An Efficient Segmentation Algorithm for CAPTCHAs with Line Cluttering and Character Warping

Shih-Yu Huang, Yeuan-Kuen Lee, Graeme Bell and Zhan-he Ou,

Department of Computer Science and Information Engineering

Ming Chuan University

5 Teh-Ming Rd., Gwei Shan District, Taoyuan Country 333, Taiwan

phone: 886-3-3507001 ext 3439

e-mail: syhuang@mcu.edu.tw

Abstract

A CAPTCHA is a test designed to distinguish computer programs from human beings, in order to prevent the abuse of networked resources. Academic research into CAPTCHAs includes designing friendly and secure CAPTCHA systems and defeating existing CAPTCHA systems. Traditionally, defeating a CAPTCHA test requires two procedures: *segmentation* and *recognition*. Recent research shows that the problem of segmentation is much harder than recognition. In this paper, two new segmentation techniques called *projection* and *middle-axis point separation* are proposed for CAPTCHAs with line cluttering and character warping. Experimental results show the proposed techniques can achieve segmentation rates of about 75%.

Keywords: CAPTCHA, segmentation, recognition, Turing Test.

I. Introduction

As the internet increases in terms of size and in terms of available services, people gain more convenience, but also face new challenges. Free services on the internet may be abused by automated computer programs (often referred to as scripts or bots – here, we use bot). Such bots may be intended to broadcast junk emails, post advertisements, or ask servers to respond at a very high frequency. All of these forms of misuse will decrease the usefulness of internet services. To prevent such abuse, it is very important to design automatic systems that can differentiate between legitimate human users and unauthorized computer bots. The Completely Automated Public Turing Test to tell Computers and Humans Apart (CAPTCHA) was created to address these needs [1].

The purpose of a CAPTCHA is to distinguish between computer programs and humans automatically, through a computer-based test. The typical CAPTCHA user interface consists of two parts: a character image with noise, and an input textbox. The CAPTCHA system will ask the user to type the characters shown in the image. However, the CAPTCHA system may have warped the shape of the characters in the image, and added some arcs or lines to confuse and prevent automated computer recognition of the characters. Fig. 1 shows an example drawn from the MSN CAPTCHA system (circa January 2008). Put simply, an automated bot cannot pass such tests, as until recently there were no character recognition techniques that can understand what these altered characters are. In contrast, humans generally have much better natural abilities when faced with the task of character recognition in a noisy environment, so humans can usually pass these tests correctly without great inconvenience. For the example shown in Fig. 1, a human would be expected to type the answer 'D23Y6M9N' very easily. Therefore if a user can correctly respond to this kind of test, the system defended by the CAPTCHA will consider the user to be a real human. Otherwise, the user may be considered to be an illegal program, and may be denied access to the service.

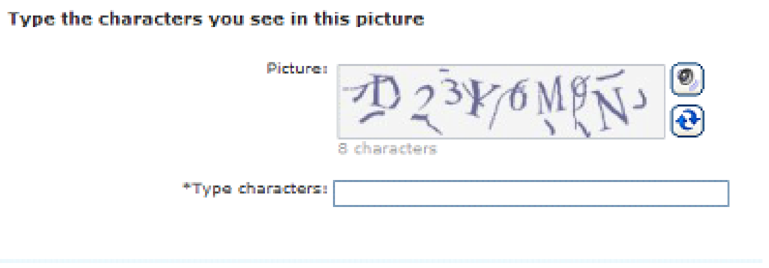


Fig. 1. An example of the MSN CAPTCHA system (c. Jan 2008).

Academic research into CAPTCHAs takes the form of a friendly 'arms race', with some researchers acting as 'malicious users' that try to attack and defeat the latest CAPTCHA systems automatically, e.g. [2]-[5], while other researchers seek to design new defensive CAPTCHA techniques in response to known or anticipated attacks [6]-[10]. When designing defensive CAPTCHA techniques, a CAPTCHA system designer should give consideration both to computer security and human-friendliness. In practice, balancing these two needs in opposition to one another is very difficult.

In considering the design principles of well-known CAPTCHA systems, we see that many well-known websites such as MSN, Yahoo, Google, Badongo, RapidShare and Youtube have been employing user interfaces similar to that of Fig. 1. Each website employs different heuristics to prevent malicious users. Badongo uses colored lines to clutter the image, and Youtube uses colored blocks; whereas RapidShare uses smaller colored characters as image noise to increase security. MSN and Yahoo do not use colored characters as noise. Instead, they use straight and curved lines as image clutter to confuse the attacking program. Although the security of these CAPTCHAs is increased by these heuristics, their human-friendliness is decreased. Accordingly, many researchers are trying to find other useful principles which will help human users to pass a CAPTCHA test more easily, while presenting new difficulties to automated programs. These principles include techniques such as the use of highly contrasting colors, or the use of fewer characters, to assist humans.

Turning now to the task of attacking CAPTCHAs. As mentioned in the previous paragraphs, there are a wide variety of CAPTCHA systems. It is hard to attack all CAPTCHA tests with a single type of segmentation algorithm. We assume that the CAPTCHAs being attacked have the following characteristics: a single-color picture, using warped characters, and straight or curved lines as image noise, to confuse the attacking program. The MSN and Yahoo systems (c. Jan 2008) are representative of this form of CAPTCHA, and are used as the basis for the main discussion in this paper. We observe that the task of attacking CAPTCHAs typically involves two procedures, *segmentation* and *recognition*. The segmentation procedure requires identification of the correct positions for each character, while the recognition procedure identifies which character is in each position. Recent research shows that *segmentation* is a much more difficult problem than *recognition* [4]. This is because machine learning algorithms can efficiently solve the recognition problem, but currently there is no effective general algorithm to solve the segmentation problem. In [5], Chellapilla and other researchers use an image opening and labeling technique to design a segmentation algorithm. When image noise components and character components have noticeably different widths, Chellapilla's technique is able to separate noise from characters effectively. However, when the difference in width 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. In 2008, Yan and Ahmad proposed a low-cost attack against a CAPTCHA designed by Microsoft. Their technique can efficiently separate noise from characters within the Microsoft CAPTCHA [12]. Nonetheless, its performance leaves room for improvement when facing CAPTCHA images with large warped characters, and long curved lines. This paper therefore proposes an efficient segmentation algorithm for attacking CAPTCHAs that can cope with these new cases as well as other recent CAPTCHA features. This paper introduces two new segmentation techniques, and outlines a new algorithm, that together make novel and useful contributions in the field of CAPTCHA analysis. The first of these techniques, *projection*, has also been recently introduced in [11].

The rest of this paper is organized as follows. Section II analyzes characteristics of CAPTCHAs with line cluttering and character warping. Section III illustrates Chellapilla's segmentation algorithm. Section IV presents the proposed segmentation scheme, including the *projection* technique, and the *middle-axis point separation* technique. Technical comparisons between the proposed technique and the algorithm presented by Yan and Ahmad. [12] are also given in Section IV. Section V covers experimental results. The paper's conclusions and suggestions for future works are presented in Section VI.

II. Analysis of CAPTCHAs

Straight lines, curved lines, and warped characters are widely used as image noise in CAPTCHA systems. MSN and Yahoo use these kinds of noise in their CAPTCHAs as the main techniques for confusing segmentation algorithms. The key differences between the two systems are that the MSN system uses more line noise than Yahoo's system, whereas Yahoo's system uses a higher degree of character warping. Here, the noisy lines are labeled as 'clutter'. Careful observation suggests at least eight properties that can be used to describe these clutter items. Here, the properties considered are color, intersection with non-clutter characters, size, curvature, length, width, relative position to a character, and angle. These properties allow categorization of clutter into nine types, as shown in Fig. 2.

