Breaking reCAPTCHA: A Holistic Approach via Shape Recognition

Paul Baecher, Niklas Büscher, Marc Fischlin, Benjamin Milde

Breaking reCAPTCHA: A Holistic Approach via Shape Recognition

Paul Baecher, Niklas Büscher, Marc Fischlin, and Benjamin Milde

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. CAPTCHAs are small puzzles which should be easily solvable by human beings but hard to solve for computers. They build a security cornerstone of the modern Internet service landscape, deployed in essentially any kind of login service, allowing to distinguish authorized human beings from automated attacks. One of the most popular and successful systems today is reCAPTCHA. As many other systems, reCAPTCHA is based on distorted images of words, where the distortion system evolves over time and determines different generations of the system. In this work, we analyze three recent generations of reCAPTCHA and present an algorithm that is capable of solving at least 5% of the challenges generated by these versions. We achieve this by applying a specialized variant of shape contexts proposed by Belongie et al. to match entire words at once. In order to handle the ellipse shaped distortions employed in one of the generations, we propose a machine learning algorithm that virtually eliminates the distortion. Finally, an improved shape matching strategy allows us to use word dictionaries of a reasonable size (with approximately 20,000 entries).

1 Introduction

CAPTCHAs—Completely Automated Public Turing tests to tell Computers and Humans Apart—are used to prevent automated use of online services. Typically, this is a challenge/response protocol where the user is, for example, required to read and submit a heavily distorted image of a word. While there exists an unmanageable number of such systems, upon closer examination most of them turn out to be insecure against automated solvers. This usually happens as soon as the system is used on a popular website and exposed to many users. Since CAPTCHAs are omnipresent and constitute an integral security mechanism in today's Internet services, this situation is highly unsatisfying. It is thus even more important to investigate promising systems and determine the level of security they provide.

The reCAPTCHA System. In contrast to the vast number of broken schemes, one particular implementation, reCAPTCHA [1], has been successfully in use for several years now. Two distinct key features are seemingly responsible for this comparatively long lifespan. First, the generation algorithm of reCAPTCHA

challenges is proprietary and not public, meaning that challenges are provided via a centralized infrastructure. This allows reCAPTCHA to adjust their system at any given time, for example in the event of a successful attack on the system. Since such adjustments immediately affect all users of reCAPTCHA, no legacy installations exist that could still be prone to the attack. Moreover, since the algorithm is kept secret, it is tedious to analyze the variance of the challenges. Second, every challenge is guaranteed to have a minimum level of resistance against common OCR techniques. This is due to the way challenges are generated: instead of artificially rendered and distorted characters, reCAPTCHA uses words on which two OCR systems failed; a by-product of digitizing huge amounts of text.

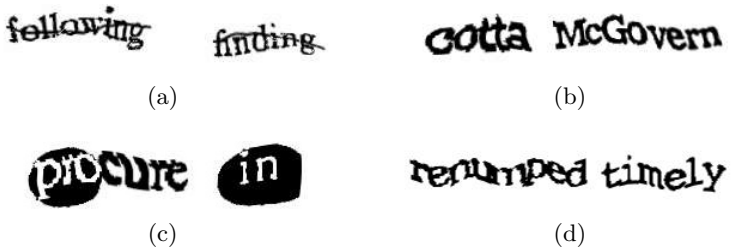
The answer to such challenges is thus inherently unknown to the system. In order to verify the user's response, reCAPTCHA follows a statistic approach and presents two words in each challenge. One word is the unknown *scan word*, the other word is a known *verification word*. As long as the user provides the correct answer for the verification word, the response is considered correct and the given solution to the scan word is recorded. It is important to observe that the solution to the scan word, when viewed separately, is not relevant to pass the test. This is a critical detail when it comes to estimating success rates. Ideally, both classes of words should be indistinguishable, but this is not the case. For instance, it is entirely possible to have an algorithm that reliably recognizes scan words but performs poorly on verification words. Clearly, such an algorithm will not be suitable to break the system.

