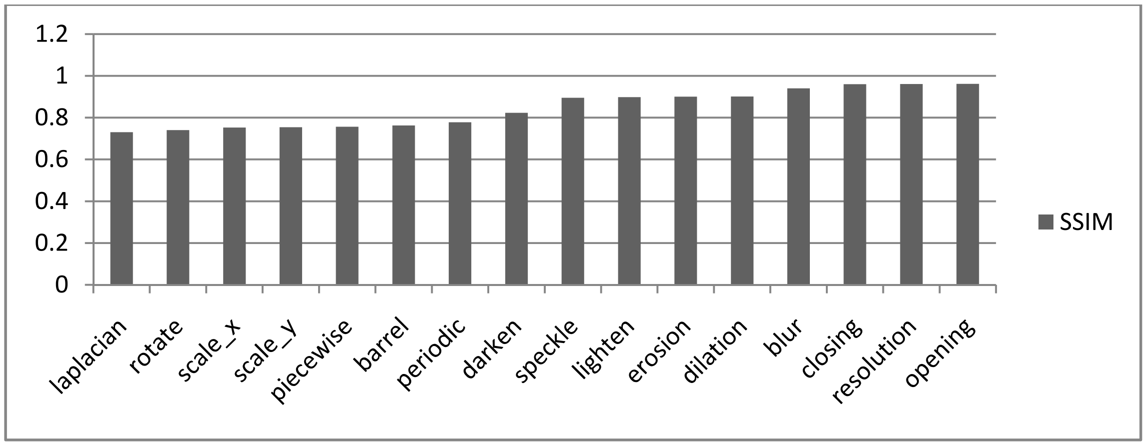


Fig. 41. SSIM Metrics

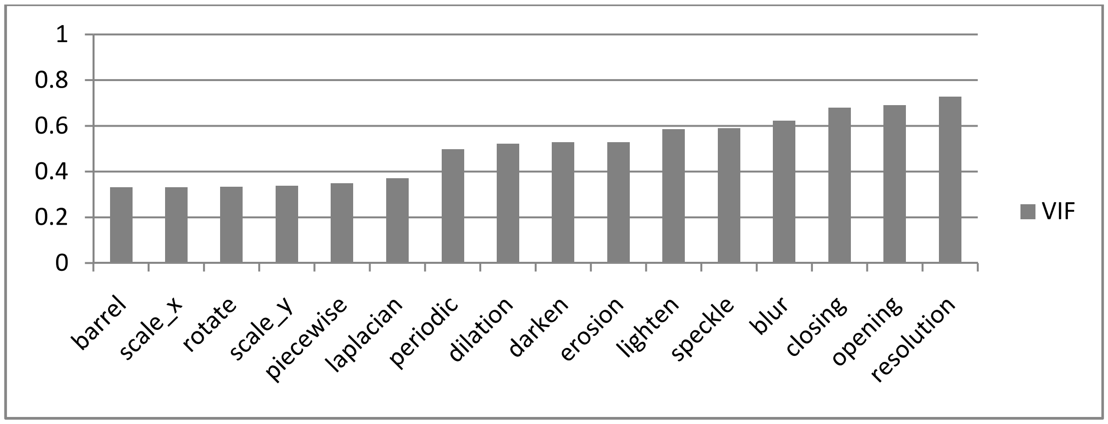


Fig. 42. VIF METRICS

5.1.4 Conclusion of Metric Analysis

Overall, the metrics showed that geometric distortions yield the least detectable similarity between original and modified images. This is due to the metrics generally being spatially dependent algorithms. The Laplacian filtering also ranked amongst the worst across all metrics due to the amount of data lost during the filtering. The intensity, noise, erosion and dilation modifications yielded quite variable results across the tests, but all ranked among the middle of the results. Distortions which did little to modify intensity values or pixel locations, such as the resolution reduction, blurring, the opening operation and the closing operation yield the greatest detectable similarity across all tests. Overall, the SSIM showed the best performance by yielding high similarity rankings across all tests.

5.2 HUMAN RESULTS

The human results consist of 7897 attempts from students using the WVU CS101 attendance website. The users of the attendance website are undergraduate students from all majors. The overall success rate of the experiment was 71.71% across all distortion types and distortion levels. The data regarding success rates was also broken down into more specific categories in order to show the correlation between distortion type and distortion amount.