A. Clutter

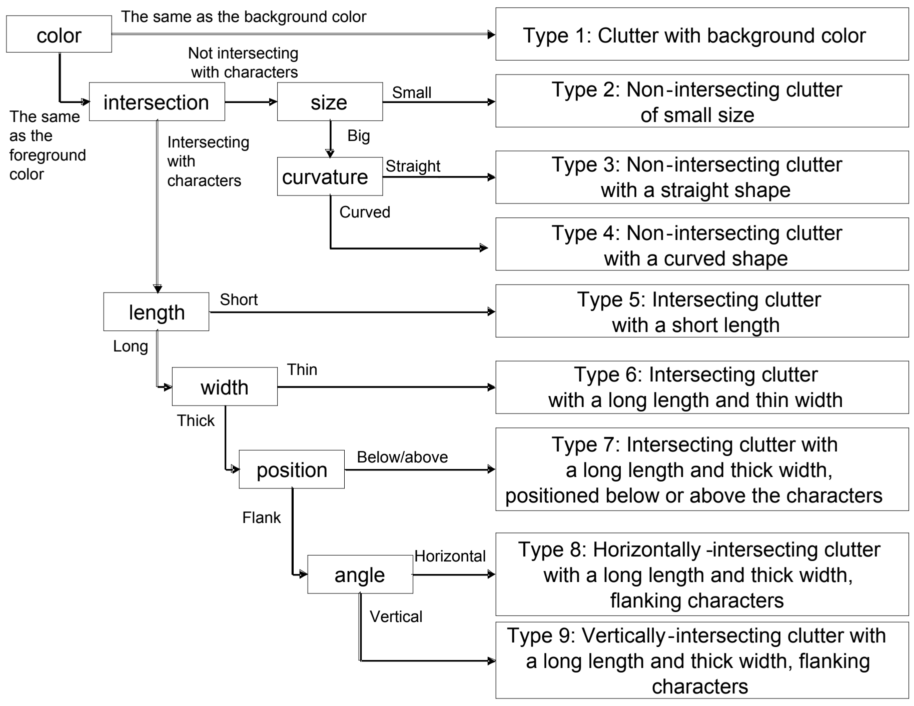

color
The same as the background color
Type 1: Clutter with background color
Not intersecting with characters
intersection
size
Small
Type 2: Non-intersecting clutter of small size
The same as the foreground color
Intersecting with characters
Big
curvature
Straight
Type 3: Non-intersecting clutter with a straight shape
Curved
Type 4: Non-intersecting clutter with a curved shape
length
Short
Type 5: Intersecting clutter with a short length
Long
width
Thin
Type 6: Intersecting clutter with a long length and thin width
Thick
position
Below/above
Type 7: Intersecting clutter with a long length and thick width, positioned below or above the characters
Flank
angle
Horizontal
Type 8: Horizontally -intersecting clutter with a long length and thick width, flanking characters
Vertical
Type 9: Vertically-intersecting clutter with a long length and thick width, flanking characters

Fig. 2: A categorization of nine different types of clutter.

Type 1: Clutter with background color. Clutter items in the MSN system can be divided into two groups by color. The first group consists of the clutter items drawn in the foreground color, whereas the second group is those drawn with the background color. When clutter items with the background color intersect the foreground characters, the characters will appear to be broken into separate pieces, as shown in

Fig. 3(a). Humans usually see these separated character fragments as a single character because of the shape constancy ability of the human visual system. However, a computer cannot identify these separated parts as a single character without some automated process for doing so. The second group of clutter items are those with the foreground color. This group of clutter items is now further subdivided, as follows.

Type 2: Non-intersecting clutter of small size. Type 2 clutter items have a small size and do not intersect with any characters, as shown in Fig. 3(b). They do not present a significant problem for attacking programs using existing techniques.

Type 3: Non-intersecting clutter with a straight shape. When a clutter item has an approximately straight shape, does not intersect other characters, and has a size similar to normal characters, it is classified as type 3, shown in Fig. 3(c).

Type 4: Non-intersecting clutter with a curved shape. This type of clutter item is similar to type 3, in that it does not intersect with other characters and has a character-like size, but it is not straight, as shown in Fig. 3(d).

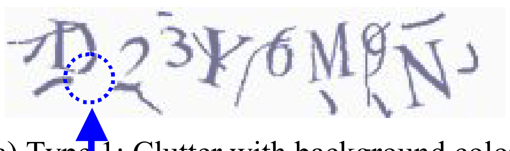
Type 5: Intersecting clutter with a short length. In practice, this type of clutter item has only a minimal impact upon the success rate of existing segmentation techniques. It is shown in Fig. 3(e).

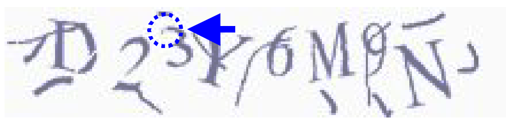
Type 6: Intersecting clutter with a long length and thin width. Regardless of width, when a clutter item has a long length, it may affect character detection; and worse, it may connect together several characters into a single item, in the view of a segmentation algorithm. These situations may cause segmentation algorithms to fail to identify characters successfully. However, long and thin clutter items can usually be deleted by the image opening process detailed in [4]. Examples of type 6 are shown in Fig. 3(f). The remaining problem is therefore detection/removal of thick clutter items.

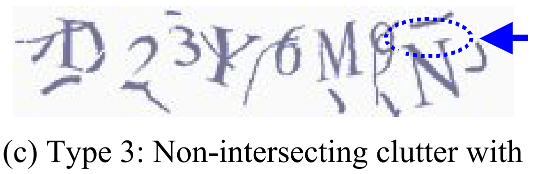
Type 7: Intersecting clutter with a long length and thick width, positioned below or above the characters. Type 7 is a form of thick clutter item where the position of the clutter is mostly below or above some characters, as shown in Fig. 3(g).

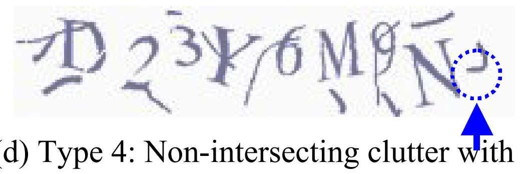
Type 8: Horizontally-intersecting clutter with a long length and thick width, flanking characters. This is a second form of thick clutter, consisting of horizontal clutter items whose position is to the left or right side of the characters, as shown in Fig. 3(h). The long length of this type of clutter means that it can connect several different characters easily.

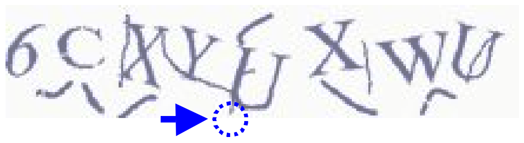
Type 9: Vertically-intersecting clutter with a long length and thick width, flanking characters. This is the last type of thick clutter item addressed in this paper. Where the type 9 clutter item is very close to the characters and is totally vertical, it is unlikely to influence the segmentation process, shown in Fig. 3(i).

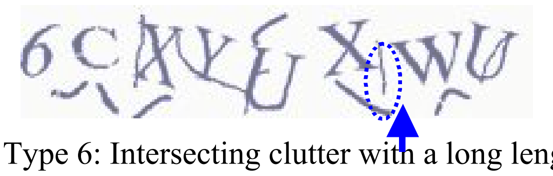

(a) Type 1: Clutter with background color.

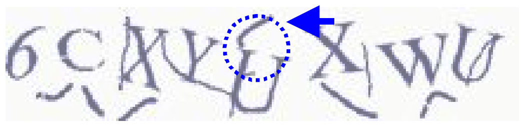

(b) Type 2: Non-intersecting clutter of small size.

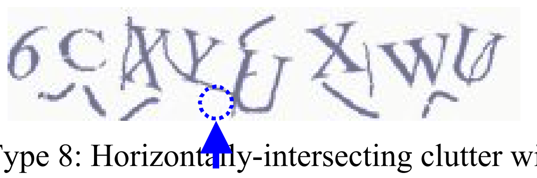

(c) Type 3: Non-intersecting clutter with a straight shape.


(d) Type 4: Non-intersecting clutter with a curved shape.


(e) Type 5: Intersecting clutter with a short length.


(f) Type 6: Intersecting clutter with a long length and thin width.


(g) Type 7: Intersecting clutter with a long length and thick width, below/above the characters.


(h) Type 8: Horizontally-intersecting clutter with a long length and thick width, to the side of a character.

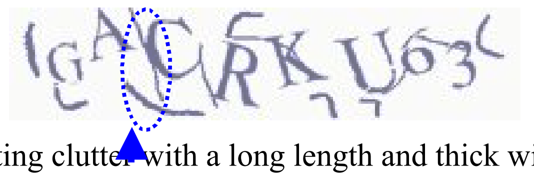
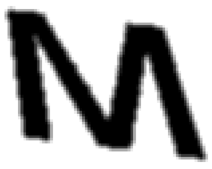

(i) Type 9: Vertically-intersecting clutter with a long length and thick width, to the side of a character.

