JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Study of Hollow Letter CAPTCHAs Recognition Technology Based on Color Filling Algorithm

Huishuang Shao[1], Yurong Xia[2], Kai Meng[3], and Changhao Piao[4,*]

Abstract

The hollow letter CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is an optimized version of solid CAPTCHA, specifically designed to weaken characteristic information and increase the difficulty of machine recognition. Although convolutional neural networks can solve CAPTCHA in a single step, a good attack result heavily relies on sufficient training data. To address this challenge, we propose a seed filling algorithm that converts hollow characters to solid ones after contour line restoration and applies three rounds of detection to remove noise background by eliminating noise blocks. Subsequently, we utilize a support vector machine to construct a feature vector for recognition. Security analysis and experiments show the effectiveness of this algorithm during the pre-processing stage, providing favorable conditions for subsequent recognition tasks and enhancing the accuracy of recognition for hollow CAPTCHA.

Keywords

Color Filling, Contour Restoration, De-noising, Hollow CAPTCHA

1. Introduction

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) technology is used to distinguish between computer programs and human users through an automatic Turing test program. This technology has found widespread usage in safeguarding networks against malicious attacks [1]. A good design criterion for CAPTCHAs is to ensure that the machine recognition rate remains below 1% while guaranteeing that humans can recognize them with at least 80% accuracy [2]. The main steps involved in solving test-based CAPTCHA encompass preprocessing, feature extraction and recognition. These steps play a pivotal role in enhancing the security of CAPTCHA systems, making them resilient against automated attacks, and at the same time, remaining user-friendly for human users.

Although convolutional neural network (CNN) [3] and deep learning [4] have shown remarkable performance in various tasks such as CAPTCHA recognition, image recognition, object localization, and classification, they often require a large amount of data per class in the dataset. However, collecting a substantial amount of data in real-world scenarios can be challenging. In contrast, support vector machine

(SVM) offers several advantages in dealing with small sample sizes and classification [5], recognition [6-8], nonlinear and high dimensional pattern recognition [9]. It is a machine learning algorithm based on the statistical learning theory of Vapnik-Chervonenkis (VC) dimension and structural risk minimization. It uses the kernel function to map the training samples into a high-dimensional subspace, where it can obtain the best classification feature vector [6]. This characteristic makes SVM an accurate classifier even in limited datasets [10].

The CAPTCHA created using hollow letters with a simple outline contains limited feature information. To increase the challenge for machine recognition, each character undergoes rotation within a certain angle and partially overlaps with neighboring characters, as depicted in Fig. 1. By incorporating interference technology, the difficulty level for machine recognition is significantly enhanced. Additionally, the character contour color is deliberately made identical to the background interference, preventing direct extraction of characters based on color information.

Fig. 1. CAPTCHA images.

While hollow CAPTCHA has become common on various websites, their resistance to automated attacks remains to be verified. This article focuses on exploring and highlighting the security vulnerabilities of distorted connected hollow CAPTCHA. Traditional approaches to attacking CAPTCHAs face limitations when dealing with this type of CAPTCHA. To address this challenge, we propose a novel automated attack framework. Our approach incorporates the seed filling algorithm and three detection methods to effectively remove noise blocks for the segmentation process. We then transform the hollow characters into solid ones, eliminate background interference and extract character feature for recognition using SVM. In this paper, we demonstrate that our series of pre-processing steps successfully address the segmentation challenges posed by distorted connected hollow CAPTCHAs. The rest of this paper is organized as follows. Section 2 introduces the seed filling algorithm used to transform hollow characters into solid ones, while Section 3 describes the process of removing background interference. Section 4 introduces the extraction of character feature. Section 5 presents the experimental results and analysis. Section 6 concludes the paper.

2. Filling Hollow Characters

In image preprocessing, when the binary images have no background interference, hollow characters are surrounded by black contour lines. To increase the effective feature information and improve recognition, hollow characters can be transformed into solid characters by filling the white area within each closed contour.

2.1 Seed Filling Algorithm

In order to fill each region of the binary image, we employ the seed-filling algorithm. Each foreground

pixel point serves as the seed point for color filling and the gray value of each closed area is incrementally increased by 10. The algorithm starts from a specified seed point and searches for adjacent pixels in the upward, downward, leftward, and rightward directions. If the color of the pixel matches the color of the seed point, this pixel is set to the new color and this newly colored pixel becomes the seed point for the next iteration. The process continues until the border of the region is reached. The specific steps of the seed-filling algorithm are as follows:

Step 1: Traverse each pixel from left to right and top to bottom, searching for white pixels.

