2015-01-21

# Utilizing Human Cognitive Abilities in User Identification and CAPTCHA

Galib, Asadullah Al

UNIVERSITY OF CALGARY

# Utilizing Human Cognitive Abilities in User Identification and CAPTCHA

by

Asadullah Al Galib

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

JANUARY, 2015

# Abstract

Secure authentication is necessary for everyday applications, such as logging in to a personal computer, making a financial transaction or boarding an aircraft. We present a novel approach to user authentication in which biometric data related to human cognitive processes, in particular visual search, working memory and priming effect on automatic processing, are captured and used to identify users. Our proposed system uses a carefully designed Cognitive Task (CT) that is presented to the user as a game, in order to capture a "cognitive signature" of the user. Our empirical results support the hypothesis that the captured cognitive signatures can identify users across different platforms. Our system provides a proof-of-concept for cognitive-based biometric authentication. We validate the robustness of our system against impersonation attack by experienced users, and show that it is hard to reproduce the cognitive signature by mimicking users' gameplay.

Ensuring that the access to a system is performed only by a human rather than a computer program or bot is another important security concern in online services. 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. We show that Movtcha is resistant against random, automated, inference and static relay attacks. Our system is suitable for large scale applications since image selection, challenge generation and response evaluation are automated. Movtcha 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.

# Acknowledgements

First and foremost, I would like to express my utmost gratitude to my supervisor, Dr. Reihaneh Safavi-Naini for her continuous support and guidance throughout my MSc. study. I am extremely grateful to her for all the constructive feedback and suggestions that helped my learning process and achieve my research goals. Since the beginning of my graduate study, I have been inspired by her enthusiasm and dedication towards research. I am very fortunate to have her as my supervisor. Without her expertise and guidance this would not have been possible.

I would also like to thank my examination committee members, Dr. Jeffrey Edwin Boyd and Dr. Svetlana Yanushkevich for their valuable comments and feedback on this thesis. Thanks to all the iCIS group members for their questions and comments during my project presentations. Also thanks to Deb for administrative support.

I gratefully acknowledge the financial support that I have received from the Department of Computer Science, University of Calgary, Natural Sciences and Engineering Research Council of Canada (NSERC) and Telus Mobility Canada in the form of Teaching Assistantship, Research Assistantship and scholarships.

This page would be incomplete without thanking my wife, Mehnaz Tarannum, who has always been supportive and patient. I am thankful to have her by my side to face all the challenges of life. I am also thankful to my parents for their encouragement and inspiration at every steps of my life.

# Table of Contents

**Abstract** . . . i
**Acknowledgements** . . . ii
Table of Contents . . . iii
List of Tables . . . iv
List of Figures . . . v
List of Symbols . . . vi
1 INTRODUCTION . . . 1
1.1 User Authentication . . . 2
1.2 Captcha to tell Computers and Humans Apart . . . 3
1.3 Human Cognitive Abilities . . . 4
1.4 Our Contributions . . . 4
1.4.1 User Identification using Human Cognitive Abilities . . . 4
1.4.2 Captcha using Human Cognitive Abilities . . . 6
1.5 Thesis Overview . . . 8
2 BACKGROUND AND RELATED WORK . . . 9
2.1 Biometric Authentication Systems . . . 9
2.1.1 Behavioral Biometrics . . . 11
2.1.2 Physiological Biometrics . . . 12
2.2 Properties of Biometric System . . . 13
2.2.1 Physiological vs Behavioral . . . 13
2.2.2 Identification vs Verification . . . 14
2.2.3 Continuous vs Static Authentication . . . 15
2.2.4 Resistance to Impersonation Attack . . . 15
2.3 Captcha Systems . . . 16
2.3.1 Properties of a Captcha System . . . 16
2.3.2 Captcha Classification . . . 17
2.4 Cognitive Processes . . . 20
2.4.1 Individual Difference in Cognitive Abilities . . . 20
2.4.2 Visual Search . . . 21
2.4.3 Self-terminating vs Exhaustive Search . . . 21
2.4.4 Serial Visual Search . . . 22
2.4.5 Factors Affecting the Visual Search Process . . . 22
2.4.6 Guided Search Theory . . . 24
2.4.7 Working Memory . . . 24
2.4.8 Priming Effect on Automatic Processing . . . 26
2.5 Related Work . . . 27
2.5.1 Related Works in Behavioral Biometrics . . . 28
2.5.2 Related Work in Image and Game Captcha . . . 31
2.5.3 Related Work in Cognitive Systems . . . 32
2.6 Statistical and Image Processing Tools . . . 35
2.6.1 Probability Density Function Estimator . . . 35
2.6.2 Bandwidth Selection . . . 39

2.6.3 Edge Detection Algorithm . . . 40
3 USER IDENTIFICATION USING HUMAN COGNITIVE ABILITIES . . . 42
3.1 System Design . . . 43
3.1.1 Design of the Cognitive Task . . . 43
3.1.2 The CT Constraints and Cognitive Processes . . . 48
3.1.3 Cognitive Features Estimation . . . 51
3.1.4 User Classification Technique. . . . 56
3.2 Security Model . . . 58
3.2.1 Error Metrics . . . 59
3.2.2 Impersonation Attacks . . . 59
3.3 Experiments And Results . . . 60
3.3.1 Experiment Setup . . . 60
3.3.2 Experimental Results and Analysis . . . 62
3.3.3 Evidence of Cognitive Processes . . . 69
3.3.4 Usability of Authentication Systems . . . 71
3.4 Discussion on User Identification System . . . 72
4 UTILIZING HUMAN COGNITIVE ABILITIES IN CAPTCHA . . . 73
4.1 Movtcha Design and Execution . . . 74
4.1.1 The Cognitive Task as Movtcha . . . 74
4.1.2 Cognitive and Behavioral Feature Extraction . . . 76
4.1.3 Telling Humans and Computers Apart . . . 78
4.2 Movtcha Challenge generation . . . 80
4.2.1 Selecting Images to be Tailored . . . 80
4.2.2 Generation of Search Set . . . 81
4.2.3 Displaying an Instance . . . 84
4.3 Security Analysis . . . 84
4.3.1 Random Attacks . . . 85
4.3.2 Automated Attacks . . . 85
4.3.3 Position Inference Attack . . . 88
4.3.4 Static Relay Attacks . . . 89
4.4 Experiments And Results . . . 90
4.4.1 Experiment Setup . . . 90
4.4.2 Experimental Results and Analysis . . . 92
4.5 Discussion on Movtcha . . . 96
5 CONCLUSIONS AND FUTURE WORK . . . 99
5.1 Thesis Summary . . . 99
5.1.1 A Novel Approach to User Authentication . . . 99
5.1.2 A Novel Approach to Captcha System . . . 100
5.2 Future Works . . . 101
5.2.1 User Authentication System . . . 101
5.2.2 Movtcha . . . 102
Bibliography . . . 103
A SYSTEM IMPLEMENTATION . . . 116
A.1 User Authentication System Implementation . . . 116
A.1.1 Data Acquisition Module . . . 116

A.1.2 User Identification Module . . . . . 118
A.1.3 Simulation Module . . . . . 121
A.1.4 Usability Module . . . . . 122
A.2 Movtcha System Implementation . . . . . 122
A.2.1 Data Acquisition Module . . . . . 122
A.2.2 Evaluation Module . . . . . 124

# List of Tables

3.1 Comparison of verification time, enrollment time, test and enrollment (train+test) session size for a particular user. . . . . 63
3.2 Ranking of features based on avg. EER. FAR and FRR at $\alpha = 0.5$ (Experiment-I) . . . . . 65
3.3 Comparison with other approaches. Mouse Dynamics System(MDS), Keystroke Dyanmics System (KDS), CBS (Cognitive-based Biometric System), HBS (Homogeneous physio-behavioral Biometric System) . . . . . 67

4.1 Results from Experiment-I . . . . . 93
4.2 Results from Experiment-II . . . . . 94
4.3 Notations lookup . . . . . 98

A.1 User Authentication Constraints look-up table (brief descriptions). Refer to Section 3.1.1 for details. . . . . 119

# List of Figures and Illustrations


2.1 Biometric System Architecture . . . . . 10
2.2 Captcha encountered while registering for a Gmail account . . . . . 18
2.3 Challenges from (a) Assira Captcha [35], (b) Semage Captcha [97] . . . . . 19
2.4 Guided Search: Find the equilateral triangle. . . . . 25
2.5 Baddeley's model of working memory with two storage subsystem serving the central executive co-ordinating system. . . . . 26
2.6 Sample shuffled *virtual* keyboard [50] . . . . . 30
2.7 Examples of grids for VPT. These grids are drawn by us as examples and are *not* necessarily the ones used in [31] . . . . . 34

3.1 (a) User is presented with a challenge tile, $t_c$, at the beginning of the $21^{st}$ *instance*. (b) User performs $A_{resp}$, i.e. drags and drops $t_c$ onto $t_r$ inside the grid. On a correct match the *loose* tiles disappear showing the current game status (at $21^{st}$ *instance*). (c) User performs $A_{rew}$, i.e. drags and drops gold coin, $g_c$, onto $P_{t_c}$. Top guiding line color changes from green to red as the coin touches the line. (d) User successfully deposits $g_c$ and gets the next challenge tile. All 25 tiles are visible at this point. Notice that the 21 *unmovable* tiles in b and d have not changed their positions. All the *loose* tiles have changed their position (compare a and d). The target tile appears at its original position in the image in (d). . . . . 45
3.2 (a) Random symbols from our system . . . . . 47
3.3 (a) An instance of our game, when $A_{resp}$ is completed. The current status of the game is visible. (b) Example grid from a Visual Pattern Test [31]. Notice the similarity between the presentation of (a) and (b). . . . . 49
3.4 (a) Click error angle and click error distance (center to click point). (b) Drop error angle and drop error distance (from the click point to the center of the destination, red bordered tile). (c) Straightness measured as ratio of actual distance to the distance moved. . . . . 54
3.5 Intra-session Evaluation: Avg. FAR and FRR at $\alpha = 0.5$ with varying number of *instances*. . . . . 63
3.6 Intra-session Evaluation: Avg. FAR and FRR at $\alpha = 0.5$ with varying number of users. . . . . 64
3.7 Correlation coefficients [-1,1] of pairs of features in a color-coded plot. . . . 66
3.8 Results from Impersonation attack. Most *instances* are accepted as "own" rather than as the victims' (attack-1, 2 and 3). . . . . 68
3.9 Box plot before noise removal (whiskers set to 3) showing outliers, (a) Tile Pick Reaction Time (in ms) (b) Gold Pick Reaction Time (in second). . . . . 68

3.10 Users randomly chosen from Experiment-II. (a) Average sub-sequence length with a linear relationship between $VST$s and search set sizes indicating differences in working memory capacity of different users. Different curves represent varying number of violations from linear relationship. (b) Percentage of the three cases when prime is triggered (Section 3.4). Users are influenced in different ways. . . . . 70
3.11 Matrix representation of $n$ test sessions matched against $n$ templates. Threshold set at $\approx 0.5$ marked by red line. The horizontal bar shows the number of *instances* and corresponding color codes. . . . . 70

4.1 Best viewed in soft copy. (a) An *instance* of Movtcha where user has collected 6 stars and made 2 mistakes. The grid size is $|\theta| = 49$, search set size is $|\theta_{sub}| = 6$, and the position of target tile inside the search set is $|\theta_{sub}^{P_{t_r}}| = 4$. (b) Edges of $I_P$ using Canny edge detection algorithm. (c) Contour plot of $I_P$. . 76
4.2 A skipped search, triggered by $C2$, for an *instance*. Missed target tile is highlighted. Bounded boxes are not in scale. . . . . 78
4.3 Success Probability varying with $\triangle_{VST}^{A}$ *#instances (5-9)*. Dotted line represents 0.6% attacker's success probability. . . . . 88
4.4 (Experiment-III, $1^{st}$ HIT)(a) Shows the maximum $MT_{A_{rew}}$ of successful users out of all the *instances* they played (b) Shows $VST$ when $|\theta_{sub}| = 1$. If we restrict $MT_{A_{rew}}$ and $VST$ ($|\theta_{sub}| = 1$) to $1.4s$ and $2.5s$ 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 $3.9s$ 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 2.1s and 2.8s 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 scheme essentially limits the solving time of that Captcha. 96

# List of Abbreviations and Nomenclature

| Symbol | Definition |
|---|---|
| U of C | University of Calgary |
| CAPTCHA | Completely Automated Public Turing Test to tell Computers and Humans Apart |
| LOO LSCV | Leave-One-Out Least Square Cross-Validation |
| PDF | Probability Density Function |
| MISE | Mean Integrated Square Error |
| ISE | Integrated Square Error |
| CT | Cognitive Task |
| VPT | Visual Pattern Test |
| VST | Visual Search Time |
| EOP | Effectiveness Of Priming |
| P&S | Pause and Search |
| RT | Reaction Time |
| DPT | Drop and Pick Reaction Time |
| FAR | False Acceptance Rate |
| FRR | False Rejection Rate |
| EER | Equal Error Rate |
| FA | False Accept |
| FA | False Reject |
| TP | True Positive |
| TN | True Negative |
| MDS | Mouse Dynamics System |
| HBS | Homogeneous Biometric System |
| KDS | Keystroke Dynamics System |

| | |
|---|---|
| CBS | Cognitive based Biometric System |
| IR | Image Recognition Captcha |
| CR | Character Recognition Captcha |
| HTML | Hypertext Markup Language |
| JS | JavaScript |
| AJAX | Asynchronous JavaScript + XML |
| PHP | Hypertext Preprocessor |

# Chapter 1

# INTRODUCTION

Authentication is the process by which a system verifies the identity of an individual who wants to access it. Importance of secure authentication as the entry point to systems cannot be overestimated. Securing digital information is not only a way of keeping the intruders away. In fact, in the business world fraudulent authentication might result into unrecoverable loss of intellectual property or critical information, loss of revenue and loss of customers.

Another important concern of the digital age is related to restricting access to systems to humans only. CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) is a form of security test that restrict access to a website or other forms of digital resources to humans only. In other words, Captcha ensures that the access to a system is being performed by a human rather than an automated computer program or bot. Captcha is implemented to prevent automated account registration, automated online voting, dictionary attacks in password system and others.

**Thesis Statement.** We propose to use human cognitive abilities to differentiate among humans as well as differentiate humans from computers. In addition to meeting general system requirements, we show that both our systems surpass the security and usability aspects of their corresponding existing works.

Section 1.1 and Section 1.2 give short overviews on the current user authentication and Captcha systems, and their limitations. Section 1.3 provides a discussion on human cognitive abilities. We provide discussion on our proposed systems in the contribution Section (Section 1.4).

## 1.1 User Authentication

Traditional methods of authentication use "what we know", "what we have" and "what we are". In other words, the methods include: (1) knowledge that are supposed to be kept secret by individuals e.g. passwords and pass phrases, (2) physical possessions such as key, smart cards, passports, (3) characteristics of the human body such as fingerprint, iris and retina. More recent approaches to authentication use behavioral appearances or characteristics of individuals such as voice, walking gait, typing pattern and mouse dynamics. These modes of authentication can be used in combination. For example, an ATM card is a possession that requires knowledge (PIN) in order to carry out a transaction, a login system might require a combination of knowledge (password) and behavioral biometrics (keyboard typing pattern), a passport is a possession with a face picture and signature biometrics.

The most widely used form of authentication in today's computer systems is the use of alphanumeric PINs or passwords, since it does not require any special input hardware. However, these systems are vulnerable to forgotten secrets and guessing attacks [117, 14]. On the other hand, biometric systems [30] are immune to lost or theft. Nevertheless, the operation of these biometric systems requires the use of special hardware such as scanner, camera or microphone in order to collect the necessary biometric data. Among the behavioral based biometric systems keystroke and mouse dynamics biometrics are the exceptions, since any traditional computing system comes with a mouse and a keyboard.

One of the main drawbacks of keystroke and mouse dynamics system is that they require long enrollment and verification time. Therefore, most mouse dynamics systems [2, 118, 71, 79] proposed to date are used for continuous authentication rather than verifying the user at the beginning of a session. In addition, identification rather than verification of users in a mouse dynamic system is considered a more difficult task and exhibits higher error rates [13, 80].

## 1.2 Captcha to tell Computers and Humans Apart

Captcha [101] 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 [35, 29] were compromised. Captchas exploiting semantic relationships between images [97, 24] or between images and words [36], 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 less explored, is the issue of Captcha being language, culture or experience dependent. Text or Audio based Captchas are entirely language dependent. Some image-based Captchas [97, 119] though language independent, rely heavily on user's past experience or exposure to certain things (Section 2.5.2). 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. For example, imagine Gmail's text-based Captcha being provided in clear text. This means that both the challenge and response are 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 [101]. Another important security challenge in Captcha is preventing relay attack where the attacker (the bot) relays or sends the Captcha challenge to a human user who in turn solves the Captcha [99].

## 1.3 Human Cognitive Abilities

Cognition refers to higher level brain functions (or mental processes) such as perception, learning, problem solving and producing language [41, 91]. An individual's capacity in carrying out any cognitive task (a task requiring a mental process such as perception, thinking and reasoning [91]) is referred to as the cognitive ability of that individual. For example, a visual search task requires finding out a target object among other distractors. The visual search ability is generally measured by the search time. A working memory allows individual to hold information in mind and manipulate or process it. Therefore, one way of measuring the capacity of a working memory is by estimating the information processing speed. To estimate such abilities, the corresponding mental processes must be invoked first. This requires careful design of the experiments. We provide a detail discussion on cognitive processes in Section 2.4.

## 1.4 Our Contributions

Here, we discuss our research contributions in a user identification and a Captcha system.

### 1.4.1 User Identification using Human Cognitive Abilities

We present an authentication system that collects and utilizes biometric data that are intrinsic to human cognitive abilities. Similar to a scanner capturing the impression left by the ridges of a finger, our system is able to capture the "cognitive signature" generated by the user during a cognitive task. Our approach differs from behavioral biometrics which do not necessarily invoke any particular mental processes.

Francis Galton was the first to start the investigation of individual differences among people based on their cognitive abilities [42]. He studied a variety of cognitive abilities and in each case focused on measuring the ability and noting the variations among different individuals. Recent research on individual differences are focused on two distinct categories

[41] (a) individual differences in the *capacity* to *execute* a cognitive task (b) individual differences in the *manner* in which one *approaches* a cognitive task. The main motivation behind our work emerges from the study of individual differences in the *capacity* to *execute* cognitive tasks.

In our work, we present the user with a *cognitive task* (CT), which is executed by the user using a pointing device (mouse or touchpad). The raw data collected during execution of the CT, allow us to estimate features related to visual search ability, working memory and priming effect on automatic processing. Features derived from these cognitive processes in combination with other basic stimulus-response features are used to build unique profiles for individual users. The CT is presented to the user in the form of an interactive visual search game. The game starts by presenting the user with 25 different objects arranged in a 5×5 grid. The task of the user is to find a particular target object among other distracting objects. The user has to drag and drop the target object onto the matching object inside the grid. This act is equivalent to a correct visual search task by the user. On a correct search task (or correct match), the user is rewarded with an incentive, a gold coin. The user is then instructed to deposit this gold coin at a particular place (bank). On a correct deposit, the user is presented with another target object and a similar interaction follows. The estimated features derived from these interactions are used to construct an authentication scheme with accuracy comparable to other established biometric approaches. A typical behavioral biometric system such as those based on mouse dynamics, measure behavioral traits that are inadvertent. Systems designed to estimate cognitive features such as ours, can be augmented to use behavioral features related to mouse dynamics to improve authentication accuracy.

Our system provides a proof-of-concept for cognitive based biometric authentication. Attempts were made to ensure that the game is intuitive and interesting. The accuracies obtained in our system are comparable to other state-of-the-art behavioral biometric systems [44, 2, 118, 38]. To evaluate security of our system we considered an impersonation attack

where the attacker attempts to "mimic" a target user. We developed a web-based program which can simulate a user performing the authentication task, using the data collected from his data acquisition period. We then allowed the attackers to train themselves using the simulation and then attempt to mimic or impersonate the user

### 1.4.2 Captcha using Human Cognitive Abilities

Our Captcha, called Movtcha is presented to the user as a Cognitive Task (CT). It estimates cognitive ability of a human, in particular visual search ability. We consider visual searches that are serial [105] 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). 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 [73, 64].

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 or 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 the *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 through guided search.

Movtcha consists of a carefully tailored image and a challenge tile. An image is first divided into $\theta$ 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 the human user [105]. A target tile $t_r$ is then selected randomly from the search set, $\theta_{sub}$, 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 $\theta_{sub}$ is equivalent to a correct visual search task. A human easily distinguishes and recognizes these *exotic* tiles via parallel search, $\theta_{sub}$, and then performs serial search only on them i.e. on $\theta_{sub}$, to find $t_r$. 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. $\theta_{sub}$. 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 noises 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.

A Captcha based only on recognizing the *exotic* tiles out of the image might seem adequate at the first glance. However, such a Captcha does ***not*** (1) present challenge and response in *clear form*, (2) resist static relay attacks, (3) introduce the concept of using human behavioral and cognitive feature analysis to differentiate the two.

## 1.5 Thesis Overview

We discuss background and related works on biometric systems, Captcha systems and cognitive processes in Chapter 2. Chapter 3 provides detail discussion on our user authentication system. Then we provide discussion on our novel approach to Captcha Systems in Chapter 4. Finally, we provide conclusion and future works in Chapter 5.

# Chapter 2

# BACKGROUND AND RELATED WORK

This chapter has four sections. Section 2.1 gives an overview on the current biometric technologies. Section 2.2 provides a discussion on the properties of biometric systems. Then we provide an overview on the current Captcha systems and their properties in Section 2.3. In Section 2.4 we discuss about the cognitive processes that have been incorporated in our systems. The design of our systems are guided by the factors that invoke and affect these cognitive processes. As a result, we have referred back to Section 2.4 frequently from Chapter 3 and 4. Closely related works in behavioral biometrics, Captcha and cognitive processes are provided in Section 2.5. Finally, Section 2.6 provides discussion on the statistical learning algorithm and the image processing tools that we have used.

## 2.1 Biometric Authentication Systems

Biometric identification or verification refers to an automatic identification or verification of individuals based on a feature vector(s) derived from the physiological and/or behavioral characteristics of human [30]. Physiological biometric systems use *what we are* e.g. fingerprints, iris pattern, facial features, whereas behavioral biometric systems use *what we do* e.g. voice recognition, keyboard typing rhythm, mouse dynamics, walking gait, handwritten signature. The operation of most biometric systems require the use of special hardware such as scanner, camera or microphone in order to collect the necessary biometric data. Mouse and keyboard dynamics are two exceptions since any traditional computing system comes with a mouse and a keyboard.

Biometric system has two major phases: (1) Enrollment phase and (2) Matching Phase. During the enrollment phase, biometric information regarding the user is captured and stored

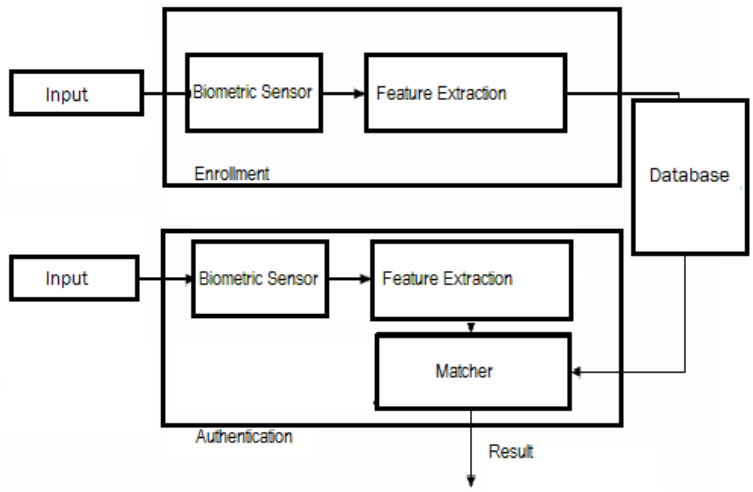


Figure 2.1: Biometric System Architecture

in the system database. In subsequent usage, the biometric information is again captured and compared with the existing information from the enrollment phase. The system then signals either a match or a no-match. The enrollment phase can be subdivided into the following stages: 1(a) An input sensor collects the necessary biometric data of the user. 1(b) A data module pre-processes the data and extracts the feature vectors to build the user profile or template. 1(c) The template is then stored somewhere (system database, card). On the other hand, the matching phase has the following stages: 2(a) An input sensor captures the biometric data for subsequent authentication attempts by the user. 2(b) A data module pre-processes the data and extracts the feature vectors. 2(c) A matcher compares the authentication data with the enrollment data and signals a match or a no-match. Figure 2.1 shows the general architecture of such systems.

To evaluate the correctness and security of a biometric system, the most commonly used metrics are the False Rejection Rate (FRR) and the False Acceptance Rate (FAR). The FRR is the percentage of the authorized users that are rejected by the system. Whereas, the FAR is the percentage of unauthorized users that are accepted by the system. A good identification or verification system should have a low FAR and FRR.

Biometric systems can either verify or identify a user. Identification involves compar-

ing the acquired biometric information against templates corresponding to all users in the database. Whereas, verification involves comparison with only those templates corresponding to the claimed identity. From now onwards, we will use the generic term *authentication* where we do not wish to distinguish between verification and identification.

### 2.1.1 Behavioral Biometrics

There are a variety of subdivisions within the behavioral biometrics domain. These systems have their own characterisitics in terms of enrollment requirements, ease of deployment, and authentication performance. Here, we provide short overviews on some recent behavioral biometric systems.

**Mouse Dynamics.**

A mouse dynamics biometric system uses a pointing device (mouse, touchpad) movement information such as speed, acceleration, trajectories, jitters to build a template of the user. The information can be collected while the user is interacting within a confined environment (e.g. playing a game) or when performing day-to-day task (e.g. browsing). This approach is well suited for remote authentication approach, since no specialized hardware is required. We provide a detail discussion on works related to mouse dynamics in Section 2.5.1.

**Keystroke Dynamics.**

A keystroke dynamics biometric system relies on the way the user types with a keyboard or a keypad type device. Certain attributes such as the time between striking successive keys, duration of key press are used to build templates of users. The user typing pattern can be captured either in a text-dependent or text-independent mode. The text-dependent mode requires the user to enter a fixed string (e.g. username, password, random string). On the other hand, the text independent mode allows the user to enter any string as they wish. Bergadano et al. [11] reported 0.01% FAR and 4% FRR on their keystroke based identification and verification system.

**Voice.**

In a voice recognition system speech patterns are analyzed to authenticate a user. The user is required to enunciate texts which can either be fixed or free. In a fixed-text mode the user is required to enunciate a specific phrase. Whereas, in a free-text mode the user can enunciate any phrase of his choice. The later approach requires model of each speaker making the system computationally more expensive [80]. Voice recognition system can have a low EER of 0.28% [26].

