Recent Innovations in Mechatronics (RIiM) Vol. 5. (2018). No. SI.
DOI: 10.17667/riim.2018.si/11.

Jelzőtáblák mozgóképről történő felismerésére létrehozott szoftver OpenCV alapon

Balázs Viktor
Debreceni Egyetem,
Informatikai Kar
Debrecen,
Magyarország
roolue@gmail.com

Szilágyi László
Debreceni Egyetem,
Informatikai Kar
Debrecen,
Magyarország
Sziszi610@gmail.com

Erdei Erika
Debreceni Egyetem,
Műszaki Kar
Debrecen, Magyarország
erika2322@gmail.com

Erdei Timotei István
Mechatronikai Tanszék
Debreceni Egyetem,
Műszaki Kar
Debrecen, Magyarország
timoteierdei@eng.unideb.hu

***Absztrakt*—Manapság az autóbalesetek nagy százaléka azért történik meg, mivel az autóvezetők valamilyen külső/belső behatás következtében figyelmen kívül hagyják az út menti jelzőtáblákat. Az ilyen mulasztások miatt bekövetkezett károk sokszor komoly anyagi terhet jelentenek, rosszabb esetben, mint ismeretes az emberi élet nem pótolható. A projektben ezért törekedtünk arra, hogy az eddigi fejlesztések során megalkotott munkát abba az irányba fejlesszük tovább, ami segíthet ezt kiküszöbölni vagy iránymutatásként szolgálhat. A munka során megalkotott program lehetővé teszi, hogy egy tetszőlegesen megadott videó felvételen szereplő jelző táblát vagy valódi táblát felismerjen.**

Kulcsszavak—*OpenCV; C++; jelzőtáblák; Windows 10; Visual Studio 2015; mozgókép; Computer Vision, Industry 4.0*

I. BEVEZETŐ

A Debreceni Egyetem, Műszaki Karán kialakított, Épületmechatronikai Kutatóközpont biztosította az infrastruktúrát a képelemzési rendszerek megalkotásához, az elmúlt években [11]. A térfigyelő rendszerek kialakítása kulcsfontosságú vagyonvédelem szempontjából, bűncselekmények megelőzése érdekében [12].

A mai világban a képfeldolgozás nagy lépéseket tett fejlődés terén, ezért is választottuk ezt a projektet. A kezdeti kiindulási pont az alakzat detektáló szoftver volt, amely képekről háromszög, kör, illetve sokszög alakokat ismert fel és jelölt meg különböző színekkel, valamint előre gondolva a további fejlesztésekre megkülönböztette a piros háromszög és a piros kör alakokat is [13].

Ebben a projektben ez a szoftver tovább fejlesztésre, illetve olyan szinten átalakításra került, hogy képes legyen mozgóképről felismerni közúti jelzőtáblákat.

A program fejlesztése során motivációként szolgált, hogy a Google, a Tesla, az Uber és nagyobb autógyártó cégek önjáró autók folyamatos tervezését és tesztelését végzik, amelyek egy része már forgalomban is debütált. A Google önjáró autóflottája – ami Waymo néven fut – már több mint 3 millió mérföldet tett meg [1].

Az önjáró autók (autonomous car, self-driving car, robotic car) egy része az úgy nevezett SLAM (Simultaneous localization and mapping) algoritmust használja, ami szenzorok és térképek segítségével számolja ki az elkövetkezendő eseményeket. Az autók másik része szintén szenzorok segítségével a környezetet felmérve alakzat felismerést végez, hogy kiszűrje az autóra veszélyt jelentő elemeket és segítse a forgalomban való közlekedést [2].

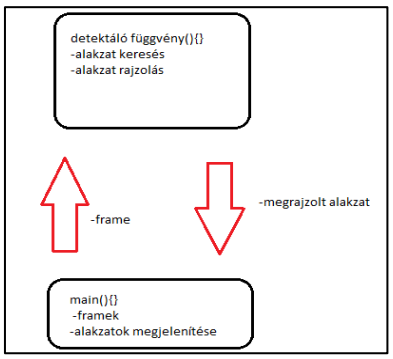
A szoftver Windows 10 operációs rendszeren futtatott Visual Studio 2015 fejlesztői környezetben C++ nyelven íródott, amibe az OpenCV került meghívásra a képfeldolgozás céljából.