Step 2: If white pixels are found in step 1, we set one of these pixels as the seed point and change the gray value of all white pixels in this connected area to 0. If no white pixel is found in step 1, it indicates that all regions in the image have been filled with colors, and the algorithm terminates.

Step 3: After each fill, the gray value increases by 10 and we return to step 1 for the next iteration.

By filling the white areas with different colors in the binary image, the original hollow CAPTCHA is effectively converted into solid characters, providing increased feature information for further processing and recognition.

2.2 Contour Restoration

In binary processing, some character contour may not be completely closed, as depicted in Fig. 2. This occurs because CAPTCHA designers intentionally alter the color intensity in specific areas of the contour line to resist attacks. Consequently, some parts of the contour line may be mistakenly treated as the background during the binary processing. To address this issue, it is necessary to repair the fracture points in the character contours before applying the color-filling algorithm.

Fig. 2. Schematic diagram of fracture contour lines.

Contour line restoration includes two steps: breakpoint location and connection. The breakpoint location involves finding the endpoint of each contour line. Traditional path search algorithms, such as Dijkstra algorithm [11] and maze algorithms, are commonly used for this purpose. The maze algorithm is to find an optimal path when the inlet and outlet is fixed [12]. The breakpoint process is based on the maze algorithm in this paper. Although the starting and ending positions of the character contour lines are not fixed, we can treat the whole contour line as a big maze. The starting position can be any point on the contour line and the impasse represents the breakpoint in the contour that we aim to find. To search for breakpoints, we explore the contour pixels that have not been traversed. If a black contour point is found within its eight neighborhoods, we set the point's color and proceed to explore the next contour points in its eight neighborhoods. This process continues until no new pixels can be explored, indicating that we have reached the breakpoint of the contour line. When the search is completed, the remaining black points are the breakpoints in contour lines. The pseudo-code for the breakpoint search is shown in Algorithm 1.

Algorithm 1. Breakpoint search algorithm
Input: Binary image
Output: Black breakpoints in contour lines

Begin

  Traverse from top to bottom and left to right, finding the black contour pixel points *p(i,j)* in turn.

  If each *p(i,j)* has unexplored contour paths in its eight neighborhoods.

    Set *p(i,j)* to gray.

    Push the unexplored contour paths into a stack as in the clockwise direction.

    While the stack is not empty.

      Pop and let the current pixel is *Q(i,j)*.

      If *Q(i,j)* has returns to the entry position, it means the contour line has no breakpoint. End the loop.

      If *Q(i,j)* has unexplored points in its eight neighborhoods, continue exploring the contour.

      If *Q(i,j)* has only gray points and background points in its eight neighborhoods, this point is a breakpoint.

    End while

  End if

End
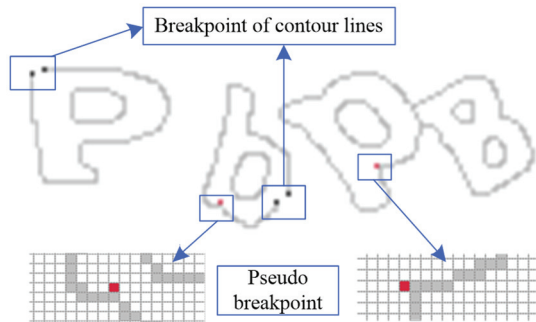
Fig. 3. Schematic diagram of breakpoints.

As shown in Fig. 3, contour lines are in gray color after the search of breakpoint. The red point (pseudo-breakpoint) is used for descriptive convenience, while the actual breakpoints of contour line are marked in black. The number of gray pixels in the eight neighborhoods of a pseudo-breakpoint is more than one, whereas a real breakpoint has only one pixel. Based on this feature, breakpoints can be detected. If any two breakpoint coordinates in the breakpoint coordinate set satisfy the following formula (1), they are considered a pair of matching points.

$$\begin{cases} x_1 < x_2 < x_3 < \cdots < x_n \\ \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} < 7.1 \quad i \in (1, n) \end{cases} \tag{1}$$

The algorithm for connecting breakpoint is as follows:

(1) Arrange the breakpoint set $[P_1, P_2, \ldots, P_n]$ in ascending order of their abscissa values. When multiple points have the same abscissa, sort them based on their ordinate values.

