Algorithm To Break Visual CAPTCHA

Prof. (Mrs.) A.A. Chandavale and Prof. Dr.A.M. Sapkal and Dr.R.M.Jalnekar
Member , IEEE and Member ,IETE and Life Member IETE
C_anjali38@yahoo.com and ams@extc.coep.org.in and rajesh_jalnekar@yahoo.com

Abstract

Completely Automated Public Turing Tests to Tell Computers and Humans Apart (CAPTCHAs) are the automatic filters that are widely used these days to disallow any automated script that can perform the work of a human. CAPTCHAs are built in such a way that it is very difficult for any automated script to break them. In this paper, an approach to break text based CAPTCHAs has been proposed that first preprocesses the given CAPTCHA, segments its characters, and then recognizes the characters depending on it's features. The breaking of CAPTCHA gives strength of CAPTCHA which in turn helps to develop more robust CAPTCHA The testing results shows that the algorithm successfully breaks the CAPCHA. This algorithm will be also helpful to screen reader program. The results show that it is not a trivial task to design a CAPTCHA scheme that is both usable and robust.

1. INTRODUCTION

CAPTCHAs are widely used these days to prevent any script that is automated to perform an activity that is meant for a human. They are basically used to prevent bots which use these scripts to make false accounts at the websites. CAPTCHAs can be categorized into two types mainly: - (i) Visual CAPTCHA and (ii) Audio CAPTCHA. Visual CAPTCHAs have characters with noise applied in background. Audio CAPTCHAs have characters written on a complex background like Visual CAPTCHAs but they also read out the characters which is helpful for a physically disabled person or in case some character cannot be recognized. An example of Audio CAPTCHA is shown in fig.1

.Fig.1. An example of Audio CAPTCHA of MSN/Hotmail.

A CAPTCHA is a program that can generate and grade tests that: (A) most humans can pass, but (B) current computer programs can't pass. Such a program can be used to differentiate humans from computers and has many applications for practical security, including (but not limited to) as :

(A) Online Polls. In November 1999, slashdot.com released an online poll asking which was the best graduate school in computer science (a dangerous question to ask over the web!). As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots by using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own voting program and the poll became a contest between voting\bots". MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll requires that only humans can vote.

(B) Free Email Services. Several companies (Yahoo!, Microsoft, etc.) free email services, most of which suffer from a specific type of attack: bots that sign up for thousands of email accounts every minute. This situation can be improved by requiring users to prove they are human before they can get a free email account. Yahoo!, for instance, uses a CAPTCHA of our design to prevent bots from registering for accounts. Their CAPTCHA asks user to read a distorted word such as the one shown below fig.2.(current computer programs are not as good as humans at reading distorted text).

(C) Search Engine Bots. Some web sites don't want to be indexed by search engines. There is an html tag to prevent search engine bots from reading web pages, but the tag doesn't guarantee that bots won't read the pages; it only serves to say \no bots, please. However, in order to truly guarantee that bots won't enter a web site, CAPTCHAs are needed.

(D) Worms and Spam: CAPTCHAs also offer a plausible solution against email. CAPTCHA guarantees an email acceptance only if you know there is a human behind the other computer. A few companies, such as www.spamarrest.com are already marketing this idea.

(E) Preventing Dictionary Attacks. Pinkas and Sander [16] have suggested using CAPTCHAs to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring a human to type the passwords.

The rest of the paper is organized as follows: Related work about CAPTCHA is given in section 2. Section 3 gives implementation details of algorithm used for breaking CAPTCHA. The testing results are given in section 4. Finally section 5 concludes the paper.

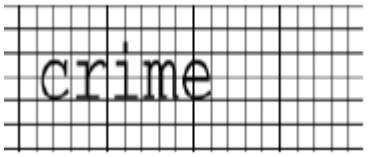
978-0-7695-3884-6/09 $26.00 © 2009 IEEE

2. RELATED WORK

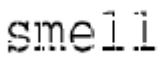
These days, CAPTCHAs are designed in the toughest possible way that prevents any algorithm to break them but still significant work has been done to break these CAPTCHAs .In 2003, Mori and Malik [15] proposed a shape matching algorithm to break EZ-Gimpy and Gimpy CAPTCHAs. They achieved a success rate of 92% in case of EZ-Gimpy and 33% in case of Gimpy. In 2004, Moy et. al. [12] use distortion estimation technique to break EZ-Gimpy CAPTCHAs and achieved a great success rate. In this paper only EZ- Gimpy CAPTCHAs has been focused. In recent research, [13] shows "segmentation" is a much more difficult problem than "recognition" since machine learning algorithms can efficiently solve the recognition problem, but currently we have no effective general algorithm to solve the segmentation problem causing by these added clutters. In [14], Chellapilla and other researchers use the image opening and labeling technique to design a segmentation algorithm. When the difference of width between clutters and characters is very noticeable, it is able to separate the noise from characters effectively. However, when the difference is not so noticeable, this algorithm will either be unable to eliminate noise, or it may break the characters when attempting to remove image noise. Also work is going on to design more robust CAPTCHA .

