Fokozottan emberszerű interfész fejlesztése számítógépes rendszerekhez

Részletes Rendszerterv

Merész Patrik

Tartalomjegyzék

[Absztrakt 3](#_Toc132297280)

[Bevezetés 3](#_Toc132297281)

[Motiváció 4](#_Toc132297282)

[Hasonló projektek 4](#_Toc132297283)

[Megvalósítás 5](#_Toc132297284)

[Eszközök: Python, LabelImg, Google Colab, YOLOv5, CUDA, PyTorch 5](#_Toc132297285)

[Adatgyűjtés 5](#_Toc132297286)

[Adatok Feldolgozása 5](#_Toc132297287)

[Tanítás 5](#_Toc132297288)

[Célzás implementálása 6](#_Toc132297289)

[Rejtve maradás 6](#_Toc132297290)

[Jogi kérdések 7](#_Toc132297291)

[Hivatkozások 8](#_Toc132297292)

[Idegen szavak jegyzéke 8](#_Toc132297293)

# Absztrakt

A szakdolgozat tartalmazza azon preventív rendszerek bemutatását, amelyek képesek megkülönböztetni gépi, illetve emberi interakciókat, különös tekintettel a [bot](#bot) detektálás területén. A dolgozatban kifejtem, hogy milyen következményekkel jár, ha az említett ellenőrzés elmarad, továbbá ezen rendszerek előnyeit és hátrányait fogom taglalni. A szakdolgozat második részében egy olyan szoftver tervezéséről és megvalósításáról lesz szó, amelyet a fentebb említett rendszerek még nem megbízhatóan képesek eldönteni a generált válaszok valódiságát. A tesztkörnyezet, amiben ezt a szoftvert fogom tesztelni az online számítógépes játékok világa lesz, amelyben a modern bot felismerő rendszereket fogom próbatétel elé állítani. Egy olyan szoftvert fogok fejleszteni, amely úgy mozgatja az egeret, hogy az a lehető legemberszerűbb legyen és jelentősen megkönnyíti a célzási műveleteket. A program emberi voltát képfelismeréssel, ami lehetővé teszi a kívánt objektumok felismerését és lokalizációját, majd egy neurális hálóval, ami az egérmozgatásért felel, szeretném garantálni. A neurális hálóval a saját egérmozgatási adataimmal feltanítva fogok emberszerű egérmozdulatokat generálni. A dolgozat végén összegzem a fejlesztés menetét és értékelni fogom a projekt sikerességét

# Bevezetés

Az ember-számítógép kapcsolatok (HCI) kutatása egyik kulcsterület napjainkban. Számos olyan alkalmazás jelenik meg, ahol a kommunikációs partner egy számítógép, és az szolgáltat emberszerű válaszokat. Ilyenek például a chatbotok, hangalapú asszisztensek vagy tágabb értelemben vett virtuális ügynökök. Ezek a rendszerek rengeteget fejlődtek az elmúlt pár évben és a korlátaik terén is jelentős előre lépés történt, köszönhetően az AI és machine learning technológiáknak. Azonban érdemes megjegyezni, hogy az alább felsorolt rendszerek még mindig nem tökéletesek és alkalom adtán képesek olyan választ adni amely messze távol van egy emberi kimenettől. Találkozhatunk olyan esetekkel is amikor ezek a botok rendkívül kártékonyak lehetnek, mivel teljesen megtévesztik a kommunikáló felet és fontos információk kiadását vagy az egyén véleményének befolyásolására használják.

