リレーアタック耐性と BOT 耐性の両立を目指したインタラクティブな動画 CAPTCHA 方式に関する研究

富田 旋 [a)]・立田 怜平 [a)]・山場 久昭 [b)]・油田 健太郎 [c)]・岡崎 直宣 [d)]

A Consideration of Interactive Motion Picture CAPTCHA System Aiming at Compatibility between Relay Attack Tolerance and BOT Tolerance

Meguru TOMITA, Ryohei TATSUDA, Hisaaki YAMABA, Kentaro ABURADA, Naonobu OKAZAKI

Abstract

CAPTCHA is a type of challenge response test used to distinguish human users from malicious computer programs such as bots, and is used to protect email, blogs, and other web services from bot attacks. So far, research on enhance of CAPTCHA' s resitance to bot attacks has been proceeded to counter advanced automated attacks method. However, an attack technique known as a relay attack has been devised to circumvent CAPTCHA. In this attack, since human solves CAPTCHA, the existing measures assuming bots have no effect on this attack. We designed a new CAPTCHA scheme for relay attacks tolerance and automated attacks tolerance. In this paper, we tested the robustness of the proposed method against several types of automated attacks. We constructed an experimental environment in which a relay attack can be simulated, and designed a series of experiments to evaluate the performance of the proposed method. As a result, we found that the proposed CAPTCHA scheme offers some of level of resistance to automated attacks and relay attacks.

Keywords: CAPTCHA, relay attacks, BOT, challenge response

1. はじめに

Web サービスの普及により、誰でも様々なサービスを利用することが可能となっている。それらの Web サービスに対して、ボットと呼ばれる自動プログラムを用いた不正行為が行われている。例えば、メールサービスのアカウントをボットを用いて自動的に大量取得し、スパムメールの送信に利用するなどの事例が挙げられる。このような、不正行為を防止するために、CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) と呼ばれる反転チューリングテストによる判別手法が広く利用されている [1)]。CAPTCHA は、チャレンジ/レスポンス型テストの一種であり、人間には容易に解答できるがコンピューターには困難な問題を出題し、正しい解答をした者を人間と判断するシステムである。

一般的に利用されている手法には、Web ページ上に歪みやノイズを加えた文字列画像を提示し、Web サイトの閲覧者がその文字列を判読できるか否かを試す文字列 CAPTCHA がある。しかし、OCR 技術の進歩や、解読アルゴリズムの向上により、文字列 CAPTCHA は容易に突破されるようになってきている。そのため、動物や物などの画像を識別する人間の高度な能力を利用する画像 CAPTCHA や文字列 CAPTCHA を動画へ応用した動画 CAPTCHA など数多くの方式が提案されてきた。このように、人間には判読しやすく、かつ、ボットには解読が難しい CAPTCHA を実現するために数多くの研究が行われてきたが、CAPTCHA を回避する手法として、リレーアタックと呼ばれる攻撃手法が用いられることがある [2)]。リレーアタックは、インターネット上の一般ユーザーや報酬に誘引された人間を利用して CAPTCHA を解読させ, その解答を利用する手法である。 本稿では、リレーアタックに加担する人間を「幇助ユーザー」と呼ぶことにする。リレーアタックでは、人間が CAPTCHA の解読を行うのでコンピューターを想定した対策では効果がなく、新たな対策が求められている。そこで、本稿ではリレーアタックを行った際に生じる遅延時間に着目し、リレーアタックでの CAPTCHA の解答を困難にすることを目指した CAPTCHA 方式を提案する。提案方式は、表示される動画に対してマウスカーソルを移動させるアクションを行う動画型 CAPTCHA である。 ランダムな位置に出現する複数の妨害オブジェクトの中から連続的に移動してその位置を変化させる移動オブジェクトを認識し、マウスカーソルで追跡できるか否かで人間かボットかを判別する。リレーアタックでは、攻撃者が幇助ユーザーに CAPTCHA の出題画像を転送する通信の遅延時間が発生するため、提案方式 CAPTCHA の場合、攻撃者に提示されている動画と幇助

[a)] 工学専攻機械・情報系コース大学院生
[b)] 情報システム工学科助教
[c)] 情報システム工学科准教授
[d)] 情報システム工学科教授

ユーザーに中継されている動画には、ずれが生じ、リレーアタックによる移動オブジェクトの追跡が困難になると考えた。本稿では、リレーアタックを再現し、CAPTCHA の転送で生じる遅延時間で提案方式 CAPTCHA の解答が困難になるかを検証し、リレーアタック耐性を確認した。また、画像処理に基づいた自動的な攻撃を実装し、ボットへの耐性を確認した。

2. 関連研究

2.1 CAPTCHA とは

CAPTCHA（Completely Automated Public Turing test to tell Computers and Humans Apart）は、2000 年にカーネギーメロン大学の Luis von Ahn, Manuel Blum, Nicholas Hopper, John Langford によって開発された、人間とコンピュータを区別するための反転チューリングテストである [1)]。

CAPTCHA は、自動化された Web クライアントによって、大量のリクエストを投入されては困る案件において、相手がコンピュータではないことを確かめる目的で用いられる。例えば、無料メールサービスのアカウント登録フォームやブログのコメント投稿フォームのようなケースでの利用が挙げられる。

オリジナルのチューリングテストは、アラン・チューリングによって考案された。あるコンピュータが知的かどうか（人工知能かどうか）を判定するためのテストである。このテストには、人間、コンピュータ、判定者（人間）がいる。判定者（人間）は、人間とコンピュータに対して、通常の言語で一連の質問をする。この時、人間とコンピュータは「人間」であるかのように振る舞う。これらの参加者は、それぞれ隔離されている。判定者（人間）は、コンンピュータの言葉を音声に変換する機能に左右されることなく、その知性を判定するために、会話はディスプレイやキーボードといった、文字のみでの更新に限定する。判定者の仕事は、参加者のどれが人間であり、コンピュータであるかを判定することである。判定者がコンピュータと人間の正確な区別ができなかった場合、このコンピュータはテストに合格したと言える。

CAPTCHA は、名前に “Turing test” が含まれているが、オリジナルのチューリングテストと唯一の違いがある。それは、対象者やそれぞれの役目、目的における「コンピュータ」・「人間」を反転させているという点である。つまり、CAPTCHA の場合は関わっている相手が人間か別のコンピュータかをコンピュータ自身が判定することになる。これが、反転チューリングテストと呼ばれる理由である。

2.2 CAPTCHA の定義

ここで、CAPTCHA の定義について述べる。C を起こりうる事象とすると、その事象 C が起こる確率を $[C]$ と表す。また、P が確率的プログラムである場合、ランダムな値 r を使用する時に生じる決定論的プログラムを P_r で表す。(P, V) を確率的相互作用のプログラムのペアとする。ランダムな値 u_1 と u_2 を用いて、P と V の相互作用が終了したと仮定した時の V の出力を $\langle P_{u_1}, V_{u_2} \rangle$ と表す。全ての P と u_1、u_2 について、P_{u_1} と V_{u_2} との相互作用が終了し、$\langle P_{u_1}, V_{u_2} \rangle \in \{accept, reject\}$ の場合、プログラムはテストと呼ばれる。V を検証者及びテスターと呼び、V と相互作用する P を証明者と呼ぶ。テスト V を介するエンティティ A の成功は、

$$\boldsymbol{Succ}_A^V = P_{r,r'}[\langle A_r, V_{r'} \rangle = accept] \quad (1)$$

と定義できる [3)]。A は、V の仕組みを正確に知ることができ、A が知ることのできない唯一の情報は、V の隠れたランダム性 r' と仮定する。CAPTCHA は、ほとんどの人間が “1” に近い成功率を収めるテスト V であり、人間を超える成功率を持つコンピュータを書くのは難しい。ただし、人間の成功は、視覚的なハンデなどに依存する場合がある。例えば、色盲を抱えている人は、色の区別が必要なテストでは、成功率が低い可能性がある。

また、CAPTCHA は自動化されており、テストを管理運用するにあたって人間の介在をあるいはほとんど必要としない。これは、テストにおける人間の管理や介入の必要性を避けることができ、コストや信頼性において明らかに有益である。CAPTCHA に、使用されているアルゴリズムは多くの場合、公開される。アルゴリズムの公開は、CAPTCHA のセキュリティ的に問題にはならない。CAPTCHA の突破には、リバースエンジニアリングなどの手法を用いて達成できるような秘密のアルゴリズムの発見よりも、人工知能の分野における難問を解決することが必要なためである。

2.3 CAPTCHA の利用

一般的な CAPTCHA は、Web サイトの登録フォームの下部などに表示されるいくつかの歪んだ文字を含む画像である。ユーザーは、人間であることを証明するために、歪んだ文字を入力することを求められる。例として、図 1 には、Microsoft 社の Web サービスのアカウント登録時に提示される CAPTCHA を示し、図 2 には、Yahoo Japan のメールアカウントの登録時に提示される CAPTCHA を示す。

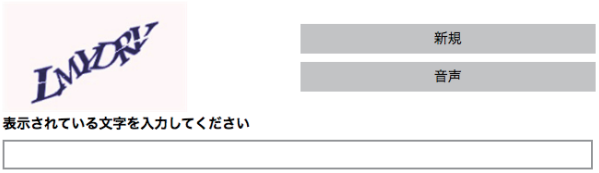

図 1. Microsoft 社のサイトで利用されている CAPTCHA[4)]

2.4 CAPTCHA の分類

CAPTCHA は、文字や画像などの難読化（歪みやノイズの追加）の対象に基づいて分類される。また、近年の研究では、コンピュータでの模倣が難しい人間の高度な認知能力を問うものがある。

2.4.1 文字列 CAPTCHA

最も広く利用されてきた CAPTCHA 方式であり、人間には認識できるが、自動化されたコンピュータ（以下、ボットとする）には認識することが困難な歪みやノイズを含んだ文字を出題する。図 3 に例を示す。

この CAPTCHA 方式のメリットは、システムが単純であり、Web システムへの導入が簡単である点と、総当たり攻撃に強い耐性があるという点である。一般的な文字列 CAPTCHA は、英字 52 字（大文字と小文字を含む）と数字 10 字の合計

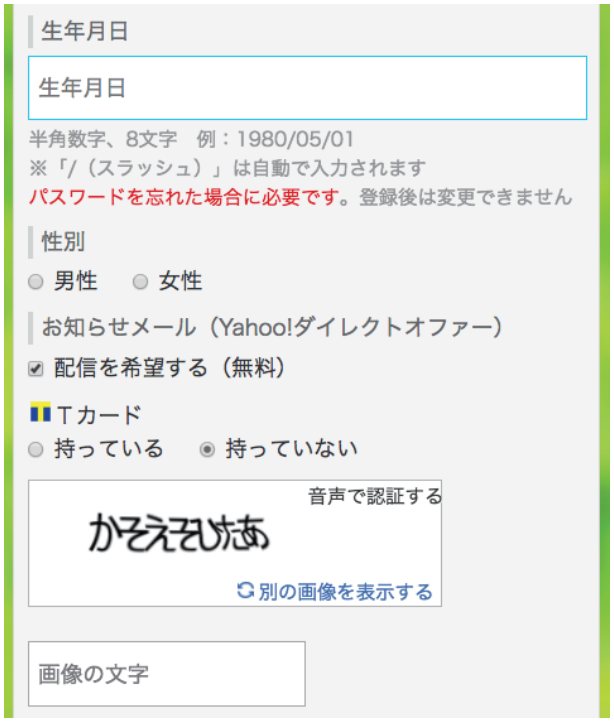

図 2. Yahoo Japan のアカウント登録時の CAPTCHA[7)]

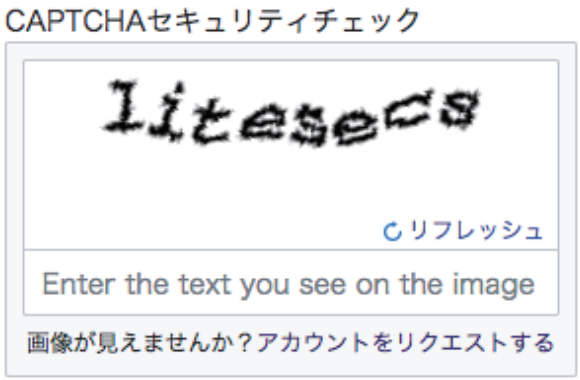

図 3. Wikipedia のアカウント登録時の CAPTCHA[6)]

62 字の英数字が用いられるので、CAPTCHA の文字数が a だとすると、文字列の画像のパターン数は 62^a 通りということになる。ボットがこの文字列 CAPTCHA を総当たりで突破する場合、62^a 通りの答えを試さなければならない。

表 1. 入力文字数 a における文字列画像のパターン数

文字数 a	4	5	6
総当たり数	1.48×10^7	9.16×10^8	5.68×10^{10}

文字列 CAPTCHA のデメリットは、OCR（光学文字認識）攻撃への耐性が弱いことである。技術の発達に伴い、OCR の文字列の認識精度が向上したことにより、難読化を施した文字であっても、ボットによって突破されてしまうという事態が発生するようになってしまった [7)8)]。この事態に対応するために様々な文字列 CAPTCHA が開発されてきた。

2.4.2　Gimpy

Gimpy[9)] は、2 つの単語が重複して表示されているものを 1 セットとし、画像の中にそれが 5 セット表示されている。ユーザーには、表示された 10 個の単語の中から、3 つ答えさせる CAPTCHA である。ボットに対しては、文字列の歪みや重複による難読化で認識を困難にしている。Gimpy で使用されている全ての単語は、Ogden's Basic English word List[10)] にある 850 単語から取得される。

図 4. Gimpy[9)]

2.4.3　EZ-Gimpy

EZ-Gimpy[11)] は、Gimpy よりも単純な方式である。ボットを防ぐために Yahoo のページに実装された。この方式は、1 つの単語、あるいは、アルファベットと数字をランダムに並べた文字列の画像を歪ませて表示し、その解答を入力させる。EZ-Gimpy では、Gimpy と同様に Ogden's Basic English word List から単語を選び、それに難読化を施した。また、ボットを混乱させるために、複雑な背景を追加している。

図 5. EZ-Gimpy[9)]

2.4.4　Gimpy、EZ-Gimpy の突破

Gimpy、EZ-Gimpy は、バークリー校が 2002 年に CAPTCHA のコンピュータによる解読に取り組み、「EZ-Gimpy」の成功率が 83%、EZ-Gimpy よりも困難な「Gimpy」の成功率が 30%で突破されてしまった [11)]。例え、30%の成功率であっても、攻撃者はボットを使って、スパムメールなどに利用するアカウントを大量発行するので、10 万件のうち 3 万件でも CAPTCHA を突破できれば、攻撃者は十分に元が取れてしまう。

2.4.5　reCAPTCHA

reCAPTCHA は、ボットの Web サイトの不正利用を防ぐために CAPTCHA を利用するのと同時に、その CAPTCHA に対する返答を書籍のデジタル化に活かすシステムである [12)13)]。オリジナルは、2007 年にカーネギーメロン大学・ピッツバーグ本校にて開発された。2009 年 9 月 16 日に Google はこのテクノロジーを買い取っている。reCAPTCHA は、ニューヨークタイムズが持つ記事アーカイブの電子化及び、Google ブックスの書籍電子化に利用され、前者は、2009 年の時点で 130 年分を超える全記事のうち約 20 年分のデジタル化を、2、3 ヶ

月で完了した。

reCAPTCHA は、デジタル化した書籍データの中から、OCR で正しく識別されなかった単語を切り取り、CAPTCHA として出題する。しかし、CAPTCHA はコンピュータと人間を区別することが主な目的であり、正しく入力されたか判定するための「正解」が必要となる。そこで、OCR で正しく識別されなかった単語に加え、正しく識別された単語を用いる。出題される文字列の画像には、2 つの単語が含まれており、一方は正しく識別されており、正解が存在する。もう一方は、正しく識別されなかった人間に認識してもらう必要があるものである。