A new CAPTCHA based on identifying an images in upright orientation is being seen. This has advantage over the traditional text recognition techniques are that it is language-independent, does not require text-entry (*e.g.* for a mobile device), and employs another domain for CAPTCHA generation beyond character obfuscation but it needs upright orientation by human.[8] In 2005 , R. Datta et. Produce controlled distortions on randomly chosen images and present them to the user for annotation from a given list of words. The distortions are performed in a way that satisfies the incongruous requirements of low perceptual degradation and high resistance to attack by content-based image retrieval systems. Word choices are carefully generated to avoid ambiguity as well as to avoid attacks based on the choices them selves but results show that system is not full proof attack resistant so not in use and popular. Yan and Ahmad [9] address a problem of systematic method for verifying whether a CAPTCHA is indeed robust. EZ-Gimpy and Gimpy CAPTCHAs can be divided into four categories namely (i) Simple (No mesh) Background, (ii) Black Mesh Background, (iii) White Mesh Background and (iv) Loosely Connected Characters. In Simple (No Mesh) EZ-Gimpy, the characters are written on a simple background whereas in Black and White Mesh EZ-Gimpy the characters are written on black and white mesh respectively. In Loosely Connected Characters type, the pixels of the characters are loosely connected. Algorithm developed in this paper uses feature extraction technique to recognize the characters and thus analyses security of all types of CAPTCHAs mentioned here.

(a)



(b)



(c)

(d)

Fig.2 EZ-Gimpy CAPTCHAs. (a) Simple (No Mesh) Background. (b) Black Mesh Background. (c) White Mesh Background. (d) Loosely Connected Characters.

3. IMPLEMENTATION

As mentioned earlier an algorithm has different phases such as preprocessing, segmentation, feature extraction and character recognition which are explained in detail as mentioned below:

3.1 Preprocessing

Preprocessing will convert input CAPTCHA image into cleared image by first converting into gray scale, then carry out binarization , then removes line, & dots (if any present on image) Since there are four types of EZ-Gimpy CAPTCHA, each one has different preprocessing operations which are as follows:

3.1.1. Simple (No Mesh) Background

(a) The given CAPTCHA is first converted to gray scale.

(b) Binarization of an image can lead to an image whose pixels have only two possible intensity values.

3.1.1.1. Color To Gray

The CAPTCHA image is given as an input to the program. The image should be in .bmp format. The CAPTCHA image is then converted into gray scale image.

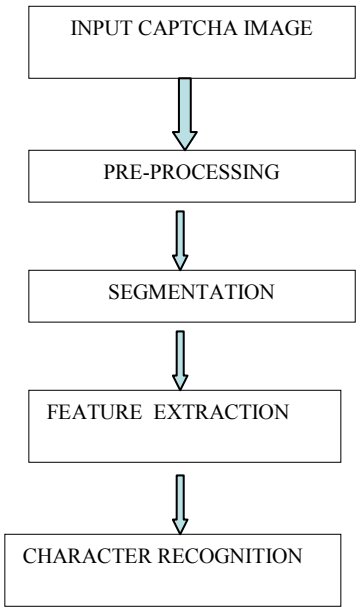


Fig.3 System Architecture

It is converted into gray scale because CAPTCHA image contains many colors and to work on each of them is very difficult so converting it into gray scale helps only to work on 256 intensity values.

Algorithm to Convert 24bit to 8bit Gray Scale Image

- Get BMP File As Input File
- Extract Header Information
- If input is 24 bit , Create Output Header & Palette.
- Read three pixels from input file & calculate average.
- Write Average in Output File.
- Repeat steps 4 & 5 till end of file.

3.1.1.2. Binarization

Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black, and either 1 or 255 for white.

Binary images are often produced by thresholding a grayscale or color image, in order to separate an object in the image from the background. The color of the object (usually white) is referred to as the foreground color. The rest (usually black) is referred to as the background color.

3.1.2. Black Mesh Background

(a) The given CAPTCHA is first converted to gray scale and binarized as mentioned above.

