

TDK-dolgozat

Kalázdai Bálint
Szabó Balázs

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertchnológia Intézet



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

VISUAL DRIVING ASSISTANT
JÁRMŰVEZETÉST TÁMOGATÓ KÉPFELDOLGOZÓRENDSZER

Szerzők: **Kalázdi Bálint**
 mérnök informatikus szak, IV. évf.
Szabó Balázs
 mérnök informatikus szak, IV. évf.

Konzulens: **Dr. Vámosy Zoltán**
 egyetemi docens

Tartalomjegyzék

Tartalomjegyzék	2
1. A projekt célja.....	4
1.1 A feladat leírása, célkitűzései	4
1.1.1 Tervezett funkciók	5
1.1.2 Alkalmazási lehetőségek.....	5
1.2 Alkalmazási és fejlesztési környezet	6
1.2.1 Hardverkörnyezet.....	6
1.2.2 Szoftverkörnyezet	6
1.2.3 Fejlesztési eszközök.....	7
1.3 A rendszer korlátai	7
1.3.1 A hardveres környezetből adódó korlátok	7
1.3.2 A szoftveres környezetből adódó korlátok	9
2. Hasonló rendszerek elemzése	10
2.1 Automatizált rendszerek a közlekedésben	10
2.2 Képfeldolgozáson alapuló támogató rendszerek	11
2.2.1 Prometheus TSR	11
2.2.2 Real-time Traffic Sign Detection.....	12
2.2.3 TSR for Intelligent Vehicle/Driver Assistance System	13
2.2.4 Automatic TSR in Digital Images	14
2.2.5 Road Sign Detection and Recognition.....	14
2.2.6 A megvizsgált rendszerek összehasonlítása	15
2.3 Összefoglalás a projekt szemszögéből.....	16
3. A rendszer tervezése	17
3.1 A projekt céljából adódó sajátosságok.....	17
3.1.1 Az evolúciós fejlesztési modell	17
3.1.2 A tesztelés és hangolás kiemelkedő szerepe.....	19
3.1.3 Az átmeneti verziók tesztelési szempontjai	19
3.1.4 Az evolúciós fejlesztésből adódó hátrányok kiküszöbölése	19
3.2 Előzetes rendszerterv	20
3.2.1 A képelemző és objektumfelismerő alrendszer felépítése	21
3.2.2 A teljes szoftver-rendszer felépítése	22
3.3 Az átmeneti verziók során alkalmazott változtatások.....	23
3.3.1 A videó fájlok rögzítése	23
3.3.2 A célkitűzések áttekintése a felvételek tapasztalatai által	24
3.3.3 Az azonosítási folyamatterv problémája.....	24
3.4 A végleges verzió rendszerterve	25
3.4.1 Objektumcsoportok.....	25
3.4.2 Az előre definiált érdekelt régiók	26
3.4.3 A felismerés mechanizmusának felvázolása.....	27
3.4.4 A képelemző és objektumfelismerő alrendszer végleges modellje	28
3.4.5 A teljes rendszer végleges modellje.....	28

4. A rendszer megvalósítása	29
4.1 A mintavételező modul	29
4.1.1 A modul megvalósítása	29
4.2 Az előfeldolgozó modul	30
4.2.1 A modul megvalósítása	30
4.2.2 Optimalizálás	31
4.2.3 Elvetett ötletek	31
4.3 A felismerő modul	33
4.3.1 A régió bejárása	34
4.3.2 A kiértékelő egységek működése	36
4.3.3 Az eredmények értékelése	38
4.4 A felhasználói felület	39
5. Tesztelés	40
5.1 A tesztelések szerepe	40
5.1.1 A folyamatos tesztelés igénye	40
5.1.2 A részeredmények felhasználása	40
5.2 A tesztkörnyezet	41
5.2.1 A bemeneti egység	41
5.2.2 Az előfeldolgozó egység	42
5.2.3 A felismerő egység	42
5.3 A tesztelés folyamata	43
5.4 Teszteredmények	43
5.4.1 Délben készült felvételek eredményei	43
5.4.2 Kora délután készült felvételek eredményei	45
5.4.3 Délután készült felvételek eredményei	46
5.4.4 Késő délután készült felvételek eredményei	48
5.4.5 Éjszaka készült felvételek eredményei	49
5.5 Az eredmények értékelése	51
5.6 Futási sebesség különböző konfigurációkon	52
6. Összefoglalás	53
6.1 Kitűzött és elért célok	53
6.1.1 Fejlesztési tapasztalatok	53
6.1.2 A korlátok behatárolása	55
6.2 Összehasonlítás hasonló rendszerekkel	56
6.2.1 Néhány rendszer eredményeinek vizsgálata	56
6.2.2 A rendszer előnyei és hátrányai	58
6.3 Továbbfejlesztési lehetőségek	58
6.4 Konklúzió	59
Irodalomjegyzék	60

