2011

AniCAP: An animated 3D CAPTCHA scheme based on motion parallax

Yang-Wai Chow
University of Wollongong, caseyc@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Publication Details

Chow, Y. & Susilo, W. (2011). AniCAP: An animated 3D CAPTCHA scheme based on motion parallax. Lecture Notes in Computer Science, 7092 (N/A), 255-271.

AniCAP: An Animated 3D CAPTCHA Scheme based on Motion Parallax

Yang-Wai Chow[1] and Willy Susilo[2⋆]

[1] Advanced Multimedia Research Laboratory
[2] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
{caseyc, wsusilo}@uow.edu.au

Abstract. CAPTCHAs are essentially challenge-response tests that are used to distinguish whether a user is a human or a computer. To date, numerous CAPTCHA schemes have been proposed and deployed on various websites to secure online services from abuse by automated programs. However, many of these CAPTCHAs have been found to suffer from design flaws that can be exploited to break the CAPTCHA. Hence, the development of a good CAPTCHA scheme that is both secure and human usable is an important research problem. This paper addresses this problem by presenting AniCAP, a new animated 3D CAPTCHA scheme that is designed to capitalize on the difference in ability between humans and computers at the task of perceiving depth through motion. In this paper, we present the design of AniCAP, along with a formal definition of its underlying Artificial Intelligence (AI) problem family. In addition, we analyze the security issues and considerations concerning AniCAP.

Keywords: CAPTCHA, animation, segmentation-resistant, motion parallax

1 Introduction

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are automated tests that humans can pass but current computers programs cannot pass [23]. These days, CAPTCHAs are a ubiquitous part of the Internet and have been effective in deterring automated abuse of online services intended for humans [7]. A variety of different CAPTCHA schemes have emerged over the years, many of which have been deployed on numerous websites. Even major companies such as Google, Yahoo! and Microsoft, and social networks like Facebook, employ the use of CAPTCHAs to provide some level of security against online abuse.

However, many CAPTCHAs have been found to be insecure against automated attacks. Several researchers have demonstrated that certain design flaws in a number of CAPTCHAs can be exploited to break the CAPTCHA with a high degree of success [17, 18, 9, 26, 28, 1, 3]. This has given rise to the important research problem of how to develop CAPTCHAs that are secure against such attacks. The task of developing a good CAPTCHA scheme is a challenging problem. This is because the resulting CAPTCHA must be secure against attacks, yet at the same time it must be usable by humans.

The tradeoff between CAPTCHA security and usability is a hard act to balance. In addition, it has been argued that the difficulty in creating robust CAPTCHAs is further compounded by the fact that the current collective understanding of CAPTCHAs is rather limited [28]. The design of a robust CAPTCHA must in someway capitalize on the difference in ability between humans and current computer programs [7]. This raises the question about whether or not it is possible to design a CAPTCHA that is easy for humans but difficult for computers [8].

This paper addresses the important problem of developing a robust CAPTCHA scheme. While there are three main categories of CAPTCHAs; namely, text-based CAPTCHAs, audio-based CAPTCHAs and image-based CAPTCHAs, this paper focuses on text-based CAPTCHAs. In this paper, we propose the design of a new animated 3D text-based CAPTCHA scheme that attempts to exploit the gap between natural human perception and the ability of computers to emulate perception.

⋆ This work is supported by ARC Future Fellowship FT0991397.

Previous Work. This research was motivated by our previous work on stereoscopic 3D CAPTCHAs, or STE3D-CAP [22]. In previous work, we introduced a novel approach of presenting text-based CAPTCHA challenges in 3D by using stereoscopic images. The key idea behind STE3D-CAP is that humans can perceive depth from stereoscopic images. Thus, by adding random clutter to the scene, the resulting CAPTCHA would be hard for a computer to solve, whereas a human should easily be able solve the CAPTCHA as the text would appear to stand out from the rest of the scene when perceived in 3D. However, the limitation behind this approach was that it relied on the availability of specialized stereoscopic viewing devices, which may be the way of the future but are not ubiquitous at present. Nevertheless, our previous work gave rise to the notion of creating CAPTCHAs based on depth perception.

Our Contributions. This paper presents a new approach to creating animated 3D CAP-TCHA challenges based on the concept of motion parallax. Motion parallax is a monocular cue that allows an observer to perceive depth information from the relative motion between objects in a scene. From the viewpoint of a moving observer, objects that are closer to the observer will appear to move by larger distances as compared to objects that are located further away from the observer. This apparent difference in the motion of objects is one of the means by which the human visual system can perceive depth.

We dubbed our novel animated 3D CAPTCHA scheme 'AniCAP', and its key features are listed as follows:

- Unlike other approaches that add random clutter to the CAPTCHA challenge in an attempt to deter automated attacks, AniCAP, which is a text-based CAPTCHA, uses text itself to increase the difficulty of the challenge. When viewed as a static image, AniCAP has the appearance of overlapping text-on-text, and with no distinct colors or borders around the characters it is not possible to solve the challenge in that manner. However, when AniCAP is viewed from the point of view of a moving camera, this gives rise to motion parallax. As such, it is not possible to use this unique text-on-text approach in the absence of depth perception.
- Hence, while most existing animated CAPTCHAs are 2D CAPTCHAs, AniCAP is actually a 3D CAPTCHA.
- In contrast to a number of other animated CAPTCHA schemes, where the challenge is only displayed for a certain period of time before the user has to wait for the next animation cycle, in AniCAP the challenge is present at all times throughout the animation cycle.
- In addition, the distortion in AniCAP constantly changes in successive frames, thus increasing security by making it difficult for a computer to compare pixel positions between frames.
- Furthermore, unlike the depth perception approach used in previous work [22], this approach does not rely on availability of specialized viewing devices. Instead, AniCAP can be implemented as an animated Graphics Interchange Format (GIF) file, or a video file, which can easily be included on webpages and viewed with standard computer equipment. We have provided an example of AniCAP (an actual animated version) that is available at

  http://www.uow.edu.au/∼wsusilo/CAPTCHA/CAPTCHA.html.

  The correct solution to the challenge is 'SYAK'.

This paper presents the design principles and implementation details of AniCAP. We then formalized the notion of AniCAP and describe the hard Artificial Intelligence (AI) problem underlying this unique CAPTCHA scheme. Additionally, we present a discussion about the various security issues that had to be considered in relation to this novel CAPTCHA technique.