The data that was gathered shows that, while increasing the distortion to the images does decrease the success rate, that a light to medium level of distortion can be applied without a significant reduction. Increasing the distortion from level 0 (no distortion) to level 1 decreased the overall success rate by only 2.88%. The overall success rate of the level 3 distortion at 65.46% shows that this level of distortion is too high for general use. The combined computer/human success rates section will further discuss the best level of distortion for yielding the best human results and the worst computer results. **Error! Reference source not found.**Table 11 shows the success rate based on the amount of distortion applied.

Table 11. Human Success Rate by Distortion Amount

Distortion Amount	Success Rate
0	80.00%
1	77.12%
2	71.96%
3	65.46%

The next step in analyzing the data was to determine which distortion types had the least effect on the success rates. The results showed that a number of the distortions had little effect on the success rate. The most successful of the distortion types was the resolution reduction; six other distortions also had a very high success rate and showed only a 3% reduction in success rate from the unmodified images. There was very little correlation found between categories of distortions and the success rate. For example, the three modifications that involved scale distortion, which were piecewise, scale_x, and scale_y, had vastly different success rates, with the scale_y operation performing quite well, the piecewise operation showing an average success and the scale_x performing the worst among all distortions. Table 12 shows the success rates based on distortion type.

Table 12. Human Success Rates by Distortion Type

Distortion Type	Success Rate
original	80.00%
resolution	79.10%
blur	78.37%
scale_y	78.26%
lighten	78.11%
rotate	77.32%
opening	77.25%
periodic	73.06%
closing	73.00%
piecewise	68.94%
dilation	68.79%
darken	67.92%
speckle	66.87%
erosion	65.29%
barrel	65.19%
laplacian	63.35%
scale_x	61.52%

Some of the most important data for further research involves the analysis of the results based on both distortion type and amount. The gathered data allows us to determine which distortions should be applied more or less heavily in future testing. These complete results are shown in Fig. 43. The lack of results for rotation distortion level 1 is due to an error in coordinate calculation. The remaining levels of distortion gave results that were the opposite of their assigned degree of distortion. This is likely due to our subjective assignment of distortion levels. Similarly, our ordering of the degree of distortion on the Laplacian filtering operation also shows opposite of the actual results. With the Laplacian filtering, as our distortion level increases, the amount of edges shown increases. It was assumed that with the higher number of edges that too much detail would be shown and that the face would be difficult to detect, although the results show the opposite effect. Another interesting result is that some operations actually increased the overall success rate. This result occurred on the following level 1 distortions: blur, closing, lighten, opening, piecewise and scale_y. It also occurred on the level 2 resolution and scale_y operations. While some of these results may be due to slight inaccuracies with a limited dataset, it is also possible that in some cases that the distortions actually made the face more easily detectable. For instance, the lightening operation based on histogram adjustment places greater emphasis on the highlights of the image and very well could make the face stand out more than in the original image.

The distortions that will most likely be used in further testing are those that achieved the highest success rates under high levels of distortion. These include the blur, lightening, opening, resolution and scale_y operations. These objectively ranked distortions all showed a reduction in success rate of less than 10% from the unmodified images.

