Configurable image recognition framework design based on KNN and bit-based similarity model

Gang Lin[a,*], Yanchun Liang[b], Yonglin Chen[a], Wenbo Pan[a]
[a]School of Computer Science, Zhuhai College of Science and Technology, Zhuhai, Guangdong Province, China; [b]School of Computer Science, Jilin University, Jilin, Changchun Province, China

ABSTRACT

CAPTCHAs are widely utilized on the Internet to partially protect against computer attacks, and text-based CAPTCHAs are commonly used. In order to make the more flexible attack, this paper provides a framework with configurable options based on k-NN, including three major parts: preprocessing the binary image, building standard library and recognizing image. The standard library is built from training data set, where the third part can be an option to drop out some characters with a high similarity, and the library is used for testing data set. A bit-based similarity model is proposed, where "and" and "or" bit operations are executed, and the result is the ratio of both operations. Finally, the framework is applied into 4 typical scenarios, MNIST handwriting database, CAPTCHAs built by the CAPTCHA generator, online CAPTCHAs of CNKI website, and CAPTCHAs within open source PHP DedeCMS, the average classification accuracy is 97.05%. As a result, the model is simple but effective, the framework can work well for text-based CAPTCHAs and handwritten numbers, which may make associated websites pay more attention to current authentication mechanism, and it offers flexibility to cover more algorithms and application scenarios by implementing different logics of preprocessing according to defined APIs.

Keywords: CAPTCHAs, binary image recognition framework, KNN, bit-based similarity model

1. INTRODUCTION

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are widely used on the Internet to distinguish between human users and computer programs[1-5]. There are many kinds of CAPTCHAs available, such as text-based, image-based, audio-based, video/animation-based, puzzle, graphical slider, face-recognition, etc.[6-9]. Text-based CAPTCHAs are frequently used by whether enterprises or individuals due to convenience and user-friendly look[10, 11]. There are a lot of researches on text-based CAPTCHAs, but few of them focused on and depicted a generic framework to attack CAPTCHAs, much less building a framework flexible enough to meet different needs, such as different priority of accuracy, time-consuming, data storage space of standard library, and so on.

The *k*-NN (*k*-nearest neighbor) algorithm as a typical classification algorithm is one of the simplest algorithms among almost all current famous data mining algorithms such as C4.5, *k*-Means, SVM, Apriori, EM, PageRank, etc., which principle is also widely applied into other algorithms such as collaborative filtering, Rocchio[12]. The *k* nearest neighbor or *k*-NN classification determines the decision boundary locally where *k* is a parameter[13, 14]. Generally, in many cases, the 1-NN rule may be strictly better than the other *k*-NN ($k > 1$) rules[12, 15]. It is important to design a similarity model for *k*-NN algorithm with a high classification accuracy and a low storage requirement.

The standard process consists of both segmentation process and recognition process to identify Text-based CAPTCHAs[16, 17], which can be designed in sequence or not. The segmentation then recognition approach executes segmentation process first, and the final results largely depend on the correct number of segments[4]. However, the segmentation simultaneously recognition approach scores all possible ways to segment a CAPTCHA and decides which combination is the most likely to be the correct one through 4 major components: the Cut-point Detector, the Slicer, the Scorer, and the Arbiter[4]. Additionally, a generic approach based on Gabor filters uses a single segmentation and recognition strategy with two main steps: “extracting components” and “partition and recognition”, where Gabor filters is applies to extract character components instead of an intact character from Captcha images along different directions respectively[2]. Although both the segmentation simultaneously recognition approach and the generic approach based on

[*]lingang19@zcst.edu.cn

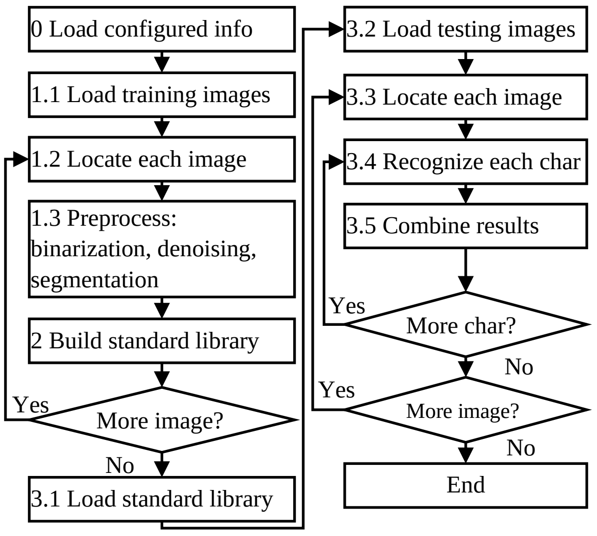
International Conference on Computer Application and Information Security (ICCAIS 2021), edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12260, 122601I · © The Authors. Published under a Creative Commons Attribution CC-BY 3.0 License · doi: 10.1117/12.2637494

