SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR
GÉPÉSZMÉRNÖKI, INFORMATIKAI ÉS VILLAMOSMÉRNÖKI KAR

GÉPI LÁTÁS

GKNB_INTM038

BIZTONSÁGI KAMERA:

KAMERA VIDEÓFELVÉTELEN EGYIDEJŰLEG EGY MOZGÁS FELISMERÉSE ÉS NYOMON KÖVETÉSE.

Szerző:
Litter Ádám
FFX181
adamlitter99@gmail.com

Feladat

Kamera videofelvételen egyidejűleg egy mozgás felismerése és nyomon követése. A felvétel lehet kamera élő képe, vagy pedig előre elkészített felvétel. Az forrás kiválasztását lehetővé kell tenni a felhasználó számára.
A detektált mozgás koordinátáit és a detektálás idejét páronként megkell jeleníteni a felhasználó számára, valamint lehetővé kell tenni a gyűjtött adatok mentését a program használatát követően.

Környezet

- A fejlesztés nyelve: C#,
- target framework: .Net 4.7.2,
- target platform: x86
- target OS: Windows

Elméleti háttér

Ahhoz, hogy kapcsolatot teremtsünk a gépek és a világ között, a gépeknek fel kell ismerniük a valós elemeket, hogy minél egyszerűbb legyen az átjárás és a kommunikáció, gép és ember számára.

„A gépi látás fogalma tágabb a robotlátás fogalmánál. Legáltalánosabban a gépi látás a gép által a környezetéről, vagy a működése által érintett objektumokról készített kép számítógépes elemzését jelenti abból a célból, hogy a gép a készített képből az általa végzett működésben hasznosítható információt nyerjen ki. A látvány fényképező eszközzel, kamerával, vagy egyéb speciális szenzorral való elkészítése és a képállomány memóriában való letárolása után az érdemi folyamat a digitális képfeldolgozás témakörébe tartozik." (1)

Egy mozgás felismerésére és követésére több alternatíva is létezik. A probléma megoldásához először mindenképpen szükségünk van a fizikai látvány feldolgozására, és olyan formátumúvá alakítására, hogy azt egy számítógép tárolni tudja és a későbbiekben képes legyen különböző transzformációkat végrehajtani rajta.
Ezt nevezzük digitális képfeldolgozásnak, melynek végére egy mátrixot hozunk létre, amelyben a különböző pixelek intenzitásának értékét tároljuk.

Az intenzitás egy pozitív 0 és 255 közötti érték lehet, mivel minden érték számára 8-8 bit helyet tart fent a legtöbb program. Fekete-fehér kép esetén a mátrix minden pixelhez egy-egy értéket rendel, azonban, ha színes képet szeretnénk eltárolni ahhoz egy hasonló módszert kell alkalmaznunk, ahol az RGB formátumnak megfelelően minden pixelhez 3 különböző intenzitást párosítunk, ezek az intenzitások képviselik a különböző alapszíneket és színcsatornáknak nevezzük őket.

Miután elkészült a digitális kép, a gépi látás különböző módszereivel létrehozhatjuk annak leírását. A gépi látás több előnnyel is rendelkezik az emberi látással szemben. Minden egyes részletet azonos, magas fokú pontossággal vizsgál meg, valamint nem érzékeny az időbeliségre és dinamikusságra, amelyek különböző optikai csalódáshoz vezetnek az emberi látás során.

A digitális képek tárolásuk formátumának következtében megfeleltethetőek kétváltozós függvényeknek, amelyeken így könnyen tudunk definiálni különböző értelmezési tartomány és értékkészlet transzformációkat.

Egy videófelvételen bekövetkező mozgás detektálására több módszert is választhatunk, ezek mind különböző transzformációk kombinációi.
A feladat két nagy részegységre bontható. Először az egymást követő képkockának vesszük az értékenkénti abszolút differenciáját, ekkor az olyan pontokban, ahol nem történt változás, az új intenzitási érték 0 lesz, míg a többiben eltér majd nullától.
Második lépésként az éldetektálás kerül a középpontba. Él ott található, ahol a digitális kép intenzitásának értéke hirtelen, nagy mértékben változik.