具体的な仕組みについて述べる。スキャンされた文字列を 2 つの OCR で各々解析する。両 OCR の結果が異なった場合、疑わしい文字として、CAPTCHA に変換する。ただし、この時、既に OCR で認識できている文字を「対照文字」として、この CAPTCHA に追加する。2 つの単語は、ボットによる CAPTCHA の突破を困難にするため、難読化を施す。文字を読み取った人間が「対照文字」を正しく認識していた場合、OCR で正確に読み取れていなかった文字に対する解答も正しいものであるとシステムは仮定する。各 OCR プログラムによる文字認識結果には、0.5 点を与え、人間の文字認識結果には、1.0 点が与えられる。特典が 2.5 点に達した時、スキャンされた文字の認識結果が決定する。例えば、あるスキャンされた 1 つのテキストに対し、2 つの OCR プログラムの認識結果がそれぞれ、“dog” と “cog” だったとする。これを reCAPTCHA を通じて人間に見せた場合、出題 syutudai 開始から 2 人が “dog” と解答した時点でこのスキャン結果は “dog” だったとみなす。また、2 人が “cog” と解答した時点でも同様である。そして、3 人が “bog” と解答した時点で両 OCR のスキャン結果を破棄し “bog” だったとみなす。人間の判断により単一の認識結果が一貫して与えられた文字は対照文字として再利用される。

図 6. reCAPTCHA[12)13)]

2.4.6 文字列 CAPTCHA の限界

2017 年、米国の AI 企業 Vicarious の研究者らが、CAPTCHA で表示される文字を認識できる新たな学習モデルの結果を発表した [14)]。発表された手法では、CAPTCHA の文字を認識するために必要となる訓練データは従来のディープラーニング手法に比べて、約 300 倍効率的であるという。彼らは、少数の例から学習して、一般化できる能力を持つ人間の脳を参考に、神経科学の知見を導入した「再帰的皮質ネットワーク」を作成した。結果として、1 文字あたり 5 つの訓練サンプルを用意するだけで、reCAPTCHA の文字画像を文字単位で見ると「94.3%」、単語単位で見ると「66.6%」の精度で正答することができた。少ない訓練データで CAPTCHA を突破できることから、研究者らは、より強固なメカニズムに移行すべきだと述べている。

2.5 画像 CAPTCHA

文字列 CAPTCHA 方式における脆弱性が多くの研究者に指摘され、文字列に加える変形やノイズを大きくすることで、ボットへの耐性を向上させようとしたが、そのような文字は、人間にとっても難しくなってしまい、人間の正答率まで低下させてしまう事態になった。そこで、文字認識以外の人工知能における難問の提示が必要になった。その 1 つが画像 CAPTCHA 方式である。画像 CAPTCHA 方式は、具体物の画像を用いることで、人間とコンピュータを判別する。出題する問題の種類は様々あり、用いる画像の枚数や解答方式に違いがある。

2.5.1 PIX

PIX[15)16)] は、共通した色や行動、形を認識できる画像を複数枚表示し、ユーザーに共通する分類を 1 つ答えさせる方式をとっている。その答えが正しかった場合には、解答したユーザーを「人間」とみなす。例えば、図 7 のような画像が提示された場合は、ユーザーは画像に共通する分類として「赤ちゃん・baby」を入力する。この手法は、画像が表す情報の共通点を人間ならば容易に見いだすことができる能力に基づき選択させるユーザーを判別する。画像の内容を理解することは、人工知能における難問であるため有効であると考えられた。

図 7. PIX[15)]

しかし、PIX はデータベース攻撃に弱いことが分かっている。データベース攻撃とは、出題される画像とその解を記録したデータベースを構築し、このデータベースを用いて CAPTCHA を突破する方法である。PIX では、画像に対してその特徴別に分類がされている。攻撃者は、何度も CAPTCHA による認証を繰り返すことで、画像データを取得していく。取得した画像データに対して、特徴別に分類を行い、データベースの構築を行う。出題者は、画像データに対して、人間が認識できる特徴から分類を行わなくてはならないため、無作為に選択した画像を用いて出題することができない。従って、問題

として構築されるデータベースの情報量は、人間が手入力で行える範囲である。このことから、攻撃者がデータベースを構築するのは比較的容易であることが推測され、PIX がデータベース攻撃に対して脆弱であると考えられる。

また、画像検索攻撃に対しても脆弱であると考えられる。画像検索攻撃とは、CAPTCHA の問題として提示された画像を Web 状の画像検索エンジンで検索することで、正答または、正答に直結するキーワードを取得し、CAPTCHA を自動的に通過する手法である。PIX は、複数の画像に共通しているキーワードを入力する形式であるため、画像検索の結果次第では、共通したキーワードを得ることは容易であると考えられる。

2.5.2　Asirra

Asirra[17] では、複数の動物の画像を表示し、その中から特定の動物の画像を選択させる。具体的には、ユーザーは提示された 12 枚のイヌまたは、ネコの画像のうち、ネコの画像を選択することを要求される。ネコの画像を全て選択することができれば、「人間」だと判定される。Asirra も PIX と同様に、画像の内容を理解するという、人工知能における難問を利用している。

図 8. Asirra[17]

PIX は、データベース攻撃に対する脆弱性が指摘されたが、Asirra は、この攻撃に対する耐性を備えている。Assira では、出題用の画像データベースを常に更新することでデータベース攻撃を避けている。Assira で用いられている画像は、Petfinder。com[18] というペットの里親募集サイトと提携して取得している。その数は、およそ 300 万枚であり、全ての画像に対してイヌとネコのラベル付けがされている。そのため、スクリプトによって Asirra から画像を取得してデータベースを構築するためには、1 秒で 1 枚追加すると仮定すると、1 ヶ月以上を要する。さらに、Petfinder.com に登録されているペットは半年で完全に入れ替わることが予想されており、そのことから、1 日に約 10000 枚というペースで画像データベースが更新されることが考えられる。また、Petfinder.com 上で表示されている画像は、実際に蓄積している画像の 10%にも満たない。従って、登録されている全ての画像を即座に取得する方法がなく、常に新しい画像が登録され続けているため、Web スクレイピングによって自動的にデータベースを構築することも不可能である。以上のことから、Asirra を突破するためのデータベースを構築することは、現実的ではない。

しかし、SVM を用いた機械学習により、10.3%の確率で Asirra による判別テストが破られたと報告されている [19]。

2.5.3　画像 CAPTCHA のメリット・デメリット

画像 CAPTCHA 方式のメリットは、文字列 CAPTCHA 方式の脅威であった OCR 機能を持ったボットが通用しないことや、人間が直感的な画像認識を行うことができ、Asirra においては画像を選択するだけで良い点が挙げられる。

デメリットは、誤って CAPTCHA の判定テストを通過する確率（偽陽率）が高い点である。例えば、1 回の CAPTCHA 画像を 12 枚、そのうち選択すべき正答の画像が a 枚である場合、偽陽率は、次式となる。

$$\frac{1}{{}_{12}C_a} \tag{2}$$

正答画像の枚数 a が明らかでない場合、攻撃者は正答画像の枚数 a を知らないため、$\frac{1}{\sum_{a=1}^{12} {}_{12}C_a} = \frac{1}{4095}$ の確率でテストを通過する可能性がある。この確率は、表 1 にある文字数 4 つの場合と比較しても非常に高い。偽陽性を下げるためには、出題する画像の選択肢を増やす方法が考えられるが、大きな表示スペースの必要性や、各画像の一覧性が悪くなり、使い勝手が悪くなってしまう。また、Asirra が機械学習によって突破されたことから、「画像の内容を理解する」という人工知能における難問が解決されることは時間の問題である。そのため、CAPTCHA のテストをより高度化していく必要があると考えられる。

2.6　動画 CAPTCHA

動画 CAPTCHA 方式は、文字列 CAPTCHA や画像 CAPTCHA の拡張方式となっており、静的な画像の出題形式の後継として開発された。代表的な動画 CAPTCHA 方式には、NuCAPTCHA が挙げられる。

2.6.1　NuCAPTCHA

NuCAPTCHA[20][21] は、カナダのソフトウェア企業 Leap Marketing Thechnologies が開発した。この CAPTCHA は、複数のフォントを用いたランダムな文字列が動画で表示され、ユーザーは動画上部に表示される色指定などを読み取り、動画中に流れる文字列の中から該当文字列をテキストボックスに入力する。この CAPTCHA 方式は、動的な背景や文字列の複雑な動き・歪みがあったとしても、文字を正しく認識できる人間の高度な能力を利用したものである。

図 9. NuCAPTCHA[21]

動画 CAPTCHA のメリットとして、動画を用いることにより、文字列の色の変化や動きなど、歪みやノイズなどの従来の文字列 CAPTCHA の難読化に新しい要素を追加することができることである。このような、難読化のバリエーションの増加は、過度な歪みやノイズによって、人間にも文字列 CAPTCHA が読めなくなるという事態をある程度抑えることができ、ボットにとってもより難しい問題になると考えられる。

しかし、スタンフォード大学のセキュリティ研究者が、90%の確率で NuCAPTCHA を破ることに成功したとブログで発表した [22]。突破は、5 段階の攻撃アルゴリズムを利用しており、まず背景を取り除き、文字列を白黒化した上で、フレーム解析を行って、各フレーム内のオブジェクトを特定。クロスフレーム解析とセグメンテーションを通じて文字列を抜き出し、個々の文字を判別する。この行程は、市販のソフトウェアを利用して実行できるとされている。

2.7 人間の高度な認知能力を利用した CAPTCHA

2.7.1 アモーダル補完を利用した動画 CAPTCHA

この CAPTCHA は、従来型の文字列 CAPTCHA をベースにした動画 CAPTCHA で、人間の視覚補完を利用することでユーザビリティを確保しつつ、ボットの突破率を低下させるものである [23]。人間には、物体が遮蔽された状態であっても内容を認知することができるアモーダル補完と呼ばれる視覚補完能力がある。アモーダル補完が起こると、遮蔽された文字であったとしても人間は瞬時にその文字が何であるかを知覚することが可能である。対して、ボットは認識率が大幅に低下する。また、知覚神学の側面から文字の見易さに着目し、これを動画に応用することでボットは一意に解答が出せないよう曖昧さを持たせることで解析コストを高めている。欠点としては、文献 [23] で挙げられているとおり、最終的にはボットであっても動画中の文字を認識できるため、人間と機械の認識にかかる時間の差異を用いているが、その差が 20 秒と十分ではない。今後、コンピュータの性能が向上すると、人間と機械の認識にかかる時間の差が逆転することもあり得る。

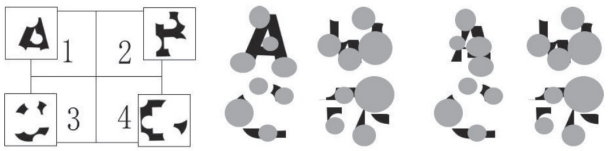

図 10. アモーダル補完を利用した CAPTCHA[23]

2.7.2 メンタルローテーションを利用した画像 CAPTCHA

人間の高度な認知処理を利用した CAPTCHA の一つとして、メンタルローテーションを利用した CAPTCHA が知られている [24]。メンタルローテーションは、1 つの視点から写された 2 次元物体や 3 次元物体を頭の中で回転させ、異なる視点から写された形・姿を認識する能力である。「メンタルローテーション」の能力を利用した CAPTCHA には、複数あるが、最初に提案されてのは、YUNiTi's CAPTCHA である。この CAPTCHA では、3D オブジェクトが 3 個、写された画像が出題され、それと共に、3D オブジェクトが 18 個、写された解答候補画像が提示される。解答候補画像の中には、出題画像の中の 3D オブジェクトが向きを変えた状態の画像が含まれており、ユーザーは、同一の 3D オブジェクトを選択しなければならない。3 次元の空間認識は、コンピューターが苦手とする分野の 1 つであり、YuNiTi's CAPTCHA は、ボットが正解困難である理想的な CAPTCHA の 1 つとして注目を集めたが、テンプレートマッチングを用いた攻撃に脆弱性が存在することが報告された。これに対し、セキュリティを強化したメンタルローテーションを利用した CAPTCHA 方式は、いくつか報告されている [25][26][27][28]。

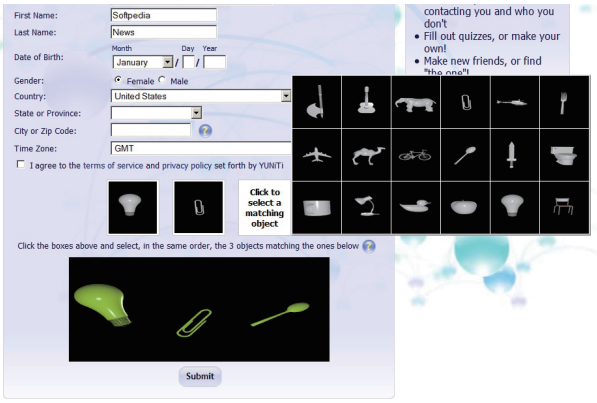

図 11. YuNiTi's CAPTCHA[24]

2.8 リレーアタックとは

2.9 リレーアタックの概要

リレーアタックは、攻撃者が正規サイトから CAPTCHA の問題画像を取得し、第三者の人間に CAPTCHA の問題画像を中継して解答してもらい、その解答を利用することで CAPTCHA を突破する手法である [2]。問題画像の取得や第三者への問題の中継などは、攻撃者の作成したプログラムで自動的に行われる。ボットを使って CAPTCHA を解読するのではなく、ネット上の一般ユーザーの労力を活用し CAPTCHA を解く攻撃ともいえる。以降の説明では、リレーアタックに際して、CAPTCHA の解答を提供する者を幇助ユーザと呼ぶことにする。

2.10 リレーアタックの種類

この節では、リレーアタックの種類について述べる。

2.10.1 リレーサイトを利用する手法

不正者が運営するサイト（以下、リレーサイトと呼ぶ。）にインターネット上の一般ユーザーが訪問して来たら、正規サイトから取得してきた CAPTCHA の問題画像を提示し、リレーサイトのコンテンツを閲覧することと引き換えに CAPTCHA を解読させ、解答を送信させる。この時の、リレーサイトの訪問者である幇助ユーザーは、リレーアタックに加担していると知らずに CAPCHA を解いていることが多い [30][31]。

2.10.2 報酬に誘引された人間を利用する手法

不正者は、賃金などの報酬と引き換えに人間を雇い、解読したい CAPTCHA を大量に雇用した人間に送り、解答させる。リレーアタックに加担した人間が受け取る賃金は、CAPTCHA の解読 1000 個あたり US$0.5〜US$3 程度である [32]。ちなみに、インドでは 1000 個の CAPTCHA の解決につき、約 2$の報酬が与えられている [33]。この時の幇助ユーザーは、報酬のためにリレーアタックだと知っていて CAPTCHA を解く場合が多い。

2.10.3 トロイの木馬を利用する手法

表向きは無害なアプリケーションを装っているが、ユーザーがこのアプリケーションを実行すると、リレーサイトにアクセスを行い、プログラム実行のためには CAPTCHA を解かな

ければならないなどの表示をし、CAPTCHA の解答を得る。CAPTCHA の解答を入力しなければ強制的に PC をシャットダウンするなどの脅迫を行いユーザーに CAPTCHA を解かせるものも存在する。

2.10.4　マルウェア感染 PC の利用者に CAPTCHA 解読を肩代わりさせる手法

ユーザーの PC に感染したマルウェアは、リレーサイトへのアクセスを行い、Web ブラウザが送信するリクエスト等を遮断して、正規サイトの CAPTCHA を割り込ませて提示する。ユーザーが正しく CAPTCHA を解くことができたら、ボットはリクエストを再送信し、その後の Web アクセスを継続する。このリレーアタックでは、ユーザーには閲覧先の Web サイトの CAPTCHA が表示されたように見えるため、ユーザーにリレーアタックと気づかれることなく、ユーザーがネットサーフィンをするたびに CAPTCHA を解かせることが可能である。