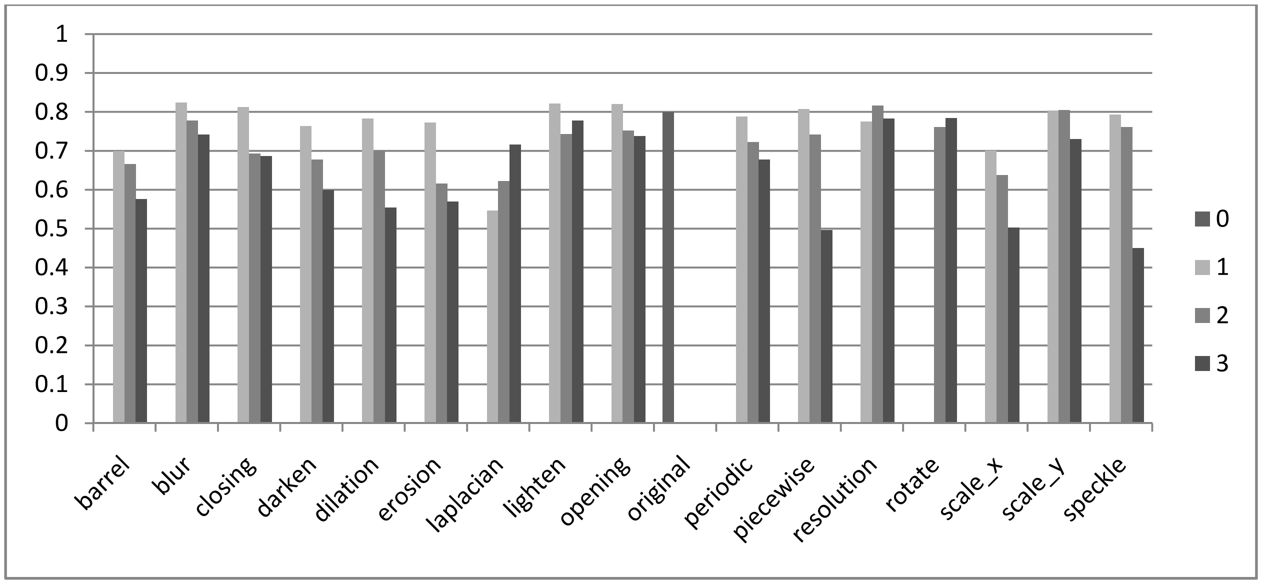


Fig. 43. Human Success Rates By Distortion and Type

5.3 COMPUTER RESULTS

One thing that is for certain is that CAPTCHAs will be subjected to automated attacks. In this section we will analyze simple attacks involving random guessing as well as advanced attacks involving the use of face detection algorithms.

One simple attack that could be performed against clickable CAPTCHAs is the random guessing of image locations. In the worst case scenario where an attacker is actually able to determine all of the image locations, but not determine if a human face exists, the attacker must still guess the number of faces involved in each CAPTCHA. The following equation can be used to determine the probability of guessing the correct number of faces:

$$4\ Total\ Images\colon \left[\binom{4}{3}+\binom{4}{2}+\binom{4}{1}\right]^{-1} = 11.11\% \qquad (5.1)$$

$$5\ Total\ Images: \left[\binom{5}{4}+\binom{5}{3}+\binom{5}{2}+\binom{5}{1}\right]^{-1} = 3.33\% \qquad (5.2)$$

These percentages can also be reduced much further based on the fact that even if the correct faces are chosen that the location of the face within the face image must then be selected.

In the case of an attacker who is using completely random clicking without the ability to determine the location of any of the face and non-face images, the attacker must first guess the number of times to click (1 through 4) and then must randomly choose locations. Each of the generated CAPTCHAs is a 500x300 size image, which results in 150,000 possible click locations. The placed face images are sized from 75 to 125 pixels on their longest side, which results in an approximate average size of 100x100 pixels. Given these approximations, the probability of clicking on the correct image(s) is given as follows:

$$\sum_{i=1}^{4} \frac{(100)(100)i}{(500)(300)} = 0.05\% \qquad (5.3)$$

Again, the above formula only gives the probability of locating the correct images and does not factor in finding the location of the face within the actual face image. The given approximations do indicate the difficulty of executing an attack based on the concept of randomly guessing the locations of the face images.

In an attempt to prevent more complicated attacks, the face detection CATPCHA was also tested against a popular face detection algorithm using various sets of training data. The algorithm that was used is the Viola-Jones algorithm for Robust Real-time Object Detection [41]. Three training data sets were used along with the algorithm to determine the overall accuracy of the face detection. The training data sets used are haarcascade_frontalface_alt2.xml, haarcascade_frontalface_alt.xml and haarcascade_frontalface_default.xml, which are provided in

[42]. In order to determine if the algorithm successfully solved a CAPTCHA, the following criteria must be met: the coordinates for the face determined by the algorithm must lie completely within the stored actual face coordinates assigned to a particular CAPTCHA, all faces in a CAPTCHA must be found, and the algorithm should not provide any false detections.

After running the algorithms on all 686 images, the three training data sets were compared against one another to determine if the different data sets provided similar results in overall accuracy. It was also determined that the three training data sets did not provide the same results, but instead had a variety of detection rates for particular images and distortion types. The overall success rates based on the training data set used are shown in Table 13.

