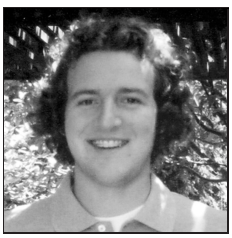
# *Breaking the Holiday Inn Priority Club CAPTCHA*

*Edward Aboufadel, Julia Olsen, and Jesse Windle*

**Edward Aboufadel** (aboufade@gvsu.edu; Grand Valley State University, Allendale, MI 49401) received his B.S. from Michigan State University in 1986, and his Ph.D. from Rutgers University in 1992. He is now an Associate Professor at Grand Valley State University. He became interested in wavelets in the early 1990's, upon learning that the Federal Bureau of Investigation had adopted a wavelet-based method to compress fingerprint images. These days, his time at work is devoted to teaching, departmental service, research with undergraduates, and day trading his CREF account.

**Julia Olsen** (jolsen@hinsdale86.org) received her B.S. from Elmhurst College in mathematics education in May 2004. She is currently teaching mathematics at Hinsdale Central High School in Hinsdale, IL. She is an advocate of using technology to enhance learning in the classroom, especially in the mathematics classroom. She is excited to be a participant in the Tablet Initiative that is in effect in Illinois H.S. District 86.



**Jesse Windle** (jwindle3@bigred.unl.edu) studies mathematics as an undergraduate at the University of Nebraska - Lincoln. He spent the summer of 2003 exploring applications of wavelets at Grand Valley State University. His hobbies include watching movies and philosophizing.

## Introduction

Yahoo! mail uses images like that in Figure 1–asking users to type in letters that they see in images–to make it harder for spammers to sign up for e-mail accounts. These simple puzzles are known as *CAPTCHAs* (Completely Automated Public Turing tests to tell Computers and Humans Apart), and the first ones were created by researchers

**Figure 1.** Example of the Gimpy-r CAPTCHA used by Yahoo! mail

at Carnegie Mellon University in 2000 [**3**]. Their mere existence presents a challenge: How easily can a CAPTCHA be solved with a computer program?

The rise of the Internet has introduced a number of challenges for businesses and consumers. Individuals sift through dozens of unwanted ("spam") messages daily.[1] Many of these messages come from spammers who use computer programs to sign up for hundreds of free accounts quickly. Ticket brokers use automated programs to attack websites that sell popular concert tickets, preventing ordinary consumers from getting good seats at reasonable prices. Over the past few years, CAPTCHAs have been developed to thwart unwanted registration and solicitation. However, some CAPTCHAs are vulnerable to attacks using elementary mathematical techniques.

CAPTCHAs are designed to help distinguish between a program and a real user. They are visual or auditory puzzles that are easily interpreted by humans, but intended to be difficult to decipher by a computer algorithm (see [**3**], [**9**], and [**10**]). There are many CAPTCHAs currently in use and they often involve an image of alphanumeric characters.

For example, e-mail providers such as Yahoo! and Hotmail require a person to solve a CAPTCHA when registering, and Ticket Master verifies a transaction only after the buyer solves a CAPTCHA correctly, while the program *SpamArrest* authenticates e-mail addresses using CAPTCHAs [**11**]. For the 2004 Michigan Democratic Presidential Caucuses, the CAPTCHA in Figure 2 was used as part of the online registration process [**7**].

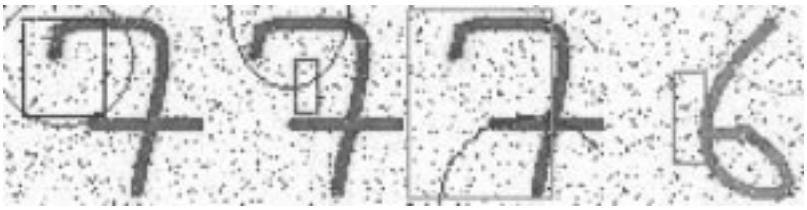


**Figure 2.** A CAPTCHA used by the Michigan Democratic Party in 2004

A CAPTCHA is considered broken if a computer algorithm can quickly solve the puzzle at least four out of five times on average. The first CAPTCHA we know to have been broken was "EZ-Gimpy," illustrated in Figure 3. Greg Mori and Jitendra Malik at the University of California, Berkeley, wrote a program implementing object recogni-

