MOVTCHA: A CAPTCHA Based on Human Cognitive and Behavioral Features Analysis

Asadullah Al Galib(✉) and Reihaneh Safavi-Naini

Department of Computer Science, University of Calgary,
2500 University Drive N.W., Calgary, Alberta T2N 1N4, Canada
{aagalib,rei}@ucalgary.ca

Abstract. We propose a new approach to Captcha which estimates human cognitive ability, in particular visual search ability, to differentiate humans from computers. We refer to this Captcha as Movtcha (**M**atching **O**bjects by **V**isual Search To Tell Computers and Humans Apart). The design of Movtcha takes into account the analysis of human behavior to minimize noise during cognitive feature estimation. Our empirical results suggest that Movtcha can provide accuracy and usability comparable to other established Captchas. Our system is suitable for large scale applications since image selection, challenge generation and response evaluation are automated. Movtcha, unlike other Captchas, surpasses language and experience barriers by presenting both challenge and response in *clear form* and therefore can be used by people all across the world.

1 Introduction

Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) [1] exploits the difference in the ability of humans and computers in performing a task to tell the two apart. The task is designed such that it cannot be solved by computers using the state-of-the-art computing technologies but can easily be solved by human users. The most commonly encountered Captchas rely on distorted alphanumeric string. The advent of image-based Captcha offered an alternative approach with promising usability and security. However, eventually schemes based on image recognition such as [2,3] were compromised. Captchas exploiting semantic relationships between images [4,5] or between images and words [6], usually claim high security and usability but fail to auto generate the challenge (secret) database, resulting in scalability issues.

Another challenge while designing Captchas which is seldom explored, is the issue of Captcha being language, culture or experience dependent. Text/audio based Captchas are entirely language dependent. Some image-based Captchas [4,7] though language independent, rely heavily on user's past experience or exposure to certain things Sect. 7. One way to remove these dependencies is to present both the challenge and response in *clear form* and use behavioral or cognitive features to differentiate human from machine. We define *clear form* as a scenario where the response is not concealed by the challenge *somehow*, e.g. through distortion. This means that both the challenge and response are

© Springer International Publishing Switzerland 2015
L.C.K. Hui et al. (Eds.): ICICS 2014, LNCS 8958, pp. 290–304, 2015.
DOI: 10.1007/978-3-319-21966-0_21

available to the users as well as the computers. *Clear form* allows both human and machine respond to the challenge with ease. However, such scenario clearly violates the current pre-requisite for security of Captcha systems [1,8]. Movtcha presents both challenge and response to a user in *clear form* and is still able to maintain high security. Another important security challenge in Captcha is preventing relay attack where the attacker (the bot) relays/sends the Captcha challenge to a human user who in turn solves the Captcha [9]. Movtcha prevents such relay attacks.

Movtcha is presented as a Cognitive Task. It estimates cognitive ability of a human, in particular visual search ability. An individual's capacity in carrying out any cognitive task (a task requiring a mental process such as perception, thinking and reasoning [10]) is referred to as the cognitive ability of that individual. We consider serial visual search [11] where each item is inspected in turn in a search set to determine if it is a target or not. For example, consider searching through a list of 100 unsorted names until the desired one is found (or the search self-terminates [12]). The unsorted list aids in sequential search unlike a sorted list where users could have skipped some names. In a serial self-terminating search, if it takes a constant amount of time to inspect each item in turn until the desired one is found then the visual search time is found to have a positive linear relationship with the position of the target item inside the search set [13,14].

We design a game-like Cognitive Task (CT) that takes advantage of this observation. The cardinal notion is to conceal the size of the list or search set containing target item from the bot but keep it visible/comprehensible to a human user. For example, consider Bob and a machine. Both are presented with a list of 10 unsorted names and challenged to find "Alice" which appears at the 9^{th} position. Bob *somehow* acquires a *knowledge* which says "Alice" does not appear in the first 5 entries. As a result Bob searches from the 6^{th} position and continues until he finds "Alice" at the 9^{th} position. If it takes 1 second to inspect each name in the list, Bob will spend 4 seconds to find "Alice". On the other hand the machine being deprived of the *knowledge* searches from the 1^{st} position until it finds "Alice" at the 9^{th} position. The machine searches faster than Bob and both the challenge ("Alice") and response ("Alice") are in *clear form* but it fails to mimic the search time of Bob without the *knowledge*. That is, although the bot can add delays and increase its search time, it does not know how much delay needs to be introduced. Movtcha has a similar design principle where the *knowledge* is conveyed only to a human user.

Movtcha consists of a carefully tailored image and a challenge tile. An image is first divided into θ items (cells) by superimposing a grid like structure. A subset of cells $\theta_{sub} \subseteq \theta$ is modified (the *knowledge*) such that, (1) a bot is not able to differentiate them from other cells and (2) the *modification* process creates random artifacts, conveying no meaning in current context to a human user [11] (Appendix A, Fig. 1(a)). A target tile t_r is then selected randomly from the search set, θ_{sub}, and an exact copy of t_r is presented to the user as the challenge tile t_c. Dragging and dropping t_c (challenge) onto t_r (response) inside θ_{sub} is equivalent to a correct visual search task. A human instantly distinguishes and separates out these *exotic* tiles, θ_{sub}, from the context of the image via parallel

search and then performs serial search on the subset, θ_{sub}, to find t_r [15]. Therefore, his search time will vary according to $|\theta_{sub}^{P_{t_r}}|$ i.e. the number of *exotic* tiles that need to be inspected before encountering t_r. On the other hand, a bot is not able to figure out the *knowledge* i.e. θ_{sub}. Therefore, it will not be able to mimic the search time of a human user.