Table 13. Success Rate By Training Data

Training Data	Success Rate
Alt	11.66%
Alt2	10.50%
Default	11.22%

The next step in the analysis of the training data involved determining if there was a connection between the amount of distortion applied to an image and the overall success rate. The data showed that with an increase in overall distortion comes an inverse relationship to the overall success rate of the computer detection algorithm. This proves that there is a tangible benefit to increasing the distortion, although the distortion must be kept within the limits of high human success rates. The success rates based on distortion type are shown in Table 14.

Table 14. Success Rate By Distortion Amount

Distortion Amount	Success Rate
0	16.67%
1	13.39%
2	11.01%
3	8.63%

Data regarding the success rate based on the type of distortion was gathered in order to determine which of the distortion types is the most useful in stopping computer based attacks. The results show that the computer-based detection does not work well in methods involving scaling, rotation and Laplacian filtering. An interesting conclusion regarding the data is that certain types of distortion seemed to have a decrease in success rate with increased levels of distortion while others were relatively unaffected. For instance, the barrel distortion shows a decrease in success rate of 9.53% when going from distortion level 1 to 2 and another 4.75% decrease when moving from distortion level 2 to 3. The opening distortion on the other hand had the exact same success rate from distortion level 1 to 2 and even increased slightly going from level 2 to 3. Another interesting point to note is that some of the distortions actually had a positive effect on the overall success rate. In most of these cases the improvements were found using level 1 and level 2 distortions. This improvement is likely in part due to the distortions adjusting the contrast and clarity levels of the picture to better match those used in training set. Overall this data provides valuable information regarding which distortion types and levels are necessary in order to prevent computer-based face detection. Fig. 44 shows the average success rate for all CAPTCHAs based on the distortion type used and level of distortion.

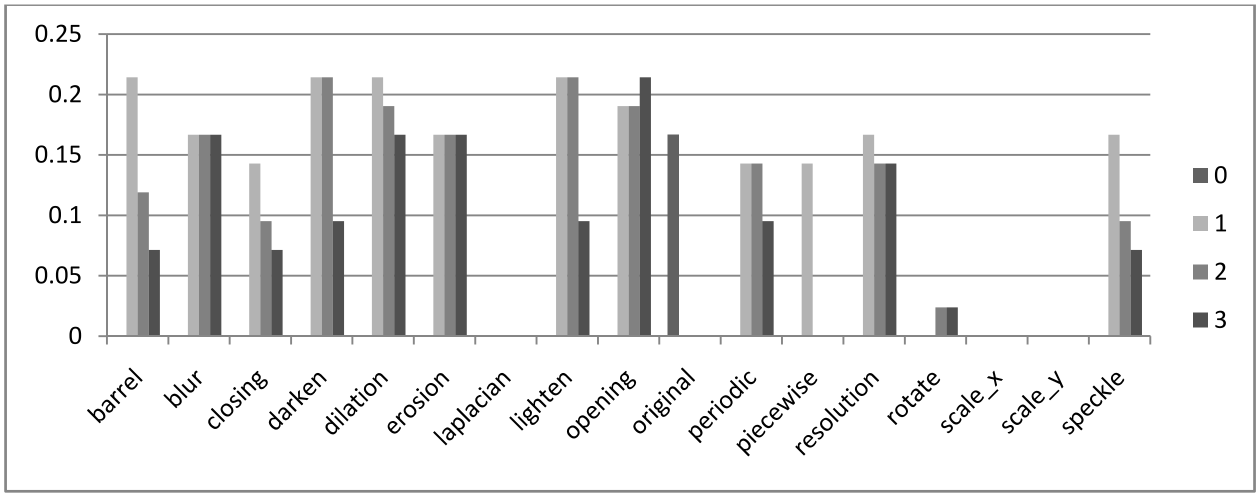


Fig. 44. Success Rate by Distortion and Level

In addition to the success rates for the computerized algorithms, data was also gathered for the time taken to solve a CAPTCHA. CAPTCHAs can be set to time out after a particular amount of time has expired on a page in order to prevent computer algorithms from spending large amounts of computation time to attack a CAPTCHA. Unfortunately, the data gathered shows that the Viola-Jones algorithm works extremely quickly in solving CAPTCHAs and in every test the algorithm solved the CAPTCHA in under 1 second. The average time taken to solve for each training data set is shown in Table 15.

