Gépi Látás (GKNB_INTM038)

Szabó Dóra

G9AJVV

Választott feladat: Biztonsági Kamera

GitHub Repository: *https://github.com/Lumiko717/Biztonsagi_Kamera_OpenCV*

Bevezetés

Mai rohanó világunkban elengedhetetlen értékeink illetve otthonunk védelme így a biztonsági kamerák gyors fejlődése mindenképpen elengedhetetlen. Szimpla videó felvételek rögzítése már nem elegendő. A túlzott tárhely foglalás miatt illetve az esetleges betöréseknél az esetek visszakeresése idő és költségigényes. Így a mai modern biztonsági kamerák már okos mozgásérzékelő szoftverekkel ellátva biztosítják, hogy a felvétel pontosan akkor történik, ha tényleges tevékenység észlelhető a kamera látószögében. Fényviszonyokhoz igazodva akár képesek infravörös módra átállni automatikusan így követve a napszakok változását.

Ennél már csak az úgynevezett IP- kamerák adnak jobb biztonságot. Az IP-kamera egy olyan digitális kamera, amely az internet protokollt (IP) használja adatátvitelre, és így egy IP-hálózatra kapcsolódva kommunikál a külvilággal, szemben a hagyományos biztonsági (CCTV) kamerákkal, melyek analóg jeleket továbbítanak (jellemzően koaxiális kábelen), és szemben a web kamerákkal, melyek bár digitális eszközök, de adataikat csak egy működő számítógépre képesek továbbítani, annak USB-portjára csatlakozva.

Az általam választott feladat egy alapvető biztonsági kamera videó felvételén egyidejűleg egy mozgás felismerése és nyomon követése. A feladat megírásához a Python nyelvet és az OpenCV beépülő modult választottam a képfeldolgozási kódsorok egyszerűsége miatt. A feladatot kibővítettem választhatóan IP-kamerakép vagy fálj-ból való videókép megnyitással.

Elméleti háttér

A feladat megoldásához szükségem volt a számítógépes képalkotás és a digitális képfeldolgozás valamint a gépi látás alapvető megértésére, az ehhez tartozó matematikai és logikai alapokra valamit ezek Python programnyelvben való használatára.

Számítógépes képalkotás

A számítógépes képalkotás az a folyamat, amely közvetett módon képeket alkot mérésekből olyan algoritmusok felhasználásával, amelyek jelentős mennyiségű számítást igényelnek. A hagyományos képalkotással ellentétben a számítástechnikai képalkotó rendszerek magukban foglalják az érzékelő rendszer és a számítás szoros integrációját az érdekes képek kialakítása érdekében.

A gyors számítástechnikai platformok elérhetősége , az algoritmusok és a modern érzékelő hardver fejlődése jelentősen megnövelt képességekkel rendelkező képalkotó rendszerekhez vezet.

A számítástechnikai képalkotó rendszerek széles körébe tartozik:

- a számítási mikroszkópia
- a tomográfiai képalkotás
- az MRI
- az ultrahangos képalkotás
- a számítógépes fényképezés
- a szintetikus apertúra radar (SAR),
- a szeizmikus képalkotás

Az érzékelés és a számítás integrálása a számítógépes képalkotó rendszerekbe lehetővé teszi a hozzáférést olyan információ, amely egyébként nem volt lehetséges. A számítástechnikai képalkotó rendszerek lehetővé teszik, hogy leküzdjük az optika és érzékelők egyes hardverkorlátozásait (felbontás, zaj stb.) . A technika fejlődésével az emberiség olyasmit is láthat, amit szabad szemmel sosem fedezhetne fel.

Digitális képfeldolgozás

A digitális képfeldolgozás egy számítógép használata a digitális képek algoritmuson keresztüli feldolgozásához.

A digitális jelfeldolgozás alkategóriájaként vagy területként a digitális képfeldolgozásnak számos előnye van az analóg képfeldolgozással szemben. Ez lehetővé teszi az algoritmusok sokkal szélesebb körének alkalmazását a bemeneti adatokra, és elkerülheti az olyan problémákat, mint a zaj felhalmozódása és a torzítás a feldolgozás során. Mivel a képeket minimum két dimenzióban definiálják, a digitális képfeldolgozás modellezhető többdimenziós rendszerek formájában.

