機械学習とアイトラッキング技術を用いた、画面盗み見対策アプリケーションの開発と評価

芝浦工業大学附属中学高等学校
髙橋司　小梶匠海　森本空良

1. 動機

近年、コロナの影響やペーパーレスの推進などで、デジタル機器を使用しての作業が増え、より多くの情報がデジタル機器を介してインターネット上でやり取りされるようになった。それにより様々な脆弱性対策が行われるようになっている。その中でも、我々は「ショルダーハッキング」に着目した。

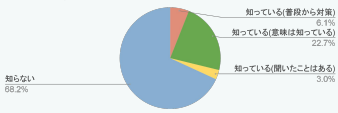
「ショルダーハッキング」とは、いわゆる”盗み見”を行い不正に情報を入手する、日常生活で最も身近かつ原始的なハッキングの手法である。しかし、ショルダーハッキングへの対策と大衆への周知は、他のサイバー犯罪ほど進んではいない。

このような理由から、このハッキング方法についての対策を考えることにした。

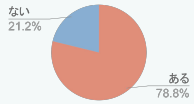
2. ショルダーハッキングに関する意識調査

GoogleFormsを利用しハッキングに対する意識調査とショルダーハッキングの注意度に関するアンケート調査を行った。[注1]

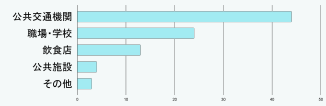
結果(一部を抜粋)



[グラフ1]ショルダーハッキングを知っているか



[グラフ2]他人が操作する画面を覗いたことがあるか



[グラフ3]他人が操作する画面を覗いてしまった場所

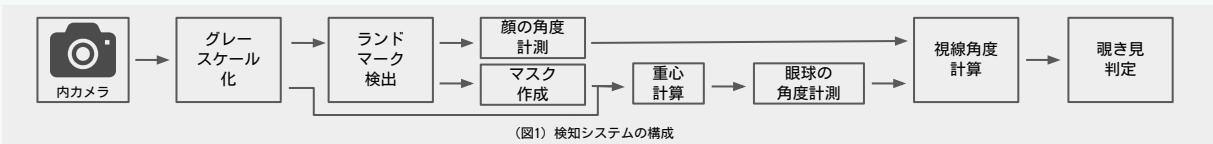
他人の端末を見たことがあると回答した人は78%[グラフ1]であったことより、意識無意識を問わずに他人の画面を覗くことは容易な状況であることがわかる。しかしショルダーハッキングという単語を知らない割合は68.2%[グラフ2]と半数を大きく超えていた。そして、普段から気をつけているハッキング方法に関する質問ではショルダーハッキングや覗き見などに気をつけているという回答はわずか7%と普段から覗き見対策をしている人などはごくわずかであることがわかった。

つまりかなり多くの人がショルダーハッキングを行なっているにも関わらず対策は普段からほどんど行われていないということが回答からわかる。またショルダーハッキングという単語自体を知らないという人が約70%いるということから、このハッキング方法に対する世間の意識はとても低いということが併せてわかる。

ショルダーハッキングを行なっている人のうち、それが行われた場所は76.8%[グラフ3]が公共交通機関である。つまり電車やバスといった公共交通機関に、ショルダーハッキングが行われやすいという状況があることもこれらからわかる。

3. 対策アプリケーションの開発

アプリケーションは検知システム(図1)と設定や通知を行うウィンドウから成る。また、Python3.10がインストールされているWindowsPCを対象として開発を行った。



(図1) 検知システムの構成

3-1. 検知システムの概要

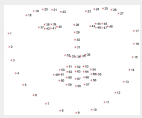
検知システムは、各デバイスのインカメラ(またはWEBカメラ)を使用する。カメラ映像から、顔ランドマーク検出を用いて顔が真正面の方向になるように射影変換を行う。次に眼の部分のみを切り抜き、二値化を行う。それらをもとに角膜以外のマスクを作成し、もとの画像に作成したマスクを反映して、角膜の中心座標を求める。その座標と顔の傾きを考慮し、視線の位置がモニターなどの検知範囲内であるかを調べることで検知を行う。これらの処理を、情報が多くライブラリが豊富なPython3を使用してスクリプトを作成した。

3-2. 顔ランドマーク検出

カメラ映像から顔のランドマークを検出する。はじめに、以後の処理のためOpenCVの関数を使用してグレースケール化する。を顔ランドマーク検出では、画像処理ライブラリのOpenCV及びDlibライブラリとDlibの公式サイト[注2]で公開されている学習済みモデルshape_predictor_68_face_landmarks.datを使用した。

稀に誤検知(図2)が見られたが、長時間発生しなかったため問題ないと考えた。他にマスクやサングラスをしている場合は検知できない問題があった。これはコロナウイルスによりマスク着用者が多いため、実用化をする際は改善しなければならないと考える。

(図2) 布の瞼と顔との誤検出



(図3) ランドマーク位置[注3]

3-3. 顔の角度の計測

XYZ軸それぞれの顔の角度をcv2.solvePnP関数とcv2.Rodrigues関数で回転行列、ヤコビアンを用いて取得する。この手法では予めカメラに対して真正面に向いた際の各ランドマークの基準となる座標が必要となる。この座標は、最初のキャリブレーション時に取得した値を用いる。しかし、個人差があるため正確に求めることが難しい。