(2) Traverse the sorted breakpoint set. For each point P_i in the set, calculate the distance between the points $P_{i+1} \sim P_n$ and P_i, and find a subset of points with a distance of less than 7.1. If this subset contains more than one point, the point which has the same abscissa or ordinate value as P_i will be the matching point. If no point has the same abscissa or ordinate value as P_i, the point with the least abscissa difference between P_i will be the matching point.

(3) Connect the points that have been successfully matched and further remove these points from breakpoint set.

3. Removal of Background Interference

The addition of interference techniques to the background can significantly complicate the CAPTCHA, making it more challenging for character preprocessing. As shown in Fig. 4, multiple thick interference arcs are added to the background and change the background into a mesh-like pattern in the left graph. The interference pattern is superimposed on the background in the right graph. Moreover, the color of the character contour is deliberately made identical to the background interference color. This design choice aims to increase the difficulty of character preprocessing, making it more challenging to directly extract characters using color information.

For hollow CAPTCHAs with interference, the noise and disturbance reduction should be resolved firstly in the preprocessing. The corrosion morphology method is commonly used for de-noising solid CAPTCHAs. However, as depicted in Fig. 4, the character contour line is generally thinner than the interference background. The character outline may be greatly destroyed or even directly eliminated when using the corrosion morphology. This approach is not suitable for hollow CAPTCHAs, as it diminishes the effective features necessary for recognition. To address this issue, this article proposes an anti-interference technique based on color-filling algorithm. In this method, all white areas of the CAPTCHAs image are filled with different colors. By doing so, eliminating background interference can be converted to remove the interference block.


Fig. 4. Background interference.

3.1 Character Feature Analysis

Upon observing the structure of characters, many letters exhibit a closed-loop characteristic. The letter without a closed loop will produce different closed-loop characteristics by adding some baselines in its upper, middle, and lower part [13]. These feature values, when weighted and subjected to relevant mathematical operations, effectively reflect the unique characteristics of each letter, making the feature extraction method highly effective for letter identification [7]. However, when distinguishing between characters and non-characters, the feature extraction method appears to be complex and redundant. To address this, we analyze the relative positions of pixel points within character strokes to identify common characteristics among all characters.

Some character skeletons are shown in Fig. 5. On the left image, certain points share the same ordinate while their abscissa values are not continuous within the same character. That means in the same line, character contour points are also mixed with background points, which is called character transverse characteristics. For example, the ordinate of point A_1, A_2 is the same, but $x_1 + 1 \neq x_2$. Similarly, in the right image, these points have the same abscissa while the ordinate is not continuous in the pixels. That

means in the same column character contour points are also mixed with background points, which is called character vertical characteristics. For instance, the abscissa of point C_1, C_2 is x_1, but the ordinate $y_1 + 1 \neq y_3$. Among all capital and lowercase letters, the letter "L" lacks both horizontal and vertical characteristics, while the rest possess either horizontal, vertical characteristics, or one of them.
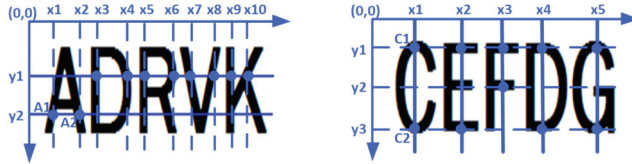

Fig. 5. Character skeleton diagram.

In the character skeleton, each letter has a width of a single pixel. By analyzing the boundary points, we can determine the characteristics of the character based on appropriate modifications and discriminant conditions related to horizontal and skeleton characteristics. A character with a boundary point set that satisfies the formula (2) is considered to have horizontal characteristics. Exchanging the X and Y in formula (2) is the expression of vertical characteristics. By applying these horizontal and vertical judgment conditions to the points in the set $B[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$, we can increment the corresponding horizontal or vertical characteristic value.

$$\begin{cases} i < j < q < \cdots < r & i,j,q,\cdots,r \in (1,n) \\ y_i = y_j = y_q = \cdots = y_r \\ x_i + 1 \neq x_j \cap x_j + 1 \neq \cdots \neq x_q \cap x_q + 1 \neq x_r \end{cases} \tag{2}$$