II. TERVEZÉSI FÁZIS & MÜKÖDÉI ELV

A projekt kezdetén az eddigi fejlesztések alatt elkészült program egésze felhasználásra került volna, de a tervezgetések és az ötletek megbeszélése során, teljesen új feature-ek kerültek hozzá adásra és néhány átdolgozásra.

A jelenlegi képelemző rendszerek egy része gépi tanulást alkalmaznak, mások előre elkészített képi adatbázisból [14] keresik ki és jelölik meg a mozgóképen látott objektumokat [3]. A szoftverek kidolgozásakor sokszor figyelembe veszik az emberi idegrendszer és az agy feldolgozó képességét is [4].

Az elkészített program a beolvasott videófájlt framenként dolgozza fel, így olyan mintha gyors egymás után egy-egy fényképet elemezne. Ezeken a képeken párhuzamosan fut le a háromszög, kör illetve négyszög felismerő függvény.



1. ábra: Működési elv

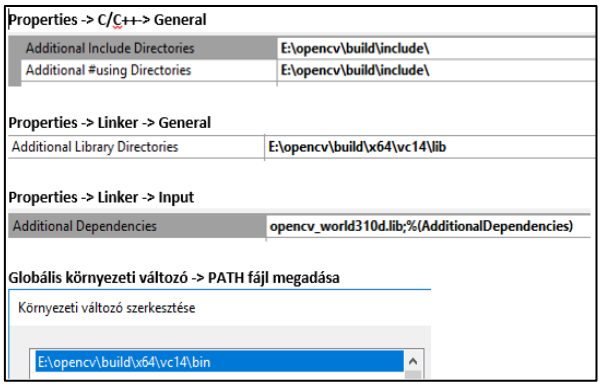
A sokszögeket felismerő függvények a kapott képen található kontúrokkal dolgoznak, amíg a kör felismerése Hough-transzformáció segítségével történik. Az alakzat felismeréshez előzőleg a kapott képeket bináris képekké kell alakítani, amelyekre így már ráengedhető az alakzat detektáló függvény rész.

A programba meghívott videofelvétel a teljesítmény miatt 640x360 felbontásra és 30képkocka/másodperces video formátumra lett konvertálva Freemake Video Converter használatával [5].

III. FELHASZNÁLT HARDWARE & SOFTWARE

A nagy számítási teljesítmény miatt, a projekt kidolgozása egy ASUS X550C típusú notebook-on lett végrehajtva. A számítógép Intel(R) Core(TM) i5-3337U @ 1.80GHZ CPU-val, 8GB DDR3 1600 MHZ memóriával és egy NVIDIA GeForce GT 720M videokártyával rendelkezik és maximum 1366 x 768 felbontást tud megvalósítani. A feladat elvégzése szempontjából kulcsfontosságú a VGA alapú képelemzés.

A számítógépen 64 bites Microsoft Windows 10 Home operációs rendszer futott. A program pedig Visual Studio 2015 fejlesztői környezetben C++ nyelven íródott az OpenCV képfeldolgozó könyvtár segítségével [6]. Különböző video konvertáló szoftverek is alkalmazásra kerültek, hogy a megfelelő video bemenetet elő tudjuk állítani. Az OpenCV programot a használatához minden újonnan kezdet project file-ba integrálni kell. Ez a project properties menüjében történik, ahol a C/C++ menüpontban meg adjuk az OpenCV fájlokat tartalmazó mappa elérési útvonalát, valamint a Linker menüpont alatt beállítjuk, hogy az elérési útvonalon melyik library fájl kerül felhasználásra.



2. ábra: OpenCV integrálása

Továbbá az operációs rendszerünk globális környezeti változói között a PATH menüben szintén csatolni kell az OpenCV mappa elérési útvonalát.