Fig. 3: Examples of nine types of clutter.

B. Character warping

Two forms of character warping are commonly employed in CAPTCHAs: global warping and local warping. Global warping is a character-level deformation. After the global warping method has been applied, the whole character is distorted, so that it is not recognized by character template matching algorithms. Fig. 4 shows an example of global warping. Fig.4(a) is the original character, and Fig. 4(b) is Fig. 4(a) after a global warping process has been applied. Unlike global warping, local warping is based on generating partial distortions across a character. The character does not possess a single general change after the local warping. Instead, it has many irregular ripples and waves that prevent character recognition by feature-based algorithms. An example is shown in Fig. 4(c).

(a) Original character.


(b) Global warping.

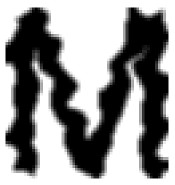

(c) Local warping.

Fig. 4: Examples of character warping.

Warping methods do not significantly increase the security of a CAPTCHA by themselves, as there are techniques to address different forms of warping approach. However, warping does present many difficult problems for attacks on CAPTCHAs when combined with a cluttering method. For example, when clutter lies between two warped characters, it becomes much harder to find it and remove it.

III. Chellapilla's Algorithm

In [4], Chellapilla et al. describe why the major problem in defeating CAPTCHAs lies in the task of segmenting the characters in the image. The paper also proposes an algorithm to segment images produced by several kinds of CAPTCHA systems. The algorithm design includes three phases - preprocessing, image opening, and labeling - in order to defeat CAPTCHA systems with line clutter and character warping. A behavior diagram is shown in Fig. 5.

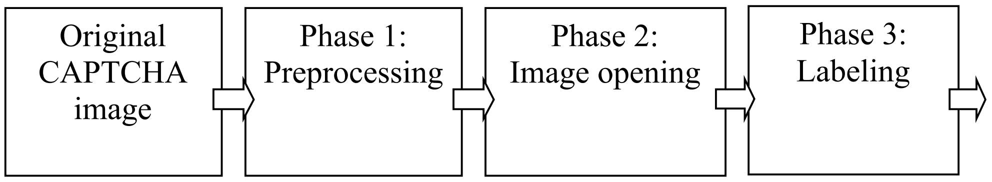
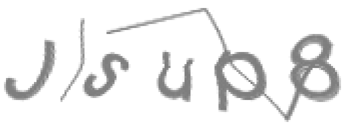


Fig. 5: Behavior diagram for Chellapilla's CAPTCHA segmentation algorithm.

The preprocessing phase involves *thresholding* and *up-sampling*. Initially, any grayscale in the original image is converted, so that it becomes a black and white image. Afterwards, the image is enlarged. Next, *image opening* is the key step that allows later segmentation of the characters in CAPTCHA images. In this phase, the preprocessed image will go through an *erosion* process several times and will then be *dilated* several times. Erosion will erase the character borders one pixel per time irreversibly, whereas dilation will increase the border size one pixel per time. Notice that these operations do not 'cancel each other out', as thin items will be removed completely and will not reappear after dilation. In terms of CAPTCHAs, we see that thin clutter items are deleted by the erosion process, so that they no longer appear following the dilation process. Therefore, some items of clutter have been deleted. Next, the *labeling* phase finds all of the connected components in the image, and considers the largest ones to be characters. Since this final phase only outputs the largest discrete items in the image as its result, any small isolated components will be considered to be clutter or noise, and will be eliminated in this phase.

This algorithm is useful whenever clutter items are of thinner width than characters - in other words, this helps to remove any type 6 clutter items. Fig. 6 shows a successful example of this technique. However, when the width of the clutter items is similar to the width of the characters, this algorithm produces errors. Since the algorithm cannot recognize the difference between characters and clutter items of similar width, it may categorize the clutter items as character data. Consequently the clutter items will not be deleted. An example is shown in Fig. 7(a) and Fig. 7(b), where 'S' and '8' are still connected, and 'G' and 'H' have a similar problem. Another possible problem occurs when the algorithm mistakenly identifies characters (or parts of characters) as being clutter items, and removes all or part of some characters. An example is shown in Fig. 7(c) and Fig. 7(d), where ‘5’ is split into two components.


(a) Original image.

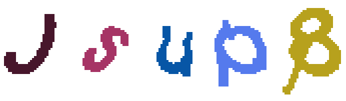
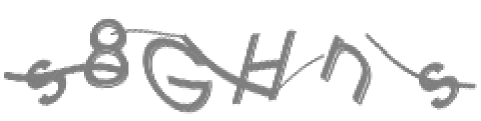
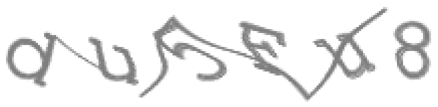
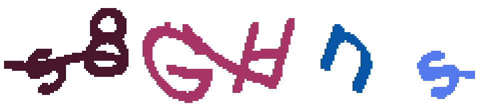

(b) The final result.

Fig. 6: Example showing Chellapilla’s algorithm operating successfully.


(a) Original image.


(c) Original image.


(b) Some clutter items cannot be erased.

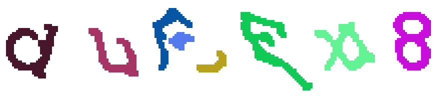

(d) Some characters become broken.

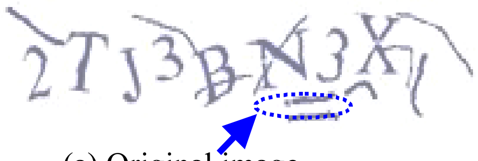
Fig. 7: Examples showing some problems with Chellapilla’s algorithm.

IV. Proposed Segmentation Algorithm

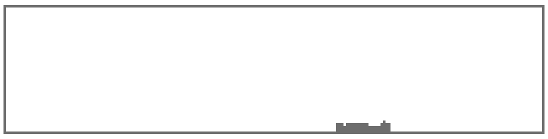
Chellapilla et al. gave the research community an effective way to address the recognition problem, but their segmentation algorithm does not represent a complete solution. This paper will therefore now propose two novel techniques - *projection* and *middle-axis point separation* - that are intended to improve the success rate of segmentation, and which yield a more effective segmentation algorithm.

A. Projection.

The projection technique in this paper is based upon the idea of projecting the image data onto the X-axis. In practice, this is implemented by summing the number of pixels in each column of the Y-axis of the image. This technique addresses problems caused by type 3, type 8 and type 9 clutter items. Fig. 8(a) shows an example of a type 3 clutter item. Type 3 clutter items are those which do not intersect with other characters, and which are a connected component by themselves, so it is possible to use the characteristics of these components to separate them from real characters. Notice that the projections of these clutter items onto the X-axis appear smaller and flatter than a normal character's projection onto the X-axis. This is shown in Fig. 8(b). The projection in the X-axis will tend to appear large and unstable, when a component represents a character rather than an item of clutter. This is shown in Fig. 9(a) and Fig. 9(b). Therefore, by computing a component's projection value and its variance, it is possible to differentiate between components that are clutter, and components that are characters.

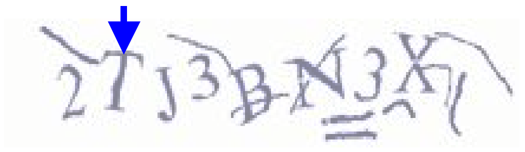


(a) Original image.

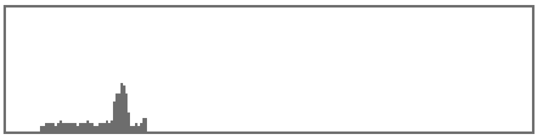


(b) The projection in the X-axis for the highlighted component.

Fig. 8: Type 3 clutter and its projection.



(a) Original image.

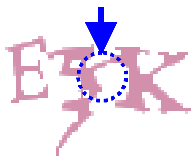


(b) The projection in the X-axis for the highlighted component.

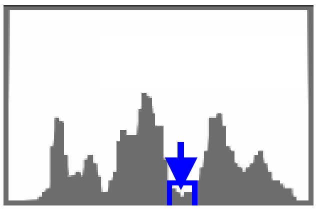
Fig. 9: A character and its projection.