A modern képérzékelők alapja a fém-oxid-félvezető (MOS) technológia , amely MOSFET (MOS terepi tranzisztor) találmányából származik. Ez digitális félvezető a képérzékelők kifejlesztéséhez vezetett, ideértve a töltéssel összekapcsolt eszközt (CCD) és később a CMOS érzékelőt is.

Képtömörítés

A digitális képtömörítési technológia fontos fejleménye a diszkrét koszinusz-transzformáció (DCT) volt, veszteséges tömörítési technika. A DCT-tömörítés vált a JPEG alapjául, amely sokkal kisebb fájlméretekre tömöríti a képeket, és az interneten a legszélesebb körben használt képfájl-formátum lett. Rendkívül hatékony DCT tömörítési algoritmus volt nagyrészt felelős a digitális képek és digitális fényképek széles körű elterjedéséért.

Digitális jelfeldolgozó (DSP)

Digitális jelfeldolgozó az elektronikus jelfeldolgozást forradalmasította a MOS integrált áramkör technológia. Ez volt az alapja az első egy-chipes mikroprocesszoroknak és mikrovezérlőknek majd az első egy-chipes digitális jelfeldolgozó (DSP) chippeknek. A DSP chipeket azóta széles körben használják a digitális képfeldolgozásban. A DCT-ket széles körben használják kódoláshoz, dekódoláshoz, videokódoláshoz, hangkódoláshoz, multiplexeléshez, vezérlőjelekhez, jelzéshez, analóg-digitális átalakításhoz, fényerő és színeltérések formázásához, valamint olyan színformátumokhoz, mint a YUV444 és YUV411.

A digitális képfeldolgozás lehetővé teszi sokkal összetettebb algoritmusok használatát, és ez által egyszerű feladatoknál kifinomultabb teljesítményt és olyan módszerek megvalósítását is kínálhatja, amelyek analóg eszközökkel lehetetlenek lennének.

Gépi látás

A gépi látás egy átfogó tudományos terület, amely azzal foglalkozik, hogy a számítógépek hogyan szerezhetnek magas szintű megértést a digitális képek vagy videók segítségével. Mérnöki szempontból arra törekszik, hogy megértse és automatizálja azokat a feladatokat, képességeket, amelyekkel az emberi vizuális rendszer rendelkezik.

A gépi látási feladatok magukban foglalják a digitális képek megszerzésének, feldolgozásának, elemzésének és megértésének módszereit, valamint a nagydimenziós adatok kinyerését a való világból numerikus vagy szimbolikus információk előállítása céljából, pl. döntések formájában. A megértés ebben a kontextusban azt jelenti, hogy a vizuális képek (a retina bemenete) átalakulnak a világ leírásává, amelyek értelmet nyújtanak a gondolkodási folyamatoknak és megfelelő cselekvést váltanak ki. Ez a képmegértés úgy tekinthető, mint a szimbolikus információk szétválasztása a képadatokról a geometria, a fizika, a statisztika és a tanuláselmélet segítségével felépített modellek felhasználásával.

A gépi látás tudományos fegyelme a mesterséges rendszerek mögött álló elmélettel foglalkozik, amelyek információkat nyernek ki a képekből. A képadatok sokféle formát ölthetnek, például video szekvenciákat, több kamerából származó nézeteket, 3D-s szkennerből származó többdimenziós adatokat vagy orvosi beolvasó eszközt. A gépi látás technológiai fegyelme elméleteit és modelljeit igyekszik alkalmazni a gépi látásrendszerek felépítésében. A gépi látás részterületei közé tartozik a jelenet rekonstrukciója, eseményészlelés, videó követés, objektum felismerés, 3D-s pózbecslés, tanulás, indexelés, mozgásbecslés, 3D-s jelenet modellezése és a képek helyreállítása.