We do not consider the actual visual search time. In fact, we look for *trends*. If the search time grows linearly (roughly, possibly with a few outliers) with $|\theta_{sub}^{P_{t_r}}|$, then the system authenticates the user as a human. Our contributions in a nutshell, (1) Movtcha estimates a cognitive feature to make authentication decision. Our design takes into account human behavioral analysis to eliminate noise during feature estimation. (2) It bypasses traditional pre-requisite for the security of Captcha by presenting both the challenge and response in *clear form*. This makes it language and experience independent. (3) It is resistant against random, automated and static relay attacks. (4) Movtcha is an automated system.

Paper Organization. Section 2: Design of Movtcha, feature estimation and authentication mechanism. Section 3: Search set and challenge generation. Section 4: Security analysis. Section 5: Experiments & results. Section 6: Relay attacks. Section 7: Related work. Section 8: Conclusion.

2 The Cognitive Task as MOVTCHA

Movtcha is a simple and intuitive object matching game. This section provides details on the (1) design of Movtcha (2) extraction of cognitive and behavioral features and (3) the authentication mechanism.

2.1 Design and Execution of Movtcha

An image of size $x \times y$ pixels, ($width \times height$), is first broken into a grid, g, containing θ pieces of square tiles of size $k \times k$ indexed as $c_1, c_2, \ldots, c_{|\theta|}$ from left to right and then top to bottom, $|\theta| = \frac{x \times y}{k^2}$. The random set of tiles that is systematically *modified*, to look *exotic* to a human user is referred to as the search set θ_{sub}. The game starts with the user being challenged with a tile t_c at position P_{t_c}. The objective of the human user is to drag and drop t_c onto the corresponding target tile, t_r, inside g. We call this search action/response A_{resp}. On a correct visual search task, the user is rewarded with a *star*, s_r, superimposed on t_r. The user then performs action A_{rew}, where he drags and drops the rewarded *star* s_r, back to P_{t_c}. One *instance* of the game is thus completed. The user is required to play certain number of *instances* in a Movtcha and each time the image, the number of *exotic* tiles $|\theta_{sub}|$, and the position of the target tile is varied.

Constraints and Helpers. The game must invoke serial self-terminating visual search of a human user after the parallel stage. In order to guarantee (1) its invocation *(C1, C2 & H2)* (2) its correct measurement under non-laboratory conditions with the aid of human behavioral analysis *(C2, C3, C4 & C5)* and (3) facilitate the visual search process *(H1 & H2)*, certain *constraints &helpers* have been placed throughout Movtcha.

C1. At the beginning of each *instance*, as the user hovers over a bounded region P_{t_c}, the tile holder and grid "pop up". The tile holder moves randomly within R_{t_c} and the tile is only visible when the user hovers over the tile holder. This ensures no prior exposure of the challenge tile or search set which can bias the search time [11] and ensures a drag and search action from user. Appendix A Fig. 1(a).

C2. Consider the pixel co-ordinate system. As the tile is dragged, if the y-coordinate of the drag event, e_d^y is in between the minimum and maximum y-coordinate of the j^{th} row in the grid, g, then that row is highlighted by two red lines. If e_d^y during A_{resp} crosses the maximum y-coordinate of target tile t_r, then t_r is highlighted signifying a failed search. The user is then presented with a new *instance*. This constraint ensures that user does not skip over and misses t_r while performing serial search from top to bottom and from left to right. Therefore, visual search time, VST collected from a skipped search is avoided similar to [13].

C3. If the time taken in A_{resp} crosses some experimentally set threshold λ (Sect. 5.1), the user immediately receives a new *instance*. This discards abnormal VST caused due to loss of attention by the user and encourages user not to get distracted while completing an *instance*.

C4. A_{rew} action demands smooth movement of star, s_r, since no cognitive thinking in particular, visual search, is required to execute A_{rew}. Large number of pauses during the movement of s_r signifies that the user was distracted. If the amount of pauses crosses some experimentally set threshold ζ (Sect. 5.1) during dragging s_r, then it moves back to t_r inside g.

C5. On dropping t_c anywhere other than on t_r inside g, the challenge tile t_c moves back to P_{t_c} signifying a *mismatch*. The user immediately receives another new *instance*.

H1. We allow some tolerance on the placement of the tile/star. This means that the user does not need pin-point accuracy when dropping t_c/s_r.

H2. A grid is drawn on the image for establishing finer distinguishability among the tiles and forming a structured visual field. This helps in invoking serial search. It also aids in resisting automated *memory* attacks (Sect. 4).

2.2 Cognitive and Behavioral Feature Extraction

We refer to features collected during the execution of a cognitive task as cognitive features. Human behavior do not necessarily invoke any particular cognitive process. We refer to features collected through the observation of human behavior, such as in behavioral biometrics (*while browsing, typing*), as behavioral features.

Visual Search Time Estimation, VST. The time required for the user to visually search, detect and match t_c onto t_r, is referred to as the visual search time VST. The VST is a cognitive feature, calculated by the subtraction method [16]. The subtraction method is an established technique in cognitive psychology that

involves subtracting the amount of time information processing takes with the process ($MT_{A_{resp}} + t^t_{RT}$) from the time it takes without the process ($MT_{A_{rew}}$). Therefore, $VST = (MT_{A_{resp}} + t^t_{RT}) - MT_{A_{rew}}$, where $MT_{A_{resp}}$ = time elapsed during A_{resp}, $MT_{A_{rew}}$ = time elapsed during A_{rew}, t^t_{RT} = (reaction) time elapsed between the appearance of the stimulus (t_c) and the user picking it up (responding).