2.11　リレーアタックの手順

この節では、図 12 を用いて 2.10 節で述べたリレーサイトを用いたリレーアタックの手順を示す。

(1)　不正者は、リレーサイトを開設する。リレーサイトは、web コンテンツの閲覧や賃金がもらえるなどの報酬によって幇助ユーザーを誘引する。

(2)　インターネット上の幇助ユーザーがリレーサイトに訪問した瞬間に、リレーサイトは自動的に正規サイトにアクセスするように作成されている。正規サイトは、リレーサイトからのアクセスに対して CAPTCHA を提示する。

(3)　リレーサイトは、正規サイトの CAPTCHA 画像を取得してリレーサイト上のその画像を表示した Web ページを生成する。

(4)　リレーサイトを訪れた幇助ユーザーに、(3) で生成した Web ページを提示し、問題を解いたら報酬を与える旨を説明する。ただし、一般には、不正者が CAPTCHA を不正に解こうとしていること、また、幇助ユーザーがその CAPTCHA を解くことで不正行為の幇助になることは説明しない。

(5)　幇助ユーザーは報酬を得るために、Web ページの CAPTCHA を解き、リレーサイトに解答を入力する。

(6)　CAPTCHA の解答を得たリレーサイトは、その解答を利用して正規サイト CAPTCHA を突破する。幇助ユーザーの解答が正解だった場合、正規サイトの CAPTCHA を突破することができ、目的に応じた不正を働くことができる。

(7)　(2) ～ (6) をリレーサイトへの訪問者である幇助ユーザーに繰り返す。この一連の動作は全て自動プログラムによって行われる。

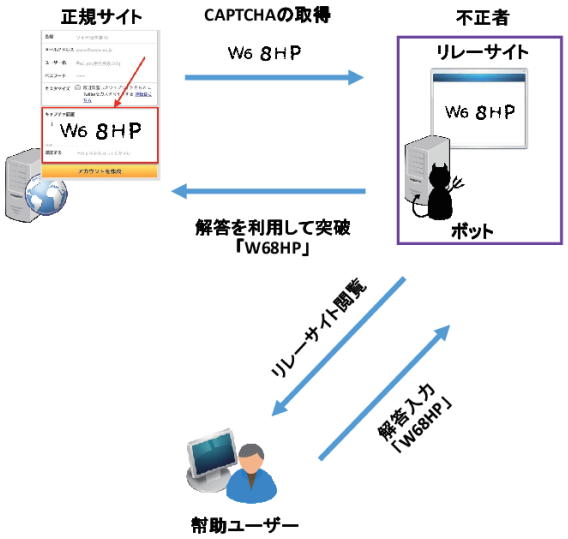

図 12. リレーアタックの手順

2.12　既存のリレーアタック対策

2.12.1　IP アドレスの違いを用いたリレーアタック検知

鈴木らはリレーアタックの特徴、すなわち、正規サイトにアクセスする PC とリレーサイトで中継された CAPTCHA を解く PC が異なっていることを利用し、リレーアタックが行われていることを検知する手法を提案している [29]。

まず、この手法を実現させるための手法について以下に示す。

- 正規サイトごとに CAPTCHA に含めるランダムな文字列 (以下、キーワードとする。) を決定する。
- web ブラウザがデータを送信する際にキーワードが含まれていた場合は、そのキーワードに対応する正規サイトのサーバーにキーワードを検出したことを知らせる CAPTCHA コンファームを送信する機能を web ブラウザに追加する。

CAPTCHA コンファームの内容は、キーワードを含んだ文字列である。CAPTCHA の解答にはキーワードが含まれているので、web ブラウザから CAPTCHA の解答が送信されると正規サーバーに CAPTCHA コンファームが送信されることになる。

これらのことを踏まえると、正規アクセスでは、正規ユーザーの PC から CAPTCHA の解答と CAPTCHA コンファームが送信され、この 2 つの送信元の IP アドレスの一致が確認できる。リレーアタックでは、幇助ユーザーがリレーサイトで CAPTCHA を解いた時と正規サイトに直接アクセスしている PC が幇助ユーザーの解答を利用して CAPTCHA の解答入力をする時の 2 回、CAPTCHA コンファームが送信されることになる。正規サーバーは、幇助ユーザーから届いた CAPTCHA コンファームと正規サーバーにアクセスしている PC からの CAPTCHA の解答の送信元の IP アドレスが一致しないためリレーアタックを検知することができる。

この手法のデメリットは、CAPTCHA の解答にキーワードを含めることができる CAPTCHA でなければならないことである。例えば、Assira のような出題される画像に文字列が割り

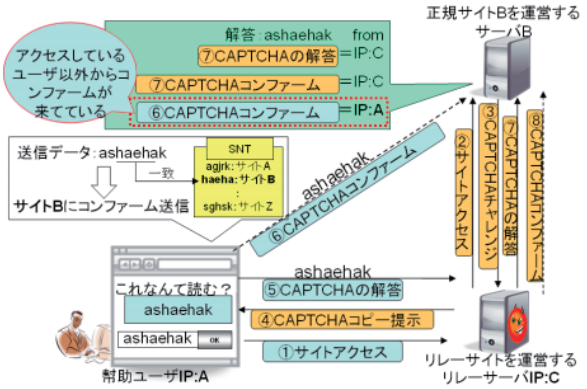

図 13. IP アドレスを用いたリレーアタックの検知 [29)]

当てられており、解答となる画像に割り当てられている文字列が CAPTCHA レスポンスとなるような CAPTCHA は、不適格である。具体的には、正規サイトから取得した CAPTCHA チャレンジの画像を用いて、リレーサイトで「解答となる画像の番号を答えよ。」という形式で CAPTCHA を提示すれば、キーワードを含むことは無いため、幇助ユーザーの PC から CAPTCHA コンファームを発信させずに、CAPTCHA の解答を得ることができる。また今回の提案手法を実現するにあたって、web ブラウザにキーワードを検知し CAPTCHA コンファームを送信する機能を追加する必要がある。しかし、報酬に誘引された人間を利用したリレーアタックのような幇助ユーザーが不正を知った上でリレーサイトにアクセスしてくる場合には、本方式の機能をオフにすることで、検知を回避することが可能である。

2.12.2 リレーアタックのパフォーマンス低減手法

この対策では、動画像中に複数の CAPTCHA を挿入することで単位時間あたりに解読できる数を減らし、報酬により誘引した人間を利用するリレーアタックを金銭コストの面から抑制することを狙っている [34)]。CAPTCHA は、図 14 のような、複数の文字列 CAPTCHA 画像を埋め込んだ動画像を用いて認証する。ユーザーは動画像を再生し、表示された CAPTCHA 画像を解読して入力を行う。時間経過と共に表示される CAPTCHA 画像が変化するので、その都度解読した内容を入力する。動画像が終わると入力した文字列を送信することで認証を行う。この CAPTCHA で正しく解答するためには、CAPTCHA の動画像を最初から最後まで解読する必要があるため、動画像の再生時間を調節することで幇助ユーザーの単位時間あたりに解く CAPTCHA の数を低減することができる。具体的な例を示す。幇助ユーザーを雇用する企業の Web ページによると、幇助ユーザーのテキストベースの CAPTCHA の解読時間は平均 9 秒であり、受け取る賃金は、CAPTCHA の解読 1000 個あたり US$0.5〜US$3 程度である [32)]。1 日の労働時間を 8 時間としたとき、解読時間が 40 秒になると解読 1000 個あたりの賃金を US$3 としても日給は、US$2.16 になる。これは、CAPTCHA 解読を行う人間の存在が確認されているインドにおける最低賃金水準、日額 115 ルピー（US$2.3) を下回っている。このことから、リレーアタックを金銭的な面から抑制できると考えられている。しかし、CAPTCHA の解答にかかる時間を増加させることで正規ユーザーの負担が大きくなり、ユーザビリティの低下につな

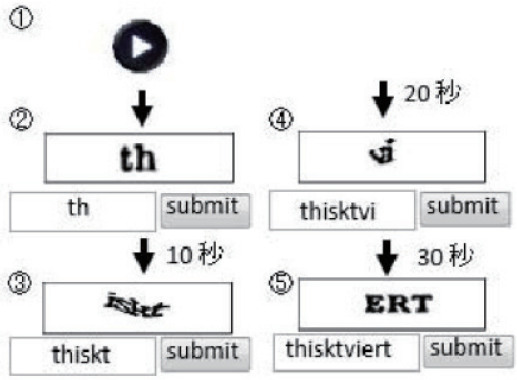

図 14. リレーアタックのパフォーマンス低減のための CAPTCHA[34)]

図 15. iCAPTCHA[35)]

がる問題点がある。

2.12.3 iCAPTCHA:遅延時間を用いたリレーアタック検知手法

iCAPTCHA は、CAPTCHA の問題画像を中継することで生じる遅延時間を利用してリレーアタックを検知している [35)]。この CAPTCHA は、図 15 に示すように CAPTCHA チャレンジの下に文字ボタンがあり、まず CAPTCHA チャレンジの最初の文字に対応する文字ボタンをクリックする。クリックすると文字ボタンが更新されるので、CAPTCHA チャレンジの各文字に対して実行されるまで、この操作を繰り返し、すべて対応する文字ボタンを選択できていたら人間とみなし認証する。リレーアタックでは、正規アクセスに CAPTCHA の問題画像、幇助ユーザーの解答を中継する通信時間が付加される。この付加された時間を利用して iCAPTCHA では、リレーアタックの検知を行っている。筆者らは、iCAPTCHA に対してリレーアタックを行うツールを実装し、iCAPTCHA のサーバーに文字ごとの解答が送信されてくるまでの時間を正規アクセスとリレーアタック、それぞれ測定した。その結果からリレーアタックを検知するための閾値を設定し、リレーアタックを検知することに成功している。しかし、iCAPTCHA 自体は文字列ベースの CAPTCHA であるため OCR（光学文字認識）によって突破される可能性がある。また、近年の OCR 機能の発達は目覚ましいものがあるため文字列ベースの CAPTCHA を利用することは、得策ではないといえる。

2.12.4 DCG-CAPTCHA

DCG-CAPTCHA は、簡単なミニゲーム形式の CAPTCHA である。この CAPTCHA は、ユーザーが与えられた指示に適するオブジェクトをマウスなどで選択し、その選択が正しければ人間とみなすものである [36)]。例えば、図 16 では、複数の異なる形状のオブジェクトの中から、青いエリアのオブジェクトと同じ形状のものを選択し、青いエリアのその形状のオブジェクトの位置にドラッグ＆ドロップで配置できれば、ユーザーを

図 16. DCG-CAPTCHA[36)37)38)]

人間とみなす。また、DCG-CAPTCHA は常にオブジェクトが移動する動的な CAPTCHA である。この CAPTCHA をリレーアタックで突破するには、CAPTCHA のフレーム画像を幇助ユーザーに送信し続けなければならない。このとき、幇助ユーザーが目視している DCG-CAPTCHA は、通信の遅延などにより中継元に表示されているものとずれが生じる。DCG-CAPTCHA を解く場合、オブジェクトの移動などにリアルタイムで対応しなければならない。そのため、幇助ユーザーの解答を利用したとしても、生じる通信の遅延により、リレーアタックでの解答が困難になる。この点に着目し、ユーザーと CAPTCHA とのインタラクションのタイミングを検査することでリレーアタックの検出を実現している [37)]。

しかし、同形状のオブジェクトを認識することや移動オブジェクトをフレーム画像を解析してプログラムで追跡することは容易にできるため、自動プログラムによる攻撃への耐性は低いといえる [38)]。

3.　提案方式の設計とセキュリティ

3.1　提案手法の設計に至った経緯

既存の CAPTCHA の多くは、文字・静止画などの有意味な情報を難読化し、ユーザーに出題している [39)40)]。例えば、文字列 CAPTCHA は、OCR による認識の結果を悪くするために歪みやノイズを加えている。この難読化は、コンピュータによる有意味な情報の認識（文字認識や画像に写っている具体物の認識）を困難にする目的がある。しかし、難読化された有意味な情報を認識できるか否かで、人間であることを確認する方式は、難読化する前の情報にできるだけ近づけて認識する手法によって破られている [41)42)]。これを解決するために、有意味な情報と人間の高度な認知能力を組み合わせた方式が提案された。人間の高度な認知能力の模倣は、ボットにとっては困難であり、近年、提案される CAPTCHA 方式は、人間の高度な認知能力を問うものが多い（アモーダル補完 [23)]、メンタルローテーション [24)]）。

このように、様々なアプローチの CAPTCHA 方式が考案されてきたが、攻撃手法も、技術の向上に伴い、多様化している。現在、最も利用されている google reCAPTCHA の画像選択型の CAPTCHA は、ディープラーニングを用いて突破できるという報告がある [43)]。また、異なるアプローチの攻撃手法には、「リレーアタック」と呼ばれるものがある。この攻撃手法は、ボットではなく、人間を利用して CAPTCHA の解読を行うため、これまでのボットを想定した対策では効果がない。そのため、人間の高度な認知能力を問う CAPTCHA であっても、突破されてしまう可能性がある。CAPTCHA のセキュリティを確保するためには、ボットだけでなく、リレーアタックの脅威にも対抗できる CAPTCHA 方式を考えることが必要となる。

リレーアタック耐性を備えた CAPTCHA 方式に DCG-CAPTCHA があるが、DCG-CAPTCHA は、第 2. 章でも触れたが、機械学習と画像処理技術を利用した攻撃によって、突破が可能とされているため、ボットに対して脆弱である。だが、DCG-CAPTCHA のようなミニゲーム形式の CAPTCHA では、ユーザはオブジェクトの移動などの変化にリアルタイムに対応しなければならない。リレーアタックで DCG-CAPTCHA を解こうとすると攻撃者側からの CAPTCHA チャレンジの転送と幇助ユーザ側からの解答の転送の 2 つの通信による遅延時間が発生するため、幇助ユーザの解答を利用して CAPTCHA の解答タスクを正確に実行することを困難にできる。このことから、リレーアタックに耐性を持たせるためには、DCG-CAPTCHA のようにオブジェクトが移動するような動画形式で、ユーザーがリアルタイムに動画の変化に対応しながら解答を行う方式が有効であると考えられる。また、この方式に加えてボットへの耐性を持たせるために工夫を施さなければならない。

DCG-CAPTCHA は、複数あるオブジェクトの中から答えとなるオブジェクトを選択するタスクが設定されている。しかし、解答を選択する CAPTCHA のほとんどは選択肢が少ないため、偶然突破確率が高くなってしまう。そのため、解答を選択する形式の CAPTCHA は避けるべきだと考えられる。また、DCG-CAPTCHA がボットに突破された原因として、有意味な情報の難読化ができていないことが挙げられる。選択するオブジェクトは、視覚的特徴（色、形状、大きさ）で容易に識別可能であり、ボットでも容易に正解となるオブジェクトを特定できる。これを防ぐためには、視覚的特徴の難読化が有効であると考えられるが、前述したように歪みやノイズなどを加える難読化は、難読化する前の情報にできるだけ戻すことで突破されるため、歪みやノイズを加える方法以外で、難読化を加える方法を考えなければならない。

これらのことを踏まえて、我々は、リレーアタック対策として移動オブジェクトをマウスカーソルで一定時間、追跡する方式を提案した。リレーアタックで発生する遅延時間によって、幇助ユーザの解答（例えば、マウスカーソル座標）を利用して移動オブジェクトを追跡することが困難になるはずである。ただ、オブジェクトを追跡するだけであれば、ボットでも容易に実現できるため、この方式にボット耐性を持たせる。それは、フレーム画像ごとにランダムに位置を変える複数の妨害用のオブジェクトを追加することである。このオブジェクトを追加することで、ボットによる自動的な追跡を防げると考えた。移動するオブジェクトと妨害用のオブジェクトは、同形状、同色、同じ大きさに設定する。これは、視覚的特徴によって移動するオブジェクトを特定されるのを防ぐためである。ボットが自動的に移動オブジェクトを追跡しようとする場合、フレーム画像を解析して追跡対象のオブジェクトを見つけようとするはずである。しかし、提案手法のフレーム画像には、同じ形・色・大きさのオブジェクトが散らばっているようにしか見えないため、特定することは難しくなる。人間には、フレーム画像に同じオブジェクトが散らばっているだけだとしても、動画で見た際に、移動オブジェクトを見つけ