Table 15. Time Taken To Solve CAPTCHAs

Training Data	Time (ms)
alt	299.98
alt2	161.04
default	285.66

5.4 OVERALL RESULTS

5.4.1 COMPARISON OF RESULTS

The ultimate goal of a successful CAPTCHA is to allow human access while denying the ability to automate the defeat of the CAPTCHA. Our overall success rate across all distortion types and all distortion levels is 71.71% human success rate versus an 11.13% computer success rate. This success rate is compared against several other popular modern CAPTCHAS in Table 16. While the numbers seem to be slightly lower than some other modern CAPTCHAs it is important to note that these results are only averages from the first run of testing a wide variety of distortion types. After removing the unsuccessful distortion types, the human success rates will increase and the computer success rates will decrease; the purpose of discussing the overall average success is to prove the viability of a face-detection CAPTCHA.

Table 16. Comparison of Human and Computer Success Rates

CAPTCHA	Type	Human Success	Computer Success
Microsoft	Text-based	90% [5]	60% [43]
Baffletext	Text-based	89% [44]	25% [44]
Handwritten	Text-based	76% [45]	13% [45]
Kleuver Video	Video-based	90% [15]	13% [15]
ASIRRA	Image-based	99% [12]	10% [14]
Day Face-Detection	Image-based	72%	11%

In order to achieve the desired goal, it is necessary to choose distortions that result in the highest levels of human detection and the lowest levels of computer detection. In this section we will discuss each distortion type and give a recommendation for its place in further testing. The combined computer and human results based on distortion type and distortion level are shown in Table 17.

Table 17. Success Rates of Human Vs Computer By Distortion Level and Type

	Distortion Level							
	0		1		2		3	
Distortion	Hum.	Com.	Hum.	Com.	Hum.	Com.	Hum.	Com.
barrel			69.94%	21.43%	66.67%	11.90%	57.66%	7.14%
blur			82.42%	16.67%	77.78%	16.67%	74.19%	16.67%
closing			81.29%	14.29%	69.32%	9.52%	68.72%	7.14%
darken			76.40%	21.43%	67.76%	21.43%	59.88%	9.52%
dilation			78.31%	21.43%	70.11%	19.05%	55.47%	16.67%
erosion			77.25%	16.67%	61.54%	16.67%	56.96%	16.67%
laplacian			54.68%	0.00%	62.28%	0.00%	71.69%	0.00%
lighten			82.16%	21.43%	74.33%	21.43%	77.84%	9.52%
opening			82.02%	19.05%	75.16%	19.05%	73.86%	21.43%
original	80.00%	16.67%						
periodic			78.82%	14.29%	72.22%	14.29%	67.72%	9.52%
piecewise			80.72%	14.29%	74.15%	0.00%	49.65%	0.00%
resolution			77.60%	16.67%	81.66%	14.29%	78.24%	14.29%
rotate					76.16%	2.38%	78.35%	2.38%
scale_x			70.06%	0.00%	63.83%	0.00%	50.34%	0.00%
scale_y			80.36%	0.00%	80.53%	0.00%	72.97%	0.00%
speckle			79.27%	16.67%	76.10%	9.52%	45.00%	7.14%

In order to determine the best distortions to use, we look at the differences in percentages between the human and computer based detection rates. It is also important to see how the success rates are affected at each level of distortion to see if the computer detection scheme is affected by the distortion. Table 18 displays a list of the distortions that are recommended and not recommended for further use or testing. The following sections will describe the results of each distortion and provide further insight into how each distortion should be used and why it should or should not be used.

Table 18. Recommendations of Distortions

Recommended	Not Recommended
Blur	Barrel
Closing	Darken
Laplacian	Dilation
Lighten	Erosion

Periodic	Opening
Piecewise	
Resolution	
Rotate	
Scale-X	
Scale-Y	
Speckle	

5.4.2 Barrel Distortion

The barrel distortion performed moderately, with human success rates ranging from 57.66%-69.94%. However, the computer success rates were quite high with as high as 21.43% at level 1 distortion and showed a direct correlation with the human success rates. The computer rates also did not drop off significantly after any of the tested levels of distortion. The barrel distortion is not recommended for further use.