IV. FŐ PROGRAM MŰKÖDÉSE

A megfelelő header fájlok beolvasása után a fő program feladata, hogy az inputként kapott videófájlt le bontsa frame-kre a további feldolgozás céljából. A videó elemzés és képfeldolgozáshoz az OpenCV cv.hpp, core.hpp, highgui.hpp és video.hpp header fájlai kerültek megnyitásra.

A videófájl beolvasása egy cvCapture típusú változóba történik, ami az OpenCV egy, a mozgóképek kezelésére szolgáló adattípusa.

Az így deklarált változóba a cvCreateFileCapture() függvényben megadott elérési útvonal alapján történik a beolvasás. A framek a cvQueryFrame() metódus segítségével „vágódnak ki” a videófelvételről. Feldolgozásuk egy ciklusban történik, itt azonban fontos megemlíteni, hogy a projekthez választott gép ugyan relatív erős hardware-el rendelkezik, de még így sem alkalmas az összes frame kezelésére egyidejűleg, ezért csak minden 10-dik frame kerül elemzésre.

A videóból kivágott képek átméretezésre kerülnek, hogy a felesleges képterületek kiessenek a feldolgozás alól. Ez azért lehetséges, mivel a forgalomjelző táblák többnyire a jobb oldalon találhatóak, ezért elégséges ha a képünk jobb részét vetjük elemzés alá. Ezt a cvSetImageROI() függvénnyel hajtjuk végre(Region of Interest). A framek IplImage strukturában vannak tárolva, ami az Intel Image Processing Library formátumnak felel meg. Egy framet 3 ilyen strukturában tároljuk, hogy a párhuzamos feldolgozás során mind a 3 képre külön megtudjuk hívni az egyik elemző függvényt. A főprogramban a cikluson belül, minden egyes szálra meghívjuk az adott függvényt, valamint az azok által visszaadott képet egy előre felrajzolt ablakban megjelenítjük. A párhuzamosság az <omp.h> header fájl segítségével valósítható meg. A program csak akkor jeleníti meg az ablakot, ha a függvény talált alakzatot, így is javítva a teljesítményt. A megjelenítendő ablakok a cvShowWindow metódussal jelennek meg a lépernyőn, amíg a cvMoveWindow funkcióval a paraméterként megadott x, y koordinátákra irányíthatjuk azokat.

A futás végén ezek az ablakok, illetve a foglalt memória helyek egy gombnyomásra lebontódnak és felszabadulnak.

V. KÖR DETEKTÁLÁSA ÉS RAJZOLÁSA

A kör detektáláshoz a fő programban meghívásra kerül a Kor nevű függvény, aminek a függvény paramétere szintén egy IplImage. Azért, hogy ne az eredeti képen folyjon az átalakítás, másolat készül róla, amit ezek után a cvCvtColor() függvény szürkeárnyalatossá alakít a CV_BGR2GRAY paraméterével, ami azt jelenti, hogy BGR színekből szürkévé alakít, az első két paramétere pedig a bemeneti és a kimeneti változót takarja.

A cvSmooth() függvény segítségével simítás hajtódik végre a paraméterként megadott képen, majd ezek után a cvThreshold() úgynevezett bináris képpé formáz, ami azt jelenti, hogy egy olyan képet kapunk ahol az alakzatok egyértelműen elkülönülnek a háttértől. Ezek a műveletek azért hasznosak, mert az így kapott képen az elemző függvények hamarabb és nagyobb hatékonysággal találják meg az alakzatokat [7][8].

Az átalakítás után a cvHoughCircles() metódus van meghívva, ami Hough-transzformációt végez a képen.

A transzformáció után kapunk egy struktúrát, amelyben a talált körök középpontjai találhatóak. Ezek segítségével

egy ciklusban a cvCircle() függvény megrajzolja a kört a képen. Paramétereiben megadható, hogy milyen színnel és milyen vastagsággal rajzoljon. A szín pedig megadható az RGB skála használatával. A SetImageROI() függvénnyel beállítódik a visszaküldendő kép mérete, mélysége és csatornája. A méretezésre azért van szükség, hogy a megjelenítendő ablak körülbelül akkora méretben jelenjen meg, mint a körbe rajzolt forma. A kép egy másik néven lementve a Kor() függvény végén visszaküldésre kerül a fő programba, a kialakításra került képek, amelyeken dolgozott a metódus lebontásra kerülnek a cvReleaseImage() és cvReleaseMemStorage() segítségével. Ez szintén a memória felhasználás és a teljesítmény miatt fontos.

