Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Meglécz Eszter

KJ2CVF

MOZGÓ OBJEKTUMOK REAL-TIME AZONOSÍTÁSA A KÖZLEKEDÉSBEN

KONZULENS

Dr. Max Gyula

BUDAPEST, 2014

Tartalomjegyzék

Összefoglaló **4**

Abstract **6**

1 Az OpenCV (Open Source Computer Vision Library) **7**

2 Ismerkedés az OpenCV-vel **7**

2.1 Kép megjelenítése 7

2.2 Videó megjelenítése 8

3 Az általam készített program **9**

4 Videóból háttér kiszűrése **10**

4.1 Kódrészlet a háttér kiszűréséből 10

4.2 A BackgroundSubtractorMOG2 osztály 11

4.3 Gaussian Mixture Model (GMM) 11

4.4 Részeredmények a háttér kiszűrése során 12

4.4.1 Az ehhez tartozó kódrészlet 13

4.4.2 Az első videóból a háttérszűrés eredménye 20 másodpercenként 13

4.4.3 A második videóból a háttérszűrés eredménye 20 másodpercenként 15

5 Objektum-felismerő algoritmusok **18**

5.1 Mozgó objektumok felismerése: Videóból háttér kivonása (fekete-fehér) (Frame Differencing) 18

5.1.1 A kivonást elvégző függvényhez tartozó kódrészlet 19

5.1.2 A kivonást elvégző függvény használata 19

5.2 SIFT algoritmus (Scale-Invariant Feature Transform) 19

5.3 Haar-szerű jellegzetességek alapján való felismerés 19

5.4 SURF algoritmus (Speed Up Robust Features) 20

5.4.1 Autó felismerése: Egy képen felismerés SURF algoritmussal (fekete-fehér) 21

5.4.2 Autó felismerése: Videóban felismerés SURF algoritmussal (fekete-fehér). 22

5.4.3 Autó felismerése: Videóban felismerés SURF algoritmussal (színes) 22

6 Jövőbeli tervek **24**

Irodalomjegyzék **25**

HALLGATÓI NYILATKOZAT

Alulírott **Meglécz Eszter**, hallgató kijelentem, hogy ezt a dolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik.

Kelt: Budapest, 2014. 10. 21.

...……

Meglécz Eszter

Összefoglaló

A mozgóképes forgalom-megfigyelő rendszerek gyorsaságának növekedését a számítástechnikai teljesítmények robbanásszerű bővülése tette lehetővé. Az elmúlt években lezajlott fejlődés már arra is lehetőséget ad számunkra, hogy real-time módszerekkel analizálhassuk a kamerák által készített felvételeket. A képfeldolgozás módszertana ugyan már évek óta ismert, de mivel egy-egy, a vizsgálat szempontjából megfelelő felbontású és darabszámú kép tárolása igen nagy memóriaterületet és gyors processzor teljesítményt követelt, a képek real-time feldolgozása a megfigyelő rendszerek lassúsága miatt nem volt kivitelezhető, ezért megoldásként a legtöbb esetben a „humán" megfigyelő módszer jöhetett csak számításba. Már rövid ideig tartó forgalom-megfigyelés is hihetetlenül nagy adatmennyiséget produkál. Megfelelő paraméterek mellett, bár minden fontos jellemző megtalálható a képeken, ami pl. egy város forgalomirányításához, rendőrségi megfigyeléséhez, stb. szükséges, mégis nagy problémát jelent a feldolgozó szoftver számára ebből a hatalmas adatmennyiségből az adott feladathoz szükséges részelemek kiválasztása.

A képek által hordozott nagymennyiségű információk közül dolgozatomban a közutakon mozgó járművek felismerését tanulmányoztam. Rögzített videós felvételek segítségével tettem lehetővé az események megismételhetőséget és az ellenőrizhetőséget. Egy autópálya felüljáróról készítettem videókat a szembejövő autókról, amelyek a későbbiekben az analízisem alapjai lettek. A felismerési folyamat megkezdéséhez először külön kellett választanom a hátteret az előtértől. A háttér kiszűréséhez szükség volt a Gaussian Mixture valószínűségi modell használatára is. A tényleges megfigyelési terület azonosítása után kezdhettem meg a mozgó objektumok megfigyelését. A mozgó objektumok felismeréséhez többféle algoritmust is megismertem, ezek közül néhányat alkalmaztam is, majd összehasonlítottam őket pontosság és gyorsaság szempontjából. Az algoritmusoknak fekete-fehér és színes változatait is tanulmányoztam és értékeltem. A feladat elkészítéséhez az OpenCV nyílt forráskódú számítógépes látást megvalósító függvénykönyvtárát használtam fel. Magát a forgalom analizálását végző program megírására pedig a C++ nyelvet választottam, mert a hozzá tartozó OpenCV könyvtár a legteljesebb. Az OpenCV-ben található több mint 500, részben adott processzorra optimalizált függvény a későbbiekben majd arra is