As shown in Fig. 6, the letter "P" in the character skeleton has a line width greater than 1, and the black pixels serve as the boundary points. The ordinate of P_1, P_3, P_8, P_{10} are both y_3 and their abscissa satisfy the operation relation of $(x_1 + 1 < x_3) \cap (x_3 + 1 < x_5) \cap (x_5 + 1 < x_7)$, thereby meeting the horizontal judgment condition. As a result, the horizontal characteristic value CharacX(p) is incremented by 1. Similarly, the points of P_4, P_5, P_6, P_7 meet the vertical judgment conditions, so the vertical characteristics CharacY(p) add 1.
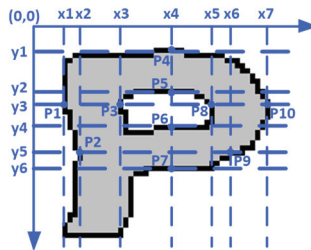
Fig. 6. Horizontal and vertical characteristics of hollow character.

After filling hollow parts, traversing each colour region in order and finding color block boundary point set $B[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$, we can calculate the horizontal and vertical characteristic value. The pseudo-code is shown in Algorithm 2.

Algorithm 2. Algorithm for calculation of horizontal and vertical characteristic value
Input: boundary point set
Output: horizontal and vertical characteristic value

Begin

   While $y_1 \leq ordinate \leq y_n$

     1. Find the abscissa of all points that belong to the same column and sort them in ascending order.

     2. Each abscissa x is traversed in order. x_{i+1} will be removed from the set B, if $x_i + 1 = x_{i+1}$. Then, we calculate the number of the remaining x.

     3. If the number of the remaining x is greater than or equal to 4, increment the horizontal characteristic value CharacX by 1.

   End While

   We output the horizontal characteristic value CharacX.

   Similarly, we can obtain the vertical characteristic value CharacY by focusing on the ordinate (y-coordinate) of the points in the set B.

3.2 Removing Interference by Three Times Detection

As shown in Fig. 7, the background was black after filling to the left CAPTCHA of Fig. 4. The interferential curve disappeared and changed to the noise blocks with different colors. The distinct color blocks exhibit a 10-grayscale difference. Based on the practical experience, color blocks with an area smaller than a certain threshold (set to 15 in this article) are identified as noise blocks. The effect diagram of removing small blocks is shown in Fig. 8.


Fig. 7. Effect diagram of color filling.

Fig. 8. Effect diagram of removing small blocks.

Fig. 8 illustrates that some color blocks have noticeable burrs caused by binarization. However, these burrs can be eliminated through a series of morphological operations, including morphological expansion, morphological corrosion, and morphological expansion again. As a result, the block becomes smoother and more complete, making it easier to perform subsequent denoising. Fig. 9 displays the classification of color blocks after morphological processing. Extracting all character blocks is difficult, as some character blocks may lose their horizontal and vertical characteristics after fracture. Mistakenly removing a character block as noise can significantly impact recognition accuracy. Hence, the process of removing noise blocks in this article is designed to be combined with repairing character blocks in three steps.
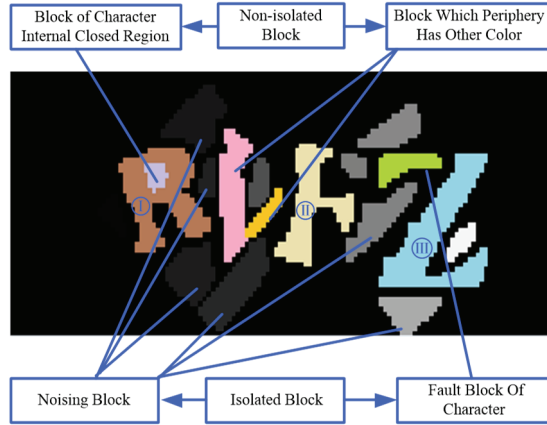
Fig. 9. The classification of color blocks after morphological processing. Isolated color blocks include noise blocks and character fault blocks. Non-isolated blocks include character internal closed regions and blocks of other pixels surrounding them.