VI. NÉGYSZÖG DETEKTÁLÁSA ÉS RAJZOLÁSA

A fő programban meghívott Négyszög nevű függvény, szintén egy IplImage képet kap meg paraméterként. Ezen a képen is elvégződik a másolás, majd a másolaton tovább dolgozva a szürkeárnyalatos, a simított és a bináris kép kialakítása. A HoughCircles() helyett viszont itt a cvFindContours() metódus van meghívva, ami a bináris képen található kontúrvonalak végpontjait menti le egy struktúrába. Paramétereit tekintve meg kell adni a bemeneti képet, memória helyet, kimeneti struktúrát, a kontúrkeresés módját és a kezdőpont helyét [15]. Az így megtalált kontúrpontokon egy ciklusban végig haladva a cvApproxPoly() metódus meghívásával egy struktúrába lementődnek a kontúrok segítségével található sokszögszerű elemek. A cvApproxPolyban meghívásra kerül egy cvContourPerimeter() ami a kontúrokból alkotott kerületeket határozza meg. Az eredmény struktúrában található sokszögek közül csak azzal dolgozik tovább a függvény, amelyiknek maximális értéke 4, azaz négyszög. Ezeket a cvLine() meghívásával körbe rajzolja a paramétereiben megadott színnel és vastagsággal. A megrajzolás után léptetésre kerül a kontúr struktúra és a következő sokszög vizsgálatára kerül sor.

Ahhoz, hogy a létrehozott képet átméretezzük meg kell határozni a pontos koordinátákat, majd ezek segítségével a SetImageRoi() függvény meghívásával megtörténik az átméretezés.

```
int min_x, max_x, min_y, max_y = 0;
min_x = pt[0]->x;
min_y = pt[0]->y;
int j;
for (j = 1; j < 4; j++) {
    if (pt[j]->x < min_x)
        min_x = pt[j]->x;
    if (pt[j]->y < min_y)
        min_y = pt[j]->y;
}

max_x = pt[0]->x;
max_y = pt[0]->y;
for (j = 1; j < 4; j++) {
    if (pt[j]->x > max_x)
        max_x = pt[j]->x;
    if (pt[j]->y > max_y)
        max_y = pt[j]->y;
}


if (((max_x - min_x) > 50) && ((max_y - min_y) > 50) && ((640 - max_x) > 50)) {
    cvSetImageROI(temp2, cvRect(min_x, min_y,
        (max_x - min_x), (max_y - min_y)));
    uj = cvCreateImage(cvSize(max_x - min_x, max_y - min_y),
        temp2->depth, temp2->nChannels);

    cvCopy(temp2, uj);
```

3. ábra: Átméretezés

A koordináták kiszámolása a cvRect() függvény meghívásához fontosak mivel ebben a függvényben egy téglalap bal felső sarkának x és y koordinátája, valamint a szélessége és a hosszúsága kerül megadásra. Ez a téglalap pedig a SetImageRoi() metódus egyik paramétere lesz.

Ezek után az átalakított kép szintén lementődik és a függvény végén visszaküldésre kerül a fő programba. A fennmaradt másolatok és memória helyek itt is lebontódnak és felszabadulnak.

VII. HÁROMSZÖG DETEKTÁLÁSA ÉS RAJZOLÁSA

A Háromszög() függvény meghívásakor megkapja az IplImage képet. Megtörténik a másolatok létrehozása, amelyeken újra végrehajtódnak a CvtColor(), Smooth() és Threshold() metódusok. De ez esetben egy cvDilate() is alkalmazásra kerül, ami egy kisebb nyújtást hajt végre a képen.