lehetőséget ad számomra, hogy ne csak egyszerű feladatok feldolgozását, hanem bonyolultabb (pl. szabálytalanságokat elemző, sebességmérő) programcsomagot is kialakíthassak.

Abstract

The rise in the speed of the traffic observation systems can be accredited to the exponentially growing performance of computing systems. The improvement of the last years even gives us the opportunity to analyse the videos recorded by cameras. The methodology of image processing has been known since several years, but the storage of proper amount of pictures with proper resolution in terms of the observation required a huge memory capacity and a fast processor, which made the real-time analysing of the pictures nearly impossible. This slowness of the observation systems left us with the remaining possibility of ’human’ observation. Even a short period of traffic observation can produce a huge amount of data. For the processing software it is a significant problem to sort out the neccessary sub-elements for the actual project, even if the parameters are correct and all of the important traits which are required e.g. for the traffic management or police surveillance of a city, are present on the pictures.

From the huge amount of information contained by the pictures, this dissertation studies the recognition of moving vehicles on public roads. The repeatibility and the controllability of the events are enabled with fixed video records. After recording the videos of the oncoming traffic from an overpass above a highway, they were analysed thoroughly. The first step in the recognition process is to subtract the background from the foreground. To subtract the background the Gaussian Mixture probabilistic model was used. The observation of the moving objects could be started only after the identification of the real observation area. First, I got acquainted with several algorithms for detecting moving objects, then I tested some of these algorithms and compared them by precision and speed. For this, both the black and white and color versions of the algorithms were used and rated. I used the OpenCV which is an open source computer vision library of programming. To code the traffic analysing program the C++ language was chosen, because it has the fullest OpenCV library. OpenCV contains more than 500 functions from which some are optimalized to the given processor, and these functions provide the opportunity to create a more complex program package (e.g. detecting irregularities, speed measuring) in the future.

1 Az OpenCV (Open Source Computer Vision Library)

Az OpenCV egy nyílt forráskódú számítógépes látást megvalósító függvénykönyvtár. Elsődleges interfésze C++, de már van Python, MATLAB, Java nyelvű interfész is, de például C#-pal és számos másik programozási nyelvvel is használható csomagolók (pl. C# esetében az EmguCV) segítségével.

Én kipróbáltam a C++-os, Java-s, és C#-os verziót is, és végül a C++ nyelvet választottam a programom megírására, mert a hozzá tartozó OpenCV könyvtár a legteljesebb, az összes függvény nincs átírva a többi programozási nyelvre.

2 Ismerkedés az OpenCV-vel

Az OpenCV a képeket, mátrixokat, hisztogramokat több dimenziós tömbökben tárolja. OpenCV-ben tárolhatóak az adatok dinamikus struktúrákban (pl. CvSeq, CvGraph), valamint léteznek segéd adattípusok is, mint a CvPoint vagy a CvSize.

Képek írására és olvasására beépített OpenCV függvények állnak rendelkezésünkre, ezek többféle kiterjesztésű fájlokkal tudnak dolgozni, akár AVI videókkal. Lehet ablakokat készíteni, azokban a képeket és képek sorozatát, vagyis videókat megjeleníteni. Egyszerűbb interaktivitás kezelésére is lehetőség van, például billentyűzet és egér használatára, ezeket alkalmaztam is a programomban.

2.1 Kép megjelenítése

```
string filename = "photo.jpg";
Mat img = imread(filename);

namedWindow("image", CV_WINDOW_AUTOSIZE);
imshow("image", img);
…
```

Képmegjelenítést többféle módon is végre lehet hajtani. A bemutatott módszer mellett például másik lehetőség nem Mat, hanem IPlImage típusként tárolni a képet, ekkor a cvLoadImage függvény végezheti a kép betöltését, és a cvShowImage pedig a megjelenítését.