**Gait.**

Human gait is a person's manner of walking. A person's manner of walking can be determined by his weight, limb length, footwear, and posture combined with harmonic motions [61]. There are different approaches to extracting the gait features such as sensor-based and machine-vision based. For example sensor-based systems acquire dynamic data and use it for authentication [80]. Current trend is to focus on dynamic aspects of walking rather than the static features. Kale et. al [59] reported an accuracy of 90% on gait detection.

### 2.1.2 Physiological Biometrics

There are a number of physiological based biometric systems. All of these use some biological attributes of human being. They require the use of special hardware such as scanner, camera or microphone in order to collect the necessary biometric data. Below we provide overviews of some of the established physiological based biometric systems.

**Fingerprint.**

A fingerprint is an impression or mark made on a surface by a person's fingertip. The ridges, valleys and loops give each fingerprint its own uniqueness. Fingerprints are one of the oldest form of physiological biometric with a high matching accuracy of around 1% EER [22]. The user template consists of feature values related to the position and orientation of certain

critical points known as minutiae points [83]. Today, fingerprint scanners have become quite affordable and are even incorporated in laptops and mobile devices.

**Iris.**

The structures responsible for the patterns of the iris are largely completed by the eighth month of gestation. The complex pattern of the iris can contain many distinctive features such as arching ligaments, furrows, ridges, crypts, rings, corona, freckles, and a zigzag collarette [28]. Iris scanning can be done from several meters away. However, external factors such as light, focus, resolution and contrast must be carefully balanced in order to extract a good feature vector from the image. Iris recognition system can reach 91% correct verification rate at a 0.1% FAR [77].

**Face.**

Face verification involves extracting facial characteristics such as nose shape, eye socket position, and distance between different facial elements from an image of the user's face and matching it with the template stored in a database. Face recognition system is non-intrusive and is suitable for covert recognition applications. However, it still faces challenges such as illumination and facial expression variance effects [90]. Face recognition system can have more than 90% accuracy [77]

## 2.2 Properties of Biometric System

### 2.2.1 Physiological vs Behavioral

Behavioral based authentication methods perform the identification task based on people's behavioral patterns. These systems are more acceptable to users and generally cost less to implement. System using mouse and keyboard dynamics do not require any specialized hardware at all. Most behavioral biometrics such as voice, mouse, keyboard can be deployed for remote authentication system. However, behavioral characteristics can be difficult to

measure because of influences such as fatigue and illness.

On the other hand, physiological based biometric system verify a person's identity by his or her physiological characteristics. In general, such characteristics are more stable than behavioral characteristics. However, these systems require specialized hardware and generally cost more to implement.

Both physiological and behavioral characteristics are intrinsic to users and are mostly immune to lost and theft. Current research in both the fields report high accuracy and usability [60]. Each system has their own strengths and weaknesses. Therefore, we should match a specific biometric to an application depending on the application's operation mode. For example, both fingerprint and iris-based techniques are more accurate then voice based technique. However, in a telebanking application, the voice-based technique can be integrated seamlessly with the existing telephone system [78].

### 2.2.2 Identification vs Verification

There are two means of authentication: (1) Identification and (2) Verification . In *positive* identification mode, the system recognizes an individual by searching the entire template database for a match. That means the system conducts a one-to-many comparison to establish the user's identity. On the other hand, verification is the process of verifying a user-claimed identity by comparing the captured sample only with that user's template. That means the system conducts a one-to-one comparison to determine whether the claim is true.

Therefore, an identification task is a much harder problem than verification since an identification system must perform a large number of comparison [78]. However, from the user's point of view identification systems are more convenient [60].

### 2.2.3 Continuous vs Static Authentication

In static user authentication, a user logs in to the system by providing his identity. His session remains valid until he logs off from the session. However, a continuous authentication system monitors if the current user is the same as the user who performed the initial static authentication. Physiological based biometrics provide static user authentication system. For example, fingerprint and iris scans, require a short time to capture the biometric data and make an authentication decision. On the other hand, most behavioral biometric system require longer sessions to make an authentication decision, which makes them suitable for continuous authentication. Therefore, more recent approaches to behavioral biometrics, especially mouse and keystroke dynamics, are focused on designing system with shorter verification time for static user authentication [80].

### 2.2.4 Resistance to Impersonation Attack

Perhaps biometrics' most infamous liability is vulnerability to impersonation [13]. Impersonation attack refers to the scenario where the biometric information of a legitimate owner is *somehow* generated by an attacker, without the owner being present at the scenario. For example lifting latent fingerprints from objects, observing and imitating typing pattern or mouse dynamics and others. These type of impersonations require attackers capable of surreptitiously observing the actual owner and reproducing the biometric information later. Impersonating a biometric is difficult compare to impersonating an identity by using a stolen password [13]. This is due to the intrinsic nature of physiological and behavioral characteristics of human. Nevertheless, any biometrics, especially behavioral biometric systems should be thoroughly evaluated against impersonation attacks. This is because in behavioral biometrics generating biometric information is relatively easy even when the owner is absent. For example typing on the keyboard or using the pointing device.

## 2.3 Captcha Systems

Captchas are used for differentiating humans and computers. We generally encounter Captchas while registering for an email or social network account. Captchas can be used in applications to prevent the following: (1) outgoing spam, (2) automated account generation, (3) brute force attack on passwords, (4) phishing attack, (5) blog and forum spam, (6) artificial product ratings, (7) voting in online polls. On the other hand, one can utilize the Captcha system in order to improve (1) tagging of images by learning tags from legitimate users who solve image-based CAPTCHAs and (2) digitizing old books by using text from the books, that a computer struggles to digitize, as text-based Captchas.

### 2.3.1 Properties of a Captcha System

We provide a set of properties for a typical Captcha system building off on the recommendations of [102, 119] and scrutinizing the advantages and disadvantages of some existing Captcha systems [35, 97, 69].

**Scalability: Auto-generation of Database.**

Any Captcha system should require the least amount of manual interventions. In other words, the challenge selection, generation, and response evaluation must be fully automated to meet the demand of a large scale system.

**Security: Resistance Against Random, Automated, Relay Attacks.**

The system should be resistant against three major types of attacks. (1) Random Guessing, where the bot attempts to solve the challenge in a random fashion. (2) Automated attacks, where the bot attempts to use clever algorithms to solve the challenge. (3) Relay attacks [69] where the bot relays the challenge to a human attacker, who aids in solving the Captcha.

**Usability: Language and Experience Independence**

Any system requiring users to interact with computers must be made easy and intuitive. This is also applicable to any Captcha systems [114]. In addition, a Captcha should be made as independent as possible of the user's language, culture, exposure to certain experience, geographical location and educational background.

### 2.3.2 Captcha Classification

There are various forms of Captchas ranging from text-based to game-based Captchas. Here, we group them into five major categories. For each form of Captcha, we first discuss how they work. Then we discuss their advantages and drawbacks taking into consideration the aforementioned properties. For related works refer to Section 2.5.2.

**Text-based Captcha.**

Text-based Captcha is the oldest and the most popular form of Captcha. A user is presented with a distorted alphanumeric string. The task of the user is to type out the sequence of characters. Figure 2.2 shows an example of a text based Captcha encountered while registering for a Gmail account. One of the first implementations of a text-based Captcha was developed in 1997 by AltaVista. They needed a way of preventing the automatic adding of URLs to the website for indexing [6]. The security of a text-based Captch relies heavily on the difficulty of the character segmentation task. Attacks on the text-based Captchas typically involve optical character recognition tools [113, 116]. Attempts on making the text Captcha more immune to attacks include the introduction of systematic noise and distortion to the challenge to make the character segmentation task harder for a bot. However, heavy distortion usually leads to poor usability. Therefore, while designing text-based Captchas one has to be aware of the trade-off between security and usability.

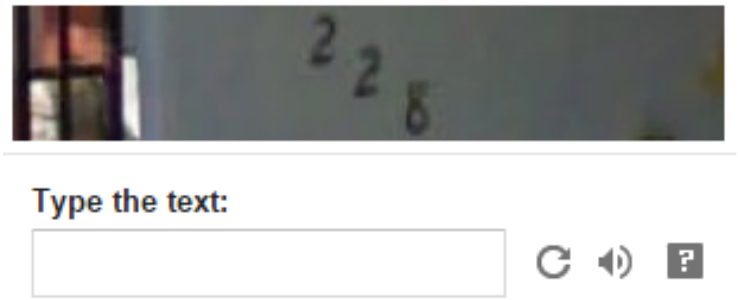


Figure 2.2: Captcha encountered while registering for a Gmail account

**Image based Captcha.**

Image based Captchas consist of images or objects. The user is generally required to identify or recognize some images from the others. For example in Asirra [35] users are required to distinguish cats from dogs, Figure 2.3(a). Other image-based Captchas require establishing the semantic relationships between images or objects. For example in Semage [97](Figure 2.3(b)), the user is required to find semantic similarity between images e.g. a real image of a tiger and a cartoon tiger. Image-based Captcha can have either a static or a dynamic image. For example, NuCaptcha [74] consists of distorted text moving on a dynamic background. The advent of image-based Captcha offered an alternative approach to traditional text-based Captcha with promising usability and security. However, eventually schemes based on image recognition such as [29, 35] were compromised. Captchas exploiting semantic relationships between images [24, 97] or between images and words [36], usually claim high usability but fail to auto generate the challenge database.

**Dynamic Captcha.**

More recent approaches to Captcha involves interactive Captchas, where users interact with static or dynamic images or objects by clicks, screen touches, or drag-drop actions. These dynamic Captchas can be image recognition, classification or object matching tasks. For example in Sketcha [82] user is required to rotate (using a pointing device) a selection of 3D objects until they are upright. In [3] users have to drag and drop semantically matching

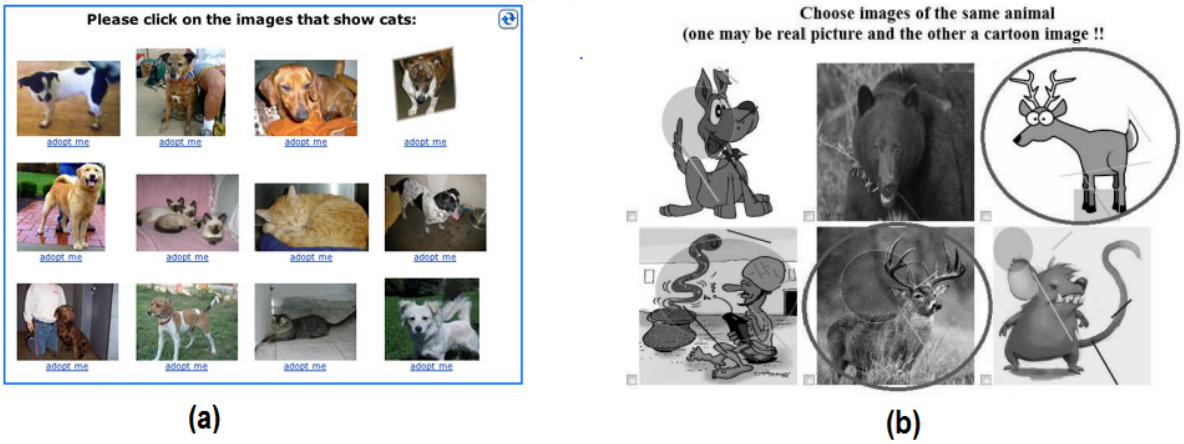


Figure 2.3: Challenges from (a) Assira Captcha [35], (b) Semage Captcha [97]

objects such as a "horse" and a "saddle". Due to the nature of the drag and drop actions or clicks, some of these Captchas can be categorized as game Captcha. Game Captchas like [69] with dynamic challenge objects are resistant to relay attacks [69]. This is mainly because the position of the challenge object changes with time. Therefore, the position of the moving object send by the human at time, $t$, mismatches with that of the current position of the object at time $t + k$, where $k$ is the time taken for the answer to reach the relay bot from the human sender. However, generating challenge objects that are semantically related to each other requires human interventions and results into scalability problem.

**Audio-based Captcha.**

An audio Captcha exploits the observation that a user can recognize familiar speech easily in a noisy environment compared to a computer program. In fact, we need a signal-to-noise ratio of approximately 1.5 dB to recognize and understand speech [92]. Whereas, automatic speech recognition programs require a signal-to-noise ratio between 5 and 15 dB [110]. Therefore, humans are able to comprehend speech easily in a noisy background whereas the bots find it difficult. Vision-impaired users are more likely to benefit from audio Captchas. However, audio Captchas are language dependent. Moreover, recent works suggest that they suffer both security and usability problems [19, 20].

**Linguistic Captcha.**

Linguistic CAPTCHAs include challenges that are rendered in standard character encoding schemes, such as ASCII. The text appears as normal, inline text on a web page. For example "If yesterday was Saturday, what day is today?". When presented in plain text, such challenges could be solved by users with visual and/or hearing disabilities by using Braille readers (where necessary). However, the major drawback of such Captchas is that their challenge database is extremely difficult to auto-generate. Moreover, this type of Captcha is language-dependent.

## 2.4 Cognitive Processes

In this section, we first provide a discussion on individual difference in cognitive abilities. Later, we discuss certain cognitive or mental processes. The properties of these processes and the factors influencing them have guided the design of both our systems (user identification and Captcha) from the beginning.

### 2.4.1 Individual Difference in Cognitive Abilities

Francis Galton was the first to start the investigation of individual differences among people based on their cognitive abilities [42]. He studied a variety of cognitive abilities and in each case focused on measuring the ability and noting the variations among different individuals. For example, he studied mental imagery [43] of a group of participants in both controlled and non-controlled conditions. Respondents were asked to think of some definite scenario, such as their breakfast-table that morning. The respondents were then asked whether the image was dim or clear and whether all the objects in the image were well-defined and other related questions. Galton found much variability in the capacity of mental imagery. Some participants reported almost no imagery, while others experienced clear images.

Individual difference is meant to capture the intuition that different people can approach

the same cognitive task in different ways. Recent research on individual differences is focused on two distinct categories [41] (a) individual differences in abilities (the capacity to *execute* a cognitive task) (b) individual differences in style (the *manner* in which one approaches a cognitive task). The main motivation behind our work emerges from the study of individual differences in *executing* cognitive tasks.

### 2.4.2 Visual Search

Visual search is a type of perceptual task in which the operator searches through a visual field for a target among other non-targets or distractors [105]. It is generally measured by the search time. The search time depends on multiple factors [96]. These include the strategies employed by a user in searching through the alternatives, the rate at which the user reads or scans the alternatives and the time spent to select the correct alternative. We will refer to the visual field consisting of the targets and distractors as the search set.

### 2.4.3 Self-terminating vs Exhaustive Search

There are two types of search strategies, exhaustive search and self-terminating search. In a self-terminating search, the user stops the searching process as soon as he finds the alternative he thinks is appropriate. This means that the search set must consist of the target item. The presence of the target makes the search set a positive search set [96]. Once the target item is found the user is not required to search anymore. For example, consider a search set containing a permutation of English letters arranged in a column (structured visual field). If the user is asked to find a specific letter, "L", he will inspect each letter in turn until he finds "L", at which point his search terminates. The user thus performs a self-terminating search on a positive search set.

On the contrary, in an exhaustive search, the search always extends throughout the entire available search set [96, 91]. Consider, the above example where the user is asked to find the number "9". Assuming that the user has no prior knowledge on the content of the list

presented, he inspects each item in turn until he reaches the end of the list, at which point the search terminates. Search sets which do not contain the target item are referred to as negative search sets [96]. A user always performs an exhaustive search on a negative search set.

### 2.4.4 Serial Visual Search

In a serial visual search 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). 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 [73, 64]. In other words, the visual search time is observed to increase with the number of items or distractors that need to be inspected before reaching the target.

If the visual search field is organized in a structured manner, people tend to search from top to bottom and from left to right [105]. However, if the visual search field is not coherently arranged, then search can be more random in structure [106, 89]. Visual search field, therefore, play an important role in predicting search times in time-critical environment.

### 2.4.5 Factors Affecting the Visual Search Process

A serial visual search process can be influenced by several factors. Before discussing these factors in details, we will introduce the following terms: top-down theory and bottom-up theory from [91]. "Theories starting with processing of low-level features are termed as bottom-up theories. Whereas, theories that are driven by high-level cognitive process, existing knowledge or prior expectation are termed as top-down theories."[91]. For example, a top-down influence on the visual process is a search for a word in a *sorted* list. The user's

prior knowledge on the arrangement of the words in the dictionary allows him to start his search near or around the spelling of the target word. On the other hand, consider a scenario where the target word is highlighted (using bold fonts). Such modification in the low-level feature of the target word makes it easily distinguishable from other words. This is a bottom-up influence on the visual search process. We now discuss the factors that influences serial search in details. These factors are needed to be considered while designing any matching game involving visual search (Section 3.1.1 and Section 4.1.1).

**Conspicuity.**

Recall that a search set consists of targets and non-targets (distractors). It is important that targets are similar to non-targets in the visual field. In other words, if the target is conspicuous, non-targets do not need to be inspected [112]. This affects the serial search process. The conspicuity of the target invokes a bottom-up influence on the visual process. The search for such targets are parallel, since all items are examined at once and the target is easily recognized. In parallel search, the size of the search set does not influence the corresponding visual search time. Therefore, the search set must contain targets and non-targets of similar color (preferably a single color), size, brightness to invoke serial search.

**Automaticity.**

If the user is highly familiar with the target a top-down influence on the search process might occur. For example the target is the user's own name. Prior exposure to the target or familiarity with the target, therefore, biases the serial visual search process.

**Expectancies.**

Another factor that prevents a strict serial search model has to do with the user expectation of where the target might lie inside the search set. For example, the knowledge of a sorted list allows the user to start the search near the spelling of the target word. In order, to avoid such top-down influence, the target needs to be placed randomly inside the search set.

### 2.4.6 Guided Search Theory

Cave and Wolfe in 1990 proposed the guided search theory [23]. The guided search model suggests that all searches involve two consecutive stages [23, 109]. The first is the parallel stage where the user simultaneously activates a mental representation of all the potential targets. In a subsequent serial search stage, the user performs inspection on each of the activated items, according to the degree of activation. The user then chooses the true target from the activated elements. For example, consider finding the odd pieces out of the image shown in Figure 2.4. The parallel stage will activate all the triangles but will not activate the squares. In other words, user's attention will be shifted towards the triangles. This activated set of triangles is now the search set. At the second stage, consider finding a single target (e.g. an equilateral triangle). A serial search is now in process to find the equilateral triangle in a search set of size 8 (triangles). In a guided search theory the parallel stage guides the serial stage because the former chooses the items to be processed. That is the user's attention will be directed to the items with highest activation or priority.

The mental representation of all the potential targets is also known as the activation map. An activation map is a representation of visual field in which the level of activation at a location reflects the likelihood that the location contains a target. This likelihood can be determined by bottom-up influences or top-down influences or both. For example, the low-level feature (structure) difference between a triangle and a square might activate the triangles. Similarly, prior exposures to shapes of squares and triangles might aid in activating the triangles. Therefore, the guided search model can direct the user's attention to certain items in the visual field by top-down and bottom-up influences [91].

### 2.4.7 Working Memory

There are two different types of memory storage [105]. One is the long-term memory and the other is the working memory. Long term memory involves the storage of information

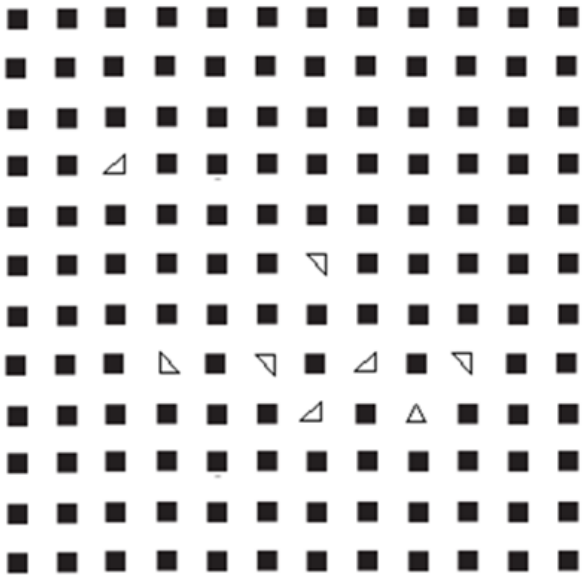

Figure 2.4: Guided Search: Find the equilateral triangle.

after it is no longer active in working memory and then the subsequent extraction of the information later. Working memory holds and manipulates information in mind over short periods of time for a cognitive task such as learning or reasoning [5].

There is a personal limit to working memory, with each individual having a relatively fixed capacity in terms of the *amount* of information that can be kept active and the *duration* it can be kept active. This makes the working memory capacity vary between individuals [39]. The ability to reason and solve problems requires the use of information stored in working memory. However, this information is vulnerable to interruption and decay. Due to this volatility, faster processing and manipulation of this information is necessary for successful completion of a task.

Baddley [5, 4] originally proposed a model of working memory consisting of four elements. The first is a visuospatial (visual and spatial-sequential) sketchpad which briefly holds some visual images. These images consists of encoded information coming from the visual sensory register. Thus a police officer will use the visual-spatial sketchpad to retain information regarding the license plate of a vehicle. Or this component can be used to hold information

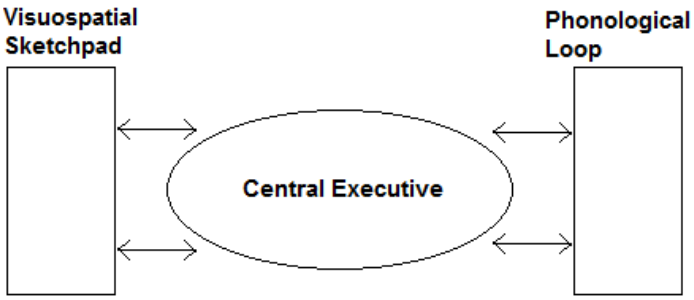


Figure 2.5: Baddeley's model of working memory with two storage subsystem serving the central executive co-ordinating system.

regarding the sequential occurrences of events. The second is a phonological loop, which holds speech or sounds for verbal comprehension and for acoustic rehearsal. The loop is kept active or rehearsed by articulating words or sounds, either vocally or subvocally. For example, when we are trying to remember a phone number we vocally or subvocally sound out the number. The third is the central executive components that co-ordinates information from the aforementioned two elements. Figure 2.5 provides a pictorial depiction of the working memory model of Baddley [5, 4]. The central executive decides what information to process further and how to process it.

Working memory can be measured through a number of different tasks. The type of task determines the type of performance metric or score to be used as a measure of working memory. Therefore, the metrics can range from simple binary "yes/no" to complex ones.

### 2.4.8 Priming Effect on Automatic Processing

Automaticity of actions are processes or behaviors that occur efficiently without the need for conscious guidance or monitoring [91]. There are two main categories of automaticity [104], (1) Conscious automaticity, where a conscious act of will is required to start automatic processing. For example, once the conscious decision is made to drive to home, the drive itself is automatic. We do not remember much of the experience of shifting gears and steering when

we arrive at our destination. (2) Unconscious automaticity, where the automatic process is triggered unconsciously by some external stimulus. It is the processing of information that guides behavior, but without conscious awareness, and without interfering with other conscious activity that may be going on at the same time [9]. Therefore, to study or quantify such processes, a technique called *priming* can be used.

Unconscious automatic processes can be invoked in many cases by a technique called *priming*. A *prime* is a stimulus or event that influences further processing of the same or related stimulus or action. In priming, participants are presented with a first stimulus (the prime), followed by a break [91]. After the break, which can range from milliseconds to days, the participants are presented with a second stimulus/action (same or related) to see whether the first stimulus affected the perception of the second. For example, Bargh et al. [10] carried out an experiment where one group of participants were exposed to words related to the concept of elderly, the participants who were primed with the elderly concept was found walking relatively slower than the others while leaving the experiment. However, when asked later, the participants responded with having no conscious awareness of the concept of the elderly or of their reaction to it.

Priming stimulus can be presented in two broad ways [91], one where the user is aware of the priming stimulus and the other where the stimulus is presented in a noisy background such that it is not registered by conscious awareness. We use both kinds of priming in our game design for user authentication (Section 3.1.2).

## 2.5 Related Work

We now provide a discussion on the related works in (1) behavioral biometrics, (2) image+game Captchas and (3) cognitive systems. In each Sub-Sections, we discuss in details the works that are closely related to ours.

### 2.5.1 Related Works in Behavioral Biometrics

There is a considerable amount of work on behavioral-based biometric authentication systems (such as mouse dynamics [2, 118, 44, 15, 81, 71, 79], keyboard typing pattern[86, 40, 98], and several others [111]). The underlying principal of these systems is centered on "what an individual does". Our work focuses on individual differences in cognitive abilities or essentially "what an individual *is capable of doing cognitively*" and could potentially fall under a new category of biometric. In this section, we discuss prior works done on behavioral biometrics, in particular, mouse dynamic biometrics.

**Mouse Dynamics.**

Mouse dynamics biometric systems often use mouse movement characteristics as the basis of authentication [2, 118, 71, 79]. They typically use speed, acceleration, angles, clicks statistics and other mouse movement features. These systems provide continuous authentication mechanism and allow free mouse movements. In [44, 15, 81] mouse movements are captured from a controlled environment (games and similar interfaces). However, none of these systems attempt to capture cognitive factors.

Gamboa et al. [44] proposed a static authentication system using a web-based game. They collected 10 hours of mouse interaction data from 50 users and reached an EER of 6.3% (with 20 strokes). Each stroke consists of a set of points between two mouse clicks. The feature set consisted of spatial and temporal features related to the mouse dynamics. For example, spatial features consisted of distances, (x, y) co-ordinates and others, whereas the temporal features consisted of velocity and acceleration.

Ahmed et al. [2] used mouse dynamics in a continuous authentication system and reported an EER of 2.47%. They passively collected mouse movement data while the users performed regular task on a dedicated station. Then they extracted features related to mouse dynamics such as movement speed, direction, traveled distance and others. They carried out their experiments with 22 participants.

One of the drawbacks of using mouse dynamics features is that they are influenced by environment variables. Jorgensen et al. [58] carried out a study in which they showed that error rates for schemes, in particular, Ahmed et al. [2] and Gamboa et al. [44], increased under controlled environments. On the other hand, Zheng et al. [118] showed that some mouse dynamics exploiting directions, ratios and angles are relatively platform independent and results in low EER of 1.3% [118].