1. A projekt célja

1.1 A feladat leírása, célkitűzései

A közlekedésbiztonság napjainkban kulcskérdés, a benne résztvevők száma folyamatosan növekszik. Ebből következően elengedhetetlen, hogy a gépjármű vezetőjét a lehető legtöbb felhasználható eszközzel támogassuk a hatékonyság és a biztonság fokozása érdekében. Ezen eszközök közé sorolhatóak a hagyományos útjelző tábláktól kezdve, a jelzőlámpákon át, egészen a járművekbe építhető komplex elektronikai rendszerekig bezárólag minden, ami hasznos információkat nyújthat számára.

A járművezető elsődleges információforrása a látása. Sajnos azonban az emberi szem látószöge korlátozott, amelyet még tovább ront a járműben elfoglalt pozíciójából adódó holtterek létrejötte, valamint a sebességből adódó látótér-kiesés. A kérdést még tovább bonyolítja az, hogy az emberi agy valós idejű képfeldolgozó képessége bár kiváló ugyan, de még egy átlagos sebességgel mozgó jármű vezetése közben is túlságosan nagy a gyorsan változó látómező azonosítandó objektumainak száma, és könnyen előfordulhat, hogy a vezető figyelmét elkerüli egy-egy fontosabb objektum. A gépjárművek számának növekedése és az egyre komplikáltabbá váló közlekedési hálózatok mellett ennek veszélye még nagyobb, ráadásul az olyan, nehezen mérhető emberi tényezők, mint a fáradtság vagy a koncentrációhiány is komoly kockázati tényezőként jelennek meg.

Egy olyan eszköz, amely a bejövő képi információt hallható jelekké alakítja át, hasznos támogatást jelenthet a gépjármű vezetőjének, segítheti egy-egy objektum „tudatosulását”. Természetesen gondolni kell arra is, hogy az eszköz – azaz esetünkben a szoftver – könnyen testre szabható, többfokozatú támogatásra legyen képes, és az esetlegesen felesleges, „túlzásba vitt” hangjelzések ne zavarják a vezetőt.

A Visual Driving Assistant projekt feladata egy ennek megfelelő szoftverrendszer megalkotása, azaz amely képes a járművezetőt hangjelzésekkel, valós időben, a közlekedéssel kapcsolatos legfontosabb eseményekről a látható objektumok alapján tájékoztatni. Mivel minden lehetséges közúti jelzés felismerése, és a pillanatnyi helyzet kiértékelése hatalmas feladat – melynek megvalósításához számtalan szakember és nagyon hosszú fejlesztési idő lenne szükséges –, ezért a projektben ezek közül azokra koncentrálunk, melyek elmulasztása a többinél sokkal komolyabb veszélyforrást hordoznak magukban. Ebbe a csoportba tartoznak a közúti és vasúti keresztezések tilos jelzései, valamint a kötelező elsőbbségadásra figyelmeztető táblák, így ezek felismerése tekinthető a rendszer legfontosabb funkciójának.

Fontos szempont továbbá – mint már említettük – az, hogy a programnak semmiképpen sem szabad „túlzásba esnie”, mert az már segítség helyett inkább zavaró lehet. Ez elsősorban azt jelenti, hogy a hibás, illetve szükségtelen jelzések száma minimális legyen. Gépi látásról lévén szó, hibás jelzések előfordulhatnak, például reklámfeliratok, féklámpák esetén, vagy bármely, az alkalmazott algoritmus „gyenge pontjainak” tekinthető objektumok megjelenésénél. Ilyen esetekben tehát szükséges, hogy erősebb legyen az ellenőrzés. A szükségtelen esetekre tipikus példa az, amikor egy kötelező elsőbbségadásra figyelmeztető tábla csak akkor van érvényben, amikor a jelzőlámpa nem üzemel.