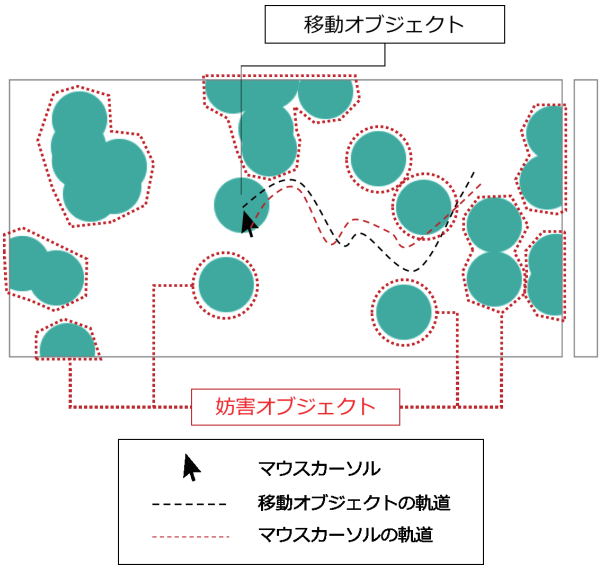

図 17. 提案方式の CAPTCHA

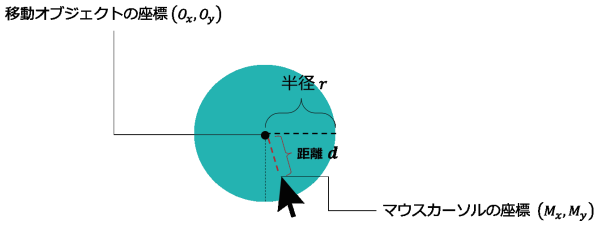

図 18. 追跡成功と判定するときの位置関係

ることは容易にできる高度な認知能力があるため CAPTCHA として成立すると考えた。

3.2 提案手法の概要

本提案手法は、ランダムに位置を変える複数のオブジェクト（以降、妨害オブジェクトとする）の中から、連続的に移動するオブジェクト（以降、移動オブジェクトとする）を見つけ出し、それをマウスカーソルで一定時間以上、追跡できるか否かで、人間かボットかを確認する動画型の CAPTCHA 方式になっている。

具体的には、ユーザーが移動オブジェクト上にマウスカーソルを移動させてから、10 秒間追跡してもらい、その 10 秒の間に何秒間追跡できるかで、ボットか人間かを判断する。移動オブジェクトの追跡判定は、移動オブジェクトの座標 (O_x, O_y) と マウスカーソルの座標 (M_x, M_y) の距離 d を用いる。距離 d は、

$$d = \sqrt{(O_x - M_x)^2 + (O_y - M_y)^2} \tag{3}$$

で求める。この距離 d が移動オブジェクトの半径 r より小さければ、追跡できているとみなす。つまり、追跡成功の判定の条件は、

$$d < r \tag{4}$$

となり、この条件を満たさなければ、追跡できていないと判定する。

提案する CAPTCHA のオブジェクト表示領域の右には、ユーザーが時間を直感的に把握するための時間メーターを配置している。このメーターでは、ユーザーが移動オブジェクトの追跡を開始してからの 10 秒間を緑色で表す。また、移動

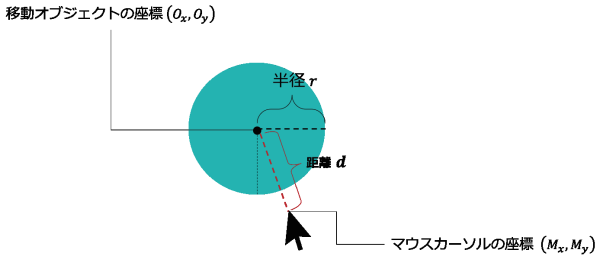

図 19. 追跡失敗と判定するときの位置関係

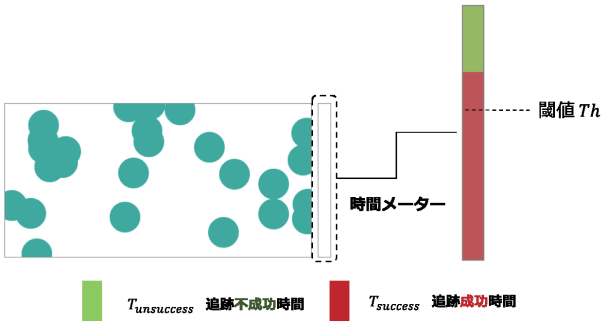

図 20. 提案 CAPTCHA の時間メーター

オブジェクトの追跡に成功している時間（以下、追跡成功時間とする）を赤色で表す。時間メーターは、解答時間（10 秒）の経過や追跡成功時間がリアルタイムに反映される。

図 20 に、CAPTCHA の解答が終わった後の時間メーターを示す。赤色のメーターは、追跡成功時間 $T_{success}$ を表しており、緑色のメーターは、追跡不成功時間 $T_{unsuccess}$ を表している。($T_{suucess}$ と $T_{unsuccess}$ の合計は、10 秒になる。)

提案手法では、次の条件式を満たした場合「人間」であると判定する。

$$Th < T_{success} \tag{5}$$

この条件式を満たさない場合、「ボット」であると判定する。図 20 の例であれば、「人間」だと判定されることになる。この閾値 Th の設定は、ボットとリレーアタックによる攻撃で達成することが難しい値に設定すべきであり、十分な検証が必要である。この設定については、後に記述する。この節の最後に、提案 CAPTCHA の認証手順を図 21 に示す。

3.3 リレーアタックへの耐性について

提案方式の CAPTCHA をリレーアタックで突破するには、CAPTCHA のフレーム画像を幇助ユーザーに送信し、幇助ユーザーから解答情報（マウスカーソルの座標）を得る必要があると考えられる。図 22 に提案 CAPTCHA に対してリレーアタックを行った時の通信についてのシーケンス図を示す。なお、図 22 の提案 CAPTCHA の妨害オブジェクトは省略しているものとする。

図 22 で用いている記号の意味を以下に示す。

Ox_t, Oy_t: 時間 t の移動オブジェクトの座標。

Mx_t, My_t: 幇助ユーザーが転送されたフレーム画像 t に対応した時のマウスカーソル座標

$\Delta t1$: 中継 PC から幇助ユーザーに CAPCHA のフレーム画像が送信されてくるまでの時間。

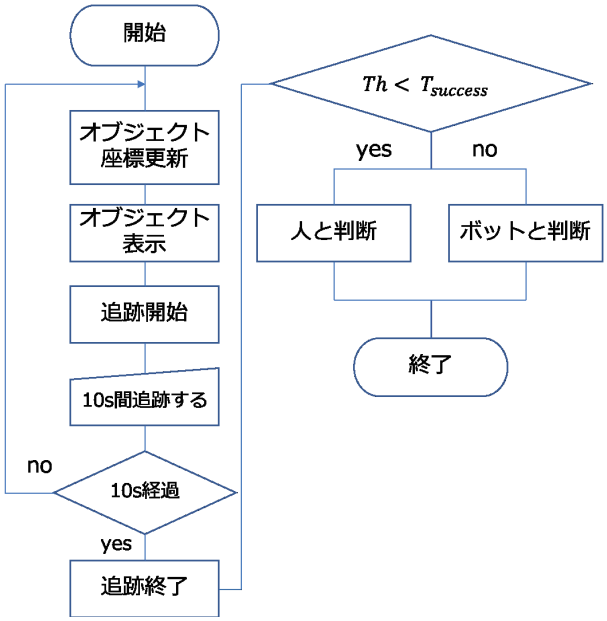

図 21. 提案 CAPTCHA の認証手順

$\Delta t2$: 幇助ユーザーから中継 PC に解答に用いるマウスカーソルの座標が送信されてくるまでの時間。

3.3.1　提案手法に対するリレーアタックの流れ

提案方式の CAPTCHA にリレーアタックを行ったときの振る舞いを以下に示す。

(1) 時刻 $t0$、移動オブジェクトの位置 (Ox_{t0}, Oy_{t0}) のフレーム画像を取得し、幇助ユーザーに送信する。(図 22 の ①)

(2) (1) から $\Delta t1$ 経った時に幇助ユーザーには、(Ox_{t0}, Oy_{t0}) に移動オブジェクトがあるように見える。

(3) 幇助ユーザーは、移動オブジェクト上にマウスカーソルを移動させる。この時のマウスカーソルの座標を (Mx_t0, My_t0) とする。この座標は、中継 PC に送信される。(図 22 の ②)

(4) マウスカーソルの座標 (Mx_t0, My_t0) は、(3) から $\Delta t2$ 経った時に、中継 PC に到着する。この時、中継 PC 上の移動オブジェクトの位置は、
座標 $(Ox_{t0+\Delta t1+\Delta t2}, Oy_{t0+\Delta t1+\Delta t2})$ まで移動している。

以上より、幇助ユーザーが目視している移動オブジェクトに対して、中継 PC 上の移動オブジェクトの位置座標は、CAPTCHA のフレーム画像の送信時間である Δt_1 の分だけ移動している。さらに、幇助ユーザーがマウスカーソルの位置座標を中継 PC に送信する時間である Δt_2 も発生するので、結果的に、幇助ユーザーが目視しているオブジェクトの座標の位置から、$\Delta t1 + \Delta t2$ の分だけ位置がずれるためリレーアタックでの追跡が困難になるはずである。

このように、オブジェクトの移動にリアルタイムで対応しなければならない方式の提案手法をリレーアタックで解くことは難しい。リレーアタックで発生する通信の遅延時間の大きさによっては、より困難になると考えられる。この遅延時間を利用してリレーアタックによる CAPTCHA 突破を防ごうというのが、提案手法のリレーアタック耐性の基本的な考え方となっている。

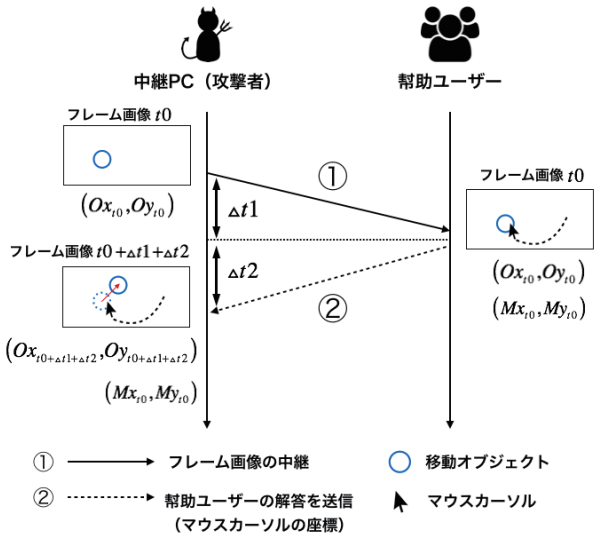

図 22. リレーアタックのシーケンス図

3.4　ボットへの耐性について

提案方式の CAPTCHA では、基本的には、移動する円形オブジェクトをマウスカーソルで追跡する解答方法をとっている。その点に着目すると、物体追跡技術を用いて移動オブジェクトを自動的に追跡する攻撃が考えられる。移動オブジェクトを自動的に追跡するためには、フレーム画像中からリアルタイムで移動オブジェクトを検出する必要がある。コンピュータによる物体追跡技術では、動画像中に映る追跡対象となる物体の特徴を捉えて、時々刻々と変化する物体の位置を推定している。

提案する CAPTCHA 方式では、妨害オブジェクトと移動オブジェクトとの間に視覚的特徴の違いがないため、追跡対象のパターンを用いて、フレーム画像中から移動オブジェクトを検出することは困難である。例えば、テンプレートマッチングのような追跡対象のパターンを用いる手法では、移動オブジェクトを追跡することは不可能である。

3.4.1　テンプレートマッチング

入力画像の中から、テンプレート（原型：template）となる画像と一致（matching）する位置を探索する処理である。一致する度合いを類似度と呼び、さまざまな類似度の計算方法が提案されている。単純な方法としては、入力画像の左上から右下に向かって走査して探索すればよい。$I_s(x,y), I_t(x,y)$ をそれぞれ座標 (i,j) における入力画像とテンプレート画像の画像値とする。また、x_s, y_s は入力画像における走査の開始位置とし、W_s、H_s は、それぞれ入力画像の幅（width）と高さ（height）とする。

$0 \leq x_s \leq W_s - 1$, $0 \leq y_s \leq H_s - 1$ の全範囲で類似度を算出し、最大もしくは最小となる位置を求めることで、検出したい対象（テンプレート画像）を探索できる。この流れを、動画のフレーム画像ごとに行うことで物体追跡が可能になる。

代表的な類似度の計算方法としては、以下の 3 つがある。

差分絶対値和（sum of absolute differences, SAD）

入力画像とテンプレート画像の差の絶対値を計算して

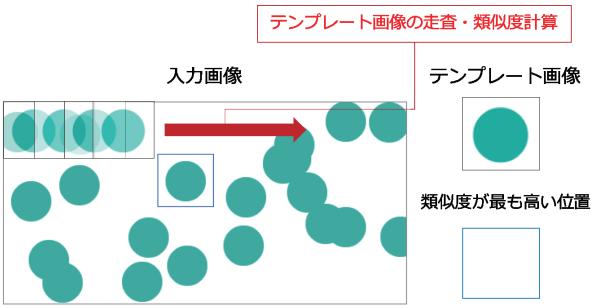

図 23. テンプレートマッチングによる移動オブジェクト追跡の試み

総和する。値が小さいほど類似度が高い。

$$S_{SAD}(x_s,y_s)=\sum_{y_t=0}^{H_t-1}\sum_{x_t=0}^{W_t-1}|I_s(x_s+x_t,y_s+y_t)-I_t(x_t,y_t)| \tag{6}$$

差分 2 乗和（sum of squared difference, SSD）

SAD の絶対値の計算は、計算コストが高いため、2 乗で代用して高速化する。SAD と同様に、値が小さいほど類似度が高い。

$$S_{SSD}(x_s,y_s)=\sum_{y_t=0}^{H_t-1}\sum_{x_t=0}^{W_t-1}\{I_s(x_s+x_t,y_s+y_t)-I_t(x_t,y_t)\}^2 \tag{7}$$

正規化差分 2 乗和（normalized sum of squared difference, NSSD）

SSD の単純な差分計算では入力画像の明度変化により類似度が変化してしまうという問題がある。そこで、入力画像とテンプレート画像をベクトルと見なして、それらのベクトルのなす角の余弦（最大値は 1）を類似度とすることで明度変化に影響されない類似度を計算できる。

$$S_{NSSD}(x_s,y_s)=\frac{\sum_{y_t=0}^{H_t-1}\sum_{x_t=0}^{W_t-1}\{I_s(x_s+x_t,y_s+y_t)-I_t(x_t,y_t)\}^2}{\sqrt{\sum_{y_t=0}^{H_t-1}\sum_{x_t=0}^{W_t-1}I_s(x_s+x_t,y_s+y_t)^2}\sqrt{\sum_{y_t=0}^{H_t-1}\sum_{x_t=0}^{W_t-1}I_t(x_t,y_t)^2}} \tag{8}$$

このように、テンプレートマッチングを行うには、追跡対象の画像（テンプレート画像）が必要になる。提案手法においては、追跡対象となる移動オブジェクトと妨害オブジェクトの視覚的特徴は同じである。図 23 のように、オブジェクトのテンプレート画像を用意し、テンプレートマッチングを試みたとしても、妨害オブジェクトを誤検出することになる。つまり、オブジェクトの視覚的特徴を用いてフレーム画像ごとに移動オブジェクトを検出し追跡することは、不可能である。

