Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Defeating line-noise CAPTCHAs with multiple quadratic snakes

Yoichi Nakaguro[a], Matthew N. Dailey[b], Sanparith Marukatat[c], Stanislav S. Makhanov[a,*]

[a] Sirindhorn International Institute of Technology, Thammasat University, Thailand
[b] Computer Science and Information Management, Asian Institute of Technology, Thailand
[c] National Electronics and Computer Technology Center, Thailand

ARTICLE INFO

Article history:
Received 22 August 2012
Received in revised form
19 March 2013
Accepted 6 May 2013

Keywords:
CAPTCHA
Segmentation
GVF snake
OpenCV
OCR

ABSTRACT

Optical character recognition (OCR) is one of the fundamental problems in artificial intelligence and image processing, but recent progress in OCR represents a security challenge for Web sites that throttle requests with image based CAPTCHAs (Completely Automated Public Turing Tests to Tell Computers and Humans Apart). A CAPTCHA is challenge-response test placed within web forms to determine whether the user is human. Unfortunately, algorithms capable of solving image based CAPTCHAs can be used to create spam accounts and design malicious denial of service (DoS) attacks, causing financial and social damage. The problem of defeating digital image CAPTCHAs is thus twofold. On the one hand, it is an important problem in artificial intelligence and image processing. On the other hand, publicly available CAPTCHAs that are not tested against state of the art machine recognition algorithms may make the systems vulnerable to attack by software bots.

This paper considers a very important subclass of text CAPTCHAs, those characterized by salt and pepper noise combined with line (curve) noise. Thus far, attacks on CAPTCHAs with this type of noise have used relatively simple image processing methods with some success, but state-of-the-art segmentation methods have not been fully exploited. In this paper, we propose and benchmark two strong segmentation methods. The first method is a modification of a multiple quadratic snake proposed for road extraction from satellite images. The second competing method is a boundary tracing routine available in the OpenCV open source library.

A first numerical experiment indicates excellent accuracy for both methods. A second experiment on human recognition shows that the CAPTCHAs used in the study are already near the threshold of being too hard for humans. Finally, a third numerical experiment presents a more difficult set of CAPTCHAs with the addition of anti-binarization methods. The snake-based method is shown to be more resilient to anti-binarization schemes than boundary tracing and state-of-the art projection-based attacks on CAPTCHAs.

Since CAPTCHAs corrupted by small line noise are shown to be difficult for humans and relatively easy for our algorithm, CAPTCHA designers should introduce more challenging distortions into their CAPTCHAs, lest the security of systems based on them be compromised.

* *Corresponding author*. Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12000, Thailand. Tel.: +66 2 501 3505; fax: +66 2 501 3524.
E-mail addresses: makhanov@siit.tu.ac.th, smakhanov@yahoo.com (S.S. Makhanov).

0167-4048/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2013.05.003

1. Introduction

CAPTCHAs (Completely Automated Public Turing Tests to Tell Computers and Humans Apart) are tests that are easy for humans but difficult for computers (von Ahn et al., 2003). CAPTCHAs can be used in any system that benefits from distinguishing between humans and software bots. The first known CAPTCHA was developed at Digital Equipment Corporation's Systems Research Center and used by AltaVista (Lillibridge et al., 2001) to protect the search engine's URL submission form. Early work at Carnegie Mellon University was deployed by Yahoo! to prevent automated spam bots from signing up for large numbers of free accounts (von Ahn et al., 2004).

Besides protecting data entry forms and account registration systems from bots, other applications include protecting recommender systems and polls from skewing by bots, increasing security and verifiability in e-voting systems (Oppliger et al., 2008; Simha and Vora), protecting information in Web pages from search engine bots and screen scrapers (Carnegie Mellon University, 2000), preventing email and SMS spam (Dwork and Naor, 1992; Shirali-Shahreza and Movaghar, 2007; He et al., 2008), deterring dictionary attacks in password authentication systems (Pinkas and Sander, 2002; Xu et al., 2003; Namprempre and Dailey, 2007), securing online e-commerce transactions (Steeves and Snyder, 2007), and acquiring human knowledge (da Silva and Garcia, 2007).

There are many types of CAPTCHAs. The most common type of CAPTCHA displays a degraded image of a word or character sequence to the human, who responds by typing the character sequence he or she sees (Coates et al., 2001; Chewa and Baird, 2003; Baird et al., 2005; Baird and Bentley, 2005; von Ahn et al., 2008). Digitized characters are not the only things humans can potentially recognize more accurately than computers. Other image-based CAPTCHAs have been proposed (Elson et al., 2007; Datta et al., 2005; Rusu and Govindaraju, 2004, 2005; Gossweiler et al., 2009; Chow et al., 2008; Chew and Tygar, 2004; Misra and Gaj, 2006). Since digital images may not be suitable for all communication media, other types of CAPTCHAs use sound clips (von Ahn et al., 2008; Holman et al., 2007), synthetic handwritten characters (Achint et al., 2009) or text and text graphics (Godfrey, 2002; Namprempre and Dailey, 2007; Ximenes et al., 2006), and others require the user to perform a specific task using a stylus or finger such as connecting noisy dots on a tablet (Shirali-Shahreza and Shirali-Shahreza, 2006).

Since so many different Internet applications base some aspect of their security models on the CAPTCHA properties of being easy for humans and difficult for computers, it is critically important that CAPTCHAs truly have the assumed properties. Not long after CAPTCHAs began to appear on Web sites, artificial intelligence and computer security researchers began to test these assumptions, taking up the challenge of breaking deployed CAPTCHAs.

There are two major reasons to attack public CAPTCHAs. First, a program that can pass a CAPTCHA with high probability is effectively solving a hard artificial intelligence problem, so progress in defeating CAPTCHAs designed to defeat machine recognition systems advances the state of the art in artificial intelligence. Second, and possibly more important in terms of immediate consequences, system designers simply use CAPTCHA libraries presuming the tests to satisfy the CAPTCHA properties of being easy for humans and hard for machines, so publicly available CAPTCHAs that have not been tested against state of the art machine recognition algorithms may make the systems that use them vulnerable to attack by software bots without the designer's knowledge. Through a continuous process of breaking then improving CAPTCHAs, we expect the community to eventually arrive at a set of design principles or guidelines for ensuring success against known state of the art attacks.

This paper focuses on the most widely used form of CAPTCHA, the digitized character CAPTCHA, in which a text string is digitized and distorted before being presented to the user. Studies of human recognition conclude that these distortions are the "ideal choices for usable and secure CAPTCHA distortion techniques" (Lee and Hsu, 2011). Examples of sites using these types of CAPTCHAs include prominent social networking websites such as "Tagged" (Tagged, Inc., 2012), "Friendster" (Friendster, Inc., 2012), "MetroFLOG" (metroFLOG, 2012), "Netlog" (Netlog, 2012), and "VKontakte" (VKontakte, 2012), as well as expert electronic money companies such as Moneybookers (Moneybookers, 2012) (see also (Hernandez-Castro and Ribagorda, 2010)). Many CAPTCHAs in this category are generated using widely-used open source libraries or public services.

The experiments reported in this paper empirically evaluate the utility of two strong segmentation techniques for attacking text CAPTCHAs with line noise. It first proposes a modification of a multiple quadratic snake method originally used for extraction of roads and other network structures from digital images, and the second method uses the same preprocessing as the snake method followed by Otsu binarization and standard contour tracing.

As previously mentioned, in this paper we evaluate two methods, the first of which is based on multiple quadratic snakes. This method builds on the large literature on active contours, which provide an elegant framework for optimal estimation of object boundaries. Snake-based segmentation is potentially powerful for defeating CAPTCHAs. It does not make any assumptions regarding the size and position of the characters. The characters can be rotated, skewed, and scaled without affecting the accuracy of the segmentation. The resulting contour can be evaluated by fast pattern recognition methods designed for written or typed characters. If it is possible to obtain accurate segmentation via snakes, classification will be fast and efficient.

In addition to the quadratic snake model, as a baseline for comparison, this paper also benchmarks the capabilities of a simpler contour extraction approach based on the same preprocessing method as the snake-based method followed by simple Otsu binarization and foreground object tracing. The tracing method is the standard algorithm implemented by the OpenCV `cvFindContours` function (Suzuki and Abe, 1985).

In an extensive empirical evaluation on images generated by a widely used CAPTCHA API (Version 6.x-2.4) developed under the Drupal CAPTCHA project (2012), we first find that the quadratic snake based method and the standard contour tracing method are both extremely accurate at character segmentation, with segmentation accuracies of 90% and 86% compared to the ground truth, respectively. We further find that the quadratic snake based method is superior to standard contour tracing under low contrast conditions. Finally, in terms of recognition ability, when the CAPTCHA segmentation methods are combined with a standard general-purpose character OCR algorithm, we obtain accuracies of 73% and 70%. Note that to deem a scheme insecure the attacking system only needs to reach 1% precision (Bursztein et al., 2011).

Although further improvements in accuracy are no doubt possible, at the reported levels of accuracy, we believe that CAPTCHA segmentation under conditions of "small line" noise should now be considered a solved problem and that CAPTCHA designers should introduce more challenging distortions into their CAPTCHAs, lest the security of systems based on them be compromised. We hope that the results reported in this paper will spur CAPTCHA designers to further improve the segmentation resistance of their methods.

The rest of the paper is organized as follows. In Section 2, we discuss related work from the literature on CAPTCHA attacks and our specific segmentation methods. In Section 3, we provide a detailed account of our algorithms and techniques for CAPTCHA segmentation. In Section 4, we report on the design of and results of a series of experiments to evaluate the proposed CAPTCHA segmentation in comparison with other methods and humans. In Section 5, we discuss the impact of the research. Finally, in Section 6, we provide concluding remarks and pointers to future work.

2. Related work

Mori and Malik (2003) were the first researchers to publish attacks on visual CAPTCHAs. They applied shape context matching to the EZ-Gimpy and Gimpy CAPTCHAs with high success rates. Yan and Ahmad (2008) show that a relatively simple processing stream comprising of binarization, fixing of broken characters, removal of simple arcs, and breaking of connected characters could reliably segment the characters in the Microsoft digitized character CAPTCHA. The Microsoft CAPTCHA turned out to be breakable even though it was specifically designed by experts to be segmentation-resistant. Other researchers have broken other types of image CAPTCHAs such as the Asirra object recognition CAPTCHA using standard machine learning techniques (Golle, 2008) and audio CAPTCHAs based on text-to-speech synthesis using standard speech recognition techniques (Chan, 2003).

Researchers at Stanford University (Bursztein et al., 2011) have recently published the most comprehensive study to date of techniques for breaking text CAPTCHAs along with a series of lessons learned for CAPTCHA designers to improve the resilience of their CAPTCHAs under attack (Zhu et al., 2010; Qu and Wu, 2012). They analyze the three main steps of CAPTCHA processing: pre-processing, segmentation, and recognition. The pre-processing stage attempts to make the CAPTCHA easier to segment, through noise removal, color removal, and so on. The segmentation step attempts to isolate the text characters from the background and noise. The recognition step then attempts to classify the segmented image into a character string. They corroborate what several researchers have pointed out, that the main challenge in attacking digitized character CAPTCHAs is the character segmentation problem, in which we aim to separate the text from the background and any noise present in the image. Once this is done, standard machine learning based character recognition techniques are extremely effective. This means that it is essentially possible to reduce the digitized text CAPTCHA problem to the problem of text segmentation in the presence of noise.

One of the most common types of distortions used in digitized text CAPTCHAs is the introduction of line noise. The idea is to overlay the noisy, distorted characters with lines and curves of various thickness. These lines interfere with segmentation methods because they may be confused with the actual target characters or join multiple target characters into a single foreground object. Bursztein et al. (2011) distinguish between "big" and "small" lines, for which different strategies are recommended. They find that small lines can be removed by binarization then Gibbs denoising with character reconstruction, and that big lines can be removed by Hough transform based line finding followed by simple heuristics for removing foreground pixels on Hough lines.