The centralized system makes it also hard to analyze the security of reCAPTCHA, since there are no public distinct and explicit versions of the generation algorithm. Hence, subtle modifications and revisions of the algorithm are not necessarily visible to the user. It is, however, possible to identify a set of major generations as shown in Figure 1. In the first generation, for example, words are struck through with a horizontal line; the third generation adds inverted ellipse-shaped blobs. Although the challenges of the second and fourth generation look similar, the distortion of the latter is more regular and exhibits a compact mathematical description. Furthermore, it seems that the fourth generation also uses less common words which tend to be excluded from dictionaries.

In this work, we focus on the security of the third and fourth[1] generations of reCAPTCHA.

Our Contributions. We present an implementation to break the latest generation of reCAPTCHA using shape contexts [2]. As opposed to previous work in this area, our algorithm is quite efficient with reasonably sized dictionaries of 20,000 words (i.e., shapes). To our best knowledge, this is also the first attempt to break reCAPTCHA using shape contexts and, in particular, to do this in a holistic fashion where entire words are matched atomically at once. Since reCAPTCHA is based upon the hardness of character recognition our results may therefore also stimulate new approaches for OCR.

[1] This is the latest version of reCAPTCHA as of November 2010.



(a) (b) (c) (d)

Fig. 1: Major generations of reCAPTCHA, in chronological order.

In order to attack the third generation of reCAPTCHA, which includes an ellipse-shaped distortion object, we propose a machine learning framework that is able to detect and remove this distortion almost entirely. This allows us to treat challenges from the second to the fourth generation uniformly with one algorithm since the challenges of these generations are then sufficiently similar. Finally, we employ a novel method to quickly match a given query shape against a large list of dictionary words. This is done by taking the first and last character of the challenge word into account (which are considerably easier to segment) and then subsequently reducing the search space logarithmically.

2 Related Work

Attacks on reCAPTCHA. As mentioned before, reCAPTCHA has been immune against major attacks for a long time. Wilkins [16] announced in 2009 to have broken the first generation in Figure 1 of the reCAPTCHA system with a success rate of 5% (conservative estimate) to 17.5% (optimistic estimate) in early 2008. The two types of estimates stem from the fact that reCAPTCHA offers two classes of words for which it only knows the solution to the verification words. In 5% of the cases Wilkins got both words on his 200 test data right, in another 25% he got only one word correctly. Making the optimistic assumption that in half of these cases this is indeed the verification word yields the bound of 17.5%.

Wilkins essentially uses three techniques for his attack: the distortion line is removed by applying some combination of erode/dilate matrices, then he runs an OCR program, and finally he uses a dictionary to make a guess for the word. This is iterated for several matrices and the most likely answer about all these runs is output. Wilkins also ran tests again the second generation of reCAPTCHA (without the distortion line). Here, he was able to solve in a data set of 40 about 5% of the puzzles, simply running an OCR program. He concluded that the new system should be even weaker than the previous generation but this claim seems to be hard to formally back up by the restricted experiments.

At DefCon 2010, and independently of our work, Houck [7] announced successful attacks on the third generation reCAPTCHA (with the ellipse). He estimates to achieve a success rate of about 10%, based on experiments on about

5,000 CAPTCHAs. However, this optimistic estimate is again based on the assumption that, in about 75% of the cases, the solution for one word only is indeed for the verification word. Houck's approach is based on removing the ellipse, segmenting the word into characters via "dips" in the upper margin—making this attack fundamentally different from our holistic approach—and running an OCR program.

Soon after these attacks became public, reCAPTCHA changed to the fourth generation. Houck also briefly discussed extensions of his attack to this generation, yielding an estimated success rate of 30%. In this sense, reCATPCHA became actually weaker.