5.4.3 Blur Distortion

The blur distortion showed high human success rates ranging from 82.42% to 74.19%, and a computer success rate of 16.67% for all levels of distortion. The blur distortion on its own does not distort the image content greatly enough even under extremely high levels of distortion. With the high human success rates, the blur distortion definitely has its place in further testing, but would most likely need to be combined with additional forms of distortion to stop computerized detection.

5.4.4 Closing Operation

The closing operation performed well at level 1, with an 81.69% success rate, but dropped off significantly under distortion levels 2 and 3. The computer-based results were moderate and showed a direct correlation to the human results. If a low level of distortion is used, the closing operation may be combined with another operation for improved results.

5.4.5 Darkening/Dilation/Erosion Distortion

All three of these operations had far too high of computer success rates to be useable. At higher levels of distortion, the computerized detection may be defeated, but the human success rates would be far too low to be acceptable. These operations therefore are not recommended for further testing.

5.4.6 Laplacian Filtering

The Laplacian filtering at level 3 also performed among the best results. The greater the number of displayed edges, the greater the corresponding human results. The drastic changes that this filtering causes completely defeated the computer based scheme, but led to very good human detection under the correct settings. With further changes to the filtering and additional preprocessing, the Laplacian filtering also has room for improvement. This distortion-type should definitely have a place in further testing and use.

5.4.7 Lightening Distortion

At distortion levels 1 and 2, the lightening distortion had a computerized success rate of 21.43%, which is far too high for use in a CAPTCHA system. At level 3, the lightening operation yielded among the highest human and lowest computer-based success rates. The lightening operation has a definite place in further use, but needs to be used under high levels and possibly in combination with other distortions.

5.4.8 Opening

The opening operation did not show any significant drop in computer-based detection rates under increased levels of distortion. The computer detection scheme seems unaffected by this type of distortion and should not be used in further testing.

5.4.9 Periodic

The periodic distortion performed moderately, with average levels of both human and computer based distortion. At the level 1 distortion, the periodic distortion showed a 78.82% human success rate and a 14.29% computer success rate. For further use, the distortion should be used at lower levels of distortion as at higher levels the human success rates dropped below desirable levels. This distortion could also be combined with other distortions in an attempt to lower the computer success rate.

5.4.10 Piecewise

The piecewise distortion was detectable by the computer-based algorithms only at level 1 distortion. Beyond this level, the face images were undetectable. At level 3, the distortion was too great and brought the human-based success to under 50%. If the piecewise distortion is used in the future, the amount of distortion should either be set at level 2, or at a value in between level 1 and level 2.

5.4.11 Resolution

The resolution distortion showed quite interesting results, with the human results for level 2 being higher than the results for level 1. Additionally, the computer scheme seemed unaffected by the increasing levels of distortion. At level 3, the human success rate was 78.24%; the distortion level should be increased in further testing to see if human success rate can be maintained with a decrease in the computer success rate.

5.4.12 Rotate

The rotation distortion performed quite well with extremely low computer detection rates and higher human success rates. Although the computer results are low, it is highly likely that algorithms could easily be designed to defeat a distortion scheme based solely on rotation.

Therefore the rotation distortion should be used in further testing, but most likely only combined with other levels of distortion.

5.4.13 Scale-X

Although the scale-x distortion completely defeated the computer face detection algorithm, it performed only moderately at human detection. This distortion should be tested at lower levels to see if the human detection can be improved without any improvement to the computer detection.

5.4.14 Scale-Y

The best distortion was found to be the scale-y at distortion level 2 with an 80.5% human success rate and 0% computer success rate. The computer was unable to detect the distortion at all levels of distortion. The distortion is recommended for further use.

5.4.15 Speckle

The speckle distortion showed one of the greatest decreases in human detection rates at higher levels, with the lowest overall human success rate of 45% at level 3 distortion. The level 1 and 2 distortions showed moderate computer success rates, with fairly high human success rates. The increase from level 2 to 3 reduced the human success rate from 76.10% to 45% while only reducing the computer success rate from 9.52% to 7.14%. It is recommended that the speckle filtering only be used under light levels and possibly combined with additional distortions.