These attacks and other attacks on CAPTCHAs with line noise, while successful on some of today's popular CAPTCHAs, use relatively simple image processing techniques. To date, state of the art line noise removal techniques have not been fully explored on CAPTCHAs. In the literature, there exist a variety of scratch removal methods that researchers have found useful for restoring old films and photographs. Some of these methods are appropriate for line and curve removal from digital CAPTCHAs. Most include a detection phase and a removal phase. The detection phase for film scratch removal consists of searching, among all vertical lines in an images, for those that are not natural scene lines. This stage may employ low or high pass filters (Kokaram et al., 1995; Kokaram, 1998), morphological filters (Joyeux et al., 2001; Saito et al., 2000), adaptive binarization (Kao and Engehausen, 2000), discrete wavelet decomposition (Bretscheneider et al., 2000), statistics and MAP techniques (Geman and Geman, 1993; Morris et al., 1996; Tegolo and Isgro, 2001), local gradient measures (Anzalone and Machi, 2001), Hough transforms (Machi et al., 2002), or Kalman filters (Joyeux et al., 2002), possibly followed by Bayesian refinement strategies (Kokaram, 1998). The result of the detection phase is a binary scratch mask indicating which pixels in the original frame are likely scratch pixels.

Many of these film restoration methods deal only with vertical scratches introduced by mechanical rubbing while copying a film (Kim and Kim, 2010; Maddalena and Petrosino, 2008), so they cannot be directly applied to CAPTCHAs, in which line slopes are arbitrary. However, many can be generalized and applied to attacking CAPTCHAs. One suitable method is the Markov random field (MRF) or Gibbs algorithm (Geman and Geman, 1993), which iteratively computes an energy for each pixel based on the pixel's surroundings, removes pixels that have energy beyond a certain threshold, and repeats until convergence. Bursztein et al. (2011) report that line noise removal with their Gibbs implementation enables them to achieve a total precision of 20% on Digg

CAPTCHAs, which contain dense cross hatching with small-line noise.

Classical active contour models were introduced by Kass et al. (1987), Cohen (1991) and Cohen and Cohen (1993). The main drawback of the classical discrete point based representation of active contours for CAPTCHA segmentation is its lack of topological flexibility. In the case of CAPTCHAs, there are multiple characters in the image, requiring manual initialization of multiple contours or active methods able to "break" contours into multiple pieces (Samadani, 1989; Durkovich et al., 1995; Ngoi and Jia, 1999; Choi et al., 2001; McInerney and Terzopoulos, 2000; Giraldi et al., 2003). Even if contours are allowed to split and merge, we encounter the problem that individual snakes can intersect themselves and each other, requiring geometric constraints to prevent intersections (Ivins and Porrill, 1995) or explicit detection and handling of intersections (Wong et al., 1998; Ngoi and Jia, 1999; Ji and Yan, 2002). This problem is especially troublesome when nested snakes are initialized inside one another, but this is exactly what is required in the case of character segmentation, since many characters contain one (e.g. "O") or two (e.g. "B") holes. Note that the number of internal boundaries is itself a very useful feature; for example, a correctly segmented character containing two internal contours can be only be an "8" or a "B."

Given these considerations, we employ an approach based on Xu and Prince's gradient vector flow (Xu and Prince, 1997) and Rochery's quadratic active contours (Rochery et al., 2006) endowed with efficient routines for splitting, merging, and deletion of contours. The method incorporates quadratic constraints (Rochery et al., 2006) to avoid self-intersections and loops in individual snakes as well as to avoid intersection of snakes corresponding to different characters and their internal boundaries. The method also exploits split and merge algorithms using straightforward conditions on the closeness of non-adjacent contour points, providing the topological adaptability of implicit geometric models without sacrificing the simplicity, efficiency, or flexibility of traditional discrete point sample based models. This simple gradient vector flow representation can be easily modified to use other representations such as the generalized gradient vector flow (Xu and Prince, 1998a) or the multi-direction GVF (Tang, 2009).

Once successful segmentation is achieved, as already mentioned, any of a large variety of recognition algorithms could be used. As examples, recognition rates on the pre-segmented NIST handwritten digit database long ago reached 98.5% using Fourier descriptors computed from the digit contours (Cheng and Yan, 1998) and more recently 99.65% using large neural networks (Ciresan et al., 2010).

3. Method

3.1. *Quadratic snake model*

This section provides a brief overview of the quadratic snake proposed by Rochery et al. (2006). An *active contour* or *snake* is parametrically defined as

$$\gamma(p) = [x(p) \quad y(p)]^T, \tag{1}$$

where p is the curvilinear abscissa of the contour and the vector $[x(p) \quad y(p)]^T$ defines the Cartesian coordinates of the point $\gamma(p)$.

The energy functional is given by

$$E_s(\gamma) = E_g(\gamma) + \lambda E_i(\gamma), \tag{2}$$

where $E_g(\gamma)$ is the *geometric energy* and $E_i(\gamma)$ is the *image energy* of the contour γ. λ is a free parameter determining the relative importance of the two terms.

To apply the method to CAPTCHA segmentation with certain widths, we define the geometric energy functional to be

$$E_g(\gamma) = L(\gamma) + \alpha A(\gamma) - \frac{\beta}{2} \iint \mathbf{t}(p) \cdot \mathbf{t}(p') \Psi(\|\gamma(p) - \gamma(p')\|) \mathrm{d}p \mathrm{d}p', \tag{3}$$

where $L(\gamma)$ is the Euclidean length of γ, $A(\gamma)$ is the area enclosed by γ, $\mathbf{t}(p)$ is the unit-length tangent to γ at point p, and $\Psi(z)$, given the distance z between two points on the contour, is used to weight the interaction between those two points (see below). α and β are constants weighting the relative importance of the terms. The functional combines two linear, Euclidean invariant terms: the contour's area and its length. The length term acts as a regularizer, whereas the area term controls the expansion of the region. For positive β, $E_g(\gamma)$ is minimized by contours with short length and parallel tangents. If α is positive, contours with small enclosed area are favored; if it is negative, contours with large enclosed area are favored.

The third quadratic (also Euclidean invariant) term is responsible for interactions between points on the snake. The sigmoid function $\Psi(\cdot)$ defines the radius of the region in which anti-parallel tangents should be discouraged:

$$\Psi(z) = \begin{cases} 1 & \text{if } z < d - \epsilon, \\ 0 & \text{if } z > d + \epsilon, \\ \frac{1}{2}\left(1 - \frac{z-d}{\epsilon} - \frac{1}{\pi}\sin\pi\frac{z-d}{\epsilon}\right) & \text{otherwise.} \end{cases} \tag{4}$$

In application to CAPTCHA character extraction, d is the approximate minimum character width and ϵ expresses the sharpness of the sigmoid part of $\Psi(\cdot)$. During snake evolution, weighting by $\Psi(z)$ in Equation (3) discourages two points with anti-parallel tangents (the opposite sides of a putative character) from coming closer than distance d from each other.

The image energy functional $E_i(\gamma)$ is defined as

$$E_i(\gamma) = -\int \boldsymbol{n}(p) \cdot \nabla I(\gamma(p)) \mathrm{d}p, \tag{5}$$

where $I = I(x, y)$ is the CAPTCHA-image $\boldsymbol{n}(p)$ is the unit-length vector normal to γ at point p, and $\nabla I(\gamma(p))$ is the gradient of I evaluated at $\gamma(p)$.

This term favors anti-parallel normal and gradient vectors, encouraging counterclockwise snakes to shrink around or clockwise snakes to expand to enclose light regions surrounded by dark CAPTCHA characters.[1]

[1] For light characters on a dark background, we simply negate the term involving the image. In the rest of the paper, we assume dark CAPTCHA characters on a light background.

To find a curve γ minimizing $E_s(\gamma)$, one obtains the Euler equations using the calculus of variations. Introducing the gradient descent method and ignoring flow in the direction tangent to γ, one obtains

$$\boldsymbol{n}(p)\cdot\frac{\partial\gamma}{\partial t} = -\kappa(p) - \alpha + \lambda\nabla^2 I(\gamma(p)) + \beta\int \boldsymbol{r}(\gamma(p),\gamma(p'))\cdot\boldsymbol{n}(p')\Psi' \times (\|\gamma(p)-\gamma(p')\|)\mathrm{d}p'. \tag{6}$$

In the equation, $\kappa(p)$ is the curvature of γ at $\gamma(p)$ and $\nabla^2 I(\gamma(p))$ is the Laplacian of I evaluated at $\gamma(p)$. The vector function $\boldsymbol{r}(\cdot,\cdot)$ is the unit vector pointing from point $\gamma(p')$ towards $\gamma(p)$, defined by

$$\boldsymbol{r}(\gamma(p),\gamma(p')) = \frac{\gamma(p)-\gamma(p')}{\|\gamma(p)-\gamma(p')\|}.$$

α, β, and λ are free parameters that need to be determined experimentally. d and ϵ are specified a priori according to the desired CAPTCHA character width (essentially, if different parts of the same snake become closer than a distance of d from each other, they start repel each other).

Absent any image energy, the evolution generates a contracting snake when the direction of the contour is counterclockwise and an expanding snake otherwise. A single snake initialized at the image boundary will generally contract and get attracted to the boundaries of the characters in the CAPTCHA, then automatically split into several snakes corresponding to the individual characters.

Once the algorithm converges, that is, when the snakes attach themselves to the boundaries of the character, we generate *anti-snakes* by offsetting the original snakes by one pixel inside the object. This enables us to extract the internal boundaries in characters such as ‘O’ and ‘8’. For characters that do not have internal boundaries, the anti-snake will contract to a point and get deleted. The offset snake or anti-snake follows the negative of the gradient. It moves through dark regions and stops at a boundary with a light region. Mathematically, our anti-snakes use the negative of the image energy (Equation (5)).

The snakes and anti-snakes communicate through the quadratic term in Equation (3). In order to prevent intersections, anti-snakes require a change to the sign of the parameter β. When the snake and anti-snake tangents point in the same direction, a repelling force can be generated with a negative β. To accomplish this, the quadratic term in Equation (3) must be interpreted as follows: if p and p' belong to the same snake, use $\beta > 0$ otherwise use $\beta < 0$.

Finally, in a more general context, the procedure could be used recursively to capture the structure of hierarchical objects characterized by nested boundaries. Then the anti-snakes would themselves generate further offspring. However, in the case of character segmentation, only a two-level hierarchy is required.

3.2. *GVF external force*

The term $\alpha A(\gamma)$ in Equation (3) leads to the constant term $-\alpha$ in Equation (6). This produces a contracting force similar to the “balloon force” introduced by Cohen and Cohen (1993), increasing the capture region around objects. Unfortunately since this term does not take the image into account, it is difficult to specify a value for α that is appropriate in all regions of the image.

Xu and Prince (1997, 1998b), rather than using a global balloon force, use a smooth, diffuse gradient field as a local external force with the traditional linear snake. This gradient vector flow (GVF) force improves the traditional snake's convergence to a minimum energy configuration. We use the GVF with our quadratic CAPTCHA extraction snakes.

3.2.1. *GVF*

The GVF is a vector field $\mathbf{V}(x,y) = [u(x,y) \quad v(x,y)]^T$ minimizing the energy functional

$$E(\mathbf{V}) = \int_\Omega \mu\left(\left\|\frac{\partial\mathbf{V}(x,y)}{\partial x}\right\|^2 + \left\|\frac{\partial\mathbf{V}(x,y)}{\partial y}\right\|^2\right) + \|\nabla\tilde{I}(x,y)\|^2\|\mathbf{V}(x,y)-\nabla\tilde{I}(x,y)\|^2\mathrm{d}x\mathrm{d}y, \tag{7}$$

where $\tilde{I}$ is a preprocessed version of image I, typically an edge image of some kind. The first term encourages fidelity to $\nabla\tilde{I}$ whereas the second term encourages a smooth vector field. μ is a free parameter controlling the relative importance of the two terms. Minimizing functional (7) leads to the Euler equation given by

$$\mu\nabla^2\mathbf{V} - (\mathbf{V}-\nabla\tilde{I})\|\nabla\tilde{I}\|^2 = 0. \tag{8}$$

Equation (8) is then solved numerically by iterations. Alternatively, by replacing μ in (8) with two weighting functions depending on $\nabla\tilde{I}$ to control the relative importance of the two terms $\nabla^2\mathbf{V}$ and $(\mathbf{V}-\nabla\tilde{I})\|\nabla\tilde{I}\|^2$, we obtain the so-called generalized version of the GVF (Xu and Prince, 1998a).