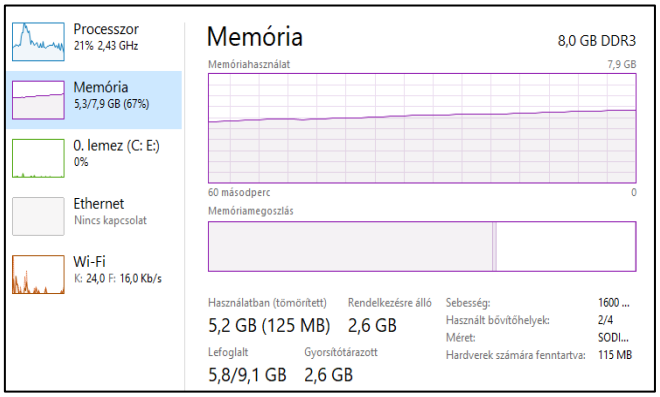
Mivel a háromszög is sokszög ezért a négyszöghöz hasonlóan itt is a FindContours() függvény kerül meghívásra. Majd az így kapott kontúrokon végre hajtódik az ApproxPoly() és letárolódik az eredmény struktúrában. Az eredményeknél most a függvény azokat rajzolja ki, amelyek 3 kontúrvonalból állnak, tehát 3 szög.

A Line() metódussal megrajzolásra kerül a háromszög, majd méreteződik és mentésre kerül. Ez az új kép végül visszaküldésre kerül a fő programba. A fennmaradt memória helyek felszabadulnak, a képek pedig lebontódnak.

VIII. TELJESÍTMÉNY ÉS MŰKÖDÉS

A program egy 5 perces videofelvétel használatával lett tesztelve, amely során kisebb nagyobb hibával ismerte fel a különböző közlekedési táblákat. Megfigyelhető, hogy a video kis felbontása ellenére, és hogy folyamatosan bontódnak le a képek az 5 perc végére a program így is több, mint 2GB memóriát használ fel. Emellett a processzor kihasználtsága nem emelkedett 25% (2,50 GHz) felé. Ez a

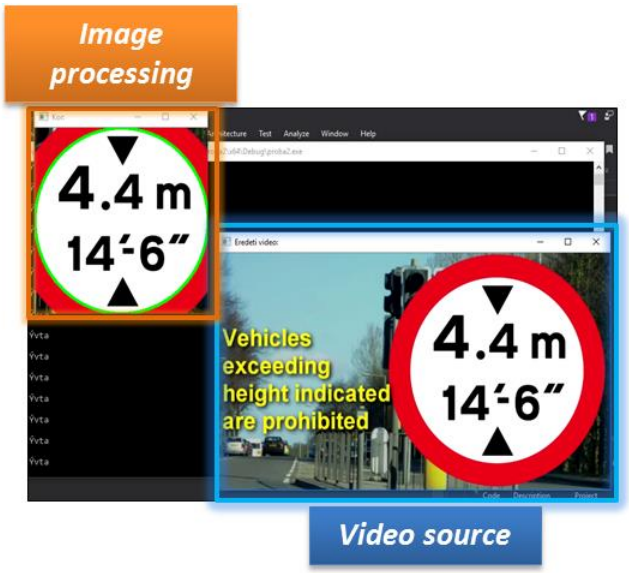
memória igény a video hosszával és felbontásával tovább nőhet.



4. ábra: Teljesítmény a futás végén

A program a futás során a tesztfelvételről a közlekedési jelzőtáblák körülbelül 70%-os (69.1%) pontossági rátával ismerte fel és rajzolta körbe. Ez statisztikailag jó arány, amit a további algoritmusok bevonásával és nagyobb felbontású video anyagok felhasználásával tovább növelhető. Viszont ronthat a feldolgozási arányon, ha nagy sebesség mellett felvett videofelvételeket elemzünk le vele, mivel így nagy arányban közbe szólhat a kép elmosódása, valamint az autóban és a környezetben megjelenő csillogások is.

Az elemzés során külön ablakban megjelenik a video, amelyen dolgozik a program és mellette megjelenítésre kerülnek külön-külön a kör, a háromszög és a négyzet alakú táblákat megjelenítő ablakok. Ezek mind az elemző függvényekből visszakapott képek. A Kor nevezetűben a kör, a Haromszog-ben a háromszög, a Negyszog-ben pedig a négyszög alakú jelzőtáblák jelennek meg.