**Hamdy and Traore [50] on Mouse Dynamics + Cognitive Factors.**

The work closest to ours, although it is a combination of mouse dynamics and cognitive factors, is that of Hamdy and Traore [50]. The authors claim to combine visual search and short-term memory features with mouse dynamics. Their system requires the user to search for letters on a shuffled *virtual* keyboard (appearing on a computer screen). At the beginning, the user is presented with a shuffled *virtual* keyboard and a target string of letters, Figure 2.6. The user looks at the first letter of the string and searches for that letter on the virtual keyboard. The user clicks on it as soon as he finds the letter. According to the author this task is equivalent to a visual search task. The user then continues searching the next letter. It is debatable whether the exposure of the same shuffled virtual keyboard and the string of letters to be searched throughout a session has correctly measured the visual search time. Visual search time should be measured by a subtraction method [33]. The subtraction method is an established technique in cognitive systems that involves subtracting the amount of time information processing takes with the process from the time it takes without the process. This implies that the user's movement time while clicking the letters does not represent a pure visual search time. In fact, the movement time is a combination of visual search time and mouse movement time.

Moreover, their system is language dependent because they use English letters as the target string. Consider, there are two users: User A from England and User B from Japan. If the system generates a random string of English letters, User A will have an advantage


Figure 2.6: Sample shuffled *virtual* keyboard [50]

over User B in terms of familiarity with the letters. Now consider if the system replaces the target string of letters and the keys of the virtual keyboard with random shapes. There is a high chance that both the users are getting exposed to those random symbols for the first time. In such case, the estimated features of visual search or short term memory will remain uninfluenced.

Along with their cognitive features and other mouse dynamic features, they have reported an EER of 2.11% (for verification) and an accuracy of 72% (for identification). Some of their features related to mouse dynamics such as speed can be affected by platforms as evident from Jorgensen et al. [58]. However, no such discussion was provided. Their system performance in remote authentication scenario is also not discussed.

**Authentication via Implicit Learning.**

Other works such as [12, 32] use the concept of implicit learning, whereby the user is trained on a secret, which can later be used during authentication but cannot be described explicitly by the user. Implicit learning such as learning to swim or learning to ride a bicycle gathers knowledge which is not consciously accessible to the trained person. The authors [12] use implicit learning to plant a password in the human brain that can later be detected during authentication. The system essentially reduces the cognitive burden of remembering

a password and does not require any special hardware. However, it has a long enrollment and verification time. Our system does not invoke any implicit learning, since the challenge sequences are random, and therefore, is different from these approaches [12, 32]. There is no learning effect on our system.

### 2.5.2 Related Work in Image and Game Captcha

There is a considerable amount of work on Captchas utilizing the scope of text, image, audio, video or game, in systematically concealing some information, preferably only to be revealed by a human user. The most common among these types today, is the CR (Character Recognition) Captcha, where the user is required to input alphanumeric character in response to a challenged distorted text. However, many of the proposed or deployed CR Captchas have been broken with high success rate [115, 70]. Image-based Captchas are alternative approach to CR promising better usability and security ratings [119, 97]. They require the user to recognize or distinguish [119, 29] or establish semantic relationship among images [97] or between images and words [36]. A security pre-requisite of all these Captchas is that the response to the challenge must not be provided to the client machine. However, keeping them in *clear form* allow both human and machine to respond to the challenge with ease. Nevertheless, a machine should always fail to authenticate as a human.

In such cases, differentiation should heavily depend on analysis of cognitive and behavioral features. 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" [3]. Although lacking any scientific evidence or literature, the company claims to differentiate human and machine based on behavioral data such as mouse events [3, 69]. Their system challenges the user with a game, which involves objects floating around within a frame. The task of the human user is to establish semantic relationship between the floating object and the corresponding target, by a drag and drop action. Dynamic Captchas are generally resistant to static relay attacks [69].

*Clear form* Captchas are supposed to be inherently language and culture independent, because the response to a challenge is basically the challenge itself. Semage [97] claims to surpass the boundaries of languages but fails to be culture independent. In Semage, user is required to find semantic similarity between images e.g. a real image of a tiger and a cartoon tiger. Though Semage has a very high fun-factor, it suffers from the inadequacy of automatically generating the image database. Moreover, due to traditional and cultural variations throughout the world, challenge images might look foreign to some people. It might be easier for someone living in a colder region to figure out the relationship between ice hockey gloves and skates, and at the same time confusing for someone living near the equator. Image Recognition Captcha such as Cortcha [119] can automatically generate the image database. However, it also requires prior knowledge on the relationship between the decoy object and the inpainted image.

### 2.5.3 Related Work in Cognitive Systems

In this section we discuss the related work done on the aforementioned mental processes. The first four Sub-Sections discuss works which were closely followed while designing our systems. The last Sub-Section discusses works that provide evidence for individual difference in cognitive processes.

**Serial Search.**

Neisser [73] carried out an experiment where users were instructed to find the *absence* of letter $Z$ in a list. The list contained 49, 6-character strings such as J$Z$TXVB, DQFJH$Z$, $Z$XLSMT and one target 6-character string VXRLFH arranged in a column. 15 such lists (with random target item position and random strings of letters) were given to each user. It was observed that the visual search time has a positive linear relationship with the position of the target item inside the list of 50 items. Such type of *trend* in structured self-terminating visual search is also evident from the work of [64]. We utilize this relationship between visual

search time and search set size for authenticating user as human in Movtcha.

**Guided Search.**

Our design of Movtcha also utilizes the guided search theory as discussed in Section 2.4.6. Wolfe and Cave [109, 23] presented a model for the human search behavior, known as the Guided Search Model. Wolfe [109] measured the reaction time of the user as a function of the search set size. The search set consisted of a vertical line (target) among other horizontal lines (distractors). The participant has to respond with "yes" (target present) or "no" (target absent). The response time suggested that the reaction time is independent of the search set size. Therefore, all items inside the search set were processed for orientation in parallel. Whereas, when the search set contained items of "T" and "L" with different orientation, the search time is found to be increasing linearly with the set size. That means in a guided search the first stage is the parallel search stage and the second stage is the serial search stage. This trend is similar to other self-terminating serial search [73, 64].

**Working Memory.**

Although non-verbal working memory is often referred to by the more general term "visual-spatial" memory, Sala et al. [31] suggested that the non-verbal working memory might be comprised of distinct visual and spatio-sequential components. They used a Visual Pattern Test (VPT) to measure *pure* short-term visual working memory (largely shorn of its spatial component). In the VPT test users are presented with grids similar to Figure 2.7. Each grid or matrix is presented to the user for three seconds and then removed. The user is then asked to reproduce the pattern by marking squares in an empty grid of the same size. A performance score is the number of filled squares that has been correctly recalled. The complexity of the task is varied by varying the grid size and the number of filled cells.

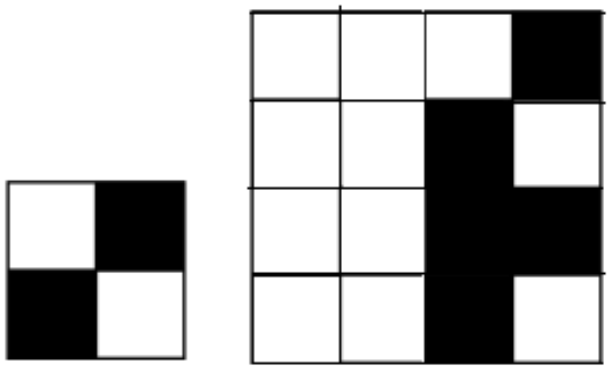

Figure 2.7: Examples of grids for VPT. These grids are drawn by us as examples and are *not* necessarily the ones used in [31]

**Automatic Processing and Priming Effect.**

As discussed in Section 2.4.8, unconscious automatic processing are studied using priming stimulus. The effect of the priming stimulus can be estimated by noting the difference in behavior caused before and after the priming stimulus. Bargh et al. [10] carried out an experiment where one group of participants were exposed to words related to the concept of elderly, and the other group was exposed to neutral words. For the elderly prime version, the critical stimuli were worried, Florida, old, lonely, grey, selfishly, careful, sentimental and others. In the neutral version, the elderly prime words, were replaced with words shorn of elderly concepts such as thirsty, clean, private and others. A confederate of the experimenter then surreptitiously recorded the time needed for the participants to walk down a particular length of the corridor while leaving the experiment. It was observed that the participants who were primed with the elderly concept were walking slower than the participants from the neutral group. However, when asked later, the participants responded with having no conscious awareness of the concept of the elderly or of their reaction to it.

**Individual Difference in Cognitive Systems.**

Horn [53] points out a set of cognitive abilities that might help in capturing individuals differences. Individual differences in visual search task is also evident from the pioneering work of Chiang and Atkinson [25]. They carried out a visual search experiment and on analyzing the reaction time with the search set size found significant differences among the individuals. Recent researches done on individual difference in serial search task includes [100, 95].

Individual differences in the processing speed in a choice reaction experiment is evident from the work of Jensen [55]. An analysis between the reaction time with the logarithm of the number of alternatives showed variances among the participants in a choice-reaction experiment. Other works focusing on individual differences in working memory and information processing includes literatures [54, 39]. Individual differences in automatic processing due to priming are also evident from recent works [34, 37]. They have indicated that priming may be sensitive enough to serve as a measure of individual differences in automatic processing.

## 2.6 Statistical and Image Processing Tools

In this section, we first discuss the statistical learning techniques that have been used in our user identification system. Then we provide a discussion on the image processing tools that we used in Movtcha.

### 2.6.1 Probability Density Function Estimator

A random variable, r.v., quantifies the outcome of a random process like tossing a coin, throwing a dice. For example, the random variable $X$ can take in 1 when a coin lands head and take in 0 when the coin lands tail. There are two types of random variable, discrete and continuous. Discrete variables takes in a countable list of outcomes whereas the continuous variables can take in any values between an interval.

The probability distribution of a continuous-valued random variable $X$ is conventionally

described in terms of its probability density function (pdf), $f(x)$, where the probability associated with $X$ is given by the expression (2.1). Note that for continuous-valued variables the probability at any exact point is equal to zero. Therefore, probability is found for an interval, which is a particular area under the function $f(x)$,

$$P(a \leq X \leq b) = \int_a^b f(x)dx. \tag{2.1}$$

Sometimes it is necessary to estimate the underlying population distribution, $f(x)$ from a sample of observations. We will assume that the observations are independent realizations of $X$. In other words, the observations $x_1, .., x_n$ is the set of randomly drawn values of $X$. There are two way of estimating $f(x)$, (1) parametric approach and (2) non-parametric approach.

**Parametric Approach.**

The parametric approach of density estimation is to assume that $f(x)$ belongs to a parametric family of distributions such as the normal or gamma family. For example, let us assume that $f(x)$ is a member of a normal distribution $N(\mu, \sigma^2)$. This leads to the following estimator, $\hat{f}(x)$,

$$\hat{f}(x) = \frac{1}{\hat{\sigma}\sqrt{2\pi}} e^{-(x-\hat{\mu})^2/2\hat{\sigma}^2} \tag{2.2}$$

where the parameters $\hat{\mu} = \frac{1}{n}\sum_{i=1}^n x_i$ and $\hat{\sigma} = \frac{1}{n-1}\sum_{i=1}^n (x_i - \hat{\mu})^2$ are estimated from the sample data. If the distributional assumption of the underlying population is correct to some extent, this approach can yield suitable estimates.

However, such parametric approach imposes restrictions on the shapes of $f(x)$. In the above example $\hat{f}(x)$ is symmetrical and bell-shaped and therefore is not suitable for representing skewed densities or bimodal densities.

**Non-parametric Approach.**

A non-parametric approach to density estimation can be used to avoid making any assumption on the distribution of the underlying population and to better understand the structure of the data. The easiest non-parametric estimation of a probability distribution is the use of histogram. It is simple but has disadvantage such as discontinuity, high sensitivity to bin edges, and starting points [103].

A histogram is constructed selecting a left bound or starting point $x_0$ and the bin width, $b$. The "bin width" is sometimes referred to as the "binwidth" (one word). The bins are usually formed by dividing the real line into equally sized intervals. The histogram is then a step function, and the height is the proportion of samples contained in that bin divided by the bin width. The histogram estimator $\hat{f}_H(x)$ at a point $x$ for a sample size of $n$ can then be given by expression (2.3),

$$\hat{f}_H(x;b) = \frac{number\ of\ observations\ in\ a\ bin\ containing\ x}{nb}. \tag{2.3}$$

The choice of binwidth and the positioning of the bin edges has a substantial effect on the shape and properties of $\hat{f}_H(x;b)$. A smaller binwidth leads to a relatively rough or jagged histogram, while a larger binwidth results in a smoother looking histogram. Even when the binwidth is kept fixed, the choice of the bin edges might lead to different histogram shapes.

On the other hand, kernel density estimators are superior to histogram and are quite intuitive [87, 103]. The sensitivity of the histogram to the placement of bins is a problem not shared by kernel density estimators. Kernel density estimators smooth out the contribution of each data point $x_i$ over a local vicinity of that point by replacing the point with a kernel function. In case of a Gaussian kernel the probability is highest at point $x_i$ and then the probability slowly attenuates symmetrically on both sides. The contribution of the point $x_i$ to the estimate at the point x depends on the distance between $x_i$ and $x$. The extent of contribution depends on the shape of the kernel function and the bandwidth selected for it.

For kernel estimators, the bandwidth, $h$, of the kernel is analogous to the binwidth, $b$, of the histogram.

The probability can be estimated by a relative frequency at point $x$ by expression (2.4),

$$\hat{f}(x) = \frac{number\ of\ observations\ in\ (x-h, x+h)}{2nh}. \tag{2.4}$$

Expression (2.5) provides an alternative way to represent $\hat{f}(x)$,

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} w(x - x_i, h) \tag{2.5}$$

where $x_1,\ x_2,\ \ldots, x_n$ are the observed values and weighting function $w$ is defined as follows:

$$w(t,h) = \begin{cases} \frac{1}{2h}\ for\ |t|<h, \\ 0\ otherwise. \end{cases}$$

The way to think about the above weighing function is to imagine a rectangle of height $1/2h$ and width $2h$ at each of the observed points. Therefore the estimate of the pdf at a given point is $1/n$ times the sum of the heights of all the rectangles that cover the point. If $h$ is increased the resulting function is less jagged or smoother and vice versa. However, using rectangular kernel as above, still might result in discontinuity similar to histogram. To obtain a smoother function, one can sought to smoother kernel functions like Gaussian. We therefore use a Gaussian weighting function,

$$w(t,h) = \frac{1}{h\sqrt{2\pi}} e^{-(t)^2/2h^2} \quad -\infty<t<\infty. \tag{2.6}$$

The above weighting functions are all of the form of expression (2.7), where K is the kernel function,

$$w(t,h) = \frac{1}{h} K\left(\frac{t}{h}\right). \tag{2.7}$$

This allows us to rewrite expression (2.5) with the kernel as,

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{2.8}$$

The kernel estimate is constructed by centering the kernel (e.g. Gaussian, rectangular, Epanechnikov) at each observation. Therefore, the value of the kernel estimated at a point $x$ is simply the average of the $n$ kernel ordinates at that point. The bandwidth, $h$, of the chosen kernel function determines the smoothness of the resulting density function.

### 2.6.2 Bandwidth Selection

At this point, the estimation of $\hat{f}(x)$ narrows down to choosing (1) the kernel function and (2) the bandwidth $h$. Given a bandwidth h and the density estimate $\hat{f}(x)$, we are interested to know how closely it represents the true population density $f(x)$. Therefore, we consider the difference in area between this two functions. In other words, we take the integrated squared error (ISE),

$$ISE(\hat{f}(\cdot; h)) = \int \left(\hat{f}(x) - f(x)^2\right) dx = \int \hat{f}(x)^2 dx - 2\int \hat{f}(x) f(x) dx + \int f(x)^2 dx. \tag{2.9}$$

If we have more than one possible datasets from the true density $f(x)$, we can take into account the expected value of the random quantity, using mean integrated squared error (MISE),

$$MISE(\hat{f}(\cdot; h)) = E[ISE(\hat{f}(\cdot; h))] = E\int \left(\hat{f}(x) - f(x)^2\right) dx. \tag{2.10}$$

The goal here is to choose a value of h such that $ISE(\hat{f}(\cdot; h))$, expression (2.9), is minimized. Notice that the first term of expression (2.9), $\int \hat{f}(x)^2 dx$, can be calculated from the data. The third term, $\int f(x)^2 dx$, can be ignored since it does not depend on the bandwidth $h$. The second term in the expression (2.9), $2\int \hat{f}(x) f(x) dx$, depends on $f(x)$ which is the true density of the underlying population and is unknown. In order to solve this issue, we

resort to a leave-one-out LSCV. A LOO LSCV, leave-one-out least square cross-validation, splits the observations into two parts. The $m-1$ first observations are used for density estimation and the remaining one is used for assessing the accuracy of this estimation. This is repeated $m$ times, each time leaving out a new observation. The mean of the resulting $m$ quantities is then obtained. The process is repeated until a bandwidth, $h$, is found which minimizes the mean error (Expression 2.10).

### 2.6.3 Edge Detection Algorithm

Edges detection is the mechanism of detecting points in a digital image where the brightness changes abruptly [88, 65]. Edge detection reduces the amount of data in an image while preserving the skeleton of the image. Canny edge detector [21] is a popular edge detection algorithm. The algorithm has four steps, (1) Smoothing, where the image is blurred to remove noise that might result into false edges, (2) Finding gradient, where a gradient operator is applied for obtaining the gradients' intensities and directions, (3) Non-Maximum suppression, where only local maxima are marked as edges, (4) Hysteresis thresholding, where the potential edges are determined.

**Smoothing.**

To prevent noises from being mistakenly identified as edges, noises must be reduced. Therefore the image is first smoothed by applying a Gaussian filter. This results into a slightly blurred version of the original image.

**Finding Gradient.**

Gradients at each pixel in the smoothed image are determined by applying a Sobel-operator. The operator uses two $3 \times 3$ kernels which are convolved with the original image to approximate the gradient $G_x$ and $G_y$ in the x and y directions respectively. The gradient magnitudes

can then be determined as an Euclidean distance measure as shown in Expression 2.11,

$$|G| \;=\; \sqrt{G_x^2 + G_y^2}. \tag{2.11}$$

The gradient direction is found using Expression 2.12,

$$\theta \;=\; arctan\left(\frac{|G_y|}{|G_x|}\right). \tag{2.12}$$

**Non-Maximum suppression.**

The purpose of this stage is to convert the "blurred" edges in the image of the gradient magnitude to "sharp" edges. This is done by determining if the gradient magnitude is a local maximum in the gradient direction. The gradient is quantized to one of the values $0°$, $45°$, $90°$, $135°$. For example, consider that the gradient at position $(x, y)$ was $\theta$. Now $\theta'$ is obtained after quantizing $\theta$ to one of the aforementioned angles. Considering the two pixels, $p_1$ and $p_2$ in direction $\theta'$ and $\theta' + 180$ from $(x, y)$, if the edge magnitude at either $p_1$ or $p_2$ is greater than the magnitude of $(x, y)$ then $(x, y)$ is marked. Once this is done on the whole image, the marked pixels are deleted.

**Hysteresis Thresholding.**

The resulting image from the non-maximum suppression phase is thresholded to obtain a binary edge image. Hysteresis thresholding uses two threshold values. One is the upper value $t_U$ and the other is the lower value $t_L$. Any pixel with a value greater than the upper threshold $t_U$ is assumed to be an edge pixel. In addition, any pixel with a value $p$ and adjacent to an edge pixel is also considered an edge pixel provided $t_L \leq p \leq t_U$.

# Chapter 3

# USER IDENTIFICATION USING HUMAN COGNITIVE ABILITIES

We present a *cognitive task* (CT) to the user in the form of a game that will be performed by the user by interacting with the computer, and using a mouse or a touchpad. The game starts by presenting a set of 25 different objects arranged in a 5×5 grid to the user. The task of the user is to find a particular target object in the set. The user has to drag and drop the challenge object onto the matching target object in the set. This is equivalent to performing a visual search task by the user. On performing a correct search task (or correct match), the user is rewarded with a gold coin. The user is instructed to deposit the gold coin in a "bank". On a correct deposit, the user is presented with another challenge object and a similar interaction follows. The collected data during the execution of the CT will be used to extract cognitive features related to visual search ability, working memory and the effect of priming on automatic processing of the user. Features derived from these cognitive processes in combination with other basic stimulus-response features are used to build profiles for the users that can later identify them. We use a statistical approach to identifying the users.

Our system can be divided into three major modules. They are as follows: (1) The Interaction Module consists of the input device that is used in executing the cognitive task. We use a mouse as the only input device to our system. However, there is no hard-and-fast rule on the choice of the input device, as long as the device used is able to capture cognitive features. (2) The Data Acquisition Module provides a cognitive task in the form of a web-based game. We developed the game using the Paper.js framework [76], an open source vector graphics scripting framework that runs on top of the HTML5 Canvas. The user interaction data are recorded using JavaScript and submitted passively via AJAX requests to the web

server. (3) The User Identification Module preprocesses the captured data and compares and matches the captured data with existing templates. It then provides an authentication decision.

This chapter has four sections. The first section (Section 3.1) provides details on the design of the game and how the game invokes certain mental processes. Section 3.2 provides discussion on the security of the system. Experiments and results are provided in Section 3.3. Finally, in Section 3.4 we discuss the limitations of our system.

## 3.1 System Design

The cognitive task (CT) is central to our authentication system. We present the CT to the user in the form of a web-based game. In this section, we provide a detail discussion on the design of the game. Later, in subsection 3.1.2, we examine how the design of the game invokes the mental processes.

### 3.1.1 Design of the Cognitive Task

Our game provides a simple challenge-response task. In each *instance* of the challenge-response, the user is given a challenge, which is an object. The user responds by dragging the challenge object onto the matching object inside the search set. The user then receives a gold coin as a reward and deposits it in a bank. On a correct deposit, the user is challenged with a new object and the game continues as before. Our goal was to invoke the aforementioned mental processes within a minimal design space.

An image is first broken into a grid of $5 \times 5 = 25$ square cells. We refer to this partitioned image as the search set $\theta$, $|\theta| = 25$. Each square is called a *tile*. The game starts by presenting a random challenge tile $t_c$ at a location $P_{t_c}$ outside the partitioned image. The tile, $t_c$, is a copy of a tile $t_r \in \theta$. We have divided the user's response into two actions. (1) The user drags $t_c$ and drops it onto $t_r$ located at position $P_{t_r}$ within the search set, in which case $t_c$

rests on $t_r$ and becomes *unmovable*. We refer to this action as $A_{resp}$. On an incorrect match, $t_c$ automatically moves back to position $P_{t_c}$ signifying a *mismatch* and allowing the user to retry. (2) On a correct placement of $t_c$ on $t_r$, the user is rewarded with a gold coin, $g_c$, which appears exactly at $P_{t_r}$ (superimposed). The user is then instructed to deposit (*drag* and *drop*) $g_c$ in a bank (a bounded box with the same dimensions as that of a tile) appearing at $P_{t_c}$. We refer to this action as $A_{rew}$. On depositing the coin, the user is challenged with the next tile, and the game continues as before. Therefore, one *instance* of this game is comprised of the correct placement of the target tile, $A_{resp}$ and correct deposit of the gold coin, $A_{rew}$ (Figure 3.1).

We provide a detailed discussion on the design of the game in the following paragraphs. The game is a simple object matching game. On a correct match, the user gets an incentive or gold and then deposits the gold in a bank.

**From Conceptual Modeling to Implementation.**

In order to guarantee the invocation of the mental processes, in particular visual search, working memory, and priming effect on unconscious automatic processing, certain constraints have been placed throughout the game. We refer to these constraints as $UAC1$, $UAC2$ ..., $UACn$. To differentiate these constraints from the ones used in Movtcha in Chapter 4, we add the prefix "UA", which stands for "User Authentication". Refer to Appendix A Table A.1 for a quick look-up of constraints and their usage.

*UAC1:* At the beginning of each *instance*, a copy of a randomly chosen original tile $t_r \in \theta$ appears at $P_{t_c}$ as the challenge tile $t_c$. Challenge tiles cannot be repeated. Therefore, each of the 25 partitioned portions (original tiles) of the image must appear only once as the challenge tile. This constraint ensures the following: (1) The presented search set is always a positive search set and invokes self-terminating search. Refer to Section 2.4.3 for a discussion. (2) The user is not exposed to the same challenge tile, $t_c$, more than once. If $t_c$ is repeated more than once, the user might recall the target tile's location, $P_{t_r}$. The

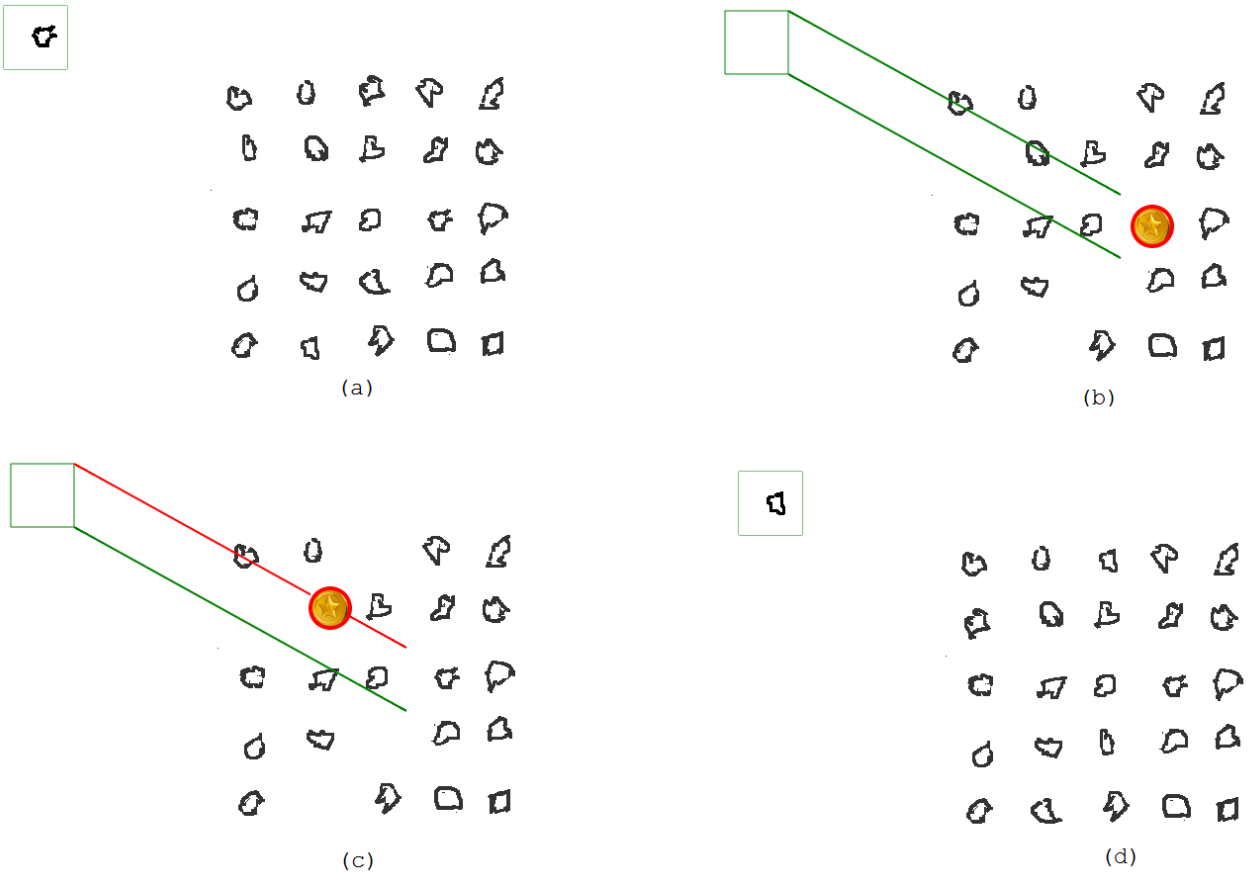


Figure 3.1: (a) User is presented with a challenge tile, $t_c$, at the beginning of the $21^{st}$ *instance*. (b) User performs $A_{resp}$, i.e. drags and drops $t_c$ onto $t_r$ inside the grid. On a correct match the *loose* tiles disappear showing the current game status (at $21^{st}$ *instance*). (c) User performs $A_{rew}$, i.e. drags and drops gold coin, $g_c$, onto $P_{t_c}$. Top guiding line color changes from green to red as the coin touches the line. (d) User successfully deposits $g_c$ and gets the next challenge tile. All 25 tiles are visible at this point. Notice that the 21 *unmovable* tiles in b and d have not changed their positions. All the *loose* tiles have changed their position (compare a and d). The target tile appears at its original position in the image in (d).

recall process will then contaminate the visual search process. In other words, a top-down influence is likely to occur on the search process in such cases. Refer to Section 2.4.5 for factors affecting the search process.