Ablak létrehozása a namedWindow vagy cvNamedWindow függvényekkel valósítható meg, paraméterként átadható az ablak neve, valamint a flag-ek, például CV_WINDOW_AUTOSIZE (automatikus méretezés beállítása). Az ablakok megszünetethetők a destroyWindow és cvDestroyWindow függvényekkel.

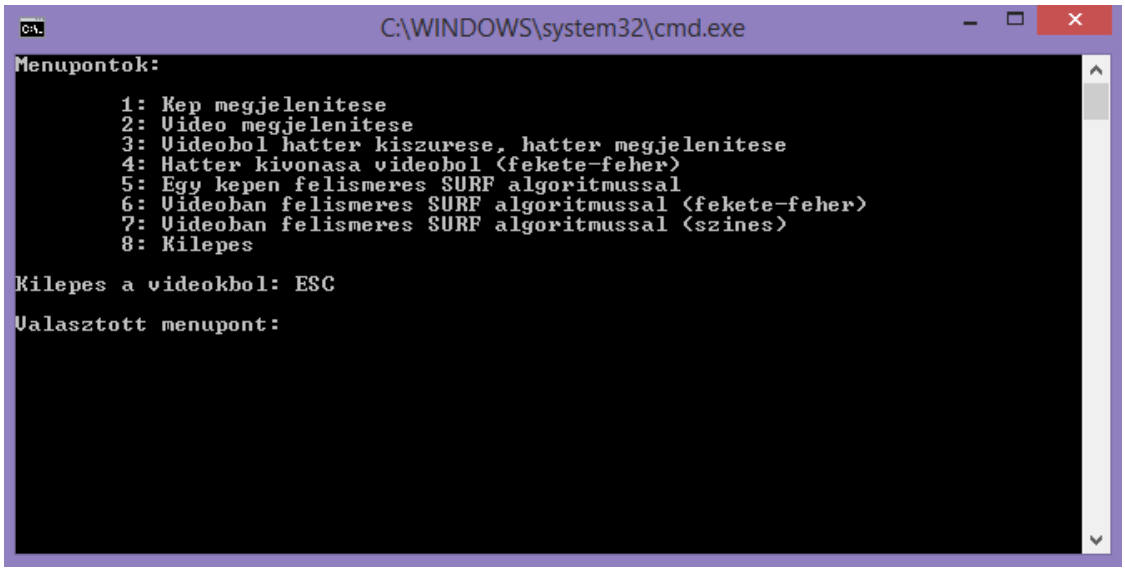
2.2 Videó megjelenítése

Az alábbi programrészlet betölt egy avi kiterjesztésű videót, létrehoz egy ablakot, amelyben meg fogja jeleníteni, majd a videó összes képkockáját egymás után megjeleníti.

```
char* name = "Video";
const char* filename = "video.avi";
cvNamedWindow(name, CV_WINDOW_AUTOSIZE );
CvCapture* capture = cvCreateFileCapture(filename);
IplImage* frame;
while(1)
{
     frame = cvQueryFrame(capture);
     if(!frame) break;
     cvShowImage(name, frame);
}
cvReleaseCapture(&capture);
cvDestroyWindow(name);
```