2 Background

2.1 Security and Usability

In order for a CAPTCHA scheme to have any practical value, humans must be able to correctly solve it with a high success rate, whilst the chances for a computer to solve it must be very small. While security considerations push designers to increase the difficulty of CAPTCHAs, usability issues force the designer

to make the CAPTCHA only as hard as they need to be and still be effective at deterring abuse. These conflicting requirements have resulted in an ongoing arms race between CAPTCHA designers and those who try to break them [7].

With advances in research areas like computer vision, pattern recognition and machine learning, and enhancements in Optical Character Recognition (OCR) software, increasingly sophisticated attacks have been developed to break CAPTCHAs. On the other hand, humans have to rely on their inherent abilities and are unlikely to get better at solving CAPTCHAs. Hence, in order to exploit the gap in ability between human and computers it is vital to examine work by others, which highlight the security flaws and usability issues of various CAPTCHAs.

In terms of usability, text-based CAPTCHAs that are based on dictionary words are intuitive and easier for humans to solve. This is because humans find familiar text easier to read as opposed to unfamiliar text [24]. At the same time, CAPTCHAs based on language models are easier to break via dictionary attacks. Mori and Malik [17] were successful in breaking a number of CAPTCHAs that were based on the English language. Rather than attempting to identify individual characters, they used a holistic approach of recognizing entire words at once. Similar attacks exploiting language models have also been performed on a number of other CAPTCHAs [4, 10].

To take advantage of text familiarity without using actual dictionary words, it is possible to use 'language-like' strings instead. Phonetic text or Markov dictionary strings are pronounceable strings that are not words of any language. Humans perform better at correctly identifying pronounceable strings in contrast to purely random character strings. Nevertheless, the disadvantage of using this approach is that certain characters (e.g. vowels) will appear at higher frequencies compared to other characters in pronounceable strings [24].

In an attempt to show that machine learning techniques could be used to break CAPTCHAs, Chellapilla and Simard [9] deliberately avoided exploiting language models and were still successful at breaking in a variety of CAPTCHAs. Solving text-based CAPTCHAs consists of a segmentation challenge, the identification of character locations in the right order, followed by recognition challenges, recognizing individual characters [7]. Their work demonstrated that computers could outperform humans at the task of character recognition. Hence, this led to the important principle that if a CAPTCHA could be segmented, it was essentially broken. As such, the state-of-the-art in robust text-based CAPTCHA design relies on the difference in ability between humans and computers when it comes to the task of segmentation [1].

In order to increase the difficulty of segmentation, techniques such as crowding or connecting characters together can be employed. In addition, the use of both local and global warping to distort characters can also make the task of segmentation harder [25, 28]. It should also be noted that CAPTCHAs with fixed length strings, with characters that possibly appear at fixed locations, are easier to segment [26]. While color and/or textures can be used for aesthetic reasons, or for making it easier to distinguish text from background clutter, the inappropriate use of color and textures can have detrimental effects on both the security and usability of a CAPTCHA [27]. In general, if the use of color or textures does not contribute to the security strength of the CAPTCHA, it may be better not to use any.

2.2 Animated CAPTCHAs

Animated CAPTCHAs have been proposed to overcome the limitations of static CAP-TCHAs. There are a number of existing animated text-based CAPTCHA schemes that are currently deployed on various websites. This section presents an overview of the main ideas behind the construction of a number of these animated CAPTCHAs.

The HELLOCAPTCHA [20] is an animated 2D CAPTCHA that is freely available via the developers' web service. The developers of HELLOCAPTCHA provide a number of different variants to their attractive CAPTCHA scheme. We select a characteristic subset of these, shown in Fig. 1, for discussion in relation to the security considerations presented in the preceding section. The examples depict frames taken at different times, where time increases from the left frame to the right frame. In most of the examples shown in Fig. 1, excluding Fig. 1(e), the challenge is not always on display. Therefore, if the users misses these specific frames, he/she will have to wait for the next animation cycle. The variant in Fig. 1(c), has multiple correct answers because of the changing sequence of characters. This will increase an attacker's chances of success. Background text in the variant shown in Fig. 1(d) can easily be filtered out as the challenge text is in a distinct color. In the examples shown in Fig. 1(a), Fig. 1(b), Fig. 1(c) and Fig. 1(d) the characters

are located at fixed locations, thus making it easier to predict where the challenge will appear. All variants are fixed length character string challenges, thus a computer would have foreknowledge about the total number of required characters. In addition, none of the variants employ the segmentation-resistant principle or character warping, so an OCR program can easily recognize the characters. Hence, by correlating information between different frames, it is highly likely that a computer can break this CAPTCHA.

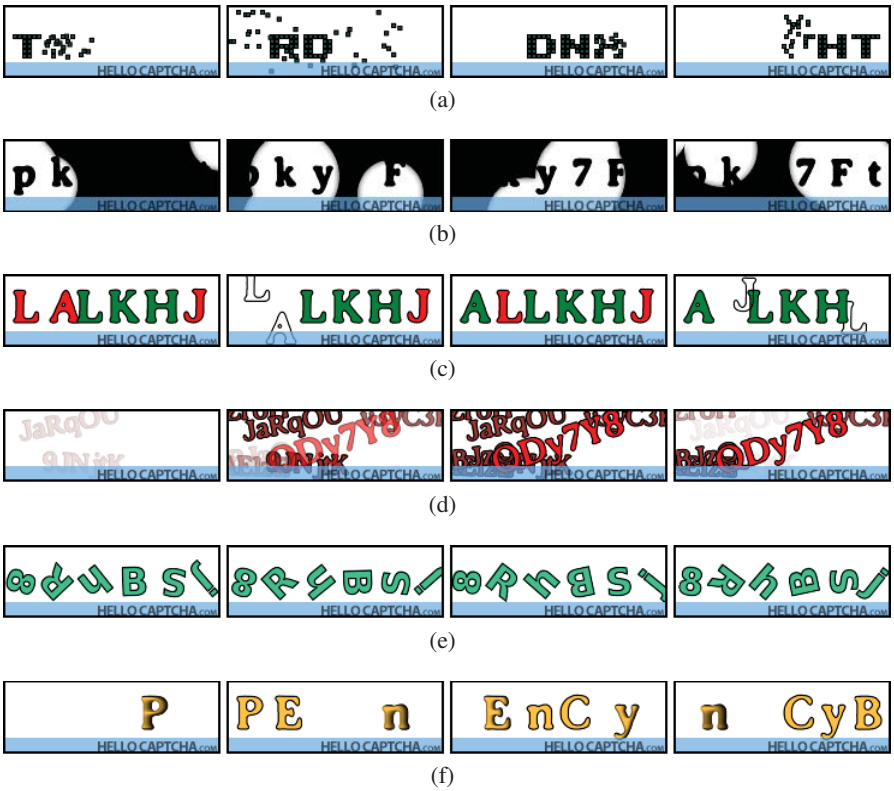
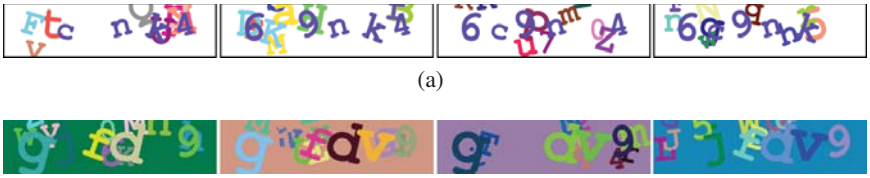