(b) Binarization of an image can lead to an image whose pixels have only two possible intensity values

(c) CAPTCHA image has noise such as horizontal lines, vertical lines and dots that needs to remove so as to get clear image.

3.1.2.1 Line Removal

The CAPTCHA images sometimes contain horizontal lines and vertical lines. To remove lines the number of continuous black pixels in row or columns is counted. If the count is more than 80% of total width or height of the image, then detected as a line and thus removed it by making it white.

3.1.2.2 Discontinuity Removal

After the lines are removed, characters become discontinuous. To remove this discontinuity, the original image (image before line removal) is compared with the newer one and fills the gaps in between so that characters remain continuous. This will help in recognizing the characters

3.1.2.3 Dot Removal

After Binarization sometimes CAPTCHA images may contain unnecessary set of black pixels i.e. dot. To remove them, the image is scanned. Then after getting first black pixel check its neighboring 8 pixels. If all of them are white, then make the black pixel white. If these dots are bigger (greater than one pixel) then count the pixels in it, if the count is less than 40 then make it white. This is because, 40 pixels can't make a character, so it is an unnecessary dot.

3.1.3. White Mesh Background

(a) The given CAPTCHA is first converted to gray scale and binarized.

(b) The image is now inverted. Now the same black mesh background as stated above is applied

(c) The image is now inverted to get back the original image

3.1.4. Loosely Connected Characters

(a) The given CAPTCHA is first converted to gray scale and binarized.

(b) The noise is removed and get the clear image as stated above.

3.2 Segmentation

After the image passes the preprocessing stage, characters need to be segmented because if characters are joined then it is very difficult to recognize them. In EZ-Gimpy CAPTCHA all the characters are distant so it is not very difficult to segment them. Checking continuous black pixels separates characters. Once the program checks black pixels it is made red so that program can understand that the specific character is already separated.

3.3 Feature Extraction and Character Recognition

Each character has some unique set of features. The features, which were used, are as follows:

(i)*Number of Holes* - Each character is checked whether it has a hole or not. Characters a, b, d, e, g, o, p, q each have 1hole and rest of the characters do not have a hole.
(ii) *Height of Character* - Each character is categorized into small or big on the basis of its height. A threshold is taken which categorizes the given character.
(iii) *Maximum Number of White-Black Transitions* (Vertical intersection) - A line cutting the character is drawn and the maximum numbers of white-black transitions that are possible are noted. Example: - When a line is drawn through character 'a', maximum 3 transitions are possible. Similarly transitions for other characters were also noted.
(iv)*Nature of Vertical Stroke*: - A vertical stroke (Blue in color) is drawn for character with 'big' height along the vertical stroke of the character. If the character is of 'small' height '*' is noted. If there is no hole in character '1' is noted. And if there is a hole in character then the position of vertical stroke relative to the hole i.e. either left or right is noted. According to the features of the characters, characters are recognized. This module can also be useful in recognizing handwritten characters.

4. TESTING RESULTS

The samples for testing images are taken from web sites of Mori and Malik. Out of given standard database, the author could able to get a success for at least 80% images. It has been successfully implemented over the college lab's host and has given satisfactory performance. Because of the user-friendly interface provided, it could be easily incorporated for the any firm interested. This software used in the college laboratory proved to be very useful. This paper encapsulates all the features of the application as well as standard software engineering principles that have been implemented for making of the application. It gave us good accuracy and success. Fig.4 and Fig.5 give preprocessing results where as fig.6 and fig.7 gives Graphical user interface (GUI) implementation results.

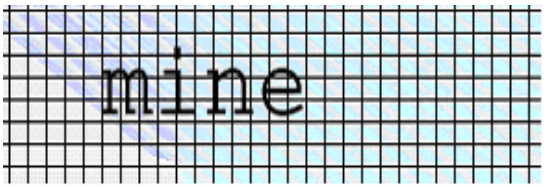
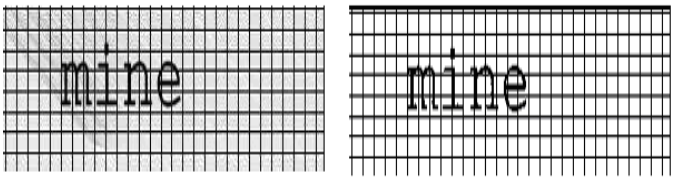
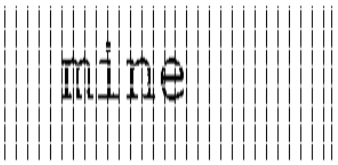


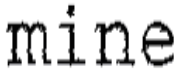
Fig. 4. Pre condition for preprocessing



(a) (b)



(c) (d)



(e)

Fig.5. stages of preprocessing (a) After Gray Scale (b) After Binarization (c) After horizontal line removal
(d) After vertical line removal (e) Result after preprocessing

Fig.6. GUI Implementation after preprocessing

Fig.7. GUI Implementation after character recognition.