3 Az általam készített program

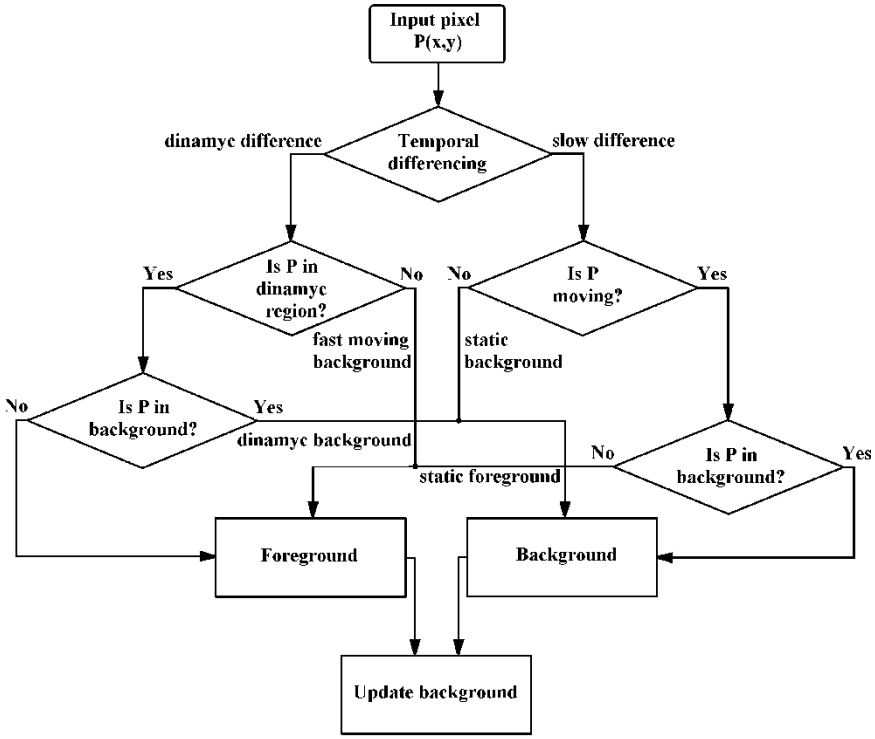
A programot elindítva az alábbi menü jelenik meg a konzolon:



1. ábra: A program elindítás után

Az első két menüpont a már említett kép- és videó megjelenítés. A többi funkciót részletesebben is bemutatom.

4 Videóból háttér kiszűrése



2. ábra: A háttérkészítés folyamata

A háttér kiszűrését a BackgroundSubtractorMOG2 beépített osztályt használva oldottam meg. A program megvizsgálja a videó minden egyes képkockáját, és fokozatosan frissíti a szűrt hátteret. A háttér kiszűrését gyakran alkalmazzák videókban mozgó objektumok felismerésére.

4.1 Kódrészlet a háttér kiszűréséből

```
bool shadow_detection = false;
int history = 999999999999999999;
int distance_threshold = 16;

cv::Mat frame;
cv::Mat fore;
cv::VideoCapture cap("video.avi");
cv::BackgroundSubtractorMOG2 bg(history, distance_threshold,
shadow_detection);

for(;;)
{…
      cap >> frame;
      bg.operator()(frame,fore);
      bg.getBackgroundImage(back);
…}
```

4.2 A BackgroundSubtractorMOG2 osztály

Az OpenCV-ben a BackgroundSubtractorMOG2 osztály a BackgroundSubtractor osztály leszármazottja, ami pedig az Algorithm osztályé.

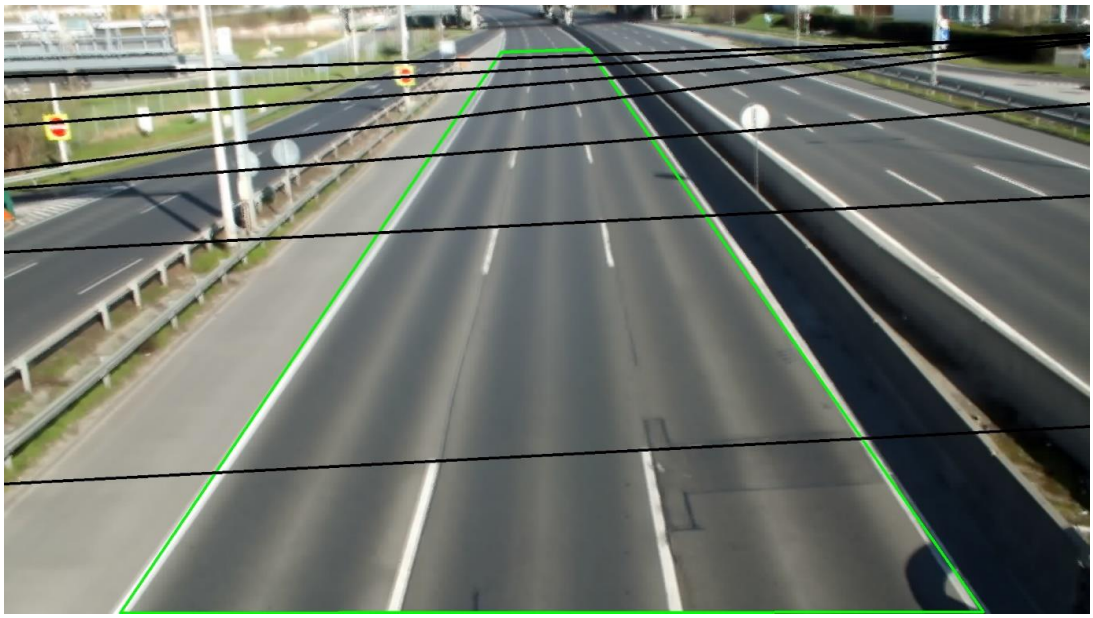
Gaussian Mixture alapú háttér és előtér szétválasztó algoritmus szerint működik. [1]

