International Journal of Computer Applications (0975 – 8887)
Volume 89 – No 20, March 2014

Towards Solving the Google CAPTCHA

Akriti Mehta
Student, G D Goenka World Institute
Sohna, Gurgaon 12202, Haryana, India

Deepak Sharma
Lecturer, G D Goenka World Institute
Sohna, Gurgaon 122002, Haryana, India

ABSTRACT

CAPTCHA is now a technology which is used all over the internet to protect websites from program bots. One of the major issues which CAPTCHA provides a solution to is, stopping spammers for creating unlimited accounts and sending emails to millions of people spam mails. In this paper, we are proposing our algorithm for solving the Google CAPTCHA. Being able to solve the Google CAPTCHA using a program would mean that there are some security issues which need to be considered at the earliest and this would assist us develop a better CAPTCHA.

General Terms

Security, Artificial Intelligence, CAPTCHA

Keywords

CAPTCHA, Google, Completely Automated Public Turing test to tell Computers and Humans Apart

1. INTRODUCTION

Completely Automated Public Turing Test to Tell Computer and Humans Apart (CAPTCHA) is a test in which a human friendly task is given to the users to make sure that they are Humans and not computers. It can be seen anywhere on the internet, it is very hard to find a sign up page where CAPTCHA isn't found. There are many different kinds of CAPTCHAs found like voice recognition, image recognition to character recognition. The most common CAPTCHAs are the text based CAPTCHAs and require the user to recognize the text in a distorted image, something that supposedly can't be done by computers.

There are numerous aspects which are to be looked into while evaluating CAPTCHAs. According to Kurt Alfred Kluver and Richard Zanibbi there are four properties which are desirable for a CAPTCHA, [1,2]namely:

- Automated
- Open
- Usable
- Secure

These properties make sure that the CAPTCHAs are user friendly but not program bot friendly and securing the system.

What makes the text CAPTCHAs so robust is locating the characters, locating and dividing characters in the correct order and lastly recognizing them correctly. But, "Computers beat humans at single character recognition in reading-based Human Interaction Proofs" [3]. Therefore, the primary strength of CAPTCHAs is that it hard dividing them into individual character rather than to recognize those characters (which is relatively an easier job for the computer). Figure 1 shows some of the Google CAPTCHAs.

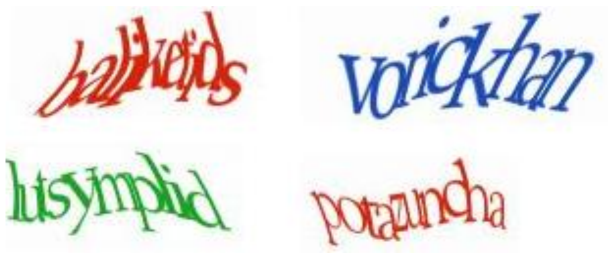


Figure 1: Examples of Google CAPTCHAs exhibiting resistance to division

There have been many websites which have used this technique of crowding the characters together in the past including Google, Microsoft, Yahoo, and Megaupload [4] this can be seen in figure 1.

2. RELATED WORK

A lot of research and effort has been put in solving the different types of CAPTCHAs. It comes as no surprise that none of them have achieved a 100% success rate.

The authors of the paper "Robustness of Google CAPTCHAs" [4] have prepared a novel attack on Google CAPTCHA. Although Google CAPTCHA uses different styles and font typefaces, applies CCT, and uses heavy distortion; there is one thing that remains invariant: shape pattern in some characters. This attack impacts a success rate equal to 95% for individual segmented characters and for an average 5.5 characters in Google CAPTCHA, the attack achieves a success rate of 46.75%.

The paper "A low-cost Attack on a Microsoft CAPTCHA" [5] discusses the attack on MSN CAPTCHAs. The author mentions that this attack yields an efficiency rate of 12% for the Google CAPTCHAs. The potential issue of the whole program being solvable are acknowledged and looked into. An attack can be defeated, or, it can be said that a CAPTCHA can be made more robust by making it division(segmentation) resistant. The Google scheme of CAPTCHAs appears to provide better security because of Crowding Characters Together algorithm and the increased level of distortion.

"Algorithm to break visual CAPTCHA" [6] ponders on other important things, along with discussing about the algorithm to break the CAPTCHA, i.e., it also talks about the design. The basic algorithm presented by the author is based on pattern matching. Division of the characters is done by checking for continuous black pixels and color change.

