FORGALOMSZÁMLÁLÓ ALKALMAZÁS

Készítette:
Látrányi Gergely Balázs
Neptun kód: A6SGZH

Gépi látás
2020/21/1. félév

Forgalomszámlálás Dokumentáció

Tartalomjegyzék

Bevezetés

Nyilván a mai világban a manuális **forgalomszámlálás** igen munka- és időigényes kötelezettség. Viszont mára már eljutott oda a technológia, hogy ezt a feladatot költséghatékony módon automatizálni lehessen. **Célom** egy olyan **számítógépes program** elkészítése amely képes egy rögzített állású **videófelvétel** alapján a rajta szereplő **objektumok** (főleg gépjárművek) közúton történő áthaladás észlelésére és **megszámlálására**. Ez egy elég **komplex feladat**, hiszen a számítógépet megkell tanítani, arra hogy hogyan ismerje fel a gépjárműveket valamint mivel több képkockán is szerepel ugyan az az gépjármű így ezt számolni sem könnyű.

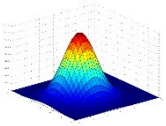
Megoldáshoz szükséges elméleti háttér

Ahhoz, hogy hatékony legyen a megoldásunk, több különböző képfeldolgozási lépést hajtunk végre.

0. lépés: A videófelvétel **beolvasása** és képkockánként történő feldolgozása (megelőző és következő képkocka). A továbbiakban a videóklip-et két állóképként fogjuk kezelni.

1. lépés: Mindkét képet **szürkeárnyalatossá** konvertálunk.
Erre azért van szükség, hogy a számítógépnek lényegesen gyorsabban tudjuk vele számolni (1bit elég), valamint a színek nem segítenek az élek megtalálásában, ezért ez egy fontos lépés. Némi zajvédettséget is eredményez. A két kép kivonásához (absdiff,3.lépés) programozástechnikai szempontból muszáj szürkeárnyalatos képnek lennie.

2. lépés: **Gauss homályosítást** (Gauss Blur) 5x5-ös kernellel alkalmazunk.
Ennek a lépésnek segítségével a kép élesebbé válik, a Gauss-i zajt eltávolítja.



3. lépés: A második **képkockából kivonjuk** az azt megelőzőt (elsőt).
Így pontosan megkapjuk a **változást,** két kép közötti pontos különbséget.
Programozási szempontból absdiff segítségével pixelenként vonjuk ki őket egymásból.

Alapfeltételezés 1: **Intenzitás megmaradás**: Vagyis az elmozdulás következtében a pixelek **nem** változtatják meg intenzitásukat.
Alapfeltételezés 2: **Térbeli koherencia**: Mivel a szomszédos pixelek a térbeli objektum felületének is szomszédos pontjai, ezért hasonlóan mozognak.
Alapfeltételezés 3: **Időbeli állandóság** : Egy felületelem képi elmozdulása időben lassan változik.

4. lépés: **Küszöbölést** (Threshold) alkalmazunk.
Ennek a metódus segítségével megkapjuk a kép körvonalát(éleit).

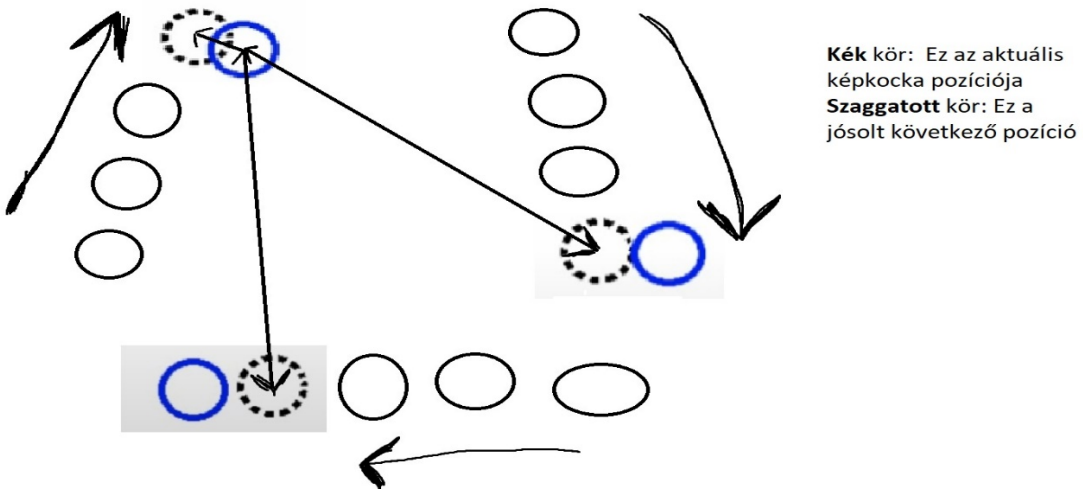
5. lépés: A **körvonalat** amit a küszöböléssel kaptunk „**megnyújtjuk**”(dilation). Ennek segítségével a képen szereplő objektumok nagyobbak (vastagabb) lesznek.