4.3 Gaussian Mixture Model (GMM)

A Gaussian Mixture Model egy statisztikában használt valószínűségi modell, amely feltételezi, hogy az összes adatpont egy véges számú ismeretlen paraméterű Gauss-eloszlásból van generálva. [2]

A videó összes pixelének intenzitás értéke modellezhető Gaussian Mixture modellel. Egy egyszerű heurisztika meghatározza, mely intenzitások tartoznak legnagyobb eséllyel a háttérhez. Azok a pixelek, amelyek nem felelnek meg ennek, az előtérhez tartoznak. [3]

A GMM a k-közép módszer algoritmusnak egy EM-algoritmussal (expectation-maximization algorithm) kialakított variációja, amely valószínűségi feladatokat kezel klaszterekre determinisztikus feladatok helyett, valamint többváltozós Gauss-eloszlásokra átlagok helyett. [2] A k-közép módszer egy nem hierarchikus klaszterelemzési algoritmus, amely a dimenziócsökkentést minden egyes elemnek ahhoz a klaszterhez sorolásával oldja meg, amelyiknek a középpontja a legközelebb esik az adott elemhez. A klaszterek azok a csoportok, amelyeket az adattömbökből dimenziócsökkentő eljárás során homogén csoportokba soroltak. [4]



3. ábra: Az elkészült háttér és a bejelölhető vonalak

Ha elkészült a végleges háttér, ezt a program kiírja egy jpg kiterjesztésű fájlba, majd megjeleníti. Ekkor lehetőség van kattintásokkal bejelölni a vizsgált útszakaszt (a négyszög csúcspontjait) és a felezővonalakat (végpontjait), erre látható egy példa a 3. ábrán. Ezekre az utat határoló pontokra később, a program továbbfejlesztése során lesz szükség, hogy csak a lényeges területet vizsgálja, felesleges az egész képen elvégezni a számításokat. A vizsgált területet (Region of interest) a cvSetImageROI beépített függvénnyel lehet majd beállítani. A vízszintes vonalak pontjait a sebesség meghatározásakor lehet majd felhasználni, hogy be lehessen határolni, mikor milyen messze van az autó a videóban (minden vízszintes vonal között ugyanannyi távolság van a valóságban, ez például Google Earth műholdas nézetének segítségével lemérhető).

Két videót készítettem, az egyiket borúsabb időben, a másikat napsütésben, az eredményeken látszik, hogy abból a rossz időben készült videóból kevésbé jó minőségű hátteret szűr ki a program.

4.4 Részeredmények a háttér kiszűrése során

A háttér elkészítése során 20 másodpercenként kimentettem az aktuális állapotát a szűrt háttérnek.

4.4.1 Az ehhez tartozó kódrészlet

```
if ((elapsed_time % 20) == 0) imwrite(file_name ,back);
```

4.4.2 Az első videóból a háttérszűrés eredménye 20 másodpercenként

4. ábra

5. ábra

6. ábra

7. ábra

8. ábra

9. ábra

4.4.3 A második videóból a háttérszűrés eredménye 20 másodpercenként

10. ábra

11. ábra

12. ábra

13. ábra

14. ábra

15. ábra

16. ábra

17. ábra

5 Objektum-felismerő algoritmusok

5.1 Mozgó objektumok felismerése: Videóból háttér kivonása (fekete-fehér) (Frame Differencing)

18. ábra: Egy kép a videóból, amelyen a frame diffecencing algoritmust alkalmaztam

A frame differencing is egy mozgást detektáló algoritmus, amelyet teszteltem. Megnéztem, milyen eredményt kapok, ha a videó minden egyes képkockájából kivonom a már létrehozott szűrt hátteret (mindkettőt fekete-fehérre állítva), de így a várt eredménnyel ellentétben nemcsak a mozgó objektumok maradtak fehér színűek a képen, hanem a háttér egyes részei is, ez látható a 18. ábrán. Ennek oka a kamera szél miatti enyhe mozgása. Erre a problémára egy megoldás lehet a küszöbérték használata a kivonás során.