Az éldetektálás az emberi, mind pedig a gépi látás során meghatározó. Ennek segítségével képesek vagyunk objektumok határvonalait meghatározni.
Az alapvető probléma azonban ilyenkor, hogy élek nem csak egy test határain vannak. A különböző alakzatok az objektumokon és az árnyékok, valamint az áttetsző felületek gyakran megnehezítik a feladatot, ezekre számos megoldás létezik már transzformációk segítségével, amelyeket a modern éldetektáló algoritmusok alkalmaznak.

A feladat terve és megvalósítása

A feladat megoldásához a Python, C++ és C# programozási nyelvek nyújtották a legtöbb eszközt, mivel mindegyiken elérhető az OpenCV cross platform könyvtára.
Először egy egyszerű python konzol alkalmazásban próbáltam ki a könyvtárat, hogy lássam annak felhasználási lehetőségeit, végül pedig egy asztali alkalmazást készítettem .NET keretrendszer használatával, hogy egy könnyen kezelhető felhasználói felületet hozhassak létre a lehető leggyorsabb módon.
A GUI-ra azért volt szükség, hogy a későbbi finomítások során az paraméter értékeket, amelyekkel a különböző transzformációkat végző metódusok dolgoznak, könnyen változtathassam a kód átírása és újra fordítása nélkül. A végleges beállítások után a kezelő felület feleslegessé vált részeit eltávolítottam.

Az OpenCV könyvtárat közvetlenül nem lehetet importálni, ezért az Emgu CV .NET wrappert használtam, amely segítségével elérhetőek a szükséges függvények.

Az alkalmazás egyszerre mindig 3 képkockát kezel. Indításkor beolvassa az első két képkockát, amelyek segítségével meghatározza, hogy történt-e a két felvétel között elmozdulás, majd pedig az egyik eredeti képkocka másolatára rárajzolja a megfelelő jelöléseket és megjeleníti a képernyőn. Végül pedig egy léptetést hajt végre, ami abból áll, hogy az első felvétel helyére lépteti a másodikat, a második helyére pedig beolvas egy újat. Mindezt egy végtelen ciklusban végzi leállításig, vagy pedig előzetesen elkészített videófelvétel esetén legfeljebb az utolsó képkockáig.

A program minden egyes alkalommal a két eredeti képkockának elkészíti a szürkeárnyalati konverzióit, ugyanis ez nagy mértékben megnöveli a későbbi transzformációk pontosságát, majd pedig létrehozza a felvételek értékenkénti abszolút differenciáját, amelyen így csakis a két felvétel közötti elmozdulás fog megjelenni, fekete háttéren, amin éldetektálást alkalmazva meghatározhatjuk az elmozdulás méreteit.

Az éldetektálásra több kiforrott módszer is létezik már a gyakorlatban. A projekt során én a Canny éldetektor néven ismert megoldást alkalmaztam.

A Canny éldetektor az egyik legelterjedtebb élkereső módszer a gyakorlatban, amely nem csak egyszerű szűrő, hanem egy több lépésből álló folyamat. Ugyan magának a folyamatnak nem része, de előfeltétele a felvétel szürkeárnyalati konverziója mindenekelőtt.

Zajszűrés

A bemeneti képen először Gauss simítást végzünk. A simításnak, vagy zajcsökkentésnek az alapvető lényege, hogy elsimítsa az intenzitás nagy változásait, a magas-frekvenciás tartalom csökkentésével. Ezzel az élek és a hirtelen átmenetek elhomályosodnak.

$$K(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

σ: A függvény szórása megadja, hogy a középponttól távolodva milyen gyorsan csökkennek a súlyok.

Gradiens operátor

Következő lépésként differencia számítást alkalmazunk a kisimított képen, horizontális és vertikális irányba is. Jelenleg több operátor közül is választhatunk, azonban a gyakorlatban bevált módszer szerint a Sobel operátort szokták alkalmazni.

Az X- és Y- irányú parciális derivált közelítésére a következő konvolúciós maszkokat használjuk:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Ennek eredményeként egy olyan képet kapunk, amin az élek intenzitásainak értékei kihangsúlyozva lesznek.

Nem-maximális élek elnyomása

Ennél a lépésnél eltávolítjuk a maximumra merőleges élgyanús pontokat.