Az elmúlt években jelentősen megnövekedett a kártékony botok száma ez különösen veszélyes lehet az interneten….. létre hozni egy olyan programot, amely valósidőben képes detektálni ellenfeleket a képernyőn és a pozíciójukat meghatározva a másodpercek tört része alatt 0.1-0.2 másodpercen belül becélozni azokat. Ezt a gyakorlatban úgy szeretném megvalósítani, hogy számítógépes játékokban, kifejezetten FPS (belső nézetes lövöldözős) típusúakban fogom implementálni. A mágikus 0.1-0.2 másodperces szám, egy átlagos fizetett profi játékos reakció ideje az ellenfél felismerésétől az első célzott lövés elsütéséig, de ha a program ennél is gyorsabb lesz az csak külön öröm. A projekt másodlagos célja az, hogy mind ezt úgy hajtsam végre, hogy a profilunkat ne érje kitiltás. A projektet két játékon tervezem megvalósítani az egyik a Counter Strike: Global Offensive (CSGO) a másik játék pedig a Valorant lesz. A CSGO-ra az egyszerűség és a kevésbé kifinomult csalásellenes szoftvere miatt esett a választás, a Valorantra pedig a híresen erős csalásellenes szoftvere miatt a választottam. Mind a két játék ingyenes, így a tesztelés közben felmerülő anyagi károk valószínűleg nem lesznek súlyosak.

## Motiváció a projekt elkészítéséhez

Kompetitív személyiségűnek tartom magam, gyerekkoromban sok versenyen részt vettem, mind sportban, tanulmányi, illetve e-sport szinten is. Tanulmányaim során nem jut elég idő a gyakorlásra az utóbbi területre, ezért a játékokban való teljesítményem is drasztikusan romlik, ez különösképpen akkor a legbosszantóbb, amikor egy közeli ismerős vagy egykori csapattárs hívja fel a figyelmed arra, hogy mennyire nem vagy formában. Megoldást ebben a projektmunkában látom.

## Hasonló projektek

Hivatalos forrásból származó hasonló projektekről nem találtam információt. Az interneten különböző fórumokon kutatva fellelhető pár hasonló megoldás, de ezek elég gyengére sikerültek hatékonyság szempontjából. A legalapvetőbb probléma az, hogy könnyedén detektálhatóak más részről a teljesítményük is elmarad a kívánttól.

# Megvalósítás

## Eszközök: Python, LabelImg, Google Colab, YOLOv5, CUDA, PyTorch

## Adatgyűjtés

A projektmunka megvalósításának az első lépése az adatgyűjtés, ezt úgy oldom meg, hogy írok egy scriptet, ami bizonyos időközönként vagy egy billentyű parancsra képernyőfelvételt készít. Ezt valószínűleg játék közben, illetve játékról szóló videók nézése közben fogom futtatni, hogy a lehető legtöbb képet sikerüljön összegyűjtenem.

## Adatok Feldolgozása

Miután összeszedtem a több ezer képet, ezután fel kell őket dolgoznom, erre a LabelImg nevű szoftvert fogom alkalmazni, ez egy nyíltforráskódú, egyszerűen használható program. Ez a folyamat a projektmunka legidőigényésebb része, de szerencsére csak egyszer kell megcsinálni, illetve annyiszor ahány játékra megszeretnénk írni programot. A feldolgozás folyamatát úgy kell elképzelni, hogy a nyers képeken kézzel kijelöljük a detektálni kívánt objektumokat. Ezzel a módszerrel nagyjából 150-200 képet lehet feldolgozni óránként.

## Tanítás

Ezt a rengeteg képet egy YOLOv5 modellnek fogom betanítani. A YOLO röviden egy nyíltforráskódú objektum felismerő algoritmus, amely a képeket egy mátrix rendszerré alakítja, amelyből képesek vagyunk kiolvasni a felismert objektum x,y koordinátáját és az osztályának nevét [1]. A YOLO alapvetően öt modellel rendelkezik, rendre N, S, M, L, X ezek legfőképp a számítási teljesítményükben, pontosságban és legfőképp a detektálási idejükben különböznek, ahogy az alábbi táblázatban látható.

|  |  |  |  |
| --- | --- | --- | --- |
| Modell | Kép méret  (pixel) | Detektálás sebessége  (ms) | Pontosság  (%) |
| YOLOv5n | 1280 | 8 | 62 |
| S | 1280 | 34 | 76 |
| M | 1280 | 153 | 81 |
| L | 1280 | 424 | 84 |
| X | 1280 | 1249 | 89 |

A fentebb látható eredményeket egy százelemű képhalmazból sikerült produkálnom egy GTX 1060-as videókártyával. Egy erősebb videókártyával jelentősen javítható a sebesség. A projektemben az S modellt választottam, mivel viszonylag gyors és kellő precizitással rendelkezik. Az én esetemben a pontosság nem annyira releváns mivel, ha egyszer-kétszer ráugrik az egér egy virágcserépre vagy egy árnyékra az bocsánatos bűn.