Matematikai egyenlettel a következőképpen írható le:

$$P[F(t)] = P[I(t)] - P[B]$$

Küszöbérték használatával:

$$|P[F(t)] - P[F(t+1)]| > \text{Threshold}$$ [3]

5.1.1 A kivonást elvégző függvényhez tartozó kódrészlet

```
void FindDifference(cv::Mat src1, cv::Mat src2, cv::Mat &dst, int
threshold) {
     dst = cv::abs(src2 - src1);
     cv::threshold(dst, dst, threshold, 255, cv::THRESH_BINARY);
} [5]
```

5.1.2 A kivonást elvégző függvény használata

```
FindDifference(frame,background,foreground, 12);
```

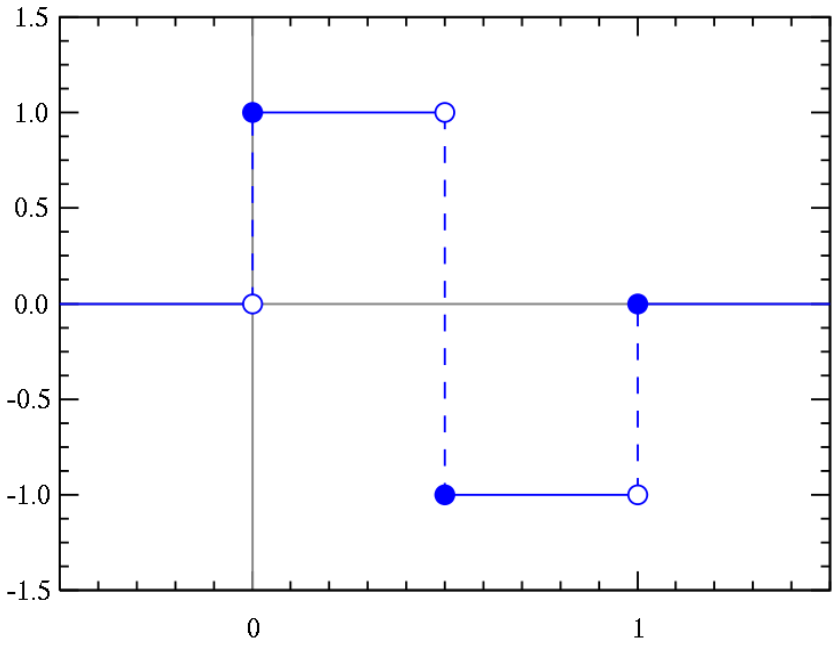
Ez a metódus egyesével vizsgálja a videó összes képkockáját, és egy for ciklusban folyamatosan kivonja az aktuális képből (frame) a hátteret (background), majd a végeredményt (foreground) jeleníti meg.

5.2 SIFT algoritmus (Scale-Invariant Feature Transform)

A SIFT algoritmus egy számítógépes látásban használt jellemző alapú objektumfelismerő algoritmus. A referencia képekből jellemzőket számol, ezeket eltárolja egy adatbázisban, majd az új képek elemzésekor is kiszámolja a jellemzőket, és az adatbázisban keres hasonló értékeket. Ha a referencia képekkel különböző szempontok (típus, elhelyezkedés) alapján megegyezik, jelzi az objektum felismerését. Invariáns a méret és affin transzformációkra (például lineáris transzformációk, eltolás), valamint részben invariáns a megvilágítás megváltozására. A SIFT egy felügyelt tanuló algoritmusként is felfogható. A jobb teljesítmény elérésének érdekében létrejöttek a SIFT-nek különböző változatai, például a RIFT (Rotation Invariant Feature Transform), amely a SIFT forgatás invariáns változata. [6] [7]

5.3 Haar-szerű jellegzetességek alapján való felismerés

A Haar-wavelet átméretezett négyzet-alakú függvények szekvenciája. Az első ismert wavelet bázisként tartjuk számon, valamint ez a lehető legegyszerűbb wavelet. Mivel Haar-wavelet a Daubechies wavelet egy speciális változata, ezért D2 néven is ismert. Használata tanító példaként terjedt el. [9]



19. ábra: A Haar-wavelet

A Haar-szerű jellegzetességek objektumfelismerésben használt digitális képek jellegzetességei. Az első real-time arcfelismerő is ezen alapul.