*UAC2:* On completing the action $A_{resp}$, $t_c$ becomes *unmovable*. At this point all the *loose* tiles (tiles that have not appeared in the challenge phase till now) disappear from the grid leaving only the *unmovable* ones. This provides the user a chance of knowing the current status of the game. The user can observe the tiles that have been placed correctly since now and the remaining empty square cells on the grid. Refer to Figure 3.1 and 3.3. This is similar to the Visual Pattern Test (VPT) of Sala et al. [31] where the grid consisted of some empty cells and filled cells.

*UAC3:* At the beginning of $A_{resp}$ all *loose* tiles in the grid are shuffled. They randomly change their positions on the grid except for the target tile $t_r$ and the *unmovable* ones. All 25 tiles are visible during $A_{resp}$. Therefore the actual positions of the *loose* tiles remain unknown to the user until they appear as the challenge tile. Refer to Figure 3.1. This constraint ensures that users are not exposed to the target location $P_{t_r}$ prior to the challenge. The visual process will be affected by a top-down influence, if the user already knew the target location, $P_{t_r}$.

*UAC4:* Two straight lines from $P_{t_r}$ to $P_{t_c}$ appears during the $A_{rew}$ action. During this action if the gold coin touches any one of the straight lines, their color changes from green to red, without hampering the current movement (Figure 3.1). The color change acts as an *obtrusive* priming stimulus (Section 2.4.8) for the user. Refer to Section 3.1.2 for a discussion on how this constraint invokes unconscious automatic processing.

*UAC5:* The tiles consist of random-shaped black symbols on a white background. All tiles have the same color intensity and brightness and appears degraded (opacity 0.8 on a scale of 0-1) throughout the game. The symbols being of random shapes do not *necessarily* represent or convey any meaning to the user. The symbols are hand-drawn using a paint

Figure 3.2: (a) Random symbols from our system

program [75]. When we drew the shapes we attempted not to make any symbol more conspicuous than the others. Our shapes do not have any "criss-cross" lines and are drawn mostly with a continuous stroke (Figure 3.2).

*UAC6:* We allow some tolerance on the placement of the tile and the gold. This means that the user does not need pin-point accuracy when dropping $t_c$ at $P_{t_r}$ or when depositing $g_c$ at $P_{t_c}$. A *drop* is considered a match if $t_c$ or $g_c$ cover 60% area of the underlying tile $t_r$ or bank respectively. And on releasing they automatically get superimposed over their destinations. However, there are exceptions, on $K$ random *instances* the tolerance is decreased significantly during $A_{rew}$ *only*, requiring 90% overlapping area. $K = k_1 \ldots k_5$ are chosen randomly but consecutively from *instance* intervals $\{i_1 \ldots i_5\}$, $\{i_6 \ldots i_{10}\}$, $\ldots$, $\{i_{21} \ldots i_{25}\}$. The decrease of tolerance acts as an *unobtrusive* priming stimulus in a noisy background (Section 2.4.8). We have provided more discussion on constraint $UAC6$ in Section 3.1.2.

*UAC7:* Each time there is a *mismatch* or the *drop* does not meet the tolerance threshold of $UAC6$, $t_c$ or $g_c$ automatically moves back to $P_{t_c}$ or $P_{t_r}$ respectively. The user is then allowed to re-try. However, as soon as the user hovers over $t_c$, the grid is again shuffled according to constraint $UAC3$.

### 3.1.2 The CT Constraints and Cognitive Processes

Here, we summarize the importance of the aforementioned constraints and how they relate to the mental processes.

**Revisiting Visual Search Ability.**

Our game (CT) has been designed to invoke only self-terminating searches. As soon as the user finds the target tile, further searches are not required. Due to constraint *UAC1*, we refer to our search set as a (+ve) search set, meaning that the target must appear inside the search set so that a match can occur. On the other hand, a (-ve) search set does not consist of the target object but only non-targets (distractors), thus invoking exhaustive search. Refer to Section 2.4.3 for a discussion on the difference between these two types of search. Constraint *UAC3* and *UAC5* invokes serial search and reduces the conspicuity of the target respectively. In other words, *UAC3* attempts to eliminate any top-down influences and *UAC5* attempts to eliminate any bottom-up influences on the visual search process. Recall, that if a target is too conspicuous they may easily be found, without the need for inspecting other non-targets [108, 96, 105]. If the grid was not shuffled at each instance according to *UAC3 and UAC7*, the user could have remembered certain target positions beforehand. This top-down influence could have contaminated the visual search processes. Section 2.4.5 discusses the factors affecting visual search process.

**Revisiting Working Memory and Information Processing Speed.**

Due to Constraint $UAC2$, the user can observe the current game status. The user can observe the empty grid cells and the already placed tiles. He can hold this information in his working memory for a short interval of time while he completes action $A_{rew}$. If the information is not lost, he will be able to decrease the size of the search set $|\theta|$ for the next challenge. For example, for the 11th *instance* of the game he will be able to shrink $|\theta|$ to 15, thus skipping over the already placed 10 tiles. Recall that according to Constraint $UAC3$, after the user

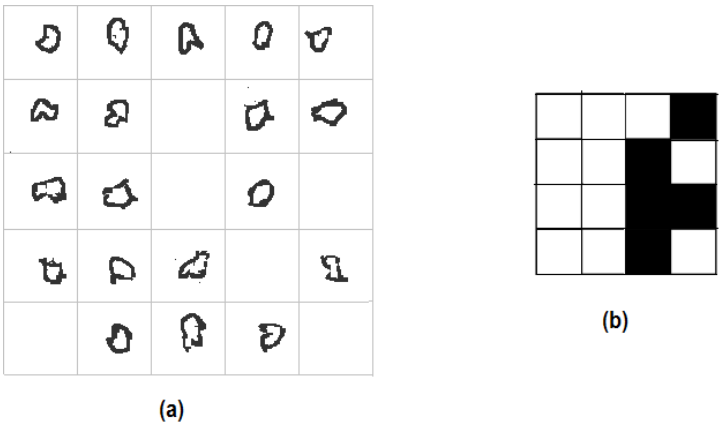


Figure 3.3: (a) An instance of our game, when $A_{resp}$ is completed. The current status of the game is visible. (b) Example grid from a Visual Pattern Test [31]. Notice the similarity between the presentation of (a) and (b).

completes $A_{rew}$, all 25 tiles are visible. Therefore, if the user fails to hold the information (status of the game) in his working memory, his $|\theta|$ must be lower bounded by 15. In such case, the time elapsed on placing the target tile correctly is relatively longer. Therefore, it is necessary to investigate the working memory capacity in terms of information processing speed for each individual user.

Our design approach for measuring working memory capacity is very similar to the Visual Pattern Test (VPT) (Figure 3.3) of Sala et al. [31] (Section 2.5.3). In the VPT, users were presented with a grid consisting of some empty cells and filled cells for a few seconds. The users were then asked to reproduce the pattern by marking squares in an empty grid of the same size. The users received scores based on the number of filled squares that have been correctly recalled. VPT measures the pure visual working memory and excludes the spatial-sequential component. Recall, that according to Baddley's original working memory model [5, 4], the visual-spatial component was considered to be a single component. We followed Sala et al. design of VPT closely, in order to estimate the pure visual working memory capacity from our game. For a discussion on Baddley's model [5, 4], refer to Section 2.4.7. Discussion on VPT of Sala et al. [31] can be found in Section 2.5.3.

The type of task determines the type of performance metric or score to be used as a

measure of working memory capacity. We measure the capacity of working memory in our game using information processing speed. Information processing speed is generally measured with the aid of the reaction time in a choice-reaction experiment [1, 52, 39]. At the $i^{th}$ instance of the game, if information about the status of the game is not lost from the working memory then the user is left with $25 - i + 1$ alternatives for the search operation. Therefore according to Hick-Hymen law [1, 52],

$$RT = a + \frac{1}{b} H_i, \tag{3.1}$$

*where,*

$RT$ = Total Reaction Time,

$H_i$ = Information content at the $i^{th}$ instance,

$a$ = Simple reaction time ($H_i = 0$),

$b$ = Information processing speed in bits/s.

The amount of information at the $i^{th}$ *instance* that is present in the search set $\theta$, $|\theta| = 25 - i + 1$, can be expressed as,

$$H_i = \sum_{k=1}^{|\theta|} P_k \left( log_2 \left[ \frac{1}{P_k} \right] \right) \tag{3.2}$$

where $P_k$ is the probability of the $k^{th}$ alternative at the $i^{th}$ instance in the search set of $25 - i + 1$ alternatives. Due to constraints *UAC3, UAC7* all these alternatives are equally probable. We provide experimental evidence on working memory capacity in Section 3.3.3.

**Revisiting Automatic Processing and Priming Effect.**

During $A_{rew}$, two straight lines are drawn from $P_{t_c}$ to $P_{t_r}$ to guide the movement of the gold coin (Figure 3.1). Constraint *UAC4* provides the priming effect necessary to invoke the automatic processing. We assume that the change of color provides an *obtrusive* priming stimulus. Subsequently, on receiving the prime, the user might have some effect on the

straightness (unconscious automatic processing) of his movement (conscious processing) for the following game *instances*. We measure the effectiveness of the prime, $EOP_{C4}$, as the ratio of the length of the line *not* overlapped by the gold coin to the length of the guiding line. We then measure the resulting straightness as the ratio of the distances moved to the actual distance of $P_{t_c}$ to $P_{t_r}$.

On the other hand, constraint $UAC6$ provides an *unobtrusive* priming stimulus in a *noisy* background, unlike $UAC4$. That means the user is not aware of the priming stimulus. The gold moves back to $P_{t_r}$ if it does not satisfy the tolerance threshold (Refer to Constraint $UAC6$). And the change in tolerance threshold of 30% (from 60% to 90%) is trivial enough to go unnoticed by the user. After receiving this prime the user will have some effect on the way the gold coin and tiles are dropped on the bank and grid respectively for the following *instances*. We measure the effectiveness of this priming as the ratio of two overlapping areas, $EOP_{C6}$. The primes are considered effective on a particular user if the $EOP$s are very close to 1. On the other hand, the effectiveness of the prime decreases as $EOP$s tend to 0. We provide experimental evidence of priming effect on automatic processing in Section 3.3.3.

### 3.1.3 Cognitive Features Estimation

**Raw Data.**

The interaction data during the task execution is collected for each user. The raw data consist of the pointing device's absolute position (x and y co-ordinates), event types (click, drag, release) and the time when these events occur. We explore these raw data in order to estimate features related to cognitive abilities. During an *instance* of the game the following raw data are collected. Raw data are only collected for an *instance* and not for an incorrect visual search task.

*OnClickEvent*: The $x_{ce}$ and $y_{ce}$ co-ordinates of the click event $e$ and the corresponding timestamp $t_{ce}$ given that $e$ has been triggered on either the tiles or the gold.

*OnReleaseEvent*: The $x_{re}$ and $y_{re}$ co-ordinates of the release event $e$ and the corresponding

timestamp $t_{re}$ given that $e$ has been triggered on either the tiles or the gold.

*OnDragEvent*: The horizontal co-ordinate $x_{de},\ de\ =\ 1 \ldots n$ and the vertical co-ordinates $y_{de},\ de\ =\ 1 \ldots n$ of the pointing device sampled at 100 ms intervals given that the drag event $e$ occurs on either the tiles or the gold.

**Deriving Cognitive Features.**

We estimate features that capture cognitive abilities from the aforementioned raw data. We also discuss other important features that are based on the user's response to certain stimulus during the execution of an *instance*.

*Drop and Pick Reaction Time, DPT ($f_1$, $f_2$).* At the end of the $A_{resp}$ action, the user picks up the gold coin, $g_c$, appearing at $P_{t_r}$. The time elapsed between the stimulus ($g_c$) appearing and the user picking it up (response) is referred to as the $t^g_{DPT}$ ($f_1$). On the contrary, at the end of the $A_{rew}$ action i.e. after depositing the gold coin at $P_{t_c}$, the user picks up the challenge tile, $t_c$ from location $P_{t_c}$. The time elapsed between the appearance of the stimulus ($t_c$) and the user picking it up (response) is referred to as the $t^t_{DPT}$ ($f_2$). At the first glance, $DPT$ might seem similar to traditional pause-and-click, but there is a fundamental difference between the two. Traditional pause-and-click are highly dependent on what the user is currently reading or exploring [118]. However, $DPT$ is the result of a controlled stimulus and therefore, is not content-specific. $t^t_{DPT}$ is also part of the visual search process where user first performs search and then picks up the tile.

*Visual Search Time, VST & ratio($f_3$, $f_4$).* $VST$ is the time required for the user to visually search and detect the target tile inside the grid. $VST$ is calculated by the subtraction method [33]. The subtraction method involves subtracting the amount of time information processing takes with the process (time elapsed during action $A_{resp}$, denoted as $A^t_{resp}$ and $t^t_{DPT}$) from the time it takes without the process (time elapsed during the action, $A_{rew}$, denoted as $A^t_{rew}$),

$$VST = (A^t_{resp}\ +\ t^t_{DPT})\ -\ A^t_{rew}. \tag{3.3}$$

It is important to consider $t^t_{DPT}$ in the above equation, since a $t_c$ is exposed as soon as a $g_c$ is deposited. So the minuend of equation (3.3) refers to the time elapsed between the exposure of $t_c$ and its correct placement inside the grid. The time elapsed during $A_{rew}$ is simply the movement time and does not involve user's thinking or search time. Therefore, we are able to distill out the plain visual search time for each *instance*. Moreover, note that the subtraction method allows $VST$ to self-adjust to user's specific environment by remaining immune to differing mouse speed or acceleration. We also consider the ratio of $A^t_{resp}$ to ($A^t_{resp}$ + $t^t_{DPT}$), ($f_4$) to capture the phenomena where user can either "search while dragging" or "search and then drag".

*Information Processing Speed, IPS ($f_5$.)* Recall, that at any instance if information (game status) is not lost from the user's working memory then at the $i^{th}$ instance the user is left with 25 - $i$ +1 number of alternatives for the search operation,

$$IPS = \frac{H_i}{VST}. \tag{3.4}$$

The information content $H_i$ at the last instance of the game, when the number of alternatives is 1, is $log_2(1) = 0$. Therefore, $VST$ at that point is equal to $A^t_{rew}$ in an ideal condition. This implies that the action $A_{resp}$ is equivalent to the action $A_{rew}$ at the last instance of the game. Apparently, it is also observable from the game that placing $t_c$ *provided* $|\theta| = 1$ is similar to depositing $g_c$ and requires no active processing.

*Pause and Search, P&S ($f_6$, $f_7$).* While dragging the target tile we noticed that the user sometimes pause and search for the target inside the grid. If user remains at the same pixel for more than $\alpha = 0.1$ seconds while dragging the tile or gold we refer to it as a pause. We measure the ratio of tile paused time to $A^t_{resp}$ during $A_{resp}$ ($f_6$), and the ratio of gold paused time to $A^t_{rew}$, ($f_7$) during $A_{rew}$.

*Effectiveness Of Priming, EOP ($f_{8-20}$).* Recall that constraint *UAC6* provides the priming effect necessary to invoke an automatic processing. We measure the effectiveness of this

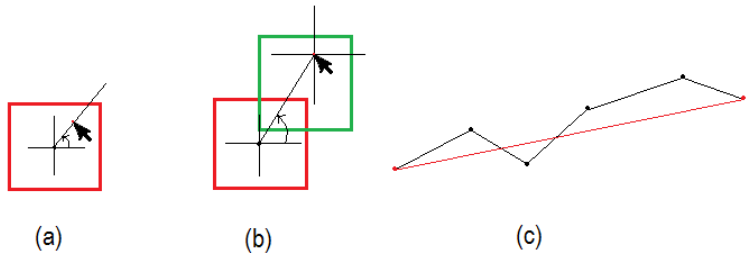


Figure 3.4: (a) Click error angle and click error distance (center to click point). (b) Drop error angle and drop error distance (from the click point to the center of the destination, red bordered tile). (c) Straightness measured as ratio of actual distance to the distance moved.

priming through $EOP_{C6}$,

$$EOP_{C6} = \frac{Area\ overlapped\ between\ src\ and\ dest}{Area\ of\ source\ or\ destination}, \tag{3.5}$$

$$Area\ overlapped = g\left(t_c^{dim},\ t_r^{dim},\ f\left(\angle^{E_t}_{(x_{ce},\ y_{ce})},\ \triangle^{E_t}_{x_{ce},\ y_{ce}},\ x_{re},\ y_{re}\right)\right), \tag{3.6}$$

$$f\left(\angle^{E_t}_{(x_{ce},\ y_{ce})},\ \triangle^{E_t}_{x_{ce},\ y_{ce}},\ x_{re},\ y_{re}\right) = t_c^{centerX,\ centerY}, \tag{3.7}$$

*where*,

$\angle^{E_t}_{(x_{ce},\ y_{ce})}$ = The *Click Error Angle* made from the click point to the (+ve) x-axis with center of the tile being the vertex, at the start of $A_{resp}$;

$\triangle^{E_t}_{x_{ce},\ y_{ce}}$ = The *Click Error Distance* is the distance of the click point to the center of $t_c$ at the start of $A_{resp}$;

$x^{dim}$ = Dimensions of the object $x$;

$f(\cdot)$ = A function returning the co-ordinates of the center of the object;

$g(\cdot)$ = A function returning the area overlapped.

$EOP^g_{C6}$ ($f_8$) refers to the effectiveness of priming while depositing $g_c$ (source) in the bank (destination). $EOP^t_{C6}$ ($f_9$) refers to the effectiveness of priming while placing the $t_c$

(source) on matching tile $t_r$ (destination). $EOP^g_{C6}$ is calculated in a similar fashion as above. We consider related features that might capture the effectiveness of priming as well. We consider the *Drop Error Distance* for tile, $\triangle^{E_t}_{x_{re},\ y_{re}}$ ($f_{10}$) and gold, $\triangle^{E_g}_{x_{re},\ y_{re}}$ ($f_{11}$), defined as the distance from the drop point to the center of their destination. We measure *Click Error Distance* (defined earlier) for tile $\triangle^{E_t}_{x_{ce},\ y_{ce}}$ ($f_{12}$) and gold $\triangle^{E_g}_{x_{ce},\ y_{ce}}$ ($f_{13}$). We also consider the *Drop Error Angle* for tile/gold $\angle^{E_{t/g}}_{(x_{re},\ y_{re})}$ ($f_{14},\ f_{15}$) which is the angle made from the drop point to the (+ve) x-axis with the center of the destination being the vertex and *Click Error Angle* (defined earlier) for tile/gold $\angle^{E_{t/g}}_{(x_{ce},\ y_{ce})}$ ($f_{16},\ f_{17}$). On the other hand we measure the effectiveness of priming due to $C4$ as the ratio of two distances,

$$EOP_{C4} \ = \ \frac{distance\ moved\ w/o\ overlapping\ the\ lines}{distance\ from\ P_{t_c}\ to\ P_{t_r}}. \tag{3.8}$$

$EOP^{L1}_{C4}$ ($f_{18}$) refers to the effectiveness of priming on the top guiding line and $EOP^{L2}_{C4}$ ($f_{19}$) refers to similar measure on the bottom guiding line. The resulting straightness ($f_{20}$) of the movement is measured as the ratio of the distance from $P_{t_c}$ *to* $P_{t_r}$ to the distance moved,

$$Straightness(f_{20}) \ = \ \frac{distance\ from\ P_{t_c}\ to\ P_{t_r}}{distance\ moved,\ D}, \tag{3.9}$$

$$D = \sum_{i=1}^{n-1} \sqrt{\delta x_i^2 + \delta y_i^2}, \tag{3.10}$$

where during *OnDragEvent*,

$\delta x_i^2 = x_i - x_{i+1}$,

$\delta y_i^2 = y_i - y_{i+1}$.

Refer to Figure 3.4 for the angles and straightness features. Note that the angles (measured in degree) and the ratios (with no units) self-adjust to users' specific environment [118].

### 3.1.4 User Classification Technique.

In this Section, we provide details on the classification technique used to classify the users. We use a statistical approach of classifying the users. We model each of the features as random variables $F_1$, $F_2$, ..., $F_n$ and assume class-conditional independence between them. At the $i^{th}$ *instance* of the game a row of feature values $F^i = (f_{1,i}, f_{2,i}, \ldots, f_{n,i})$ are generated. Therefore, for a sequence of $k$ *instances* denoted by $F = (F^1, \ldots, F^k)$, the interaction information can be denoted using a matrix of size $k \times n$,

$$\begin{matrix} f_{1,1} & .. & f_{n,1} \\ f_{1,2} & .. & f_{n,2} \\ .. & .. & .. \\ .. & .. & .. \\ f_{1,k} & .. & f_{n,k} \end{matrix}$$

During the learning stage the probability density functions of the features are estimated using a non-parametric approach. In the classification stage a posterior probability function estimates the probability of a classification being correct.

**Learning.**

The learning phase consists of the estimation of the probability density function for each of the feature vectors. A parametric approach of estimating a density $f$ involves assuming that $f$ belongs to a parametric family of distributions such as the normal or the gamma family. The unknown parameters are then estimated e.g. by maximum likelihood estimation [85]. We resort to using non-parametric approach, in particular, the kernel density estimator [103] to avoid making any assumption on the distribution of the underlying population and to better understand the structure of the data. The easiest non-parametric estimation of a probability distribution is the use of histogram. It is simple but has disadvantages such as discontinuity and high sensitivity to bin edges. Kernel density estimators are superior to histogram and are quite intuitive [103, 120]. For a discussion on density estimation, refer to

Section 2.6.

We estimate the unknown density function $f_j(x)$ of the $j^{th}$ feature vector, represented by a random variable $F_j$, based on its $m$ samples (or training data) $x_1$, .., $x_m$. Assuming that the observations are independent realizations of $F_j$, equation (3.11) represents the estimation of the density function, $\hat{f}_j(x)$, using a kernel density estimator, for univariate case,

$$\hat{f}_j(x) = \frac{1}{mh} \sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right). \tag{3.11}$$

At this point, the estimation of $f_j(x)$ narrows down to (1) choosing a kernel function $K$ (2) and selecting an appropriate bandwidth selection algorithm to determine $h$. Although the choice of kernel functions is not of particular importance for an experiment [103], according to some empirical results, we decided on using Gaussian kernel among others (Epanechnikov, quartic and rectangular kernels) as the kernel function $K$. The kernel estimate is constructed by centering the Gaussian kernel at each observation. Therefore, the value of the kernel estimated at a point $x_i$ is simply the average of the $m$ normal kernel ordinates at that point. Therefore, the width of the chosen kernel function determines the smoothness of the resulting density function. Oversmoothing can happen as a result of larger width whereas undersmoothing can happen due to smaller width. Therefore, selecting the appropriate width $h$ is a crucial task while estimating the density function.

The performance of the kernel density estimator depends on how closely the estimated $\hat{f}_j(x)$ resembles the true $f_j(x)$ of the $j^{th}$ feature. This performance can be measured in terms of the MISE(Mean Integrated Square Error), which globally measures the distance between $\hat{f}_j(\cdot; h)$ and $f_j(x)$,

$$MISE(\hat{f}_j(\cdot; h)) = E[ISE(\hat{f}_j(.; h))] = \; E \int \hat{f}_j(x)^2 \, dx - 2E \int \hat{f}_j(x) \, f_j(x) \, dx + \int f_j(x)^2 \, dx \; . \tag{3.12}$$

We use least square cross validation (LSCV) [84, 16] which is a simple data-driven bandwidth selector and whose main motivation comes from equation (3.12). The first two terms of the right hand side of equation (3.12) is unknown and depends on $h$. In order to solve this issue,

we resort to a leave-one-out LSCV. A LOO LSCV splits the observations of a feature vector into two parts. The $m-1$ first observations are used for density estimation and the remaining one is used for assessing the accuracy of this estimation. This is repeated $m$ times, each time leaving out a new observation. The mean of the resulting $m$ quantities is then obtained. The process is repeated until a $h$ is found which minimizes the mean error. Though there exists several other bandwidth selection algorithm [57], we chose LSCV since it is simple and intuitive. Moreover, it is data-driven and attempts to minimize MISE($\hat{f}_j(\cdot; h)$) asymptotically to some extent. The learning stage, therefore, involves constructing the density functions for each feature using Gaussian kernel as the kernel function $K$ with bandwidth $h$ selected using LOO LSCV.

**Classifying.**

We assume class-conditional independence between the features, modeled as random variables $F_1$, $F_2$, ..., $F_n$, for a user $u_w$, $w \in \{1, \ldots, m\}$ and for the $i^{th}$ *instance* $Pr\left(F^i|u_w\right) = \prod_{j=1}^{n} Pr(f_{j,i}|u_w)$. Although the class-conditional independence between features is not true in general, the assumption works well in many complex real life system. The posterior probability of a user $u_w$ given an *instance*,

$$Pr\left(u_w|F^i\right) = \frac{Pr\left(u_w\right) \prod_{j=1}^{n} Pr(f_{j,i}|u_w)}{Pr(f_{1,i},\ \ldots,\ f_{n,i})}. \tag{3.13}$$

We then classify a test sample according to the largest posterior probability. We accept a sequence of *instances* as genuine if the number of accepted *instances* exceeds some decision threshold $\alpha$. The value of threshold $\alpha$ is set in such a way so that the false acceptance rate is close to the false rejection rate.