Segmentation vs. Recognition. Text-based CAPTCHAs—the overwhelming majority of today's systems—present strings of letters and digits, possibly forming words of a natural language. These strings are rendered onto a rastered image and presented to the user. The exact process of how strings are rendered is crucial to the security of the system. Specifically, it is vital that subsequent characters overlap not only by their bounding box, but also touch each other such that the string forms one large connected component. This is absolutely necessary for security because, as Chellapilla et al. discuss in [5], *recognizing* single characters is a solved problem where computers even outperform humans. The task of *segmentation* on the other hand, where one is interested in partitioning a string in terms of a connected component into its individual characters, is still considered fairly resistant against automated attacks [4].

Shape Contexts. Recognition and comparison of shapes is a recurring problem in computer vision. Typically, shapes are transformed to a compact feature vector or descriptor which allows for fast comparison under slight variations of shapes. Belongie et al. propose a descriptor called *shape contexts* in [2] and efficient retrieval methods in [12]. A shape is described by a set of histograms, where each histogram corresponds to the distribution of vectors from one contour point to all other contour points. Shape matching against the database is then done by matching the sets of histograms of two shapes, where histograms are compared with a χ^2-metric. Clearly, this technique can be used to match characters and digits in a CAPTCHA. In [14] Mori and Malik successfully attack the EZ-Gimpy CAPTCHA using shape contexts; their approach is able to match the correct word in 93% of the time. Lladós et al. investigate this technique to spot words in historical documents from a predefined set of keywords in [9]. Although this can be viewed as a direct application to the OCR task, it is not designed to digitize entire documents. Rather, they are interested in metadata extraction by looking for specific words. Unfortunately, they do not mention the size or order of magnitude of their reference dictionary.

3 Our Techniques

In this section, we present our framework to break the recent reCAPTCHA generations 2–4. Our system can be divided roughly into two phases. In an

offline "learning" phase, we create synthetic challenges based on a dictionary of English words. Each challenge is transformed to a descriptor that consists of a set of shape context histograms. We then create a database that contains all histograms for all words in the dictionary. A given (real) challenge in the online phase is transformed exactly the same way and the resulting histograms are matched against the database; the closest match is the output of our algorithm.

Note that this basic version of our attack operates on entire words only, thus circumventing the task of segmentation. This technique is commonly known as holistic word recognition [6,10,11,8]. One can interpret this as a recognition task on a large alphabet, i.e., where entire words are the letters.

Figure 2 gives an overview of the transformation process from challenge images to descriptors.

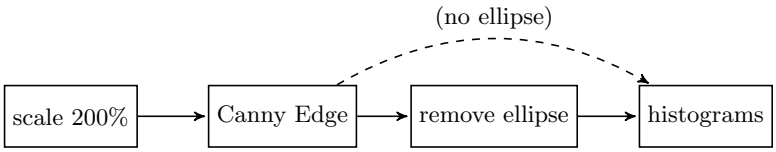


Fig. 2: High-level overview of the descriptor creation.

3.1 Database Creation

In order to create the database of reference shapes, we would ideally use actual challenges generated by reCAPTCHA. However, since reCAPTCHA is proprietary, we neither have access to this data nor to the underlying dictionary[2]. To overcome this limitation, we select a reasonably sized dictionary of frequently used words and create our own reference shapes. In order to mimic the real challenges which originate from printed text, our system uses a standard serif font face to render synthetic challenges. Even though this is only a rough and imperfect approximation, its resemblance is sufficient to be covered by the shape context variance. Note that these synthetic challenges are only used for the database; the final performance measurements are derived from real reCAPTCHA challenges.

3.2 Preprocessing

Verbatim challenge images generated by reCAPTCHA contain too much noise and redundancy for shape contexts. This includes JPEG compression artifacts (noise) and the inner area of the stems of characters (redundancy). Therefore we

[2] In fact, since the words originate from scanned text where OCR failed, the exact dictionary is not even known to the reCAPTCHA system.