Végül, de nem utolsó sorban célunk az is, hogy a rendszer mindezt a legegyszerűbb hardveres környezetben tudja végrehajtani, azaz széles körben elterjedt, nem specifikus eszközök felhasználásával, semmiképpen se legyen szükség különleges, drága alkatrészekre, célhardverekre, vagy bármilyen átalakításra a gépjárműben. Ez természetesen komoly korlátozásokat is rejt magában, ahogy azt később részletesen be is mutatjuk.

1.1.1 Tervezett funkciók

A program legfontosabb funkciója tehát a bejövő kép feldolgozása, és azon belül az alábbi objektumok felismerése, hogy azokról tájékoztatást adhasson:

- Az „elsőbbségadás kötelező”, és a „stop, elsőbbségadás kötelező” közúti jelzőtáblák felismerése, mivel ezek kiemelt fontosságúak, elmulasztásuknak súlyos következményei lehetnek. Kiemelkedő szerepüknek köszönhető, hogy ezen tábláknak az egész világon többé-kevésbé megegyező, szabványos, egyedi alakjuk és kinézetük van (1.1 ábra)¹.
- Közúti jelzőlámpák és vasúti átkelők tilos jelzéseinek felismerése, figyelembe véve az esetleges bonyolultabb kereszteződések esetén szintén látszódó, de egyértelműen nem a járművezetőre vonatkozó jelzések szűrését.
- Zöld lámpák felismerése abban az esetben, amikor a felettük elhelyezkedő táblák csupán a lámpa működésképtelensége esetén szükségesek.



1.1 ábra¹
Stop-táblák a világ különböző részeiről

Felismerés esetén a szoftver hangjelzésekkel figyelmeztesse a járművezetőt. Fontos, hogy ezek rövidek, de egyértelműek legyenek, semmi esetre sem igényelhetnek fokozott figyelmet, vagy rápillantást a kijelzőre.

1.1.2 Alkalmazási lehetőségek

Említésre került az, hogy fontos a teljes rendszer egyszerűsége, így a feladatot úgy kell megoldani, hogy lehetőleg minél szélesebb körben elterjedt eszközöket és platformot igényeljen a működéshez.

Ezen feltételeknek tökéletesen megfelel egy hordozható számítógép, és az ahhoz manapság USB csatlakozón keresztül köthető egyszerű külső webkamera (a szabad pozicionálás szükségessége végett nem alkalmas az esetlegesen előforduló beépített kamera erre a célra). Amennyiben a szoftver ennek megfelelően van megtervezve és elkészítve, akkor egy megfelelően pozicionált webkamera esetén bármilyen közúti járművön alkalmazható a rendszer.

¹ Általunk készített ábra a témában az Interneten fellelhető információk alapján.

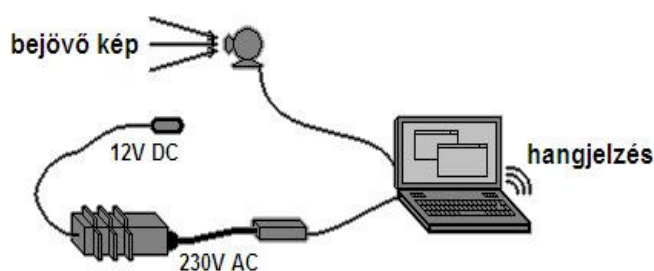
1.2 Alkalmazási és fejlesztési környezet

1.2.1 Hardverkörnyezet

Maga a szoftver a célplatformot futtatni képes hardveres környezetet igényel. A már felvázolt esetben ez egy hordozható számítógépből, annak tápellátását biztosító adapterekből és egy webkamerából áll (1.2. ábra). Ezen megoldás további előnye az egyszerűség mellett, hogy alkalmazható az egyébként más feladatok ellátására vásárolt hordozható számítógép is.