## 3.2 Security Model

We discuss the security of our system against impersonation attacks. Resistance to impersonation attack is an important aspect of any biometric system. Description of impersonation

attack in general is provided in Section 2.2.4. In this section, we also provide details on the way we measure the error rates of our system.

### 3.2.1 Error Metrics

The security of a biometric system is generally measured with False Acceptance Rate and False Rejection Rate [60]. The system should not accept an imposter (FA) and at the same time should not reject a true user (FR). Therefore, both these metrics should remain close to zero. We evaluate our system performance using user $u$'s self test sessions and other non-self test sessions from $n-1$ users. A positive test *session* of length $l$ *instances* is considered misclassified for a user $u$, if the classifier outputs a score below the threshold $\alpha$. This is referred to as a False Rejection. On the other hand, a negative test *session* is considered classified if the classifier's output score is above the threshold $\alpha$. This is referred to as a False Acceptance. We calculate the FAR as the ratio between FA and TN, where FA is the number of false acceptance and TN is the number of test *sessions* belonging to the $n-1$ other users. The FRR is calculated as the ratio between FR and TP where FR is the number of false rejection and TP is the number of test *sessions* belonging to the user $u$. And subsequently at the end, error rates for all the $n$ users are averaged to get the average FAR and FRR of the entire experiment.

### 3.2.2 Impersonation Attacks

To reduce FAR, any biometric-based technology must be resistant against attackers capable of reproducing the target user's input. In our system, the attacker has to imitate the victim's intrinsic cognitive abilities, which even if carefully estimated and used, must be very hard if not impossible. Using a CT like ours, capable of capturing a multitude of cognitive features makes it even harder for the adversary to mimic the real user. We build a web-based program using HTML5 and JS capable of simulating any user's game playing activity once fed with data collected during the data acquisition period. We then select trained users or attackers

and instruct them to observe and imitate other users' simulations. We provide the results of the impersonation attack in Section 3.3, Experiment-III.

## 3.3 Experiments And Results

We formulated two questions. (1) Is it possible to verify a user based solely on the derived cognitive features with high accuracy under (a) controlled condition and under (b) non-controlled condition? (2) How effective are impersonation attacks against our system? In Section 3.3.3 we also provide experimental evidence on working memory capacity and priming effect on automatic processing.

We devise three separate experiments to answer the above questions. We obtained approval from the Research Ethics Board of our Institution. The $1^{st}$ experiment was carried out in a controlled environment with 18 graduate students. The $2^{nd}$ experiment consisted of 130 Turkers from Amazon Mechanical Turk. And the $3^{rd}$ one was carried out with 15 Turkers. Once the participants agreed to the online consent information to proceed, a short video showed how the game is played for a few *instances*. Then participants were instructed to play the game. In all the experiments, interaction data were recorded using JavaScript and submitted passively via AJAX requests to the web server. We had a repository of 25 partitioned images. We randomly picked a set of distinct images for each user in a *session*. Users were never exposed to the same image more than once.

### 3.3.1 Experiment Setup

We now provide overviews on the experimental setup for the three experiments.

**Experiment-I: Goals and Setup**

The goals of the $1^{st}$ experiment were to figure out (1) Accuracy and efficiency (in terms of enrollment and verification time) of our system when users are trained in a non-distracting environment using a single platform, (2) perform an analysis on the derived features, (3)

evaluate the performance of the system when user tried to authenticate over a period of time. Each user in a *session* is required to play the game 7 times, every time with a new random partitioned image, resulting into a dataset of at least $25 \times 7 = 175$ *instances*. Afterwards, they completed the exit survey. All of them used a PC with 2.10 GHz Intel i3, 4GB RAM and an wireless optical USB mouse. They used Google Chrome on a screen of resolution $1366 \times 768$ (96 DPI) in Windows 7 SP1 OS.

**Experiment-II: Goals and Setup**

The goal of this experiment was to evaluate the system with random users from different parts of the world who train themselves under their own supervision and remotely authenticate later. We created 4 HITs altogether, the $1^{st}$ HIT was created with 130 assignments to have 130 unique users. Users must have $\geq 98\%$ HIT approval rate and $\geq 5000$ HITs approved to qualify for our HIT. Each assignment was worth \$0.7. We refer to each HIT as a *session*. The users were directed to the website hosting the game. After watching the video, they were required to play 7 games and then complete an exit survey. On completing the assignment, users were asked to copy-paste a code (generated on our website) back to Amazon. The $2^{nd}$, $3^{rd}$ and the $4^{th}$ HITs were created for the inter-session experiments (Section 3.3.2).

**Experiment-III: Goals and Setup (Impersonation Attack)**

Impersonation attack demands trained users capable of mimicking a user's game playing activities. Therefore, participants were selected based on how fast they completed the previous *sessions*. We selected 15 Turkers from the $2^{nd}$ pool. Each assignment was worth \$0.5. We built a simulation program using HTML5 and JS which when fed with user's raw data, could simulate accurately the game playing activities (similar to a recorded video). Each participants were required to watch and mimic the simulations of 3 users or victims. We considered a strong attack scenario. We (1) selected victims who had less cognitive capabilities compared to the attackers (i.e. the victims had relatively higher visual search times), (2) displayed a clock while the simulation was playing, (3) provided the same image or grid

in the attack phase, (4) declared a bonus of \$0.5 if the Turkers can mimic accurately and (5) allowed to repeat the attacks as many times as desired. All participants were given instructions to observe when and how tiles and gold are picked, dragged and dropped.

### 3.3.2 Experimental Results and Analysis

In this section, we provide the experimental results and analysis for all the three experiments.

**Experiment-I: Results and Analysis**

Intra-Session Evaluation. As the name implies, here we consider the training and test data from the same *session*. The dataset was divided into two parts. The first part consisting of 5 games ($g_1$, ..., $g_5$) each of $\geq 25$ *instances*, is used for training purpose and the last two games $g_6$, $g_7$ are separately used for testing purpose. The average EER is 0% suggesting that all users have consistent game playing activities in a continuous *session* (Figure 3.11). Figure 3.5 shows the variations in FAR and FRR, at threshold $\alpha \approx 0.5$, as the number of *instances* are varied. Less number of *instances* e.g. 16 would have significantly decreased the verification time with an EER of 0%. The verification time is generally referred to the time taken to collect the verifiable biometric data and the time taken to complete the classification task [60]. It took an average of 115.8 seconds to complete one game (25 *instances*) and an average of around 63.2 seconds to complete part of the game (16 *instances*). Note that these time intervals considered the time elapsed during the whole game. *Slight* user distractions that might occur even in a controlled environment are also taken into account. The time elapsed during $A_{resp}$ and $A_{rew}$ actions only (shorn of distractions) are relatively less, around 71 seconds for 25 *instances*. Nevertheless, these intervals are comparable to several recently proposed authentication techniques (Table 3.1). A comparison of our error rates with other established system is provided in Table 3.3. Our classification time does not have any particular impact on the verification time. The verification time in a system like ours, is highly dependent on the cognitive abilities of the participants and other design factors. Refer

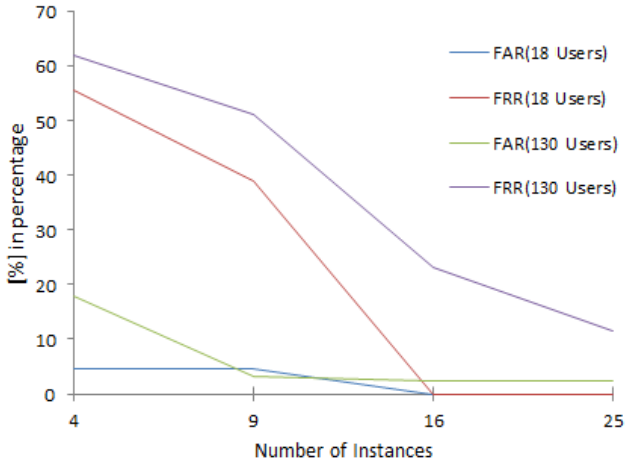


Figure 3.5: Intra-session Evaluation: Avg. FAR and FRR at $\alpha = 0.5$ with varying number of *instances*.

| | [Ours] | [50] | [44] | [12] |
|---|---|---|---|---|
| Verification Time(Approx.) | 71s, 2.5min | 39s | 25s | 5-6min |
| Test Session Size | 25 instances | 25 characters | 25 strokes | 540 items |
| Enrollment Time(Approx.) | 10min, 20min | 6.5min | 6.7min | 30-40min |
| Enrollment Session Size | $\geq$175 instances | 250 characters | 400 strokes | 3780 items |

Table 3.1: Comparison of verification time, enrollment time, test and enrollment (train+test) session size for a particular user.

to Section 3.3.4 for a discussion.

**Feature Analysis.** We use data from Experiment-I for analyzing the features. Figure 3.7 shows the Pearson correlation coefficients of pairs of features in a color-coded plot. There are a few highly correlated features. According to our observation these correlated pairs slightly improve the classification accuracy, similar to [48] without adding any significant burden to the training time. Therefore, all the 20 features are used in the later experiments. On the other hand, the plot provides us with some interesting information. Notice that features $f_8$ and $f_{11}$ are negatively correlated whereas $f_9$ and $f_{10}$ are positively correlated. $f_{10}$ and $f_{11}$ are the difference in distance from the tile and gold release co-ordinate to the center of their destinations respectively. Intuitively, the smaller these distances are, the greater the overlapped areas ($f_8$, $f_9$) should be and vice versa. This is true for the gold but not for the challenge tile. The reason is that users mostly pick up the tile by clicking on the

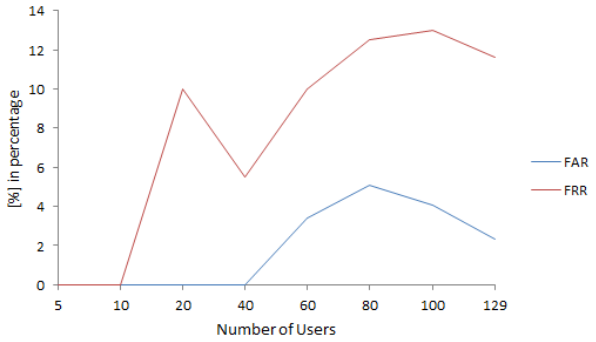


Figure 3.6: Intra-session Evaluation: Avg. FAR and FRR at $\alpha = 0.5$ with varying number of users.

peripheral region, such that the mouse pointer does not block the symbol on the tile. This allows the user to drag and search at the same time. Whereas when picking up the gold such *view* is not necessary. On the other hand, we considered the strength of each of the features in identifying the users. Each feature was in turn used for learning and classifying. We found that 11 features related to drop and pick reaction times, visual search and ratios, and priming (guiding lines and angles) are in the top 11 based on Avg. EER. Refer to Table 3.2 for details.

**Inter-session Evaluation.** The participants were invited to play on three other occasions, each separated by 1-day, 2-days and 3-days intervals respectively. We consider these intervals to be congruous with real account login intervals. The training data came from the original data acquisition session ($g_1$, ..., $g_5$). At each occasion the participants were required to complete one game using any machines (except cellular devices) and browsers at their most suitable time. At $\alpha = 0.5$, FAR remained 0% through all *sessions* with FRR being 0%, 5.6% and 5.6% on the 1, 2 and 3-days *sessions* respectively. Statistics of the *sessions* include Chrome (66.7%), Safari (27.7%), Mozilla (5.6%), others on a range of OS Win (55.6%), Mac (38.8%), Linux (5.6%). Mouse type statistics from exit survey include wireless or wired mouse (83.3%) and the rest laptop touchpad (user claimed). Whereas, only mouse was used in the training phase.

| **Feature** | **EER** [%] | **FAR** [%] | **FRR** [%] |
|---|---|---|---|
| RatioPauseTile_TileMoveTime (f6) | 55.56 | 11.76 | 88.89 |
| PrimedBottomGuidlingLine(f19) | 66.67 | 17.65 | 88.89 |
| RatioTileMoveTime_ImpureVST (f4) | 66.67 | 23.53 | 83.33 |
| GoldClickErrorAngle (f17) | 66.67 | 5.882 | 100 |
| GoldPickReactionTime (f1) | 72.22 | 11.76 | 83.33 |
| PrimedTopGuidingLine (f18) | 72.22 | 23.53 | 88.89 |
| PureVisualSearchTime (f3) | 72.22 | 5.882 | 88.89 |
| GoldDropErrorDistance (f11) | 72.22 | 0 | 100 |
| TileDropErrorAngle (f14) | 72.22 | 29.41 | 88.89 |
| Straightness (f20) | 72.22 | 0 | 94.44 |
| TileClickErrorDistance (f12) | 72.22 | 0 | 94.44 |
| RatioPauseGold_GoldMoveTime (f7) | 77.8 | 17.65 | 88.89 |
| TilePickReactionTime (f2) | 77.8 | 17.65 | 94.44 |
| GoldDropErrorAngle (f15) | 77.8 | 5.882 | 100 |
| GoldAreaOverlapped (f8) | 82.35 | 17.65 | 94.44 |
| TileClickErrorAngle (f16) | 82.35 | 5.882 | 100 |
| InformationProcessingSpeed (f5) | 82.35 | 17.65 | 88.89 |
| TileDropErrorDistance (f10) | 82.35 | 17.65 | 94.44 |
| TileAreaOverlapped (f9) | 82.35 | 17.65 | 94.44 |
| GoldClickErrorDistance (f13) | 82.35 | 0 | 94.44 |

Table 3.2: Ranking of features based on avg. EER. FAR and FRR at $\alpha = 0.5$ (Experiment-I)

**Experiment-II: Results and Analysis**

**Intra-Session Evaluation.** Similar to Experiment-I, $g_1, \ldots, g_5$, were used for training and $g_6,\ g_7$ for testing purpose. We noticed abnormal $VST$s in the dataset for a few cases. On closer scrutiny and observing the simulations for such cases we noticed very large *Drop and Pick Reaction Time*, $t^t_{DPT}$ and $t^g_{DPT}$ (Figure 3.9). This depicts that users are more likely to get distracted at the end of the actions ($A_{resp},\ A_{rew}$) rather than while performing them. We detect these *extreme* outliers using interquartile range for each user $u$, with the upper fence $UF = f_u^{Q3} + (3 \times f_u^{IQR})$, $f_u \in \{t^t_{DPT_u}, t^g_{DPT_u}\}$ and replacing them with $UF$. Noise removal was done separately for the training and test dataset. Figure 3.5 shows the avg. error rates for varying number of *instances*. Our classifier reaches an FAR of 2.3% and FRR of 11.6% with $\geq 25$ *instances*. It took an average of 95.3 seconds to complete one game (25 *instances*) and an average of around 58.7 seconds to complete part of the game (16 *instances*). The

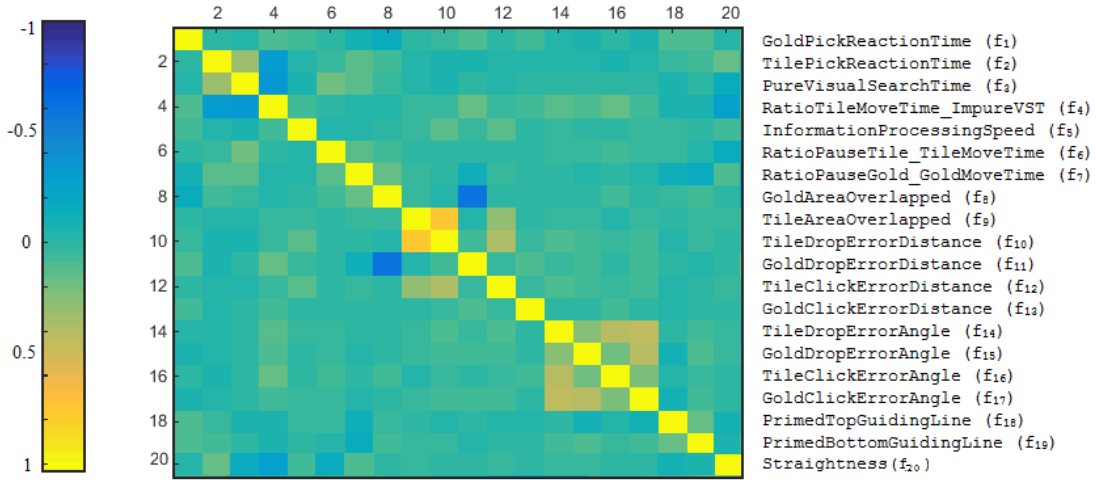


Figure 3.7: Correlation coefficients [-1,1] of pairs of features in a color-coded plot.

time intervals consider the time elapsed during actions $A_{resp}$ and $A_{rew}$ only. The actual time to complete an overall game cannot be measured properly here, since the intervals can be highly influenced by distractions. The average time it took to complete one assignment of the HIT is 24.3 minutes. This included (1) reading instructions, (2) watching video, (3) completing games, (4) completing the survey and finally (5) submitting the generated code back to Amazon. We make a plausible assumption that it takes 4/5 of 24.3 minutes to play all 7 games and 1/5 to complete the other listed 4 tasks. Then the Turkers took on an average 2.5 minutes to complete one game. This is again comparable to other established authentication systems (Table 3.1). Mouse type statistics from the exit survey included wireless/wired mouse (61.3%), laptop touchpad (38.7%) (user claimed). Figure 3.6 shows how the FAR and FRR stabilizes after training data size reaches 60 users. This suggests that the error rates are uncorrelated with the sample size (number of users) and will remain mostly uninfluenced by it.

**Inter-Session Evaluation.** We created another 3 HITs, each separated by 1, 2 and 3-days intervals respectively. We invited all the previous users through emails. We made sure users completing the $4^{th}$ HIT had already participated in the previous 3 HITs. Each assignment

| Works | FAR | FRR | Session Size | Type | Notes |
|---|---|---|---|---|---|
| [2] | 2.46% | 2.46% | 2000 Mouse Actions | MDS | Free mouse movement |
| [44] | 6.3% | 6.3% | 20 Strokes | MDS | Confined within a task |
| [118] | 1.3% | 1.3% | 20 Mouse clicks | MDS | Free mouse movement |
| [50] | 2.11% | 2.11% | 25 Text Characters | HBS | Confined within a task |
| [40] | 0.01% | 4% | 683 Characters | KDS | Fixed-text input |
| [98] | Accuracy 93.3% - 99.5 % | | 200 Characters | KDS | Free-text input |
| [Ours] | 0%, 2.3% | 0%, 11.6% | 25 Instances | CBS | Confined within a task |

Table 3.3: Comparison with other approaches. Mouse Dynamics System(MDS), Keystroke Dyanmics System (KDS), CBS (Cognitive-based Biometric System), HBS (Homogeneous physio-behavioral Biometric System)

was worth \$0.2. We received 49, 37 and 37 valid submissions until the HIT expired. The assignment required completing only one game. As before the classifier used the initial acquired data $g_1$, ..., $g_5$ for learning. The FAR were 2.08%, 0%, 0%, and FRR 8.16%, 8.11%, 5.45% with $\alpha = 0.5$ on the 1-day, 2-days, 3-days *sessions* respectively. This suggests that even after small periods of inactivity and using the original training data, the classifier can still distinguish the users.

**Experiment-III: Results and Analysis**

A successful attack would require reproducing the cognitive features of a victim. Figure 3.8 depicts the *maximum* number of *instances* (out of all attempts) that have been correctly classified to the corresponding victims. A maximum of $\geq 3$ *instances* were correctly mimicked by user 3, 10 and 13. None of the attackers would have successfully authenticated with our experimentally set threshold of $\alpha = 0.5$. In fact, user 4, 8, 9, 12 and 15 were identified as "themselves" in one of the games (attacks). A successful attack in this case would require mimicking almost all 20 cognitive features, which is a very hard task. Moreover, the challenge tiles appeared randomly, and the grid was shuffled at each instance according to constraint *UAC3*. So the sequence of challenges in the simulation and actual attack differed, making it harder to recall the corresponding $VST$s and other reaction times.

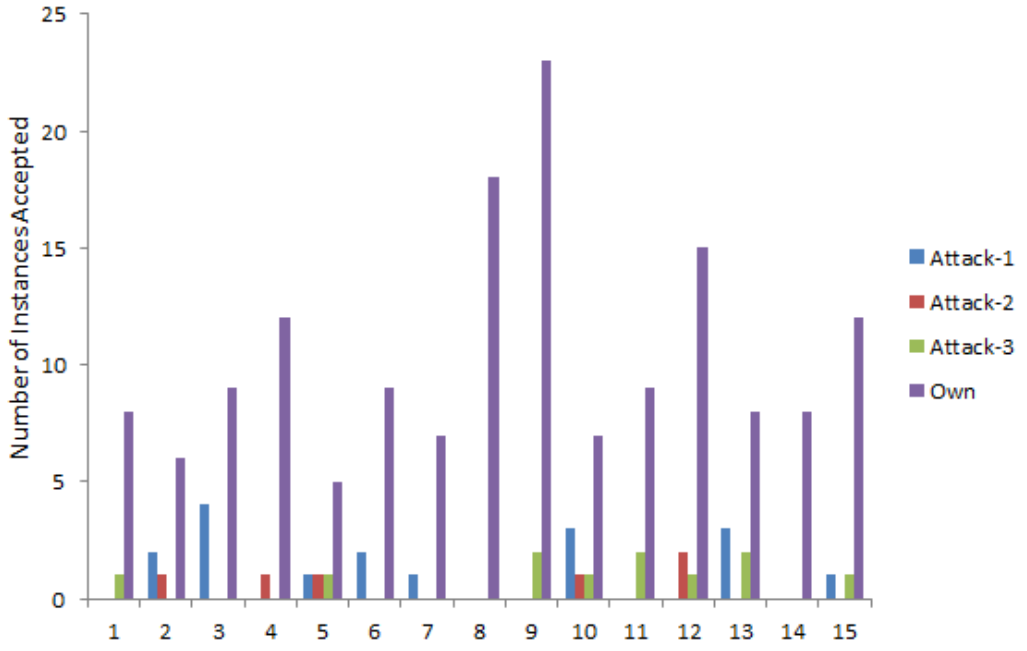


Figure 3.8: Results from Impersonation attack. Most *instances* are accepted as "own" rather than as the victims' (attack-1, 2 and 3).

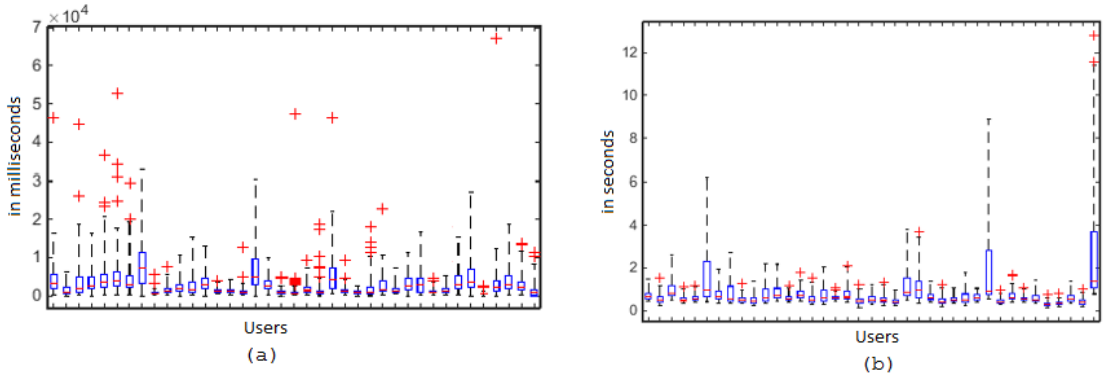


Figure 3.9: Box plot before noise removal (whiskers set to 3) showing outliers, (a) Tile Pick Reaction Time (in ms) (b) Gold Pick Reaction Time (in second).

### 3.3.3 Evidence of Cognitive Processes

We analyze our data and provide evidence for working memory and priming effect on automatic processing. Recall that after a successful match, the user can observe the game status, ($UAC2$). If this information is not lost from the working memory, then $|\theta|$ must decrease with each *instance*, resulting in a sequence of descending **V**isual **S**earch **T**ime, $VST$. Considering information storage is likely to occur near the end of the game, since it is easiest to recall the last remaining empty cell, we find the length of sub-sequence $l = n - k + 1$ of $n$ *instances* such that $VST_k > VST_{k+1} > \ldots > VST_n$ for $|\theta|_k > |\theta|_{k+1} > \ldots > |\theta|_n = 1$. Since the user might store partial information as well, we allow some tolerance such that $v$ number of violations (sign changes) can happen in the sequence. Figure 3.10(a) shows the average sub-sequence length $l$ (for 5 games) when $v$ is varied from 0-3. We can observe the variations in working memory capacities among a group of users.

On the other hand, after triggering a prime, a user might (1) receive it and get influenced (invocation of automatic processing) or (2) receive it but *not* get influenced (no invocation of automatic processing) or (3) *not* receive it at all e.g. a *cautious* user dropping a gold with overlapping area $\geq 90\%$. We now give a detail discussion. Considering that the prime has been triggered at the $i^{th}$ *instance*, the following likely explain the three cases at the $i + 1^{th}$ *instance*, for *Constraint* $UAC6$. (1) $EOP^i_{C6} < EOP^{i+1}_{C6}$ and $g_c$ got misplaced at the $i^{th}$ *instance*. In other words, user *drops* $g_c$ with relatively higher pin-point accuracy at the $i + 1^{th}$ *instance*. (2) $EOP^i_{C6} \geq EOP^{i+1}_{C6}$ and $g_c$ got misplaced at the $i^{th}$ *instance*. That is user received the prime but did not get influenced. (3) $g_c$ never got misplaced at the $i^{th}$ *instance*. Figure 3.10(b) shows the average percentage for each of these cases (for 5 games) when primes are triggered. Notice that due to slight inaccuracies in placement, all users received at least some primes. Around 30% of the users never *missed* getting influenced (without conscious awareness) by the prime whenever they received it.

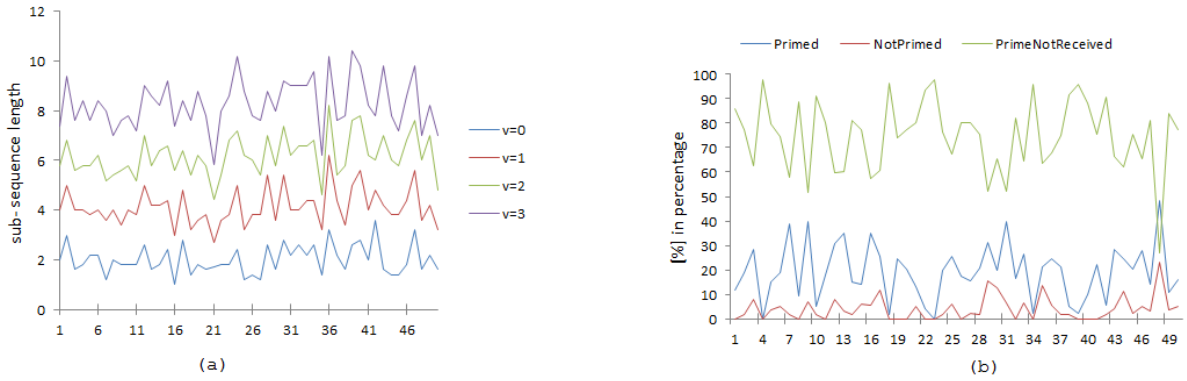


Figure 3.10: Users randomly chosen from Experiment-II. (a) Average sub-sequence length with a linear relationship between $VSTs$ and search set sizes indicating differences in working memory capacity of different users. Different curves represent varying number of violations from linear relationship. (b) Percentage of the three cases when prime is triggered (Section 3.4). Users are influenced in different ways.