Type 8 clutter items intersect characters and form part of a larger component, shown in Fig. 10(a). This type of clutter also has a smooth and small appearance when projected into the X-axis, as shown in Fig. 10(b). Therefore, it is possible to use the projection technique to find out the position of the clutter items within a large component, making it more straightforward to clean up the component. When two or more characters are connected by this form of clutter, they can be effectively split up by deleting these clutter items.

This paper uses a sliding window approach to detect type 8 clutter, because this type of clutter has a smaller projection size than a normal character over a small part of the image. In other words, the X-axis projection value of these clutter items will be smaller than some threshold for a part of the image, so it is possible to use a sliding window to check the projection value continually. When all the projection values within the sliding window are smaller than the threshold, the algorithm marks the position as containing a type 8 clutter item, so that it may be erased.



(a) An example of type 8 clutter.

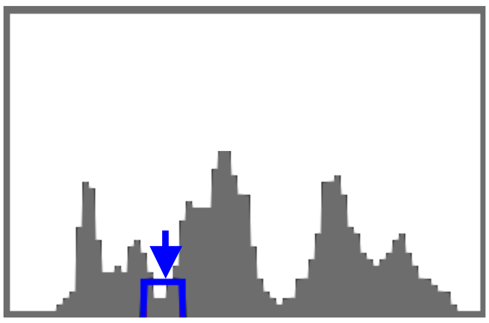


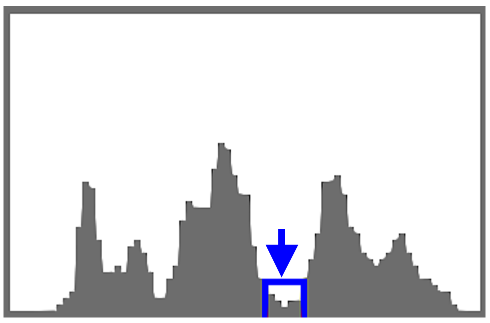
(b) Projection image, sliding window and threshold.

Fig. 10: Type 8 clutter and its projection image.

Fig. 11 gives an example of the operation of the sliding window approach with type 8 clutter. Suppose the width of the sliding window is 5, and the threshold is also 5. When the sliding window moves to the edge of the ‘E’ and the ‘5’, the projection values in the sliding window are not all smaller than the threshold, so no action will be taken at this position, shown in Fig. 11(b). When the sliding window moves to the edge of the ‘5’ and the ‘K’, all of the X-axis projection values are smaller than the threshold, so the algorithm will mark this place as containing a type 8 clutter item, and clean it from the image. After the cleaning process, the connection between the ‘5’ and the ‘K’ characters has been removed, and so these characters are split into separate components.

(a) Original image.


(b) Sliding window between the edge of the 'E' and the '5'.


(c) Sliding window between the edge of the '5'and the 'K'.

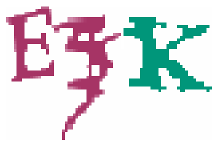

(d) Segmentation result.

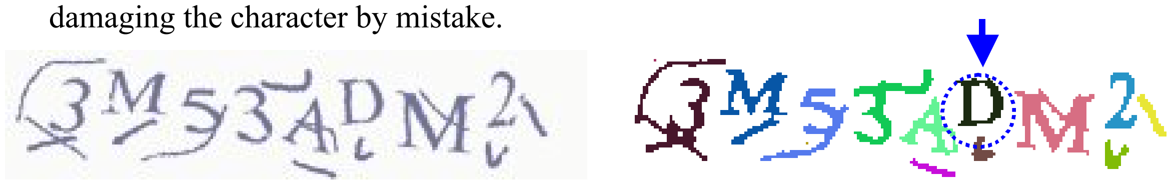
Fig. 11: An example of the sliding window technique with type 8 clutter.

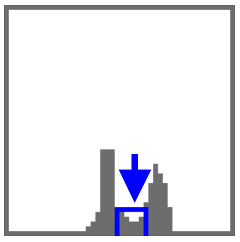
The performance of the proposed sliding window approach is strongly dependant on the value of the threshold, denoted as *T*, and the size of the sliding window, denoted as *S*. A smaller value of *T* results in difficulty when erasing items of clutter. In contrast, clutter items can be easily erased when *T* has a larger value. However, some characters could become damaged by an excessive amount of erasure. Variation in the size of the sliding window produces similar phenomena. Type 8 clutter is easily erased when *S* has a small value. However, some characters with thin strokes are also broken. In contrast, clutter can not be detected when *S* has a large value.

Every CAPTCHA image may have some variation in its clutter item size and inter-character white space size, so it may not be appropriate to always use fixed values for *T* and *S*. To address this challenge, an adaptive heuristic was developed to set values for *T* and *S*, according to the content of each CAPTCHA image. An ideal value for *T* is of course equal to the height of clutter items in the image. Here, the height of clutter items throughout the image was found to be usefully approximated by the mode of the projection height values for type 3 clutter. We calculate this as follows. After the initial phase of labeling, type 3 clutter items become distinct easily-recognized components. We project these components individually and calculate the most commonly occurring projection height value within them. We use this value as the value for *T* within the sliding window approach. Similarly, a reasonable value for *S* would be the normal size of white space gap between adjacent characters in the image. After the labeling phase, we have a number of distinct components. The size of the white space between these distinct components was found to give a useful indication of the size of white space between characters within each component. We can therefore provide a unique value for *S* for each image following the labeling phase, by calculating the mean size in pixels of the white space gap between components.

Of course, not all labeled components require the use of the sliding window approach - it is only intended to erase type 8 clutters from components with two or more characters. This approach is therefore only activated when the width of a component (in pixels) is larger than twice the width of the smallest single component found in the image. This heuristic was chosen following early experimentation and was found to be very effective.

Finally, we observe that the projection values of some characters, such as ‘0’, ‘O’, ‘D’, ‘8’, or ‘B’, can be smaller than the threshold continually over a region of the image, depending on the value *T* chosen for the threshold. For example, in Fig. 12(a), the projection values of the character ‘D’ are smaller than the threshold, resulting in this character being damaged by the decisions made by the projection method. Fortunately, these characters have a useful property – they have closed regions within them, containing the background color. This property can be utilized to avoid damaging the character by mistake.





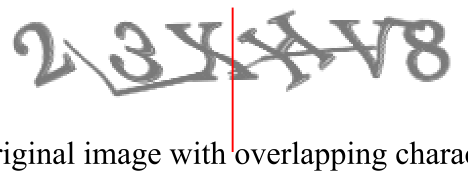
(c) Sliding window, with the X-axis projection for the highlighted component.

Fig. 12: An example of an error during the projection technique.

B. Middle-axis point separation

Some clutter items may not be erased by using the projection technique. Fig. 13 shows an example of two characters which cannot be split by the projection technique. Since these two characters are too close to allow the sliding window projection value to lie below the threshold throughout the entire window, they can not be separated by the sliding window projection technique. Although the projection value does not help with the task of deleting these clutter items, it is still possible to observe the existence of clutter items because of the unusual component length.

In the CAPTCHA systems being studied, characters will never overlap with each other, so it is possible to use the spaces between them (which are in the background color - here, white) to separate the characters. This paper therefore utilizes this property and proposes another technique named *middle-axis point separation* to clean up the remaining clutter, or at least reduce its ability to affect the characters. The middle-axis points are the white background pixels lying centrally between two disconnected black foreground pixels, determined around the Y-axis. These points will be connected together to form several candidate *cutting lines* in different places, shown in Fig. 14(b). Fig. 14(c) shows a set of correct cutting lines determined a-priori and manually – the algorithm should produce output close to this ideal set of cutting lines.


(a) Original image with overlapping characters.

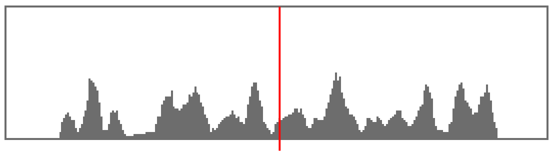
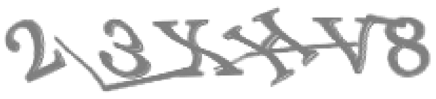
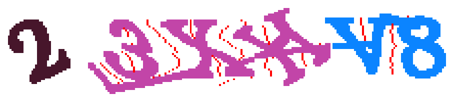

(b) The x-axis projection of the image.