Fig. 1. Examples showing different variants of HelloCaptcha [20]

The JkCaptcha, shown in Fig. 2, is an example of an animated 2D CAPTCHA that uses a text-on-text approach. In this CAPTCHA, what the user sees is a number of persistent characters over a continuously changing background. In the example shown in Fig. 2(a), the challenge is displayed using a distinct color which can easily be separated from the background. From a usability point of view, the use of color in Fig. 2(b) is highly distracting as it changes from frame to frame. The user sees this as continuously flashing color. From a security standpoint, the persistent characters can easily be separated from the characters that change from frame to frame in the background. Furthermore, the characters in the foreground always occlude characters in the background. Additionally, the foreground characters are somewhat larger than the background characters so a simple pixel-count attack can easily be used to separate them [28]. Once separated, an OCR program can easily recognize the characters.

In contrast to the previously discussed animated CAPTCHAs, NuCaptcha [19] is a state-of-the-art animated 2D CAPTCHA, which adopts the segmentation-resistant principle. The developers of this CAPTCHA state that tests have shown that animated CAPTCHA puzzles are easier for humans to recognize and solve than static, scrambled CAPTCHA images. The concept behind NuCaptcha is that when the letters are moving, a human's mind sees the different parts and fills in the blanks; the parts that are moving together are grouped together, and a human can clearly differentiate the letters. Whereas computers do not have this advantage and see a smear of pixels. In addition, unlike CAPTCHAs created in Flash which are not secure, NuCaptcha is displayed as an H.264 MPEG-4 Video Stream that is rendered in the user's browser [19]. An



(a)

(b)

Fig. 2. Examples of JkCaptcha [15]

easy-to-use example of NuCaptcha is shown in Fig. 3. The difficulty level of NuCaptcha can be augmented by increasing the number of characters in the challenge and by crowding the characters closer together. Fig. 3(a) to Fig. 3(c) demonstrate three frames taken at different times. It can be seen that the text scrolls from right to left, with the challenge, that is not always in the display, rendered in a distinct color.

(a) (b) (c)

Fig. 3. Example of NuCaptcha [19]

In the research community, animated CAPTCHAs have been proposed by a number of researchers. Cui et al. [11, 12] decribed a sketch of an animated CAPTCHA approach based on moving letters amidst a noisy background. However, this approach is hard for humans to use. An animated CAPTCHA based on the idea of presenting distorted text on the face of a deforming surface was proposed by Fischer and Herfet [14]. Another proposed animated CAPTCHA with images of moving objects was suggested by Athanasopoulos and Antonatos [2]. However, none of the above proposed methods are related to depth perception in animated images, nor do they analyze the security of their approaches.

2.3 CAPTCHA: Formal Definition and Notation

The term 'CAPTCHA' was introduced by von Ahn et al. [23]. In their seminal work, they describe CAPTCHAs as hard Artificial Intelligence (AI) problems that can be exploited for security purposes. The definitions and notation provided below are adapted and simplified from their work.

A CAPTCHA is a test V where most humans have success close to 1, while it is hard to write a computer program that has overwhelming probability of success over V. This means that any program that has a high probability of success over V can be used to solve a hard AI problem. Let $\mathcal{C}$ be a probability distribution. If $P(\cdot)$ is a probabilistic program, let $P_r(\cdot)$ denote the deterministic program that results when P uses random coins r.

Definition 1. [23] A test V is said to be (α, β)-human executable if at least an α portion of the human population has success probability greater than β over V.

Definition 2. [23] An *AI problem* is a triple $\mathcal{P} = (S, D, f)$ where S is a set of problem instances, D is a probability distribution over S and $f : S \rightarrow \{0, 1\}^*$ answers the problem instances. Let $\delta \in (0, 1]$. For $\alpha > 0$ fraction of the humans H, we require $Pr_{x \leftarrow D}\left[H(x) = f(x)\right] > \delta$.

Definition 3. [23] An AI problem $\mathcal{P}$ is said to be (ψ, τ)-solved if there exists a program $\mathcal{A}$ that runs in time for at most τ on any input from S, such that

$$Pr_{x \leftarrow D,r} \left[\mathcal{A}_r(x) = f(x)\right] \geqslant \psi.$$

Definition 4. [23] An (α, β, η)-CAPTCHA is a test V that is (α, β)-human executable and if there exists $\mathcal{B}$ that has success probability greater than η over V to solve a (ψ, τ)-hard AI problem $\mathcal{P}$, then $\mathcal{B}$ is a (ψ, τ) solution to $\mathcal{P}$.

Definition 5. [23] An (α, β, η)-CAPTCHA is *secure* iff there exists no program $\mathcal{B}$ such that

$$Pr_{x \leftarrow D,r} \left[\mathcal{B}_r(x) = f(x)\right] \geqslant \eta$$

for the underlying AI problem $\mathcal{P}$.