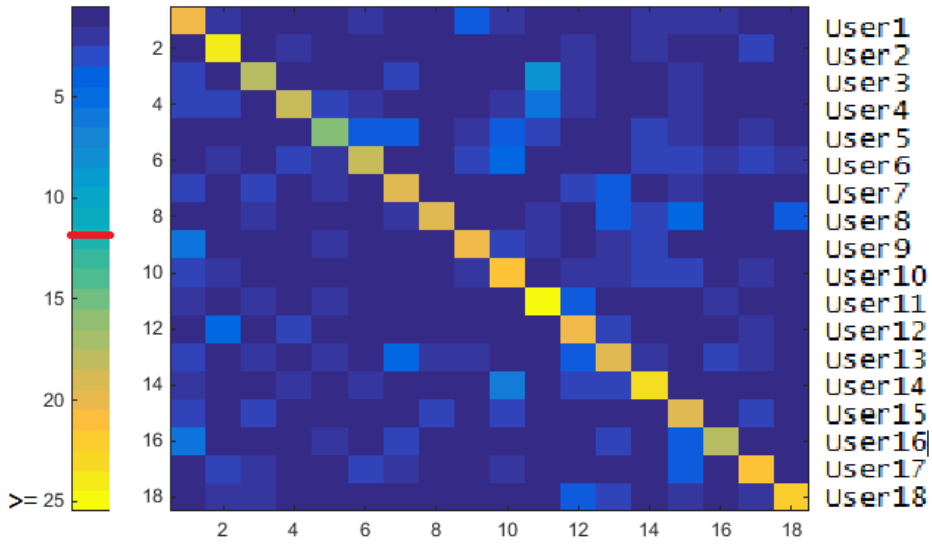


Figure 3.11: Matrix representation of $n$ test sessions matched against $n$ templates. Threshold set at $\approx 0.5$ marked by red line. The horizontal bar shows the number of *instances* and corresponding color codes.

### 3.3.4 Usability of Authentication Systems

The usability challenges here can be broadly divided into two categories (1) designing a CT that is enjoyable to execute, but has strictly defined constraints needed to invoke cognitive processes, (2) decreasing the verification time during user authentication. Biometric system requires building profiles for users during the enrollment phase. Systems like ours, requiring the user to interact within a confined environment, such as playing a game, must therefore be engaging enough to retain the interest and the attention of the user. The average SUS score is around 71. These scores are within the user-friendly industrial software ratings [63]. This suggests that the majority of the users found the game entertaining.

One of the main concern of static behavioral biometric system is the verification time. Recall, that the verification time is the amount of time it takes to collect the necessary biometric data and the time it takes to make an authentication decision [60]. The amount of time it takes to collect the necessary biometric data is of a greater concern. Unlike, other established systems, our identification process considers each *instance* of the game and attempts to classify that *instance* to a user. This means if the system can classify certain percentage of the *instances* to the legitimate user, the system can authenticate the user with some confidence. Therefore, for example, instead of 25 *instances*, the system can build up enough confidence on the authentication task based on only 16 *instances*. This would significantly decrease the log in time.

The verification time in our system is also dependent on the movement time of the tiles and gold. The closer the challenge tile is to the grid the smaller is the movement time during $A_{resp}$ and $A_{rew}$ actions. Verification time is also dependent on the size of the grid and the tiles. This again translates to smaller distance leading to smaller movement time. However, such adjustments should not drastically degrade the performance of the system.

## 3.4 Discussion on User Identification System

Our system provides a proof-of-concept for cognitive *only* biometric authentication. Our empirical results showed that the system is robust against impersonation attacks. Widespread popularity and availability of games on computers and smart phones [45, 93] suggests that a well-designed CT (game) can provide a promising approach to user authentication. In this section, we discuss the applications and limitations of our system. Chapter 5 provides a detail discussion on the attainable future goals and open problems.

Our cognitive based authentication system assumes that the user plays consistently and use his cognitive abilities appropriately. This is arguably a desirable property and careless treatment of security should be punished by denying access. Our proposed system, like other biometric systems cannot authenticate a user if his biometric data is damaged (e.g. a severe burn to one's finger). In cognitive based systems the damage may be long term and caused by cognitive and mental disorders, or short term when under the influence of substance. To provide user access in such cases depending on the type of the damage and the organizational policy, a different type of authentication system such as a password system, should be used as backup. Cognitive abilities can change slowly over time due to age and experience. In such cases, an adaptive enrollment mechanism is necessary to capture and represent the most current features of the user. Our system can be used as a stand-alone system, or can be used in a multi-factor authentication system. Since well selected cognitive features cannot easily be mimicked the authentication system will be secure.

# Chapter 4

# UTILIZING HUMAN COGNITIVE ABILITIES IN CAPTCHA

Movtcha is a *clear form* game Captcha that considers the linear relationship between visual search time and search set size to differentiate humans from computers. The design of Movtcha also takes into account human behavioral analysis to eliminate noise during search time estimation. At each *instance* of Movtcha the user performs a visual search task by dragging and dropping the challenge tile onto the target tile inside the grid. A series of visual search time collected for a particular number of *instances* are then used to differentiate humans from computers. If the search time grows linearly (roughly, possibly with a few outliers) with the search set sizes, then the system authenticates the user as a human. Unlike other Captcha systems, Movtcha is presented in *clear form*. This makes it language and experience independent. It is resistant against random, automated and static relay attacks. In Movtcha, challenge generation and response evaluation are automated. Therefore, Movtcha can be used in large scale application.

Movtcha consists of three major modules. They are as follows: (1) The Generation Module generates the Movtcha challenges, which are basically the processed images. (2) The Data Acquisition Module provides Movtcha in the form of a web-based game. The user uses a mouse to solve the Movtcha. The interaction data is recorded using JavaScript and submitted passively via AJAX requests to the web server. (3) The Evaluation Module differentiates a human from a computer based on the acquired data and an accuracy metric.

This chapter has four sections. Section 4.1 and Section 4.2 provide details on the design of the Movtcha game and how it invokes certain mental or cognitive processes. Section 4.3 provides discussion on the robustness of Movtcha against different attacks. Experiments and

results are provided in Section 4.4.

## 4.1 Movtcha Design and Execution

The design of Movtcha follows that of Neisser serial search model [73] and Wolfe guided search theory [109]. Movtcha provides a structured visual field, with the aid of a grid, and makes it feasible for the human user to perform search only on the highly *activated* tiles, $\theta_{sub}$, of the image. The parallel stage helps the user in recognizing the highly *activated/exotic* tiles in the grid. Once that stage is over, the user searches serially to find the target. It is also necessary to ensure that the design of Movtcha generates robust estimation of the visual search time. This section provides details on (1) the design of Movtcha, (2) manipulation of the image to be presented in Movtcha, (3) the extraction of cognitive and behavioral features and (4) the authentication mechanism.

### 4.1.1 The Cognitive Task as Movtcha

An image of size $x \times y$ pixels, $(width \times height)$, is first broken into a grid, $g$, containing $\theta$ 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 $\theta_{sub}$. The game starts with the user being challenged with a tile $t_c$ at position $P_{t_c}$. The objective of the user is to drag and drop $t_c$ onto the corresponding target tile, $t_r$, inside $g$. We call this search action or 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. We refer to the location of the target tile inside the grid as $P_{t_r}$. And we refer to the number of *exotic* tiles that need to be examined on reaching the

target tile as $|\theta_{sub}^{P_{t_r}}|$. Refer to Figure 4.1.

**Constraints & Helpers.** The game must invoke serial self-terminating visual search of a user after the parallel stage. In order to guarantee (1) its invocation *(C1, C2 & H2)* (2) its correct measurement with the aid of 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/image appear. The tile holder moves randomly within $R_{t_c}$ and the challenge tile is visible only when hovered or dragged. This ensures no prior exposure of the challenge tile or search set which can bias the search time [105]. This constraint also ensures that user drags while searching. In addition when dragged, the tile position is adjusted relative to the click point of the mouse such that the mouse pointer does not obstruct the view of the tile. In other words, the mouse pointer appears at the lowest right corner of the tile (Figure 4.2). This also facilitates the drag and search process.

***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 [73]. Refer to Figure 4.2.

***C3.*** If the time taken in $A_{resp}$ crosses some experimentally set threshold $\lambda$ (Section 4.4.2), 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*.

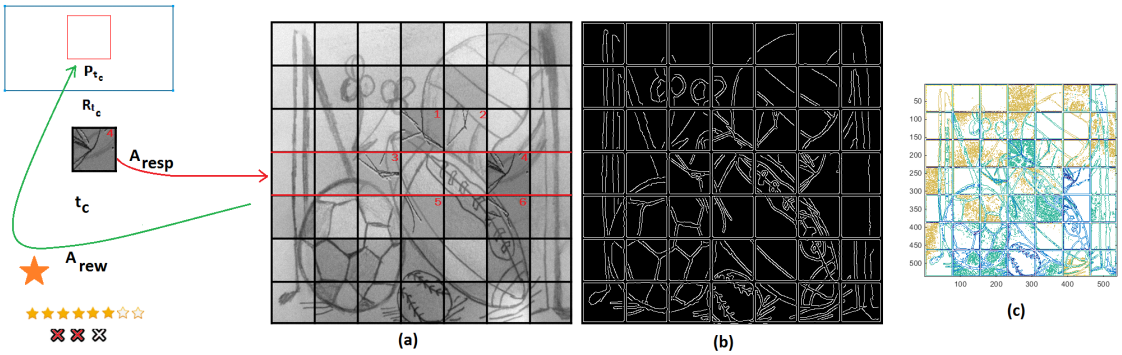


Figure 4.1: Best viewed in soft copy. (a) An *instance* of Movtcha where user has collected 6 stars and made 2 mistakes. The grid size is $|\theta| = 49$, search set size is $|\theta_{sub}| = 6$, and the position of target tile inside the search set is $|\theta_{sub}^{P_{t_r}}| = 4$. (b) Edges of $I_P$ using Canny edge detection algorithm. (c) Contour plot of $I_P$.

***C4.*** $A_{rew}$ action demands smooth movement of star, $s_r$, since no cognitive thinking in particular, visual search, is required to execute $A_{rew}$. Larger amount of pauses during the movement of $s_r$ signifies that the user was distracted. If the amount of pauses crosses some experimentally set threshold $\zeta$ (Section 4.4.2) 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 the tile/star.

***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 (Section 4.3.2).

### 4.1.2 Cognitive and Behavioral Feature Extraction

We refer to any features collected during the execution of a cognitive task as cognitive features. On the other hand, behaviors do not necessarily invoke any *particular* cognitive process. Features collected through observation of human behavior, such as in behavioral

biometrics (*while browsing, typing*), are referred to as behavioral features. Such distinction can also be found in [50].

**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 [33]. The subtraction method 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 ($MT_{A_{rew}}$),

$$VST = (MT_{A_{resp}} + t^t_{RT}) - MT_{A_{rew}}, \tag{4.1}$$

*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).

The minuend of equation (4.1) refers to the time elapsed between the exposure of the target tile and its correct placement inside the grid. The time elapsed during action $A_{rew}$ is simply the movement time and does not involve user's thinking or search time. Therefore, we are able to distill out the plain $VST$.

**Pause Time, PT.** This feature is required to enforce constraint *C4*. *C4* enforces almost smooth movement during $A_{rew}$ and in turn provides better estimation of $VST$, when using the subtraction method. While dragging $t_c$ we noticed that users sometime pause and search for the target $t_r$. If user remains at the same pixel for more than $\alpha = 0.1$ seconds we refer to it as a pause. We assume that 0.1 seconds is a short time for a human, who may or may not move the mouse at all during this period. We measure the number of pauses and derive the total paused time, $PT_{A_{resp}}$. Since $A_{resp}$ action happens during a cognitive task execution, $PT_{A_{resp}}$ is a cognitive feature. On the other hand, the $A_{rew}$ action does *not* involve any

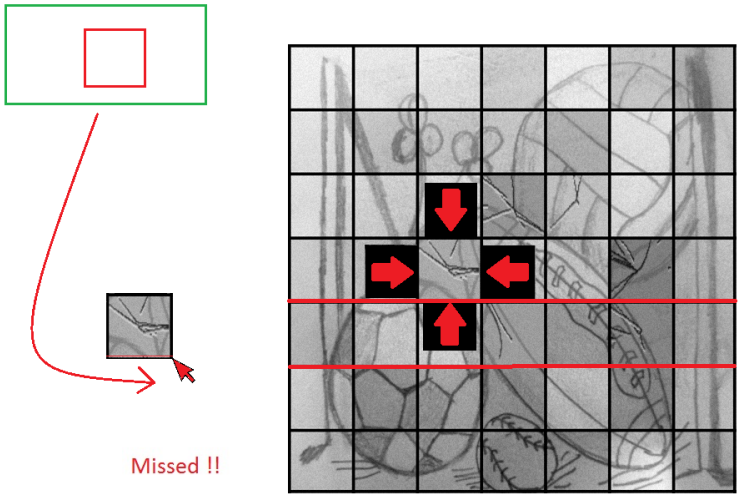


Figure 4.2: A skipped search, triggered by $C2$, for an *instance*. Missed target tile is highlighted. Bounded boxes are not in scale.

cognitive process in particular, visual search or unconscious automatic processing, and this makes it a behavioral feature. $PT_{A_{rew}}$ should be zero in an ideal condition. In practice, if $PT_{A_{rew}}$ crosses some experimentally set threshold $\zeta$ then constraint $C4$ is enforced. More specifically, $PT_{A_{rew}}$ is measured while $s_r$ is dragged from $P_{t_r}$ until it reaches $P_{t_c}$. User pausing while locating and dropping $s_r$ inside $P_{t_c}$ *bounding box*, is not taken under consideration.

$$\#\ of\ pauses\ =\ \sum_{i=1}^{n} p_i\ where\ p_i\ =\ \begin{cases} 1\ if\ t_i\ >\ \alpha \\ 0\ otherwise \end{cases}$$

$PT_{A_{rew}}$ is required to enforce constraint *C4*. *C4* reduces noise (distraction) and enforces smooth movement during $A_{rew}$ and in turn provides better estimation of $VST$, when using the subtraction method.

### 4.1.3 Telling Humans and Computers Apart

We use an accuracy matrix $\triangle^{A}_{VST}$ 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 $\langle i = 1,\ 2,\ \ldots,\ n \rangle$ from human user and bot respectively. The two series are first arranged according to decreasing order of $|\theta^{P_{t_r}}_{sub}|$. 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 less than $i_e$. An accuracy matrix $\triangle^A_{VST}$ is then obtained using Algorithm 1.

$\triangle^A_{VST}$ is calculated as the ratio of the summation of the plus points $s^+$ and the summation of a strictly decreasing series. If the plus points follows a strictly decreasing trend then the resulting accuracy matrix $\triangle^A_{VST} = 1$ 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 $\alpha$. (2) $VST - MT_{A_{rew}}$ must be least when the search set size $|\theta_{sub}| = 1$ (Claim 4.1). In other words, the visual search time is expected to be the least when the search set size is 1.

$$Accept = \begin{cases} true\ if\ \triangle^A_{VST} \geq \alpha \wedge condition(2) \\ \\ false\ otherwise, \end{cases}$$

**Claim 4.1** *If the visual search effort is represented as $\rho$, then $\rho = 0$ and $VST - MT_{A_{rew}}$ is least only when the search set size $|\theta_{sub}| = 1$.*

*Justification.* We assume that Movtcha can be modeled as a choice reaction experiment [1] where user choose $t_r$ from $\theta_{sub}$. Given that the choices are equally probable, $H_i$ can represent the amount of information content or visual search effort needed at the $i^{th}$ *instance*. According to Hick Hymen law [1], $VST = MT_{A_{rew}} + \frac{H_i}{IPS}$ where IPS is the information processing speed in bits/s. Assuming IPS to be constant for any particular user, when $|\theta_{sub}| = 1$, $H_i = log_2(1) = 0$, so $VST - MT_{A_{rew}} = 0$. On other hand when $|\theta_{sub}|>1$ then $H_i>0$ , implying $VST - MT_{A_{rew}}>0$.

According to guided search theory, when the search set size is 1, the *exotic* tile should be recognized in the parallel search stage. The need for a serial search in the second phase is therefore not required [109, 23].

We provide an example, here, to show how plus points $PP$ and $\triangle^A_{VST}$ are calculated for 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 $\theta_{sub}$ and generates $VST$. The $VST$s, $s^h = \langle 3.8, 3.3, 1.7, 3.4, 1.4, 0.3 \rangle$

and $s^m = \langle 2, 3, 3.5, 4.5, 4, 1 \rangle$ are first arranged according to decreasing order of $|\theta_{sub}^{P_{t_r}}| = \langle 33, 26, 15, 12, 5, 1 \rangle$. $PP\ s^{h+} = \langle 5, 3, 2, 2, 1, 0 \rangle$ and $PP\ s^{m+} = \langle 1, 1, 1, 2, 1, 0 \rangle$ are then used to figure out $\triangle_{VST}^{A}$. $\triangle_{VST}^{A} = 0.867$ for human and $\triangle_{VST}^{A} = 0.4$ for bot. Experimentally the value of the authentication threshold $\alpha$ is set. In this scenario, setting $\alpha$ 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. Figure 4.3 shows how the success probability varies with the amount of *tolerance* $\wedge$ *condition*(2) in $\triangle_{VST}^{A}$.

**Algorithm 1** Calculate $\triangle_{VST}^{A}$

```
Input: Search set size |θ_sub^{P_tr}|, and a sequence of visual search time, s
Output: Accuracy metric, △_VST^A
Initialize △_VST^A = 0, s^+ = 0
Obtain sequence of observations s
Sort s according to |θ_sub^{P_tr}|
for k = 1 to n do
    for j = k to n − 1 do
        if s_k > s_{j+1} then
            s_k^+ = s_k^+ + 1
        end if
    end for
end for
Calculate △_VST^A = (Σ_{i=1}^{n} s_i^+) / (n(n−1)/2)
```

## 4.2 Movtcha Challenge generation

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 discussed in Section 4.1.3).

### 4.2.1 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) a guided search is invoked in human user and (2) the machine is not able to figure out the *exotic* tiles. We use images containing pencil or pen sketches or 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 [105]. Refer to Section 2.4.5 for a discussion on the factors affecting visual search. Sketches have traversing edges, or pencil strokes, which can be easily mimicked or modified by new random strokes. We refer to an edge or object that flows across a tile as a traversing edge or object of that tile. The images are first cropped to suitable sizes $x \times y$ and divided into a grid. For *most* tiles or cells $c_1,\ c_2, \ldots,\ c_{|\theta|}$, if the number of traversing edges is outside some certain interval $[E^V_{min},\ E^V_{max}]$, the image is discarded. This image selection process guarantees that *most* of the tiles contains at least $E^V_{min} = 2$ traversing edges so that any of them can be *modified* to form some random shapes. Moreover, the process limits the number of traversing edges to $E^V_{max} = k/3$ set based on researchers' aesthetic judgment. So that once *modified* the newly created strokes do not overwhelm a $(k \times k)$-sized *exotic* tile. Any image surviving such constraints is then referred to as the candidate image $I_C$.

### 4.2.2 Generation of Search Set

At each *instance* of Movtcha, an $I_C$ is selected to be tailored to generate $\theta_{sub}$. 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 sets differ by some random amounts at each *instance*. Larger offsets ensure sparser estimated $VST$, and results into higher $\triangle^A_{VST}$ by eliminating outliers (Section 4.4). We then randomly select a subset of tiles $\theta_{sub} \in \theta$. We refer to the boundary of each tile $t_i$ in $\theta_{sub}$ 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 edge detection algorithm [21]. 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

**Algorithm 2** Tailoring $I_C$'s for a Movtcha

Input: $\#instances$, search set size $|\theta|$, candidate images $I_C$'s
Output: Processed images $I_P$'s
Initialize $AOS = \lfloor \frac{|\theta|}{\#instances} \rfloor$
**for** $i = 1$ *to* $\#instances$ **do**
    *Consider* the candidate image $I_C^i$
    **if** $i \neq 1$ **then**
        $UL = i \times AOS$, $LL = UL - AOS + 1$
    **else**
        $LL = UL = 1$
    **end if**
    *Choose* size of $\theta_{sub}^i$ randomly from $[LL, UL]$
    *Choose* $\langle t_1,\ t_2, \ldots,\ t_{|\theta_{sub}^i|} \rangle = \theta_{sub}^i$ randomly from $\theta^i$
    **for** $k = 1$ *to* $|\theta_{sub}^i|$ **do**
        *Obtain* the edge continuation points in $t_k^{\{P\}}$.
        *Modify* $t_k$ until *some* edges disappear.
        *Connect* disappeared edges in $t_k^{\{P\}}$ using Bézier curves.
        Randomly *change* the intensities on all tiles
        *Place* a grid structure over the image.
        *Add* random noises.
    **end for**
    Challenge tile $t_c = SelectChallengeTile\ (LL, i, \theta_{sub}^i)$ Algorithm 3.
**end for**

dissolved by minimizing the intensity gradient difference. We then draw strokes connecting those points of disappeared edges randomly. These strokes are approximation curves drawn across $\{p_x^i,\ p_y^i\}$, with varying number of control points randomly set in the vicinity of the center of *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. 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 $I_C$. The image is presented to the user after it 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 $I_P$. Algorithm 2 and 3 provides overview on the search set generation process.

More specifically, 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 $b_{t_i}^{|\{P\}|/2}$. $\alpha$ is set to the min and $\beta$ 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 $b_{t_i}^{|\{P\}|/2}$. $\delta$ is set such that the following holds with some tolerance, $b_{t_i}^{|\{P\}|_{j-1}} > b_{t_i}^{|\{P\}|_j}$. We initially set $\delta$ to 20 and increment it if $b_{t_i}^{|\{P\}|_{j-1}} \not> b_{t_i}^{|\{P\}|_j}$. For each pair of edge points $\{p_x, p_y\}$ that have now disappeared due to decreasing the intensity gradient difference (in a random fashion), we draw new traversing edges with a $r$-pixel width stroke. That is a random stroke of $r$-pixel width 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 $\zeta$ is set as the difference between $s_1$ and the local $(3 \times 3)$ max gray value of $b_{t_i}^{\{p_e\}},\ e \in \{x, y\}$. $r$ is varied from 4-6. $minGrayIntensity(I_C)$ refers to the minimum gray value of the candidate iamge $I_C$ and *not* the processed image. 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 here before joining a pair of continuity points. In other words, a stroke can end abruptly somewhere inside the *exotic*

tile without joining the edge points. Due to the randomness at each step of the search set generation process, we argue that the modifications done are irreversible in Section 4.3.2.

### 4.2.3 Displaying an Instance

If there are $n$ *instances* in a Movtcha, $n$ processed images $\{I_P^1,\ I_P^2, \ldots,\ I_P^n\}$ are formed from $n$ candidate images, with $\{\theta_{sub}^1,\ \theta_{sub}^2,\ \ldots,\ \theta_{sub}^n\}$ as their corresponding search sets, where $|\theta_{sub}^1| < |\theta_{sub}^2| < \ldots < |\theta_{sub}^n|$. From each of these search sets the corresponding $t_c$ are selected s.t. $|\theta_{sub}^{1,\ P_{t_r}}| < |\theta_{sub}^{2,\ P_{t_r}}| < \ldots < |\theta_{sub}^{n,\ P_{t_r}}|$. 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.

**Algorithm 3** *SelectChallengeTile* $(LL, i, \theta_{sub}^i)$

```
Input: LL, i, θ^i_sub
Output: challenge tile t_{e_i}
if i = 1 then
    Set index e_i to LL
    Select t_{e_i} from θ^i_sub
    Return challenge tile t_{e_i}
else
    Choose e_i from (e_{i-1}, |θ^i_sub|]
    Return challenge tile t_{e_i}
end if
```

## 4.3 Security Analysis

We consider and analyze the success probability of an attacker in (1) Random Guessing Attack, (2) Automated Attack, (3) Position Inference Attack and (4) Static Relay attack. Section 4.4 provides a discussion on the experiments done on Relay attacks. Our security analysis assumes that it is hard for the attacker to access the implementation code which can be encrypted and obfuscated. Further discussion falls outside the scope of this thesis.

### 4.3.1 Random Attacks

An attacker (bot) is always able to perform a random guessing attack on Movtcha. It is able to drag and drop $t_c$ onto $t_r$ randomly. In other words, the attacker is able to come up with a random strategy whereby it selects one of the location in the grid and perform $A_{resp}$. We consider that the attacker is able to know the outcome of an *instance*, similar to a human user. Since the grid size is $\theta$, the success probability of the attacker making a correct *placement* at a particular *instance* is $\frac{1}{\theta}$ and since Movtcha is of $n$ *instances* the probability becomes $\frac{1}{\theta^n}$. Moreover, since at each of these *instances* the search set size will be varied, a successful attacker must therefore vary the search time accordingly, to mimic a human user. Since there are $n$ instances there should be $n!$ ways of varying the search time. The success probability thus becomes $\frac{1}{\theta^n \times n!}$ *without tolerance*. A grid size of $\theta = 48$, and where $n = 8$ *instances*, results in $8.8 \times 10^{-17}\%$.

### 4.3.2 Automated Attacks

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. (2) Perform $A_{resp}$ and $A_{rew}$ action at a desired speed while imitating 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* $|\theta_{sub}^{P_{tr}}|$. 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 probabilities for a real world CAPTCHA system security of 0.6% [119]. However, in real settings, we allow some *tolerance* on the $VST$ *trend* and subsequently on $\triangle_{VST}^{A}$ to accept *trends* with possibly a few outliers and meeting *condition(2)* (Section 4.1.3). This *tolerance* increases the attacker's success probability. However, *condition(2)* aids in rejecting trends where $VST$ is not the

least when search set size is 1. This slightly decreases the attacker's success probability. Figure 4.3 shows a simulation of how the success probability varies with $\triangle^{A}_{VST}$ for varying number of *instances*. On the other hand, *condition(2)* upper bounds the distance between a machine and a human in relay attack (Section 4.4.2, Experiment-IV). 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)* (Figure 4.4).