Fontosabb kapcsolódó technológiák:

A mesterséges intelligencia területei önálló útvonaltervezéssel vagy tanácskozással foglalkoznak a robotrendszerek számára a környezetben való navigáláshoz. A környezettel kapcsolatos információkat gépi látórendszer nyújthatja, amely látásérzékelőként működik, és magas szintű információkat szolgáltat a környezetről és a robotról. A mesterséges intelligencia és a gépi látás más témákat is megoszt, például a mintafelismerést és a tanulási technikákat. Következésképpen a gépi látást néha a mesterséges intelligencia területének tekintik.

A szilárdtest fizika egy másik terület, amely szorosan kapcsolódik a gépi látáshoz. A legtöbb számítógépes látórendszer képérzékelőkre támaszkodik, amelyek érzékelik az elektromágneses sugárzást, amely jellemzően látható vagy infravörös fény formájában jelenik meg. Azt a folyamatot, amelyen keresztül a fény kölcsönhatásba lép a felületekkel, a fizika segítségével magyarázzák meg.

A neurobiológia, nevezetesen a biológiai látórendszer vizsgálata. Az elmúlt évszázadban kiterjedt tanulmányt végeztek a szemek, az idegsejtek és az agyi struktúrák iránt, amelyeket mind vizuális ingerek feldolgozására fordítottak mind emberekben, mind különböző állatokban. Ez durva, mégis bonyolult leírást vezetett arról, hogy az "igazi" látórendszerek hogyan működnek bizonyos látással kapcsolatos feladatok megoldása érdekében. Ezek az eredmények egy olyan részterülethez vezettek a számítógépes látásmódon belül, ahol a mesterséges rendszereket úgy tervezték, hogy utánozzák a biológiai rendszerek feldolgozását és viselkedését, különböző összetettségi szinteken. Ezenkívül a számítógépes látásmódon belül kifejlesztett tanulásalapú módszerek egy része (pl. Idegháló és mélytanuláson alapuló kép- és jellemzőelemzés és osztályozás) a biológiához kapcsolódik.

A gépi láskutatás egyes részei szorosan kapcsolódnak a biológiai látás tanulmányozásához - éppúgy, mint az AI-kutatás sok szála szorosan kapcsolódik az emberi tudat kutatásához, valamint a tárolt tudás felhasználásához a vizuális információk értelmezéséhez, integrálásához és felhasználásához. A neurobiológiai látásvizsgálatok területe modellezi az emberekben és állatokban a vizuális észlelés mögött álló fiziológiai folyamatokat. A gépi látás viszont tanulmányozza és leírja a mesterséges látásrendszerek mögött szoftverekben és hardverekben megvalósított folyamatokat.

Megvalósítás

A feladat megoldásához az órán tanult Python programnyelvet és a hozzá tartozó OpenCV kiegészítő könyvtárat használtam.

Az OpenCV egy Python-összerendelés-könyvtár, amelyet főleg gépi látási problémák megoldására terveztek. A Python egy általános célú, egyszerű és könnyen kódolható/ olvasható. Az OpenCV a Numpy-t használja, amely egy nagyon optimalizált könyvtár a numerikus műveletekhez.

A feladatot a biztonsági kamera képének beolvasásával kezdtem el. A program már elmentett videó fájlt illetve IP-kamera képet is képes legyen kezelni.

A tkinter Python csomag segítségével létrehoztam egy egyszerű felugró ablakot amely egy IP-kamera címét várja megnyitásra , amennyiben ez a mező üresen marad a program tovább lép egy elmentett video fájl megnyitására.

Fontos Python bodulok importálása:

```
import cv2
import numpy as np
import tkinter as tk
from tkinter import simpledialog
from tkinter import filedialog
```

```
# IP Video cím vagy Video fájl kiválasztáshoz felugró dialógus ablakok

root = tk.Tk()
root.withdraw()
UIP = simpledialog.askstring(title="IP Kamera", prompt="IP Kamera címe:")
if not UIP:
    root.filename = filedialog.askopenfilename(initialdir="/", title= "
Valasszon video fajlt", filetypes=(("avi files", "*.avi"), ("all files",
"*.*")))
    felv = cv2.VideoCapture(root.filename)
# Video forrás beolvasása fileból
else:
    felv = cv2.VideoCapture(UIP)
# IP Video kép beolvasása
```

Itt a „felv” változóba kerül beolvasásra a videókép. A kiválasztott IP-kamerakép vagy a beolvasott fájl után a program tovább halad a videó framekre való bontásával.

```
ret, frame1 = felv.read() # videókép első frame-jének beolvasása
ret, frame2 = felv.read() # videókép második frame-jének beolvasása
```

Ezekkel a képkockákkal indul el a videókép feldolgozása. Minden egyes lépés után újra és újra felül írásra kerülnek majd hogy a videó haladásával is elég legyen csak két változó használata.