3 AniCAP

3.1 Design and Implementation

A CAPTCHA's robustness is determined by the cumulative effects of its design choices [7]. AniCAP is an animated 3D CAPTCHA that was designed to overcome security flaws highlighted in other text-based CAPTCHA schemes. The main concept underlying AniCAP is that of motion parallax. This capitalizes on the inherent human ability to perceive depth from the apparent difference in motion of objects located at different distances from a moving viewpoint.

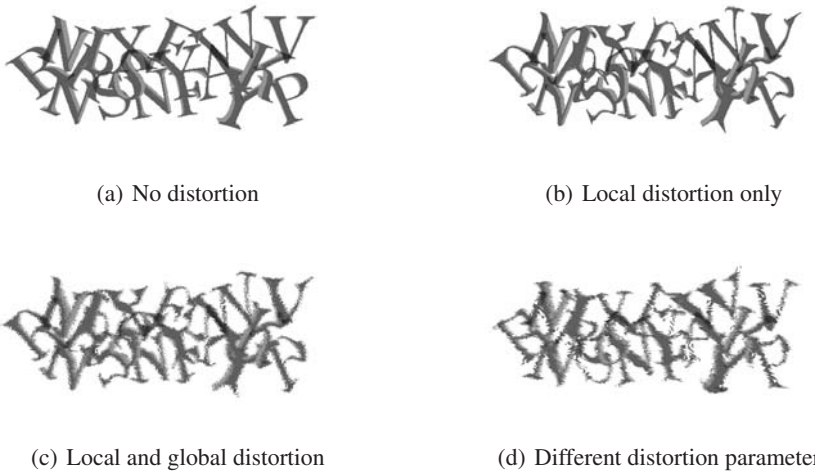
A number of approaches were employed to make AniCAP segmentation-resistant. Firstly, in AniCAP the main characters are rendered over background characters, and characters are overlapped and crowded together. To give rise to motion parallax, the foreground and background characters occupy different ranges of spatial depths. Secondly, some sections of the characters are rendered with a certain degree of translucency to prevent foreground characters from completely occluding background characters. This creates a somewhat 'see through' effect at certain places by blending the overlapping foreground and background characters together. Thirdly, all characters were deliberately rendered using the same font and color, with no distinct borders around the characters. Collectively, these factors make it difficult for a computer to segment the characters, whereas a human can distinguish the main characters in the foreground from the background characters due to motion parallax. This is because the foreground characters will appear to move at different rates compared to the background characters.

Since AniCAP is a 3D CAPTCHA, each 3D character can have random 3D transformations applied to it. For instance, rotations are in all three dimensions and are not merely restricted to the standard clockwise and counterclockwise rotations of 2D CAPTCHAs. When a 3D scene is viewed using perspective projection, objects that are closer to the viewpoint will appear larger than objects that are further away; a concept known as perspective foreshortening. This would mean that to ascertain the foreground characters, one simply had to identify the larger characters. To prevent this, we scale each character so that they all appear to have similar sizes, and AniCAP is made up of random character strings to prevent dictionary attacks.

In addition, characters are all rendered with local and global distortion to deter character recognition and pixel-count attacks. Local distortion refers to distortion applied to individual characters, whereas global distortion is distortion that is applied to the whole scene. To deter computer vision techniques like 3D scene reconstruction and optical flow (discussed in section 4), the global distortion appears to change from frame to frame. However, the change from frame to frame is not completely random, otherwise this would significantly impede human usability. Instead, the global distortion is based on the pixel's location in the frame. What the user sees is like a moving scene viewed through 'frosted glass'. Despite the distortion, when the moving scene is viewed as a whole, a human can perceive the characters because the human mind will group the fragments together as explained by the Gestalt principles of perceptual organization.

Examples of static AniCAP frames are shown in Fig. 4. Note that the same AniCAP challenge is used throughout this paper so that the reader can compare differences between the AniCAP images provided in this paper. A frame without any distortion is shown in Fig. 4(a), whereas a frame with local distortion only is provided in Fig. 4(b), Fig. 4(c) shows the same frame with both local and global distortion and Fig. 4(d)

depicts the same frame with different distortion parameters. Fig. 5 shows a number of animation frames [1]. It can be seen from the static frames themselves that one cannot differentiate the foreground characters from the background characters.



(a) No distortion (b) Local distortion only

(c) Local and global distortion (d) Different distortion parameters

Fig. 4. AniCAP distortion

Fig. 5. Example AniCAP animation frames

The current implementation of AniCAP consists of 3 rows, with a variable number of characters per row. Characters in the rows are made to overlap in the vertical direction and the characters in the columns are crowded together in the horizontal direction, at times overlapping or joining together. The foreground characters consist of a certain number of characters, that are located in sequence somewhere in the middle row. The challenge is deliberately placed at random locations in the middle to as it is conceivable that a computer may be able to identify the shape of some non-overlapping characters at the edges. The reason why they are in sequence is to help human usability, because it is more difficult for a human to identify individual characters at random locations. We can adjust the difficulty level of AniCAP by varying the number of characters in the challenge, as well as the degree of character crowding and overlapping. The number of background characters, crowding and overlapping will also have to vary proportionately. Additionally, the amount of distortion can also be varied.

In order to facilitate motion parallax, the choice of camera movement is important. The highest degree of motion parallax occurs when camera movement is perpendicular to the direction that the text is facing. Translating the camera closer or further away from the 3D text will also create motion parallax, as objects closer to the viewpoint will increase, or decrease, in size at higher rates as opposed to objects further away. However, motion parallax due to size changes are not as apparent compared to changes in horizontal and vertical movement. In addition, the camera can be rotated to view the 3D text from different angles, and it can also be made to focus on different sections of the text. In the current implementation, the camera's movement and rotation are randomized to incorporate all of the above movements at varying degrees.

[1] Please refer to http://www.uow.edu.au/∼wsusilo/CAPTCHA/CAPTCHA.html for the animated AniCAP example.

Depending of the camera's movement, the motion of the foreground characters can either be faster or slower than the background characters.

One of the drawbacks of AniCAP is that it may not immediately be obvious what the user is supposed to look for to solve the challenge. However, once this is described to the user, this should be obvious.