Az integrál képek használatának köszönhetően mérettől független a számítási sebessége, használatának ez a gyorsaság a legnagyobb előnye a többi jellegzetességgel szemben.

5.4 SURF algoritmus (Speed Up Robust Features)

A SURF algoritmus egy OpenCV-ben már implementált detektáló és leíró algoritmus. A SURF jellegzetesség megfeleltetés segítségével jól alkalmazható objektumok felismerésére, valamint 3D rekonstrukcióra is használható. Részben a SIFT-re épül, de annál jobb teljesítményű, forgatás és átméretezés invariáns. [6] Az algoritmus 2D Haar wavelet-ek összegén alapul, és a folyamatot integrál képek használatával gyorsítja. [8]

5.4.1 Autó felismerése: Egy képen felismerés SURF algoritmussal (fekete-fehér)

Alkalmaztam a SURF algoritmust egy képen, és az alábbi eredményt kaptam:

20. ábra: Egy kép, amelyen a SURF algoritmust alkalmaztam

A 20. ábrán látszik, hogy jól felismeri az autót a képen, a detektáló pontok nagy része tényleg az autóra esik. A másik két pont kiszűrhető például úgy, hogy csak azokat a pontokat vesszük figyelembe, amelyek egymástól adott távolságon belül vannak.

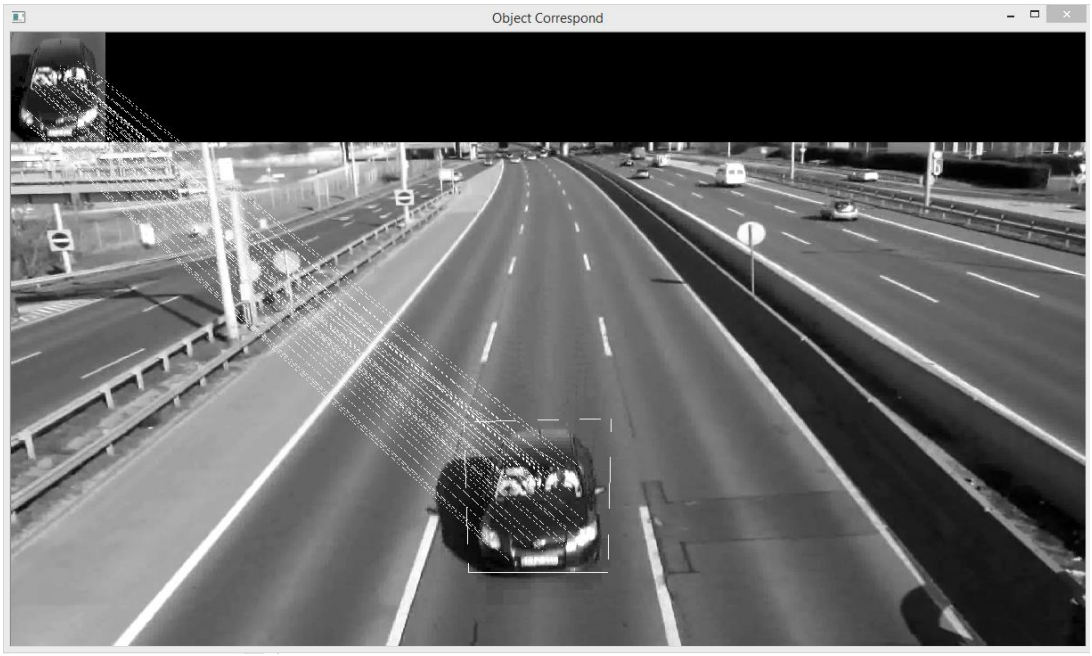
5.4.1.1 Az ehhez tartozó kódrészlet

```
IplImage* object = cvLoadImage(object_filename, CV_LOAD_IMAGE_GRAYSCALE);
IplImage*image=cvLoadImage(scene_filename, CV_LOAD_IMAGE_GRAYSCALE);
CvMemStorage* storage = cvCreateMemStorage(0);
CvSeq* objectKeypoints = 0, *objectDescriptors = 0;
CvSeq* imageKeypoints = 0, *imageDescriptors = 0;
CvSURFParams params = cvSURFParams(500, 1);
…
cvExtractSURF(object, 0, &objectKeypoints, &objectDescriptors, storage,
params);

cvExtractSURF(image, 0, &imageKeypoints, &imageDescriptors, storage,
params);

…
```

```
flannFindPairs(objectKeypoints, objectDescriptors, imageKeypoints,
imageDescriptors, ptpairs);
```