5. ábra: Test I.

Az alábbi képeken látható, hogy a program futásakor a megjelenő ablakok a cvMoveWindow() metódusban paraméterként beállított x és y koordinátákon jelennek meg, valamint megtörténik a körberajzolás is.

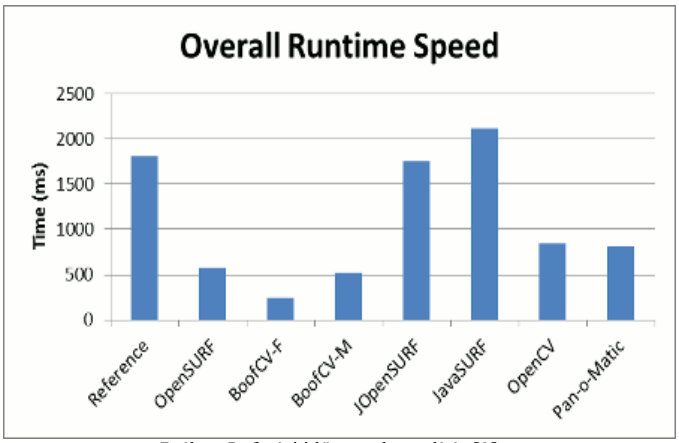
6. ábra: Test II.

IX. TOVÁBBI FEJLESZTÉSI LEHETŐSÉGEK

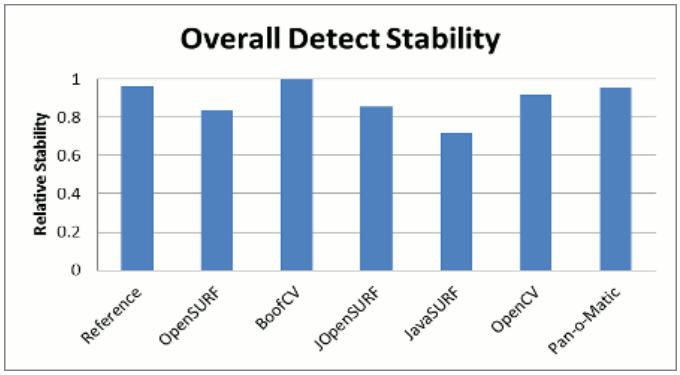
A tervezés során felmerült, hogy a jelenlegi generációs hardware eszközök már rendelkeznek komoly számítási kapacitással, így egy Rapsberry PI 3 –as is alkalmas lehet paraméterei révén „redukált” képelemzési feladat ellátására.

Az viszont kijelenthető, hogy az előző fejezetben említett teljesítmény adatok különböző operációs rendszereken és más programnyelven implementálva más eredményeket mutathatnak. Az OpenCV elérhető Java, Python, C++, és C programozási nyelvekre, továbbá támogatja a Windows, Linux, Mac OS, iOS és Android operációs rendszereket.

Szóba jöhet a későbbiekben egy olyan program verzió, amelyben kikerül az OpenCV és egy kicsit kevesebb memória igényű eljárás segítségével valósul meg a projekt. Az alábbi diagramokon látható, hogy létezik lefutási időben gyorsabb és stabilabb C++ alapú képfeldolgozó szoftverek.



7. ábra: Lefutási idő összehasonlítás [9]



8. ábra: Detektálás stabilitása [10]

Megállapítható, hogy lefutása időben az 5. legjobb az OpenCV alkalmazás és detektálás stabilitásban pedig a 4. helyre sorolható. A diagramokat nézve a további fejlesztés BoofCV-vel lenne a megfelelő.

Ha sikerülne, olyan teljesítmény optimalizációt elérni, amit egy Androidos vagy iOS-es telefon képes fenntartani, akkor továbbiakban telefonra történő portolás is szóba jöhet, így lehetővé téve a hordozhatóságot.

További fejlesztési lehetőség lehetne, ha a szoftver, olyan szintet érne el, amellyel már felhasználhatóvá válna egy fejlesztésben lévő önjáró autó részegységeként is.