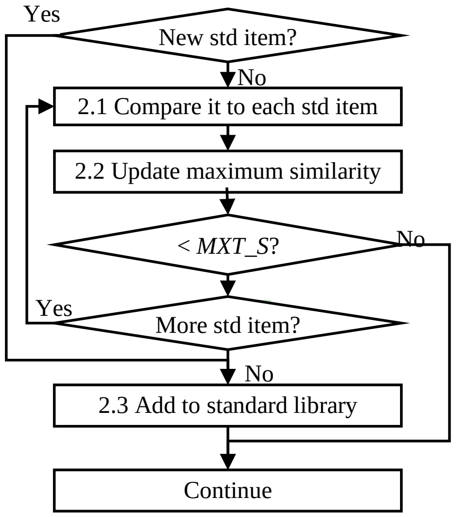
Gabor filters work well for a lot of types of CAPTCHAs, it is very difficult to integrate them with a generic framework based on *k*-NN. In order to make the framework as a global instead of local function, standard process, i.e., the segmentation then recognition approach is used.

This paper is organized as follows. Firstly, the design of the binarization image recognition framework is introduced briefly, including the overall design and key points of each process. Secondly, an implementation of the framework is illustrated with some key APIs, configuration files, key logics of *k*-NN algorithm and a bit-based similarity model. Finally, four scenarios were applied to the framework, many different sub scenarios were made for each one, and the info of each execution time, accuracy rate was gathered before its analysing.

2. DESIGN ON IMAGE RECOGNITION FRAMEWORK

The framework consists of 3 major parts: preprocessing, building standard library, and recognition, as shown in Figure 1.





New std item?
Yes
No
2.1 Compare it to each std item
2.2 Update maximum similarity
< MXT_S?
No
More std item?
Yes
No
2.3 Add to standard library
Continue

Figure 1. Total Design of Image Recognition Framework.

Figure 2. Design of Building Standard Library.

2.1. Preprocessing

After each image located, preprocessing is executed. The segmentation is based on the image after binarization and denoising, which converts to binary an input image having pixels defining text and background, each of its pixels is either dark or white[18]. Proper binarization is very important for separating the foreground object from the background for images[19][19]. The foreground is with digital, character or other valid information, and the background without it. Once the foreground is separated from the background, the image can be distinguished by recognizing the shape of each foreground image one by one, such as handwritten characters and CAPTCHAs after preprocessing.

2.2. Building standard library

All training images were annotated in advance before building standard library. There are 2 different methods when judging whether any character of each training image is to add into the standard library. One method is without any limits, i.e., the standard library contains all characters of training images, which size is always big and equal to the number of training characters. The other one is to set a maximum threshold (*MXT_S*), only characters are added when the similarity value between it and other existing items in standard library is less than the threshold, and the size of standard library is always smaller and depends on the threshold value, as shown in Figure 2. Generally, the bigger the threshold is set, the bigger the size of standard library, and the higher the accuracy. The method one could be seen as a special case of the method two since it means no limits when the threshold is set to 1. The framework uses the threshold configured in a configuration file or database.