Következményként csak a vékony él-kezdemények maradnak meg a képünkön, amire azért van szükség, hogy pontosabb eredményt kapjunk.

A fantomélek eltávolítása után a kontúrok továbbra is folytonosak kell, hogy maradjanak. Fontos szemelőt tartani, hogy az éldetektálás célja mindössze az élek létének és helyzetének meghatározása, az élek erőssége felesleges információ számunkra.

Hiszterézis küszöbölés

A következő lépésben kettős küszöbölést alkalmazunk alsó, illetve felső küszöbértékkel. Ha nagyobb az él erőssége, mint a felső küszöbérték, akkor megtartjuk. Ha kisebb az él erőssége, mint az alsó küszöbérték, akkor elvetjük. Ha a kettő küszöbérték között van az érték akkor megtartjuk abban az esetben, ha van olyan szomszédja, ami él. A megoldásomban 50-es és 100-as értéket használtam paraméterként. A javasolt arány a küszöbértékekre 2:1 és 3:1 közötti.

Kontúrvonalak keresése

A Canny éldetektálás után egy bináris képet kapunk, ami azt jelenti, hogy minden intenzitás értéke két lehetséges állapot közül az egyiket ábrázolja: van él vagy nincs él az adott pontban.

Bináris képekre könnyen alkalmazhatunk különböző kontúrvonal kereső metódusokat, amelyek megkeresik az összefüggő pontokat és csoportosítják őket élek szerint.

A kapott éleket eltároltam külön vektorokban. A zaj következtében azonban keletkeztek nem valós élek is, ezért csak azokat használtam a továbbiakban, amelyek által határolt terület elért egy minimális értéket, amely a kezelő felületen megadható.

Végül a feltételnek megfelelt éleknek megkerestem a különböző irányú maximumait és a visszajátszásra szánt képre, amely az eredeti képkocka másolata, rárajzoltam az érzékelt objektumokra az őket határoló négyzeteket és a négyzet középpontját egy vektorban eltároltam. A középpontok összekötésével pedig ábrázoltam a mozgó objektum által bejárt utat.

Tesztelés

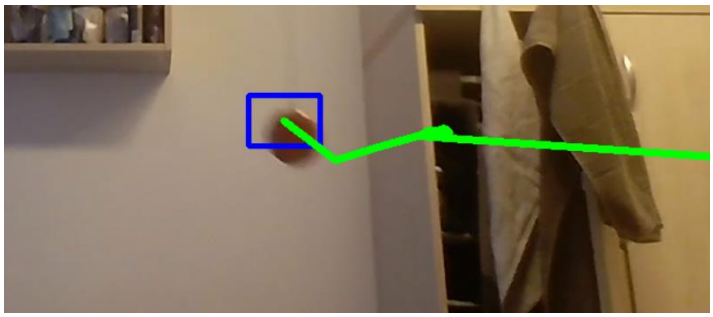
A project tesztekéséhez számos felvételt készítettem a laptopom kamerája, illetve egy másodlagos kamera segítségével, ezeknek egy része megtalálhatóak a projekt GitHub repository tartalmai között.

A tesztek két testtel lettek elkészítve, mindkettő esetében, távoli, közepesen távoli és közeli felvételek készültek, gyors és lassú mozgással egyaránt. Valamint az alacsonyabb felbontású másodlagos kamerával is meglett ismételve számos felvétel.

Az első próbálkozások is egyből rámutattak néhány súlyos problémára. Először is a detektált terület kiterjedésének értéke negatív is lehet, amennyiben az óra járásával ellentétes irányba számol a ContourArea() metódus. Ennek következtében számos eset volt amikor egy megfelelő kontúrvonal nem került megjelenítésre. Továbbá hatalmas gondot jelentett a program számára a kamera minden apró mozgása, mint például az a rezgés, amit a laptop hűtőrendszere produkál.