3-4. 眼球の角度の計測

また、各ポイントは表情により位置が変わってしまうので正確に計測できない場合もある。しかし、今回のアンケート調査からショルダーハッキングが起こりうる場所の多くは公共交通機関、職場、飲食店のため、極端に無表情から外れることは少ない。そのため影響は少ないと考えた。

顔を真正面から見た際の眼球の角度を求める。そのため、はじめに顔の角度情報から射影変換を行う。その後瞳孔の中心を求める。解像度や個人差の問題で、映像から正確に瞳孔の輪郭を抽出することが難しい。そのため今回は角膜の重心を瞳孔とすることにした。はじめに、その後3-2で検知した各ランドマークの内、目の箇所であるポイント37から42と43から48(図3)内の角膜、眼球結膜、それ以外で3値化を行い角膜以外の範囲のマスクを作成した。それを用いると角膜のみのグレースケール画像が得られる。その画像を反転させてcv2.moment関数で重心を求め、これを瞳孔の中心とした。最後に、眼球の中心と瞳孔の中心の距離と、3-2でランドマークの基準とした眼球のサイズから眼球の角度を求める。

3-5. 視線位置の計算とショルダーハッキングの判定

顔の角度及び眼球の角度を足したものを視線の角度とする。この角度が、最初のキャリブレーション時に保護対象の視線の範囲内であれば検知したと判定する。

3-6. 検知表示ウィンドウ

顔のショルダーハッキングの判定がされた場合、検知された時刻が外部ファイルに記録され、アプリケーションの通知としても表示される。ユーザーは通知により向きやウィンドウの変更を行ったりすることで対策が可能となる他、記録された時刻からパスワードを変更するなどの適切な対策を講じることが可能となる。

4. 考察と今後の展望

4-1. 考察

本システムを利用した際の正常な検知率は撮影環境にかなり左右されるため正確に計測することができなかったが、環境ごとに顔を検知できた際のショルダーハッキング検知率は概ね30%から90%であった。確度が低下する原因は、撮影環境と顔の向きの2つが挙げられる。撮影環境が暗かったり逆光だったりすることで、角膜と眼球結膜やその周囲とのコントラストがはっきりせず、3値化が正常に行われていなかった。また、顔の向きがカメラに対して横向きや下向きになると、射影変換を行っても角膜の多くが瞼等で隠れるので、正しく瞳孔の中心座標を取得できていないと考えられる。一方でほぼ真正面とやや上を向いた際は、ほぼ確実に検知を行うことができていた。

4-2. 今後の展望

このシステムを実用化させる上で、複数の改善点がある。１つ目はマスク着用時に認識しないことである。これは、顔のランドマーク検出に、マスクを着用した状態で学習させたモデルを使用することで対応可能である。これによりランドマークが減るため、多少の高速化も期待できる。２つ目は顔の角度の正確性が低いことだ。これらはDNNやSVM等の機械学習を用いることで個人差が出る初期値が不要となるので改善できるのではないかと考える。３つ目は処理の低負荷、高速化である。アンケート調査時にバッテリーの持ちが心配であるという意見があった。顔検出と3値化が最も負荷が高いため、ランドマークを必要箇所のみにするなどの対応を取りたい。４つ目は覗いた人の悪意の有無の判定機能の追加である。アンケート調査で、冤罪などを危惧する声があった。視線の静止時間を計測したりすることで相手が情報を盗み取ったかどうかの判定を行えるようにしたい。５つ目はスマートフォンへのシステムの応用だ。アンケートでは、このシステムが公開された場合スマートスマートフォンで使いたいという回答が71.9%であった。アンケートよりスマートフォンに対する需要が大きいことからスマートフォンへシステムを応用していきたい。

5. 注釈、参考文献、謝辞

注釈

1. https://docs.google.com/document/d/1VPW4iYvBk1U-Qs80wqaKsbu_QZQi5tc1GrjQixlWs0s/edit?usp=sharing
2. http://dlib.net/
3. https://github.com/ic-pc-uk/resources/facial-point-annotations/ Figure2より引用
4. 単に重心を求めるより、瞳孔のほうが暗く見えることを利用して重み付けを使用し、精度を上げるために第二引数をTrueで密度ありとした。

参考文献

1. Vardan Agarwal.”Real-time eye tracking using OpenCV and Dlib”.Medium.2020-5-5. https://towardsdatascience.com/real-time-eye-tracking-using-opencv-and-dlib-b504ca724ac6. (参照2023-01-22)
2. Asadullah-Dal17.”Eyes Tracking Opencv python👁 👁”.Github.2022-05-07. https://github.com/Asadullah-Dal17/Eyes-Tracking-Opencv-and-Dlib. (参照2023-01-22)
3. @mimitaro. “pythonとdlibでお手軽に顔のランドマークを検出してみた”.Qiita.2019-12-03. https://qiita.com/mimitaro/items/bbc58051104eafe1eb38. (参照2023-01-20)
4. @oozzZZZZ. “PythonとOpenCV+dlibを用いた頭部方向推定”.Qiita.2019-12-04. https://qiita.com/oozzZZZZ/items/1e68a7572bc5736d474e. (参照2023-01-24)
5. 望月 優輝. “厳密な重心の求め方”.CV&Tech. 2017-08-29. https://cvtech.cc/centroid/ .(参照2023-01-24)

謝辞

最後に、この研究を行うにあたってご協力頂いた以下の方々に感謝申し上げます。

- アンケートにご回答していただいた皆様
- ポスターの校閲を行ってくださった先生、生徒方
- アンケート手法検討と拡散をしてくださった先生方