In our approach, we obtain $\tilde{I}$ using median filtering, oriented filtering, and Canny edge detection. The oriented filters are elongated Laplacian of Gaussian filters that emphasize characters, suppress noise, and, to a certain extent, fill in short gaps where characters have low contrast with the background. The resulting binary Canny image includes information about the important edges that have survived sharpening by the oriented filters. The GVF field on top of the sharpened edge image points toward the character edges from a long distance, and, during snake evolution, pushes the snake in an appropriate direction. This GVF speeds evolution and makes it easier to find suitable parameters to obtain fast convergence.

3.2.2. *Preprocessing by oriented filtering*

Laplacian of Gaussian (LoG) filters are ideal for identifying characters in CAPTCHA imagery. We use the linear response of elongated LoG filters tuned to detect characters at 8 orientations then (for dark characters with light surround) take the maximum response over the 8 orientations. An example is shown in Fig. 1.

Our convolution and minimum response selection procedure responds well to long straight edges having the effect of emphasizing linear parts of the characters and deemphasizing thin line noise, and, to a certain extent, filling in short gaps where a character has low contrast with the background or was broken by the line noise.

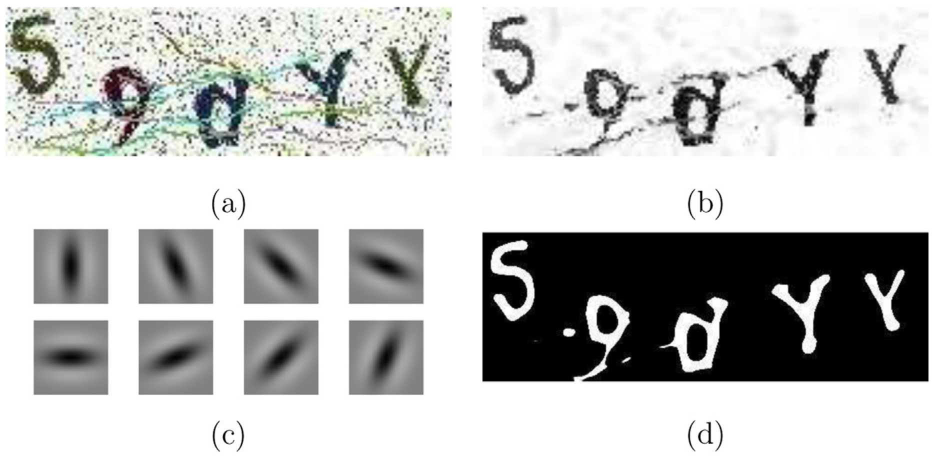

Fig. 1 – Preprocessing by oriented filtering. (a) Original image. (b) Median-filtered image. (c) Oriented-filter masks. (d) Oriented-filtered image.

3.2.3. *Obtaining the GVF field*

After oriented filtering, we obtain the Canny edge image $\tilde{I}$ from the edge-enhanced image obtained from oriented filtering. This is the input to the GVF relaxation procedure (Xu and Prince, 1997). We precalculate $\mathbf{V}$ before snake evolution begins, then, similar to Xu and Prince, during evolution, for each point $\gamma(p)$, we add the force $\lambda_{GVF}\,\mathbf{n}(p)\cdot\mathbf{V}(\gamma(p))$ directly to the update Equation (6). λ_{GVF} is a weight trading off the importance of the GVF force against the other forces in Equation (6).

Clearly, this encourages the snake to snap to the character contours, where ideally $\mathbf{V}(\gamma(p)) = 0$.

The experimental results reported in this paper show that by combining the advantages of the GVF's extended capture range and the quadratic snake's flexibility, the method improves the snake's convergence to configurations that accurately segment structures (see Fig. 2).

3.3. *Family of quadratic snakes*

A single quadratic snake (Rochery et al., 2006) is unable to extract enclosed regions and multiple disconnected characters in an image. We address this limitation by introducing a *family* of cooperating snakes that are able to split, generate offspring (anti-snakes), disappear, and merge if necessary (see Fig. 3).

In our formulation, due to the curvature term $\kappa(p)$ and the area constant α in Equation (6), specifying the points on γ in a counterclockwise direction creates a *shrinking snake*, whereas specifying the points on γ in a clockwise direction creates a *growing snake*.

3.3.1. *Splitting a snake*

Our method splits a snake into two snakes whenever two of its arms are squeezed too close together, i.e., when the distance between two snake points is less than d^{split} and those two points are at least k snake points from each other in both directions of traversal around the contour (see Fig. 3(a) for an example). d^{split} should be less than 2η, where η is the maximum step size.

3.3.2. *Deleting a snake*

A snake γ is deleted if it has perimeter less than L^{delete}. Fig. 3(c) shows an example of a snake being deleted.

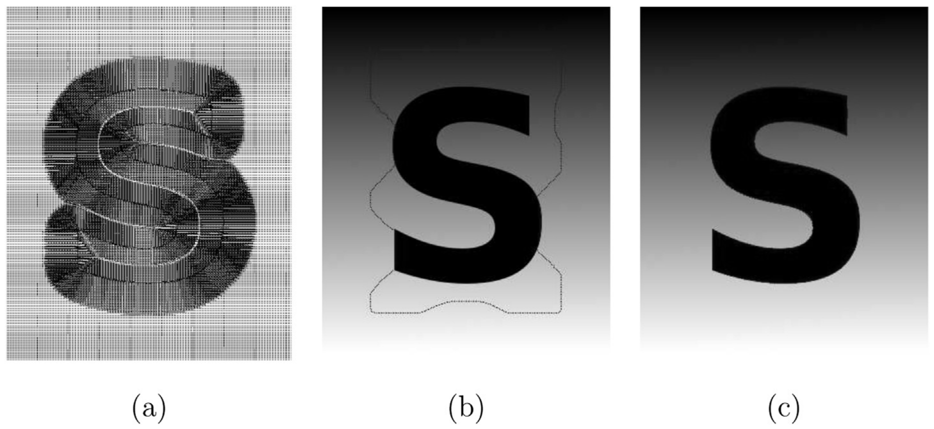

Fig. 2 – Snake-based extraction with GVF. (a) GVF vector field. (b) Converging snake. (c) Successfully converged snake.

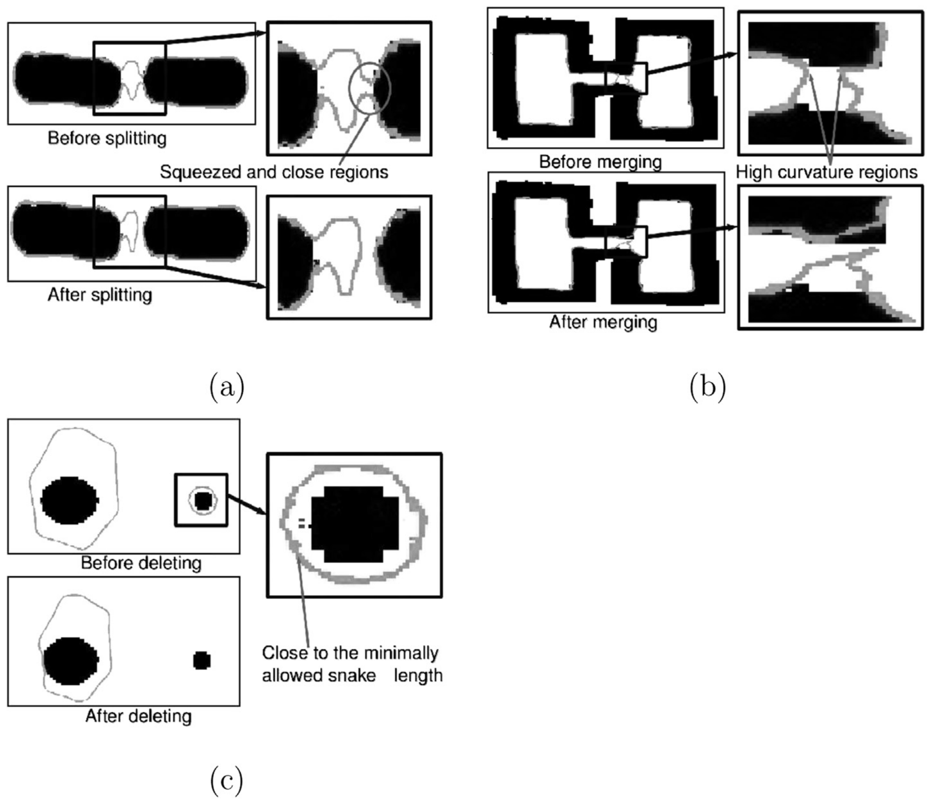

Fig. 3 – Cooperating snakes. (a) A shrinking snake splits into two snakes and finally captures two distant objects. (b) Two merging snakes. (c) Two shrinking snakes, one of which has been deleted after reaching a minimally allowed length.

3.3.3. *Generating the anti-snake*

The evolution of the snake continues until it either attaches itself to a boundary of the character or gets deleted. In the first case an anti-snake is generated by offsetting the original snake by several pixels inside the object.

This is done to extract characters characterized by internal boundaries. If the character does not have an internal boundary, the anti-snake converges to a point and gets deleted.

The offsetting is performed in the direction of the normal. Since several points in the original snake can be mapped into the same point in the offset snake, a re-parametrization of the offset snake must be performed.

3.3.4. *Merging two snakes*

Although merging two or more snakes is technically possible, the CAPTCHA-defeating snakes do not require this operation. However, this element should be included in general purpose segmentations.

4. Experiments

4.1. *Experiment 1: CAPTCHA segmentation and recognition by multiple quadratic snakes*

In Experiment 1, we evaluate the performance of our multiple quadratic snake method at defeating color CAPTCHAs subjected to a combination of line and salt-and-pepper noise. The multiple snake model is compared with a pixel-tracing contour extraction algorithm (Suzuki and Abe, 1985) available in the OpenCV library (Bradski and Kaehler, 2008).

4.1.1. *CAPTCHA data sets and evaluation methods*

The efficiency of each algorithm is evaluated pixel-wise in terms of the precision (fraction of the retrieved pixels that are character pixels), recall (fraction of character pixels that are retrieved), and F_1 (the harmonic mean of the precision and recall).

We also evaluate the extractions by means of the recognition rate (per character and per CAPTCHA). The experiments show that snakes produce almost the same pixel-wise accuracy and a slightly higher recognition rate than the OpenCV contour extraction algorithm. However, it is not hard to create readable CAPTCHAs that cannot be binarized appropriately for the contour-tracing procedure. Since snakes do not require image binarization, this method has the leading advantage. Comparison between Figs. 2 and 4 exemplifies the special convenience of the snake-based method.

To create the CAPTCHA image set used for the experiments, we used the popular Drupal CAPTCHA API (Version 6.x-2.4) (Drupal CAPTCHA project, 2012), which produces CAPTCHAs characterized by distortion and tilting of the characters followed by the addition of line and salt-and-pepper noise. Each CAPTCHA is a 480 × 160 image containing five characters. The average measured noise levels in terms of signal-to-noise ratio

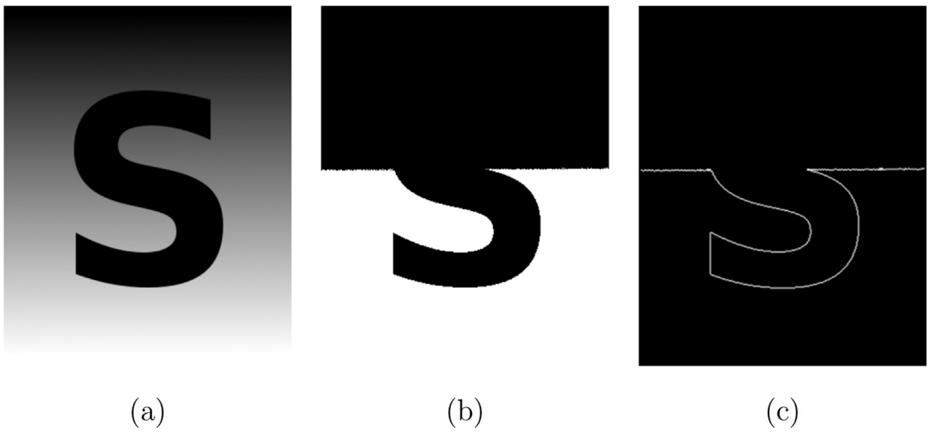
Fig. 4 – Otsu-thresholding an image with low contrast regions. (a) Original image. (b) Otsu-thresholded image. (c) Contours found by the standard OpenCV tracing algorithm.