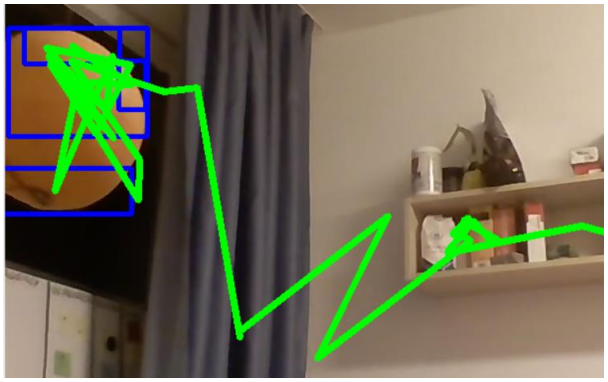
A kamera autó fókusza, illetve a mozgás következtében létrejövő árnyékok és fényviszony változások gyakran külön objektumként lettek érzékelve.

A különböző felvételekre egyéni korlátozásokat állítottam be, ezzel jelentősen javítva a tesztek kimenetelét. A legjobb eredményeket azok a felvételek produkálták, ahol az objektum mérete és a minimális kiterjedésre szabott korlát értéke alulról közelítették azt a maximális méretet, amelyet a program még egyelten összefüggő résznek érzékel.



Sajnos az utóbbi érték egészen alacsony ezért csak a kisebb tárgyakkal, illetve a kamerától távolabb elvégzett mérések szolgáltattak valós adatokat.

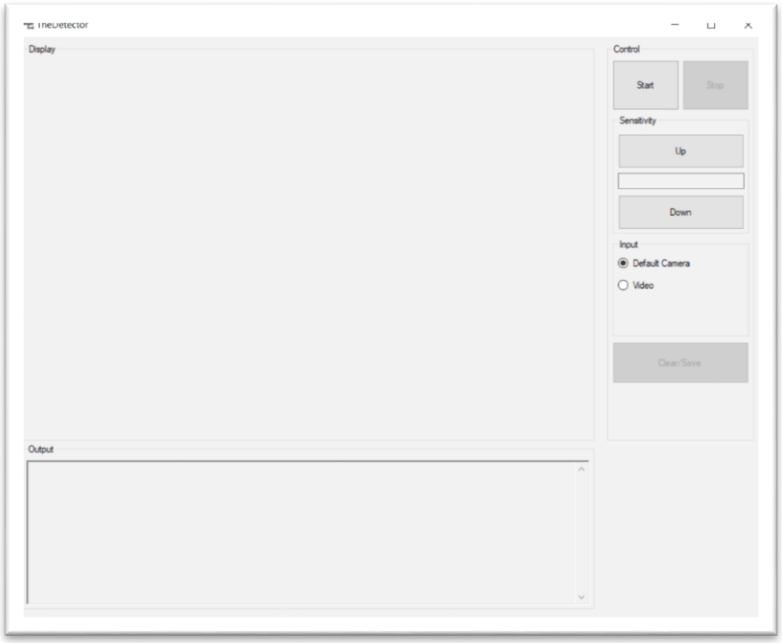
A felvételeken látható, hogy a testek saját tengely körüli forgása, valamint az egyéni mintázatok is több helyen problémát okoztak



Forgás okozta zavar

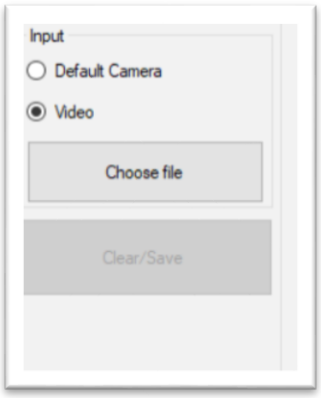
Felhasználói dokumentáció

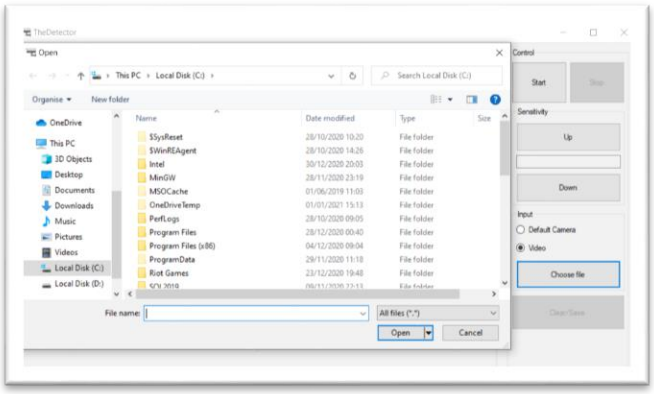
A programot egyszerűen indíthatjuk a lefordítás után keletkező EXE kiterjesztésű fájl segítségével, vagy a projektet egy megfelelő fejlesztői környezetben megnyitva is futtathatjuk. Az utóbbi megoldás esetében Microsoft Visual Studio egy aktuális verziója ajánlott, illetve fontos szemelőt tartani, hogy a program target platformja x86.