Pause Time, PT. This feature is required to enforce almost smooth movement during dragging the *star* and in turn to provide better estimation of VST. If user remains at the same pixel for more than 0.1 seconds we refer to it as a pause. We measure the number of pauses and derive the total paused time, $PT_{A_{rew}}$. The A_{rew} action does *not* involve any cognitive process. $PT_{A_{rew}}$ should be zero in an ideal condition. In practice, if $PT_{A_{rew}}$ crosses some experimentally set threshold ζ then constraint $C4$ is activated.

2.3 Telling Computers and Humans Apart

We use an accuracy metric $\triangle^A_{VST}$ in order to differentiate between a human and a bot. Let s^h_i and s^m_i represents the two series of observations (VST) at instances $\langle 1, 2, \ldots, n\rangle$ from human user and bot respectively. The two series are first arranged according to decreasing order of search set sizes. A series of plus points s^{h+}_i and s^{m+}_i are then obtained from s^h_i and s^m_i. A plus point p is awarded to an element s_e at index i_e of the sorted sequence s, if s_e is greater than p elements with indices more than i_e. An accuracy metric $\triangle^A_{VST}$ is then calculated as the ratio of the summation of the plus points s^+ and the summation of a strictly decreasing series. If the VSTs follow a strictly decreasing trend with decreasing set sizes then the resulting accuracy metric $\triangle^A_{VST} = 1$ and vice versa. A human or a machine is then authenticated based on two *conditions* (1) the $\triangle^A_{VST}$ must cross some certain threshold α. (2) $VST - MT_{A_{rew}}$ must be least when the search set size $|\theta_{sub}| = 1$.

We provide an example, here, to show how plus points and $\triangle^A_{VST}$ are calculated for $n = 6$ *instances* for a human user and a bot. The bot makes random guesses on the position of the target tile P_{t_r} inside θ_{sub} and generates VST. The VSTs, $s^h = \langle 3.8, 3.3, 1.7, 3.4, 1.4, 0.3\rangle$ and $s^m = \langle 2, 3, 3.5, 4.5, 4, 1\rangle$ are first arranged according to decreasing order of $|\theta^{P_{t_r}}_{sub}| = \langle 33, 26, 15, 12, 5, 1\rangle$. Plus points $s^{h+} = \langle 5, 3, 2, 2, 1, 0\rangle$ and $s^{m+} = \langle 1, 1, 1, 2, 1, 0\rangle$ are then used to figure out $\triangle^A_{VST} = \frac{\sum_{i=1}^{n} s^+_i}{n(n-1)/2}$. $\triangle^A_{VST} = 0.867$ for human and $\triangle^A_{VST} = 0.4$ for bot. Experimentally the value of the authentication threshold α is set. In this scenario, setting α to some values less than 0.867 allows some *tolerance* with few observations being out of place for a human user. Increasing such *tolerance* increases the success probability of the bot. Appendix A Fig. 2(a) shows how the success probability varies with the amount of *tolerance* or $\triangle^A_{VST} \wedge condition(2)$.

3 Nuts and Bolts of Our System

Movtcha consists of the following stages: (1) Selecting the appropriate images to be tailored, (2) Generating the search set and displaying the challenge and (3) Telling computers and humans apart (as described in Sect. 2.3).

3.1 Selecting Images to Be Tailored

The goal is to have some portion of the image look *exotic* to a human e.g. consider the rectangular half of a book being modified to a cone. This modification needs to be done in such a way s.t. (1) the human can distinguish the *exotic* tiles in the parallel stage and then perform a serial search to find the target, (2) the machine is not able to figure out the *exotic* tiles. We use images containing pencil or pen sketches/drawings. All images in our setting are first converted to grayscale. Uncontrolled colors generally hinders serial search and can make some tiles more conspicuous or obscure than others [11,17]. Sketches have traversing edges, or pencil strokes, which can be easily mimicked/modified by new random strokes. We refer to an edge/object that flows across a tile as a traversing edge/object of that tile. The images are first cropped to suitable sizes $x \times y$. For most tiles if the number of traversing edges is outside some certain interval the image is discarded. This image selection process guarantees that most of the tiles contains at least some traversing edges so that any of them can be *modified* to form some random shapes. Any image surviving such constraints is then referred to as the candidate image I_C.

3.2 Generation of Search Set and Displaying an Instance

At each *instance* of Movtcha, an I_C is selected to be tailored to generate θ_{sub}. For each *instance*, we randomly choose a search set size from an interval $[LL, UL]$. This interval is determined by the parameter *AmountOfSeperation*, AOS, which ensures that search set sizes differ by some random amounts at each *instance*. Larger offsets ensure sparser estimated VSTs, and result into higher $\triangle^A_{VST}$ by eliminating outliers (Sect. 5.1). We then randomly select a subset of tiles $\theta_{sub} \in \theta$. We refer to the boundary of each tile t_i in θ_{sub} as b_{t_i}. We find the continuity points $b^{\{P\}}_{t_i}$ of the traversing edges at the boundary b_{t_i} by applying Canny. If a pair of tiles $\{t_x, t_y\} \in \theta_{sub}$ share the same boundary b_{t_i}, $i \in \{x, y\}$, then they also share the same edge continuity points $b^{\{P\}}_{t_i}$, $i \in \{x, y\}$. Once the edge continuity points are found, some of the traversing edges in each tile are almost dissolved by minimizing the intensity gradient difference. We then draw strokes connecting those points randomly. These strokes are approximation curves drawn across $\{p^i_x,\ p^i_y\}$, with varying number of control points randomly set in the vicinity of the center of the *exotic* tile t_i. As a result each time a stroke is drawn, a random shape is formed. A stroke might also end abruptly midway without connecting the points. Details in Appendix A. The grayscale intensities of all the tiles $\theta = \{c_1,\ c_2, \ldots,\ c_{|\theta|}\}$ are then randomly changed. And finally a grid like structure is drawn on the image I_C. When presented to the user the image size is scaled with a nonlinear bicubic interpolation and by adding some random noise. At this stage, the candidate image I_C is referred to as the processed image I_P.