(a) (b) (c) (d)

Fig. 3: Ellipse center estimation. After 7 iterations of erosion only one connected component is left (b). After 58 iterations of dilation only a few pixels close to the center are left over (c). Figure (d) shows these pixels in relation to the original image.

apply a sequence of preprocess steps as depicted in Figure 2. The first step scales the image to 200% of its original size. A subsequent binarization operation then eliminates compression artifacts. Since only the contour of characters is relevant to their shape, we run the Canny edge detector [3] to obtain a contour image. At this point the third generation of reCAPTCHA needs another step to remove the ellipse shaped distortion object which we describe in the next section.

An observant reader may argue that the initial scaling step is technically not necessary since it cannot increase the information available in the image. However, since this is followed by a highly lossy binarization operation, we *reduce* the loss by this measure. Experimental evaluations confirm this by exhibiting higher success rates if the scaling step is performed.

3.3 Ellipse Elimination

Third generation reCAPTCHA challenges (see Figure 1c for an example) contain an ellipse-shaped object under which the colors are inverted. It seems that this object is first drawn as a perfect ellipse and then, along with the challenge word, transformed. Sometimes it is also cut off, apparently because the ellipse extends—or extended prior to the transformation—over the border of the image. Nevertheless, the area still resembles roughly an ellipse. As mentioned in Section 2, precisely this property has been exploited successfully in [7]. We take a slightly different approach here. Instead of trying to directly fit an ellipse onto a set of points, we run a machine learning algorithm that classifies pixels as "ellipse" and "not ellipse." We now describe this mechanism in greater detail.

Ellipse Center Approximation. In order to classify pixels we first require a reference point relative to the ellipse. We use the center of the ellipse for that and present an algorithm to estimate this point. Our algorithm stems from the observation that wherever the ellipse is located, a huge number of black pixels concentrate. It operates as follows. First, repeat the morphological erode operation until only one single connected component of black pixels is left over. Now repeat the dilate operation until the entire image consists of white pixels only. Undo one dilate iteration and finally calculate the center of the remaining

black pixels; this is the output of the estimation algorithm. See Figure 3 for an example of the algorithm's operation.

Features. Once the ellipse center has been estimated, a number of features relative to this center p is calculated for every black pixel q_i in the original contour image. These features, arranged in a vector, include amongst others

- the distance and angle from p to q_i,
- the tangent of the edge in q_i,
- pixel density on a line from p to the center point,
- pixel density in the neighborhood of p,

and a number of variations of these features.

Classification Training. Once these features have been calculated, we are interested in learning the mapping that maps each feature vector to its correct class ("ellipse", "not ellipse"). For this we use standard machine learning techniques. To obtain labeled training data, we classified a set of preprocessed challenges manually by removing the ellipse contour in an image editor. Using OpenCV's boosting algorithm with weak decision tree classifiers on this data then yields a strong classifier.

While this already gives a solid classification result (see Figure 4, left column), there is still room for improvement. For example, each classification decision is made only locally and independently of spatially surrounding classifications. This gives away prior knowledge such as the geometric shape of an ellipse. In order to take this into account, we employ a cascade of classifiers where the ith iteration makes use of knowledge obtained from the $(i-1)$th iteration. Moreover, in each iteration, we calculate a feature that measures the distance to a fitted ellipse for all ellipse-classified pixels.

Fig. 4: Cascaded ellipse pixel classification. First row: pixels classified as "not ellipse," second row: pixels classified as "ellipse." From left to right: Classification after iteration 1, 4, and 9.

Accuracy The fraction of pixels that are classified correctly is denoted by accuracy. We estimate this value with a 10-fold cross validation using 150 weak classifiers and reach a total accuracy of 91.5% after 9 cascade iterations. It takes roughly two hours to train this classifier and less than 300 milliseconds to classify a new example on standard off-the-shelf hardware. Figure 4 presents a classification instance after different iterations of the cascade.