There are some popular algorithms of the similarity, such as Person Correlation Coefficient, Cosine Similarity[20], Jaccard Similarity[21], etc. The bit-based similarity model the framework used is similar to cosine similarity. The image data after preprocessing are stored in the memory in binary form, 0 for white and 1 for black, or vice versa. Two images are same when each individual pixel is same in the same position, and they are similar when most pixels are same, only few are different. Given 2 images *A* and *B*, *P*(*A*) and *P*(*B*) represent the binary matrix of *A* and *B*, respectively. The result of bitwise-or between *P*(*A*) and *P*(*B*) indicates the largest range of foreground like union set of *P*(*A*) and *P*(*B*), and the result of bitwise-and between *P*(*A*) and *P*(*B*) indicates same range of foreground area like intersection of *P*(*A*) and *P*(*B*). The smaller the difference between both results, the bigger the similarity. The bit-based similarity model defines the similarity by dividing the number of same foreground area of *P*(*A*) and *P*(*B*) into the number of the largest range of foreground area of *P*(*A*) and *P*(*B*), i.e., dividing the bitwise-or result by bitwise-and result, as showed in equation (1). The higher the similarity value is, the closer these 2 images or characters are. Hence the similarity is some number between 0 and 1, $S(A, A) = 1$, in another words, A resembles itself 100%, for any size[21].

$$S(A,B) = \frac{|P(A)\&P(B)|}{|P(A)|P(B)|} \tag{1}$$

2.3. Image recognition

The *k*-NN algorithm is used to predict which category each target sample belongs to by a majority of *k* neighbor standard samples closest to the target. The bit-based similarity model is used and *k* is configured which can be changed in different scenarios. Once *k* is set, as candidates, the top *k* standard items ranked in order of similarity are cached during recognition. Standard items are compared to the target, one item a time, until a match with the similarity higher than a threshold value of "direct recognition" or no more items required to match. Among the top *k* standard items, the final value is determined by the majority of candidate character values. If more than one candidate character values are tied for the same majority, the value with a higher similarity is the final result. Therefore, the final result is not always with the highest similarity when *k* is greater than 1.

The *k*-NN algorithm has the advantage of being easy and practical, and with a suitable accuracy[20]. There are some algorithms, such as CNN (Condensed Nearest Neighbor), ball-tree, *kd*-tree (*k*-dimensional tree), LSH (Locally Sensitive Hash) etc., to reduce storage space or improve the speed of locating *k* nearest neighbors[20, 22, 23]. In the framework, the byte array is used for reduction of storage space consumption. Each pixel of the image data after preprocessing is indicates in binary form such as 0 and 1 for white and black separately and 8 pixels are grouped within a byte. The image data is stored in a byte array. As shown in Table 1, it lists a binary array for a six after preprocessing, which is stored as an array of byte with length 40. It needs about 7 times more storage space if each one or zero is used directly in the application since a byte is a basic unit.

Besides, one more feature is designed for quick locating associated standard items, such as foreground area percentage. There are many items with small similarity value wasting time if all of items within standard library are required to be matched per recognizing a character. In order to locate the standard item with high similarity value as quick as possible, a new feature, the foreground area, is used in the framework. That is to say, on executing each recognition, items in standard library are grouped by the foreground area, as a result, different group has different priority to match, and some with low priority are skipped. Let *key* as the feature value, shown in equation (2), where s_i is the foreground area of item i, and *f* is a customized function. A proper function makes a good balance between the number of groups and the coverage percentage of characters to recognize per group. In general, the less the number of groups is, more number of items each group contains, more coverage percentage of characters each group has.

$$key_i = f(s_i) \tag{2}$$

A linear function is proposed, such as shown in equation (3), where *a* is a coefficient between 0 and 1, such as 0.1. Given that *S* is the total area including foreground and background area, the domain of the function is an open range as shown in equation (4), while the range is a close range as shown in equation (5).

$$key_i = \lfloor a \cdot s_i \rfloor \tag{3}$$

$$s_i \in (0, S) \tag{4}$$

$$key_i \in [0, \lfloor a \cdot S \rfloor] \tag{5}$$

For some character to be recognized, it is easy to get the feature value of *key* according to equation (3). Once the *key* is calculated, the priority of all items in group *key* is the highest, and the priority value goes down in the direction of distance between group number and *key*. The priority value can be calculated according to equation (6), where gn_i is the group number of standard item *i*, *key* is the feature value of a character to be recognized, and *MAX* is the maximum that is greater or equal than the total number of groups.

Table 1. Array for a six after preprocessing.