A videó feldolgozása egy „while” ciklussal halad előre amelynek feltétele hogy a „felv” változó nyitott állapotban legyen vagyis a videókép folyamatos lejátszásban legyen. Ez Ip-kameránál az élő képhez képest 1-2 képkocka lemaradással jelenik meg a program bezárásáig. A videófájl esetében pedig addig folytatódik amíg a videó végére nem ér.

```
while felv.isOpened():
```

A cikluson belül több fontos lépéssel az eredeti színes videókép átalakításra kerül. Elősször a két kiválasztott képkocka közötti abszulút külömbséget keresi meg. Ezt a külömbségi képet tovább módosítjuk szürkeárnyalatos képpé majd a Gaussian elmosási algoritmussal homályosítjuk a képet.

```
kulombseg = cv2.absdiff(frame1, frame2)
# két frame közötti abszolút külömbség
szurke = cv2.cvtColor(kulombseg, cv2.COLOR_BGR2GRAY)
# szürkeárnyalatos átalakítás
elmos = cv2.GaussianBlur(szurke, (5, 5), 0)
# szürkeárnyalatos kép pixeleinek elmosás
```

Az elmosott képen ezután a fekete fehér képpontokat a teljesen fehért és a fekete küszöb irányában eltoljuk.

Így élesebb képet kapjunk a képen látható körvonalak könnyebb felismeréséhez. Ezek után kitöltjük a kpen található üres területeket feketével ezzel tovább erősítve a nagyobb/élesebb vonalakat.

```
_, kuszob = cv2.threshold(elmos, KM, 255, cv2.THRESH_BINARY)
# kontúrkereséshez küszob beállítása
kitoltes = cv2.dilate(kuszob, None, iterations=KI)
# üres területek kitöltése
```

A program ezzel elért a képen található elmozdult alakzatok élesebb megjelenítéséhez. A következő lépés a körvonalak megkeresése majd ezek megjelenítése az eredeti kép felett jelezve az elmozdult alakzatokat. Egy tömb segítségével összegyüjtöttem ezeket a kontúrokat és a program mindig csak a legutolsóként megjelenőt jelöli egy zöld négyzet körérajzolásával.

```
konturak, _ = cv2.findContours(kitoltes, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
# kontúrák megkeresése


konturtomb = np.array(konturak) # kontúrák tömbbe gyüjtése
i = 0

#cv2.drawContours(frame1, konturak, -1, (0, 255, 0), 2)
# minden kontúra vonnalla való megjelenítése

for kontura in konturak: # ciklus kontúrákon való végigléptetéshez
    if cv2.contourArea(kontura) > KM:
# kontúra méretéve szabályozható szűkítés
        konturtomb[i] = kontura
# a legutolsó érzékelt kontúra betöltése a tömbbe
    i = i + 1

    (x, y, w, h) = cv2.boundingRect(konturtomb[0])
# kontúra köré rajzolt kijelölő négyzet sarkai
    cv2.rectangle(frame1, (x, y), (x + w, y + h), (0, 255, 0), 2)
# kijelölő négyzet kirajzolása

z = i

cv2.putText(frame1, "Mozgo Alakok erzekelve: {}".format(z), (10, 20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 3)
# Felső státusz jelzés

cv2.imshow("Biztonsagi Kamera", frame1)
# Frame megjelenítés
cv2.imshow("Szurke arnyalatos elmosott kep", elmos)
# Elmosott szurkearnyalatos kep megjelenitese
cv2.imshow("Ures resz kitoltes utan", kitoltes)
# Kitoltesi iteraciok megjelenitese
frame1 = frame2
# következő frame-re váltás
ret, frame2 = felv.read()
# a következő frame beolvasása a videóképből
```