3.4 Shape Contexts

Once the challenge images are preprocessed, possibly including the ellipse removal step, we are ready to obtain a compact description of the word. As mentioned earlier, our attack uses shape contexts to represent the rendered words.

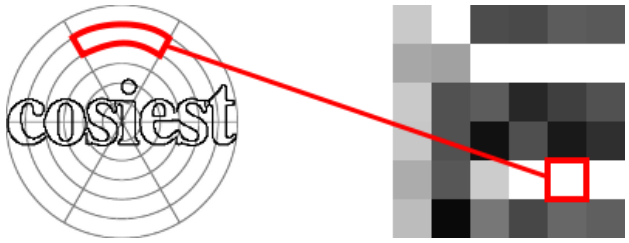


Fig. 5: Histogram bins and the corresponding angle/distance histogram for the center point of the contour line of the word "cosiest."

The key idea of shape contexts is as follows. Let $p_1, \ldots, p_n \in \mathbb{R}^2$ be the points that form the contour line of a shape. For an arbitrary point p_i, called reference point, there are $n-1$ vectors $v_{i,j}$ that describe the location of the other points relative to p_i. Consider now a histogram of the distribution of these vectors $v_{i,j}$ in a polar system[3] centered at p_i. This two dimensional histogram—consisting of angle/distance bins—constitutes a compact but lossy description of the shape with respect to reference point p_i and is called its shape context. Figure 5 visualizes the histogram bins and the resulting histogram when this transformation is applied to a rendered word. Note that there are n such histograms per shape, one histogram for each contour line point.

To measure the similarity between two shapes, one could simply match their corresponding sets of histograms, i.e., find a one-to-one mapping between both sets such that the sum of the distances between each two histograms is minimal with respect to some distance metric. However, it is inefficient and highly redundant to do this on the full set of contour line points. Thus, it makes sense to work

[3] We note that shape contexts are usually associated with log-polar systems (as opposed to polar-linear systems). In our experiments however, we were able to obtain better results with linear distance.

with a randomly selected fixed-size subset consisting of, say, 100 histograms. Furthermore, it is not strictly necessary to require a one-to-one mapping between two sets of histograms. Simply selecting the closest match is an acceptable strategy if additional constraints are introduced. One such constraint is the location of the corresponding reference points; requiring a maximal distance here ensures that no points in completely different locations are matched.

In order to further improve the descriptive quality of shape contexts, we use an extended concept called *generalized* shape contexts as proposed in [13] that allows for arbitrary features. Here, Mori et al. additionally record the average tangent of shape points in each histogram bin. This results in a richer description of the shape at the cost of a second set of histograms.

3.5 Efficient Word Matching

We now turn to the online phase of our attack. Given a database of associated shape contexts for each word, our goal is to find the most similar shape for a new query shape. A naive approach comes to mind immediately: compare the query shape with each database shape and output the closest database shape in terms of the distance function. This, however, results in enormous computational cost. Recall that each shape description consists of a set of histograms and that shapes are dictionary words in our case. Matching the histograms of two shapes results in quadratic complexity; a reasonable dictionary size is $20,000$ words. In order to distinguish the many similar words from such a dictionary, the number of reference points/histograms needs to be accordingly high.

To manage this complexity, we propose a search algorithm along the lines of "fast pruning" described in [13]. The general strategy of our algorithm is to start with the full set of database shapes and perform a crude, but fast, comparison against the query shape. Then, the algorithm prunes the most dissimilar shapes from the working set and increases the exactness of the search. Repeated application of this step results in a logarithmic search space reduction. As the shapes become more similar, more time is invested in the comparison. Finally, as soon as the number of shapes in the working set drops below a certain threshold, the algorithm switches to the naive search strategy and outputs the closest match.

The exactness of the search is controlled by the number of reference points used for comparison. For a given number of reference points, the algorithm draws a random subset from all available reference points. A noteworthy consequence is that the algorithm is probabilistic, but this is not so bad because the closest match is not always the correct solution.