Fig. 13: An example of projection failing.


(a) Original image.


(b) Candidate cutting lines.

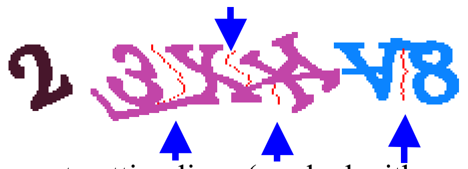

(c) Correct cutting lines (marked with arrows).

Fig. 14: An example of middle-axis point separation.

To find these cutting lines, the algorithm must estimate how many characters are in the component, in order to decide how many cutting lines are needed. A simple solution to this first problem is to statistically analyze character groups to determine the typical component length. The length of a component is measured in terms of pixels along its corresponding projection. In Table 1 below, calculations have been made for 100 MSN and 100 Yahoo CAPTCHA images, giving the statistics associated with different numbers of characters.

Table 1: The relationship between the number of characters in a component, and the typical component length.

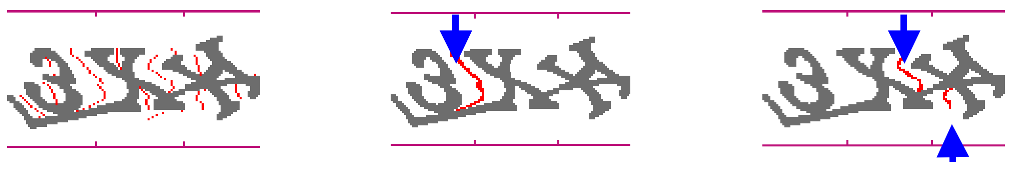
	Yahoo		MSN	
Number of characters	Average length (pixels)	Minimum/Maximum length (pixels)	Average length (pixels)	Minimum/Maximum length (pixels)
One character	29.07	8~50	24.06	7~42
Two characters	60.71	43~85	46.27	38~59

Three characters	98.25	91~126	64.76	55~79

Using Table 1, it is possible to guess the number of characters in a component. For example, in the Yahoo CAPTCHA system, the longest length of a component with one character is 50 pixels, and the shortest length of a component with two characters is 43 pixels. An estimation may be made that (50+43)/2 = 46.5 pixels should represent a suitable threshold for heuristic determination that a component has only one character. In the same way, a heuristic threshold can be assigned for components with two characters, as (85+91)/2 = 88. Therefore, when the length of a component is between 46.5 and 88 pixels long, the algorithm will regard the component as most likely consisting of two characters. The algorithm considers a component to consist of three characters, if its length is more than 88 pixels. Similarly, in the MSN CAPTCHA system, thresholds can be determined as 40 and 57 pixels respectively. These measurements must be determined for any other CAPTCHA system that is being attacked with this technique.

For example, in the Yahoo CAPTCHA image shown in Fig. 15(a), the selected component has a length of 126 pixels. Since 126 > 88, the algorithm treats it as having 3 characters and therefore requiring 2 sets of cutting lines to separate the 3 characters. If a candidate cutting line lies close to an average-case boundary position, such as (126/3 = 42) or (126*2/3 = 84), it is more likely to represent a correct cutting line. Fig. 15(b) gives a single complete cutting line located close to pixel 42.

Notice that an ideal cutting line may be interrupted by a horizontal clutter item or horizontal part of a nearby distorted character in practice. Therefore, if two cutting lines are excellent candidates, and if their orientation is vertical, and at similar x-axis positions, it is likely that both lines are necessary cutting lines, and they should be connected. This overcomes the case where an ideal cutting line is interrupted by a horizontal component such as a clutter item. Fig. 15(c) shows a set of cutting lines placed close to pixel 84. Using these cutting lines, we can split the component into different parts, as shown in Fig. 16.



(a) An example component. (b) A single complete cutting line. (c) A pair of cutting lines.

Fig. 15: An example of middle-axis point separation.

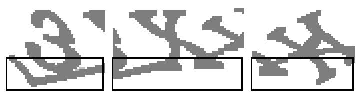


Fig. 16: The segmentation result for middle-axis point separation. The right hand boundary of the 'X' character is determined from two cutting lines separated by a horizontal component, whereas the left boundary is a complete cutting line

C. Proposed CAPTCHA segmentation algorithm.

This paper now proposes a segmentation algorithm for CAPTCHAs with line cluttering and character warping. This algorithm is based on Chellapillas et al.'s algorithm and has five phases: Preprocessing, Image Opening, Labeling, Component Splitting and Character Extracting. A behavior diagram is shown in Fig. 17.

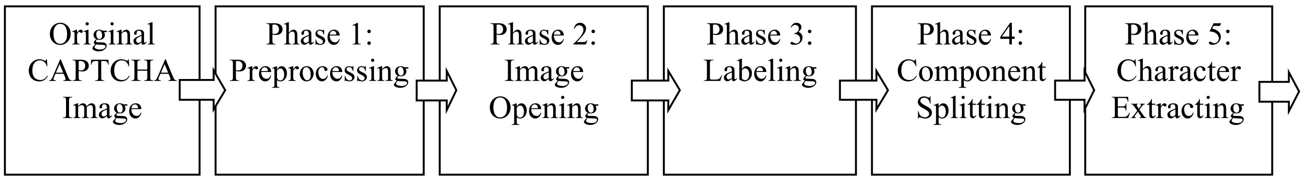


Fig. 17: Behavior diagram for the proposed CAPTCHA segmentation algorithm.

The algorithm proposed here is based upon Chellapillas et al.'s algorithm, and has a similar process for the first three phases. An important difference appears in the preprocessing phase. To prevent the mistake highlighted earlier, when the projection technique fails to operate with '0', 'O', 'D', '8' and 'B', the algorithm will detect closed regions in the preprocessing phase. It begins by computing all of the background-colored connected components first. The background-colored connected components which do not belong to a closed region are combined together, and become the largest connected background-colored component in the picture. The algorithm marks the connected background-colored components that are smaller than this largest component, as closed regions, which will not be considered by the sliding window technique later in segmentation. A simple example is shown in Fig. 18.

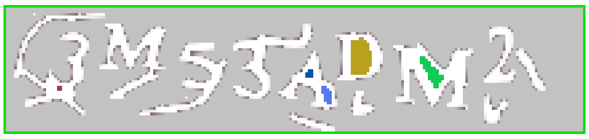


Fig. 18: Closed regions identified during preprocessing.

The second and third phases are image opening and labeling, respectively, as per Chellapilla's algorithm. The fourth phase, the component splitting phase, includes the proposed projection and middle-axis point separation techniques. The components that are not already fully separated can be separated by these techniques, since image opening will have erased the thin clutter items of type 6, and projection can solve problems caused by type 3, type 8 and type 9 clutter items. Any component which cannot be separated by these earlier operations, will then be split by middle-axis point separation.

After all of these phases are complete, the original image will have become separated into many different connected components. In Fig. 19, different shading highlights each of the different connected components that were identified. In this example, the MSN CAPTCHA has only 8 characters, but the image is broken into 15 discrete connected components. In the final phase, the *character extracting phase*, the algorithm deletes redundant components, and outputs the location of the characters. It is known that genuine character components have the biggest and most highly variable projection values in the X-axis, so the algorithm now erases any components which have small and almost flat projections, as well as type 2 clutter items. The remaining components are sorted by their size; and the algorithm outputs the largest 8 components (for MSN-type CAPTCHAs), or the largest 6 components (for Yahoo-type CAPTCHAs, which typically have 5-6 components). However, if two characters are identified near the same horizontal position, the character with the largest area is kept, and the character with the smallest area is discarded. In Fig. 19(b), this would cause the large clutter item above the letter 'G' to be automatically discarded.

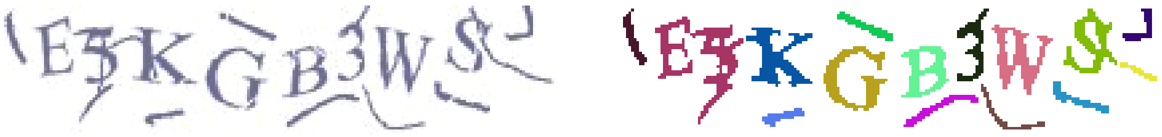

(a) Original image. (b) Image after the fourth phase.