5. CONCLUSION

The algorithm addressed in paper successfully breaks a text based CAPTCHA .It took us a lot of time to design our basic approach of character recognition. We found algorithm based on pattern matching but the accuracy of that algorithm is very less. So we designed our own algorithm for character recognition. We tried different approaches until we finally were able to figure out the successful approach. As far as technical things are to be considered we can say that the accuracy that we obtain from the feature extraction was better than the pattern matching for character recognition. Another point is a normal OCR program could not recognize character in distorted background. So the module of character recognition in breaking the CAPTCHA can help OCR to recognize the character in distorted image.

ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers for their useful suggestions that helped in improving the quality of this paper. This work was supported in part by MAEER's MIT,Pune.

REFERENCES

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learning Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[2] P. Mitra, C. Murthy, and S. Pal, "Unsupervised feature selection using feature similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 301–312, Mar. 2002.

[3] J. Dy and C. Brodley, "Feature selection for unsupervised learning," *J. Mach. Learn. Res.*, vol. 5, pp. 845–889, Aug. 2004.

[4] D. Povey, M. J. F. Gales, D. Y. Kim, and P. C. Woodland, "MMI-MAP and MPE-MAP for acoustic model adaptation," in *Proc. Eur. Conf. Speech Commun. Technol.*, Geneva, Switzerland, Sep. 2003, pp.
1981–1984.

[5] M. J. F. Gales, "Semi-tied covariance matrices for hidden Markov models," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 3, pp. 272–281, May 1999.

[6] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland, *The HTK Book, version 3.3*. Cambridge, U.K.: Cambridge Univ. Eng. Dept., 2005.

[7] G. Saon, D. Povey, and G. Zweig, "CTS decoding improvements at IBM," presented at the *Proc. EARS STT Workshop*, St. Thomas, U.S. Virgin Islands, Dec. 2003, p. XXX.

[8] Rich Gossweiler ,Maryam Kamvar , Shumeet Baluja **"**What's Up CAPTCHA? CAPTCHA Based On Image Orientation"International WWW 2009 ,April 20-24 ,2009 ,Spain ACM

[9] J Yan and A S El Ahmad. "Breaking Visual CAPTCHAs with Naïve Pattern Recognition Algorithms", in IEEE Conference Proc. of the 23rd Annual Computer Security Applications Conference (ACSAC'07)

[10] Ritendra Datta ,Jia Li and James Z.Wang " IMAGINATION : A Robust Image –based CAPTCHA Generartion System"Internation Conference ACM MM'05 ,November 6-11 ,Singapore

[11] K Chellapilla, K Larson, P Simard, M Czerwinski, "Computers beat humans at single character recognition in reading-based Human Interaction Proofs", 2nd Conference on Email and Anti-Spam (CEAS), 2005.

[12] Gabriel Moy, Nathan Jones, Curt Harkless, and Randall Potter "Distortion Estimation Techniques in Solving Visual CAPTCHAs" proceedings of the Computer Vision and Pattern Recognition (CVPR'04) Conference ,IEEE Computer Society ,vol. 2 ,pp.23-28,2004.

[13] Kumar Chellapilla Patrice Y. Simard **"**Using Machine Learning to Break Visual
Human Interaction Proofs (HIPs) **"** in L. K.Saul, Y. Weiss, and L. Bottou, editors, Advances in Neural Information Processing Systems 17, pp. 265–272. MIT Press,
Cambridge, MA, 2005.

[14] L von Ahn, M Blum and J Langford. "Telling Humans and Computer Apart Automatically", CACM(Communications of ACM), V47, No2, February 2004.

[15] G Mori and J Malik. "Recognising objects in adversarial clutter: breaking a visual CAPTCHA",IEEE Conference on Computer Vision & Pattern Recognition (CVPR), 2003 , IEEE Computer Society ,vol. 1 ,pp.I-134-I-141, June 18-20 ,2003.

[16] B. Pinkas and T. Sander, "Securing passwords against dictionary attacks," in Proc. of the 9th CCS, V. Atluri, Ed. ACM Press, Nov. 2002, pp. 161–170.