After determining the partial effective character blocks based on their horizontal and vertical characteristic, the remaining color blocks can be further divided into four types. Boundary contour point set centrally stores the boundary pixels of the color block, and the number of pixels represents the circumference of the color block. External contour point set centrally stores the pixels that do not belong to the color block but are adjacent to the boundary contour points. The number of these pixels can represent the outer circumference of the color block and is denoted as $OutC$. Based on the information in the external contour point set, each color block is further classified into isolated color block and non-isolated color block.

Isolated color blocks can be further divided into noise blocks and character fault blocks. Character fault blocks tend to have a more slender shape compared to noise blocks, and the distribution of pixels in noise blocks is more concentrated. The aspect ratio of the area occupied by each isolated color block can be calculated by $\theta = (x_{max} - x_{min})/(y_{max} - y_{min})$, where x_{max} and x_{min} are the maximum and minimum abscissa in the external contour point set of the isolated color block, respectively. y_{max} and y_{min} are the maximum and minimum ordinate in external contour point set of the isolated color block, respectively.

There are two cases in the color block of character internal closed region. One is the closed region produced by closed-loop character contour lines. In this situation, the color of the external contour points is the same as the color of the character block. Another is the closed region produced by adjacent characters. In this case, the color of the external contour points is composed of multi-characters block and the background. A common feature of such regions is that the proportion of black points in the external contour point set is small or zero. We define a parameter $\rho = \frac{OutBlackC}{OutC}$, where $OutBlackC$ is the number of background color points in the external contour point set. The color block is considered as the closed region in the character, if $\rho < 0.45$. The color blocks that are adjacent to each other are generally produced by a fracture of the same character. In this case, each block does not have horizontal and vertical characteristics. However, when these blocks are matched and combined in the same color block, the characteristic will appear. Based on the analysis of different color block, the algorithm flowchart of removing noising blocks by three detections is shown in Fig. 10.
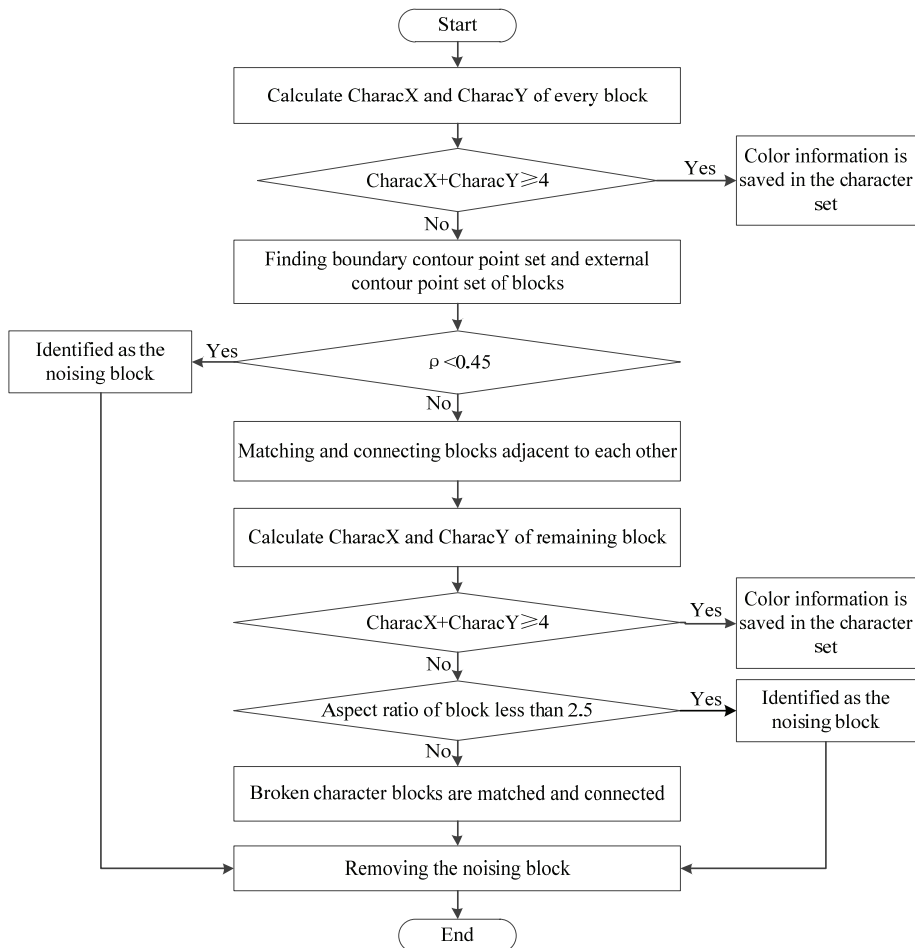
Fig. 10. The algorithm flowchart of removing noising blocks by three detections.