Movtcha inherently meets the guidelines for designing robust Captchas as discussed in [119]. The current challenge is always independent of the past challenges. Since object type (*exotic* tiles) are randomly generated, object recognition or classification is apparently a hard problem and will not work on Movtcha. Ideally, the same tile could have produced infinitely many random shapes, as it is processed each time. However, due to *memory* attack (discussed below), Movtcha presents unique image at each *instance*, again 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 $\theta_{sub}$ 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 Canny [21] 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 (Figure 4.1). So we sought to a customized boundary detection approach. 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). On the other hand, edge densities on opposite side of the boundaries remain 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 minimized the distance between actual edges in the image and edges found. Figure 4.1 shows contour plots of $I_P$s where intensity depth varies across $\theta$ providing no useful information to an attacker. On the other hand adding random noise before the image is presented prevents the attacker from exploiting any noise 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* (due to Claim 4.2) 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 $g$ and cut-and-scale essentially thwart a CBIR system like

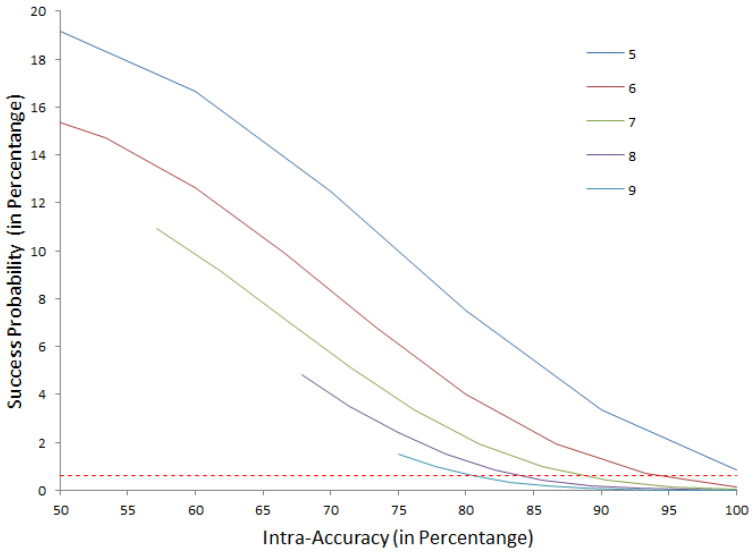


Figure 4.3: Success Probability varying with $\triangle^{A}_{VST}$ *#instances (5-9)*. Dotted line represents 0.6% attacker's success probability.

[46] in retrieving the original $I_C$ (as tested on 100 $I_P$'s).

**Claim 4.2** *A memory attacker can estimate* $|\theta^{P_{t_r}}_{sub}|$ *from at least the* $2^{nd}$ *instance onward when processed images,* $I_P$*'s are generated from the same candidate image,* $I_C$*.*

*Justification.* For $i = 1$ *instance* the *memory* attacker cannot accumulate any *memory*, which implies $|\theta^1| = |\theta^1_{sub}|$ for an attacker. For $i = 2$, a subset of tiles $\theta^2_{sub}$ is chosen out of $\theta^2$ resulting into either (1) $\theta^1_{sub} \cap \theta^2_{sub} = \emptyset$, (2) $\theta^1_{sub} \cap \theta^2_{sub} \neq \emptyset$, (3) $\theta^1_{sub} \subseteq \theta^2_{sub}$ if $\theta^1_{sub} < \theta^2_{sub}$ (4) $\theta^2_{sub} \subseteq \theta^1_{sub}$ if $\theta^2_{sub} < \theta^1_{sub}$ . We assume that the attacker is capable of figuring out the tiles that remain *almost* unchanged in both the *instances* i.e. (1) $\theta^2 \setminus (\theta^1_{sub} \cup \theta^2_{sub})$, (2) $\theta^2 \setminus (\theta^1_{sub} \cup \theta^2_{sub})$, (3) $\theta^2 \setminus \theta^2_{sub}$, (4) $\theta^2 \setminus \theta^1_{sub}$. Notice that the attacker is able to figure out exactly $\theta^2_{sub}$ in case (3). And since these cases are equally probable, it can be concluded that a *memory* attacker can estimate $|\theta^{P_{t_r}}_{sub}|$ from at least the $2^{nd}$ *instance* onward, when using the same image.

### 4.3.3 Position Inference Attack

Consider an attacker who infers the visual search time, $VST$, based on the location, $P_{t_r}$, of the target tile, $t_r$, inside the grid. The success probability of such attacker is $1/n!$, without any *tolerance*. We will now describe an inference attack in the best case scenario (for the

attacker). Consider a grid with rows = $r_1,\ r_2,\ \ldots,\ r_n$ and columns = $c_1,\ c_2,\ \ldots,\ c_n$. We refer to our challenge tile selector as $S$. Recall, that a challenge tile, $t_c$, is basically a copy of the target tile, $t_r$ inside the grid or image. We assume that $S$ selects $n$ challenge tiles such that $t_{c_i}$ is a copy of a target tile, $t_{r_i}$ from row $r_i$, $i = 1 \ldots n$. Therefore, a set of *instances* is formed with search set sizes, $|\theta_{sub}^{P_{t_{r_1}}}| < |\theta_{sub}^{P_{t_{r_2}}}| \ldots < |\theta_{sub}^{P_{t_{r_n}}}|$ and presented to the user in that order. At this point, if the position inference attacker infers the $VST$ according to the location of the target tile, the resulting $VST$ series, $VST^{P_{t_{r_1}}} < VST^{P_{t_{r_2}}} \ \ldots < VST^{P_{t_{r_n}}}$, would result in an intra-accuracy, $\triangle_{VST}^{A} = 1$. However, the probability that the challenge tile selector $S$, selects $t_{r_i}$ from row $r_i$ is $1/(n-i+1)$. For example, the probability that the inference attacker chooses a target tile from the first row when search set size is 1 is $1/n$ where $n$ is the number of total rows. That means for $n$ *instances* the success probability of an inference attack (in the best case) is $1/n!$, without considering *tolerance*.

#### 4.3.4 Static 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. We assume that the relay bot does not collude with any other automated bot. In a relay attack, the bot needs to consider three time intervals: (1) The communication delay between the bot and human solver's machine, (2) the time taken by the human to solve the challenge, (3) the time needed for the bot to solve the actual Captcha. Similarly, in our case, the bot needs to consider four time intervals for each *instance*, (1) The communication delay between the bot and the human solver's machine $\triangle_t^c$, (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 [69] 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 $t+k,\ k{>}0$. 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 $t_c$. We carried out an experiment simulating a relay attack scenario to prove our hypothesis that Movtcha is resistant to static relay attack. We carried out a small scale experiment (Section 4.4, Experiment-IV) with 10 users from the $2^{nd}$ pool (Experiment-II) to examine our hypothesis. Each assignment was worth $0.3.

## 4.4 Experiments And Results

We carried out four experiments to evaluate Movtcha in terms of (1) Design and presentation, (2) Accuracy and Usability (With varying number of *instances*) and (3) Accuracy and Usability (Solving a Movtcha *once* for the first time), (4) Resistance against Relay Attacks. 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. Participants who agreed to the consent information were allowed to proceed and play.

### 4.4.1 Experiment Setup

**Experiment I: Design & 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 $\lambda$ for $C3$ and $\zeta$ for $C4$. $C3$ and $C4$ were therefore not set in this experiment. The goals required users to solve Movtcha in a controlled condition i.e. in a non-distracting environment using a single platform. The experiment consisted of a pool of 10 graduate students. All of them used a PC with 2.10 GHz Intel i3, 4GB RAM and an wireless optical USB mouse. They used Google Chrome on a screen of resolution 1366 $\times$ 768 (96 PPI) in Windows 7 SP1 OS.

A short video showed how the game is played at the beginning. Participants received no

further instructions. Each participants were required to solve 9 Movtchas $\langle M_1,\ M_2, \ldots,\ M_9 \rangle$ each comprising of a fixed number of instances ($\#instances = 8$). The 9 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 $60 \times 60$ 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. $t_c$ moved randomly inside the bounding box at 0.1 pixels/frame @60FPS i.e. 6 pixels/s.

**Experiment II: Accuracy & Efficiency (With varying number of *instances*).**
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.

A HIT was created with 50 available assignments to have 50 unique users. Users must have $\geq 98\%$ HIT approval rate and $\geq 5000$ HITs approved to qualify for our HIT. They were required to complete 15 Movtchas with (1) fixed image size, $8 \times 6$, (2) fixed parameters $\zeta$, $\lambda$ determined from Experiment-I. They can make a maximum of 3 mistakes happening due to $C2 - C5$ while solving a Movtcha. Three separate internal assignments were used to vary the number of *instances* from 6-8. For each internal assignment, they were required to solve 5 Movtchas. Therefore, as a whole they were required to complete $(5 \times 3) = 15$ Movtchas. On submitting our generated code back to Amazon they were paid \$0.7.

**Experiment III: Accuracy & Efficiency (Solving a Movtcha *once* for the first time).**
The goal of this experiment was to observe how random users perform when solving only *one* Movtcha. Two HITs were created on two separate occasions. The first HIT was created with 70 assignments to have 70 unique users. The second HIT was created with 100 assignments to have 100 unique users. We made sure that users from the first pool did not submit any assignments in the second HIT. We also made sure that none of the users from Experiment-

II participated in these HITs. Users must have $\geq$ 98% HIT approval rate and $\geq$ 1000 HITs approved to qualify for our HITs. The users 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. The first HIT presented a $8 \times 6$ Movtcha (60-pixel tile size) and the second HIT presented a $7 \times 7$ Movtcha (70-pixel tile size). Movtchas from the second HIT, therefore required more space and demanded more mouse movements from users. There was one demo *instance* at the beginning which was not considered in accuracy calculation. They were then required to copy-paste a code (generated on our website) back to Amazon to get paid \$0.15. Since we were interested to observe users' performance when they are solving Movtcha for the first time, we did not allow them to repeat.

### 4.4.2 Experimental Results and Analysis

**Experiment I: Results and Analysis.**

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. Noises fail to affect the $\triangle^A_{VST}$, when $VST$s are sparser. Similar results can be observed in Neisser [73] where sparser target positions $P_{t_r}$ results into $VST$ with larger offsets. Therefore, in order to have larger $\triangle^A_{VST}$, Experiment-II and III were carried out using images with $8 \times 6$ 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*. 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. The average

$I_d^{C5}$ remained as low as 0.03 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 $\zeta = 0.2s$ of $C4$ to possible extreme outliers using interquartile range $PT_{A_{rew}}^{Q_3} + (3 \times PT_{A_{rew}}^{IQR})$, to allow some tolerance in non-controlled condition. The ratio of $VST$ to $|\theta_{sub}^{P_{t_r}}|$ is the *true* inspection time, $IPT$, for each *exotic* tile. The average inspection time of all the participants throughout the 3 *sessions* was $\approx$ 113 ms. The parameter $\lambda$ of $C3$ is similarly set to extreme outliers values for non-controlled conditions s.t. $\lambda = (|\theta| * IPT') + MT'_{rew}$ where $IPT' = IPT^{Q_3} + (3 \times IPT^{IQR}) \approx 0.3s$ and $MT'_{rew} = MT_{rew}^{Q_3} + (3 \times MT_{rew}^{IQR}) \approx 2.0s$ for all *instances* with varying $|\theta_{sub}^{P_{t_r}}|$. Therefore, $\lambda$ 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 e.g. $\approx$ 15 seconds for $G_3$.

Table 4.1: Results from Experiment-I

| | $\triangle_{VST}^{A}$ | $I_d^{C2}$ | $T_{Com}$(Std) | $E_c$ | $PT_{A_{rew}}$ |
|---|---|---|---|---|---|
| G1 | 56.8(6.7) | 0.13 | 19.2(1.3) | 0.07 | 82.1 |
| G2 | 75.5(3.0) | 0.05 | 23.5(2.2) | 0.06 | 43.2 |
| G3 | 84.2(1.9) | 0.06 | 26.4(2.8) | 0.03 | 65.7 |

**Experiment II: Results and Analysis.**

**Results.** It can be observed that the average $\triangle_{VST}^{A}$ 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 $\triangle_{VST}^{A}$ is larger for increasing number of *instances* (Figure 4.3). This resulted into higher $\triangle_{M}^{A}$ for increasing number of *instances*. The $\triangle_{VST}^{A}$ drops quickly with a fixed tolerance for less number of *instances*. In other words, Movtcha can tolerate more noises in the plus points series as the number of *instances* increase. The average $I_d$ is higher relative to Experiment-I. Activation of $C4$ led to relatively higher click error, $E_c$. It implies that the

constraints proved to be useful in discarding abnormal $VST$ and $MT_{A_{rew}}$ that might have resulted from distractions. 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 82% (Gmail's Captcha accuracy rate 83% [119]). Mouse type statistics from exit survey include wireless/wired mouse (72%), laptop touchpad (28%). Recall, $VST$ is calculated using subtraction method [33] which allows $VST$ to self adjust for the user's specific environment.

Table 4.2: Results from Experiment-II

| $inst$ | $\triangle^A_{VST}$(Std) | $T_{Com}$(Std) | $I_d^{C2}$ |
|---|---|---|---|
| 6 | 80.5(4.9) | 22.84(6.14) | 0.21 |
| 7 | 81.0(4.5) | 27.62(6.72) | 0.07 |
| 8 | 84.9(6.1) | 32.53(7.5) | 0.09 |

**Experiment III: Results and Analysis.**

The average *intra-accuracy*, $\triangle^A_{VST}$, is 78.2% (std 6.03%). When $\triangle^A_{VST}$ is set at $\geq 75$, 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 $38.04s$ (std $8.63s$). 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 minutes.

For the $2nd$ HIT we collected 72 valid submissions until the HIT expired. The average *intra-accuracy*, $\triangle^A_{VST}$, is 76.77% (std 8.89%). When $\triangle^A_{VST}$ is set at $\geq 71.43$, limiting the success probability of bot to 3.52% the *inter-accuracy* is $\triangle^A_M \approx 80.6\%$. The average time to complete $T_{com}$, is longer compared to the $1^{st}$ HIT. This is expected since we increased the grid size by a factor of 1.4 relative to the $1^{st}$ HIT. This resulted into longer mouse movement time during $A_{resp}$ and $A_{rew}$ actions. These results are reported from unique users solving just *one* Movtcha for the first time. The average time required to complete each assignment was 9.3 minutes. These results suggest that Movtcha is independent of language, culture and experience. It can be easily used by naive users with high success rate.

The accuracy difference between Experiment-II and Experiment-III shows that users

tend to perform well once they get accustomed to the system. Nevertheless, the accuracies obtained from Experiment-III are encouraging, provided that the users are solving Movtcha for the first time.

**Experiment IV: Relay Attack Results.**

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 $t+k,\ k{>}0$ [69]. 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. The users selected had relatively better visual search skill and less error rates. 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. As soon as $s_i$ is presented with the stimulus the users performed search and clicked on $t_r$ and then $C_{t_c}$ consecutively. If $P_{t_c} \neq C_{t_c}$ then 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 error rate per click resulting into longer $VST$ and $C3$ activations. None of them 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 human solver.

**Simulating Movtcha Solving Activities.**

We wanted to observe users' search behavior when they solve Movtcha. We developed a web-based program using HTML5 and JS capable of simulating any user's Movtcha solving activity once fed with data collected during the previous solving attempts. We observed the simulations of randomly picked users. The simulations suggest that the users perform "drag

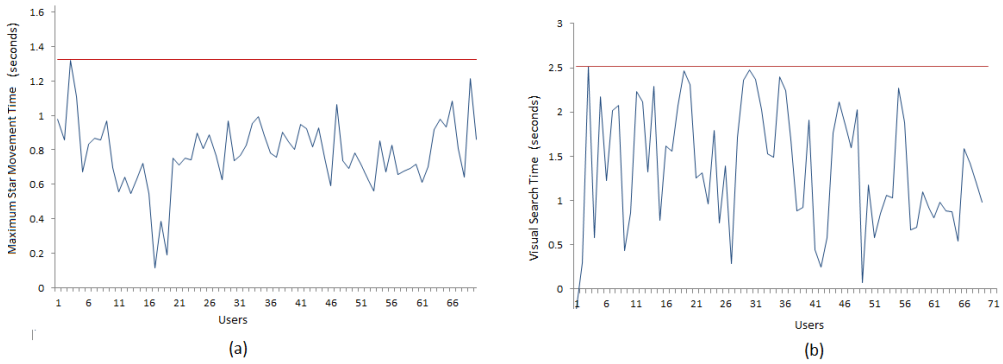


Figure 4.4: (Experiment-III, $1^{st}$ HIT)(a) Shows the maximum $MT_{A_{rew}}$ of successful users out of all the *instances* they played (b) Shows $VST$ when $|\theta_{sub}| = 1$. If we restrict $MT_{A_{rew}}$ and $VST$ ($|\theta_{sub}| = 1$) to $1.4s$ and $2.5s$ respectively as a constraint, we can 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 $3.9s$ 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 2.1s and 2.8s 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 scheme essentially limits the solving time of that Captcha.

and search" actions while solving Movtcha. The constraints we discussed in Section 4.1.1 encourages them to search serially. We also observed phenomenon when user recognizes (by parallel search) the *exotic* tile almost immediately when search set size is 1.

## 4.5 Discussion on Movtcha

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 discussed how image selection, challenge generation and response evaluation are automatically done by our system. Movtcha maintains real world security while presenting *clear* answers to challenges. This attribute makes Movtcha language, culture and experience independent.

The average SUS score for all the experiments are within the user-friendly industrial

software ratings [63]. $SUS$ ratings were relatively higher in Experiment-I compared to Experiment-II. Considering Experiment-II, 60% of the participants agreed that the game was fun to play and 82% felt it was easy and intuitive. This demonstrates that Movtcha is a user-friendly system.

Due to the modification done on Movtcha, some images might look unpleasant to the user. This might happen to any image-based Captcha which undergo modifications. For example in Cortcha portions of the image is cut off from the actual image. A solution in such cases, would be to focus on images with objects only and not include humans or pet animals.

Table 4.3: Notations lookup

| **Symbols-I** | **Description** |
|---|---|
| $A_{resp}$ | The action where challenge tile is dragged onto the target tile |
| $A_{rew}$ | The action where the star is dragged back to $P_{t_c}$ |
| $E_c$ | The average error rate per click. Refer to Experiment-I (Results) |
| $[E^V_{min}, E^V_{max}]$ | Upper and lower limit on the number of traversing edges |
| $g$ | The grid/image |
| $G1,\ G2,\ \&\ G3$ | These are the three sessions in Experiment-I |
| $I_d$ | The ratio of the #instances discarded to the total #instances |
| $I_C$ | Candidate Image |
| $I_P$ | Processed image generated from a candidate image $I_C$ |
| $IPT$ | Inspection time of each tile |
| $MT_{A_{rew}}$ | The movement time (dragging time) of the star ($A_{rew}$ action) |
| $P_{t_c}$ | Latest position of the challenge tile |
| $PT_{A_{rew}}$ | Total paused time during $A_{rew}$ |
| $s_r$ | The rewarded star for a successful match |
| $t_c$ | Challenge tile |
| $t_r$ | Target tile inside the grid |
| $t^t_{RT}$ | Time elapsed between the appearance of the tile and the user picking it up. |
| $VST$ | Visual Search time |
| **Symbols-II** | **Description** |
| $\alpha$ | Intra-accuracy must cross this threshold for user to get recognized as human |
| $\triangle^A_M$ | The ratio of number of solved Movtchas to the number of Movtcha challenges |
| $\triangle^A_{VST}$ | Intra-accuracy metric used for differentiating human from machine |
| $\zeta$ | Constraint $C4$ : Threshold limiting the amount of pauses during $A_{rew}$ |
| $\theta$ | Set containing all the tiles |
| $\theta_{sub}$ | Search set containing only *exotic* tiles |
| $\lambda$ | Constraint $C3$ : Threshold limiting the time elapsed during $A_{resp}$ action |
| **Constraints** | **Tiny Description** |
| C2 | Triggers when missed/skipped over the target tile while searching |
| C3 | Triggers on lazy or distracted search. Determining parameter: $\lambda$ |
| C4 | Triggers on too much pauses while dragging star. Determining parameter $\zeta$ |
| C5 | Triggers when challenge tile is dropped anywhere other than its destination |

# Chapter 5

# CONCLUSIONS AND FUTURE WORK

## 5.1 Thesis Summary

This thesis presents new approaches to user authentication systems and Captcha systems. We use human cognitive abilities, in particular visual search ability, working memory and priming effect on unconscious automatic processing to design and develop a user identification system. Our system is able to estimate the "cognitive signature" generated by the user. We use a mouse as the input device to our system. This means our system, in addition to capturing the cognitive features, can be used to collect other mouse dynamic features.

Our second proposed system, Movtcha, uses the guided search theory to distinguish a human from a computer. Using cognitive feature in a Captcha system eliminates the need of providing the challenge in a "cipher form". In other words, we provide a *clear form* Captcha, where the challenge and the response are exact copies of each other. Therefore, users irrespective of their language, experience, and culture can solve Movtcha.

In the following Sub-Sections, we discuss our contributions.

### 5.1.1 A Novel Approach to User Authentication

Authentication is one of the most important entities of any digital system. Although, the most widely used form of authentication in today's computer systems is password, it has the downside of being stolen, lost or forgotten. Biometric systems liberate the users from the cognitive burden of memorizing secrets or passwords. Biometric systems are also immune to lost or theft. However, behavioral biometrics like mouse or keyboard is rarely used for static authentication. This is mainly due to their large enrollment and verification time.

We present an authentication system which collects and uses human cognitive features

to identify the users. Our work focuses on individual difference in cognitive abilities or essentially "what an individual is *capable of doing cognitively*" and could potentially fall under a new category of biometric. We designed and implemented a web-based game which can invoke the required mental processes. Our system reaches an FAR of 0-2.3% and an FRR of 0-11.6%) comparable to other state-of-the-art behavioral biometric systems. We carried out impersonation attacks against our system. Our empirical results suggest that cognitive features are difficult to reproduce.

### 5.1.2 A Novel Approach to Captcha System

Captchas are necessary to safeguard systems from the intrusion of automated computer programs. Although text-based Captchas are the most popular form of Captchas, many of the proposed or deployed ones were broken with high success rate. Some image-based and game-based Captchas utilizing semantic relationship between images or objects are considered more secure and usable. However, they cannot auto-generate the challenge database.

Movtcha is presented to the user as a game. It estimates the visual search time of the user and uses it to distinguish a human from a computer. Movtcha takes into account human behavioral analysis to eliminate noise during feature estimation process. The use of cognitive and behavioral features allows Movtcha to present its challenge and response both in *clear form*. All existing Captcha systems would fail if the response to a challenge is provided to the user. In other words, "clear form" is a scenario where the challenge and response are exact copies of each other and both are available to the client and client's machine [69]. *Clear form* Captchas are language, culture and experience independent. It can be used by people all across the world. We evaluated our system against random, automated, inference and static relay attacks. In addition, to being resistance to the aforementioned attacks, Movtcha can auto-generate the challenge database and auto-evaluate the users' responses. Therefore, it can be deployed in large-scale applications.

## 5.2 Future Works

In this section, we provide the future attainable goals for both the systems. We also discuss some open problems.

### 5.2.1 User Authentication System

Individual difference in cognitive abilities is not only confined within our three aforementioned cognitive processes. It also includes several others, such as visualization (mentally manipulating forms to visualize how they would look), verbal comprehension (understanding words, sentences) [53]. Multiple cognitive abilities can be utilized in developing an authentication scheme. This would allow more cognitive features to be estimated and thus enhance the overall performance of the system. For instance, a simple modification in our game by occasionally including negative search set, where the target tile is not present, would invoke exhaustive visual search.

We have manually drawn the search sets that are presented to the user in the enrollment and verification phase. Our future work will involve automating this search set generation stage. A good source of randomness is needed to draw such shapes without any criss-cross and ultimately forming the desired shapes (Chapter 3, Figure 3.2).

One can estimate various features related to mouse dynamics from our system such as distances, horizontal velocity, vertical velocity, tangential velocity, tangential acceleration, tangential jerk, angular velocity, drag and drop, point and click and others. In future work, we will incorporate some of these mouse dynamics such as angles and ratios which are considered environment-independent. It will be interesting to find out how the performance of the system varies with and without the mouse dynamic features.

### 5.2.2 Movtcha

Our future goal will focus on increasing the accuracy and usability of Movtcha. One way to enhance the accuracy of the system is by including mouse dynamics and mouse events for differentiating humans from machines. This is an open area and demands further scrutiny. Including additional features for making an authentication decision can reduce the number of *instances* that needs be played. This would significantly reduce Movtcha solving time. However, such reduction should not increase the attacker's success probability. In future, we are also going to provide a mathematical relationship between the intra-accuracy $\triangle^{A}_{VST}$ and success probability of the attacker with *tolerance*.

It is worthwhile to find appropriate images that invoke guided search in the user. Recall, that guided search consists of two consecutive stages, a parallel and a serial search stage. A relationship between the amount of *activations* of the *exotic* tiles and its effect on parallel search can be established experimentally. Finding an alternative way of invoking guided search is also an open problem.

# Bibliography


[1] Adams, J. A. Human factors engineering. *Macmillan Publishing Co, Inc*, 1989.

[2] Ahmed, A., and Traore, I. A new biometric technology based on mouse dynamics. *Dependable and Secure Computing, IEEE Transactions on 4*, 3 (July 2007), 165–179.

[3] Are you a human. http://www.areyouahuman.com/. Last accessed on 23/09/2014.

[4] Baddeley, A. D. Human memory: Theory and practice. *Psychology Press.*, 1997.

[5] Baddeley, A. D. Working memory. *Science*, 255(5044), 556-559, 1992.

[6] Baird, H. S., & Popat, K. Human interactive proofs and document image analysis. *In Document Analysis Systems V* (pp. 507-518). Springer Berlin Heidelberg, 2002.

[7] Ballard, L., Lopresti, D., and Monrose, F. Forgery quality and its implications for behavioral biometric security. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 37*, 5 (2007), 1107–1118.

[8] Ballard, L., Monrose, F., and Lopresti, D. Biometric authentication revisited: Understanding the impact of wolves in sheep?s clothing. In *15th Annual USENIX Security Symposium* (2006), pp. 29–41.

[9] Bargh, J. A. Conditional automaticity: Varieties of automatic influence in social perception and cognition. *Unintended thought 3* (1989), 51–69.

[10] Bargh, J. A., Chen, M., and Burrows, L. Automaticity of social behavior: Direct effects of trait construct and stereotype activation on action. *Journal of personality and social psychology 71*, 2 (1996), 230.

[11] Bergadano, F., Gunetti, D., & Picardi, C. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4), 367-397, 2002.

[12] Bojinov, H., Sanchez, D., Reber, P., Boneh, D., and Lincoln, P. Neuroscience meets cryptography: designing crypto primitives secure against rubber hose attacks. In *Proceedings of the 21st USENIX Security Symposium* (2012).