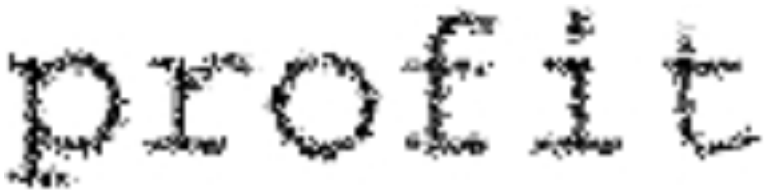



**Figure 3.** Example of the EZ-Gimpy CAPTCHA broken by Mori and Malik

[1] According to the International Telecommunications Union, nearly 80% of all e-mail in the United States is spam [**6**].

tion techniques and dictionary crosschecking that correctly interprets this CAPTCHA 93% of the time [**8**].

In this paper, we present a method for solving the CAPTCHA used by the Holiday Inn chain of hotels. Members of their Priority Club must solve a CAPTCHA to sign up for the Rewards Dining program (see Figure 4) [**5**]. Hereafter, we refer to this CAPTCHA as the HIPC CAPTCHA. Our methods of breaking the HIPC CAPTCHA employ linear regression and rotation of axes. It also uses bivariate Haar wavelet filters, which come from linear algebra and are accessible to undergraduates. Our methods were implemented in *Maple*.

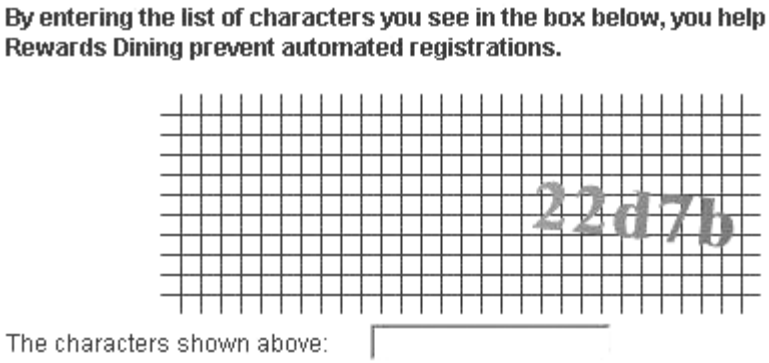



**Figure 4.** Downloaded example of the Holiday Inn Priority Club CAPTCHA

## Isolating the characters

In order to develop our methods, we wrote a *Maple* worksheet that created simulated HIPC CAPTCHAS. The code begins by generating a random string of five grey alphanumeric characters of a uniform font type and size. The string is rotated and placed on top of a regular grid of horizontal and vertical black lines, with a white background. We created test cases where the characters were limited to the following set of Roman and Arabic numerals: $\{I, V, X, C, D, L, M, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, although this restriction isn't necessary for our method to work.

The first step to break the HIPC CAPTCHA is to convert the image to portable grey map (PGM) format, using a program such as *Pic-2-Pic* [**14**]. In this format, a matrix $A$, whose entries are integers between 0 (black) and 255 (white), represents the image. Each entry of the matrix corresponds to a pixel's shade. In the HIPC CAPTCHA, pixels for the lines are 0, those for the background are 255, and those for the characters are 96 (grey). Hence, the set of indices whose value is 96 is the set of pixels that make up the characters in the image:

$$C = \{(i, j) \mid a_{i,j} = 96\}.$$

The next step is to undo the rotation of the string. To determine the rotation angle, we treat $C$ as a set of data points and apply linear regression to find the line of best fit. If $m$ is the slope of this line, then the angle of rotation is $\theta = \arctan m$. (See Figure 5, and note that the string of symbols is upside down because in the matrix $A$, $a_{1,1}$ corresponds to the upper-left corner of the image.) In the *Maple* worksheet that created CAPTCHA examples, the angle $\theta$ was chosen randomly to be between $-10°$ and $10°$.