3.4.2 差分攻撃

移動オブジェクトと妨害オブジェクトの視覚的特徴を同じにするだけでは、ボットへの耐性は十分ではない。動画中から数フレームを取り出し、取り出したフレーム画像の差分を利用して移動オブジェクトの位置を把握する攻撃が考えられる。画像の差分をとることで、異なる 2 つの時刻において撮影された 2 枚の画像内で発生している変化情報を得ることができる。そのため、画像の差分は移動物体の検出に用いられ、手法として「背景差分法」や「フレーム間差分法」などが挙げられる。

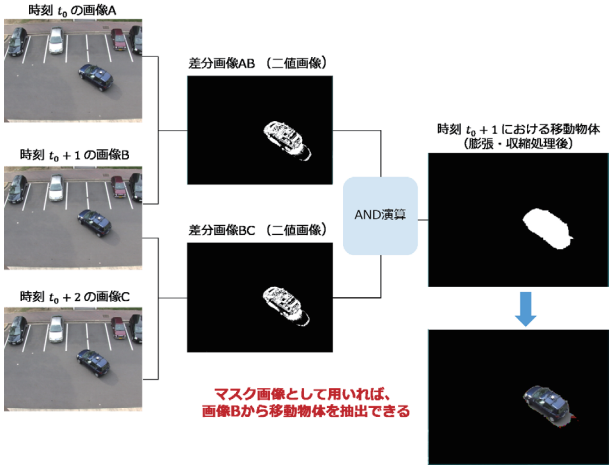

図 24. フレーム間差分法による移動物体抽出（サンプル動画像：入手元（http://www.murase.m.is.nagoya-u.ac.jp/alcon2010/?page=download[44]））

3.4.3 背景差分法

背景差分法は、入力画像と背景画像の差分を計算することで移動物体を抽出する。背景差分を行うには、背景画像を事前に用意する必要がある。入力画像 I_i と背景画像 I_b があった場合、2 枚の画像の差分の絶対値を計算し、差分画像 I_d を求める。

$$I_d(x,y)=|I_i(x,y)-I_b(x,y)| \tag{9}$$

差分画像とは、2 枚の画像において同じ位置にある画素の画素値の差の絶対値を計算する画像間演算を行って得られる出力画像のことである。次に、差分画像 I_d に対して二値化処理を行い、背景（黒色）と前景（白色）に分けたマスク画像 I_m を作成する。

$$I_m(x,y)=\begin{cases}255 & \text{if } I_d(x,y)>thresh\\ 0 & \text{if } I_d(x,y)\le thresh\end{cases} \tag{10}$$

このマスク画像 I_m を利用して、移動物体が入った画像から移動物体領域だけが切り出された画像を得ることができる。このような処理を背景差分という。

3.4.4 フレーム間差分

移動物体がない理想的な背景画像を得られないことがあった場合、移動物体を異なる 3 つの時刻において撮影した 3 枚の画像をそれぞれ、画像 A（時刻 t_0）、画像 B（時刻 t_0+1）、画像 C（時刻 t_0+2）とする。画像 A と画像 B、画像 B と画像 C の差分画像を生成して閾値処理を行い、2 枚の二値画像 AB、AC を得る。このように得られた二値画像 AB、AC の論理演算（AND 演算）を行い、これらの 2 枚の共通領域を抽出することで、時刻 t_0+1 における移動物体の領域を得ることができる。

3.4.5 考えられる攻撃と対策

視覚的特徴に違いが無い提案手法において、移動オブジェクトと妨害オブジェクトを区別するには、以下の 2 点の特徴が利用されると考えれる。

- 移動オブジェクトは連続的に移動するため連続したフレーム間での位置の変化が小さい

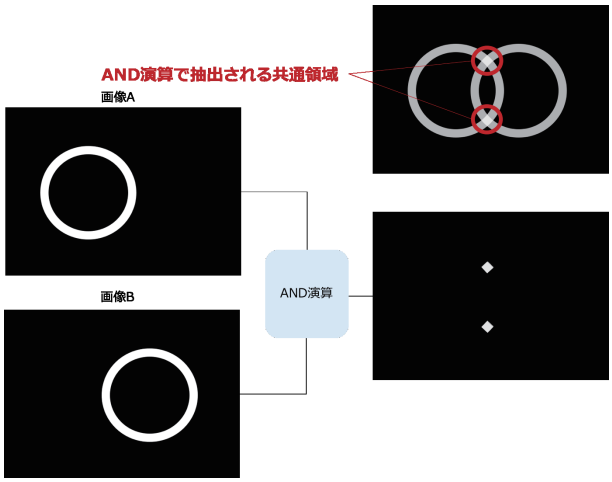

図 25. AND 演算による二値画像の共通領域の抽出

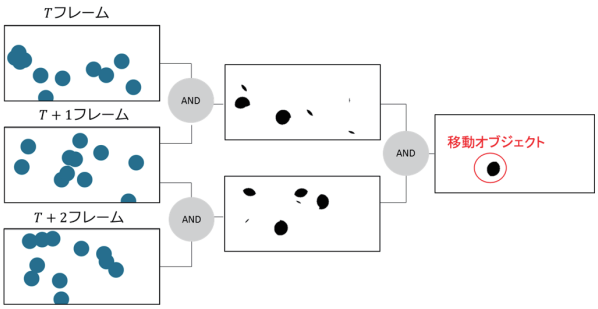

図 26. 提案手法に対する差分攻撃

- 妨害オブジェクトは 1 フレームごとにランダムな位置に出現するため、連続したフレーム間での位置の変化が大きい

上記の 2 点を考慮した場合、連続したフレーム画像を二値化した画像の論理演算（AND 演算）を繰り返すと、妨害オブジェクトを除外できるのではないかと考えられる。

ここで、画像処理のおける論理演算（AND 演算）について述べる。図 25 に示すように、2 枚の二値画像の論理演算（AND 演算）を行うと、これら 2 枚の共通領域を抽出することができる。

提案手法の特徴より、連続したフレーム間画像での論理演算を行なった場合、位置の変化が小さい移動オブジェクトは面積の大きい共通領域が取り出される。逆に妨害オブジェクトの場合、前後のフレーム画像で妨害オブジェクトの出現位置がほぼ同じにならない限り、抽出される共通面積の面積が小さい、もしくは、存在しないという結果が得られる。このように、AND 演算を行うと妨害オブジェクトを除外することができる。ただ、連続するフレーム間で 1 度、AND 演算を行うだけでは、妨害オブジェクトを完全に除外することはできない。何度か、AND 演算を繰り返すことで、妨害オブジェクトを完全に除外することができる。具体的な攻撃方法を図 26 に示す。

3 つの時系列順のフレーム画像 I_{t0}、I_{t1}、T_{t2} があった場合、I_{t0} と I_{t1} の AND 演算結果 $I_{t0andt1}$ と I_{t1} と I_{t2} の AND 演算結果 $I_{t1andt2}$ を得たとする。この時点で、妨害オブジェクトは完全に除外されていないため、$I_{t0andt1}$ と $I_{t1andt2}$ の AND 演算を行うことで、移動オブジェクトのみの共通領域が取り出される。取り出された共通領域付近に移動オブジェクトが

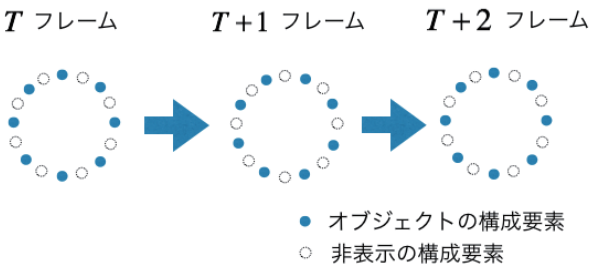

図 27. 変更したオブジェクトの設計

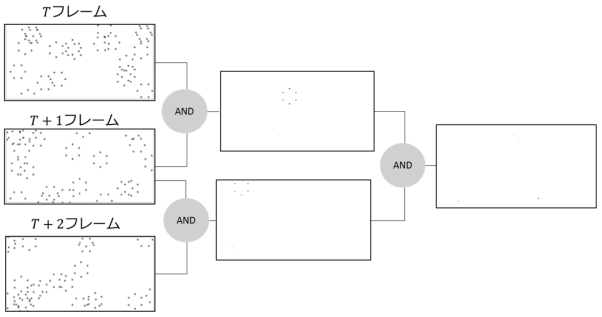

図 28. オブジェクト設計変更後の提案手法に対する差分攻撃

存在することは明確なので、位置さえ分かれば、マウスカーソルによる自動追跡は可能になる。

画像の差分を利用した攻撃によって、移動オブジェクトの位置を推定することを防ぐためには、AND 演算と行った際、妨害オブジェクトと同じように移動オブジェクトも除外されることが望ましい。これまでの円の内側を塗りつぶすオブジェクトのままでは、差分攻撃に対して脆弱であるため、連続したフレーム間での移動オブジェクトの共通領域が無くなるように設計する必要がある。

そこで、対策としてオブジェクトを図 27 のように設計した。図 27 のオブジェクトは、1 フレームごとにオブジェクトの構成要素が位置を変え、人間には回転しているように見える。各構成要素は、連続したフレーム間で位置が重ならないように位置を変えるため、共通領域が発生せず、AND 演算で移動オブジェクトの位置を推測することが困難になる。例として、設計変更後の提案手法に対する AND 演算の結果を図 28 に示す。

3.5　meanShift 法を利用した追跡手法による攻撃

物体追跡技術について、調査していたところ、meanShift 法を利用した追跡手法 [47)48)] で移動オブジェクトの追跡がある程度、可能であることが分かった。meanShift 法を利用した追跡手法は、外観がカラーヒストグラムで定義されたオブジェクトを追跡する効率的なアプローチである。まず、追跡手法の前に meanShift 法について述べる。meanShift 法は、カーネル密度推定を用いるロバストなデータ解析手法である。この手法の応用としては、画像のセグメンテーションやエッジ保存の画像の平滑化、動画中の物体のトラッキングなどへの応用例が報告される。meanShift 法は、最頻値探索問題の効率的な解法として提案された。

最頻値探索問題は、データ解析における重要な定式化手法の 1 つで、画像処理や機械学習における多くの問題にも適用されている。d 次元空間 $X \subset \mathbb{R}^d$ 上におけるベクトル点の集合 $S = \{x_i | i = 1 \cdots n\}$ が観測や実験結果のデータまたは統計的

サンプルとして与えられたとする。これらの点の集まりがどのように分布しているかを示す関数 $f(x \in X)$ は、一般に密度関数と呼ばれる。この密度関数は、空間上の任意の位置 x におけるサンプル点の密度を表し、関数値が高いところの周辺にはサンプル点が多く集まり、低いところにはサンプル点があまりないことを示す。ここでいう最頻値は、この密度関数の極大値（局所的な最大値）として定義され、対応する極大値点は、空間上で点の密度が局所的に最も高いところ、つまり、点が最も凝集している位置を指し、偏微分方程式 $\nabla x f(x) = 0$ の解に対応する。統計学的には、密度関数を離散化したものは度数分布として、密度関数の極大値と極大値点は、この度数分布のピークにおける度数と対応する事象として解釈できる。また与えられた密度関数を確率分布のモデルとみなせば、密度関数の極大値点は、最も確率の高い事象と理解できることから、この極大値が最頻値と呼ばれる。最頻値探索は、このような密度関数の極大値探索の問題で、任意の初期値 $x_0 \in X$ の近傍における密度関数の極大値を求める。この問題の解法に用いられるのが meanShift 法である。

meanShift 法を最初に提案した福永らによる定義は、meanShift 法を直感的に理解するのに適した例であるため、二次元の場合について説明する。サンプル点の集合の集合 $S = \{x_i\}$ が与えられたとする。次に任意の点 μ を中心として半径 h の円領域 $T_h(\mu)$ を考え、この円領域内にあるサンプル点の平均値（重心） $m(\mu)$ を計算する。まず、円領域内で平均（重心）を計算し、そこに現在位置を更新する手続きを繰り返し行うことで、初期値近傍の最頻値点に収束させる。福永の meanShit 法での平均値（重心）$m(x_j)$ は、次式のように定義される。

$$m(x_j) = \frac{1}{|T_h(x_j)|} \sum_{x \in T_h(x_j)} x_i \tag{11}$$

ここで、$|T_h(x)|$ は x を中心とする円領域内にあるサンプル点の数を示す。

次に、円領域 $T_h(\mu)$ を次のような関数 $K_f(x;\mu,h)$ として数式で表す。

$$K_f(x;\mu,h) = \begin{cases} 1 & \text{if } ||x-\mu|| \le h \\ 0 & \text{if } ||x-\mu|| > h \end{cases} \tag{12}$$

これは、カーネルと呼ばれる関数であり、上式以外の関数に拡張することができる。上式は、閉領域内の均一分布を表すフラットカーネルと呼ばれる。定数 h は、一般的にカーネル幅と呼ばれる。式 (11) は、このカーネルによる重み付き平均を使って、次のように再定義できる。

$$m(x) = \frac{\sum_{i=1}^{n} K_f(x_i;x,h)x_i}{\sum_{i=1}^{n} K_f(x_i;x,h)} \tag{13}$$

ここでは、数列和の範囲がサンプル S 全体に及ぶ事に注意する。任意のパラメーターは円領域の半径 h のみとなり、高次元の場合でも、同じ半径の球体を考えれば容易に拡張できる。

この福永らの提唱した meanShift 法は、Cheng らによって拡張され、動画像におけるトラッキングへの応用について重要な役割を果たすようになった。Cheng らは、福永の定義による meanShift 法を次の 3 点について拡張した。

1. 一般的なカーネル関数への拡張
2. サンプル集合に対する重み関数の導入
3. サンプル集合を固定し任意の初期値からの探索

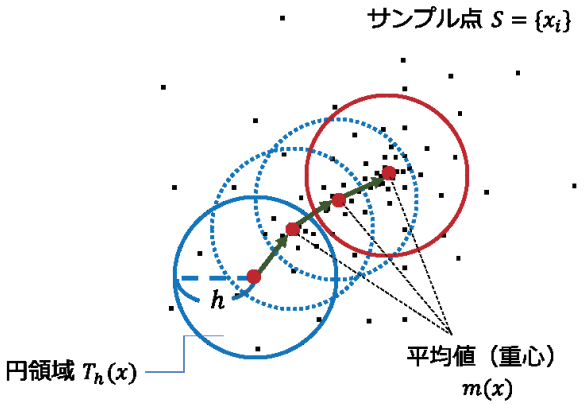

図 29. meanShift 法の概略図

フラットカーネル以外のガウス関数なども含む一般的カーネルを $K(x;\mu,h)$ で、また非負値の重み関数を $w(x) : S \to (0,\infty)$ で表すと、サンプル x_i の重み付き平均が次のように定義できる。

$$m(x) = \frac{\sum_{i=1}^{n} K_f(x_i;x,h)w(x_i)x_i}{\sum_{i=1}^{n} K_f(x_i;x,h)w(x_i)} \tag{14}$$

式 (14) は、導入された重み関数以外は、式 (13) と同じ式となる。この重み関数が、動画像におけるトラッキングへの応用に重要な役割を果たす。

ここで、meanShift 法を利用した物体追跡手法の流れを説明する。

3.5.1 追跡の開始

最初に追跡対象のカラーヒストグラムを計算する。

1. 追跡ウィンドウの位置とサイズを設定（追跡ウィンドウは、図 29 での半径 h の円）
2. 追跡ウィンドウ内のカラーヒストグラムを計算

3.5.2 追跡ウィンドウの更新

動画のフレーム画像ごとに以下の処理を行う。

1. 追跡対象の 2 次元確率分布を計算する。
2. 追跡ウィンドウ内の確率分布の重心を求める。
3. 追跡ウィンドウの中心が重心の位置に来るように追跡ウィンドウを更新。
4. 収束するまで 2、3 を繰り返す。