A program nem csak az eredeti képet jeleníti meg a kiemelt alakzattal hanem a szürkeárnyalatos és a kontúr kitöltött videóképet is a könnyebb beállításhoz.

```
# Videó beállítások nodosításához számozott csúszka

cv2.namedWindow("Parameterek")
cv2.resizeWindow("Parameterek", 900, 150)
cv2.createTrackbar("Kontur", "Parameterek", 20, 255, empty)
cv2.createTrackbar("Kitolt", "Parameterek", 15, 50, empty)


while felv.isOpened():
    KM = cv2.getTrackbarPos("Kontur", "Parameterek")
    KI = cv2.getTrackbarPos("Kitolt", "Parameterek")
```

Paraméter csúszkák segítségével a videófolyam közben is lehet módosítani az érékelésen. Így elérhető hogy részletesebb legyen , több apróbb alakzat felismerésével is vagy erősebb beállításokkal csak a nagyobb látványosabb mozgásokra fokuszáljon.

Az ablak felső részén futás közben látható szöveggel kijelezve hány alakzatot érzékelt a program képen elmozdulni.

```
cv2.putText(frame1, "Mozgo Alakok erzekelve: {}".format(z), (10, 20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 3)
# Felső státusz jelzés
```

A program futás közben bármikor leállítható az Esc –gomb megnyomásával illetve autómatikusan megáll amit a videókép folyam megszűnik. A bezárás után törlődik a memóriából a megnyitott videófolyam és a programhoz tartozó ablakok.

```
if cv2.waitKey(40) == 27:      # kilépés ESC-gombbal
        break

cv2.destroyAllWindows()        # ablak/ok bezárása
felv.release()
```

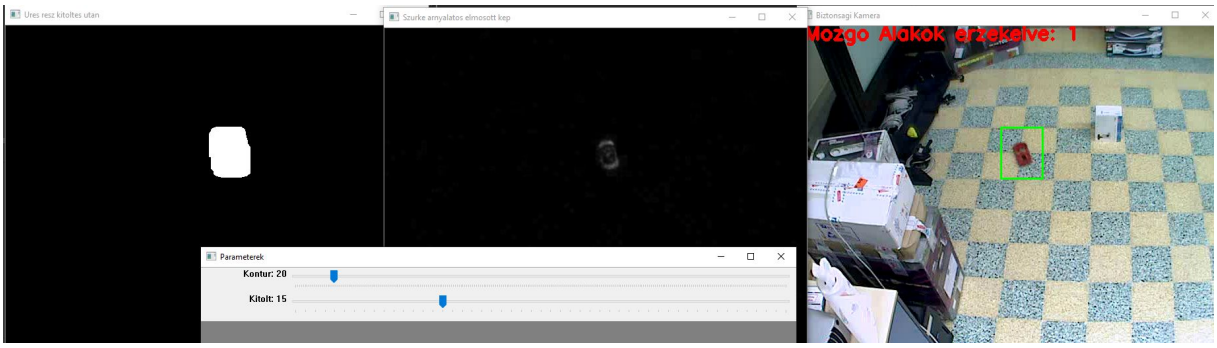
Tesztelés

A programot két elmentett videófájl és a saját mobiltelefonomon futtatott IP-kamera segítségével teszteltem. A két fájlt a program forráskódjával mellékeltem.