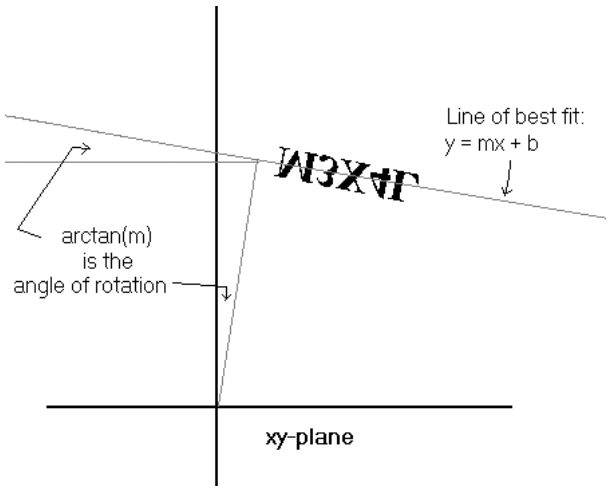



**Figure 5.** The angle of rotation is found by linear regression

The characters are rotated by applying the matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

to every point of $C$.

After rounding, we create a new matrix $\tilde{A}$, with the roles of black and white reversed, so that entries of 255 (white) are pixels from the characters, and 0 are pixels from the background (see Figure 6).

**Figure 6.** An example of $\tilde{A}$

To separate the characters, we scan through each column of the rotated image $\tilde{A}$, looking for columns that are all black pixels. This is a frequent place for one character to end and another to begin. Occasionally, there is not a column of black pixels between the characters. In this case, since all the characters are fewer than 36 columns wide, we stop scanning after 36 columns and perform a similar test, this time searching for a column that contains all black pixels except one, and so on. Through this process, the five alphanumeric characters in the CAPTCHA are isolated. The characters are placed in separate $32 \times 32$ matrices $L_1$, $L_2$, $L_3$, $L_4$, and $L_5$. (The letter M turned out to be noticably wider than the rest, so any time a large letter was found, it was assumed to be M.)

## Recognizing the characters

There is a large body of literature on character recognition (see the bibliography in [**12**]). A wavelet-based search engine for images [**13**] inspired the use of bivariate

Haar wavelet filters to identify the characters in the five matrices. (For an introduction to wavelets, see [**1**].)

Bivariate Haar wavelet filters operate on $2^n \times 2^n$ matrices. These filters transform one such matrix into another, with the idea being that the entries in the new matrix may be a rearrangement of information that will help with the application at hand, in this case, character recognition.

There are four bivariate Haar filters, $H$, $G_v$, $G_h$, and $G_d$, called the *low-pass*,[2] *vertical high-pass*, *horizontal high-pass*, and *diagonal high-pass* filters respectively [**2**]. The four filters are represented by these matrices:

$$H = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad G_v = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

$$G_h = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \qquad G_d = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

In the case where $n = 1$, filtering is done by a dot-product-like operation, called the Frobenius inner product. To filter the matrix

$$Q = \begin{bmatrix} w & x \\ y & z \end{bmatrix},$$

multiply each entry by the corresponding entry in the filter, and then add the results together. For example, applying the diagonal high-pass filter yields

$$g_d = G_d[Q] = \frac{1}{2}w - \frac{1}{2}x - \frac{1}{2}y + \frac{1}{2}z.$$

A concrete way to think of filtering is that the low-pass filter computes an average of matrix entries, while the high-pass filters measure changes between adjacent rows, columns, or along a diagonal.

After all four filters have been applied, the results are put into in a new matrix,

$$\hat{Q} = \begin{bmatrix} h & g_v \\ g_h & g_d \end{bmatrix}.$$

Here is an example of $Q$ and the resulting $\hat{Q}$:

$$Q = \begin{bmatrix} 4 & 9 \\ 7 & -8 \end{bmatrix} \qquad \hat{Q} = \begin{bmatrix} 6 & 5 \\ 7 & -10 \end{bmatrix}$$

In the case of $n = 2$, filtering is an iterative, two-step process. We begin by partitioning a $4 \times 4$ matrix into four $2 \times 2$ blocks,

$$Q = \left[\begin{array}{c|c} Q_{nw} & Q_{ne} \\ \hline Q_{sw} & Q_{se} \end{array}\right].$$