The algorithm for removing noise blocks through three detections is described in the following steps:

Step 1: We detect the horizontal and vertical characteristics of each block and search for noise blocks. Then, we match and connect conglutinated blocks, and calculate the sum of horizontal and vertical characteristics of each block. If the sum of characteristics is equal or greater than 4, this block will be identified as a character block and saved in the character set. If the proportion of background points in the external contour point set of a remaining block is less than 0.45, it is identified as a noise block and subsequently removed. Detect whether the remaining blocks are conglutinated and match with other blocks. If it happens, we then set these blocks to the same color.

Step 2: We detect the horizontal and vertical characteristics of each block for the second time and search for noise blocks. Match and connect the character blocks with their corresponding fracture blocks. We detect the characteristics of remaining blocks. If the sum of the remaining block's horizontal and vertical characteristics is equal or greater than 4, the block will be identified as the character block. Then, we calculate the length-to-width ratio of the remaining blocks. The block will be identified as the noise block, if this ratio is less than 2.5. We try to

find the fracture blocks that can match the character blocks. Generally, the block can only be the fracture block if their gray values have the smallest difference. Additionally, the ratio of the fracture block's pixels in the character region to its area is larger than 0.4.

Step 3: We detect the horizontal and vertical characteristics for the third time, remove the residual noise blocks and set their color to white.

4. Character Feature Extraction

After removing interference, character blocks are filled with different gray in the CAPTCHA image. The gray value of each character block increases sequentially from left to right. So each character block can be segmented and extracted according to the gray information. The preprocessing and feature extraction framework is shown in Fig. 11.
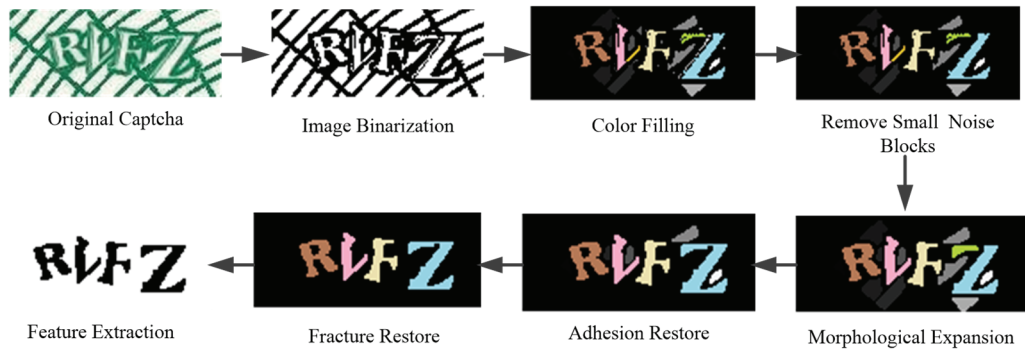
Fig. 11. Preprocessing and feature extraction framework.

Despite the varying width of each character block, the skeleton information accurately reflects the overall topology, effectively reducing redundant features [14]. The Zhang-Suen thinning algorithm is employed in this study to obtain the skeleton of character blocks [15]. Each refined character skeleton is placed uniformly at the center of a 32×32 size image, as shown in Fig. 12. We extract features of each point and establish the eigenvector, by scanning the pixels from left to right and from top to bottom. When a white pixel is scanned, its feature value is set to "0"; when a black pixel is scanned, its feature value is set to "1".