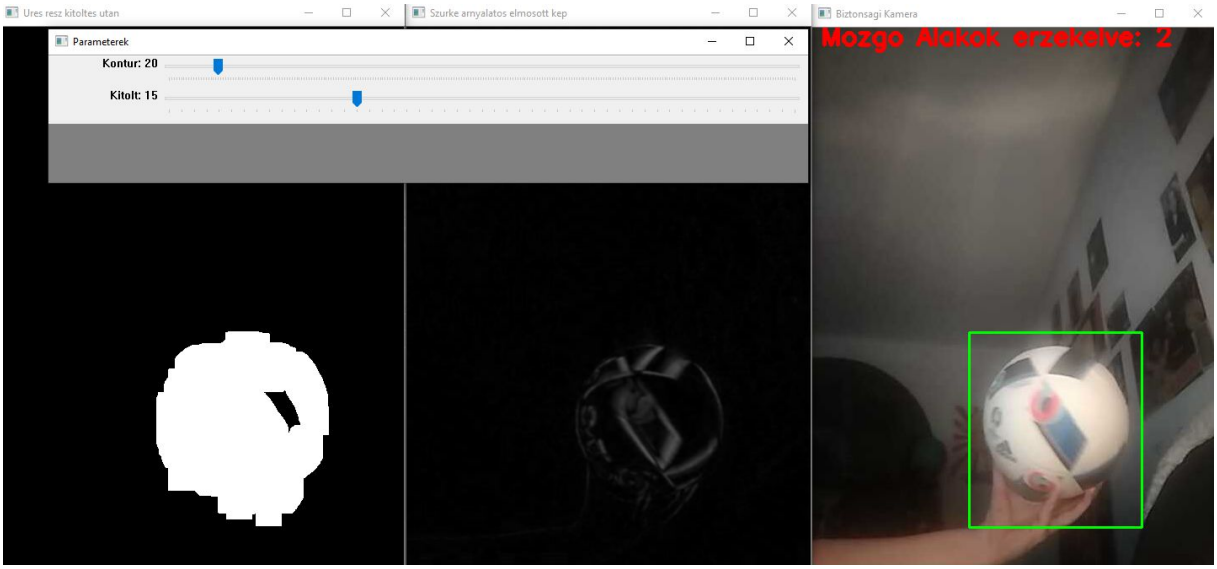
```
#felv = cv2.VideoCapture('vtest.avi')
 ##sétáló emberek tesztvideó
#felv = cv2.VideoCapture('vauto.avi')
 ##játékautó tesztvideó
#felv = cv2.VideoCapture('http://100.120.35.184:8080/video')
 ## IP Kamera példa
#felv = cv2.VideoCapture('http://Lumi:Abyss11@192.168.1.102:7778/video')
## Saját mobilról működő IP camera címe
```

Sétáló emberek – vtest.avi

Kisautó mozgás közben és a megjelenő ablakok – vauto.avi

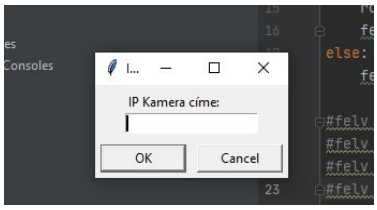


Mobiltelefon IP-kamerakép

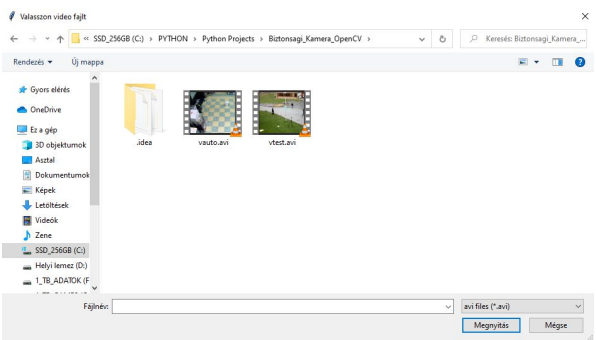


Felhasználói Útmutató

A program elindításakor egy kis felugró ablak jelenik meg amely IP-Kamera címet vár. A címre egy példa helyi hálózatról: http://Felhasználó:Jelszó@192.168.1.100:7778/video .



Amennyiben nincs IP-kamreához tartozó cím a Mégse gombra kattintva a program tovább halad egy mentett videófálj megnyitásával.



Amennyiben egyik esetben sem lehetett betölteni megfelelő videófolyamot a program kilép.

A program futása közben a Paraméter csúszkák segítségével változtatni lehet a kontúrok élességén illetve a feketével kitölések ismétlődésének számát. Ezekkel a beállításokkal valós időben látható a két mellékelt videóképen a felismert alakzatok kirajzolódása.