When the filter $H$ is applied to each of the sub-matrices, the result is the $2 \times 2$ matrix

[2] It would seem that the high-pass filters should be labeled with $H$'s, but that is not the convention.

$$H[Q] = \begin{bmatrix} H[Q_{nw}] & H[Q_{ne}] \\ H[Q_{sw}] & H[Q_{se}] \end{bmatrix}.$$

Next, we apply the other three filters to the four sub-matrices of $Q$, creating three more $2 \times 2$ matrices $G_v[Q]$, $G_h[Q]$, and $G_d[Q]$, and place the results in a new $4 \times 4$ matrix in a natural way:

$$Q_1 = \begin{bmatrix} H[Q] & G_v[Q] \\ G_h[Q] & G_d[Q] \end{bmatrix}$$

At this point, the first part of the process is done.

Here is an example of the calculations so far:

$$Q = \left[\begin{array}{cc|cc} 4 & 9 & 3 & 0 \\ 7 & -8 & 1 & 6 \\ \hline 7 & -6 & 1 & 9 \\ -2 & 1 & -1 & -3 \end{array}\right] \qquad Q_1 = \left[\begin{array}{cc|cc} 6 & 5 & 5 & -1 \\ 0 & 3 & 5 & -3 \\ \hline 7 & -2 & -10 & 4 \\ 1 & 7 & 8 & -5 \end{array}\right]$$

The second step is to use the $2 \times 2$ matrix $H[Q]$ as input into another round of filtering, in the same manner as described earlier for $n = 1$. The result is a new $2 \times 2$ matrix

$$Q_2 = \begin{bmatrix} H[H[Q]] & G_v[H[Q]] \\ G_h[H[Q]] & G_d[H[Q]] \end{bmatrix},$$

which replaces $H[Q]$ in $Q_1$ above. We call the final matrix $\hat{Q}$.

$$\hat{Q} = \begin{bmatrix} Q_2 & G_v[Q] \\ G_h[Q] & G_d[Q] \end{bmatrix}$$

To complete our example, we find

$$\hat{Q} = \left[\begin{array}{cc|cc} 7 & -1 & 5 & -1 \\ 4 & 2 & 5 & -3 \\ \hline 7 & -2 & -10 & 4 \\ 1 & 7 & 8 & -5 \end{array}\right].$$

Note that we cannot apply the filters a third time to $H[H[Q]] = [7]$, because that is a $1 \times 1$ matrix, so our process is done. The signal $Q$ is now decomposed into "wavelet coefficients," which are the entries in the $4 \times 4$ matrix $\hat{Q}$.

This process works with any matrix of order $2^n \times 2^n$, where the filters are applied up to $n$ times iteratively, each time to a smaller matrix. Returning to the method of breaking the HIPC CAPTCHA, the $L_i$ matrices are $32 \times 32$, and we choose to apply the filters five times. For each $i$, after the fifth step, we label the upper left hand $4 \times 4$ matrix $D_i$. Here is a diagram of the situation, with the size of all the blocks indicated.

$$\hat{Q} = \left[\begin{array}{cc|c|c} D_i & 4 \times 4 & & \\ \cline{1-2} 4 \times 4 & 4 \times 4 & 8 \times 8 & 16 \times 16 \\ \hline 8 \times 8 & & 8 \times 8 & \\ \hline 16 \times 16 & & & 16 \times 16 \end{array}\right]$$

The upper-left-hand entry of $D_i$ is the only element in the matrix

$$H[H[H[H[H[Q]]]]],$$

with the rest of the entries of $D_i$ coming from applying the high-pass filters to $H[H[H[Q]]]$ and $H[H[H[H[Q]]]]$. $D_i$ will contain the only information we need to use to recognize the characters. This is because the results outside $D_i$ in $\hat{Q}$ tend to be close to zero, no matter which letter matrix is being filtered, while the outputs within $D_i$ differ greatly from one letter to the next. Consequently, the only wavelet coefficients that we need to compute are in $D_i$.