A webkamera megfelelő pozicionálása nélkülözhetetlen, de semmiképpen sem igényel bonyolult felszereltséget. A bejövő élőkép felbontásaként a 640×480-as standard VGA-t alkalmazzuk, mivel ennél nagyobb felbontásra még nem mindegyik mai eszköz képes.

A hordozható számítógép tápellátását biztosítani tudja a gépjármű elektromos rendszere, mivel a szoftver alkalmazása közben feltehetőleg a jármű motorja mozgásban van, így a generátor elegendő energiát termel a megnövekedett energiafelhasználás biztosítására. (általában kb. 40–60 W).



1.2. ábra

A hardverkörnyezet sematikus ábrázolása

Ez a legegyszerűbben egy mostanában népszerű, szivargyújtóra köthető feszültség-átalakítóval oldható meg, melyben egy oszcillátor és egy transzformátor állítja elő a 230V-os feszültséget alacsony energiavesztéssel. Ennek további előnye, hogy a hordozható számítógép saját hálózati adaptere alkalmazható, nincs szükség külön átalakításra, ráadásul a legtöbb típus esetében az külön védelmet is nyújt az esetleges feszültségingadozások esetén.

1.2.2 Szoftverkörnyezet

A szoftver a Microsoft .NET 3.5-ös verziószámú keretrendszerére és egy, az azt futtatni képes operációs rendszerre épül. A fejlesztés és tesztelés a Microsoft Windows XP x86-os architektúrára épülő rendszerén történik. A hordozható számítógépek esetében ez egy elterjedt platform, és az x86-64 architektúrát kihasználó újabb operációs rendszerek (Windows Vista, Windows 7) is problémamentesen futtatják.

Mivel a szoftver működéséhez a .NET keretrendszer szükséges, így a mono¹ nyílt forráskódú projekt révén, amely implementált egy binárisan kompatibilis CLR futtató környezetet GNU/Linux disztribúciókra és Apple Mac OSX rendszerekre is, könnyen átdolgozható a program. Csupán a grafikai felületet leíró programrészt kell újra implementálni az adott operációs rendszerben alkalmazott ablakkezelőnek megfelelően (pl. Gnome² esetén a mono GTK# felhasználói felület eszköztárát felhasználva). Így a szoftver képfeldolgozó, elemző, és kiértékelő része változtatás nélkül alkalmazható más platformokon is.

¹ Mono, www.mono-project.com

² GNU Network Object Model Environment, www.gnome.org

1.2.3 Fejlesztési eszközök

Az implementáláshoz felhasznált programozási nyelv a C#, az integrált fejlesztői környezet pedig a Visual Studio 2008-as verziója.

A webkamera csatolásához, és a tesztelési, prezentációs videó fájlok kezeléséhez az Intel OpenCV¹ függvénykönyvtárát használjuk. Mivel ez a nyílt forráskódú függvénykönyvtár C (később C++) interfészen keresztül érhető el, ahhoz, hogy C# nyelven is meghívhassuk a használni kívánt függvényeket és a definiált típusait felhasználhassuk, szükség van egy ún. wrapperre. Ebből csakúgy, mint a többi népszerű programozási nyelvhez, a C#-hoz is többféleképpen megvalósított, de lényegében nagyon hasonló verzió is létezik. Választásunk végül az EmguCV² x86-os verziójára esett, mivel a fejlesztési fázis kezdetén ezt találtuk a leggyakrabban frissített ilyen eszköznek, amely az előzetes tesztelések alapján tökéletesen megfelelt a feladatnak. Az említett videó fájlokat DivX kodek segítségével tömörítjük.

Mivel az általunk alkalmazott megoldási algoritmus egyes részeinek – amelyekben a paraméterek száma és értéke komolyan befolyásolja az eredményt – nincs pontos megfelelője az OpenCV-ben, így a függvénykönyvtár a képfeldolgozási folyamat során nem használjuk. Felmerült egy-két OpenCV által hatékonyan implementált megoldás alkalmazása (pl. életlenítés), de ez olyan problémákat okozott, melyeket a későbbiekben részletesen is tárgyalunk, így maradtunk azon eredeti elképzelésünkönél, hogy az algoritmusokat egészében magunk implementáljuk.