Binarization																**Byte**	
0	0	0	0	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	14	0
0	0	0	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	30	0
0	0	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	56	0
0	0	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	56	0
0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	48	0
0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	48	0
0	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	112	0
0	**1**	**1**	0	0	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	96	127
0	**1**	**1**	0	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	96	255
1	**1**	**1**	0	0	0	**1**	**1**	**1**	**1**	**1**	0	0	0	**1**	**1**	227	227
1	**1**	**1**	0	0	**1**	**1**	**1**	**1**	0	0	0	0	0	**1**	**1**	231	131
1	**1**	**1**	0	0	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	231	3
1	**1**	**1**	0	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	**1**	238	3
1	**1**	**1**	0	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	**1**	238	3
1	**1**	**1**	0	0	**1**	**1**	**1**	0	0	0	0	0	**1**	**1**	0	231	6
1	**1**	**1**	0	0	0	**l**	**1**	**1**	0	0	0	**1**	**1**	**1**	0	227	142
1	**1**	**1**	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	0	0	224	60
0	**1**	**1**	**1**	0	0	0	**1**	**1**	**1**	**1**	**1**	0	0	0	0	113	240
0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	63	224
0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	31	128

$$p_i = MAX - |gn_i - key| \tag{6}$$

There are 2 threshold values called *MXT_E*, and *MXN_E* in the framework. When the maximum similarity is less than *MXT_E* and the number of different priorities matched is less than *MXN_E*, more groups with next priority are involved, otherwise, the recognition stops. The image result returns when all characters are recognized, as shown in Figure 3.

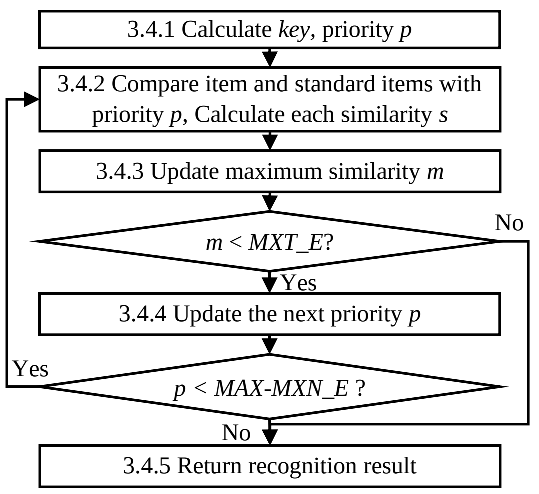

3.4.1 Calculate *key*, priority *p*
3.4.2 Compare item and standard items with priority *p*, Calculate each similarity *s*
3.4.3 Update maximum similarity *m*
m < *MXT_E*?
No
Yes
3.4.4 Update the next priority *p*
Yes
p < *MAX-MXN_E* ?
No
3.4.5 Return recognition result

Figure 3. Design of character recognition.

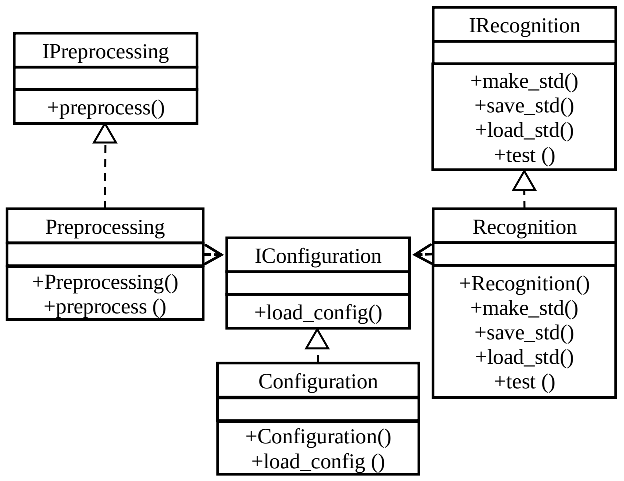


Figure 4. Three main classes of framework.

3. API AND APPLICATION

3.1. API

There are 3 main interfaces and associated classes in the framework, as shown in Figure 4.

3.2. Configuration values

According to the design, some common parameters are to configure in advance, as shown in Table 2.

Additionally, all of the possible values need being pre-configured, such as cnki, there are 27 candidates in total, as shown in Table 3. The number or index can be retrieved according to the expected or tagged character in pre-computed hash tables, and vice reverse.

3.3. Application scenarios

The framework was applied for 4 scenarios, MNIST handwriting database, CAPTCHAs built by the captcha generator, online CAPTCHAs of CNKI website, and CAPTCHAs within open source PHP DedeCMS.