2 次元確率分布は、追跡対象のカラーヒストグラムを動画のフレーム画像に逆投影（back projecion）する事で求めることができる。逆投影法は、各画素が対象物体に属している確率を表す画像を作成することである。画像の領域分割や画像中の対象物体の検出に用いられる。図**??**に逆投影法の概略図を示す。対象物体に属している確率は、0〜255 に正規化している。（逆投影画像は、グレースケール）

逆投影画像を作成した後、meanShit 法のアルゴリズムを適用する。追跡ウィンドウ内の確率分布の重心を計算し、その重心に、追跡ウィンドウの中心が来るようにウィンドウの位

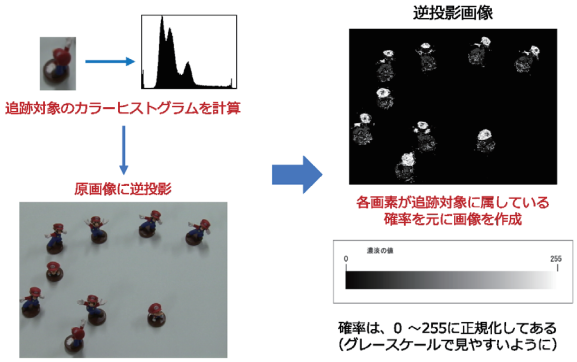

図 30. 逆投影法 [44)]

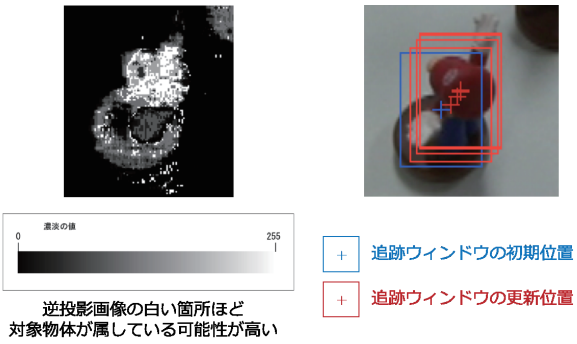

図 31. meanShift 法の実行例

置を更新する。この処理を動画の 1 枚のフレーム画像に対して繰り返し行う。この反復処理は、既定回数に達するか、追跡ウィンドウの移動量が小さくなるまで繰り返す。図 31 に、meanShift 法の実行例を示す。これは、OpenCV（オープンソースのコンピュータビジョン向けライブラリ）を用いて実装した。図 31 の実行例では、既定回数を 10 回、もしくは、追跡ウィンドウの移動量が 1px 以下になれば反復処理を終了するように設定している。このように、各フレームごとに追跡ウィンドウの中心を確率分布の重心に移動させることで、対象物体の追跡を行う。

3.5.3　提案手法への攻撃と対策

図 32 に提案手法の逆投影画像と追跡ウィンドウの移動を示す。それぞれ連続するフレーム画像であり、追跡ウィンドウが移動オブジェクトを捉えた時のものである。逆投影画像は、オブジェクトのカラーヒストグラムを元に作成する。移動オブジェクトも妨害オブジェクトも全く同じ色をしているため、図 32 のようになり、逆投影画像（グレースケール）において、両オブジェクトは画素値が 255（白）になる。従って、逆投影画像の作成時点では、移動オブジェクトの特定には至っていない。つまり、追跡ウィンドウは妨害オブジェクトによって妨害が可能であり、追跡ウィンドウを騙すことができると考えられる。

しかし、移動オブジェクトが追跡ウィンドウの範囲内に入ってしまうと、移動オブジェクトの特徴（フレーム間での移動量が小さい）から、次フレームにおいても追跡ウィンドウの範囲内に移動オブジェクトが存在している可能性が高い。反対に、妨害オブジェクトはフレームごとにランダムに位置を変えるため、追跡ウィンドウの領域内に妨害オブジェクトが入ることはあるが、次フレームで妨害オブジェクトが必ず入るとは限らない。これらのことは meanShift 法による移動オブジェクトの追跡の精度を高め、ある程度の追跡が可能になってしまう。図 33 に、自動追跡を行なった際のフレーム画像を示している。追跡成功時間を表す赤色のメーターが徐々に伸びており、自動的に追跡できていることが分かる。検証の結果、この攻撃は 3.4.2 節で差分攻撃への対策を施した設計に対しても有効であることを確認した。

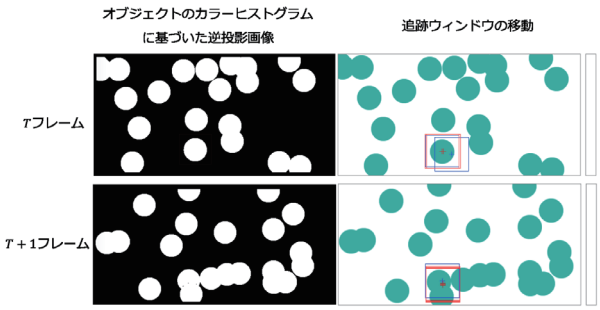

図 32. 提案手法の逆投影画像と追跡ウィンドウの移動

攻撃の対策として、移動オブジェクトが追跡ウィンドウの領域から出るほど移動量を大きくして追跡を困難にすることも考えられるが、人間に追跡が難しくなってしまうので、それは避けたい。そこで、移動領域内に妨害オブジェクトが入る頻度を多くすることで、移動オブジェクトを捕捉した追跡ウィンドウが、誤って妨害オブジェクトを捉えてしまうように仕向けて、追跡精度を低くできないかと考えた。妨害オブジェクトが移動領域内に入る頻度を増やすには、単純に妨害オブジェクトを増やすことが考えられる。だが、妨害オブジェクトを増やすことは、人間にとって見づらいものになる可能性があるため、CAPTCHA 実用化のためには、妨害オブジェクトを増加させた時の「ボットの追跡精度」と「人間の追跡精度」を確認しなければならない。この検証実験については、次章で述べる。

4.　評価実験と考察

本章では、提案手法の CAPTCHA 方式に対する攻撃を実行し、安全性に関する評価・考察を行う。第??章で、実行される可能性のある攻撃について述べたが、「テンプレートマッチング」や「差分攻撃」については、耐性がある、もしくは、対策を施したことで耐性を得たことが分かっているため、ここでは評価しない。以下、4.1 節と 4.2 節でリレーアタック耐性とボット耐性について述べる。

4.1　リレーアタック耐性の検証実験

4.1.1　実験目的

提案手法の CAPTCHA に対して、リレーアタックを行い、リレーアタックで生じる遅延時間によって、CAPTCHA の解答が困難になるかを検証する。

4.1.2　実験方法

実験は、10 名の実験参加者（21 歳～25 歳）に、正規アクセスで提示された CAPTCHA とリレーアタックで提示された CAPTCHA をそれぞれ 5 回ずつ解いてもらい、移動オブジェクトの追跡成功時間を計測した。なお、補助ユーザーと中継 PC 間の CAPTCHA のフレーム画像の転送と幇助ユーザーの解答の送信に掛かる時間は、両者間の通信環境に依存す

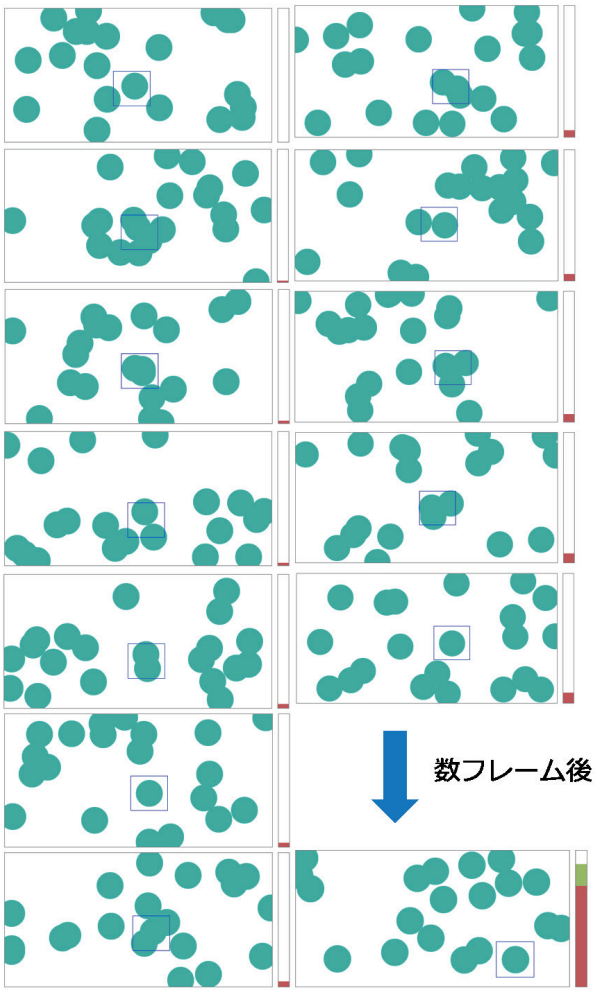

図 33. 提案手法への meanShift 法による攻撃

るため、CAPTCHA を中継する中継 PC とリレーアタックで CAPTCHA の解答を行う幇助ユーザー PC の通信に異なる遅延時間を発生させて、リレーアタックを行い、それぞれの遅延時間が発生した時の幇助ユーザーの追跡成功時間を計測した。その結果から、幇助ユーザにとって提案手法の CAPTCHA が困難なものになり得るか検証する。

4.1.3 実験環境

リレーアタックは、VirtualBox を利用し、仮装環境上で再現した。VirtualBox は、既存のオペレーティング・システム（ホスト OS）上にアプリケーションの一つとしてインストールされ、この中で追加のオペレーティング・システム（ゲスト OS）を実行することができる。実験では、中継 PC と幇助ユーザー用の PC をゲスト OS として用意し、2 つのゲスト OS 間で CAPTCHA の中継を行った。CAPTCHA の中継には文献 [37] でリレーアタックを再現するためのソフトウェアとして利用されていた VNC（Virtual Network Computing）を用いた。VNC は、ネットワークを通じて接続された他のコンピュータの画面を遠隔操作できるソフトウェアである。遅延時間の発生には、VyOS[45] を用いた。VyOS は、オープンソースで開発されているネットワーク OS であり、主にソフトウェアルータとして運用される。中継 PC と幇助ユーザ PC 間の RTT（Round - Trip Time）を約 50ms、100ms、200ms になるように設定し、これらに加え、遅延の設定を無しの状

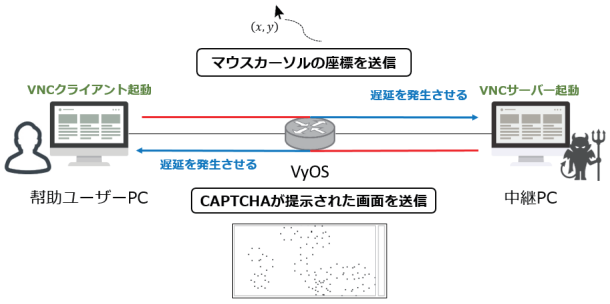

図 34. 実験環境

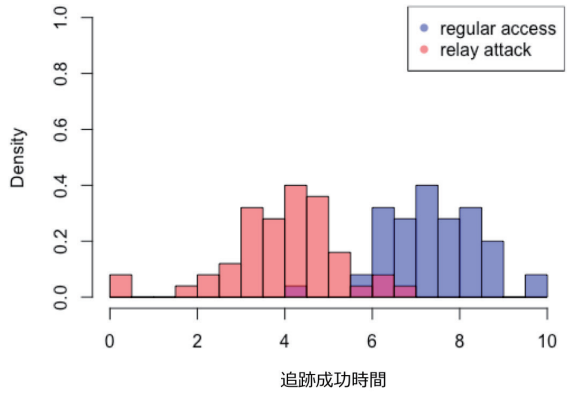

図 35. 正規アクセスとリレーアタック（遅延の設定無し）の追跡成功時間

態での実験を行った。計 4 パターンの通信環境でのリレーアタックを行い、追跡成功時間のデータを収集した。実験環境の詳細は、以下の通りである。

中継 PC、幇助ユーザー PC（ゲスト OS） ： Ubuntu 16.04 LTS

VNC サーバー ： Vino

VNC クライアント ：Remmina（色数は「256 色」、品質は「最高」に設定）

CAPTCHA のパラメータ ：妨害オブジェクトの数（20 個）、移動オブジェクトの速さ（毎フレーム、0.2 ピクセル ～ 7.0 ピクセルの移動量）、フレームレート（60fps）

4.1.4 実験結果と考察

本実験で、観測された「正規アクセスでの追跡成功時間」と「4 パターンの通信環下でのリレーアタックの追跡成功時間」の結果を図 35、図 36、図 37、図 38 に示す。

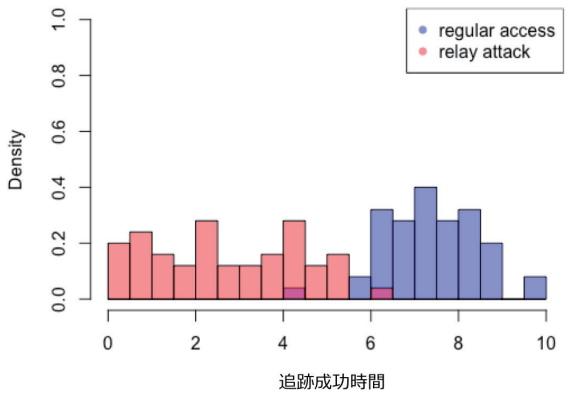

図 36. 正規アクセスとリレーアタック（50ms の遅延）の追跡成功時間

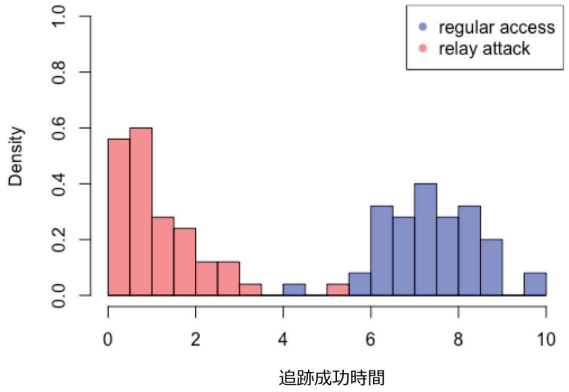

図 37. 正規アクセスとリレーアタック（100ms の遅延）の追跡成功時間

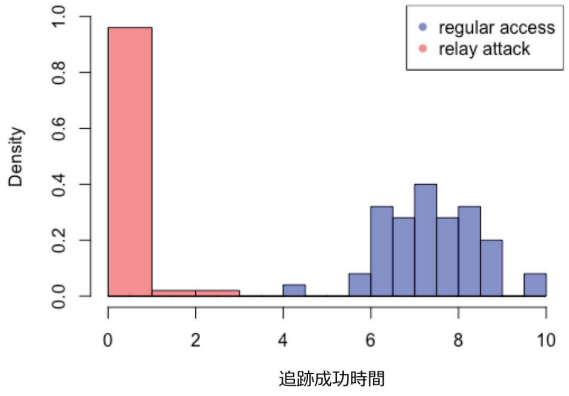

図 38. 正規アクセスとリレーアタック（200ms の遅延）の追跡成功時間

得られた追跡成功時間のデーターから、幇助ユーザーにとって達成が難しい追跡成功時間の閾値を検討する。まず、観測されたデータを元に、確率分布を推定することにした。今回は、正規アクセスと遅延設定なし、遅延時間 50ms の追跡成功時間をを正規分布であると仮定し、正規性の検定を行なった。遅延時間 100ms、200ms のリレーアタックの観測データに関しては、ヒストグラムから正規性は見られないと判断したため、検定は行わなかった。