Chapter 6

Conclusions and Future Work

We have designed a new CAPTCHA system based on the concept of human versus computer face detection. The CAPTCHA works by presenting a series of distorted face and non-face images to the user and requiring the user to click on all present faces. The system improves upon the text-based CAPTCHAs by providing a more versatile language independent interface and an easier-to-use clickable interface, which can be easily expanded onto mobile platforms. The image-based CAPTCHA also helps in preventing many common text-based attacks such as optical character recognition. Our design is also an improvement over common image-based CAPTCHAs that involve the use of labeling as these systems often have many issues regarding object ambiguity. In testing we have determined the distortions which have the least effect on humans and greatest effect on computers, which is useful not only for future improvements on face detection CAPTCHAs, but for other applications which involve the use of face detection. Potential future work for this type of CAPTCHA system includes testing a larger dataset with a smaller number of distortions. With the use of sixteen distortions and three levels of each distortion, forty-eight different CAPTCHAs were needed to properly test even a single generated CAPTCHA. The results of testing proved that certain distortion types were relatively ineffective and showed the required levels for each type of distortion to be effective. This allows future work to not require different levels of distortion and a smaller set of actual distortions. Another potential extension would be to test the effects of multiple types of distortion in an effort to look for unchanged human success rates and a decrease in computer success rates. Additionally, there is room for improvement in computer detection deterrence in the aspect of adding patterns to the generated backgrounds to make the success of face-detection algorithms even lower.

REFERENCES

[1] L.V. Ahn, M. Blum, and J. Langford, "Telling Humans and Computers Apart Automatically," *Communications of the ACM*, vol. 47, 2004, pp. 56-60.

[2] A.M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, 1950, pp. 433-460.

[3] "Spam Arrest LLC," *spamarrest.com*, Jan. 2010.

[4] B. Pinkas and T. Sander, "Securing Passwords Against Dictionary Attacks," *Proceedings of the 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA: ACM, 2002, pp. 161-170.

[5] K. Chellapilla, K. Larson, P.Y. Simard, and M. Czerwinski, "Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs)," *Human Interactive Proofs*, 2005, pp. 1-26.

[6] H. Baird and K. Popat, "Human Interactive Proofs and Document Image Analysis," *Document Analysis Systems V*, 2002, pp. 531-537.

[7] L.V. Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-Based Character Recognition via Web Security Measures," *Science*, vol. 321, Sep. 2008, pp. 1465-1468.

[8] G. Moy, N. Jones, C. Harkless, and R. Potter, "Distortion Estimation Techniques in Solving Visual CAPTCHAs," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, pp. 23-28.

[9] A. Thayananthan, B. Stenger, P.H.S. Torr, and R. Cipolla, "Shape Context and Chamfer Matching in Cluttered Scenes," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[10] G. Mori and J. Malik, "Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, pp. 134-141.

[11] "ESP-PIX," *server251.theory.cs.cmu.edu*, 2004.

[12] J. Elson, J. Douceur, J. Howell, and J. Saul, "Asirra: a CAPTCHA that Exploits Interest-Aligned Manual Image Categorization," Alexandria, Virginia: ACM, 2007, pp. 366-374.

[13] "How secure is Asirra? - Microsoft Research," *research.microsoft.com*, Jan. 2010.

[14] P. Golle, "Machine Learning Attacks Against the Asirra CAPTCHA," *Proceedings of the 15th ACM Conference on Computer and Communications Security*, New York, NY: ACM, 2008, pp. 535-542.

[15] K.A. Kluever and R. Zanibbi, "Balancing Usability and Security in a Video CAPTCHA," *Proceedings of the 5th Symposium on Usable Privacy and Security*, Mountain View, California:

ACM, 2009, pp. 1-11.

[16] J. Tam, S. Hyde, J. Simsa, and L. Von Ahn, “Breaking Audio CAPTCHAs,” *Advances in Neural Information Processing Systems*, 2009.

[17] R. Santamarta, “Breaking Gmail's Audio Captcha,” *wintercore.com*, Mar. 2008.

[18] E. Bursztein and S. Bethard, “Decaptcha: Breaking 75% of eBay Audio CAPTCHAs,” *WOOT'09 3rd USENIX Workshop on Offensive Technologies*, 2009.

[19] D. Misra and K. Gaj, “Face Recognition CAPTCHAs,” *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, 2006, p. 122.

[20] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *IEEE Transactions on Image Processing*, vol. 13, 2004, pp. 600-612.

[21] Z. Wang and A. Bovik, “A Universal Image Quality Index,” *IEEE Signal Processing Letters*, vol. 9, Mar. 2002, pp. 81-84.