Another CAPTCHA-specific pruning strategy which greatly reduces the search space makes use of the fact that the first character and last character are considerably easier to segment. It is immediately clear where the first character starts and the last character ends. A simple and basic approach is to consider an averaged fixed-width section from the start/end of the word. If a character can be detected within this area, a huge portion of the search space is superfluous and thus pruned. In fact, it is already helpful to be able to restrict these key

characters to a small set. This is done by employing the shape context matching framework for single characters and selecting the best k matches.

4 Results

Data acquisition. Recall that reCAPTCHA is a proprietary and closed system. This complicates the acquisition of (labeled) challenge/response pairs that are needed for the performance evaluation. One of our methods to collect data is to have humans solve a number of reCAPTCHA challenges and, in the background, record the solution. The advantage of this tedious method is that a human can quickly learn the difference between verification and scan words by close observation. This means that we can deliberately provide a wrong solution for the suspected scan word. If reCAPTCHA confirms this hypothesis by accepting the response, we can be certain that the other word was indeed the verification word. Consequently, we obtain a data set that is not only labeled, but consists also of verification words only, allowing us to derive true performance measurements. In contrast, many reported figures on reCAPTCHA attacks are in fact estimations where the—possibly hidden—underlying assumption is that the attack works equally well on scan and verification words.

Database creation. To build the database of reference words we use a word list prepared by Keith Vertanen [15] which is the intersection of 10 popular word lists. This list contains 22,282 words from the English language. The artificial challenges are then rendered using the Times typeface with negative inter character distance to reflect the overlap situation of real challenges. Shape contexts are created for a 6×6 histogram, i.e., 6 angle bins times 6 distance bins.

Final Results. We stress that our results have been collected from verification words only and thus reflect precisely the success rate of a real attack. See Figure 6 for detailed results. The dictionary success rate in this figure is the (ideal) success rate of our attack *if the challenge word is present* in our dictionary. We are also able to obtain substantially shorter run times in exchange for slightly lower recognition rates.

reCAPTCHA generation	2	3	4
Test set size	496	1005	301
Total success rate	12.7%	5.9%	11.6%
Run time	24.5s	17.5s	15.4s
Dictionary success rate	22%	10.43%	23.5%
First character detected	90.2%	73.2%	84.6%

Fig. 6: Experimental results of our implementation.

5 Conclusions

The reCAPTCHA system has been one of the few systems achieving the right balance between usability and security. So far. With its increasing popularity reCAPTCHA has become a major target and the recent attacks reveal significant cracks. Still, because of the centralized system reCAPTCHA allows to switch to a new generation instantaneously. While the concrete attacks may then become ineffective the attack techniques nonetheless improve.

Our attacks for example, achieving a success rate of 5%, show that holistic approaches are feasible, whereas most other attacks are based on segmentation. This is interesting because many systems and techniques so far have been designed to thwart segmentation, e.g., striking out the word. We note that our attacks have not been optimized and thus leave space for improvements. An example is the combination of our holistic approach with partial segmentation, which is—in its current version—only a crude proof of concept of the general technique.

However, we also stress that resistance against automated attacks is not the only concern for CAPTCHAs. Two other dimensions are usability, the ability of humans to solve the CAPTCHA easily, as well as practicality, describing the ability to realize the CAPTCHAs efficiently. For instance, from the security perspective, dictionary-based CAPTCHAs should be used cautiously as they facilitate attacks significantly. It must be said, though, that using dictionaries supports humans in recognizing words. Another worthwhile point is that reCAPTCHA is based on the idea that solving a CAPTCHA helps digitizing books. This idea may incite users to solve such otherwise unpopular puzzles, thus improving the overall acceptance of CAPTCHAs.