6. lépés: A körvonal **„elkoptatása”** (erosion).
Ennek köszönthetően a kicsi fehér (small white noise) zajt távolítja el, és ha az előző lépés annyira megnövelt egy objektumot, hogy egy másikkal összefonódott, akkor azt ketté veszi.

7. lépés: **Kontúrozást** használunk.
Olyan metódus amely Objektum felismerésre és Alak megállapításra használható. Egy körvonalként fogható fel, amelyben az összes pont azonos színnel vagy intenzitással rendelkezik.

8. lépés: **Konvex burokká** (folt) alakítjuk a 7.lépésben kapott eredményt.
Ennek köszönhetően az objektum alakja eléri maximális nagyságát, konvex szöggé alakul.

9. lépés: A következő képkockára **kiszámoljuk**, hogy a jelenlegi folt körülbelül hol lesz: Majd ezt összevetjük a következő képkocán szereplő folt **tényleges pozíciójával** és amelyikhez a **legközelebb** van, annak a folytatásához hozzáadjuk.
Az egyszerűség kedvéért a gépjármű alakja helyett köröket használtam.



10. lépés: Az előre definiált **egyenesen való áthaladás** ellenőrzése.
Ezt úgy lehet számolni, hogy meg kell nézni, hogy az éppen követett folt a megelőző képkockán a vonal alatt helyeszkedett el és a következő képkockán pedig már a vonal fölött volt.

11. lépés: **Áthaladás** esetén a Forgalomszámláló **összeg**et megnöveljük 1-gyel.

A megvalósítás terve és kivitelezése

A programfejlesztésnél a **fő** elv a megvalósításhoz és kivitelezéshez az volt, hogy a fenti metódusokat **külön-külön** megvalósítom, majd ezeket egymásba **integrálom**. Ennek eredményeként szinte mindennek saját függvénye van. Legtöbb esetben a változók már deklarációkor inicializálva vannak. A végső megoldás több beépített OpenCV által megírt függvényt használ fel.

A program futtatásához szükséges eszközök:

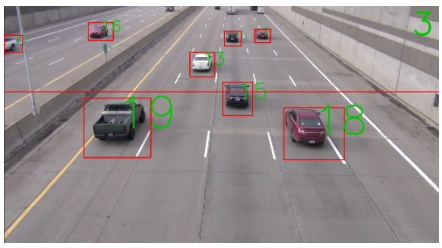
- ✓ C++ Program futtatására alkalmas fordító
- ✓ A program futtatásához szükséges OpenCV csomag telepítése
- ✓ Rögzített állású videófelvétel gépjármű forgalomról
- ✓ Blob.h, Blob.cpp és main.cpp forrásfájlok.

Tesztelés

Tesztelési céllal **több** különböző rögzített állású **videofelvételt** dolgoztam fel. Megvizsgáltam ezeket felbontás, megvilágítás és forgalomszámlálási végösszeg szempontjából.

1. Teszt1.mp4

Ez egy elég tökéletes videó a programom számára. A videó hossza **31** másodperc **1280x720** HD felbontású Itt éri el az algoritmus a maximális hatékonyságát (**100%**), miszerint a videón szereplő összes gépjárművet felismerte. Számmal pontosan kifejezve **52**-gépjárművet azonosított.

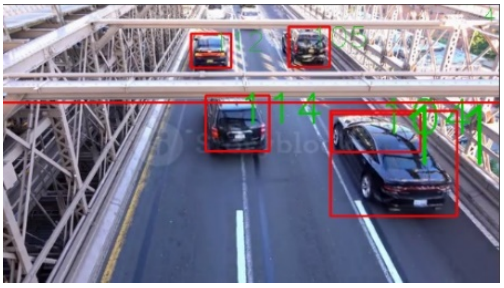


Teszt1 Előnézet

Itt azért érhette el ezt a hatékonyságot mert semmi zaj nincs a képen, alapjáraton világos a kép, valószínűleg ez egy kora délelötti videó,semmi napsütés vagy más fényforrás nem zavar be, a felbontása is már elég magas. Megállapítható, hogy a kamera egy kicsit lefelé néz, így pont a videó alján/elején a legnagyobbak a gépjárművek.