(SNR) are 6.2 dB, 4.1 dB, and 3.0 dB for salt-and-pepper, line, and combined salt-pepper-line noise, respectively. Table 1 compares the noise levels of our test CAPTCHAs with CAPTCHAs employed by some prominent websites. LN stands for line noise and S&P stands for salt-and-pepper noise. Clearly, the noise levels in the CAPTCHAs used in our experiments are comparable with those of prominent CAPTCHAs. For instance, the S&P + LN CAPTCHAs used in Experiment 1 are characterized by a high level of noise similar to that of Digg.com.

Our first experiment involves three conditions, 1) CAPTCHAs with line noise, 2) CAPTCHAs with salt-and-pepper noise, and 3) CAPTCHAs with combined line and salt-and-pepper noise. The characters and the noise were generated using randomly selected colors. For each type of noise, 10 RGB images were generated by the CAPTCHA API and converted into gray level using the red color channel. The results for the green and blue channels are similar and therefore omitted. In each condition, we tested the snake based extraction method against the ground truth segmentation and the OpenCV contour-tracing method. The performance measures, as previously mentioned, were per-pixel precision, recall, and F_1.

Table 1 – Noise levels in various CAPTCHAs.

CAPTCHA	Noise (dB)	CAPTCHA	Noise (dB)
Digg	2.9	NIH	6.3
S&P + LN (Experiment 1)	3.0	LN-low (Experiment 2)	6.5
LN (Experiment 1)	4.1	CNN	7.9
Blizzard	4.1	Wikipedia	8.1
LN-high (Experiment 2)	4.4	Baidu	8.3
Captcha.net	5.0	MSN	8.4
LN-medium (Experiment 2 and 3)	5.2	Skyrock	10.1
Slashdot	5.4		
Authorize	5.8		
S&P (Experiment 1)	6.2		

The second experiment tests CAPTCHA recognition rates for the two algorithms. Each algorithm ran against 1000 CAPTCHAs distorted by combined noise. To exclude ambiguous cases such as "c" and "C", "l" and "1", the CAPTCHAs were generated from the set {A, a, C, d, E, e, F, G, H, h, K, L, M, m, N, P, R, r, T, W, X, Y, Z, 3, 4, 7, 9}. This follows the recommendations of the seminal work of Bursztein et al. (2011), who performed an extensive series of experiments on how character set selection affects human and machine recognition of character CAPTCHAs and, based on the experiments, recommended to "use a small non-confusable charset: while using a larger charset slightly impacts classifier accuracy and decreases the scheme's learnability, the security gain is too small to be useful: forcing the attacker to learn on 40 CAPTCHAs instead of 10 reduces the accuracy from 100% to 92% which is negligible compared to the loss in human accuracy (98% for 0–9 down to 82% for azAZ09). Accordingly, since increasing the charset does not offer a significant security gain, a CAPTCHA charset should be small ... and should not contain confusing letters."

We evaluate extraction by means of the recognition rate per character and per CAPTCHA. Recognition is performed by two different routines: an inexpensive commercial OCR package (Transym, 2012) utilizing an untrained classifier, and an SVM-based custom-trained classifier. One thousand different CAPTCHAs were used as a training data set (see Section 4.1.5, forthcoming).

The goals of the experiment are first to compare the snake-based and contour tracing-based segmentation methods in terms of recognition rate, and second to evaluate the vulnerability of this particular CAPTCHA family to attack. Given that "a CAPTCHA scheme [is] broken when the attacker is able to reach a precision of at least 1%" (Bursztein et al., 2011), our experiments will show that based on good segmentation, even a simple untrained OCR system is able to break the line noise CAPTCHA scheme.

4.1.2. *Preprocessing*

We subjected each of the original CAPTCHA images to a preprocessing step used by both the multi-snake algorithm and the pixel-tracing methods. The preprocessing algorithm first converts each original RGB image into a grayscale image by taking the red channel. In order to perform the binarization required by the OpenCV tracing methods, assuming that characters have lower intensity than the light background, the method applies a median filter to remove salt-and-pepper distortion and/or small isolated dark blobs created by the line noise process. Finally, the method applies oriented filters optimized for the enhancement of the CAPTCHA characters by adjusting σ_x and σ_y (the standard deviations of the elongated Gaussian). Generally, setting σ_x to the approximate stroke width and σ_y to 1.2–3.0 times larger than σ_x yields excellent filtering results.

4.1.3. *Contour-tracing and multi-snake approaches*

Here we explain the two independent methods used in the CAPTCHA segmentation step:

- **Contour-tracing algorithm**: We employ the Otsu thresholding technique to binarize the grayscale image. Note that the test CAPTCHA images have sufficient contrast for successful binarization (a low contrast case is exemplified in Fig. 4). Finally, we apply the contour-tracing algorithm (Rosenfeld and Pfaltz, 1966) to the binarized image. Contours characterized by small length are deleted. The contour tracing method implemented by OpenCV (Rosenfeld and Pfaltz, 1966) is a generalization of a classical contour tracing method (Suzuki and Abe, 1985) to handle multiple contours. For each black pixel, the four neighboring points are examined. If none of these neighbors has a black point label, the point is assigned a new label. Otherwise, labels carried by the neighbors are said to be equivalent and the label of the point in question is replaced by the minimal equivalent label. The extended version of the contour-following algorithm discriminates between external and internal contours. The algorithm assigns a unique mark to each contour using a special procedure for obtaining the parent contour of the currently followed contour.
- **Multi-snake algorithm**: We hand-tuned the free parameters of the snake model to achieve the best possible results. The optimal values of the model parameters are given in Table 2. Parameters α, β, λ, λ_{GVF}, d, and ϵ are related to snake evolution, whereas d and ϵ are parameters of the sigmoid function (4). Note that although large α facilitates avoidance of noise spots, care should be taken not to overwhelm the contribution of the GVF, weighted by λ_{GVF}. Furthermore, β represents the relative importance of the quadratic terms. Therefore, if self-intersections and intersections of snakes and anti-snakes are possible, β must be large enough to prevent them. In our case, the quadratic terms are important since self-intersections often occur when linear snakes are employed. Finally, λ defines the total contribution of the image energy in the variational functional (2). Therefore, optimization of λ and β must be performed concurrently. We have found that $\lambda = 1.2$ and $\beta = 16.5$ guarantee good extraction results. Contour evolution is terminated whenever the energy $E_s(\gamma)$ fails to decrease for some number of iterations. Finally, we achieved $O(N \cdot \log N)$ complexity for each iteration of quadratic snake evolution, where N is the total number of the snake points, by optimizing search for neighboring contour points.

Table 2 – Optimal parameters for the snake model.

Parameter	σ_x	σ_y	α	β	λ	λ_{GVF}	d	ϵ
Value	8.0	11.9	40	16.5	1.2	6.0	10.5	0.45

4.1.4. *TOCR for CAPTCHA character recognition*

In the preference menu provided by the software package, we limited the scope of allowed characters to those actually used in our CAPTCHA samples.

Using the software's batch interface, in each batch, we supplied 100 preprocessed CAPTCHA images to the software and applied recognition. We repeated the same procedure 10 times for each different image set with 100 CAPTCHAs, finally obtaining the recognition results of 1000 CAPTCHAs.

4.1.5. *SVM for CAPTCHA character recognition*

We compare the ability of TOCR's untrained recognition to that of a custom-trained SVM. To this end, 1000 noiseless CAPTCHAs were collected and broken manually into 5000 patches, with each patch containing exactly one character. This constituted a training set of 5000 character images over 27 classes, which is quite limited. Therefore, we expanded the training set by adding normal printed characters from scanned documents of 300 DPI or more. To model CAPTCHA of the scanned characters that were large enough, we added elastic distortion as described by Jain et al. (1996) to produce new training examples.

In particular, given a character image I, we first compute, for each position (x, y), a motion vector $[\,u(x,y) \quad v(x,y)\,]$ as follows:

$$u(x,y) = \sum_{m=1}^{M}\sum_{n=1}^{N}\frac{1}{\lambda_{m,n}}\alpha_{m,n}2\,\sin(\pi n x)\cos(\pi m y) \tag{9}$$

$$v(x,y) = \sum_{m=1}^{M}\sum_{n=1}^{N}\frac{1}{\lambda_{m,n}}\beta_{m,n}2\,\cos(\pi n x)\sin(\pi m y), \tag{10}$$

where $\alpha_{m,n}$ and $\beta_{m,n}$ are chosen randomly according to a Gaussian distribution with zero mean and unit variance, i.e. $\alpha_{m,n}, \beta_{m,n} \sim \mathcal{N}(0,1)$. The normalizing constant $\lambda_{m,n}$ was experimentally set to $0.01\pi^2(n^2 + m^2)$. A distorted version of an image I, denoted I', is then

$$I'(x,y) = I(x + u(x,y), y + v(x,y)). \tag{11}$$

Fig. 5 shows an example of a character and three random distortions. In total, we computed a training set of 97,165 character images.

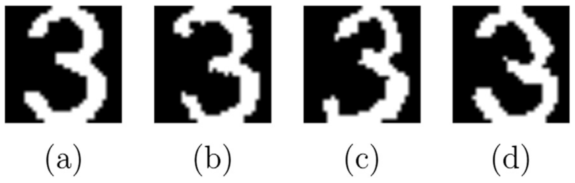

Fig. 5 – Example of a printed character (a), and three distorted versions (b, c, and d).

Each character image was scaled to a size of 30 × 30 pixels while preserving the aspect ratio of the original image. Then each character image was converted into a vector of pixel intensities. This resulted in a feature vector of 900 dimensions for each character. We used an RBF kernel with width parameter (γ) experimentally set to 0.0001.

After the training phase, an SVM with nearly 15,000 support vectors was obtained. This is a quite large model, as one might expect, since the feature space relies only on pixel intensity and a generic kernel function not designed to deal with character distortion. The variability produced by the distortion function is handled implicitly via the large number of support vectors. We believe that a better feature space or kernel could yield a more concise model; this will be investigated in future work.

4.1.6. *Numerical results*

Tables 3–5 present the results of the first numerical experiment, that is, the pixel-wise extraction of CAPTCHAs corrupted by the three noise modes (Figs. 6–8).

The extraction results show similar trends across all types of noise. We observe that the snake-based algorithm in all cases performs better in terms of recall. In precision, the contour-tracing algorithm displays a slight advantage over the snake-based algorithm. To determine the significance of the overall averaged F_1 results, we performed paired two-tailed t-tests for a significant difference in F_1 between the snake-based and contour tracing-based methods. The differences were significant at $\alpha = 0.05$ for all three noise conditions, with the direction of difference favoring the snake-based method.

The most important measure of attack effectiveness is the actual recognition rates under each algorithm. In the second experiment, we processed the 1000 5-letter CAPTCHAs by a simple untrained OCR (Transym, 2012) and a customized SVM-based classifier, trained using a different set of 1000 CAPTCHAs. The results are presented in Table 6. The multi-snake algorithm followed by Transym OCR recognizes 9% of the CAPTCHAs, whereas the OpenCV method combined with Transym OCR recognizes 8%. This is far beyond the 1% rate required for a successful attack. Note that the same OCR system applied directly to the raw images failed to break a single CAPTCHA. As far as the SVM method is concerned, the trained classifier produces 70% and 73% recognition rates with characterwise rates of 92% and 93% for the OpenCV and snake-based segmentation, respectively.

Clearly, such recognition rates are sufficient for a successful attack. Moreover, the recognition rate for the SVM classifier combined with snakes (73%) is much higher than the 20% achieved by the Gibbs algorithm on similar data (Bursztein et al., 2011).