3.2 New AI Problem Family

Here we introduce a family of AI problems that is used to build AniCAP. Let us consider two layers of space, namely $\mathcal{P}_1$ and $\mathcal{P}_2$. Layer $\mathcal{P}_2$ consists of t sub-layers, namely $\{\mathcal{P}_{21}, \mathcal{P}_{22}, \cdots, \mathcal{P}_{2t}\}$. Each layer (or sub-layer) has a transparent background. The distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ is denoted by δ_1, while the distance between $\mathcal{P}_{2i}$ and $\mathcal{P}_{2(i+1)}$ is δ_2. We require that δ_1 be sufficiently large to facilitate motion parallax, and typically $\delta_2 < \delta_1$.

Let $\mathcal{I}_{2d}$ be a distribution on characters, $\mathcal{I}_{3d}$ be a distribution on 3D characters. Let $\mathcal{I}_{mov}$ be a distribution of animation frames. Let $\Delta : \mathcal{I}_{2d} \rightarrow \mathcal{I}_{3d}$ be a lookup function that maps a character in $\mathcal{I}_{2d}$ and outputs a 3D character in $\mathcal{I}_{3d}$ with random 3D transformations. Let Ω_D be a distribution on local distortion factors. Let $\mathcal{D} : \mathcal{I}_{3d} \times \Omega_D \rightarrow \mathcal{I}_{3d}$ be a distribution of local distortion functions. Let $\mathcal{S} : \mathcal{I}_{3d} \rightarrow \mathcal{I}_{3d}$ be a distribution of scaling functions. Let $\Omega_{\tilde{D}}$ be a distribution on global distortion factors. Let $\widetilde{\mathcal{D}} : \mathcal{I}_{3d} \times \Omega_{\tilde{D}} \rightarrow \mathcal{I}_{3d}$ be a distribution of global distortion functions. The distortion function is a function that accepts a 3D image and a distortion factor $\in \Omega_D$ and outputs a distorted 3D image. Let $|A|$ denote the cardinality of A.

When a 3D character $\mathbf{i} \in \mathcal{I}_{3d}$ appears in $\mathcal{P}_1$ (or $\mathcal{P}_{2j}$, resp), we denote it as $\mathbf{i} \rhd \mathcal{P}_1$ (or $\mathbf{i} \rhd \mathcal{P}_{2j}$, resp). The camera $\mathfrak{C}$ views the stacks of layers of space from degree θ, where $\mathcal{P}_1$ is the top layer, followed by all the $\mathcal{P}_{2j}$, where $j \in \{1, \cdots, t\}$. This is denoted as $\mathfrak{C} \vdash_\theta \{\mathcal{P}_1, \mathcal{P}_{21}, \cdots, \mathcal{P}_{2j}\}$. The movement of the camera $\mathfrak{C}$ is recorded as $\texttt{View}_{\mathfrak{C}} \in \mathcal{I}_{mov}$. Let $\mathcal{P}_1||\mathcal{P}_{21}|| \cdots ||\mathcal{P}_{2t}$ be the stacks of layers of space that $\mathfrak{C}$ views. For clarify, for the rest of this paper, we will use **Roman boldface** characters to denote elements of $\mathcal{I}_{3d}$, while Sans Serif characters to denote elements of $\mathcal{I}_{2d}$.

Problem Family ($\mathcal{P}_{\mathsf{AniCAP}}$).
Assume that the CAPTCHA challenge length is ℓ. Let $\phi : \mathbb{Z} \rightarrow \{1, \cdots, \ell\}$ denote a function that maps any integer to the set $\{1, \cdots, \ell\}$. Let rand(c) be the pseudorandom generator function with the seed c. Consider the following experiment.

- **Stage 1. 3D Scene Generation**
  1. For i := 1 to ℓ do
     (a) Randomly select $\mathsf{j} \in \mathcal{I}_{2d}$.
     (b) Compute $\mathbf{j} \leftarrow \Delta(\mathsf{j})$.
     (c) Select a local distortion function $d \leftarrow \mathcal{D}$.
     (d) Compute $\mathbf{k} \leftarrow d(\mathbf{j}, \omega)$, where $\omega \in \Omega_D$ is selected randomly.
     (e) $\mathbf{k} \rhd \mathcal{P}_1$.
  2. For i := 1 to t do
     (a) For k := 1 to $\phi(\text{rand}(time))$ do
        i. Randomly select $\mathsf{j} \in \mathcal{I}_{2d}$.
        ii. Select the scaling function $s \in \mathcal{S}$.
        iii. Compute $\mathbf{j} \leftarrow s(\Delta(\mathsf{j}))$.
        iv. Select a local distortion function $d \leftarrow \mathcal{D}$.
        v. Compute $\mathbf{k} \leftarrow d(\mathbf{j}, \omega)$, where $\omega \in \Omega_D$ is selected randomly.
        vi. $\mathbf{k} \rhd \mathcal{P}_{2i}$.
- **Stage 2. Recording Animation**
  3. Select a global distortion function $d \leftarrow \widetilde{\mathcal{D}}$.
  4. For θ := start to end do
     (a) Compute $\psi \leftarrow d(\mathfrak{C} \vdash_\theta (\mathcal{P}_1||\mathcal{P}_{21}|| \cdots ||\mathcal{P}_{2t}), \omega)$, where $\omega \in \Omega_{\tilde{D}}$ is selected randomly.
     (b) $\texttt{View}_{\mathfrak{C}} := \texttt{View}_{\mathfrak{C}} \cup \{\psi\}$.

The output of the experiment is $\texttt{View}_{\mathfrak{C}} \in \mathcal{I}_{mov}$, which is an animated CAPTCHA.

$\mathcal{P}_{\mathsf{AniCAP}}$ is to write a program that takes $\texttt{View}_{\mathfrak{C}} \in \mathcal{I}_{mov}$, assuming the program has precise knowledge of $\mathfrak{C}$ and $\mathcal{I}_{2d}$, and outputs ℓ characters of $\mathsf{j} \in \mathcal{I}_{2d}$.