[22] H. Sheikh, M. Sabir, and A. Bovik, “A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms,” *IEEE Transactions on Image Processing*, vol. 15, 2006, pp. 3440-3451.

[23] N. Damera-Venkata, T. Kite, W. Geisler, B. Evans, and A. Bovik, “Image quality assessment based on a degradation model,” *IEEE Transactions on Image Processing*, vol. 9, 2000, pp. 636-650.

[24] E. Peli, “Conrtast in Complex Images,” *Journal of the Optical Society of America. A, Optics and Image Science*, vol. 7, Oct. 1990, pp. 2032-2040.

[25] D. Chandler and S. Hemami, “VSNR: A Wavelet-Based Visual Signal-to-Noise Ratio for Natural Images,” *IEEE Transactions on Image Processing*, vol. 16, 2007, pp. 2284-2298.

[26] H. Sheikh and A. Bovik, “Image Information And Visual Quality,” *IEEE Transactions on Image Processing*, vol. 15, 2006, pp. 430-444.

[27] A. Brooks, Xiaonan Zhao, and T. Pappas, “Structural Similarity Quality Metrics in a Coding Context: Exploring the Space of Realistic Distortions,” *IEEE Transactions on Image Processing*, vol. 17, 2008, pp. 1261-1273.

[28] D.M. Rouse and S.S. Hemami, “Analyzing the role of visual structure in the recognition of natural image content with multi-scale SSIM,” *Proceedings of the SPIE, Vol. 6806: Human Vision and Electronic Imaging (HVEI) XIII*, San Jose, CA: 2008, pp. 27-31.

[29] Z. Wang and E. Simoncelli, “Translation Insensitive Image Similarity in Complex Wavelet Domain,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005, pp. 573-576.

[30] Z. Wang, E. Simoncelli, and A. Bovik, “Multiscale Structural Similarity for Image Quality

Assessment," *37th IEEE Asilomar Conference on Signals, Systems and Computers*, 2003, pp. 1398-1402.

[31] H. Sheikh, A. Bovik, and G.D. Veciana, "An Information Fidelity Criterion for image quality assessment using natural scene statistics," *Image Processing, IEEE Transactions on*, vol. 14, 2005, pp. 2117-2128.

[32] M.J. Wainwright, E.P. Simoncelli, and A.S. Willsky, "Random Cascades on Wavelet Trees and Their Use in Analyzing and Modeling Natural Images," *Applied and Computational Harmonic Analysis*, vol. 11, Jul. 2001, pp. 89-123.

[33] E.W. Weisstein, "Shear - from Wolfram MathWorld," *mathworld.wolfram.com*, Nov. 2009.

[34] W.K. Pratt, *Digital Image Processing: PIKS Inside*, New York, NY: John Wiley & Sons, Inc., 2001.

[35] A. McAndrew, *An Introduction to Digital Image Processing With Matlab*, Thomson Course Technology, 2004.

[36] R.C. Gonzalez, R.E. Woods, and S.L. Eddins, *Digital image processing using MATLAB*, Upper Saddle River, NJ: Prentice Hall, 2004.

[37] G.J. Holzmann, *Beyond Photography*, Prentice Hall Professional Technical Reference, 1988.

[38] K. Sung, T. Poggio, H. Rowley, S. Baljua, and T. Kanade, "CMU Image Data Base: Face," Feb. 2010.

[39] "Creative Commons - Choose a License," *creativecommons.org*, Feb. 2010.

[40] M. Gaubatz, "MetriX MuX Home Page," *foulard.ece.cornell.edu*, Jan. 2010.

[41] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, 2002, pp. 137-154.

[42] G. Bradski, "Programmer's tool chest: The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[43] J. Yan and A.S.E. Ahmad, "A low-cost attack on a Microsoft captcha," *Proceedings of the 15th ACM conference on Computer and Communications Security*, Alexandria, Virginia, USA: ACM, 2008, pp. 543-554.

[44] M. Chew and H.S. Baird, "BaffleText: a Human Interactive Proof," *Proceedings of SPIE-IS&T Electronic Imaging, Document Recognition and Retrieval X*, 2003, pp. 305-316.

[45] A. Rusu, "Exploiting the gap between human and machine abilities in handwriting recognition for web security applications," Ph.D. dissertation, State University of New York at Buffalo, 2007.