Note that the CAPTCHA recognition rate with snake based segmentation is 1% higher than that with OpenCV when using the untrained OCR system. A 1% improvement is a large relative improvement in the case of an attack applying an untrained classifier with relatively low recognition rates. Finally, since the snake-based method does not require binarization, its advantages could be better exploited, whereas high contrast CAPTCHA characters could be efficiently extracted using either the OpenCV tracer or snake-based algorithm.

4.2. *Experiment 2: line noise and human recognition*

Ideally, Web masters should employ CAPTCHAs that are trivial for humans and totally unbreakable by computers. Do such CAPTCHAs exist? This paper does not answer this question, but it offers a relevant discussion as well as results of experiments on human recognition of the line noise CAPTCHAs used in Experiment 1.

It is easy to create a character recognition task that is difficult for humans and easy for computers. For instance, consider a noiseless image with a background gray level 0 (of 255) and printed letters represented by a gray level of 1. Such characters (invisible for humans) are easily binarizable. The computer recognition rate would likely be close to 100%, while the human recognition rate would be at chance. Therefore, making CAPTCHAs increasingly difficult for humans does not necessarily make them more difficult for computers.

The CAPTCHA proposition rests on the contrary assumption, that with the right kind of noise, it should be possible to design a CAPTCHA that is easy for humans and hard for computers. Finding the right kind of noise and the right level of noise is difficult, however, because the CAPTCHA must be quick and easy for the human to solve. CAPTCHAs that are time consuming or impose a heavy mental load represent inconvenience for the user. Any Web master would thus be cautious to offer CAPTCHAs that are too difficult — stressing

Table 3 – Segmentation of CAPTCHAs with line noise in Experiment 1.

CAPTCHA	Recall (snakes)	Recall (OpenCV)	Precision (snakes)	Precision (OpenCV)	F_1 (snakes)	F_1 (OpenCV)
Line noise 1	0.90	0.86	0.89	0.92	0.89	0.89
Line noise 2	0.90	0.86	0.90	0.92	0.90	0.89
Line noise 3	0.86	0.82	0.88	0.94	0.91	0.88
Line noise 4	0.91	0.87	0.89	0.91	0.87	0.89
Line noise 5	0.80	0.76	0.87	0.96	0.93	0.85
Line noise 6	0.79	0.74	0.86	0.97	0.93	0.84
Line noise 7	0.77	0.72	0.83	0.92	0.90	0.81
Line noise 8	0.84	0.79	0.89	0.97	0.94	0.87
Line noise 9	0.82	0.77	0.88	0.95	0.93	0.85
Line noise 10	0.74	0.80	0.82	0.96	0.94	0.87
Mean	0.83	0.79	0.87	0.94	0.91	0.86

Table 4 – Segmentation of CAPTCHAs with salt-and-pepper noise in Experiment 1.

CAPTCHA	Recall (snakes)	Recall (OpenCV)	Precision (snakes)	Precision (OpenCV)	F_1 (snakes)	F_1 (OpenCV)
Salt-and-pepper noise 1	0.88	0.83	0.89	0.93	0.90	0.88
Salt-and-pepper noise 2	0.89	0.86	0.87	0.89	0.85	0.87
Salt-and-pepper noise 3	0.85	0.80	0.88	0.95	0.91	0.87
Salt-and-pepper noise 4	0.83	0.78	0.88	0.96	0.94	0.86
Salt-and-pepper noise 5	0.84	0.80	0.89	0.96	0.93	0.87
Salt-and-pepper noise 6	0.86	0.82	0.88	0.94	0.91	0.88
Salt-and-pepper noise 7	0.85	0.80	0.88	0.95	0.93	0.87
Salt-and-pepper noise 8	0.87	0.83	0.89	0.95	0.91	0.88
Salt-and-pepper noise 9	0.82	0.77	0.88	0.95	0.91	0.85
Salt-and-pepper noise 10	0.86	0.83	0.89	0.94	0.90	0.88
Mean	0.85	0.81	0.88	0.94	0.90	0.87

their users' cognitive and vision systems will make potential customers walk away.

As previously introduced, Table 1 on page 16 displays noise levels of our test CAPTCHAs vs. CAPTCHAs offered by some prominent websites (see also Bursztein et al., 2010). Clearly, the noise levels in our test CAPTCHAs are comparable to those of other CAPTCHAs. The salt-and-pepper plus line noise CAPTCHAs used in Experiment 1 have a SNR of 3.0 dB, which is lower than any other CAPTCHA except Digg's, which has a SNR of 2.9 dB. The "LN-high" CAPTCHAs presented to humans in the present experiment also contain more noise than the majority of the sampled CAPTCHAs. This shows that our CAPTCHAs are already very near the boundary beyond which Web designers believe they will lose users.

However, beyond looking at signal-to-noise ratios, how can we determine how comfortable a user will be with a particular CAPTCHA? Very few research papers have addressed this issue. Following the long tradition of psychological studies of human visual perception (Palmer, 1999), it is possible to record users' speed and accuracy in judging CAPTCHAs. Increased time or decreased accuracy in solving CAPTCHAs would indicate higher mental loads and consequently more user dissatisfaction.

Lee and Hsu (2011) have further suggested a more direct method to measure users' comfort in solving CAPTCHAs: NASA's TLX scoring system (NASA, 2013). NASA-TLX obtains humans subjective judgments of a task's difficulty along six dimensions: the mental, physical, and temporal demand imposed by the task, as well as the user's perception of their own performance, the effort required, and their frustration with the task.

Three groups consisting of 25, 17, and 16 students at the Sirindhorn International Institute of Technology in Thailand were recruited for the study. Each user was asked to solve 40 CAPTCHAs. Each group's CAPTCHAs were characterized by a different level of line noise as follows: LN-low (6.5 dB), LN-medium (5.2 dB), or LN-high (4.4 dB). Response time and accuracy were recorded automatically during the session. Once the session was completed, each participant assessed the subjective workload using the standard NASA-TLX questionnaire (translated into Thai).

Table 7 shows the speed-accuracy results, and Table 8 shows the TLX results. We hypothesized a linear relationship between noise level and each dependent measure. We tested each hypothesis using the Matlab `linhyptest` routine (MathWorks, 2013) and retained any dependent parameters with $p \leq 0.2$. Table 8 shows the results. Mental stress, subjective performance, and effort were increasingly worse (stress and effort increased and users' subjective performance decreased) as the noise level increased. The trends for the other variables were not statistically significant.

Based on the SNR estimates and the linear relationship between line noise levels and human mental load, we conclude that the CAPTCHAs used in this study, while "easy" for automated attacks, are already near the threshold of being "too hard" for humans. Clearly, then, Web designers should not simply increase the levels of line noise in their CAPTCHAs

Table 5 – Segmentation of CAPTCHAs with combined line and salt-and-pepper noise in Experiment 1.

CAPTCHA	Recall (snakes)	Recall (OpenCV)	Precision (snakes)	Precision (OpenCV)	F_1 (snakes)	F_1 (OpenCV)
Combined noise 1	0.85	0.81	0.88	0.93	0.91	0.87
Combined noise 2	0.87	0.83	0.88	0.91	0.87	0.87
Combined noise 3	0.75	0.71	0.81	0.90	0.87	0.79
Combined noise 4	0.85	0.81	0.87	0.91	0.90	0.85
Combined noise 5	0.85	0.80	0.87	0.93	0.90	0.86
Combined noise 6	0.84	0.79	0.88	0.94	0.92	0.86
Combined noise 7	0.85	0.81	0.88	0.92	0.91	0.86
Combined noise 8	0.77	0.71	0.84	0.96	0.94	0.82
Combined noise 9	0.86	0.82	0.87	0.90	0.87	0.86
Combined noise 10	0.84	0.78	0.85	0.89	0.87	0.83
Mean	0.83	0.78	0.86	0.91	0.89	0.84

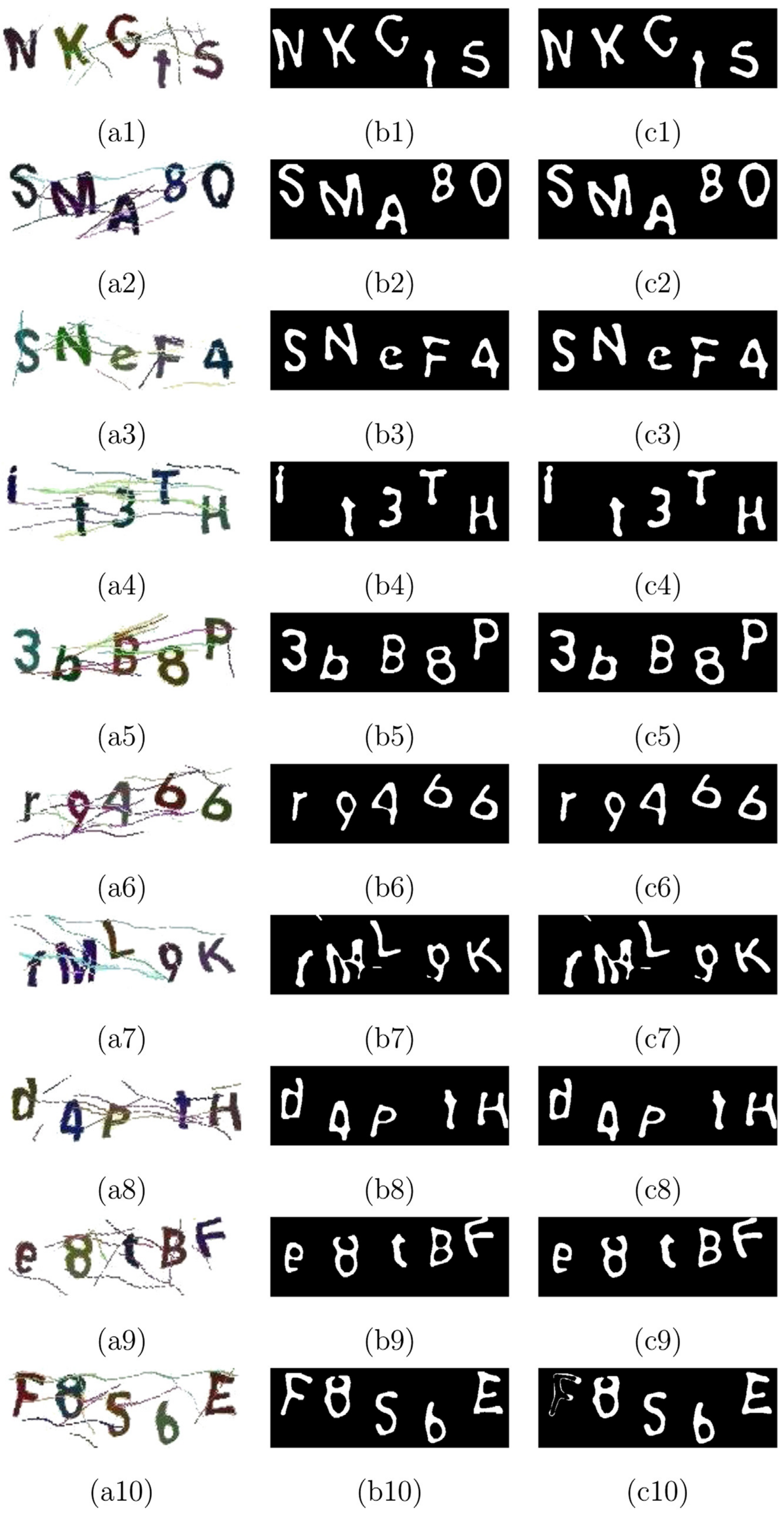

Fig. 6 – Segmentation of CAPTCHAs with line noise in Experiment 1. (a) Original image. (b) OpenCV. (c) Snakes.