Table 1: Success rates of some of the attacks on CAPTCHAs

Paper	Success Rate
"Robustness of Google CAPTCHAs" [4]	46.75%
"A low-cost Attack on a Microsoft CAPTCHA" [5]	12% (for Google CAPTCHA)
"Algorithm to break visual CAPTCHA" [6]	80%
"CAPTCHA Security: a case study" [7]	36%

.

3. STUDYING THE GOOGLE CAPTCHA

Whenever an algorithm is applied there is a set pattern that is being followed as ultimately is the computer which is following human instructions and logic.

After observing Google CAPTCHAs few patterns can be recognized which can be used to our advantage. These patterns are as follows:

- The characters are crowded together
- The background is always white
- The text color is either red, blue or green
- The width of the characters is random
- The characters may/may not be italic and/or bold

4. PROPOSED WORK

As mentioned in [4], all the character can be on the basis of pattern, there are mainly four types of patterns namely: loop, dot, cross and S shaped. All the characters can be classified according to these four patterns.

- Loop shape- "a", "b", "c", "d", "e", "g", "o", "p", "q".
- Dot shape - "i" and "j".
- Cross shape – "t' and "f".
- S shaped – "s" and "z".

In the proposed method there would be four stages. The first stage would be of preparing the image, in the second character patterns are searched for proceeded by the division process in the third stage. And finally the fourth stage would be of recognizing the characters.

While preparing the image some image processing techniques are applied on the image which helps not onlu in increasing the efficiency and effectiveness, but also in reducing the time complexity of the procedure. In the next stage the characters based on the patterns are recognized, i.e. which shaped characters are present in the image. In the third stage, on the basis of the character shapes recognized the image in divided to get individual characters which are easy to recognize by a computer [1]. And lastly, these individual characters are read using the OCR.

5. METHODOLOGIES

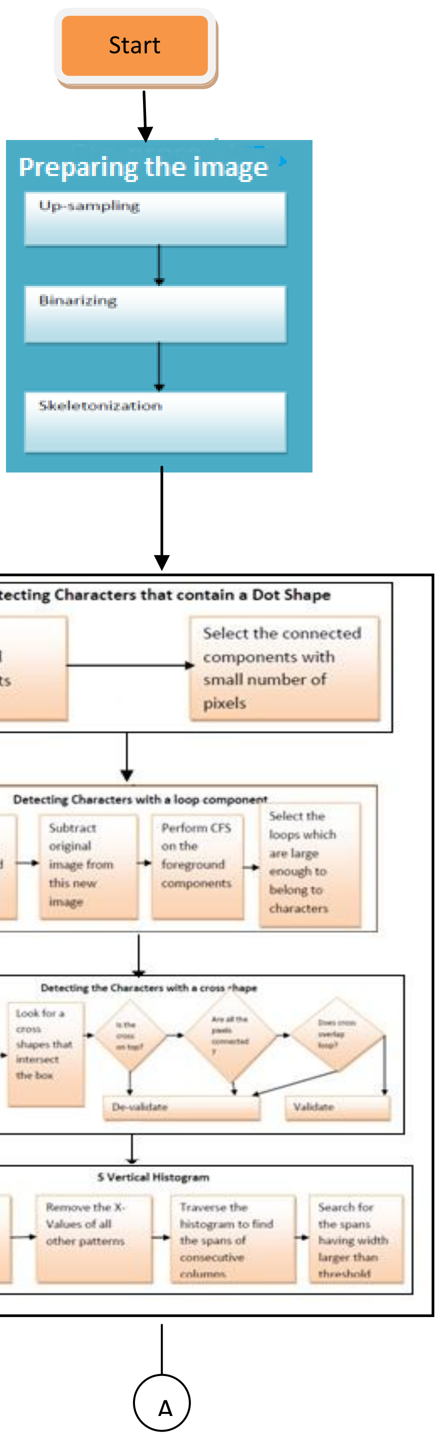
In the proposed method there would be four stages as shown in Figure 2.

Stage 1: Pre-processing the image by up-scaling, binarizing and skeletonization.

Stage 2: Identifying the character pattern.

Stage 3: Dividing the image into individual characters.

Stage 4: Recognizing the character.