Kanyarodjunk is vissza a tanításhoz, a megcímkézett képeket feltöltjük Google Colab-re ahol, rendelkezésünkre áll egy véletlenszerűen kiválasztott erős videókártya, viszont ügyelnünk kell arra, hogy csak 12 órán keresztül futtathatjuk, különben fizetni kell érte. A Google Colab segítségével a megfelelő YOLO modellnek betanítom a képeket. Miután a tanítás végzett létrejön a legjobban sikerült betanított modellünk. A legjobban sikerült modellt a PyTorch (nyílt forráskódú gépi tanulási keretrendszer) könyvtárának adom át, ami a modell alapján egy kétdimenziós tömb eredményt ad vissza, ahol a sorok egy-egy objektumot reprezentálnak az oszlopok pedig az adott objektumhoz tartozó attribútumokat tartalmazza, mint például koordináták.

## Célzás implementálása

Most, hogy már tudjuk, hogy a képernyőn felismerve hol vannak az ellenfelek, már csak annyi a teendőnk, hogy kiválasszuk a megfelelő célpontot majd az egeret mozgassuk a megfelelő helyre. Először is ki kell találni, hogy mi alapján válasszunk elsődleges célpontot, főleg akkor amikor több ellenfél is a képernyőn van. Kétféleképpen történhet egyik megoldás, hogy az egérhez a legközelebbi célpontot válasszuk vagy azt, aki a karakterünkhöz a legközelebb van, ezt úgy lehet megállapítani, hogy megmérjük az adott ellenfelet körbe rajzoló téglalap kerületét és kiválasztjuk a legnagyobbal rendelkezőt. Az egér mozgatását egy betanított neurális halóval szeretném megvalósítani, ehhez írok egy egyszerű programot, amiben különböző pontokra kell kattintani és a program folyamatosan figyelni fogja az egér pozícióinak a változásait, amit majd kimentek és ezen adatok alapján betanítok [2]. Ha nem sikerülne a neurális hálóval megoldani az egér mozgatást akkor további megoldást látok a Bézier görbe alapú [3] illetve a WindMouse [4] algoritmusban.

## Rejtve maradás

A program írásakor ügyelni kell arra, hogy kerüljük azokat a jelenségeket, amelyek nem emberi tényezőre hajaznak, ilyen például az egyenes egérmozgás, az egér állandó sebessége, gyorsulása, továbbá emberfeletti reakció idő. Pontosan ezért van szükség a fentebb említett egér mozgató algoritmusokra, hogy ezt a problémát kiküszöböljük. A csalásellenes szoftverekről nagyon kevés információ áll publikusan rendelkezésre érthető okok miatt, ez legfőképp igaz a modernebbekre, de azért egy két dolgot sejthetünk a működésükről. Tekintsük például a CSGO Valve Anti Cheat (VAC) szoftverét, ami felhasználó oldalán fut különböző szűréseket alkalmazva vizsgálja a játék memória területét elváltozások után kutatva, ha talál valami gyanúsat akkor azt megjelöli és adatokat gyűjt róla. Az összegyűjtött adatok alapján összehasonlítja a többi adatbázisban lévő kártékony kódokkal, ha egyezés van azonnal tilt, ha nincs egyezés akkor további adatot gyűjt. Az elsődleges ellenőrzések után következik a második biztonsági protokoll, ami „Code Blacklisting” névre hallgat, ezen a részen a szoftver az összes betöltött DLL-t megvizsgálja különböző injektálások után kutatva, továbbá ellenőrzi a játékhoz tartozó DLL-ek sértetlenségét. A VAC csak alkalmazás szinten végzi a vizsgálatokat, kernel szintű műveleteket nem hajt végre [5]. A jó hír az, hogy nekünk ezekkel a dolgokkal nem kell foglalkozni, mivel nem módosítunk semmilyen játék fájlt és minden segítségünkre lévő eszközt kívülről érünk el. A ring másik oldalán a Valorant BattleEye (BE) rendszerével állunk szemben, amelyről még kevesebb tudunk, mint a VAC-ról, de az biztos, hogy a BE kernel szinten is fut és már a játékosok puszta viselkedését is elemzi valamilyen gépi tanulás algoritmussal. A kernel szintű védelem miatt a programom egyszerűen nem tud egér utasításokat küldeni, ugyan is a játék csalásellenes szoftvere egyszerűen letiltja azokat. Ezért ahhoz a megoldáshoz folyamodok, hogy egy Arduinohoz csatlakoztatott USB host shieldel [6] támogatom meg az egér vezérlést. Terveim szerint ezt úgy fogom megvalósítani, hogy a program által kiszámított koordinátákat átadom az Arduinonak amely ez alapján a megfelelő koordinátákkal elmozdítja az egeret, hogy ez működjön külön egér drivert kell készítenem. Ha a képfelismerést is valamilyen módon felismeri a BE és letiltja, akkor a programot két számítógépen fogom futtatni, az egyiken a játék fog futni a másikon pedig a képfeldolgozás. A két számítógép közötti kapcsolatot pedig egy USB képernyőfelvevő fogja biztosítani.