This article employs SVM to judge the feature information. We select samples of uppercase and lowercase letters to create the training and test sets. The SVM tool used in this article is the LIBSVM software package, developed by Chang and Lin [16] of National Taiwan University. We train the SVM by using the radial basis function (RBF) kernel function. The quality of the penalty coefficient setting directly impacts the generalization ability of the support vector machine. Therefore, we employ cross-validation provided by LIBSVM to determine the optimal values for parameters C and γ before training. Then, we use the best values of C and γ to train the whole training set. Once the training is completed, the resulting model can be imported into the recognition program to identify other character feature vectors. During the construction of the training set, the feature vector of each sample character should be preserved in text format as follows:

<center><label> <index1>: <value1> <index2>: <value2>…<index n>:<value n></center>

Among them, each line represents a sample. The "<label>" indicates the classification label of the sample, indicating the character recognition results. "<index>" is an integer, starting from 1, which represents the number of characters in the sample, namely the dimension number. The "<value>" is a real number indicating the characteristic value of this dimension. If the characteristic value is 0, the "<index>" and "<value>" can be omitted. This allows for discontinuous natural numbers as "<index>", which can accelerate the training and recognition rate of SVM.
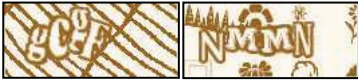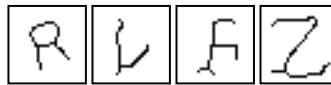

Fig. 12. Character skeleton.

5. Experimental Results and Analysis

The algorithm was implemented in Python 3.3. The image data used for experimentation was obtained from the Tencent website and consisted of two types of CAPTCHAs. Each type of CAPTCHAs had different interference forms but was composed of four characters. Firstly, 4,000 pictures with different shapes are selected as samples. These samples were manually identified and used to create the training set for SVM training. After getting the model, 100 new pictures are selected randomly from each type of CAPTCHAs. The experimental results are shown in Table 1.

Table 1. Experimental results

Type	Example	Contour thickness	Character style	Recognition rate (%)
Background interference		Basically identical	Deformation, rotation	41
None interference		Having changed	Different size, Having deformation and rotation	48

There are several reasons for the hollow character recognition errors. For no background interference CAPTCHAs, recognition errors are mainly concentrated in small letters with less distinct shape features or breakpoint connection errors. Breakpoint errors occur when there is a break in the two-letter common contour, and the connection between breakpoints leads to errors. Additionally, character corners that are damaged or have small, unclear broken blocks might be mistakenly removed as interference noise blocks. In cases where characters overlap and share the same hollow area, characters may be matched and the connection results in the remaining character's disability. For CAPTCHAs with background interference, recognition errors are mainly concentrated in characters with distorted shapes. The interference background was sometimes wrongly judged as a fault block, leading to incorrect connections. The severe distortion of characters makes their inherent characteristics less obvious, making them easily confused with other characters.

A good design criterion for CAPTCHA robustness is that the machine recognition rate should be less than 1% [2]. Therefore, the recognition algorithm proposed in this paper is effective and feasible. To validate the performance of our method, we compared it with other attacks on Tencent CAPTCHAs with background interference. As depicted in Table 2, experimental results demonstrate that our method outperforms the others [3,17].

Table 2. Success rate of different attack methods on Tencent CAPTCHAs

CAPTCHA scheme	Method	Recognition rate (%)
	Gao et al. [3]	23
	Chen et al. [17]	13
	Our method	41

5. Conclusion

In this paper, we use the color filling algorithm to convert hollow characters into solid character during the pre-processing stage, thereby enhancing the character information. Additionally, we have conducted a comprehensive analysis of character shape features and proposed a noise reduction method that involves removing noise background through the elimination of noise blocks. This approach significantly improves the pre-treatment of hollow characters, making them suitable for existing character feature extraction and recognition technologies. The experimental results show that the proposed algorithm can effectively fill characters, remove interference, and improve the pre-treatment effect. However, there are certain limitations in cases where characters are too small or exhibit adhesion deformation. These situations may lead to suboptimal pre-treatment results, subsequently reducing the recognition rate. This issue will be focused on the next stage.

Acknowledgement

References

[1] L. Von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Communications of the ACM*, vol. 47, no. 2, pp. 56-60, 2004. https://doi.org/10.1145/966389.966390

[2] J. Yan and A. S. El Ahmad, "A low-cost attack on a Microsoft CAPTCHA," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, Alexandria, VA, 2008, pp. 543-554. https://doi.org/10.1145/1455770.1455839

[3] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow CAPTCHAs," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, Berlin, Germany, 2013, pp. 1075-1086. https://doi.org/10.1145/2508859.2516732

[4] F. Stark, C. Hazirbas, R. Triebel, and D. Cremers, "Captcha recognition with active deep learning," in *Proceedings of the Workshop on New Challenges in Neural Computation*, Aachen, Germany, 2015, pp. 94-102.