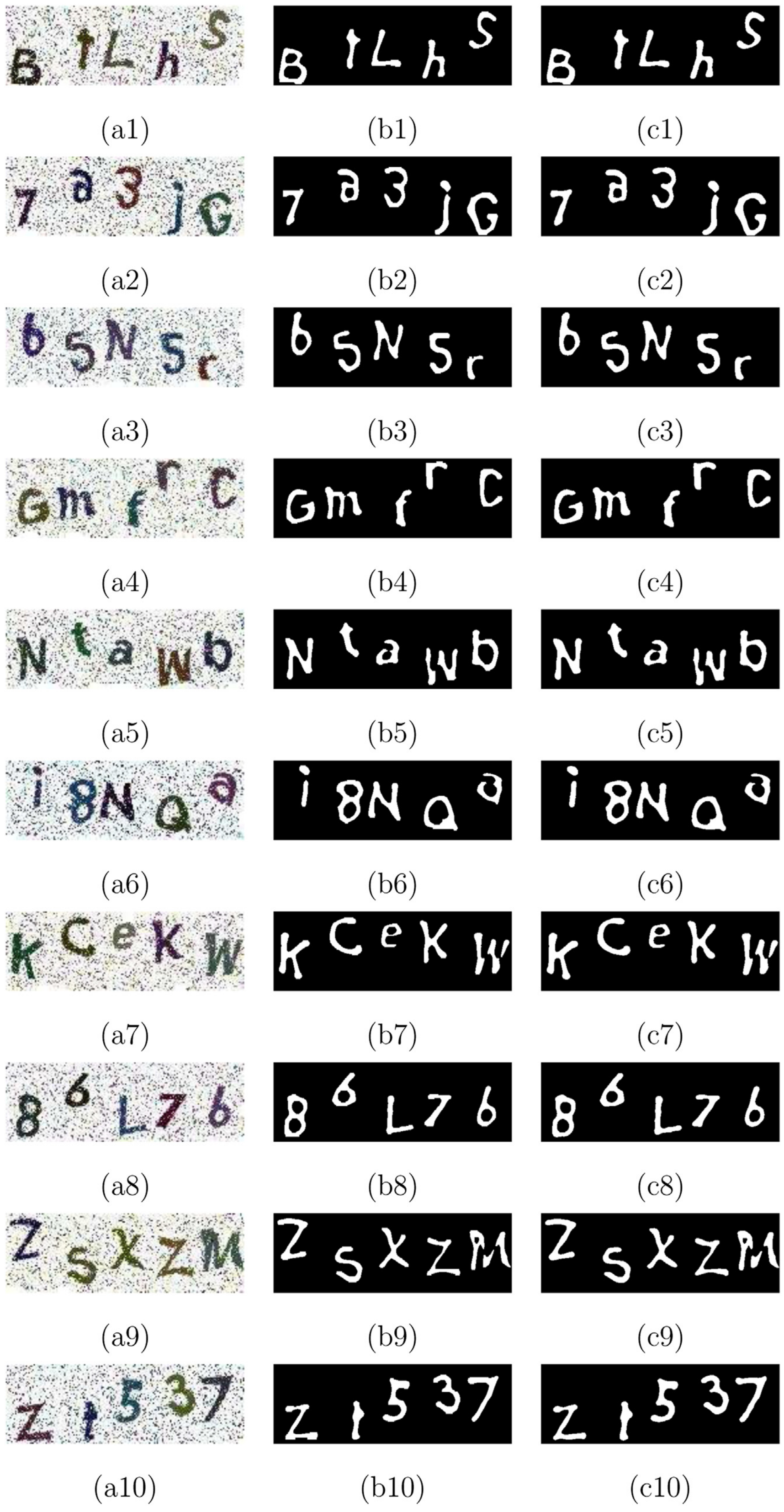

Fig. 7 – Segmentation of CAPTCHAs with salt-and-pepper noise in Experiment 1. (a) Original image. (b) OpenCV. (c) Snakes.

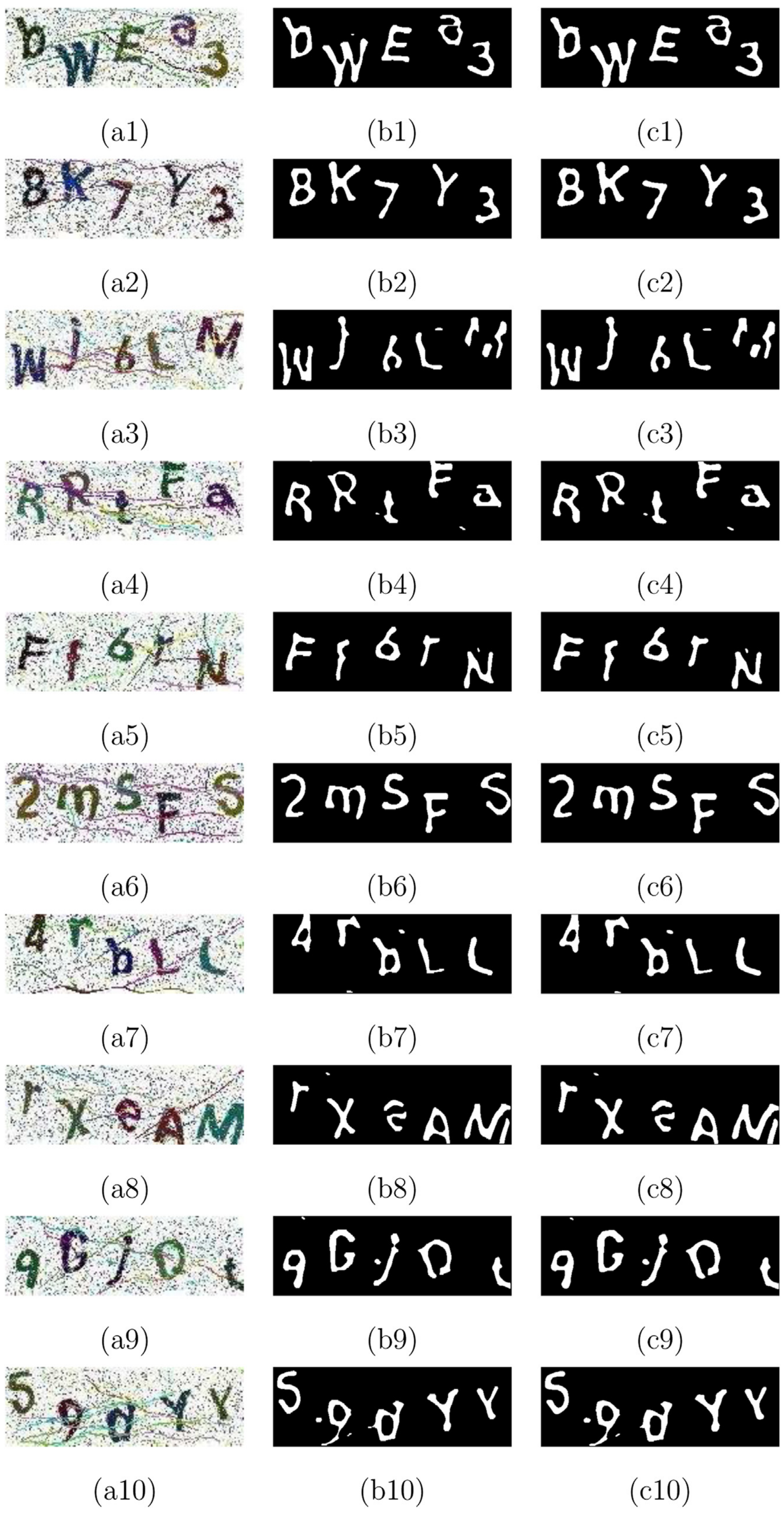

Fig. 8 – Segmentation of CAPTCHAs with combined line and salt-and-pepper noise in Experiment 1. (a) Original image. (b) OpenCV. (c) Snakes.

Table 6 – Recognition rates for CAPTCHAs with combined line and salt and pepper noise in Experiment 1.

Algorithm	Recognition per character	Recognition per CAPTCHA
OpenCV + OCR	0.59	0.08
OpenCV + SVM	0.92	0.70
Snakes + OCR	0.60	0.09
Snakes + SVM	0.93	0.73

to defeat automated attacks — this would surely turn users away.

In the next section, we go further to demonstrate that the snake-based segmentation method is also robust to a third type of noise, a sophisticated anti-binarization technique that defeats existing methods (Gao et al., 2012; Yan and Ahmad, 2008) based on binarization and splitting the image into blocks.

4.3. *Experiment 3: snakes and anti-binarization*

Several efficient CAPTCHA-solvers are based on intensity histograms and image binarization. For instance, Yan and Ahmad's (2008) algorithm projects the cumulative intensity of each column of the CAPTCHA image onto the horizontal axis to obtain a horizontal histogram. The character locations are estimated as the peaks of the histogram, then they are separated and recognized. A more recent attack on Yahoo! CAPTCHAs (Gao et al., 2012) employs projections of the image intensities onto the vertical and horizontal axis. The horizontal projection is used for segmentation of the first and the last character in the group. This (CAPTCHA-dependent) method uses insights obtained from analysis of the particular anti-segmentation techniques used by Yahoo!, but like Yan and Ahmad's algorithm, intensity projection is the central idea of the algorithm.

Appropriate anti-binarization techniques based on variable backgrounds can provide a powerful defense against such projection methods. Distorting the background can transform the intensity histograms in such a way that the characters become inseparable. One such anti-binarization technique is used by the Blizzard.net CAPTCHA (Blizzard.net, 2012); this CAPTCHA employs random backgrounds generated from game screenshots as an attempt to prevent attack software from learning and exploiting common background structure.

While random screenshots may not be the best choice (Bursztein et al., 2010), if the background and character pixels have substantially overlapping intensity distributions, projection-based methods will be defeated.

The goal of anti-binarization is to prevent CAPTCHA solvers from using binarization to find gaps between characters along which the image can be split into blocks containing exactly one character. If the background image is represented by $B(x, y)$ and the overlaid character image by $C(x, y)$. Roughly speaking, human recognition of the CAPTCHA requires that for every point (x, y) for which $C(x, y)$ has a value, we require that $|B(x, y) - C(x, y)| > \delta$, where δ is the minimal difference between the background and the text providing human recognition. As long as this constraint is met, the designer is free to manipulate the intensity distributions of $B(x, y)$ and $C(x, y)$ to defeat projection methods. For example, the CAPTCHA could combine dark characters on a light background and light characters on a dark background.

In Experiment 3, three algorithms are applied to anti-binarized CAPTCHAs: the snake-based method, the vertical projection method (Yan and Ahmad, 2008), and the contour tracing method. The experiment will show that methods based on projection and contour tracing fail, whereas the snake-based method produces sufficiently accurate segmentations to defeat the CAPTCHA.

The main advantage of the snake-based method is that it does not rely on particular values of gray levels. The snake is attracted by any gradient at the boundary between the character and background, irrespective of the overall distribution of foreground and background intensities. Even if the CAPTCHA includes both dark characters on a light background and light characters on a dark background, as long as there is a gradient, the snake will always find them.

We generated CAPTCHAs similar to those analyzed in Experiments 1 and 2 except that the background is generated according to $B(x, y) \equiv B(x) = A \cos(n_w \pi x/L)$. L is the width of the CAPTCHA image (480 currently), A is the amplitude of the cosine wave, and n_w is the number of waves. We select the foreground intensity $C(x, y) \equiv C(x)$ randomly so that $\delta \geq 40$ (weak contrast) and $\delta \geq 60$ (medium contrast). As in Experiments 1 and 2, the characters are tilted and distorted with line noise (LN-medium, Table 1) by the Drupal CAPTCHA generator (Drupal CAPTCHA project, 2012).

To make it possible to compare the results of the three algorithms, we modify them slightly. To be fair to projection algorithm, which does not employ oriented filtering, we remove the oriented filtering step from our algorithm, so that all methods work directly on raw, non-preprocessed images. (Another reason not to use oriented filters is that they require knowledge of the average width of the characters, and this information might not always be available to the attacker.) Since the problem of short arc-like noise has been successfully solved by Yan and Ahmad (2008), we do not include this type of distortion. In all three cases, we follow Yan and Ahmad (2008), placing vertical segmentation lines across the image and considering the CAPTCHA solved if it is successfully partitioned into blocks each containing exactly one character.

Table 7 – Response time and accuracy vs. noise level in Experiment 2.

Measure	LN-low	LN-medium	LN-high
Average response time (s)	6.61	7.14	7.28
Average accuracy	0.938	0.925	0.937

Table 8 – NASA TLX scores vs. noise level in Experiment 2.