A szoftver fejlesztése során komoly szerepet kap a folyamatos tesztelés, hangolás, és a futási sebesség, így mindenféleképpen több különböző hardveren, gyengébb számítási teljesítményű rendszereken is szükséges a tesztelés, az optimalizálási megoldásokat is az ezeken elért eredményekhez kell igazítani.

1.3 A rendszer korlátai

1.3.1 A hardveres környezetből adódó korlátok

A gépi látásban alapvető probléma, hogy amit az emberi szem és agy könnyedén, gyorsan felismer és feldolgoz akár a legkülönbözőbb körülmények között is, az egy olyan programozott digitális rendszernek, mint napjaink elektronikus eszközei, óriási nehézségeket okoz, ahogy ezt részletesen [Freeman] tanulmányában is olvashatjuk.

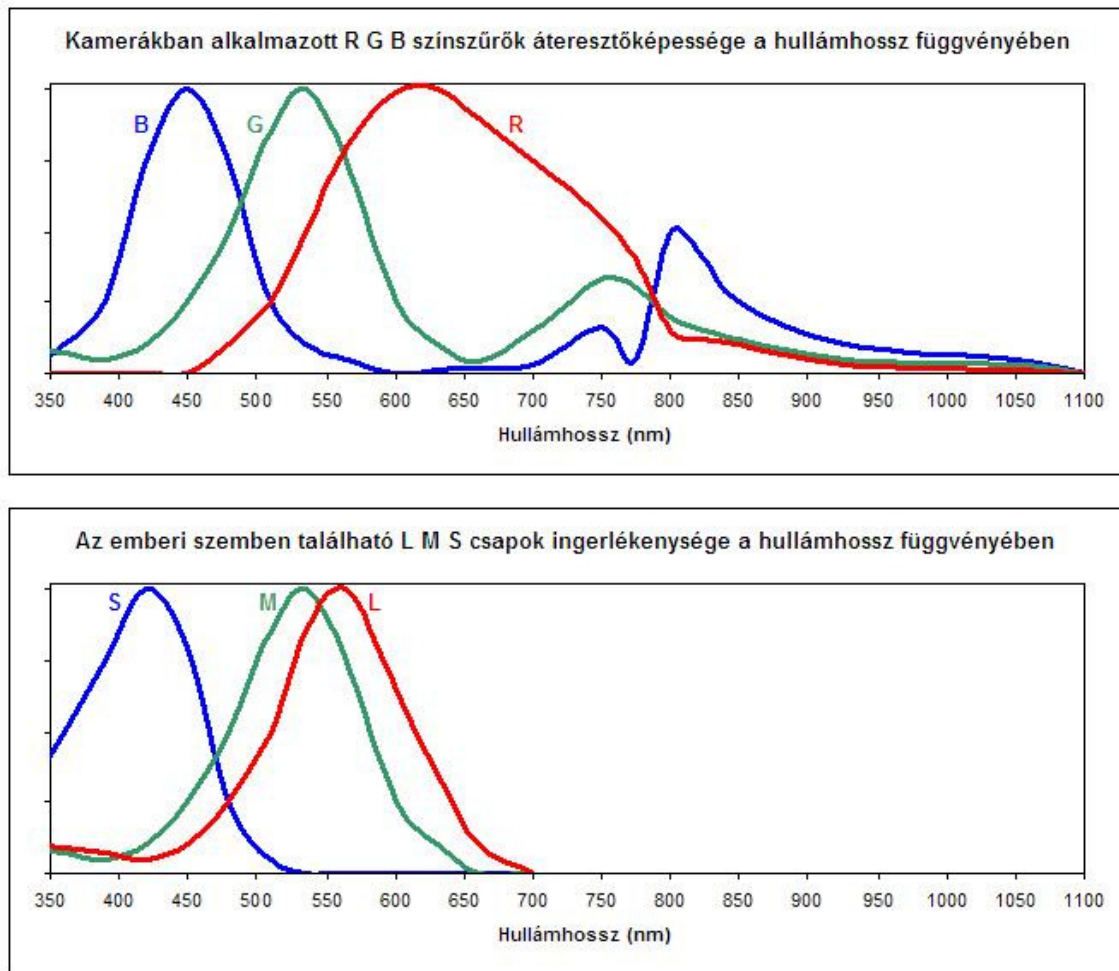
Az a kikötés, mely szerint egyszerű és olcsó képrögzítő eszközt használunk – azaz egy webkamerát – még nehezebbé teszi ezt a feladatot. Ezen eszközök tökéletesen alkalmasak arra a célra, hogy digitális élőképet biztosítsanak a távolsági kommunikáció számára, de képfeldolgozási feladatokhoz szűkösen bizonyulnak elegendőnek. Ennek legfőbb oka, az ezen eszközök által biztosított felbontás. Ma már szinte minden webkamera képes a VGA felbontásra, de e fölé még csak a drágább kategóriájúak tudnak menni. Alacsony felbontás mellett azonban sokkal nehezebb a képen a számunkra fontos, jellegzetes objektumokat megtalálni.