[5] M. J. J. Ghrabat, G. Ma, I. Y. Maolood, S. S. Alresheedi, and Z. A. Abduljabbar, "An effective image retrieval based on optimized genetic algorithm utilized a novel SVM-based convolutional neural network classifier," *Human-centric Computing and Information Sciences*, vol. 9, article no. 31, 2019. https://doi.org/10.1186/s13673-019-0191-8

[6] F. Zhang, T. Y. Wu, J. S. Pan, G. Ding, and Z. Li, "Human motion recognition based on SVM in VR art media interaction environment," *Human-centric Computing and Information Sciences*, vol. 9, article no. 40, 2019. https://doi.org/10.1186/s13673-019-0203-8

[7] S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, A. A. Abbasi, S. J. Kwon, "Deep learning scheme for character prediction with position-free touch screen-based Braille input method," *Human-centric Computing and Information Sciences*, vol. 10, article no. 41, 2020. https://doi.org/10.1186/s13673-020-00246-6

[8] G. Xu and S. Zhang, "Fast leaf recognition and retrieval using multi-scale angular description method," *Journal of Information Processing Systems*, vol. 16, no. 5, pp. 1083-1094, 2020. https://doi.org/10.3745/JIPS.02.0142

[9] R. E. Fan, P. H. Chen, C. J. Lin, and T. Joachims, "Working set selection using second order information for training support vector machines," *Journal of Machine Learning Research*, vol. 6, no. 12, pp. 1889-1918, 2005.

[10] M. A. I. Mahmoud and H. & Ren, "Forest fire detection and identification using image processing and SVM," *Journal of Information Processing Systems*, vol. 15, no. 1, pp. 159-168, 2019. https://doi.org/10.3745/JIPS.01.0038

[11] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, p 269-271, 1959. https://doi.org/10.1007/BF01386390

[12] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, vol. 10, no. 3, pp. 346-365, 1961. https://doi.org/10.1109/TEC.1961.5219222

[13] H. Gao, W. Wang, and Y. Fan, "Divide and conquer: an efficient attack on Yahoo! CAPTCHA," in *Proceedings of 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, Liverpool, UK, 2012, pp. 9-16. https://doi.org/10.1109/TrustCom.2012.131

[14] H. Liu, Z. Wu, D. F. Hsu, B. S. Peterson, and D. Xu, "On the generation and pruning of skeletons using generalized Voronoi diagrams," *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2113-2119, 2012. https://doi.org/10.1016/j.patrec.2012.07.014

[15] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236-239, 1984. https://doi.org/10.1145/357994.358023

[16] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 1-27, 2011. https://doi.org/10.1145/1961189.1961199

[17] J. Chen, X. Luo, J. Hu, D. Ye, and D. Gong, "An attack on hollow captcha using accurate filling and nonredundant merging," *IETE Technical Review*, vol. 35(sup1), pp. 106-118, 2018. https://doi.org/10.1080/02564602.2018.1520152

Huishuang Shao https://orcid.org/0000-0002-3437-8816

She received M.S. degrees in School of Automation, Chongqing University of Posts and Telecommunications in 2018. Since September 2020, she is with the College of Computer Science and Technology from Chongqing University of Posts and Telecommunications as a Ph.D. candidate. Her current research interests include blockchain and security in vehicular.

Yurong Xia https://orcid.org/0000-0002-7965-1900

She received B.S. degree in Computer Science from Shandong Normal University in 2002, M.S. degree in computer science from Qufu Normal University in 2009. She is currently a senior lecturer in the Training Center, Jining Polytechnic, Jining, China. Her research interests include computer software and applications.

Kai Meng https://orcid.org/0000-0002-2525-4942

He received M.S. degree in control engineering from Chongqing University of Posts and Telecommunications in 2017. He is currently an engineer in Unicom Shanxi industrial Internet Co. Ltd. His research interests include artificial intelligence and image processing.

Changhao Piao https://orcid.org/0000-0002-0613-4777

He received his B.E. degree in electrical engineering and automation from Xi'an Jiaotong University in 2001, M.S. and Ph.D. degree in Inha University, South Korea in 2007. He is currently a professor in School of Automation, Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include automobile electronics and energy electronics.