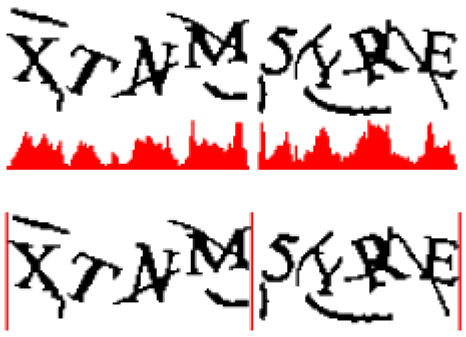
Fig. 19: An example showing different connected components.

D. Comparisons between the proposed algorithm and Yan's algorithm.

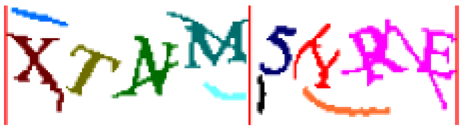
In 2008, Yan and Ahmad proposed a low-cost attack on a CAPTCHA designed by Microsoft. Yan's algorithm includes seven phases: *binarization*, *fixing broken characters*, *vertical segmentation*, *color filling segmentation*, *thick arc removal*, *locating connected characters*, and *segment connected characters*. The initial two phases of Yan's algorithm and this paper's algorithm are quite similar. *Binarization*, i.e. conversion of the original image into a black and white version, is similar to the preprocessing phase of the proposed algorithm. *Fixing broken characters* is similar to the proposed image opening phase, i.e. its purpose is to connect any damaged characters caused by type 1 clutter items.

The goals and principles of the third, fourth, and fifth phases of Yan's algorithm are also similar to the third and fourth phases of the proposed algorithm. They all utilize projection information to segment CAPTCHA images and remove clutter. In Yan's algorithm, a CAPTCHA image is divided into multiple chunks if there are zero valued projections during the vertical segmentation phase. Fig. 20(a) of this paper shows an example where two chunks are obtained. Next, the connected components of every chunk are extracted by the color filling segmentation phase (this corresponds to the proposed labeling phase of this paper). Fig. 20(b) shows the corresponding connected components produced in this paper's example. Finally, the thick arc removal phase utilizes four properties of clutter items: pixel count, location, shape, and interplay between shape and location, in order to remove thick clutter. Fig. 20(c) shows the results after the fifth phase of Yan's algorithm, where 'R' and 'E' are still connected.

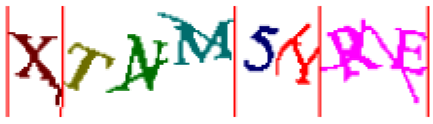
Clearly, there are some high-level similarities between Yan's algorithm and this paper's proposed algorithm, but the detail of each algorithm is different, resulting in quite different performance. A significant difference is the running order of the projection and labeling phases. In this paper's proposed algorithm, labeling is performed before projection. The benefit of this approach is that a CAPTCHA image can be quickly segmented into many small components. Reusing the earlier example, Fig. 20 (d) shows the results of labeling, where the CAPTCHA image is segmented into 10 components. Projection is then performed for every component. Type 3 clutter items can be easily removed by the proposed properties of projection. Type 8 clutter items can be extracted by the proposed sliding window approach. Fig. 20 (e) gives the corresponding results for the previous example, where 'R' and 'E' are extracted separately.



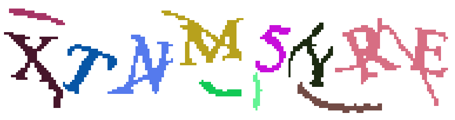
(a) Results of the vertical segmentation phase of Yan's algorithm.



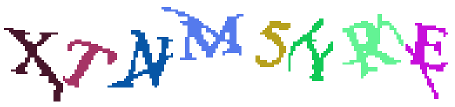
(b) Results of the color filling segmentation phase in Yan's algorithm.



(c) Results of the thick arc removal phase in Yan's algorithm.



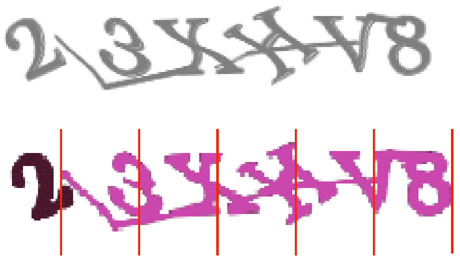
(d) Results of the labeling phase of the proposed algorithm.



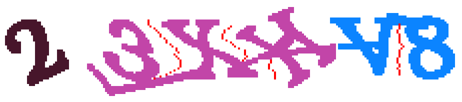
(e) Results of the projection phase of the proposed algorithm.

Fig. 20: An example comparing Yan's algorithm and the proposed algorithm.

The last two phases of Yan's algorithm are: *locating connected characters*, and *segment connected characters*. They share similarities with the proposed middle-axis point separation and character extracting techniques of this paper. Both approaches utilize statistical data about the number of chunks, the width of each chunk, and the number of objects in each chunk, in order to guess which chunk contains connected characters and also the number of characters present. The major difference between them is that an 'even cut' approach is employed in Yan's algorithm to reduce the computational cost; whereas a more sophisticated 'middle point cut' is utilized in the proposed algorithm to improve the accuracy of segmentation. Fig. 21(a) shows an example of the results of the Yan's algorithm, where '3', 'X', 'H', 'V', and '8' are left in a single chunk as there is no zero projection value position between them. 'Even cut' fails in this case. However, these characters are correctly extracted by the proposed algorithm of this paper, as shown in Fig. 21(b), where '3XH' and 'V8' are already separated by the proposed sliding window approach.



(a) Failure of segmentation resulting from Yan's algorithm.



(b) Successful segmentation produced by the proposed algorithm.

Fig. 21: A second example comparing Yan's algorithm and the proposed algorithm.

V. Experimental Results

In this section, experimental results are presented for Chellapilla's algorithm, and the proposed algorithm. These algorithms were applied to sample images from the MSN and Yahoo CAPTCHA systems collected during 2007. 100 MSN and 100 Yahoo CAPTCHA images were tested for each system, with 800 characters in the MSN CAPTCHAs and 525 characters in the Yahoo CAPTCHAs. Two kinds of segmentation rate, *SRA* and *SRB*, are used here to demonstrate the performance of these algorithms. SRA is the segmentation rate based upon the numbers of characters in each of the different images. For example, if an algorithm can segment 600 characters from 100 images of MSN CAPTCHAs, the segmentation rate will be 600/800 = 0.75, or 75%. SRB is the segmentation rate in terms of the number of images whose characters were all correctly segmented. SRB rises only when every character in a complete image is fully and correctly segmented.

We first analyze the effect of the proposed adaptive mechanism for setting *T* and *S* in the proposed algorithm. For comparison, we also implement the algorithm with fixed *T* and *S*, where *T* and *S* range from 1 pixel to 7 pixels. Table 2 shows the SRA results of the algorithm with fixed *T* and *S* and adaptive *T* and *S*. For the algorithm with fixed *T* and *S*, the optimal SRA is 77.88% when *T*=3 and *S*=3 in MSN images and the optimal SRA is 80.57% when *T*=4 and *S*=7 in Yahoo images. The segmentation rates are poorer when the algorithm uses a larger *T* and/or a smaller *S* since many characters are damaged in that case. It appears that the effect of *T* is more important than that of *S*. Once the value of *T* is correctly set, the segmentation rates are acceptable even with a poor value of *S*. For the example of *T*=3 with MSN images, all of the SRAs resulting from different values of *S* are higher than 73.75%. However, when the value of *T* isn't properly set, the SRAs are likely to be terrible. For example, when *T*=7 in MSN and Yahoo images, the SRAs are sometimes as low as 42.13% and 52.57% respectively for MSN and Yahoo images. This indicates that it is difficult to set fixed values of *T* and *S* for all CAPTCHA images. This problem is solved here by the proposed adaptive *T* and *S* mechanism. The SRAs of the adaptive algorithm are 76.00% and 79.05% for MSN and Yahoo images, respectively. These success rates are quite similar to the optimal results obtained by the algorithm with fixed *T* and *S*. Fig. 22 shows the details of the *T* and *S* settings as the algorithm executes. The average values of *T* and *S* under the adaptive mechanism are 3.7 and 3.14 in MSN images, and 4.47 and 6.18 in Yahoo images. Again, this is near to the optimal values that were discovered by testing the algorithm with fixed *T* and *S*.