A projekt szempontjából a másik nehézség, amely már nem csak a webkamerákra jellemző, a színek reprezentációjából fakad. Mivel a szoftverünk működésében kiemelt szerepet kap a képpontok színösszetevőinek elemzése, ezért ez mindenképpen kulcsfontosságú kérdés. Napjaink elektronikus képrögzítő eszközei a spektroszkópok

¹ Open Computer Vision Library, sourceforge.net/projects/opencvlibrary

² Emgu CV, www.emgu.com

kivételével három színszűrő segítségével hozzák létre a három intenzitásképet, amelyek a színes képet reprezentálják. A három alapszín – piros zöld és kék – az emberi látáshoz igazodott, ugyanis ezen három hullámhossz-tartományba eső fényből álló nyaláb, egyenként megfelelő intenzitású összetevővel ugyanolyan észlelést kelt a szemben a háromféle tartományra legérzékenyebb idegsejt-csoportok, a csapok számára, mint egy ezektől eltérő hullámhosszú monokromatikus nyaláb. Emiatt a legtöbb szín (bár nem mindegyik, de ez elhanyagolható), amelyet érzékelünk, kikeverhető ebből a három összetevőből, így a kamerák esetén is ehhez igazodva használnak RGB színszűrőket (1.3. ábra)¹.



1.3. ábra¹

A színszűrők és a színérzékeny idegsejtek összehasonlítása
(egyértelműen kitűnik a kettő kapcsolata a látható tartományban)

Ezzel az első probléma azonban az, hogy az RGB színábrázolásban hatalmas különbségek vannak azon pontok intenzitásértékei között, amelyeket azonos színűnek látunk, csak éppen különböző telítettségben és fényerősséggel. Erre a probléma megoldás lehet a HSV/HSL színábrázolás, de mivel a hardver, a képalkotó eszköz nem ilyen formában továbbítja az adatokat, azok átalakítása nagyon időigényes, ahogy azt később meg is említjük a működés bemutatásánál.

¹ Kwak Attack fórum, *aubilenon*, *Invisible light*
kwakattack.polpo.org/files/sensor_mosaic_0.png illetve
kwakattack.polpo.org/files/human_eye.png alapján

De a reprezentáció feldolgozásán kívül sokkal komolyabb gond az, hogy a fizikai világban előforduló, és az ember által ugyanolyan színűnek látott képpontok esetén a szem által megkülönböztethető fényerőségek száma sokkal több, mint amit egy 3×8 bites RGB, vagy akár HSV kódolás ábrázolni tud. Ez korlátot jelent már a bejövő jel szintjén is, ugyanis a képrögzítő eszköz ezt a problémát különböző fényviszonyok között automatikus fehéregyensúly és kontraszt beállításokkal próbálja korrigálni. Ennek eredményeképpen éjszakai megvilágításban, ahol az érzékenység maximális, egy erős, bármilyen színű fényforrás megközelítőleg tiszta fehérnek látszik. (1.4. ábra).