5.4.2 Autó felismerése: Videóban felismerés SURF algoritmussal (fekete-fehér)

Az előző kódot alakítottam át kép helyett videóban felismeréssé. A mozgó járművet a képhez hasonlóan jól felismeri, minél közelebb van, annál pontosabb. Amikor már a kép aljára kerül az autó, be is keretezi a program, mert megtalálta a keresett objektumot, ez látható a 21. ábrán.



21. ábra: Egy kép a videóból, amelyen a SURF algoritmust alkalmaztam

Ez a program elég lassan működik, és az is hátránya, hogy csak fekete-fehér képeken alkalmazható, így kerestem a SURF algoritmus megvalósításának egy színes változatát.

5.4.3 Autó felismerése: Videóban felismerés SURF algoritmussal (színes)

A színes változat a fekete-fehérnél gyorsabb, és pontosabban ismeri fel az objektumokat, ezzel kell majd tovább haladni a program továbbfejlesztése során. Erre is jellemző, hogy minél közelebb kerül az autó, annál jobban ismeri fel. Szükséges lesz még kiszűrni a nem jó pontokat.

5.4.3.1 Kódrészletek

```
…
BFMatcher matcher(NORM_L2);
…

//-- A betöltött képen a kulcspontok detektálása
SurfFeatureDetector detector(minHessian);
detector.detect(img_1, keypoints_1);

//-- A betöltött kép kulcspontjaihoz leíró hozzárendelése
SurfDescriptorExtractor extractor;
extractor.compute(img_1, keypoints_1, descriptors_1);

std::vector< DMatch > matches;
…
//-- A kameraképen kulcspontok detektálása
detector.detect(camera_frame, keypoints_2);

//-- A kameraképen kulcspontjaihoz leíró hozzárendelése
extractor.compute(camera_frame, keypoints_2, descriptors_2);

//-- Jellegzetességek megfeleltetése
matcher.match(descriptors_1, descriptors_2, matches);
```

22. ábra: Amikor messze van az autó

23. ábra: Amikor közel van az autó

6 Jövőbeli tervek

A programot lehetne fejleszteni azzal, hogy ne csak egy adott autót detektáljon helyesen, hanem többet is, valamint egy képkockán több autót ismerjen fel. Ha már jól működik az autók felismerése, lehet a mozgásukat vizsgálni, például sebességet mérni, figyelni a sávváltásokat, megállásokat, közlekedési szabálytalanságokat.

Irodalomjegyzék

[1] http://docs.opencv.org/java/org/opencv/video/BackgroundSubtractorMOG2.html (2014.10.21.)

[2] http://scikit-learn.org/stable/modules/mixture.html (2014.10.21.)

[3] http://en.wikipedia.org/wiki/Background_subtraction (2014.10.21.)

[4] http://hu.wikipedia.org/wiki/Klaszteranal%C3%ADzis (2014.10.21.)

[5] http://stackoverflow.com/questions/5738752/get-a-image-mask-of-the-differences-between-two-images-emgu-cv (2014.10.21.)

[6] http://mialmanach.mit.bme.hu/erdekessegek/skalainvarians_jellemzo_transzformacio_scale_invariant_feature_transform_-_sift (2014.10.21.)

[7] http://en.wikipedia.org/wiki/Scale-invariant_feature_transform (2014.10.21.)

[8] http://en.wikipedia.org/wiki/SURF (2014.10.21.)

[9] http://en.wikipedia.org/wiki/Haar_wavelet (2014.10.21.)