NASA TLX dimension	LN-low	LN-medium	LN-high
Mental demand	7.48	8.47	11.00
Performance	5.92	6.94	8.00
Effort	10.36	11.05	12.81

The snake-based segmentation method consists of placing an initial snake around the boundary of the image, letting the contour evolve to convergence, binarizing the result, and splitting the image as follows. The algorithm finds large segments of the histogram with a low pixel count and removes the corresponding parts from the CAPTCHA. The remaining parts of the image contain only one character (see Fig. 9(a)). The projection algorithm obtains the horizontal histogram after binarization by the Otsu method, and then segments the image using the same approach. The OpenCV method performs Otzu binarization then contour tracing and final binarization of the resulting regions.

We evaluate each algorithm according to the number of solved CAPTCHAs as well as the number of correct separating lines. Tables 9 and 10 demonstrate that the proposed anti-binarization techniques defeat the projection-based and the contour-tracing method. However, the snake-based method performs extremely well, even when the variable background is combined with a high level of noise.

Note that the accuracy increase for the low contrast is due to a particular interplay between the line noise and the anti-binarization. A low contrast between the line noise and the background allows the snake to easily pass the noise. On the other hand, it makes it more difficult to attach the snake to the exact boundary of the character. The interaction between the two forces may result in a higher accuracy for a lower contrast.

Table 9 – Accuracy of per-character partition in Experiment 3.

Contrast level	Snakes	Projection	Contour tracing
Medium	0.757	0.715	0.557
Low	0.852	0.189	0.178

Fig. 9 illustrates the advantage of the snake-based method. A strong consistent gradient between the background and foreground would allow for an appropriate Otsu threshold and consequently for accurate partitioning. However, with weak contrast, cosine wave distortion of the background, and line noise, the horizontal histogram no longer discriminates background from foreground, defeating the projection and contour-tracing techniques.

Note that the snake-based method need not generate a single contour for each character. In some cases, when the noise is high, the method produces a collection of contours attached to a single character. However, once the image is binarized according to these contours then projected onto the horizontal axis, the method is quite successful at horizontally separating the challenge image into chunks.

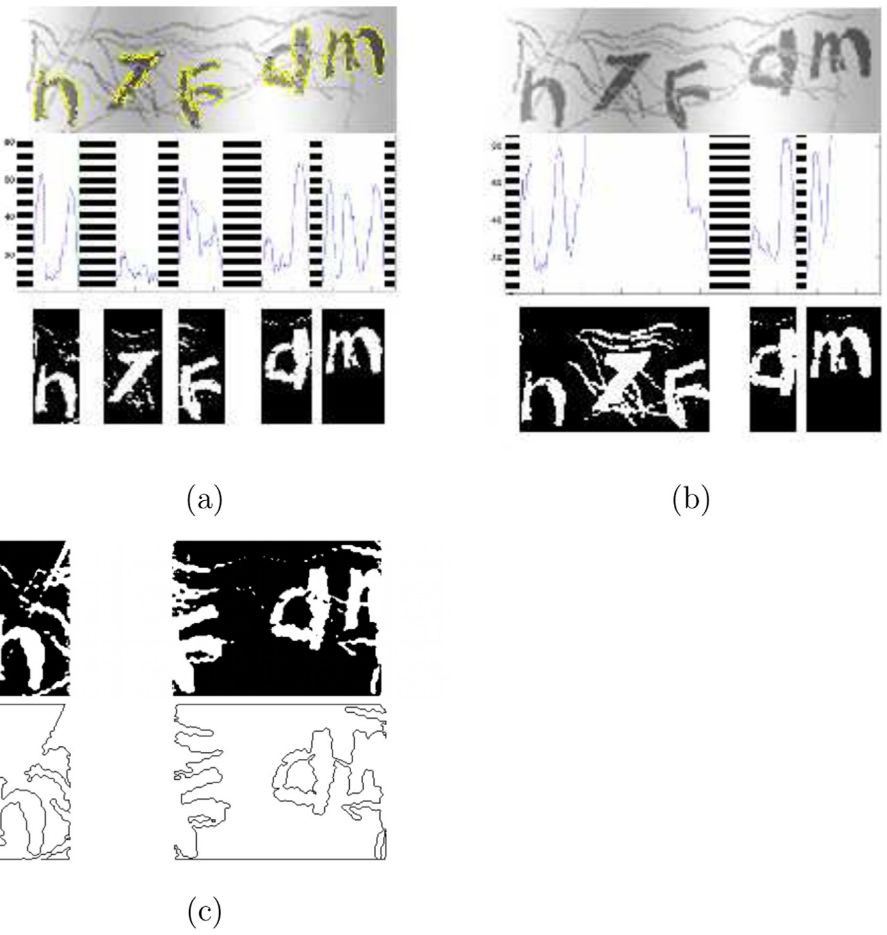

Fig. 9 – Solving anti-binarized CAPTCHAs in Experiment 3. (a) Successful snake-based segmentation. (b) Projection method (failed). (c) Contour tracing method (failed).

Table 10 – Accuracy of per-CAPTCHA partition in Experiment 3.

Contrast level	Snakes	Projection	Contour tracing
Medium	0.578	0.421	0.263
Low	0.684	0	0

Note also that the anti-binarization technique used in this experiment has been simplified to illustrate the idea. To ensure that CAPTCHA solves could not learn and exploit a static background–foreground structure, functions $B(x, y)$ and $C(x, y)$ could be randomized subject to the aforementioned condition that $|B(x, y) - C(x, y)| > \delta$ where $C(x, y)$ is defined.

Finally, it may be possible to use the projection method with multiple thresholds to separate characters, but this would require a more sophisticated algorithm that would critically depend on the size of the sampling window (Sauvola and Pietikinen, 2000).

5. Impact of the research on defeating CAPTCHAs

Recent advances in image processing and computer vision have contributed significantly to the improvement of data security, for example through online verification technologies such as fingerprint verification and other biometric proofs of identity. However, the same or similar methods can be employed to attack and break security protocols. Therefore, more effort is needed to bring the security and image processing communities together to produce interdisciplinary research that combines methods from the two flourishing disciplines to attack cybercrime.

As far as snake-based CAPTCHA recognition is concerned, for the time being, algorithms requiring solution of partial derivative equations are not available to the average hacker. Exploiting the techniques presented in this paper require knowledge of computational mathematics and image processing. However, there is a long tradition in the hacking community of advanced attacks being compiled into easy-to-use scripts or placed online as services. GVF-snakes are already automated. Very soon, individuals with little technical skill will be able to easily apply these techniques to bypass throttling mechanisms and expand their spam, Trojan, virus, and phishing campaigns. As long as visual text CAPTCHAs remain the most popular software bot throttling tool, new CAPTCHAs must be tested against state-of-the-art image processing methods such as snake-based segmentation and SVM-based OCR, lest criminal activity on the Internet further accelerate.

6. Conclusions

We have shown that multiple interacting quadratic active contours are effective at defeating the line noise CAPTCHAs currently being used by many commercial websites. Our numerical experiments performed against a series of CAPTCHAs with line and salt-and-pepper noise display 0.83 precision, 0.86 recall, and 0.89 F_1.

Pattern recognition performed by means of a standard, *untrained* OCR software package applied to the resulting contours produces 9% recognition rates, whereas the same OCR routine applied to the “raw” CAPTCHAs fails to break even a single CAPTCHA from 1000 samples. The recognition rate for the SVM classifier combined with snakes (73%) is much higher than the 20% achieved by the Gibbs algorithm on similar data.

The test CAPTCHAs are typical in terms of the average noise represented by thin lines and the salt-and-pepper distortion. The line-noise applied to the test CAPTCHAs makes them more difficult for humans and increases their discomfort, as measured in terms of NASA-TLX scores.

The competing technique that comprises binarization and contour tracing using standard OpenCV functions produces similar results when applied to high-contrast CAPTCHAs. However, an anti-segmentation method based on contrast reduction and varying backgrounds defeats the OpenCV contour-tracing as well as methods based on vertical projection (horizontal histograms).

Finally, further improvements in accuracy are no doubt possible. However, the achieved levels of accuracy and recognition are sufficient to defeat the studied type of digital CAPTCHAs. Therefore, CAPTCHA segmentation under conditions of line noise should be considered a solved problem. CAPTCHA designers should introduce more challenging distortions into their CAPTCHAs, lest the security of systems based on them be compromised.

Acknowledgments

This research is sponsored by Thailand Research Fund, grant BRG5380016.

We wish to thank the anonymous referees of the paper for their careful and inspirational reviews.

REFERENCES

Achint T, Rusu A, Govindaraju V. Synthetic handwritten CAPTCHAs. Pattern Recognition 2009;42(12):3365–73.
Anzalone A, Machi A. A method for accurate detection of linearly scratched areas in motion pictures. In: Proceeding of IASTED-VIIP01 2001. p. 565–70.
Baird H, Bentley J. Implicit CAPTCHAs. In: Proceedings of the SPIE/IS&T document recognition and retrieval conference 2005. p. 191–6.
Baird HS, Moll MA, Wang S-Y. ScatterType: a legible but hard-to-segment CAPTCHA. In: Proceedings of the eighth international conference on document analysis and recognition 2005. p. 935–9.
Blizzard.net, Blizzard.net, http://www.blizzard.net/; 2012.
Bradski G, Kaehler A. Learning OpenCV: computer vision with the OpenCV library. Sebastopol, CA: O'Reilly Media, Inc.; 2008.
Bretscheneider T, Kao O, Bones PJ. Removal of vertical scratches in digitised historical film sequences using wavelet decomposition. In: Proceedings of image and vision computing conference 2000. p. 38–43.
Bursztein E, Bethard S, Fabry C, Mitchell JC, Jurafsky D. How good are humans at solving CAPTCHAs? A large scale evaluation.

In: Proceedings of the 2010 IEEE symposium on security and privacy 2010. p. 399–413.

Bursztein E, Martin M, Mitchell J. Text-based CAPTCHA strengths and weaknesses. In: ACM conference on computer and communications security (CCS) 2011. p. 125–38.

Carnegie Mellon University. The official CAPTCHA site. http://www.captcha.net; 2000.

Chan T-Y. Using a text-to-speech synthesizer to generate a reverse turing test. In: IEEE international conference on tools with artificial intelligence 2003. p. 226.

Cheng D, Yan H. Recognition of handwritten digits based on contour information. Pattern Recognition 1998;31(3):235–55.

Chew M, Tygar J. Image recognition CAPTCHAs. In: Information security. Lecture notes in computer science, vol. 3225. Springer Verlag; 2004. p. 268–79.

Chewa M, Baird HS. BaffleText: a human interactive proof. In: Proceedings of the SPIE/IS&T document recognition and retrieval conference 2003. p. 305–16.

Choi W, Lam K, Siu W. An adaptive contour model for highly irregular boundaries. Pattern Recognition 2001;34(2):323–31.

Chow R, Golle P, Jakobsson M, Wang L, Wang X. Making CAPTCHAs clickable. In: Proceedings of the 9th workshop on mobile computing systems and applications 2008. p. 91–4.

Ciresan DC, Meie U, Gambardella LM, Schmidhuber J. Deep big simple neural nets excel on handwritten digit recognition. Neural Computation 2010;22(12):3207–20.

Coates A, Baird H, Faternan R. Pessimal print: a reverse turing test. In: International conference on document analysis and recognition 2001. p. 1154–8.

Cohen LD. On active contour models and balloons. CVGIP: Image Understanding 1991;53(2):211–8.

Cohen LD, Cohen I. Finite-element methods for active contour models and balloons for 2-d and 3-d images. IEEE Transactions on Pattern Analysis and Machine Intelligence 1993;15(11):131–47.

da Silva BN, Garcia ACB. KA-CAPTCHA: an opportunity for knowledge acquisition on the web. In: Proceedings of the national conference on artificial intelligence (AAAI) 2007. p. 1322–7.

Datta R, Li J, Wang JZ. IMAGINATION: a robust image-based CAPTCHA generation system. In: Proceedings of the 13th annual ACM international conference on multimedia 2005. p. 331–4.