Displaying an Instance. If there are n *instances* in a Movtcha, n processed images $\{I^1_P,\ I^2_P, \ldots,\ I^n_P\}$ are formed from n candidate images, with $\{\theta^1_{sub},\ \theta^2_{sub},\ \ldots,\ \theta^n_{sub}\}$ as their corresponding search sets, where

$|\theta^1_{sub}|<|\theta^2_{sub}|<\ldots<|\theta^n_{sub}|$. From each of these search sets the corresponding t_c are selected s.t. $|\theta^{1,\ P_{t_r}}_{sub}|<|\theta^{2,\ P_{t_r}}_{sub}|<\ldots<|\theta^{n,\ P_{t_r}}_{sub}|$. A random permutation $\pi : [n] \rightarrow [n]$ is selected and applied to the processed images $\langle I_P^{\pi(1)},\ I_P^{\pi(2)}, \ldots,\ I_P^{\pi(n)}\rangle$ and the challenge tiles $\langle t_{c_{\pi(1)}},\ t_{c_{\pi(2)}}, \ldots,\ t_{c_{\pi(n)}}\rangle$. At the i^{th} *instance* of Movtcha, a processed image $I_P^{\pi(i)}$ and target tile $t_{c_{\pi(i)}}$ is selected and displayed to the user.

4 Security Analysis

We analyze the success probability of an attacker in random guessing attack and automated attack. Section 7 provides a discussion on static relay attack.

We consider that an automated attacker uses a framework f specially designed to attack our system. It can (1) separate out the background and the foreground objects and identify moving challenge tile and grid tiles centroids in negligible time, and (2) perform A_{resp} and A_{rew} action at a desired speed while mimicking human user's mouse dynamics (such as addition of jitters). At each *instance*, the attacker matches the tiles using f and generates VST by guessing the *concealed* $|\theta^{P_{t_r}}_{sub}|$. In such scenario for n instances there should be $n!$ ways of varying the VST. The probability for a successful attack thus becomes $\frac{1}{n!}$. Movtcha involving 8 *instances* results in 0.0025 % success rate, much smaller than the target probability for a practical Captcha system security of 0.6 % [7]. However, in real settings, we allow some *tolerance* on the VST *trend* and subsequently on $\triangle^A_{VST}$ to accept *trends* with possibly a few outliers. Appendix A, Fig. 2(a) shows a simulation of how the success probability varies with $\triangle^A_{VST}$ for varying number of *instances*. In practice, f needs some processing time (such as separating the background and foreground objects). This and other similar processing time can also be upper bounded based on *condition(2)* (Appendix A, Fig. 2(b, c)). On the other hand, a random guessing attacker drags and drops t_c onto t_r randomly. Since the grid size is θ, the success probability of a random guessing attack, without *tolerance*, is very small $\frac{1}{\theta^n \times n!}$. Even with *tolerance* this probability remains small.

The current challenge in Movtcha is always independent of the past challenges. Since object type (*exotic* tiles) are randomly generated, object recognition or classification is apparently a hard problem in Movtcha. Ideally, the same tile could have produced infinitely many random shapes, as it is processed each time. Movtcha presents unique image at each *instance*, making the challenges independent of each other. The attacker is then left to exploit the low level cues in order to identify the *exotic* tiles. We discuss in details the possible attacks and the associated empirical results.

Attack Using Low-Level Cues. The *exotic* tiles in θ_{sub} might differ in grayscale intensity from its neighboring tiles due to the modifications applied. Considering there is no grid like structure, the attacker can therefore, use off-the-shelf edge detection algorithm such as Sobel or Canny to figure out the boundaries of the *exotic* tiles. A simple approach of hindering such naive attack is to introduce false tiles boundaries by randomly changing the tile intensities across

the grid. With the grid structure in position such gradient-based methods detect the whole grid (Appendix A, Fig. 1(b)). So we sought to a customized boundary detection approach similar to [7]. The image is first smoothed by a 5×5 Gaussian filter in order to reduce noise. We consider squares for each location, along the tile boundaries. The squares are then divided into halves at 0°, 45°, 90°, 135°. The goal is to have a large enough square so that any pair of halves covers portions of the neighboring tiles pixels (grid pixels being symmetric on both halves). The difference in the gray-scale intensity between the two halves of the square is then estimated by calculating $\triangle(h_1,\ h_2) = \frac{1}{2}\sum_{n=1}^{\#bins} \frac{(h_{1n}-h_{2n})^2}{h_{1n}+h_{2n}}$, where h_1 and h_2 represents the gray-scale intensity histogram of the two halves respectively. Gradient direction and magnitude of a location are set as the direction with the maximum grayscale intensity and the maximum intensity respectively. We then apply non-maximum suppression and threshold the resulting image incrementally until the candidate tile set size, $|C_T|$ converges to $|\theta_{sub}|$, at which point if $C_T = \theta_{sub}$, the attack would be considered effective. Any $c_{t_i} \in C_T$ has a boundary *edge* weight of $w_i > p_i/2$ (p_i is the *shared* perimeter of c_{t_i} with other tiles). Attacks on 100 I_C's with $|\theta_{sub}| = 10$, $|\theta| = 48$ resulted into $\left(\frac{|C_T \cap \theta_{sub}|}{|\theta_{sub}|}\right) = 0.039$(average). Edge density on opposite side of the boundaries remains almost similar due to the false traversing edges constructed randomly among the edge continuation points. Any local artifacts at the tile boundaries, that would have been exploited by an edge traversing algorithm, are concealed by the grid structure (3-pixel width). Besides, finding out the edge continuation points at the processing stage, using Canny edge detection algorithm minimized the distance between actual edges in the image and the edges found. Appendix A, Fig. 1(c) shows a contour plot of an I_P where intensity depth varies across θ providing no useful information to an attacker. On the other hand, scaling and adding random noise before the image is presented prevents the attacker from exploiting any noise or quantization patterns.