Problem Description: $\mathcal{P}_{\mathsf{AniCAP}}$.
Essentially, a problem instance in $\mathcal{P}_{\mathsf{AniCAP}}$ comprises two stages, namely 3D scene generation and recording the animation frames. The first stage, denoted as S_1, accepts the length of the CAPTCHA challenge, ℓ, and outputs a 3D scene, **im**. **im** consists of $t+1$ layers. Formally, this is defined as

$$\mathbf{im} \leftarrow \mathsf{S}_1(\ell).$$

The second stage, denoted as S_2, accepts a 3D scene, **im**, and a range of movements for the camera $\mathfrak{C}$, from $start$ to end, which defines the camera motion, and outputs the sequence of camera recordings $\mathtt{View}_{\mathfrak{C}} \in \mathcal{I}_{mov}$. Formally, this is defined as

$$\mathtt{View}_{\mathfrak{C}} \leftarrow \mathsf{S}_2(\mathbf{im}, start, end).$$

Hard Problem in $\mathcal{P}_{\mathsf{AniCAP}}$.
We believe that $\mathcal{P}_{\mathsf{AniCAP}}$ contains a hard problem. Given the distribution of $\mathfrak{C}$ and $\mathcal{I}_{2d}$, for any program $\mathcal{B}$,

$$Pr_{\forall \mathcal{B},\mathfrak{C},\mathcal{I}_{2d}}(\mathsf{j}^{\ell} \leftarrow ((\mathbf{im} \leftarrow \mathsf{S}_1(\ell)), (\mathtt{View}_{\mathfrak{C}} \leftarrow \mathsf{S}_2(\mathbf{im}, start, end)))) < \eta,$$

where $\mathsf{j} \in \mathcal{I}_{2d}$, and ℓ is the length of the CAPTCHA challenge. Based on this hard problem, we construct a secure (α, β, η)-CAPTCHA.

Theorem 1. *A secure (α, β, η)-CAPTCHA can be constructed from $\mathcal{P}_{\mathsf{AniCAP}}$ as defined above.*

Proof. Based on the problem family $\mathcal{P}_{\mathsf{AniCAP}}$, we construct a secure (α, β, η)-CAPTCHA. We show the proof of this statement in two stages, namely showing that the instance of $\mathcal{P}_{\mathsf{AniCAP}}$ is (α, β)-human executable. Then, we need to show that (α, β, η)-CAPTCHA is hard for a computer to solve.

Given $\mathcal{P}_{\mathsf{AniCAP}}$, humans receive an instance of $\mathtt{View}_{\mathfrak{C}} \leftarrow \mathsf{S}_2(\mathsf{S}_1(\ell), start, end)$. We note that the only viewable contents from this instance is $\mathtt{View}_{\mathfrak{C}}$. When the $start$ and end are selected to provide motion parallax,

then humans can easily output j^{ℓ}, which is the ℓ characters in $\mathcal{P}_1$. Hence, the instance of $\mathcal{P}_{\mathsf{AniCAP}}$ is (α, β)-human executable.

However, given the instance of $\mathcal{P}_{\mathsf{AniCAP}}$, computers cannot output j^{ℓ}. Note that by only analyzing $\mathtt{View}_{\mathfrak{C}}$, computers need to use computer vision or other techniques to recognize the characters. Since machines cannot view the 3D contents, hence this problem is hard for computers. This justifies that the instance of $\mathcal{P}_{\mathsf{AniCAP}}$ is (α, β, η) hard, as claimed. □

4 Security Considerations for AniCAP

In this section, we analyze the security of AniCAP by considering several different attack scenarios.

4.1 Image Processing and Computer Vision Attacks

In image processing and computer vision attacks, the adversary $\mathcal{A}$ is provided with an AniCAP challenge, $\mathtt{View}_{\mathfrak{C}}$. The task of the adversary is to output the CAPTCHA solution, j^{ℓ}. In other words, $\mathcal{A}$ would like to extract j^{ℓ} from $\mathtt{View}_{\mathfrak{C}}$. In order to achieve this goal, $\mathcal{A}$ can launch attacks based on a number of different strategies. These are described as follows.

Edge Detection
The aim of the edge detection technique is to find the edges of the objects in the given image. To perform this attack, $\mathcal{A}$ will first have to decompose $\mathtt{View}_{\mathfrak{C}}$ to its constituting frames. For clarity, we denote the frames contained in $\mathtt{View}_{\mathfrak{C}}$ as

$$\mathtt{View}_{\mathfrak{C}} := \{\mathtt{View}_{\mathfrak{C}1}, \cdots, \mathtt{View}_{\mathfrak{C}n}\}$$

where without losing generality, we assume that there are n frames in $\mathtt{View}_{\mathfrak{C}}$. Note that $\mathtt{View}_{\mathfrak{C}i}$, $i \in \{1, \cdots, n\}$ is a 2D image. Then, $\mathcal{A}$ will conduct edge detection on these images which include all the

Fig. 6. Example of an edge detection image

foreground and background characters as well as the distortion embedded in the image. Since the global distortion, $d \in \widetilde{\mathcal{D}}$, changes from frame to frame, $\texttt{View}_{\mathfrak{C}_i}$, there is little correlation between the resulting edge detection images. Fig. 6 depicts an example of a resulting Canny edge detection image. As can be seen this does not help in solving the challenge.

Image Difference
This attack can be conducted in a manner similar to that of the edge detection technique. First, the frames in $\texttt{View}_{\mathfrak{C}}$ will have to be decomposed. Hence, we obtain

$$\texttt{View}_{\mathfrak{C}} := \{\texttt{View}_{\mathfrak{C}_1}, \cdots, \texttt{View}_{\mathfrak{C}_n}\}$$

as defined earlier. Now, $\mathcal{A}$ will compute the difference between $\{\texttt{View}_{\mathfrak{C}_i}\}$ and $\{\texttt{View}_{\mathfrak{C}_{i+j}}\}$, where $i = \{1, \cdots, n-1\}$, $j = \{1, \cdots, n-i\}$. The results of difference images, obtained between two views, can be further analyzed by using other techniques, such as edge detection techniques. Nevertheless, this still does not yield much useful information for the task of segmentation, as there are too many overlapping characters. Fig. 7(a) shows an example of a resulting difference image between successive frames, and Fig. 7(b) shows the edge detection image after edge detection is performed on Fig. 7(a). It can be seen that no useful information can be obtained to solve the challenge.