Start
Preparing the image
Up-sampling
Binarizing
Skeletonization
Detecting Characters that contain a Dot Shape
CFS on foreground components
Select the connected components with small number of pixels
Detecting Characters with a loop component
Fill all the holes in the skeletonized image
Subtract original image from this new image
Perform CFS on the foreground components
Select the loops which are large enough to belong to characters
Detecting the Characters with a cross shape
Traverse the foreground component with an imaginary box
Look for a cross shapes that intersect the box
Is the cross on top?
Are all the pixels connected ?
Does cross overlap loop?
De-validate
Validate
S Vertical Histogram
Map the image against vertical histogram
Remove the X-Values of all other patterns
Traverse the histogram to find the spans of consecutive columns
Search for the spans having width larger than threshold
A

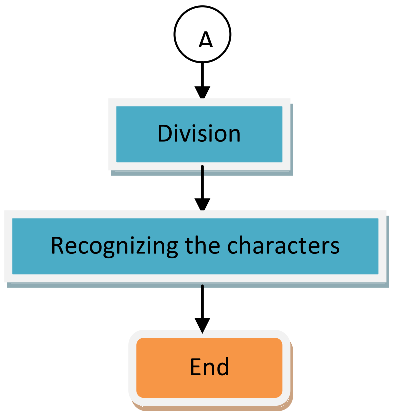


Figure 2: Flow of methodology

The below sections describe the detailed explanation of each phase of proposed methodology.

5.1 Preparing the image

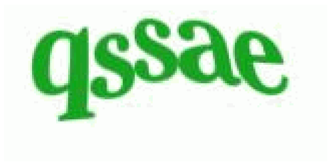


Figure 3: Sample Google CAPTCHA

First step is to scale the image. Scaling increases the pixel density of the image. Hence, smoothening the image and making it easier to find the patterns. The higher the factor of scaling, the smoother the image is but more expensive is the operation. The image's height is scaled to a fixed size to maintain constancy in the algorithm. A balance is maintained between speed and accuracy by up scaling the image to 3 times [4]

Once the image is scaled, it is binarized, i.e. all the pixels are either converted into black or white depending on whether they are above or below a particular threshold value. In case of Google CAPTCHAs, the pixels in the background are kept white and the text/characters in the image are converted into black.

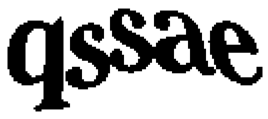


Figure 4: Binarized Image

The next step is skeletonization. An image is skeletonized to get rid of the different character thickness by converting the whole image to 1 pixel width. As discussed in the beginning, the width of the characters in the image are varied, skeletonizing the image solves this problem. Also, apart from this reason, thinning the image reduced the pixels and thus the complexity needs to be dealt with. It can be clearly observed that the image is not changed, the shape of the characters and the connections are as it is in the original image.

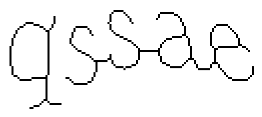


Figure 5: Skeletonized Image

5.2 Character Patterns

The Color Filling Segmentation (CFS) algorithm is widely used while categorizing the characters into the shapes aforementioned. The generic idea behind this algorithm is to call any connected component of an image as a chunk. For eg. in the above example, q makes one chunk and rest of the characters the other. CFS is the process in which each connected component is separated, which is essentially like filling each connected component with a different color, hence the name. [5]

Dot shaped characters: From CFS it is obvious that "dot" will be of different color because it certainly is not connected to a large chunk of characters. Thus, it can be that the component having small number of pixels will most likely be a dot shaped character.

Loop Shaped Characters: For this pattern an algorithm different from the one used in [4] is proposed although, it is derived from the same fundamental principles. First, fill all the holes in the skeletonized image. Then, subtract the original image from this new image. Again perform CFS on the foreground components and select the loops which are large enough to belong to a character. Loops can be classified into two categories:

i. **Character loops**: These are the loops which belong to the characters

ii. **Connection loops**: These are the loops that are created due to overlapping of characters.

Clearly, connection loops need to be removed from the loops we have found. These connection loops have the following properties which help to identify and remove them from the detected loop shaped. Firstly, the connection loops vertically overlap with other loop shapes. Secondly, they have a relatively large pixel count. All loops which satisfy both these properties can be removed and store the remaining loops.

Cross Shaped Characters: The basic algorithm is an extension of the one described in [4]. The unique characteristic of the cross shape that is being exploited here is that it has four sides, the top side, the left side, the right side and the bottom side. On drawing an imaginary box around such a cross shape, the shape will intersect the box exactly once at each of the four sides of the box. So on traversing the foreground components one by one with an imaginary box, and cross shapes that intersect the box as mentioned above can be found out. Then apply the three conditions to discard the invalid shapes discovered. Each shape discovered must satisfy all of the following to qualify as a valid cross shape:

- The cross shape must be in the top part of the foreground component being traversed.
- All the pixels within the imaginary box must be connected to each other.
- The cross shape must not vertically overlap with a loop shape.