Attack Using Memory. We define *memory* as an accumulation of low level features from more than one *instance*. However, since unique images are used for each *instance* the current challenge is always independent of the past challenges, resulting into absolutely *no* accumulation of *memory*. We now look into another attempt of acquiring such *memory* using a Context-Based Image Retrieval (CBIR) system S_C, s.t. S_C retrieves the original image I_C from I_P. However, the amount of irreversible distortion added to the processed image I_P by the grid g, cut-and-scale, random strokes essentially thwart a CBIR system like [18] in retrieving the original I_C (as tested on 100 I_P's).

5 Experiments and Results

We carried out three experiments to evaluate Movtcha in terms of (1) Design and presentation, (2) & (3) Accuracy, efficiency and usability. The first experiment was carried out in a controlled condition to ascertain some parameters and design of Movtcha. While the second and third were carried out in a non-controlled condition, mimicking a real life Captcha solving scenario. We obtained

approval from the Research Ethics Board of our Institution for the experiments. All experiments were divided into three phases (1) Phase-I, where participants agreed to the consent information. (2) In Phase-II, they were instructed to solve Movtchas. (3) And in Phase-III, participants were required to fill up an exit survey consisting of the standard SUS (Simple Usability Scale) questions [19] and a few other related questions S_{Fun}.

5.1 Experiment I: Design and Presentation

The goals of the first experiment were to figure out (1) how the *intra-accuracy*, $\triangle^A_{VST}$ varies with the size of the grid and (2) the parameters λ for $C3$ and ζ for $C4$. $C3$ & $C4$ were therefore not set in this experiment. The goals required human users to solve Movtcha in a laboratory condition i.e. in a non-distracting environment using a single platform. The experiment consisted of a pool of 24 students comprising of equal number of males and females aged between 21–36. All of them used a PC with 2.10 GHz Intel i3, 4 GB RAM and an wireless optical USB mouse. They used Google Chrome on a screen of resolution 1366 × 768 (96 PPI) in Windows 7 SP1 OS.

Setup. A short video showed how the game is played at the beginning. Participants received no further instructions. Each participant was required to solve 15 Movtchas $\langle M_1,\ M_2, \ldots,\ M_{15}\rangle$ each comprising of a fixed number of instances ($\#instances = 8$). The 15 Movtchas were divided into three groups. The group G_1 consisted of images divided into $5 \times 5 = 25$ ($Row \times Column$) tiles each of size 60×60 pixels. Similarly, G_2 and G_3 consisted of the same size tiles with images divided into $6 \times 6 = 36$ and $8 \times 6 = 48$ tiles. t_c holder moved randomly inside the bounding box at 0.1 pixels/frame @60 FPS i.e. 6 pixels/s.

Results. It is observable from Table 1 that the average $\triangle^A_{VST}$, increased with increasing grid size. *AmountOfSeperation* increased with grid size, resulting into VST with increasing standard deviation. Noise fail to affect the $\triangle^A_{VST}$, when VSTs are sparser. Similar results can be observed in Neisser [13] where sparser target positions P_{t_r} results into VST with larger offsets. Therefore, in order to have larger $\triangle^A_{VST}$, Experiment-II & III were carried out using images with 8×6 tiles. G_3 has the highest $\triangle^A_{VST}$ and a relatively longer completion time, T_{Com}. T_{Com} decreases with decreasing $|\theta|$ as expected. E_c is the average error rate per click and refers to the ratio of number of times user missed picking up or dropped midway t_c or s_r during A_{resp} or A_{rew} to the total number of actual $(A_{resp} + A_{rew})$ actions in a Movtcha. The observed E_c suggested that users felt comfortable with the moving speed of the tile. I_d refers to the ratio of number of *instances* discarded (due to $C2$, $C3$, & $C5$) to the total number of *instances*. Although the average E_c remained almost unchanging among the groups, the average I_d^{C2} was relatively higher in G_1, an implication that as the users get familiar with Movtcha, they tend *not* to skip targets during search. Further analysis of data from G1, showed that I_d^{C2} was highest for the 1^{st} Movtcha, M_1, for 79.17 % of the users. The average I_d^{C5} remained as low as 0.052 suggesting comfortable visual search task performance in the current visual field.

Fixing Parameters. In an ideal condition $PT_{A_{rew}}$ is supposed to be zero, since dragging the star, A_{rew} action, does not involve any cognitive process. As can be observed from Table 1, the average $PT_{A_{rew}}$ is <100 ms, implying an almost smooth non-distracted movement during A_{rew}. We set $\zeta = 0.2\,s$ of $C4$ to possible extreme outliers using interquartile range $PT^{Q_3}_{A_{rew}} + (3 \times PT^{IQR}_{A_{rew}})$, to allow some tolerance in non-controlled condition. The ratio of VST to $|\theta^{P_{tr}}_{sub}|$ is the *true* inspection time, IPT, for each *exotic* tile. The average inspection time of all the participants throughout the 3 *sessions* was 102.1 ms. The parameter λ of $C3$ is similarly set to extreme outliers values for non-laboratory conditions s.t. $\lambda = (|\theta| * IPT') + MT'_{rew}$ where $IPT' = IPT^{Q_3} + (3 \times IPT^{IQR}) \approx 0.27\,s$ and $MT'_{rew} = MT^{Q_3}_{rew} + (3 \times MT^{IQR}_{rew}) \approx 2.0\,s$ for all *instances* with varying $|\theta^{P_{tr}}_{sub}|$. Therefore, λ provided comfortable time span while searching at any *instance* and only triggered $C3$ when the user is distracted (or "lazy" searching) for a relatively long time.