(a) (b)

Fig. 7. Difference image and edge detection image

3D Reconstruction
The purpose of this attack is to attempt to reconstruct an approximate 3D scene from $\texttt{View}_{\mathfrak{C}}$ in order to separate the foreground characters from the background characters in three dimensional space. Formally, $\mathcal{A}$ would like to solve

$$\mathsf{j}^{\ell} \leftarrow (\texttt{View}_{\mathfrak{C}} \leftarrow \mathsf{S}_2(\mathsf{S}_1(\ell), start, end))$$

A fundamental problem in 3D reconstruction is assigning correspondence between points in two or more images that are projections of the same point in three dimensional space. Most automated 3D reconstruction approaches use pixels or object silhouettes to specify correspondence [29]. Factors that impede correspondence between frames include noise, textureless regions, non-rigid objects, etc. as this creates ambiguity as to whether or not the selected point is actually the same point in other frames. Current 3D reconstruction algorithms are meant for images or image sequences typically captured from real world

scenarios [16], which do no continuously distort from frame to frame. AniCAP is designed with global distortion that changes from frame to frame, in order to inhibit correspondence required for 3D reconstruction. In addition, the distortion, translucency and camera parameters are randomized, so $\mathcal{A}$ does not have prior knowledge about these.

Optical Flow
Optical flow in general refers to determining the apparent motion of objects in a scene based on the relative motion of the observer. Some definitions vary somewhat and differentiate between motion field estimation and apparent motion estimation [6]. Nevertheless, these are related techniques that may be used in an attempt to break AniCAP. The basic idea is similar to that of 3D reconstruction techniques in that certain points in $\texttt{View}_{\mathfrak{C}}$ have to be selected and tracked between successive frames. As with 3D reconstruction techniques, many current optical flow methods fail when it comes to handling hard problem that involve scenarios with noise, textureless regions, non-rigid objects, etc. [5]. This is again due to ambiguity in the selected points that have to be tracked from frame to frame [21]. AniCAP is designed to facilitate this ambiguity via randomized distortion, translucency and camera parameters, as well as textureless regions with no distinct colors or borders distinguishing the characters.

4.2 Brute Force Attacks

To attack AniCAP, $\mathcal{A}$ can conduct a straightforward attack by adopting the brute force strategy. In this type of attack, $\mathcal{A}$ will provide a random solution to the challenge until one succeeds. We note that the length of the CAPTCHA challenge in AniCAP is ℓ. Suppose there are 26 possible characters which comprise of uppercase alphabetic characters, then the chance of a successful brute force attack is $\frac{1}{26^{\ell}}$, which is negligible. Additionally, in practice CAPTCHAs are usually equipped with techniques such as token bucket algorithms to combat denial-of-service attacks [13].

4.3 Machine Learning Attacks

The aim of this attack is to provide *supervised* training data to the adversary, $\mathcal{A}$, in order to equip $\mathcal{A}$ with sufficient knowledge that can be used to attack the system. Intuitively, a training set of AniCAP challenges will have to be provided with their respective solutions, υ's. Then, after the training is conducted, $\mathcal{A}$ will be given a fresh AniCAP challenge, in which $\mathcal{A}$ has to solve using the knowledge from its database. This attack is inspired by the supervised learning approach in machine learning and the notion of known plaintext attacks in cryptographic literature.

The outline of a practical situation adopting this attack is as follows. Consider a 'smart' attacker program being trained by a human. The human is presented with several AniCAP challenges, and the human can answer these challenges correctly. This information supplied to the attacker program as supervised training data and will be conducted during the *learning* stage. Once the learning stage is over, the program will be presented with a fresh AniCAP challenge. This time, the attacker program will need to answer the challenge itself, given the knowledge that it has gathered during the learning stage. The second stage is known as the *attacking* stage. The attack is considered successful if the attacker program can answer the fresh AniCAP challenge correctly. Formally, this attack is defined as a game among the challenger $\mathcal{C}$, an attacker $\mathcal{A}$ and a human $\mathcal{H}$ as follows.

Stage 1. Learning Stage

1. Define $\mathcal{L} := \emptyset$.
2. Repeat this process q times: For all CAPTCHA challenges given by $\mathcal{C}$ (i.e. $\texttt{View}_{\mathfrak{C}i}$), the human $\mathcal{H}$ will perform the following.
   (a) Output the correct answer υ_i.
   (b) Add this knowledge to $\mathcal{L}$, i.e. $\mathcal{L} := \mathcal{L} \cup \{\texttt{View}_{\mathfrak{C}i}, \upsilon_i\}$.
3. Output $\mathcal{L}$.

Stage 2. Attacking Stage

At this stage the attacker $\mathcal{A}$ is equipped with $\mathcal{L} = \forall_i(\texttt{View}_{\mathfrak{C}i}, \upsilon_i)$, where $|\mathcal{L}| = q$.

1. $\mathcal{C}$ outputs a fresh CAPTCHA challenge $\texttt{View}_{\mathfrak{C}}{}^* \not\subset \forall_i\{\texttt{View}_{\mathfrak{C}_i}\}$, where $\forall_i\{\texttt{View}_{\mathfrak{C}_i}\} \in \mathcal{L}$.
2. $\mathcal{A}$ needs to answer with the correct υ^*.

Note that the required $\texttt{View}_{\mathfrak{C}}{}^*$ in the attacking stage is $\texttt{View}_{\mathfrak{C}}{}^* \not\subset \forall_i\{\texttt{View}_{\mathfrak{C}_i}\}$, where $\forall_i\{\texttt{View}_{\mathfrak{C}_i}\} \in \mathcal{L}$.

Definition 1. *A CAPTCHA is secure against machine learning attacks if no adversary can win the above game with a probability that is non-negligibly greater than $(\frac{1}{n})^\ell$, where ℓ is the length of the CAPTCHA challenge, and n represents the number of characters used in the CAPTCHA challenge.*

Theorem 2. *AniCAP is secure against machine learning attacks.*

Proof (sketch). During the learning stage, $\mathcal{A}$ can form a data set $\mathcal{L} := \{\texttt{View}_{\mathfrak{C}_i}, \upsilon_i\}$, for $i = 1, \cdots, n$. During the attacking stage, $\mathcal{A}$ will be provided with a AniCAP challenge $\texttt{View}_{\mathfrak{C}}{}^*$. Note that $\texttt{View}_{\mathfrak{C}}{}^* \not\subset \forall_i\{\texttt{View}_{\mathfrak{C}_i}\}$, where $\forall_i\{\texttt{View}_{\mathfrak{C}_i}\} \in \mathcal{L}$. Therefore,

$$Pr\left(\texttt{View}_{\mathfrak{C}}{}^* | \{\texttt{View}_{\mathfrak{C}_i}, \upsilon_i\}, \text{where } \mathcal{L} := \{\texttt{View}_{\mathfrak{C}_i}, \upsilon_i\}\right) = Pr(\texttt{View}_{\mathfrak{C}}{}^*).$$

Hence, the knowledge on $\mathcal{L}$ clearly does not help $\mathcal{A}$ to solve the fresh AniCAP challenge, $\texttt{View}_{\mathfrak{C}}{}^*$. □

5 Conclusion

This paper presents AniCAP, a novel text-based animated 3D CAPTCHA. AniCAP is built on the underlying concept that humans can perceive depth through motion parallax, thus capitalizing on the difference in ability between humans and computers at the task of perceiving depth through motion. Foreground characters and background characters in AniCAP are placed at different depths in the 3D scene. Thus, from the point of view of a moving camera, humans can distinguish the main characters in the foreground from the background characters, because the foreground characters will appear to move at different rates compared to the background characters.