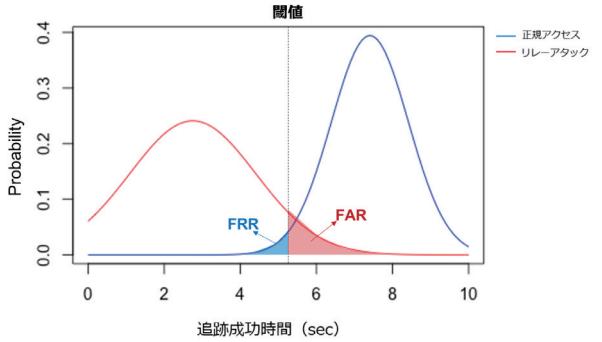

図 39. 正規アクセスと遅延時間 50ms のリレーアタックでの追跡成功時間の正規分布

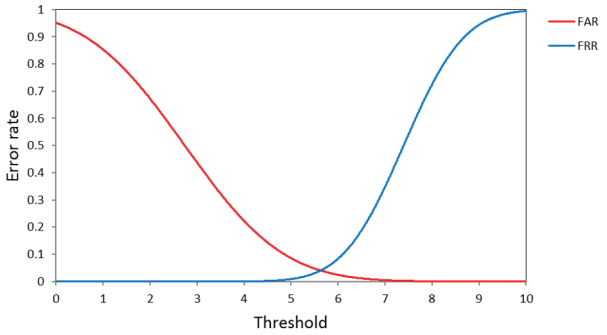

図 40. 幇助ユーザー受容率（FAR：False Acceptance Rate）と正規ユーザー拒否率（FRR：False Rejection Rate）

正規性の検定は、シャピロウィルク検定 [46] を用いる。この手法では、「標本分布は、正規分布に従う」という帰無仮説を検定する。検定結果の指標は、p 値（有意確率）を用い、有意水準 5%で検定を行うため、$p > 0.05$ となれば、正規分布に従うと判断できる。検定結果を表 2 に示す。

表 2. 追跡成功時間の正規性の検定の結果

	p 値
正規アクセス	0.2728
遅延設定無し	0.0667
遅延時間 50ms	0.0626

検定の結果、正規アクセスの追跡成功時間と遅延無し、遅延時間 50ms のリレーアタックの追跡成功時間は、正規分布に従うことが分かった。今回は、正規アクセスと遅延時間 50ms のリレーアタックの追跡成功時間のデータを正規分布で近似し、その正規分布から、幇助ユーザー受容率（FAR：False Acceptance Rate）と正規ユーザー拒否率（FRR：False Rejection Rate）を算出して、閾値を検討する。

正規アクセスと遅延時間 50ms のリレーアタックの追跡成功時間の正規分布は、図 39 に示す。この正規分布から、算出した FAR と FRR については、図 40 のような結果が得られた。この結果によると、FAR と FRR が等しくなる等価エラー率（EER：Equal Error Rate）は、約 4%であり、その時の追跡

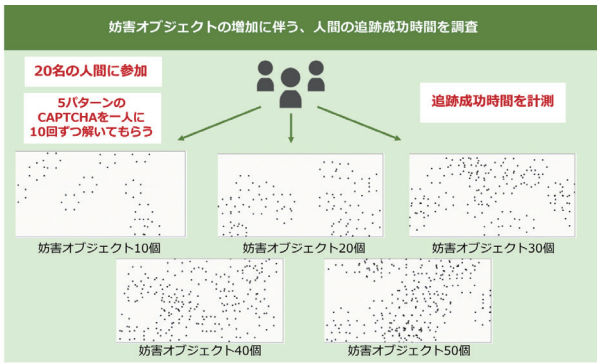

図 41. 実験の概略図（人間のデータ収集）

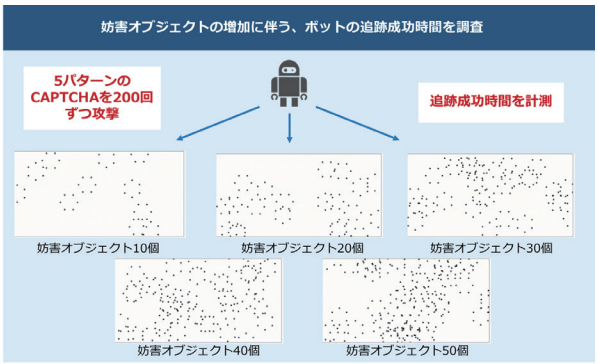

図 42. 実験の概略図（ボットのデータ収集）

```
int cv::meanShift (
    InputArray probImage,
    Rect &window,
    TermCriteria TermCriteria
)
```

引数1：入力画像（逆投影画像）
引数2：探索窓の大きさ・初期位置
引数3：反復探索アルゴリズムの停止基準

図 43. meanShift の関数仕様

```
for(int iter = 0; iter < 反復回数; ++iter){
    1. 重心の計算

    2. 探索窓の位置を更新

    if(探索窓の移動量 < 設定した移動量) break;
}
```

図 44. 探索窓の位置更新のプログラム

成功時間は、5.63 秒であった。ここで、CAPTCHA として実用的な閾値を考える。文献[49]によれば、一般的に利用されている google の re-CAPTCHA の平均成功率は、97%である。今回の提案手法の CAPTCHA に同程度の正規ユーザーの成功率を持たせたい場合は、FAR が 3%になるように閾値を設定する。この時の追跡成功時間の閾値は、約 5.5 秒であり、FAR は、約 5%になる。幇助ユーザーは、CAPTCHA1000 個の解読につき報酬を得ているが、FAR が 5%であれば、1000 個解いたとしても、このうち成功するのは、50 個程度となるため、リレーアタックは経済的に成り立たなくなる。

また、100ms、200ms と遅延が大きくなるにつれて、幇助ユーザーの追跡成功時間が短くなっていることから、ここで設定した閾値は、50ms より大きい遅延が発生するリレーアタックに対しても有効であると考えられる。しかしながら、CAPTCHA の中継に用いたソフトウェアの性能を無視することはできない。利用するソフトウェアの性能によっては、遅延時間の影響が少なくなる可能性があるため、今回用いたソフトウェア（VNC）より優れた性能のソフトウェアを利用したリレーアタックの検証実験を行う必要がある。

4.2 ボット耐性の検証実験

4.2.1 実験目的

ボットが提案手法を自動的に突破しようとする場合、現段階で、最も有効な手法が「meanShift 法を用いた物体追跡」だと考えられる。第??章でも述べたが、この攻撃への対抗策として、妨害オブジェクトの数を増やすことにした。

本実験では、妨害オブジェクトの数の増加による難読化が「ボット」と「人間」に与える影響を調査し、「ボット」と「人間」を分ける適切な追跡成功時間の閾値について考察、実行した攻撃に対する安全性を評価する。

4.2.2 実験方法

4.2.3 人間のデータの収集

20 名の実験参加者 (21 歳〜24 歳) に提案手法の CAPTCHA を解いてもらい、移動オブジェクトの「追跡成功時間」を測定する。CAPTCHA は、「差分対策」を施したもので、妨害オブジェクトの数が 10 個、20 個、30 個、40 個、50 個の 5 パターンを、それぞれ 10 回ずつ解いてもらった。最終的には、各パターンにつき 200 個の「人間の追跡成功時間」のデータが集められた。

4.2.4 ボットのデータ収集

「meanShift 法を用いた物体追跡」のプログラムを実装し、自動的にマウスカーソルで移動オブジェクトを追跡するボットを作成した。ボットの詳細は、4.2.5 節で述べる。実装したボットを使用して、5 パターンの CAPTCHA に対し、200 回ずつ攻撃を行い、「追跡成功時間」を測定した。最終的には、各パターンにつき 200 個の「ボットの追跡成功時間」のデータが集められた。

4.2.5 ボットの実装について

ボットのプログラムは、画像処理ライブラリの OpenCV[50]を用いて実装した。OpenCV には、meanShift 法のアルゴリズムが実装されているため、それを利用する。

OpenCV での meanShift 法の関数仕様は、図 43 のようになっている。移動オブジェクトの追跡精度の影響を与えるのは、引数 2 と引数 3 であるため、この 2 つを攻撃者に最も有利なパラメーターで設定する。引数 2 で設定する探索窓は、大きすぎると妨害オブジェクトの影響も大きくなってしまうため、移動オブジェクトがちょうど入る程度の大きさの正方形に設定した。(縦：70px、横 70px) 探索窓の初期位置に関しては、CAPTCHA 開始時点の移動オブジェクトの位置がランダムになっており、推測が難しいため、CAPTCHA のオブジェクト描画領域の中央に設定しておくことにした。

引数 3 では、探索窓の位置の更新の停止基準を設定する。探索窓の位置の更新は、反復回数が既定回数に達するか、移動量が小さくなるまで繰り返される。具体的なプログラムの例を図 44 に示す。

第??章でも述べたが、meanShift 法では、探索窓の領域内の確率分布の重心を計算し、その位置に探索窓の中心が来るように位置を更新する。これをフレーム画像ごとに繰り返す

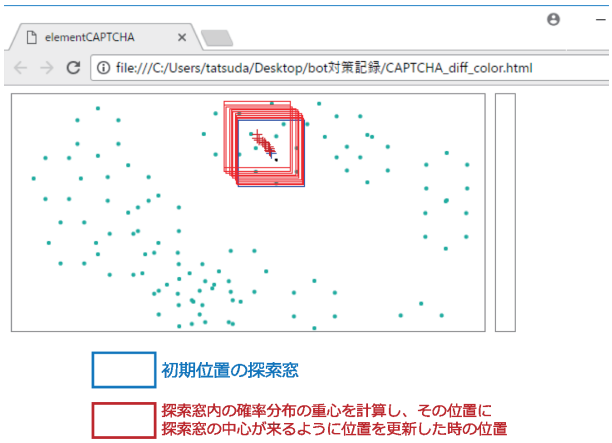

図 45. 反復回数：10 回、移動量：1px 以下

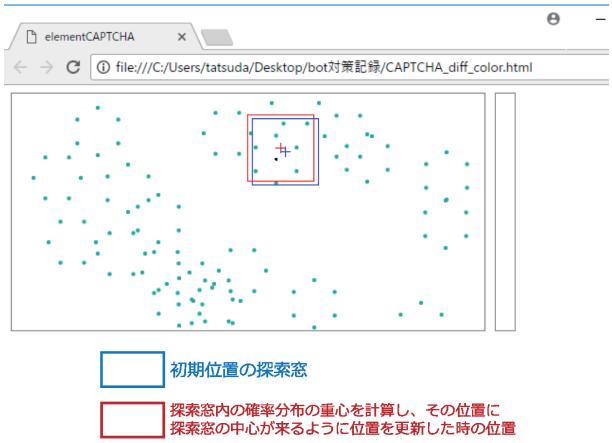

図 46. 反復回数：1 回、移動量：1px 以下

ことで、物体の追跡を行う。確率分布を求めるため作成される逆投影画像では、追跡対象である可能性が高い領域ほど白色で表示される。これは、カラーヒストグラムに基づいて作成されるが、提案手法において、追跡対象である移動オブジェクトの色と妨害オブジェクトの色は同じであるため、逆投影画像を作成した際、妨害オブジェクトも追跡対象である可能性が高いと判定される。

探索窓が移動オブジェクトを補足していた状況を考えると、移動オブジェクトの周囲に妨害オブジェクトが複数、出現した場合、探索窓内の重心の計算は、周囲の妨害オブジェクトに影響を受け、補足している移動オブジェクトから離れてしまう可能性がある。探索窓の位置の更新が多いほど、移動オブジェクトから離れる恐れがあるため、攻撃者にとって、1 フレームごとの探索窓の位置の更新は、少ない方が良い。

図 45 に引数 3 のパラメータを変えた際の探索窓の位置更新の例を示している。図 45、46 の移動オブジェクトには、6 個の構成要素の中央に黒点を描いている。

反復回数を多めに設定した、図 45 の方が、移動オブジェクトから探索窓が離れていく様子が分かる。よって、今回の実験では、攻撃者に最も有利なパラメータで攻撃を行うために、引数 3 の反復回数を 1 回に設定することにした。反復回数が 1 回であるため、図 44 から分かるように、探索窓の移動量の設定は、考慮する必要はない。

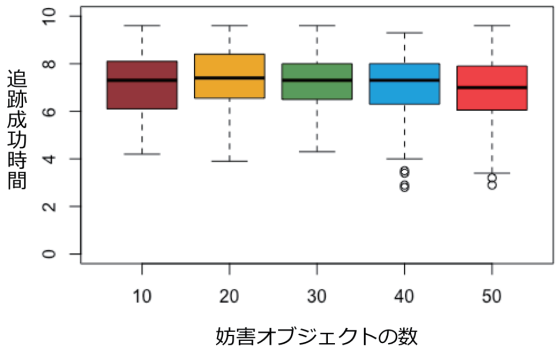

図 47. 人間の追跡成功時間

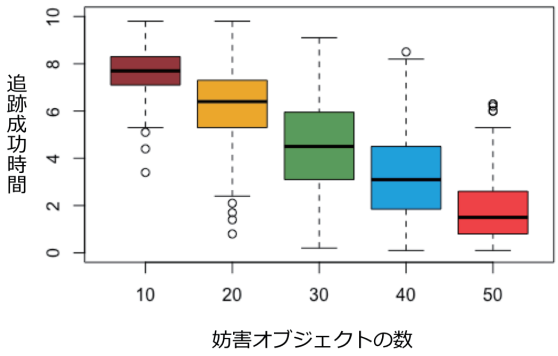

図 48. ボットの追跡成功時間

4.3　実験結果

本実験で、収集した追跡成功時間のデータを図 47 と図 48 に示す。まず、図 47 と図 48 では、妨害オブジェクトの数を変化させた時の追跡成功時間の箱ひげ図を示している。第 1 四分位点（中央値より小さい値に限定したデータの中央値）から第 3 四分位点（中央値より大きい値に限定したデータの中央値）までの高さに箱を描き、箱から上下に伸びている黒線（ひげ）は、それぞれ最大値、最小値を示している。白丸で描画しているのは、外れ値とみなされたデータであり、ここでは、第 3 四分位数 + 1.5 ×四分位範囲 を超えた値と第 1 四分位数 − 1.5 ×四分位範囲 を下回る値を外れ値とみなしている。

図 47 と図 48 を見ると、妨害オブジェクトの増加に伴い、ボットの追跡成功時間は低下していき、人間の追跡成功時間は、変化がほとんどない。このことから、妨害オブジェクトの増加によるボットへの影響は大きく、人間への影響は少ないことが分かった。

4.4　ボットに対する安全性の評価・考察

この節では、4.3 節で得られたデータに基づいて安全性を評価し、考察を行う。CAPTCHA は、コンピュータ（ボット）にとって達成が難しいタスクであることが望ましいため、「ボット」にとって達成が難しい追跡成功時間の閾値を検討する必要がある。観測されたデータを元に確率分布を推定し、ボット受容率（FAR：False Acceptance Rate）と人間拒否率（FRR：