5.2 Experiment II: Accuracy and Efficiency

The goal of this experiment was to determine (1) how the *intra-accuracy*, $\triangle^A_{VST}$, and *inter-accuracy*, $\triangle^A_M$, varied with the number of *instances* and (2) the efficiency or completion time of a Movtcha. *Inter-accuracy* is the ratio of number of solved Movtchas to the number of Movtcha challenges. Unlike, experiment-I, this experiment was carried out in a non-controlled condition.

Setup. The users were emailed to solve Movtcha on 3 *occasions*. At each *occasion* the users were required to complete 5 Movtchas with (1) fixed image size, 8×6, (2) fixed parameters ζ, λ. They can make a maximum of 3 mistakes happening due to $C2 - C5$ while solving a Movtcha. For each *occasion*, *#instances* is varied from 6–8. They completed the exit survey as well.

Results. We collected complete submissions from 42 participants/workers. It can be observed that the average $\triangle^A_{VST}$ in all three cases is around 80 %. However, considering the success probability of a bot is tuned to a particular value, the allowable decrease in $\triangle^A_{VST}$ is larger for increasing *#instances* (Appendix A, Fig. 2(a)). This resulted into higher $\triangle^A_M$ for increasing *#instances*. The average I_d is higher relative to Experiment-I. Activation of $C4$ led to relatively higher click error, E_c. These increases were expected in non-controlled condition. Furthermore, it implies that the constraints proved to be useful in discarding abnormal VST and $MT_{A_{rew}}$ that might have resulted from distractions. Table 2 shows $\triangle^A_M$ when $\triangle^A_{VST} > 80\,\%$. Even when $\triangle^A_{VST}$ is set at $\geq 82.14\,\%$, limiting bot success to 0.85 % for *#instance* = 8, the *inter-accuracy* $\triangle^A_M$ is around 81 % (Gmail's Captcha accuracy rate 82.8 % [7]). Mouse type statistics from exit survey include wireless/wired mouse (61.9 %), laptop touchpad (38.1 %). Recall, VST is calculated using subtraction method [16] which allows VST to self adjust for the user's specific environment.

Table 1. Results from Experiment-I

	$\triangle^A_{VST}$	I^{C2}_d	T_{Com}(Std)	E_c	$PT_{A_{rew}}$
G1	67.94(4.81)	0.110	17.19(1.89)	0.058	76.8
G2	78.40(4.28)	0.078	21.90(2.08)	0.051	40.1
G3	85.10(4.43)	0.073	24.96(2.52)	0.062	44.2

Table 2. Results from Experiment-II

inst	$\triangle^A_{VST}$(Std)	$\triangle^A_M$	T_{Com}(Std)	I^{C2}_d
6	80.22(4.76)	67.14	21.57(5.02)	0.21
7	79.97(7.40)	71.9	26.43(7.58)	0.29
8	82.90(6.77)	80.96	28.11(6.03)	0.24

5.3 Experiment III: Accuracy and Efficiency

The goal of this experiment was to observe how random users from different parts of the world perform on Movtcha through Amazon Mechanical Turk [20]. A single HIT was created with 70 assignments to have 70 unique workers. The workers were directed to the website hosting Movtcha. After watching the video, they were required to solve one Movtcha and collect 8 stars (complete 8 *instances* successfully) and in the process can make a maximum of 3 mistakes. Afterwards, they completed an exit survey. There was one demo *instance* at the beginning which was not considered in accuracy calculation. Workers were then required to copy-paste a code (generated on our website) back to Amazon to get paid $\$0.3$. The average *intra-accuracy*, $\triangle^A_{VST}$, is 78.2 % (std 6.03 %). When $\triangle^A_{VST}$ is set at ≥ 75, limiting the success probability of bot to 2.38 % the *inter-accuracy* is $\triangle^A_M = 78.9\,\%$. The average time to complete T_{com}, is 38.04 s (std 8.63 s). We highlight that these results are reported from unique users solving just *one* Movtcha for the first time. The average time required to complete each assignment was 5.4 min. Workers participated from 7 different countries (based on IP) with 84.3 % using mouse and rest touchpad (user-claimed).

5.4 User Experience

The average SUS score for all the experiments were within the user-friendly industrial software ratings [21]. The average SUS score for Amazon workers was 66.17. The S_{Fun} questions asked the user to rate the game in (1) "fun to play" (2) "easy and intuitive" (1–5, "5" signifying "Strong agreement" and vice versa). SUS and S_{Fun} rating was relatively higher in Experiment-I. This might be the result of users being more "polite" under a supervised condition. Considering Experiment-II & III, 57.1 % & 67.14 % of the participants agreed that the game was fun to play and 71.4 % & 74.2 % felt it was easy and intuitive respectively. This demonstrates that Movtcha is a user-friendly system.

6 Relay Attacks

We consider attacks, where the bot takes snapshots of *instances*, send it to a human solver and subsequently uses the responses to solve Movtcha [8]. In such attacks, the bot needs to consider four time intervals for each *instance*, (1) The communication delay between the bot and the human solver's machine $\triangle^c_t$,

(2) the time taken by the human solver to perform the visual search task and provide P_{t_r} and $|\theta_{sub}^{P_{t_r}}|$. (3) the time taken for the bot to match the tile. (4) the time taken to move the star back to P_{t_c}.