Default GUI megjelenése

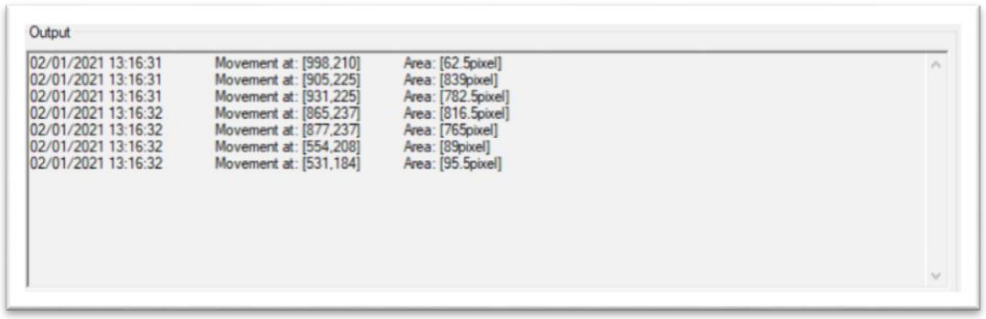
A **Start** gomb segítségével elindíthatjuk az alapértelmezett kamerát, viszont, ha egy már előre elkészített felvételt szeretnénk alkalmazni, akkor a **„Video”** feliratú rádiógombot kiválasztva elérhetővé válik a felhasználó számára a **„Choose file”** gomb, amire kattintva kiválaszthatja a megfelelő felvételt.



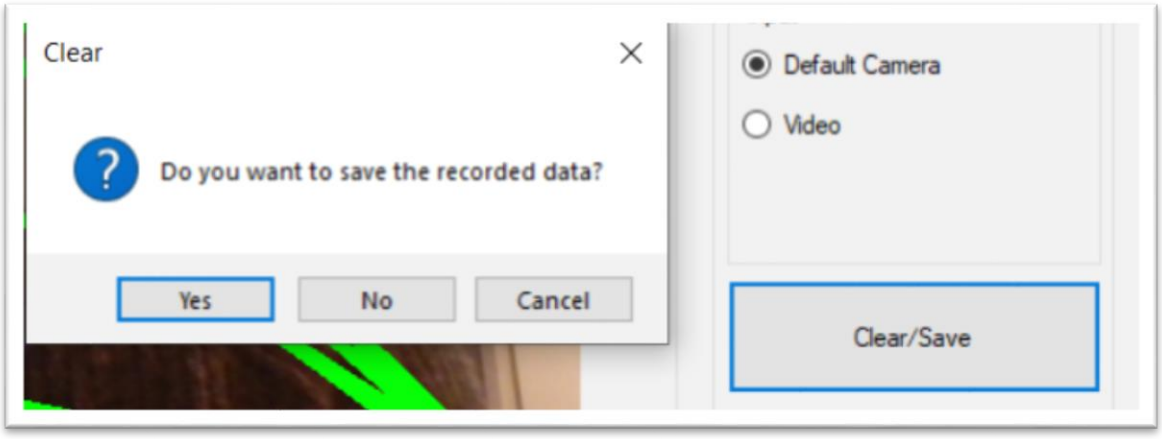


A megfelelő fájl kiválasztása után szintén a „Start” gomb segítségével indíthatjuk el a lejátszást, majd pedig a „Stop” gombbal leállíthatjuk.

A program a felvétel lejátszása során folyamatosan rögzít adatokat a detektált mozgásokról, amelyeket megjelenít egy textboxban.



A leállítás után lehetőségünk van törölni a textbox tartalmát, illetve egyidejűleg elmenthetjük az adatokat egy text fájlba a **„Clear/Save”** gomb segítségével.



Tartalom

Irodalom

1. **Dudás, L.** *Alkalmazott Mesterséges Intelligencia.* 2011.