2. Teszt2.mp4

A videó 640x340 felbontású, 18 másodperc hosszú. Itt sajnos a program megközelítőleg csak az autók 20%-át (20ból 4-et) ismerte fel.
Ez sajnos annak köszönhető, hogy az autók elég lassan mennek. Ez azért probléma mert a program csak 5 folt erejéig figyeli a járműveket, ami ilyen lassú sebesség esetén, még az előtt eltűnik minthogy elérné a kijelölt sávot. Manuálisan lehet javítani ezen, nyílván több folt erejéig kell figyelni ezeket, valamint a forgalomszámláló egyenest is lejjebb kell pozícionálni. Lejjebb pozícionálva a gépjárművek 90% át (18jármű) ismerte fel.



3. Teszt3.mp4

A videó 640x360 felbontású, 28 másodperc hosszú. Itt sajnos a program megközelítőleg csak az autók 19%-át(28 gépjárműt az 150 ből) ismerte fel. Ez sajnos a gyenge fényviszonyoknak köszönhető, esetleg külső fényforrás segítségével lehetne rajta növelni.

Teszt3 Előnézet

4. Teszt4.webm

A felvétel 600x316 felbontású 25 másodperc hosszú. 36 járműből 31 járműt ismert fel. Ez 85%-nak felel meg.

Teszt4 Előnézet

Összességében a 4 videó alatt **258** jármű haladt el ebből **129**-et ismert fel. Ez pontosan a gépjárművek **50%**-nak a felismerését jelenti. De ha nem vesszük figyelembe az éjszakai felvételt(Teszt3.mp4 ahol elég rosszul teljesített) ez esetben 108 objektumból 101 járműt ismert fel ami **93,5%-**nak felel meg.

Ha a körülmények biztosítottak (kellő mennyiségű fény, a kamera az utat követi és kicsit lefele néz ekkor biztos, hogy a gépjárművek **93,5%-**át felismeri. A program ellenáll az időjárási körülményeknek.(esőt vagy havazást kiszűri).

Felhasználói leírás

Ahhoz, hogy a Forgalomszámláló programot használni tudjuk ahhoz egy olyan C++ nyelvű **fejlesztőkörnyezet** szükséges ami képes Python (OpenCV) beágyazott függvények futtatására.

Ilyen például a Microsoft **Visual Studio** OpenCV csommaggal telepítve.

Főbb Funkciók:

- **Videó beolvasása:**
  Egy videófájl beolvasásához a Forráskód **45-sor**ában található capVideo.open utáni zárójeles részben kell megadni a fájl nevét kiterjesztéssel együtt idézőjelek között, valamint a videófájlt a program könyvtárába másolni.
  pl. capVideo.open**("Teszt1.mp4")**
- **Forráskód futtatása**:
  A fejlesztői környezet alapértelmezett futtatási billentyűkombinációjának lenyomásával. Microsoft Visual Studio esetében **Crtl+F5**.
- **Objektumok nyomon követése:**
  Az alkalmazás a felismert objektumokat (gépjárműveket) önvezérelt módon piros keretbe teszi és ellátja egy azonosítóval. A szoftver előre kijelöl egy egyenest (piros sáv) amin ha egy objektum áthalad a forgalomszámláló összeget megnöveli 1-gyel.
- **Forgalomszámláló összeg megjelenítése:**
  A program automatikusan a jobb felső sarokban megjeleníti az összes felismert objektum (gépjármű) számát zöld betűszínnel.
- **Videóból történő kilépés:**
  A program futása közben az ESC-billentyű lenyomásával léphetünk ki, valamint a program magától kilép ha a lejátszandó videó a végére ért. (ekkor kiírja, hogy „end of the video” majd leáll)

Irodalomjegyzék

[1] OpenCV Weboldal: LINK

[2] Anton Andriyenko, Konrad Schindler, Stefan Roth. Discrete-Continuous Optimization for Multi-Target Tracking LINK

[3] Khac-Hoai Nam Bui, Hongsuk Yi, and Jiho Cho Korea Institute of Science and Technology Information. A Vehicle Counts by Class Framework using Distinguished Regions Tracking at Multiple Intersections. LINK

[4] Logisys https://logisys.hu/termekek/vca-logipix-video-analizis/forgalomszamlalas

[5] Dankovics József István Szakdolgozat. LINK
http://iar.bmfnik.hu/2008_2009/TrafMon/docs/TrafMon_Dokumentacio_v1.pdf

[6] IEEE Video-Based Distance Traffic Analysis: Application to Vehicle Tracking and Counting LINK
https://ieeexplore.ieee.org/abstract/document/5654483

[7] Kató Zoltán Optikai áramlás és követés LINK

http://www.inf.u-szeged.hu/~kato/teaching/IpariKepfeldolgozas/08-Motion.pdf

[8] Dr. Gácsi Zoltán, Dr. Barkóczy Péter. Számítógépi képelemzés. LINK
https://regi.tankonyvtar.hu/hu/tartalom/tamop425/0001_1A_A1_01_ebook_szamitogepi_kepelemzes_eloadas_vazlat/adatok.html