AniCAP is designed to be segmentation-resistant by adopting a number of features such as the overlapping and crowding of characters together. Furthermore, by deliberately adopting a distortion approach that changes from frame to frame, this will prevent techniques that attempt to correlate or track points between frames, from succeeding. Other features employed in the design of AniCAP to deter automated attacks include randomized distortion, translucency and camera parameters, as well as textureless regions with no distinct colors or borders to distinguish between characters.

References

1. A. S. E. Ahmad, J. Yan, and L. Marshall. The Robustness of a New CAPTCHA. In M. Costa and E. Kirda, editors, *EUROSEC*, pages 36–41. ACM, 2010.
2. E. Athanasopoulos and S. Antonatos. Enhanced captchas: Using animation to tell humans and computers apart. In H. Leitold and E. P. Markatos, editors, *Communications and Multimedia Security*, volume 4237 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2006.
3. P. Baecher, M. Fischlin, L. Gordon, R. Langenberg, M. Lützow, and D. Schröder. Captchas: The good, the bad, and the ugly. In F. C. Freiling, editor, *Sicherheit*, volume 170 of *LNI*, pages 353–365. GI, 2010.
4. H. S. Baird, A. L. Coates, and R. J. Fateman. PessimalPrint: a Reverse Turing Test. *IJDAR*, 5(2-3):158–163, 2003.
5. S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV*, pages 1–8. IEEE, 2007.
6. S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.
7. K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski. Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs). In H. S. Baird and D. P. Lopresti, editors, *HIP*, volume 3517 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2005.
8. K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski. Designing Human Friendly Human Interaction Proofs (HIPs). In G. C. van der Veer and C. Gale, editors, *CHI*, pages 711–720. ACM, 2005.
9. K. Chellapilla and P. Y. Simard. Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In *NIPS*, 2004.

10. M. Chew and H. S. Baird. BaffleText: a Human Interactive Proof. In T. Kanungo, E. H. B. Smith, J. Hu, and P. B. Kantor, editors, *DRR*, volume 5010 of *SPIE Proceedings*, pages 305–316. SPIE, 2003.
11. J.-S. Cui, J.-T. Mei, X. Wang, D. Zhang, and W.-Z. Zhang. A captcha implementation based on 3d animation. In *Proceedings of the 2009 International Conference on Multimedia Information Networking and Security - Volume 02*, MINES '09, pages 179–182, Washington, DC, USA, 2009. IEEE Computer Society.
12. J.-S. Cui, J.-T. Mei, W.-Z. Zhang, X. Wang, and D. Zhang. A captcha implementation based on moving objects recognition problem. In *ICEE*, pages 1277–1280. IEEE, 2010.
13. J. Elson, J. R. Douceur, J. Howell, and J. Saul. Asirra: a CAPTCHA that Exploits Interest-Aligned Manual Image Categorization. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 366–374. ACM, 2007.
14. I. Fischer and T. Herfet. Visual captchas for document authentication. In *8th IEEE International Workshop on Multimedia Signal Processing (MMSP 2006)*, pages 471–474, 2006.
15. J. C. Kessels. *JkCaptcha*. http://kessels.com/captcha/.
16. Y. Lu, J. Z. Zhang, Q. M. J. Wu, and Z.-N. Li. A survey of motion-parallax-based 3-d reconstruction algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 34:532–548, 2004.
17. G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. In *CVPR (1)*, pages 134–144, 2003.
18. G. Moy, N. Jones, C. Harkless, and R. Potter. Distortion Estimation Techniques in Solving Visual CAPTCHAs. In *CVPR (2)*, pages 23–28, 2004.
19. NuCaptcha Inc. *NuCaptcha*. http://www.nucaptcha.com/.
20. Program Produkt. *HELLOCAPTCHA*. http://www.hellocaptcha.com/.
21. J. Shi and C. Tomasi. Good features to track. Technical report, Ithaca, NY, USA, 1993.
22. W. Susilo, Y.-W. Chow, and H.-Y. Zhou. Ste3d-cap: Stereoscopic 3d captcha. In S.-H. Heng, R. N. Wright, and B.-M. Goi, editors, *CANS*, volume 6467 of *Lecture Notes in Computer Science*, pages 221–240. Springer, 2010.
23. L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.
24. S.-Y. Wang, H. S. Baird, and J. L. Bentley. CAPTCHA Challenge Tradeoffs: Familiarity of Strings versus Degradation of Images. In *ICPR (3)*, pages 164–167. IEEE Computer Society, 2006.
25. J. Yan and A. S. E. Ahmad. Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms. In *ACSAC*, pages 279–291. IEEE Computer Society, 2007.
26. J. Yan and A. S. E. Ahmad. A Low-Cost Attack on a Microsoft CAPTCHA. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM Conference on Computer and Communications Security*, pages 543–554. ACM, 2008.
27. J. Yan and A. S. E. Ahmad. Usability of CAPTCHAs or Usability Issues in CAPTCHA Design. In L. F. Cranor, editor, *SOUPS*, ACM International Conference Proceeding Series, pages 44–52. ACM, 2008.
28. J. Yan and A. S. E. Ahmad. CAPTCHA Security: A Case Study. *IEEE Security & Privacy*, 7(4):22–28, 2009.
29. R. Ziegler, W. Matusik, H. Pfister, and L. McMillan. 3d reconstruction using labeled image regions. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '03, pages 248–259, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.