X. ÖSSZEGZÉS

A projekt befejeztével a közúti jelzőtábla felismerő szoftver elkészült. A program az elérési úton megadott videót elemzi, adott pontossági rátával felismeri, és kis ablakokban megjeleníti a mozgóképen látható jelzőtáblákat a felhasználónak. A program futása befejeztével gombnyomásra lebontja a képernyőn megjelenített ablakokat és bezárul.

Továbbiakban, ha lehetőség nyílik rá, akkor a projekt nagyobb teljesítményű munkakörnyezetre átültetve, magasabb képminőségű videókat feldolgozva folytatódna és esetleg egy kisebb teljesítményigényű működés kerülne kidolgozásra.

KÖSZÖNETNYILVÁNÍTÁS

A publikáció elkészítését az EFOP-3.6.1-16-2016-00022 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

HIVATKOZÁSOK

[1] Waymo, (2017, May 18). [Online]. Available: https://waymo.com/ontheroad/

[2] Kichun Jo, Junsoo Kim, Dongchul Kim, Myoungho Sunwoo, „Development of Autonomous Car—Part I: Distributed System Architecture and Development Process”, IEEE Transactions on Industrial Electronics 61(12):7131-7140, December 2014

[3] Matthew Russel, Scott Fischaber, „OpenCV based road sign recognition on Zynq”, Industrial Informatics (INDIN), 2013 11th IEEE International Conference,2013

[4] C. Y. Fang, C. S. Fuh, P. S. Yen, S. Cherng, and S. W. Chen, “An Automatic Road Sign Recognition System based on a Computational Model of Human Recognition Processing”, Computer Vision and Image Understanding, Vol. 96 , Issue 2 (November 2004), pp. 237 – 268.

[5] Freemake Video Converter, (2017, May 14). [Online]. Available: http://www.freemake.com/hu/downloads/

[6] OpenCV, (2017, May 14). [Online]. Available: http://opencv.org/releases.html

[7] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, T. Koehler, “A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information”, Proceedings of the 2005 IEEE Intelligent Vehicles Symposium, Las Vegas, USA., June 6 - 8, 2005

[8] H. X. Liu, and B. Ran, “Vision-Based Stop Sign Detection and Recognition System for Intelligent Vehicle”, Transportation Research Board (TRB) Annual Meeting 2001, Washington, D.C., USA, January 7-11, 2001.

[9] Rapsberry PI 3 specifications, (2017, May 14). [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[10] BoofCV,(2017, May 17).[Online]. Available: http://boofcv.org/notwiki/images/benchmark_surf/overall_all_speed.gif

[11] G. Husi, T. I. Erdei, Zs. Molnár, „A Novel Design of an Augmented Reality Based Navigation System & its Industrial Applications,” 15th IMEKO TC10 – Technical Diagnostics in Cyber-Physical Era Budapest, Hungary, 6 – 7 June, 2017 - Organised by: MTA SZTAKI – Hungarian Academy of Sciences - Institute for Computer Science and Control

[12] G. Husi, „Minőségmenedzsment-rendszerek módszereinek alkalmazása a Magyar Köztársaság Rendőrségénél,” Megjelenés/Fokozatszerzés éve: 2006. – 120p.

[13] A. Husam, A. S. Adila, Zs. Molnár, T. I. Erdei, G. Husi, „Reviewing the notable progress of effective techniques in the development of stroke hand rehabilitation,” FMTÜ - XXII. International Scientific Conference of Young Engineers - Kolozsvár, 23-24.03.2017.

[14] N. C. Obinna, T. I. Erdei, Zs. Molnár, G. Husi, „LabVIEW Motion Planning and Tracking of an Industrial Robotic Manipulator (KUKA KR5 arc): Design, Modelling, and Simulating the Robot’s Controller Unit,” FMTÜ - XXII. International Scientific Conference of Young Engineers - Kolozsvár, 23-24.03.2017.

[15] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „AGV cyber physical navigation system,” FMTÜ - XXII. International Scientific Conference of Young Engineers - Kolozsvár, 23-24.03.2017.