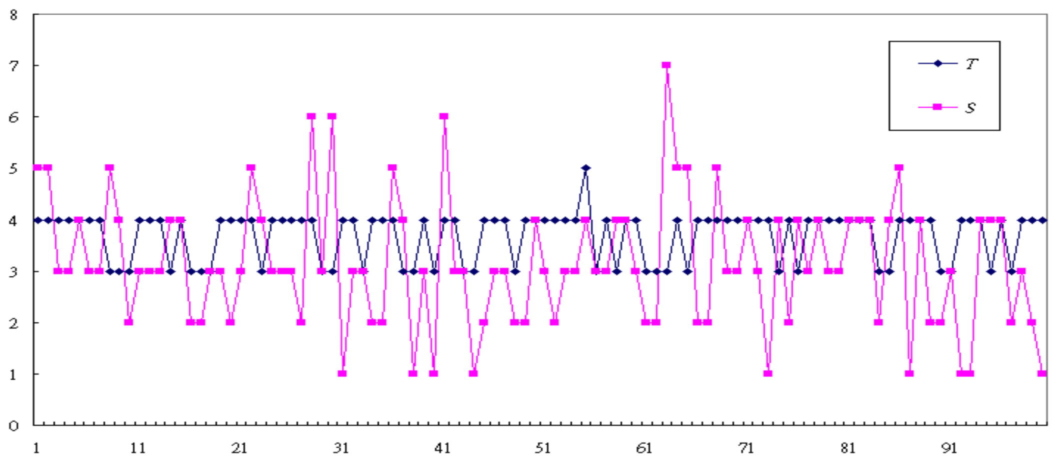
Table 2: SRA results for the proposed algorithm with fixed *T* and *S* and adaptive *T* and *S*.

(a) MSN images

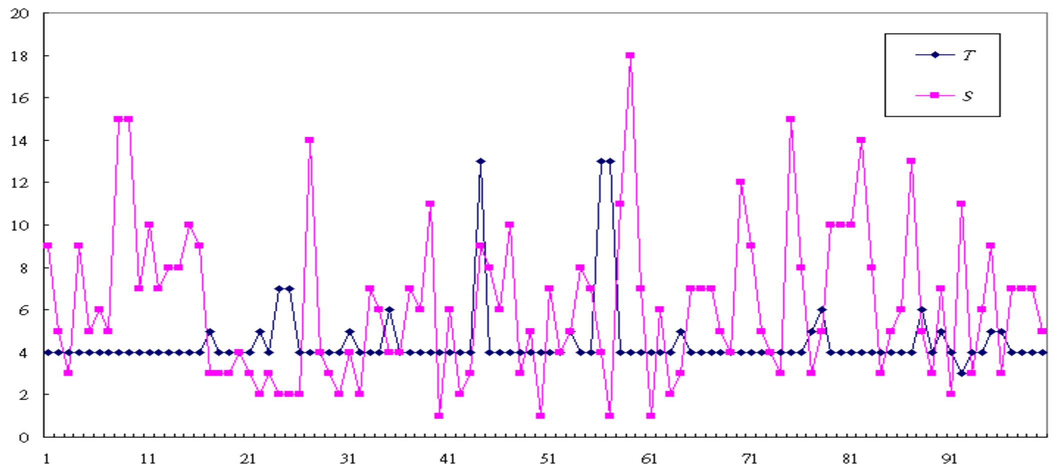
Fixed *T* and *S*								Adaptive *T* and *S*
	S=1	S=2	S=3	S=4	S=5	S=6	S=7	
T=1	70.38%	70.13%	70.00%	70.00%	70.00%	70.00%	70.00%	
T=2	75.00%	74.25%	74.13%	72.38%	71.38%	71.25%	70.88%	
T=3	73.75%	76.50%	77.88%	77.38%	75.38%	75.13%	74.38%	
T=4	64.38%	69.13%	73.25%	75.88%	75.25%	76.13%	76.00%	76.00%
T=5	57.50%	63.00%	68.50%	71.88%	72.75%	73.50%	73.38%	
T=6	50.00%	56.63%	62.63%	65.13%	69.75%	70.88%	72.13%	
T=7	42.13%	50.13%	55.00%	61.13%	64.63%	65.50%	67.63%	

(b) Yahoo images

Fixed *T* and *S*								Adaptive *T* and *S*
	S=1	S=2	S=3	S=4	S=5	S=6	S=7	
T=1	77.33%	77.33%	77.33%	77.33%	77.33%	77.33%	77.33%	
T=2	77.33%	77.33%	77.33%	77.33%	77.33%	77.33%	77.33%	
T=3	77.33%	78.10%	79.05%	78.29%	78.48%	78.29%	78.10%	
T=4	71.81%	75.43%	78.48%	78.67%	79.24%	80.19%	80.57%	79.05%
T=5	64.38%	69.33%	73.14%	75.43%	78.29%	79.81%	80.38%	
T=6	58.10%	64.57%	67.62%	69.52%	74.48%	75.62%	76.95%	
T=7	52.57%	57.52%	63.24%	65.52%	70.10%	72.38%	74.67%	



(a) Variation in T and S during execution of the adaptive method with MSN images.



(b) Variation in T and S during execution of the adaptive method with Yahoo images.

Fig. 22. Variation in T and S during execution with the adaptive mechanism proposed for this paper's algorithm.

This paragraph now compares the results from the proposed algorithm against those from Chellapilla's algorithm. In order to isolate and examine the effects of the proposed projection heuristic, a 'projection-only' algorithm was also implemented by disabling the middle-axis point separation phase. Table 3 shows the SRA and SRB results. The SRA results for the projection-only algorithm were higher than Chellapilla's algorithm by 6.48% for the Yahoo CAPTCHA system. Similarly, when attacking the MSN CAPTCHA system, the algorithm yields a 14% higher SRA result. For the SRB results, the projection-only algorithm's results were also higher than Chellapilla's algorithm's results by 6% and 9% for the Yahoo and MSN CAPTCHA system, respectively. Fig. 23 shows a set of examples demonstrating behavior for both Chellapilla's algorithm and the projection-only algorithm. In the Yahoo CAPTCHA in Fig. 23, Chellapilla's algorithm leaves a component which is made from clutter items, whereas the projection-only algorithm in this paper totally separates all the characters without leaving any clutter. In the MSN CAPTCHA in Fig. 23, Chellapilla's algorithm cannot split the '5' and the 'K', and leaves many items of clutter, but the projection-only algorithm can separate the '5' and the 'K' and can also remove most of the clutter items.

Finally, the results shown in Table 3 indicate the performance of the complete proposed algorithm which includes projection and middle-axis point separation. It was found that the segmentation rate is further increased by middle-axis point separation. With Yahoo CAPTCHAs, the SRA increased to 79.05%, compared against 67.50% for the projection-only algorithm. With MSN CAPTCHAs, the SRA increased to 76.00%, compared to 55.13% for the projection-only algorithm. For the SRB results, the complete proposed algorithm's results were also higher than the projection-only algorithm's results by 11% and 6% for the Yahoo and MSN CAPTCHA system, respectively. The degree to which segmentation is improved is much higher with MSN CAPTCHAs than Yahoo CAPTCHAs. This is because of the amount of clutter items in the image. The MSN CAPTCHA system uses more clutter items than the Yahoo CAPTCHA system. Many of the components in the MSN CAPTCHA system cannot be separated by Chellapilla's algorithm or the projection technique, but the middle-axis point separation technique can overcome this difficulty effectively. Fig. 24 gives an example of the behavior of the complete algorithm; components which cannot be separated by the projection technique - such as '3', 'X', and 'H' in the Yahoo CAPTCHA, and 'R', 'U' 'S' and '8' in the MSN CAPTCHA - can be separated successfully by middle axis point separation.

Table 3: Segmentation rates, SRA and SRB, for the proposed algorithms and Chellapilla's algorithm.