Table 2. Some configuration values.

Name	Example	Comments	Name	Example	Comments
DATA_TYPE	1	1:mnist/2:captcha/3:cnki/4:cms	*MXT_E*	0.85	Threshold of extending the search
MXT_I	0.95	Threshold of direct recognition	*MXN_E*	5	Max ranges of priority to search
MXT_S	0.95	Threshold of standard library	*a*	0.1	The coefficient for feature value of *key*

Table 3. Relationship between sequence number and char.

No.	Char	No.	Char	No.	Char	No.	Char	No.	Char	No.	Char	No.	Char	No.	Char	No.	Char
0	2	1	3	2	4	3	5	4	6	5	8	6	9	7	A	8	B
9	C	10	D	11	E	12	F	13	G	14	H	15	J	16	K	17	L
18	N	19	M	20	P	21	R	22	S	23	T	24	W	25	X	26	Y

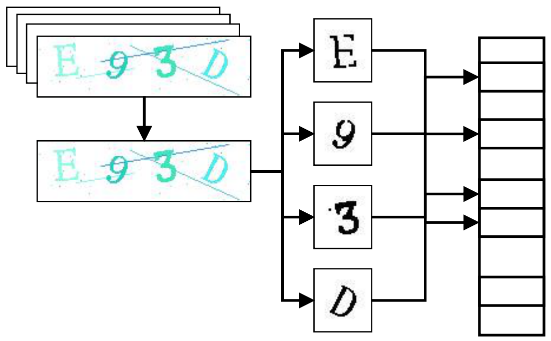


Figure 5. Preprocessing and locating standard items.

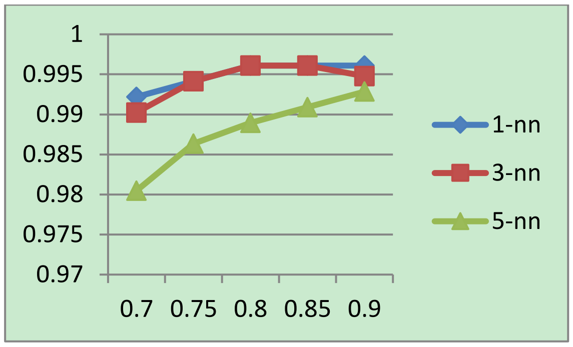


Figure 6. Relationships between accuracy and MXT_E.

Table 4. Test results for 4 different scenarios.

	MNIST	CAPTCHAS	CNKI	CMS
knn/MXN_E	1/10	1/5	1/5	1/5
MXT_I/MXT_S/MXT_E	0.95/0.9/0.8	0.995/0.995/0.95	0.995/0.995/0.95	0.95/0.95/0.8
Count of training/test/right items	60000/10000/9667	6000/4000/3826	4000/8400/8085	700/1537/1531
accuracy	96.67%	95.65%	96.25%	99.61%
Testing/average time	46320.38/4.63	53892.36/13.47	41554.62/4.95	93.27/0.06

On making the standard set or recognition, preprocessing is the first step, e.g., the image “E93D” was converted to 4 individual images “E”, “9”, “3” and “D”, and then the next step is locating the position of hash tables according to Table 3. With the hash tables, it is quick to locate associated group according to the tagged value, as shown in Figure 5. Table 4 lists test results for all these 4 scenarios. It is seen that the framework work well for different scenarios with the accuracy 97.05% on average. Generally, the accuracy differs a little according to the parameters configured. Take the CMS scenario as an example, the accuracy differs according to different *knn* and *MXT_E*, as shown in Figure 6.

4. CONCLUSION

This paper provides a configurable image recognition framework based on *k*-NN and bit-based similarity model, where 4 different scenarios were successfully applied. For different scenarios, different parameters can be configured in order to get better results. Additionally, the framework can be expended to other scenarios conveniently by setting different parameters, or be updated to involve more algorithms by adding some new concrete classes to implement the interfaces mentioned in the papers.

ACKNOWLEDGMENTS

The authors are grateful to the support of the Guangdong Key Disciplines Project (2021ZDJS138). This work was financially supported by 2021 University-level Teaching Quality Project supported by Zhuhai College of Science and Technology (Grant No. ZLGC20210203).