[13] Bolle, R. Guide to biometrics. *Springer*, 2004.

[14] Bonneau, J. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2012), SP '12, IEEE Computer Society, pp. 538–552.

[15] Bours, P., and Fullu, C. J. A login system using mouse dynamics. In *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP'09. Fifth International Conference on* (2009), IEEE, pp. 1072–1077.

[16] Bowman, A. W. An alternative method of cross-validation for the smoothing of density estimates. *Biometrika 71*, 2 (1984), 353–360.

[17] Brooke, J.: Sus-a quick and dirty usability scale. Usability evaluation in industry 189, 194 (1996)

[18] Bromme, A. A classification of biometric signatures. In *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 3 (ICME '03) - Volume 03* (Washington, DC, USA, 2003), ICME '03, IEEE Computer Society, pp. 17–20.

[19] Bursztein, E., & Bethard, S. Decaptcha: breaking 75% of eBay audio CAPTCHAs. *In Proceedings of the 3rd USENIX conference on Offensive technologies* (p. 8). USENIX Association, 2009.

[20] Bursztein, E., Beauxis, R., Paskov, H., Perito, D., Fabry, C., & Mitchell, J. The failure of noise-based non-continuous audio captchas. *In Security and Privacy (SP)*, 2011 IEEE Symposium on (pp. 19-31). IEEE, 2011

[21] Canny, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,*, (6), 679-698, 1986.

[22] Cappelli, R., Ferrara, M., Franco, A., & Maltoni, D. Fingerprint verification competition 2006. *Biometric Technology Today*, 15(7), 7-9, 2007.

[23] Cave, K. R., & Wolfe, J. M. Modeling the role of parallel processing in visual search. *Cognitive psychology*, 22(2), 225-271, 1990.

[24] Chew, M., Tygar, J.D. Image recognition captchas. *Springer* (2004)

[25] Chiang, A., and Atkinson, R. C. Individual differences and interrelationships among a select set of cognitive skills. *Memory & Cognition 4*, 6 (1976), 661–672.

[26] Colombi, J. M., Ruck, D. W., Anderson, T. R., Rogers, S. K., & Oxley, M. Cohort selection and word grammar effects for speaker recognition. *In Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference*, on (Vol. 1, pp. 85-88), IEEE, 1996.

[27] Craik, F. I., and Salthouse, T. A. The handbook of aging and cognition. *Psychology Press*, 2011.

[28] Daugman, J. How iris recognition works. *Circuits and Systems for Video Technology*, IEEE Transactions on, 14(1), 21-30, 2004.

[29] 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.

[30] Delac, K., & Grgic, M. A survey of biometric recognition methods. *In Electronics in Marine, 2004. Proceedings Elmar 2004. 46th International Symposium* (pp. 184-193), IEEE, 2004, June.

[31] Della Sala, S., Gray, C., Baddeley, A., Allamano, N., & Wilson, L. Pattern span: a tool for unwelding visuospatial memory. *Neuropsychologia*, 37(10), 1189-1199, 1999.

[32] Denning, T., Bowers, K., van Dijk, M., and Juels, A. Exploring implicit memory for painless password recovery. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), ACM, pp. 2615–2618.

[33] Donders, F. Over de snelheid van psychische processen. onderzoekingen gedaan *in het physiologisch laboratorium der utrechtsche hoogeschool* 1868; 30: 412–431. 1868–1869, tweede reeks, ii: 92–120. reprinted as donders, franciscus c.(1969). on the speed of mental processes. Acta Psychologica

[34] Dovidio, J. F., and Gaertner, S. L. Stereotyping, prejudice, and discrimination: Spontaneous and deliberative processes. *Paper presented at the meeting of the Society of Experimental Social Psychology, Washington, DC* (1995, October).

[35] Elson, J., Douceur, J.R., Howell, J., Saul, J. Asirra: a captcha that exploits interest-aligned manual image categorization. *ACM Conference on Computer and Communications Security.* pp. 366–374 (2007)

[36] Esp-pix. http://server251.theory.cs.cmu.edu/cgi-bin/esp-pix/esp-pix

[37] Fazio, R. H., Jackson, J. R., Dunton, B. C., and Williams, C. J. Variability in automatic activation as an unobtrusive measure of racial attitudes: a bona fide pipeline? *Journal of personality and social psychology 69*, 6 (1995), 1013.

[38] Frank, M., Biedert, R., Ma, E., Martinovic, I., & Song, D. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *Information Forensics and Security*, IEEE Transactions on, 8(1), 136-148.

[39] Fry, A. F., and Hale, S. Relationships among processing speed, working memory, and fluid intelligence in children. *Biological psychology 54*, 1 (2000), 1–34.

[40] Gaines, R. S., Lisowski, W., Press, S. J., and Shapiro, N. Authentication by keystroke timing: Some preliminary results. *Tech. rep., DTIC Document*, 1980.

[41] Galotti, K. M. Cognitive Psychology In and Out of the Laboratory. *SAGE Publications, Inc*, 2013.

[42] Galton, F. Inquiries into Human Faculty and Its Development. *Kessinger Publishing, LLC*, 2010.

[43] Galton, Francis. I. Statistics of mental Imagery. *Mind* 19 (1880): 301-318.

[44] Gamboa, H., and Fred, A. A behavioral biometric system based on human-computer interaction. *Proc. SPIE 5404* (2004), 381–392.

[45] Gee, J. P. What video games have to teach us about learning and literacy. *Computers in Entertainment (CIE) 1*, 1 (2003), 20–20.

[46] Google search by image. http://images.google.com/imghp?hl=en. Last accessed on 23/09/2014.

[47] Gossweiler, R., Kamvar, M., & Baluja, S. What's up CAPTCHA?: a CAPTCHA based on image orientation. *In Proceedings of the 18th international conference on World wide web* (pp. 841-850). ACM, 2009.

[48] Guyon, I., & Elisseeff, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3, 1157-1182, 2003.

[49] Hall, P., and Marron, J. S. Extent to which least-squares cross-validation minimises integrated square error in nonparametric density estimation. *Probability Theory and Related Fields 74*, 4 (1987), 567–581.

[50] Hamdy, O., and Traoré, I. Homogeneous physio-behavioral visual and mouse-based biometric. *ACM Transactions on Computer-Human Interaction (TOCHI) 18*, 3 (2011), 12.

[51] Haxby, J. V., Ungerleider, L. G., Horwitz, B., Rapoport, S. I., & Grady, C. L. Hemispheric differences in neural systems for face working memory: A PETrCBF study. *Human Brain Mapping*, 3(2), 68-82, 1995.

[52] Hick, W. E. On the rate of gain of information. *Quarterly Journal of Experimental Psychology 4*, 1 (1952), 11–26.

[53] Horn, J. L. Cognitive diversity: A framework of learning. *WH Freeman/Times Books/Henry Holt & Co*, 1989.

[54] Jarrold, C., and Towse, J. N. Individual differences in working memory. *Neuroscience 139*, 1 (2006), 39–50.

[55] Jensen, A. R. *Individual differences in the Hick paradigm.* Ablex Publishing, 1987.

[56] Jensen, A. R. Why is reaction time correlated with psychometric g? *Current Directions in Psychological Science* (1993).

[57] Jones, M. C., Marron, J. S., and Sheather, S. J. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association 91*, 433 (1996), 401–407.

[58] Jorgensen, Z., and Yu, T. On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the 6th ACM Symposium on Information, Computer and*

*Communications Security* (New York, NY, USA, 2011), ASIACCS '11, ACM, pp. 476–482.

[59] Kale, A., Sundaresan, A., Rajagopalan, A. N., Cuntoor, N. P., Roy-Chowdhury, A. K., Kruger, V., & Chellappa, R. Identification of humans using gait. *Image Processing, IEEE Transactions on*, 13(9), 1163-1173, 2004.

[60] Kung, S. Y., Mak, M.-W., and Lin, S.-H. *Biometric authentication: a machine learning approach*. Prentice Hall Professional Technical Reference, 2005.

[61] Lee, L., & Grimson, W. E. L. Gait analysis for recognition and classification. *In Automatic Face and Gesture Recognition, 2002. Proceedings*. Fifth IEEE International Conference on (pp. 148-155). IEEE, May 2002.

[62] Leiserson, C. E., Rivest, R. L., Stein, C., and Cormen, T. H. *Introduction to algorithms*. MIT press, 2001.

[63] Lewis, J.R., Sauro, J. The factor structure of the system usability scale. *Human Centered Design*, pp. 94–103. Springer (2009)

[64] Liu, Y. Interactions between memory scanning and visual scanning in process monitoring. *Ann Arbor 1001*, 48109–2117 (1995)

[65] McAndrew, A. An introduction to digital image processing with matlab. *An introduction to digital image processing with matlab notes for scm2511 image processing. school of computer science and Mathematics, Victoria university of technology*, 1-264, 2004.

[66] McCarney, R., Warner, J., Iliffe, S., van Haselen, R., Griffin, M., & Fisher, P. The Hawthorne Effect: a randomised, controlled trial. *BMC medical research methodology*, 7(1), 30. (2007).

[67] Mechanical turk. https://www.mturk.com/mturk/welcome. Last accessed on 19/11/2014

[68] Miller, L. T., and Vernon, P. A. Intelligence, reaction time, and working memory in 4-to 6-year-old children. *Intelligence 22*, 2 (1996), 155–190.

[69] 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. *Proceedings of the 9th ACM symposium on Information, computer and communications security.* pp. 195–206. ACM (2014)

[70] Moy, G., Jones, N., Harkless, C., and Potter, R. Distortion estimation techniques in solving visual captchas. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–23. IEEE, 2004.

[71] Nakkabi, Y., Traoré, I., and Ahmed, A. A. E. Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 40*, 6 (2010), 1345–1353.

[72] Neisser, U., and Lazar, R. Searching for novel targets. *Perceptual and Motor Skills 19*, 2 (1964), 427–432.

[73] Neisser, U. Decision-time without reaction-time: Experiments in visual scanning. *The American Journal of Psychology* pp. 376–385 (1963)

[74] Nucaptcha: Adaptive captcha authentication. http://www.nucaptcha.com/. Last accessed on 11/06/2014.

[75] Paint: http://windows.microsoft.com/en-ca/windows7/products/features/paint. Last accessed on 11/12/2014.

[76] PaperJS: http://paperjs.org/ Last accessed on 11/13/2014.

[77] Phillips, P. J., Scruggs, W. T., OToole, A. J., Flynn, P. J., Bowyer, K. W., Schott, C. L., & Sharpe, M. User authentication through keystroke dynamics. *FRVT 2006 and ICE 2006 large-scale results. National Institute of Standards and Technology*, NISTIR, 7408, 2007.

[78] Prabhakar, S., Pankanti, S., & Jain, A. K. Biometric recognition: Security and privacy concerns. *IEEE Security & Privacy*, 1(2), 33-42, 2003.

[79] Pusara, M., and Brodley, C. E. User re-authentication via mouse movements. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* (2004), ACM, pp. 1–8.

[80] Revett, K. Behavioral biometrics: a remote access approach. *John Wiley & Sons*, 2008.

[81] Revett, K., Jahankhani, H., de Magalhães, S. T., and Santos, H. M. A survey of user authentication based on mouse dynamics. In *Global E-Security*. Springer, 2008, pp. 210–219.

[82] Ross, S. A., Halderman, J. A., & Finkelstein, A. Sketcha: a CAPTCHA based on Line Drawings of 3D Models. *In Proceedings of the 19th international conference on World wide web* (pp. 821-830). ACM, 2010.

[83] Ross, A., & Jain, A. Information fusion in biometrics. *Pattern recognition letters*, 24(13), 2115-2125, 2003.

[84] Rudemo, M. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics* (1982), 65–78.

[85] Scholz, F. Maximum likelihood estimation. *Encyclopedia of Statistical Sciences* (1985).

[86] Shanmugapriya, D., and Padmavathi, G. A survey of biometric keystroke dynamics: Approaches, security and challenges. *arXiv preprint arXiv:0910.0817* (2009).

[87] Sheskin, D. J. Parametric and nonparametric statistical procedures. *Boca Raton: CRC.*, 2000.

[88] Sonka, M., Hlavac, V., & Boyle, R. Image processing, analysis, and machine vision. *Cengage Learning.*, 2014

[89] Stager, P., & Angus, R. Locating crash sites in simulated air-to-ground visual search. *Human Factors: The Journal of the Human Factors and Ergonomics Society* (1978), 20(4), 453-466.

[90] Starovoitov, V. V., Samal, D. I., & Briliuk, D. V. Three approaches for face recognition. *In The 6-th International Conference on Pattern Recognition and Image Analysis, Velikiy Novgorod, Russia* (pp. 707-711), October, 2002.

[91] Sternberg, R. J. *Cognitive Psychology.* Cengage Learning, 2011.

[92] Stuart, A., & Phillips, D. P. Word recognition in continuous and interrupted broadband noise by young normal-hearing, older normal-hearing, and presbyacusic listeners. *Ear and hearing*, 17(6), 478-489, 1996.

[93] Tapscott, D. *Grown up digital: How the net generation is changing your world HC.* McGraw-Hill, 2008.

*Perception & Psychophysics 66*, 6 (2004), 953–962.

[94] Thorpe, S., Fize, D., Marlot, C., et al. Speed of processing in the human visual system. *Nature* 381(6582), 520–522 (1996)

[95] Townsend, J. T., and Fifić, M. Parallel versus serial processing and individual differences in high-speed search in human memory. *Perception & Psychophysics* 66.6 (2004): 953-962

[96] Van Zandt, T., Townsend, J.T. Self-terminating versus exhaustive processes in rapid visual and memory search: An evaluative review. *Perception & Psychophysics* 53(5), 563–580 (1993)

[97] Vikram, S., Fan, Y., Gu, G. Semage: a new image-based two-factor captcha. *Proceedings of the 27th Annual Computer Security Applications Conference.* pp. 237–246. ACM (2011)

[98] Villani, M., Tappert, C., Ngo, G., Simone, J., Fort, H. S., and Cha, S.-H. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on* (2006), IEEE, pp. 39–39.

[99] Virtual sweatshop. http://krebsonsecurity.com/2012/01/virtual-sweatshops-defeat-bot-or-not-tests/. Last accessed on 23/09/2014.

[100] Vogel, E. K., and Machizawa, M. G. Neural activity predicts individual differences in visual working memory capacity. *Nature 428*, 6984 (2004), 748–751.

[101] Von Ahn, L., Blum, M., Hopper, N.J., Langford, J. Captcha: Using hard ai problems for security. *Advances in CryptologyEUROCRYPT* 2003, pp. 294–311. Springer (2003)

[102] Von Ahn, L., Blum, M., & Langford, J. (2004). Telling humans and computers apart automatically. *Communications of the ACM*, 47(2), 56-60, 2004.

[103] Wand, M. P., and Jones, M. C. Kernel smoothing, vol. 60. *Crc Press*, 1994.

[104] Wheatley, T., & Wegner, D. M. Psychology of automaticity in action. *In N. J. Smelser & P. B Baltes (Eds.), International encyclopedia of the social and behavioral sciences. New York: Elsevier Science*, 2001.

[105] Wickens, C. D., Lee, J. D., Liu, Y., and Gordon-Becker, S. Introduction to Human Factors Engineering (2nd Edition). *Pearson*, 2003.

[106] Wickens, C. D. Engineering psychology and human performance . *HarperCollins Publishers*, 1992.

[107] Wilson, J. T. L. Visual short-term memory. *Development in the assessment and rehabilitation of brain-damaged patients. Krager-Verlag*, 1993.

[108] Wilson, S. G., and Chou, K. L. Separation of low-level and high-level factors in complex tasks: Visual search. *Psychological Review 102*, 2 (1995), 356–378.

[109] Wolfe, J. M. Guided search 2.0 a revised model of visual search. *Psychonomic bulletin & review*, 1(2), 202-238, 1994.

[110] Woudenberg, E., Soong, F. K., & West, J. E. Acoustic echo cancellation for hands-free ASR applications in noise. *In Proc. of the Workshop on Acoustic Echo and Noise Control* (pp. 160-163), 1999

[111] Yampolskiy, R. V., and Govindaraju, V. Behavioural biometrics: a survey and classification. *International Journal of Biometrics 1*, 1 (2008), 81–113.

[112] Yantis, S. Stimulus-driven attentional capture. *Current Directions in Psychological Science*, 156-161, 1993

[113] Yan, J., & El Ahmad, A. S. Breaking visual captchas with naive pattern recognition algorithms. *In Computer Security Applications Conference*, 2007. ACSAC. Twenty-Third Annual (pp. 279-291). IEEE, 2007.

[114] Yan, J., & El Ahmad, A. S. Usability of CAPTCHAs or usability issues in CAPTCHA design. *In Proceedings of the 4th symposium on Usable privacy and security* (pp. 44-52). ACM, 2008.

[115] Yan, J., & El Ahmad, A. S. A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 543–554. ACM, 2008

[116] Yan, J., & El Ahmad, A. S. Captcha robustness: A security engineering perspective. *Computer*, 44(2), 0054-60, 2011.

[117] Zhang, Y., Monrose, F., and Reiter, M. K. The security of modern password expiration: An algorithmic framework and empirical analysis. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 176–186.

[118] Zheng, N., Paloski, A., and Wang, H. An efficient user verification system via mouse movements. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2011), CCS '11, ACM, pp. 139–150.

[119] Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K. Attacks and design of image recognition captchas. *Proceedings of the 17th ACM conference on Computer and communications security*. pp. 187–200. ACM (2010)

[120] Zucchini, W., Berzel, A., and Nenadic, O. Applied smoothing techniques, 2003.

# Appendix A

# SYSTEM IMPLEMENTATION

## A.1 User Authentication System Implementation

As discussed in the beginning of Chapter 3, our system consists of three major modules: (1) The Interaction Module consists of the input device (mouse or laptop touchpad) that is used in executing the cognitive task. (2) The Data Acquisition Module presents the cognitive task in the form of a web-based game. (3) The User Identification Module pre-processes the captured data and provides an authentication decision. We also have three other periphery modules: (4) Simulation Module which trains attackers for impersonation attacks, (5) Usability Module which calculates the usability score and other statistics of the users and (6) Correctness Check Module for users to verify the correctness of the users' submissions. In the following subsections we discuss the programming languages, frameworks, databases used in our system. We then provide the pseudo-code of some of the major functions.

### A.1.1 Data Acquisition Module

This module provides the cognitive task in the form of a web-based game. The search set or the image that is presented to the user is hand-drawn using a Paint program [75]. We use the Paper.js framework [76], an open source vector graphics scripting framework that runs on top of the HTML5 Canvas, to develop the logics of the game. The user interaction data are recorded using JavaScript and submitted passively via AJAX requests to the web server. Specifically, we record data during certain mouse events. The data are then kept in individual text files (at the server side) each corresponding to a user.

Raw Data Collection:

```
on click event triggered:
```

```
        if(event == click on tile)
                record tile click time
                record click co-ordinates
        end
        if(event == click on gold)
                record gold click time
                record click co-ordinates
        end
end

setInterval(recordDragEvent, 100ms):
recordDragEvent:
        if(event == drag on tile)
                record drag event co-ordinates
        end
        if(event == drag on gold)
                record drag event co-ordinates
        end
end
on release event triggered:
        if(event == tile release)
                record tile release time
                record release co-ordinates
        end
        if(event == gold release)
                record gold release time
```

```
                record release co-ordinates
        end
end
```

Raw Data Submission:

```
if(instanceCompleted)
        $.ajax
                type: POST
                data: data
        end
end
```

JavaScript functions are used to enforce some of the constraints discussed in Section 3.1.1. Table A.1 provides a brief overview of the constraints, $UAC1$, $UAC2$, ..., $UAC7$.

### A.1.2 User Identification Module

Once the data are collected from the data acquisition module, it needs to be preprocessed. Preprocessing involves ordering the incoming data packets and removing noise from some features (Section 3.3.2). Note that training and test data are preprocessed separately. We then derive the cognitive features using programs written in C Sharp.

**Ordering Packets**

Due to network delay, the data packets might not arrive sequentially. Recall, that an *instance* is composed of the $A_{resp}$ and $A_{rew}$ actions. The $A_{resp}$ action involves the correct matching of the challenge tile and the target tile and the $A_{rew}$ action involves depositing the gold coin in the bank. We highlight here that network delays do not affect the content of the packets. This is because we use JavaScript to collect local machine timestamps for the raw mouse events. Once these raw data for an *action* are accumulated, they are sent over the network to

| **Constraints** | **Usage and Importance** |
|---|---|
| *UAC1* | Always presents a (+ve) search set |
| *UAC2* | Allows user to observe current game status. |
| *UAC3* | Hides actual position of *loose* tiles until they appear as $t_c$ and aids in invoking serial search. Without this constraint users could have remembered certain target positions beforehand. This would have biased the visual search process |
| *UAC4* | Provides prime while dragging coin to trigger automatic processing. Guiding line color changes from green to red |
| *UAC5* | Reduces conspicuity of a target among distractors. Aids in invoking serial search |
| *UAC6* | Provides prime while dropping coin to trigger unconscious automatic processing. |
| *UAC7* | Without this constraint user could have remembered certain target positions beforehand. This would have biased the visual search process. |

Table A.1: User Authentication Constraints look-up table (brief descriptions). Refer to Section 3.1.1 for details.

our server. We can then face two circumstances: (1) packets might arrive disordered or (2) packets might go missing. We wrote a program which go through these packets or data for each user and alerts us when ordering is necessary or when packets are missing. If packet for an *action* e.g. $A_{resp}$ is missing then the corresponding $A_{rew}$ is not taken into consideration. Packet missing is a rare phenomenon in our case.

**Data Organization**

We developed a program which can organize the data into training and test set, according to user-defined parameters. The program also derives a few features such as pauses, straightness. Once these features related to the drag events are calculated the drag events are discarded. This is done to reduce the bulkiness of the files. The resulting organized file have a fixed number of columns for each user.

NumberOfPauses:

```
for i = 1: length (dragEventXY) - 1
        if(dragEventXY(i) == dragEventXY(i + 1))
                NumberOfPauses ++;
        end
end
RemoveAllDragEventCoordinates();
```

**Feature Estimation and User Identification**

The data are then fed into programs which remove noise (where necessary) and derive the rest of the cognitive features (Section 3.1.3). The training and test feature vectors are then kept into two separate files. For each feature vector we construct the probability distribution function using a Gaussian kernel. We determine the bandwidth using a leave-one-out least square cross validation technique. Our program can take other kernels (such as rectangular, epanechnikov) into consideration. The test data points are then used *instance-*

wise to calculate the probability of the *instance* belonging to a particular user.

```
Repeat for each user:
FeatureVector = EstimateFeatures();
for i = 1: length(FeatureVector),
        ConstructAndStorePDF();
end

Repeat for each user:
for an instance
        CalculatePosteriorProbability
        ClassifyInstanceToMaxPosteriorProbability
end

CalculateErrorMetrics();
```

### A.1.3 Simulation Module

We developed a web-based program using Paper.js, HTML5 and JS which can simulate a user's game playing activities. Recall, that in the data acquisition period, users' data are recorded. During an impersonation attack we select some of those users' data and feed them into the simulation program. The attackers are then trained on these simulations. The program takes into consideration the movement co-ordinates, timestamps, click and release points of mouse, pick and release co-ordinates of tiles/golds, and other minute details while making the simulation. It even shows the changes in color of the guiding lines as the gold is dragged. Therefore, it is very accurate and seems like a recorded video.

### A.1.4 Usability Module

As discussed in Section 3.3.4, we provided an exit survey consisting of the SUS. We developed a C Sharp program to automatically calculate the SUS score. To calculate the SUS score [17], we first sum the score contributions from each item. Each item's score contribution is converted from a 1-5 to 0-4 scale. For items 1, 3, 5, 7, and 9 the score contribution is the scale position minus 1. For items 2, 4, 6, 8 and 10, the contribution is 5 minus the scale position. We then multiply the sum of the scores by 2.5 to obtain the overall value of SUS. SUS scores have a range of 0 to 100.

## A.2 Movtcha System Implementation

As discussed earlier in Chapter 4, Movtcha consists of three major modules: (1) The Generation Module generates the Movtcha challenges, which are basically the processed images. (2) The Data Acquisition Module provides Movtcha in the form of a web-based game. The user uses a mouse to solve the Movtcha. The interaction data is recorded using JavaScript and submitted passively via AJAX requests to the web server. (3) The Evaluation Module differentiates a human from a computer based on the acquired data and an accuracy metric. We had two separate periphery modules similar to the user identification system. (4) The Usability Module that calculates $SUS$. (5) The Simulation Module which can simulate any user's Movtcha solving activities. As discussed earlier in Section 4.4.2 the purpose of this module was to observe the search strategy of the user. (6) Correctness Check Module is used to verify the correctness of the users' Movtcha solutions. We only provide discussion on the Data Acquisition Module, Evaluation Module.

### A.2.1 Data Acquisition Module

The Data Acquisition Module is responsible for presenting the Movtcha challenges. Similar to the user authentication system, we have used the Paper.js framework on top of HTML5

canvas to develop the logics of the game. All the constraints discussed in Section 4.1.1 are implemented in our system. Recall, that the challenge tile moved randomly inside a bounding region (Figure 4.2). To create animations in Paper.js, we use the onFrame handler. When this function is defined, it is called up to 60 times a second by Paper.js [76]. The view is redrawn automatically after the onFrame function has been executed. The user interaction data are recorded using JS and submitted passively via AJAX requests to the web server. The data are then kept in individual text files (at the server side) each corresponding to a user. The raw data consisted of mouse events as discussed in Appendix A.1.1. The visual search time, $VST$, is calculated at the end of each *instance*, i.e. when the star is deposited. The $VST$ is sent along with the other raw data.

Movtcha demands almost a smooth movement during the $A_{rew}$ actions for reasons discussed in Section 4.1.1. If the user crosses a certain threshold of 200 ms, the star, $s_r$, moves back to the target tile, $P_{t_r}$.

Smooth Star Movement during $A_{rew}$:

```
setInterval(function(){CheckPause(pauseEvent)},200);
function CheckPause(pauseEvent)
        if(pauseEvent.point == pastPauseEvent.point && starPicked == true)
                numberOfPauses = numberOfPauses + 1;
                if(numberOfPauses > 1)
                        PauseConstraintActivate();
                end
        end
        pastPauseEvent = pauseEvent;
end
```

### A.2.2 Evaluation Module

When the user solves all the required *instances* of a Movtcha, the back-end server reads the user's file and executes Algorithm 1, Chapter 4. If the resulting accuracy metric, $\triangle^{A}_{VST}$, crosses the system threshold, the user is authenticated as a human. The algorithm is implemented in a PHP file, therefore, the results are immediately available, when required. Moreover, since the algorithm sorts the data according to the search set size, disordered packets did not affect the accuracy calculation.