Overall, the recent attacks on reCAPTCHA somehow leave us in a vague state. It remains an open problem if there exist CAPTCHAs which are simultaneously secure, usable, and practical. Given the status of CAPTCHAs in modern login services, a CAPTCHA system meeting all these requirements is of great demand.

On a slightly positive note, however, even though our results indicate that the security of yet another CAPTCHA system has become dubious, there is also an upside in the particular case of reCAPTCHA. By design, any system that breaks reCAPTCHA is a step towards better OCR software.[4] Our results indicate that Shape Contexts could be a valuable fallback solution in the domain of character recognition.

Acknowledgements

We thank the anonymous reviewers for valuable comments. Paul Baecher and Marc Fischlin are supported by the Emmy Noether Grant Fi 940/2-1 of the German Research Foundation (DFG).

[4] It must be said, though, that a successful attack may only achieve a recognition rate of say, 10% of the challenges, which is too low for a full-fledged OCR program.

References

1. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: reCAPTCHA: Human-based character recognition via web security measures. Science 321(5895), 1465–1468 (2008) Cited on page 1.
2. Belongie, S., Malik, J., Puzicha, J.: Shape context: A new descriptor for shape matching and object recognition. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) NIPS. pp. 831–837. MIT Press (2000) Cited on pages 2 and 4.
3. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8, 679–698 (November 1986), `http://portal.acm.org/citation.cfm?id=11274.11275` Cited on page 6.
4. Chellapilla, K., Larson, K., Simard, P.Y., Czerwinski, M.: Building segmentation based human-friendly human interaction proofs (HIPs). In: HIP. Lecture Notes in Computer Science, vol. 3517, pp. 1–26. Springer-Verlag (2005) Cited on page 4.
5. Chellapilla, K., Larson, K., Simard, P.Y., Czerwinski, M.: Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In: CEAS (2005) Cited on page 4.
6. Govindaraju, V., Krishnamurthy, R.K.: Holistic handwritten word recognition using temporal features derived from off-line images. Pattern Recognition Letters 17(5), 537–540 (1996) Cited on page 5.
7. Houck, C.W.: Decoding recaptcha. `http://www.n3on.org/projects/reCAPTCHA/docs/reCAPTCHA.docx` (2010) Cited on pages 3 and 6.
8. Lavrenko, V., Rath, T.M., Manmatha, R.: Holistic word recognition for handwritten historical documents. In: DIAL. pp. 278–287. IEEE Computer Society Press (2004) Cited on page 5.
9. Lladós, J., Roy, P.P., Rodríguez, J.A., Sánchez, G.: Word spotting in archive documents using shape contexts. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (eds.) IbPRIA (2). Lecture Notes in Computer Science, vol. 4478, pp. 290–297. Springer-Verlag (2007) Cited on page 4.
10. Madhvanath, S., Govindaraju, V.: Contour-based image preprocessing for holistic handwritten word recognition. In: ICDAR. pp. 536–539. IEEE Computer Society Press (1997) Cited on page 5.
11. Madhvanath, S., Govindaraju, V.: The role of holistic paradigms in handwritten word recognition. IEEE Trans. Pattern Anal. Mach. Intell. 23(2), 149–164 (2001) Cited on page 5.
12. Mori, G., Belongie, S., Malik, J.: Shape contexts enable efficient retrieval of similar shapes. In: CVPR (1). pp. 723–730. IEEE Computer Society Press (2001) Cited on page 4.
13. Mori, G., Belongie, S.J., Malik, J.: Efficient shape matching using shape contexts. IEEE Trans. Pattern Anal. Mach. Intell. 27(11), 1832–1837 (2005) Cited on page 9.
14. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: CVPR (1). pp. 134–144. IEEE Computer Society Press (2003) Cited on page 4.
15. Vertanen, K.: Words in 10 lists. `http://www.keithv.com/software/` (2010) Cited on page 10.
16. Wilkins, J.: Strong CAPTCHA guidelines v1.2. `http://www.bitland.net/` (2009) Cited on page 3.