REFERENCES

[1] Thobhani, A., Gao, M. S., Hawbani, A., et al., “CAPTCHA recognition using deep learning with attached binary images,” Electronics, 9(1522), 1-19 (2020).

[2] Gao, H., Yan, J., Fang, C., Zhang, Z. and Li, J., "A simple generic attack on text captchas," Network & Distributed System Security Symp., 1-14 (2016).
[3] George, D. et al., "A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs," Science, 358(6368), 1-9 (2017).
[4] Bursztein, E., Aigrain, J., Moscicki, A. and Mitchell, J. C., "The end is nigh: Generic solving of text-based CAPTCHAs," USENIX Conf. on Offensive Technologies, 1-15 (2014).
[5] Hussain, R., Kumar, K. H., Gao, H. and Khan, I., "Recognition of merged characters in text based CAPTCHAs," Proc. of the 2016 3rd Inter. Conf. on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 3917-3921 (2016).
[6] Kumar, M. and Jindal, M. K., "A systematic survey on CAPTCHA recognition: Types, creation and breaking techniques," Archives of Computational Methods in Engineering, 28(4), 1-30 (2021).
[7] Hernandez-Castro C. J., R-Moreno M. D. and Barrero D. F., "Using JPEG to measure image continuity and break capy and other puzzle CAPTCHAs," IEEE Internet Computing 19(6), 46-53 (2015).
[8] Schryen, G., Wagner, G. and Schlegel, A., "Development of two novel face-recognition CAPTCHAs: A security and usability study," Computers & Security, 60, 95-116 (2016).
[9] Xu Y., Reynaga G., Chiasson S., et al., "Security analysis and related usability of motion-based CAPTCHAs: Decoding codewords in motion," IEEE Transactions on Dependable & Secure Computing, 11(5), 480-493 (2014).
[10] Wang, Y. and Lu, M., "An optimized system to solve text-based CAPTCHA," International Journal of Artificial Intelligence and Applications (IJAIA), 9(3), 19-36 (2018).
[11] Bursztein, E., Martin, M. and Mitchell, J., "Text-based captcha strengths and weaknesses," Proc. of the 18th ACM Conf. on Computer and Communications Security CCS'11, New York, USA, 125-137 (2011).
[12] Zhou, T., Yuan, F. and Zhuang, X., [Simplest Data Mining], Publishing House of Electronics Industry, Beijing, chapter 3, 31-45 (2020). (in Chinese)
[13] Manning, C. D., Raghavan, P. and Schütze H., [Introduction to Information Retrieval], Beijing Posts & Telecom Press), Beijing, chapter 14, 200-220 (2010). (in Chinese)
[14] Lin G., Liang Y., Fu X., Chen G., Cai S., "Design of a daily brief business report generator based on web scraping with KNN algorithm," Journal of Physics: Conference Series, 1345(5), 1-7 (2019).
[15] Cover, T. M. and Hart, P. E., "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, 13(1), 21-27 (1967).
[16] Chellapilla, K. and Simard, P., "Using machine learning to break visual human interaction proofs (HIPs)," Advances in Neural Information Processing Systems, 17, 1-8 (2004).
[17] Li Q. J., Mao Y. B. and Wang Z. Q., "A survey of CAPTCHA technology," Journal of Computer Research and Development, 49(3), 469-480 (2012).
[18] Booth, R. R. and Phelps, M. J., [Image Binarization], US 2016/0014300 A1, 05/17/2016 (2016).
[19] Chaki, N., Shaikh, S. H. and Saeed, K., "A comprehensive survey on image binarization techniques," Studies in Computational Intelligence, 560, 5-15 (2014).
[20] Lin, Z., [Principles and Applications of Big Data Technology], Beijing Posts & Telecom Press, Beijing, 3rd edition, chapter 15, 286-296 (2021). (in Chinese)
[21] Broder, A. Z., "On the resemblance and containment of documents," Proc. of the Compression and Complexity of Sequences, 21-29 (1997).
[22] Hart P. E., "The condensed nearest neighbor rule," IEEE Transactions on Information Theory, 18, 515-516 (1968).
[23] Gaede, V. and Gunther, O., "Multidimensional access methods," ACM Computing Surveys, 30(2), 170-231 (1998).