# Jogi kérdések

Magyarországon nincsen számítógépes játékokban alkalmazott segédszoftverekre kiterjedő törvény. A csalókkal szembeni fellépést általában a játékfejlesztő teszi meg, amely megfosztja a felhasználót a játékszoftver használatától. Komolyabb jogi szankciók akkor érhetik a segédszoftvert használó felhasználót, ha pénzért terjeszti szoftverét, illetve, ha súlyos anyagi károkat okoz a játékot forgalmazó cég számára. Erre már példa is volt Amerikában, amikor az Activison játékfejlesztő cég a Call of Duty (COD) egyik szériájában elvesztette játékosállományának jelentős részét, egy csalószoftvereket terjesztő cég közreműködésének köszönhetően. [7]

# Hivatkozások

|  |  |
| --- | --- |
| [1] | G. Jocher, „YOLOv5 documentation,” 18 Május 2020. [Online]. Available: https://docs.ultralytics.com/. [Hozzáférés dátuma: 8 12 2022]. |
| [2] | I. Arapakis és L. A. Leiva, „Learning Efficient Representations of Mouse Movements to Predict User Attention,” 25 07 2020. [Online]. Available: https://doi.org/10.1145/3397271.3401031. [Hozzáférés dátuma: 08 12 2022]. |
| [3] | J. Davies, 2010. [Online]. Available: https://www.jasondavies.com/animated-bezier/. [Hozzáférés dátuma: 8 12 2022]. |
| [4] | B. J. Land, „WindMouse, an algorithm for generating human-like mouse motion,” 25 4 2021. [Online]. Available: https://ben.land/post/2021/04/25/windmouse-human-mouse-movement/. [Hozzáférés dátuma: 8 12 2022]. |
| [5] | T. Curda, „Analysis and detection of online game cheating software BACHELOR THESIS,” Brno, 2014. |
| [6] | Arduino Corporation, „Arduino USB Host Shield,” Arduino, 7 12 2022. [Online]. Available: https://docs.arduino.cc/retired/shields/arduino-usb-host-shield. [Hozzáférés dátuma: 9 12 2022]. |
| [7] | E. Nightingale, 5 1 2022. [Online]. Available: https://www.eurogamer.net/activision-files-lawsuit-against-call-of-duty-cheat-site. [Hozzáférés dátuma: 9 12 2022]. |

# Idegen szavak jegyzéke

Bot: Egy olyan szoftveres alkalmazás, amit arra terveztek, hogy különböző feladatokat oldjanak meg automatikusan, jellemzően azzal a céllal, hogy helyettesítse vagy utánozza az emberi beavatkozást. Ezek a feladatok általában ismétlődő jellegűek és sokkal hatékonyabban oldják meg, mint mi emberek.