(a) SRA results

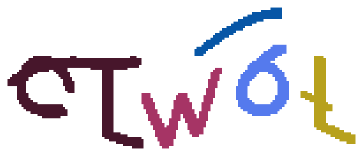
	Yahoo	MSN
Chellapilla's algorithm	60.57%	41.13%
Projection-only algorithm	67.05%	55.13%
Final algorithm	79.05%	76.00%

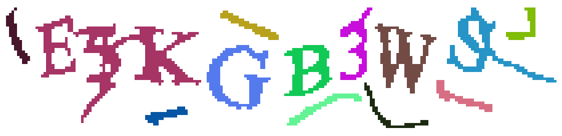
(b) SRB results

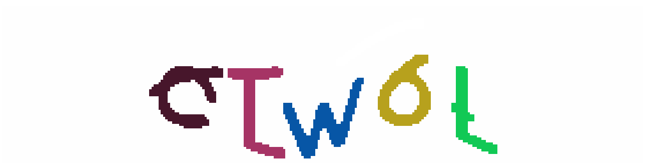
	Yahoo	MSN
Chellapilla's algorithm	28.00%	3.00%
Projection-only algorithm	34.00%	12.00%
Final algorithm	45.00%	18.00%

(a) Original Yahoo image.

(d) Original MSN image.


(b) Chellapilla's algorithm.


(e) Chellapilla's algorithm.


(c) Projection-only result.

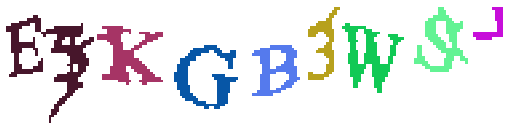
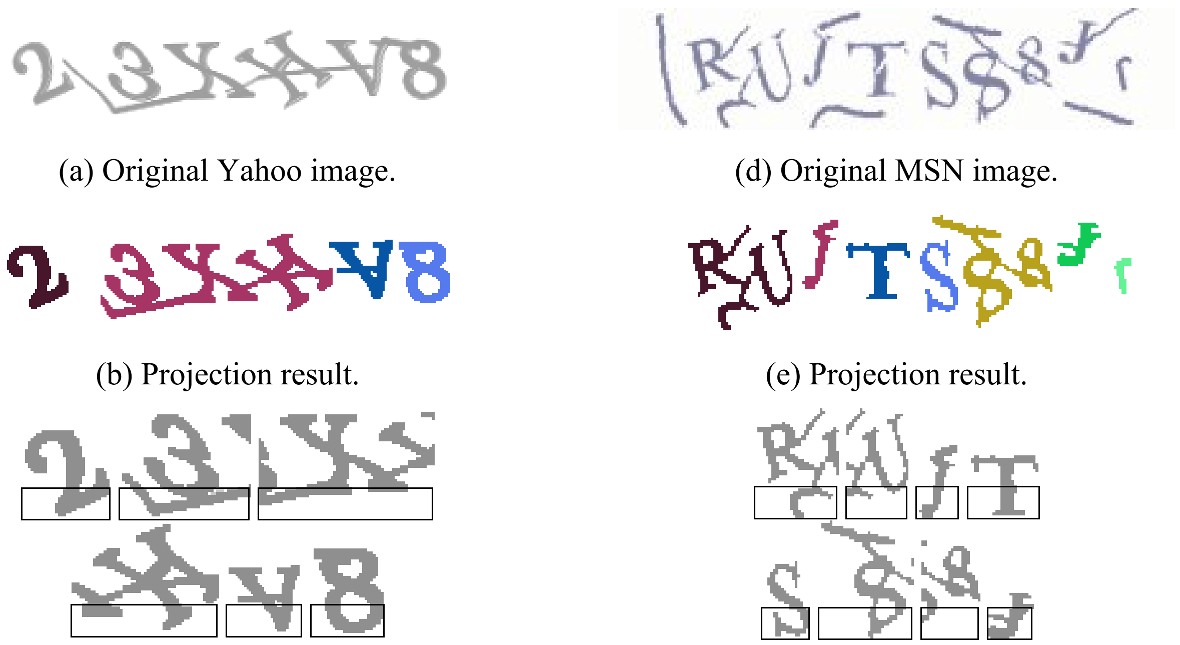

(f) Projection-only result.

Fig. 23: Algorithm results for two different types of CAPTCHA.



(a) Original Yahoo image. (d) Original MSN image.

(b) Projection result. (e) Projection result.

(c) Complete algorithm result. (f.) Complete algorithm result.

Fig. 24: Complete algorithm results for two different CAPTCHA systems.

VI. Conclusions and Future Works

In this paper, an effective novel algorithm was proposed for the segmentation of MSN and Yahoo CAPTCHA images containing line cluttering and character warping. The novel algorithm uses *projection* and *middle-axis point separation*, two new techniques which are proven to be effective against these CAPTCHAs. In the experimental results, it was found that the proposed algorithm results in a 18% relative increase in the segmentation rate compared with Chellapilla's algorithm when attacking Yahoo CAPTCHAs, and a 35% relative increase in the segmentation rate when attacking MSN CAPTCHAs. The proposed algorithm makes novel and useful contributions in the field of CAPTCHA analysis and image processing.

To increase the degree of successful segmentation of MSN and Yahoo CAPTCHA images, our future work will focus on developing and testing the following two techniques. Firstly, *character location prediction*. It is likely that characters have a predictable distribution. Developing an approach that can reliably guess the locations of remaining characters by analyzing the locations which are already known, may allow a useful increase in the segmentation rate. Secondly, *dynamic image opening*: different images have different fonts and different clutter widths. Therefore, using the same number of opening iterations for all of these images may cause mistakes during image erosion and dilation. A technique to dynamically change the number of image opening iterations might allow a further useful improvement in the segmentation rate.

This method described in this paper is specifically intended to address the problem of attacking CAPTCHAs as they are found in the real world, in the MSN and Yahoo systems (c. 2008). However, future CAPTCHA systems may generate characters in new ways, perhaps having varying numbers of characters, and dynamically varying lengths and widths of characters within each CAPTCHA image. We believe it may be possible to extend our technique to preemptively cope with such variations, by developing a scheme for horizontal projection per-component, in addition to vertical projection. The challenge of providing the flexibility to cope with future CAPTCHA designs is an interesting target for further research. For the time being, however, this paper's method represents a novel and useful contribution that effectively addresses the problem of segmentation for real-world modern CAPTCHA systems.

References

[1] M. Blum, L. A. von Ahn, and J. Langford, The CAPTCHA Project, "Completely Automatic Public Turing Test to tell Computers and Humans Apart," www.captcha.net, Dept. of Computer Science, Carnegie-Mellon Univ., and personal communications, November, 2000.

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

[3] G. Moy, N. Jones, C. Harkless, and R. Potter, "Distortion Estimation Techniques in Solving Visual CAPTCHAs," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 23-28, 2004.

[4] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski, "Computers Beat Humans at Single Character Recognition in Reading Based Human Interaction Proofs (HIPs)," in *Proceedings of the Third Conference on E-Mail and Anti-Spam*, 2005.

[5] K. Chellapilla and P. 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.

[6] A. L. Coates, H. S. Baird, and R. J. Fateman, "Pessimal Print: A Reverse Turing Test," in *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pp. 1154-1158, 2001.

[7] M. E. Hoque, D. J. Russomanno, and M. Yeasin "2D Captchas from 3D Models," in *Proceedings of the IEEE SoutheastCon*, pp. 165-170, 2006.

[8] H. S. Baird and J. L. Bentley, "Implicit CAPTCHAs," in *Proceedings of Document Recognition and Retrieval XII*, pp. 191-196, 2005.

[9] M. Shirali-Shahreza and S. Shirali-Shahreza, "Drawing CAPTCHA," in *Proceedings of the 28th International Conference on Information Technology Interfaces*, pp. 475-480, 2006.

[10] D. Misra and K. Gaj, "Face Recognition CAPTCHAs," in *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, pp. 122-127, 2006.

[11] S.Y. Huang, Y.K. Lee, G. Bell, and Z.H. Ou, "A Projection-based Segmentation Algorithm for Breaking MSN and YAHOO CAPTCHAs", in *Proceedings of the 2008 International Conference of Signal and Image Engineering (ICSIE'08)*, London, UK. (2008).

[12] J. Yan and A.S.E. Ahmad, "A low-cost attack on a Microsoft CAPTCHA," in *Proceedings of 15th ACM Conference on Computer and Communications Security,* Alexandria, Virginia, USA: ACM, 2008.