S Shaped Characters: The algorithm used here is exactly the same as used in [4]. The unique shape characteristics exploited here are that the S shaped characters; also referred to as at S Vertical Histogram shape had three vertically overlapping strokes. The process is as follows. First map the image against a vertical histogram that gives us the number of pixels in each column of the array. From this histogram the X values of all other patterns/shapes can found to avoid confusion with other patterns/shapes. Then, traverse this histogram to find spans of consecutive columns that have three or more pixels. All the spans that have a width larger than specified threshold are recorded as an S Shaped Character.

Other Characters: Once all the above characters' patterns are detected, find the XValues remaining in the skeletonized image (after increasing the widths of the patterns detected so that parts of these characters do not appear in the segmented "Other Characters"), hence obtaining the remaining regions in the image. Now, segment may be done within these regions. In each region, separate the elements separated by an empty column of width>= 1 pixel, and store their locations.

According to [4], the dot shape has a unique shape and its detection method has only 1% false positive. Hence, it is detected first. The loop shape is detected next. Valid cross shapes shouldn't vertically overlap with loop shapes, hence they need to be detected after loop shapes. The reason for this is that sometimes, the connections between characters look like cross shapes. The S shaped characters require us to remove all the other patterns' XValues from the vertical histogram, and therefore needs to be detected last. Otherwise, vertically overlapping of characters occurs simultaneously, which occurs commonly due to italic characters. And for obvious reason, other characters are detected last.

5.3 Dividing the Image

Once each pattern shape is detected, each type of characters needs to be divided and cut according to their width. The cutting distance from the pattern is different for each character which depends on the unique properties of each pattern.

For cross pattern containing characters, the characters are very thin. So the cutting distance must be kept low.

For the dot shaped characters, which are practically recognized only by the dots, the cutting distance must be kept high as the dot need not necessarily be above the character i.e. "i" or "j".

Similarly, for the loop shaped characters, the cutting distance must be kept relatively high, as the loop shape is actually inside the character. So, to segment the whole character we need to cut from a larger distance. Cutting distance for S shaped characters follows a similar reasoning.

5.4 Recognizing the characters

Once the characters are divided and cut, they can individually be sent to an Optical character Recognizer (OCR) which gives good results for recognizing these individual characters.

6. CONCLUSION

Cracking the CAPTCHA has been a challenge to AI Research community, and till date there has been no system that has been able to achieve a 100% accuracy and efficiency rate.

Therefore, CAPTCHAs represent a win-win situation in the world of Artificial Intelligence in particular and Computer Science in general. The insolvability of CAPTCHAs by computers has allowed us to propose new methodologies to solve the Google CAPTCHA. On implementing these methodologies, if we find a way to break the CAPTCHA system, it would mean that we have solved a difficult AI problem and have found a loop hole in the current CAPTCHA technology used by the Google; which, in turn would help us create our own, more robust CAPTCHA.

7. REFERENCES

[1] Kurt Alfred Kluever. Evaluating the Usability and Security of a Video CAPTCHA. Master's thesis, Rochester Institute of Technology, Rochester, NY, August 2008

[2] G. W. Hart. To Decode Short Cryptograms. Communications of the ACM 37, 9 (1994), 102–108

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

[4] Ahmed S. El Ahmad, Jeff Yan and Mohamad Tayara, Robustness of Google CAPTCHAs, Newcastle, 2011

[5] Ahmed S. El Ahmad and Jeff Yan, A low-cost attack on a Microsoft CAPTCHA, Proc. Of 15th ACM Conference on Computer and Communications Security, 2008

[6] Chandavale A. Anjali, Sapkal A., and Jalnekar R., Algorithm to break visual CAPTCHA, 2009, ICETET Proceedings of the 2009 Second International Conference on Emerging Trends in Engineering and Technology, 258-262

[7] Ahmed S El Ahmad and Jeff Yan,2009, "CAPTCHA security: A case study", IEEE Security and Privacy

[8] Kun Fang, Zhan Bu and Zhen Y. Xia, A projection-based segmentation algorithm for breaking MSN and YAHOO CAPTCHAs, Artificial Intelligence and Computational Intelligence, 2012 Lecture Notes in Computer Science, 7530, 735-743