Drupal CAPTCHA project. CAPTCHA. http://drupal.org/project/captcha; 2012.

Durkovich R, Kaneda K, Yamashita H. Dynamic contour: a texture approach and contour operations. Visual Computing 1995;11:227–89.

Dwork C, Naor M. Pricing via processing or combatting junk mail. In: CRYPTO 1992. Lecture notes in computer science, vol. 740. Springer-Verlag; 1992. p. 139–47.

Elson J, Douceur JR, Howell J, Saul J. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In: Proceedings of the 14th ACM conference on computer and communications security 2007. p. 366–74.

Friendster, Inc., Friendster.com, http://www.friendster.com/; 2012.

Gao H, Wang W, Fan Y. Divide and conquer: an efficient attack on Yahoo! CAPTCHA. In: Proceedings of the 11th IEEE international conference on trust, security and privacy in computing and communications 2012. p. 9–16.

Geman S, Geman D. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. Journal of Applied Statistics 1993;20(5):25–62.

Giraldi G, Strauss E, Oliveira A. Dual-T-Snakes model for medical imaging segmentation. Pattern Recognition Letters 2003;24:993–1003.

Godfrey PB. Text-based CAPTCHA algorithms. In: First workshop on human interactive proofs. unpublished manuscript available at, http://www.adaddin.cs.cmu.edu/hips/events/abs/godfreybabstract.pdf; 2002.

Golle P. Machine learning attacks against the Asirra CAPTCHA. In: ACM conference on computer and communications security (CCS) 2008.

Gossweiler R, Kamvar M, Baluja S. What's up CAPTCHA?: a CAPTCHA based on image orientation. In: Proceedings of the 18th international conference on world wide web 2009. p. 841–50.

He P, Sun Y, Zheng W, Wen X. Filtering short message spam of group sending using CAPTCHA. In: First international workshop on knowledge discovery and data mining 2008. p. 558–61.

Hernandez-Castro CJ, Ribagorda A. Pitfalls in CAPTCHA design and implementation: the math CAPTCHA, a case study. Computers & Security 2010;29(1):141–57.

Holman J, Lazar J, Feng JH, D'Arcy J. Developing usable CAPTCHAs for blind users. In: Proceedings of the 9th international ACM SIGACCESS conference on computers and accessibility 2007. p. 245–6.

Ivins J, Porrill J. Active region models for segmenting regions and colors. Image and Vision Computing 1995;13(5):431–8.

Jain A, Zhong Y, Lakshmanan S. Object matching using deformable templates. IEEE Transactions on Pattern Analysis and Machine Intelligence 1996;18(3):267–78.

Ji L, Yan H. Robust topology-adaptive snakes for image segmentation. Image and Vision Computing 2002;20:147–64.

Joyeux L, Boukir S, Besserer B, Buisson O. Reconstruction of degraded image sequences. Image and Vision Computing 2001;19:503–16.

Joyeux L, Boukir S, Besserer B. Tracking and map reconstruction of line scratches in degraded motion pictures. Machine Vision and Applications 2002;13:119–28.

Kao O, Engehausen J. Scratch removal in digitised film sequences. In: Proceedings of international conference on imaging science, systems, and technology. CSREA Press; 2000. p. 171–9.

Kass M, Witkin A, Terzopoulos D. Snakes: active contour models. International Journal of Computer Vision 1987;1(4):321–31.

Kim K-T, Kim E-Y. Film line scratch detection using texture and shape information. Pattern Recognition Letters 2010;31(3):250–8.

Kokaram AC. Motion picture restoration: digital algorithms for artefact suppression in degraded motion picture film and video. Berlin: Springer; 1998.

Kokaram AC, Morris R, Fitzgerald W, Rayner P. Detection interpolation of missing data in image sequences. IEEE Transactions on Image Processing 1995;4:1496–519.

Lee Y-L, Hsu C-H. Usability study of text-based CAPTCHAs. Displays 2011;32(2):81–6.

Lillibridge MD, Abadi M, Bharat K, Broder AZ. Method for selectively restricting access to computer systems 2001. Tech. Rep., U.S. Patent No. 6,195,698, Issued February 27, 2001.

Machi A, Collura F, Nicotra F. Detection of irregular linear scratches in aged motion picture frames and restoration using adaptive masks. In: Proceedings of IASTED international conference on image processing 2002. p. 254–9.

Maddalena L, Petrosino A. Restoration of blue scratches in digital image sequences. Image and Vision Computing 2008;26(10):1314–26.

MathWorks. Linear hypothesis test. http://www.mathworks.com/help/stats/linhyptest.html; 2013.

McInerney T, Terzopoulos D. T-snakes: topology adaptive snakes. Medical Image Analysis 2000;4(2):73–91.

metroFLOG, metroFLOG.com, http://www.metroFLOG.com/; 2012.

Misra D, Gaj K. Face recognition CAPTCHAs. In: Advanced international conference on telecommunications and international conference on internet and web applications and services 2006. p. 122.

Moneybookers, Moneybookers.com, http://moneybookers.com/; 2012.
Mori G, Malik J. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: IEEE conference on computer vision and pattern recognition 2003. p. 134–8.
Morris RD, Fitzgerald WJ, Kokaram AC. A sampling based approach to line scratch removal from motion picture frames. In: . Proceedings of IEEE international conference on image processing 1996;vol. 1. p. 801–4.
Namprempre C, Dailey M. Mitigating dictionary attacks with text-graphics character CAPTCHAs. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 2007;E90-A(1):179–86.
National Aeronautics and Space Administration (NASA). NASA TLX: Task Load Index. http://humansystems.arc.nasa.gov/groups/TLX/index.html; 2013.
Netlog, netlog.be, http://www.netlog.be/; 2012.
Ngoi K, Jia J. An active contour model for color region extraction in natural scenes. Image and Vision Computing 1999;17(3):955–66.
Oppliger R, Schwenk J, Löhr C. CAPTCHA-based code voting. In: International conference on e-voting 2008. p. 223–36.
Palmer SE. Vision science: photons to phenomenology, Bradford 1999.
Pinkas B, Sander T. Securing passwords against dictionary attacks. In: ACM conference on computer and communications security (CCS) 2002. p. 161–70.
Qu B, Wu Z. Anti breaking CAPTCHA design in C for MIS. In: Proceedings of 2012 IEEE 3rd international conference on software engineering and service science (ICSESS) 2012. p. 415–8.
Rochery M, Jermyn IH, Zerubia J. Higher order active contours. International Journal of Computer Vision 2006;69(1):27–42.
Rosenfeld A, Pfaltz P. Sequential operations in digital picture processing. Journal of the Association for Computing Machinery 1966;12:471–94.
Rusu A, Govindaraju V. Handwritten CAPTCHA: using the difference in the abilities of humans and machines in reading handwritten words. In: Proceedings of the 9th international workshop on frontiers in handwriting recognition 2004. p. 226–31.
Rusu A, Govindaraju V. Visual CAPTCHA with handwritten image analysis. In: Human interactive proofs. Lecture notes in computer science, vol. 3517. Springer Verlag; 2005. p. 153–72.
Saito T, Komatsu T, Ohuchi T, Seto T. Image processing for restoration of heavily-corrupted old film sequences. In: Proceedings of international conference on pattern recognition. IEEE Computer Society; 2000. p. 17–20.
Samadani R. Changes in connectivity in active contour models. In: Proceedings of the workshop on vision motion 1989. p. 337–43.
Sauvola J, Pietikinen M. Adaptive document image binarization. Pattern Recognition 2000;33:225–36.
Shirali-Shahreza S, Movaghar A. A new anti-spam protocol using CAPTCHA. In: IEEE international conference on networking, sensing and control 2007. p. 234–8.
Shirali-Shahreza M, Shirali-Shahreza S. Drawing CAPTCHA. In: Information technology interfaces 2006. p. 475–80.
Simha R., Vora PL. Vote verification using hard AI problems. Journal of Information Assurance and Security 3(4).
Steeves DJ, Snyder MW. Secure online transactions using a CAPTCHA as a watermark 2007. U.S. Patent 7,200,576.
Suzuki S, Abe K. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 1985;30:32–46.
Tagged, Inc., Tagged.com, http://www.tagged.com/; 2012.
Tang J. A multi-direction GVF snake for the segmentation of skin cancer images. Pattern Recognition 2009;42(6):1172–9.
Tegolo D, Isgro F. A genetic algorithm for scratch removal in static images. In: Proceedings of international conference on image analysis and processing. IEEE Computer Society; 2001. p. 507–11.
Transym. Transym OCR. http://www.transym.com/; 2012.
von Ahn L, Blum M, Hopper N, Langford J. Captcha: using hard AI problems for security. In: Advances in cryptology — EUROCRYPT 2003. Lecture notes in computer science, vol. 2665. Springer-Verlag; 2003. p. 294–311.
von Ahn L, Blum M, Langford J. Telling humans and computers apart automatically. Communications of the ACM 2004;47(2):57–60.
von Ahn L, Maurer B, McMillen C, Abraham D, Blum M. reCAPTCHA: human-based character recognition via web security measures. Science 2008;321(5895):1465–8.
VKontakte, vk.com, http://vk.com/; 2012.
Wong Y, Yuen P, Tong C. Segmented snake for contour detection. Pattern Recognition 1998;31(11):1669–79.
Ximenes P, dos Santos A, Fernandez M, Celestino J. A CAPTCHA in the text domain. In: On the move to meaningful internet systems. Lecture notes in computer science, vol. 4277. Springer Verlag; 2006. p. 605–15.
Xu C, Prince JL. Gradient vector flow: a new external force for snakes. In: IEEE conference on computer vision and pattern recognition (CVPR) 1997. p. 66–71.
Xu C, Prince J. Generalized gradient vector flow external forces for active contours. Signal Processing 1998a;71(2):131–9.
Xu C, Prince J. Snakes, shapes, and gradient vector flow. IEEE Transactions on Image Processing 1998b;7(3):359–69.
Xu J, Lipton R, Essa I, Sung M, Zhu Y. Mandatory human participation: a new authentication scheme for building secure systems. In: Twelfth international conference on computer communications and networks 2003.
Yan J, Ahmad ASE. A low-cost attack on a Microsoft CAPTCHA. In: ACM conference on computer and communications security (CCS) 2008.
Zhu BB, Yan J, Li Q, Yang C, Liu J, Xu N, et al. Attacks and design of image recognition CAPTCHAs. In: Proceedings of the 17th ACM conference on computer and communications security 2010. p. 187–200.

Yoichi Nakaguro received the B.S. degree in physics from Waseda University, Tokyo, Japan, in 2002. He is currently working toward the Ph.D. degree with the School of Information and Computer Technology, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand. His current research interests include robotics, computer vision, and image processing.

Matthew N. Dailey received the B.S. and M.S. in Computer Science from North Carolina State University and the Ph.D. in Computer Science and Cognitive Science from the University of California, San Diego. He worked as a Research Scientist at Vision Robotics Corporation of San Diego, CA USA and as a Lecturer in the Computer Science and Information Technology programs at Sirindhorn International Institute of Technology, Thammasat University, Thailand. In 2006, he joined the Computer Science and Information Management department at the Asian Institute of Technology, Thailand. His research interests lie in machine learning, machine vision, robotics, and systems security.

Sanparith Marukatat received a graduate degree in computer science in 1998 from the University of Franche-Comté, Besançon, France. In 2004, he received the Ph.D. in online handwriting recognition from University Paris 6, France. Currently, he is a researcher in the Image Laboratory at the National Electronics and Computer Technology Center (NECTEC), Thailand. Dr. Marukatat's research interests include pattern recognition using statistical tools, sequence modeling, and statistical learning theory.

Stanislav S. Makhanov received the M.Sc in Applied Mathematics from Moscow State University and Dr.Sc. from Computing Center of the Russian Academy of Science in 1988, where he worked as Associate Professor until 1993. He is currently a Full Professor with Sirindhorn International Institute of Technology, Thammasat University of Thailand. Dr. Makhanov has published more than 100 research papers on optimization methods and applications in robotics and image processing. Dr. Makhanov's biography has been published by Who is Who in Asia, Who is Who in Science and Engineering and Who is Who in the World.