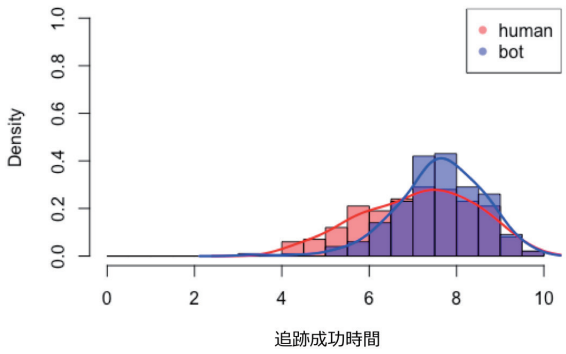

図 49. 人間とボットの追跡成功時間（妨害オブジェクト 10 個）

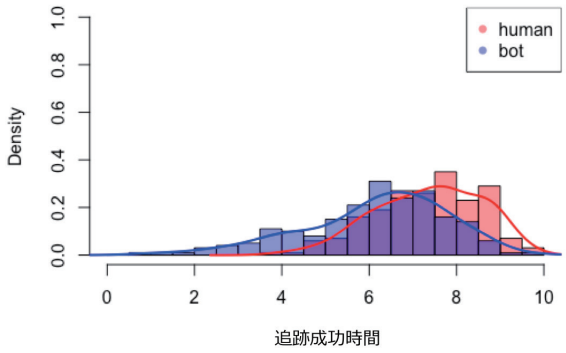

図 50. 人間とボットの追跡成功時間（妨害オブジェクト 20 個）

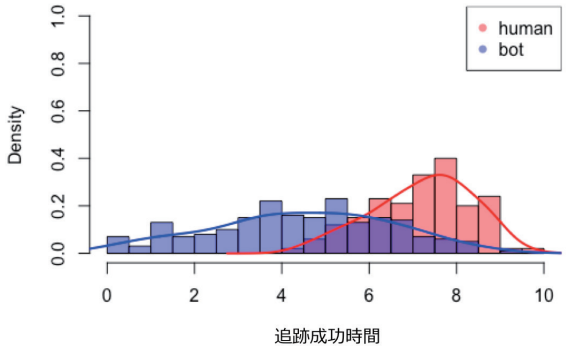

図 51. 人間とボットの追跡成功時間（妨害オブジェクト 30 個）

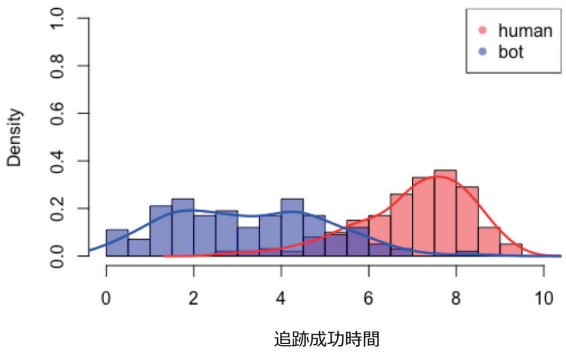

図 52. 人間とボットの追跡成功時間（妨害オブジェクト 40 個）

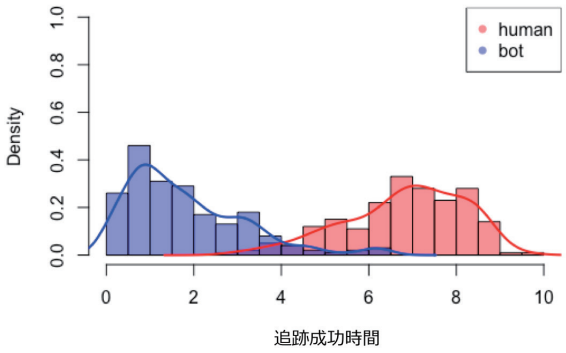

図 53. 人間とボットの追跡成功時間（妨害オブジェクト 50 個）

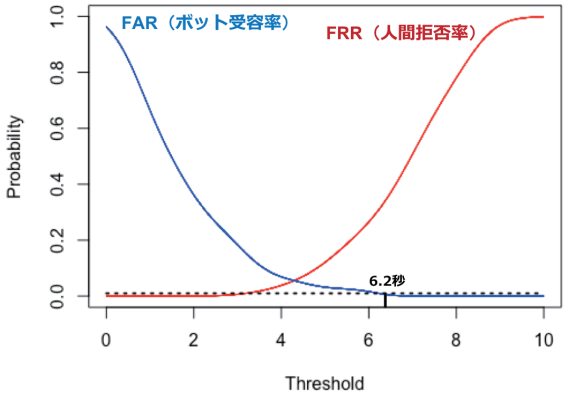

図 54. FAR：ボット受容率と FRR：人間拒否率（妨害オブジェクト 50 個）

False Rejection Rate) を算出した。確率分布の推定には、カーネル密度推定を用いた。4.1 節と同じように、シャピロ・ウィルク検定で正規性の検定を行ったが、人間のデータ、ボットのデータの両方とも正規分布に従うとはいえないという結果だったため、カーネル密度推定を用いて、確率分布を算出した。結果を、図 49、図 50、図 51、図 52、図 53、図 54 に示す。図には、収集したデータのヒストグラムと、算出した確率密度曲線を描画している。

妨害オブジェクト 50 個の場合の、追跡成功時間の閾値を考える。CAPTCHA のセキュリティ目標として、ボットの突破率は、0.01%を超えてはいけないとされている [51)52)]。図 54 の FAR（ボット受容率）が 0.01%になるように閾値を設定した場合、追跡成功時間の閾値は、約 6.2 秒となる。この閾値に設定した場合、CAPTCHA のセキュリティ目標は達成できるが、この時の FRR（人間拒否率）は、約 30%になるため、ユーザビリティが低下する可能性がある。

CAPTCHA のセキュリティ目標を達成しつつ、ユーザビリティを低下させないためには、妨害オブジェクトをもう少し増やすことが考えられる。図 47 と図 48 で示したように、妨

害オブジェクトの増加に伴う追跡精度の低下は、ボットには見られたが、人間にはあまり見られなかった。ただ、妨害オブジェクト 50 個ともなると、移動オブジェクトを見つけるのに、少し時間のかかる人もいたため、全く人間に影響がないわけでない。セキュリティを保ちつつ、人間のユーザビリティを低下させないような CAPTCHA のパラメータの設定が要求され、今後の課題と言える。

5. まとめ

本稿では、現在の CAPTCHA のセキュリティ問題を取り上げ、CAPTCHA に対する攻撃の代表例である「リレーアタック」と「ボットによる CAPTCHA の脆弱性を突いた自動的な攻撃」への耐性を両立するための CAPTCHA 方式についての研究に取り組んだ。「リレーアタック」と「ボット」への耐性を持たせるために必要な要件について論じ、その要件を満たす CAPTCHA 方式を提案した。提案手法は、連続的に移動するオブジェクトをマウスカーソルで追跡し、その追跡が成功していた時間で「人間」か「ボット」かを判別する。この手法は、2 つの脅威に対抗するために必要な要件、「リレーアタックを実行した際に生じる通信の遅延時間の利用」と「ボットが出題されたタスクを簡単にこなすことができないような難読化を施す」を満たせるように設計した。

さらに、リレーアタックをシミュレートするための実験環境を構築し、提案手法に対してリレーアタックを行なった。その結果、リレーアタックに対して、ある程度の耐性を持つことを確認した。また、提案手法に対して現在最も有効だと考えられる、画像処理に基づいた攻撃を実装し、その攻撃への耐性を評価した。この評価から、難読化の度合いを強めることで実装した攻撃への耐性は強化されることが分かったが、ユーザビリティを低下させる恐れがある。

今後の課題として、提案手法のユーザビリティ評価や難読化の強化によるユーザビリティへの影響を調査していく必要がある。また、様々な年齢層や身体的にハンデのある人々のユーザビリティ評価も行い、提案手法の適切なパラメータ決定が必要だと考えられる。

参考文献

1) L. Von Ahn, M. Blum, N. Hopper, and J. Langford, “Telling humans and computers apart automatically,” Comm. ACM, Vol.47, pp.50-60, 2004.
2) “Inaccessibility of CAPTCHA,”https://www.w3.org/TR/turingtest/,（accessed 2018-01-15）.
3) L. Von Ahn, M. Blum, N.Hopper, and J. Langford, “CAPTCHA: Using hard AI problems for security,” In Proceedings of Eurocrypt, Vol.2656, pp. 294-311, 2003.
4) “Microsoft アカウント,”https://signup.live.com/signup?uaid=b5cfce63531041ef804393f97a8b9509lic=1, （accessed 2017-05-14).
5) “YahooJapan アカウント,”https://account.edit.yahoo.co.jp/ registration?.src=www.done=https%3A%2f%2fwww.yahoo.co.jp,（accessed 2017-12-26）.
6) “Wikipedia アカウント,”（accessed 2017-12-26）.
7) J. Yan, and A.S. El Ahmad, “Breaking visual CAPTCHAs with naive pattern recognition algorithms,” the 23rd Annual Computer Security Applications, pp. 279-291, IEEE computer society, 2007.
8) K. Chellapilla, and P.Y. Simard, “Using machine learning to break visual human interaction proofs（HIPs）,” Advances in Neural Infomation Processing Systems, vol.17, pp.265-272, 2005.
9) Gimpy、Carnegie Mellon University (online),http://www.captcha.net/captchas/gimpy/, (accessed 2018-01-15).
10) OGDEN’sBASICENGLISH (online),http://ogden.basic-english.org/, (accessed 2018-01-15).
11) G. Mori, and J. Malik, “Recognizing Objects in Adversarial Clutter:Breaking a Visual CAPTCHA,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol.1, pp.134-144, 2003.
12) L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, “reCAPTCHA: Human-based character recognition via Web security measures,” Science, vol.321, No.5895, pp.1465-1468, 2008
13) reCAPTCHA, Google (online), https://www.google.com/recaptcha/intro/, (accessed 2018-01-21).
14) D. George, W. Lehrach, K. Kansky, M. L?zaro-Gredilla, C. Laan, B. Marthi, and A. Lavin, “A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs,” Science, Vol.358, eaag2612, 2017.
15) PIX, Carnegie Mellon University (online),http://www.captcha.net/captchas/pix/, (accessed 2018-01-15).
16) M. Chew, and J. D. Tygar, “Image recognition captchas,” International Conference on Information Security, pp.268-279, Springer, 2004.
17) Assira, http://research.microsoft.com/en-us/um/redmond/projects/asirra/, (accessed 2018-01-15).
18) PetFinder, PetFinder.com (online), http://www.petfinder.com/, (accessed 2018-01-29).
19) 飯田貴章、藤本衡、“ASIRRA システムに対する機械的な突破の検証、” 日本ソフトウェア科学会大会論文集、Vol.31、 pp.473-479、2014.
20) NuCAPTCHA:NuCAPTCHA (online), http://www.nucaptcha.com/ (accessed 2018-01-15).

21) NuCaptcha takes on Google's ReCaptcha, clnet (online), http://news.cnet.com/8301-19882 3-20091006-250/nu captcha-takes-on-googles-recaptcha/, (accessed 2018-01-15).

22) Y. Xu, G. Reynaga, S. Chiasson, J. M. Frahm, F. Monrose, and P. C. van Oorschot, "Security and Usability Challenges of Moving-Object CAPTCHAs: Decoding Codewords in Motion," USENIX security symposium, pp.49-64(2012).

23) 森 拓真、宇田 隆哉、菊池 眞之、"アモーダル補完を利用した動画 CAPTCHA の提案、" マルチメディア、分散協調とモバイルシンポジウム 2011 論文集、pp.1518-1525、2011.

24) 3D-based Captchas become reality, clnet (online), https://www.cnet.com/news/3d-based-captchas-become-reality/, (accessed 2018-01-15).

25) 藤田真浩、池谷勇樹、可児潤也、米山裕太、西垣正勝、"高度なメンタルローテーションを利用した画像 CAPTCHA の提案、" 電子情報通信学会技術研究報告、Vol.114、No.83、pp.7-12、2014.

26) 藤田真浩、池谷勇樹、可児潤也、西垣正勝、"Locimetric 型メンタルローテーション CAPTCHA、" 情報処理学会論文誌、 Vol.57、 No.9、pp.1954-1964、2016.

27) 佐野 絢音、藤田 真浩、西垣 正勝、"Spatiometric 型メンタルローテーション CAPTCHA の提案、" 暗号と情報セキュリティシンポジウム (SCIS) 予稿集、3C2-1、2016.

28) 藤田 真浩、池谷 勇樹、西垣 正勝、"全周囲型メンタルローテーション CAPTCHA の提案、" 情報処理学会シンポジウムシリーズ (CD-ROM)、Vol.2015、No.1、pp.1816-1822、 2015.

29) 鈴木 徳一郎、山本 匠、西垣 正勝、"リレーアタックに耐性をもつ CAPTCHA の提案、" 情報処理学会研究報告 IPSJ、Vol.2010、No.21、pp.1-8、2010.

30) "pixprofit.com",http://pixprofit.com, (accessed 2017-06-10).

31) 3D-based Captchas become reality, clnet (online), http://ascii.jp/elem/000/000/483/483759/index-2.html, (accessed 2017-06-10).

32) M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, M.G. Voelker and S. Savage, "Re:CAPTCHAs-Understanding CATPCHASolving Services in an Economic Context," USENIX Security Symposium, Washington, Vol.10, pp.1-18, 2010.

33) Inside India's CAPTCHA solving economy, (online)、http://www.zdnet.com/article/inside-indias-captcha-solving-economy/, (accessed 2017-06-10).

34) 小宮山哲俊、梅澤猛、大澤範高、"CAPTCHA リレーアタックのパフォーマンス低減手法の提案、" 情報科学技術フォーラム講演論文集, Vol.13、No.4、pp.171-172、2014.

35) Huy D. Truong, F. Turner Christopher, and C. Zou. Cliff, "iCAPTCHA: the next generation of CAPTCHA designed to defend against 3rd party human attacks," IEEE International Conference on Communications, pp.1-6, IEEE, 2011.

36) M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, and W. B. Chen, "A three-way investgation of a game-CAPTCHA: automated attack, relay attacks and usability," Proceedings of the 9th ACM symposium on Information, computer and communications srcurity, pp.195-206, ACM, 2014.

37) M. Mohamed, S. Gao, N. Saxena, and C. Zhang, "Dynamic cognitive game captcha usability and detection of streaming-based farming," the Workshop on Usable Security (USEC), co-located with NDSS, 2014.

38) S. Gao, M. Mohamed, N. Saxena, and C. Zhange, "Gaming the game: Defeating a game captcha with efficient and robust hybrid attacks," Multimedia and Expo (ICME), 2014 IEEE International Conference, pp.1-6, IEEE, 2014.

39) "HELLO CAPTCHA",http://www.hellocaptcha.com/, (accessed 2018-01-15).

40) "MINTEYE - 捻じ曲げられた画像をスライドして正しく直すことで人間であることを判定する新方式の CAPTCHA",http://developer.cybozu.co.jp/akky/2012/12/minteye-slider-captcha/, (accessed 2018-01-15).

41) V. Nguyen, Y. W. Chor, and W. Susilo, "Breaking an animated CAPTCHA scheme," Applied Cryptography and Network Security, pp.12-29, Springer, 2012.

42) Breaking the minteye captcha again - Hack a Day, (online), https://hackaday.com/2013/01/19/breaking-the-minteye-captcha-again/, (accessed 2017-07-25).

43) S. Sivakorn, J. Polakis, A. D. Keromytis, "I'm not a human: Breaking the Google reCAPTCHA," Black Hat ASIA, pp.1-12(2016).

44) 第 14 回パターン認識・メディア理解研究会 (PRMU) アルゴリズムコンテスト ターゲットをロックオンせよ! ～移動物体の追跡～, (オンライン),http://www.murase.m.is.nagoya-u.ac.jp/alcon2010/?page=download, (accessed 2018-01-15)

45) Index of /software/vyos/iso/release/1.1.7 , (online),http://ftp.tsukuba.wide.ad.jp/software/vyos/iso/release/1.1.7/, (accessed 2017-05-25).

46) S. S. Shapiro, and M. B. Wilk, "An analysis of variance test for normality (complete samples)," Biometrika, Vol.52, No.3, pp.591-611, 1965.

47) 岡田和典、"ミーンシフトの原理と応用、" 情報処理学会研究報告コンピュータビジョンとイメージメディア (CVIM)、信学技報、 Vol.107, No.539, pp.401-414, 2008.

48) I. R. Khan, and F. Farbiz, “A back projection scheme for accurate mean shift based tracking,” Image Processing (ICIP), pp.33-36, IEEE, 2010.

49) J. Yan, and A. S. El Ahmad, “Usability of CAPTCHAs or usability isuues in CAPTCHA design,” Proceedings of the 4th sympousium on Usable privacy and security, pp44-52, ACM, 2008.

50) Open CV:OpenCV.jp（オンライン）,http://opencv.jp/ (参照 2017-09-10).

51) A. S. El Ahmad, J. Yan, and W. Y. Ng, “CAPTCHA design: Color, usability, and security,” IEEE Internet Computing, Vol.16, No.2, pp.44-51, 2012.

52) K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, “Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs),” Human Interactive Proof, Vol.3517, pp.1-26, 2005.