Captchas with dynamic challenge objects are generally resistant against relay attacks because the object co-ordinate sent by the human solver, C_{t_c}, at time t mismatches with that of P_{t_c} of the moving object at $t + k,\ k > 0$ [8]. The probability that $C_{t_c} = P_{t_c}$ at $t + k$ can be given as the ratio of the object area and the bounding box area where it randomly moves. In our setting, there is roughly 1/5 chance that the bot correctly picks up t_c. Such chance should produce relatively higher E_c and I_d in relay attacks. We carried out a small scale experiment with 10 users from the 1^{st} pool to examine our hypothesis. We considered a *strong* relay attack scenario where $\triangle_c^t = 0$. Therefore, the task of the human solver is to respond with $|\theta_{sub}^{P_{t_r}}|$ and C_{t_c}. To setup the experiment, *snapshots* $(s_1, s_2, \ \ldots s_n)$ at time $(\tau_1, \tau_2, \ \ldots, \tau_n)$ of the HTML5 canvas were taken along with the co-ordinates of t_c for 3 Movtchas. During the experiment, users were provided the *snapshots* one by one along with a beeping sound (audio stimulus) for a ready alert, similar to [8]. As soon as s_i is presented with the stimulus the user performed search and clicked on t_r and then C_{t_c} consecutively. If $P_{t_c} \neq C_{t_c}$ then the user is presented with the same s_i and were required to provide *only* a new C_{t_c}. There was a significant increase in $E_c = 2.81$ (avg) resulting into longer VST and $C3$ activations. None of the users were able to authenticate in the 3 Movtchas, with a maximum of 3 mistakes. In a real setting, where $\triangle_t^c \neq 0$, constraint $C3$ puts an upper bound on the distance between the relay bot and the human solver.

7 Related Work

May be the work closest to ours, which claims to consider human behavioral analysis is the dynamic game Captcha owned by a startup company called "Are you a Human" [22]. The company claims to differentiate human and machine based on behavioral data such as mouse events [8,22]. Their system challenges the user to drag-drop semantically related objects such as "baby" & "milk". *Clear form* Captchas are supposed to be inherently language and culture independent, because the response to a challenge is basically the challenge itself. Semage [4] claims to surpass the boundaries of languages but fails to auto-generate the challenge database. IR such as Cortcha [7] can automatically generate the image database. However, it also requires prior knowledge on the relationship between the decoy object and the inpainted image. On the other hand, our system demands the user to have the minimum ability of distinguishing the *exotic* tiles, conveying no meaning, from the original image. And this can be done with apparent ease and without establishing any form of semantic relationship [15].

Visual Search in Cognitive Psychology. Neisser [13] carried out an experiment where users were instructed to find the *absence* of letter Z in a list. The list contained 49 items like JZTXVB, DQFJHZ, ZXLSMT and one target item

VXRLFH arranged in a column. 15 such lists (with random target item position and random strings of letters) were given to each human user. It was observed for each user that the visual search time has a positive linear relationship with the position of the target item inside the list of 50 items.

8 Conclusion

We have provided a new approach to Captcha by estimating a cognitive feature. Human behavioral analysis was used to eliminate noise in the feature estimation process. Our empirical results suggest comparable accuracy, efficiency and usability to existing Captcha systems. We have discussed how image selection, challenge generation and response evaluation are automatically accomplished by our system. Movtcha maintains real world security while presenting *clear* answers to challenges. This attribute makes Movtcha language, culture and experience independent.

Acknowledgments. This research is in part supported by Alberta Innovates Technology Futures and Telus Mobility Canada.

A Generation of Search Set

We refer to the boundary of each tile t_i (to be made *exotic*) as b_{t_i}. We find the continuity points $b_{t_i}^{\{P\}}$ of the traversing edges at the boundary b_{t_i} by applying Canny. The gray values of t_i is changed to be within $[\alpha, \beta]$, until the number of traversing edges fall below $b_{t_i}^{|\{P\}|/2}$. α is set to the min and β to max gray value of t_i. At each step j, $(\alpha{+}{+}, \beta - -)$ any pixel value $> \beta$ and $< \alpha$ is set randomly to $(\alpha, \alpha + \delta]$ and $[\beta - \delta, \beta)$ respectively until $b_{t_i}^{|\{P\}|}$ decreases to $b_{t_i}^{|\{P\}|/2}$. δ is set s.t. $b_{t_i}^{|\{P\}|_{j-1}} > b_{t_i}^{|\{P\}|_j}$. For each pair of edge points $\{p_x, p_y\}$ obtained from $b_{t_i}^{\{P\}}$

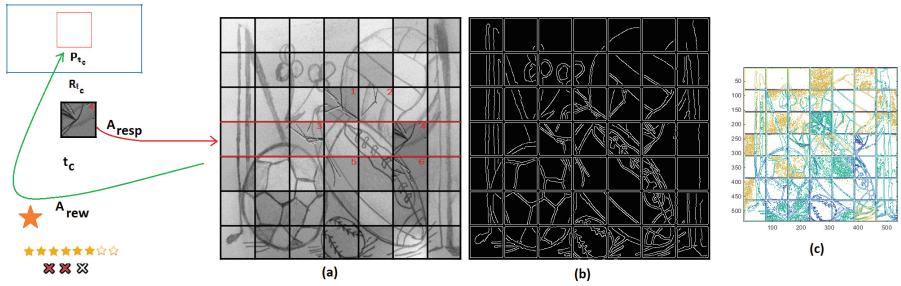

Fig. 1. Best viewed in soft copy. (a) An *instance* of Movtcha where user has collected 6 stars and made 2 mistakes. The search set size, $|\theta_{sub}| = 6$, and the position of target tile inside the search set, $|\theta_{sub}^{P_{t_r}}| = 4$. (b) Edges of I_P using Canny edge detection algorithm. (c) Contour plot of I_P.