Before determining $D_1$, $D_2$, $D_3$, $D_4$, and $D_5$, it is necessary to create canonical images of both the Roman and Arabic numerals, to decompose each image by filtering, and to record the upper-left $4 \times 4$ corner of the decomposed images in a *dictionary*. Character recognition then comes down to comparing the $D_i$'s with each entry $M$ in the dictionary. For each $D_i$, we compute "distances" between every dictionary entry and $D_i$ by using the matrix 2-norm, denoted $\|D_i - M\|_2$, computed by subtracting the two matrices and adding the squares of entries in the difference.

Naturally, we choose for the primary candidate the dictionary entry that makes this norm the smallest. When the primary candidate is "I" or "L", it is necessary to see if the number of pixels in the original image of the character is closer to the number of pixels in the canonical "I" or the canonical "L".

## Results

This algorithm, written in *Maple* code, solves the HIPC CAPTCHA variant in under 10 seconds on a new PC, and is successful nearly 100% of the time. Note that bivariate wavelets reduced the amount of information needed to distinguish characters from $32^2$ to $4^2$ pieces of data, resulting in accurate results in reasonable times. Further investigations could consider the entire English alphabet with Arabic numerals, other fonts, and different letter sizes. Despite such changes, other work we have done leads us to expect the success rate of the algorithm to remain above the important 80% threshold, and we consider the HIPC CAPTCHA to be broken.

There are other CAPTCHAs in use (as of Summer, 2004) that can be broken using this method. Users of General Electric's website to schedule appliance service calls are confronted with the CAPTCHA in Figure 7 as part of the reservation process. The Chicago Cubs baseball team requires ticket purchasers to complete the CAPTCHA in Figure 8 when ordering tickets. Both use a standard font and a background that can be removed.


**Figure 7.** Example of a CAPTCHA used by General Electric


**Figure 8.** Example of a CAPTCHA used by the Chicago Cubs

As we indicated at the beginning, there are a number of different CAPTCHAs in use today. Because examples are easily downloaded from websites, CAPTCHAs are open to various attacks. Internet sites should use CAPTCHAs without a standard font–some possibilities are Gimpy-r or BaffleText [**4**]. At the same time, if CAPTCHAs become too difficult for humans to recognize, then their purpose will be lost.

## References


1. Edward Aboufadel and Steven Schlicker, *Discovering Wavelets*, Wiley, 1999.
2. Paul S. Addison, *The Illustrated Wavelet Transform Handbook*, Napier University, U.K., 2002.
3. Manuel Blum, Nicholas Hopper, John Langford, and Luis von Ahn, The CAPTCHA Project (Carnegie Mellon University), www.captcha.net.
4. Monica Chew and Henry Baird, BaffleText: A human interactive proof, *Document Recognition and Retrieval X*, 2003.
5. Holiday Inn Priority Club homepage, www.sixcontinentshotels.com/priorityclub.
6. International Telecommunication Union, Spam messages on the increase, www.itu.int/osg/spu/newslog/2004/05/26.html#a652.
7. Michigan Democratic Party, Michigan Democratic Party Presidential Caucuses homepage, www.applytovote.com.
8. Greg Mori and Jitendra Malik, Recognizing objects in adversarial clutter: breaking a bisual CAPTCHA, *2003 Conference on Computer Vision and Pattern Recognition (CVPR '03)* , Vol. I, (2003) 134–144.
9. Sara Robinson, Can hard AI problems foil Internet interlopers?, *SIAM News*, **35** (3) (April 2002) 1.
10. ———, Up to the challenge: Computer scientists crack a set of AI-based puzzles, *SIAM News*, **35** (9) (November 2002) 24.
11. SpamArrest homepage, www.spamarrest.com.
12. Øivind Due Trier, Anil Jain, and Torfinn Taxt, Feature extraction methods for character recognition—A survey, *Pattern Recognition*, **29** (4) (1996) 641–662.
13. James Wang, Content-based Image Retrieval Project. www-db.stanford.edu/IMAGE.
14. *WaveL Pic2Pic Converter*, WaveL Software homepage, www.wavelsoftware.com.


The assumption that such discrete data can be plotted as a continuous curve is often referred to as the *continuum hypothesis*.

*Principles of Mathematical Modeling*, C. L. Dym and E. S. Ivey, Academic Press (1980), p. 75.