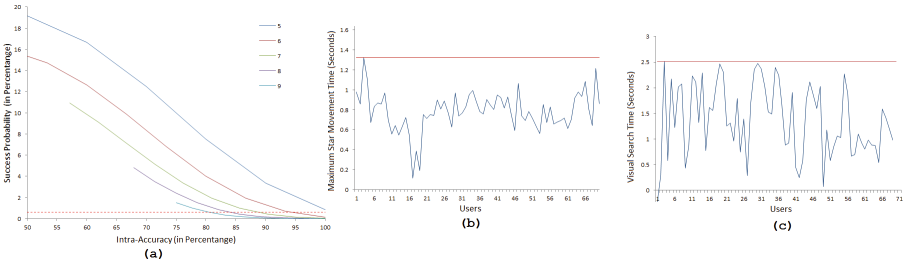

Fig. 2. (a) Simulation of the attacker's success probability for $\#instances$ =5–9. Dotted line represents 0.6 %. (b) Shows the maximum $MT_{A_{rew}}$ of successful Amazon users out of all the *instances* they played (c) Shows VST when $|\theta_{sub}| = 1$. If we restrict $MT_{A_{rew}}$ and VST ($|\theta_{sub}| = 1$) to 1.4 s and 2.5 s respectively as a constraint, we can upper bound the computational time of an attacker by 3.9s due to *condition(2)*. In other words, the attacker needs to successfully complete a visual search task within 3.9 s which involves separating objects from background, locating dynamic t_c, identifying search set size, and dragging-dropping t_c onto t_r. Successful users from Experiment-I and II provides even smaller bounds of 1.6 s and 2.1 s respectively. Most importantly, this constraint/bound can be set without interfering much with user's Movtcha solving activity. On the other hand, limiting computational time of attacker for traditional Captcha systems essentially limits the solving time of that Captcha.

a r-pixel width random stroke is drawn $\langle s_1, s_2, s_3, \ldots, s_r \rangle$, $s_i = Rand(s_i, s_i + \frac{\zeta}{r}]$. $s_1 = minGrayIntensity(I_C)$ and ζ is set as the difference between s_1 and the local (3×3) max gray value of $b_{t_i}^{\{p_e\}}$, $e \in \{x, y\}$. r is varied from 4–6. Each stroke is varied in intensity along its length to give it a sense of natural expression of pencil sketch. We set a small probability at each pixel that the stroke will stop at that pixel before joining a pair of continuity points.

References

1. Von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)
2. Elson, J., Douceur, J.R., Howell, J., Saul, J.: Asirra: a captcha that exploits interest-aligned manual image categorization. In: ACM Conference on Computer and Communications Security, pp. 366–374 (2007)
3. Datta, R., Li, J., Wang, J.Z.: Imagination: a robust image-based captcha generation system. In: Proceedings of the 13th annual ACM international conference on Multimedia, pp. 331–334. ACM (2005)
4. Vikram, S., Fan, Y., Gu, G.: Semage: a new image-based two-factor captcha. In: Proceedings of the 27th Annual Computer Security Applications Conference, pp. 237–246. ACM (2011)
5. Chew, M., Tygar, J.D.: Image recognition CAPTCHAs. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 268–279. Springer, Heidelberg (2004)
6. Esp-pix. http://server251.theory.cs.cmu.edu/cgi-bin/esp-pix/esp-pix

7. Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K.: Attacks and design of image recognition captchas. In: Proceedings of the 17th ACM conference on Computer and communications security, pp. 187–200. ACM (2010)
8. Mohamed, M., Sachdeva, N., Georgescu, M., Gao, S., Saxena, N., Zhang, C., Kumaraguru, P., van Oorschot, P.C., Chen, W.B.: A three-way investigation of a game-CAPTCHA: automated attacks, relay attacks and usability. In: Proceedings of the 9th ACM symposium on Information, computer and communications security, pp. 195–206. ACM, June 2014
9. Virtual sweatshop. Accessed 23 September 2014. http://krebsonsecurity.com/2012/01/virtual-sweatshops-defeat-bot-or-not-tests/
10. Sternberg, R.J.: Cognitive Psychology. Cengage Learning, Belmont (2011)
11. Wickens, C.D., Gordon, S.E., Liu, Y.: An Introduction to Human Factors Engineering. Pearson Prentice Hall, Upper Saddle River (2004)
12. Van Zandt, T., Townsend, J.T.: Self-terminating versus exhaustive processes in rapid visual and memory search: an evaluative review. Percept. Psychophysics **53**(5), 563–580 (1993)
13. Neisser, U.: Decision-time without reaction-time: experiments in visual scanning. Am. J. Psychol. **76**, 376–385 (1963)
14. Liu, Y.: Interactions between memory scanning and visual scanning in process monitoring. Ann Arbor 1001, 48109-2117 (1995)
15. Cave, K.R., Wolfe, J.M.: Modeling the role of parallel processing in visual search. Cogn. psychol. **22**(2), 225–271 (1990)
16. Donders, F.: 1868 Over de snelheid van psychische processen. onderzoekingen gedaan in het physiologisch laboratorium der utrechtsche hoogeschool, (1868–1869). Tweede reeks, ii: 92–120. Reprinted as donders, Franciscus, C.: On the speed of mental processes. Acta Psychologica **30**, 412–431 (1969)
17. Treisman, A.: Properties, Parts, and Objects. Wiley, New York (1986)
18. Google search by image. Accessed 23 September 2014. http://images.google.com/imghp?hl=en
19. Brooke, J.: Sus-a quick and dirty usability scale. Usability Eval. Ind. **189**, 194 (1996)
20. Amazon mechanical turk. Accessed 23 September 2014. https://www.mturk.com/mturk/welcome
21. Lewis, J.R., Sauro, J.: The factor structure of the system usability scale. In: Kurosu, M. (ed.) HCD 2009. LNCS, vol. 5619, pp. 94–103. Springer, Heidelberg (2009)
22. Are you a human. Accessed 23 September 2014. http://www.areyouahuman.com/