1.4. ábra
Piros lámpa éjszaka, és nappali fényviszonyok között

Amint a szoftver működésénél látni fogjuk, ennél a feladatnál a piros lámpákhoz nappal és éjszaka más megoldást fogunk alkalmazni. Mivel a program alapvetően a színek alapján történő küszöböléssel előfeldolgozott képen fogja az objektumokat keresni, így azt használjuk ki éjjel, hogy a piros lámpák körül egyfajta „aura” jelenik meg, amely színében elűt az egyéb fényforrásoknál tapasztalhatóakétól.

1.3.2 A szoftveres környezetből adódó korlátok

A képfeldolgozó rendszerek esetében a szükséges számítási teljesítmény egyéb, általános célú alkalmazásokhoz képest magasnak mondható. Esetünkben a sebesség még fontosabb kérdés, mivel élőképet kell elemeznünk, és a lehető legjobban megközelíteni az optimális, folyamatos kimenetet. Az egész szoftver hasznosságát veszélyeztetné, hogyha nem lenne képes valós időben jelezni a járművezetőnek.

A teljes rendszer feldolgozási sebessége alapvetően két összetevőtől függ. Először is a rendelkezésre álló hardverek által nyújtott számítási teljesítménytől, amelynél a tervezés és tesztelések során az átlaggal kell terveznünk. Nem szabad feltételezni, hogy a célhardver nagyon gyors, mert akkor az átlagos eszközökön a szoftver használhatatlanná válna. Másik oldalról, hogyha túlzottan elavult, lassú számítógépre próbálnánk meg a programot megírni, az annyira korlátozná az algoritmusok közötti választási lehetőséget és a lehetséges komplexitást, hogy az alkalmazás megbízhatósága erősen kétségessé válna.

A másik összetevő a program relatív sebessége. Ez az alkalmazott absztrakciós szinttől függően széles skálán mozoghat, egészen a hardver közeli szinten megírt kódtól kezdve – mely nyilvánvalóan a leggyorsabb, de mindamellett a legnehezebben megírható és a legerősebben architektúra- és platformfüggő –, egészen a magas, teljesen objektum orientált, köztes nyelvre forduló programozási nyelvekig. Mivel ez utóbbit választottuk, a futási sebesség erősen korlátozott, jelentős szerepet kap az optimalizálás.

2. Hasonló rendszerek elemzése

2.1 Automatizált rendszerek a közlekedésben

Az automatizált közlekedés-támogató rendszerek a technikai fejlődés eredményeképpen egyre nagyobb szerepet kapnak a különféle közlekedési ágazatokban, legyen az vasúti forgalomirányítás, vagy közúti járművekbe épített akadály-felismerés. Az igény efféle automatizált megoldásokra folyamatosan növekszik, ahogy fejlődésük következtében újabb és újabb területeken bizonyulnak hatékonynak.

Ezen rendszerek alapvetően két, élesen elkülöníthető csoportot alkotnak. A kérdéskör egyik oldalán a közlekedés irányításának fejlesztése, és ennek a lehető legmagasabb szintű automatizálása áll. Ebbe beletartozik a forgalom szabályozásának a szükségletek és megjelenő problémák tükrében való rendszeres módosítása, melyhez a döntéstámogató rendszerek nyújthatnak segítséget, de elsősorban a szakképzett emberi erőforrásokra alapszik. A közlekedésirányítás automatizáltságának kérdése így elsősorban azon eszközök és rendszerek esetén merül fel, melyek aktív szabályozó szerepet töltenek be. Ilyenek például a forgalomirányító lámpák, melyek esetén az adaptivitás és az összeköttetés bizonyos forgalomfigyelő rendszerekkel növelheti a hatékonyságot, vagy a vasúti sorompók és jelzőlámpák, melyeknél a vasúti rendszer flottakövetéséhez kapcsolódva fokozhatják a biztonságot.

A másik oldalon a jármű bizonyos szintű automatizálása áll. Ez jelentheti a teljes automatizálást, azaz hogy a jármű önállóan hozza a döntéseket, és vezérli a járművet. Az ilyen szintű automatizálás megvalósítására sok kísérlet történt, több nagyobb projekt is született. Közülük talán az egyik leghíresebb az EUREKA Prometheus Projekt, mely egy közös európai fejlesztés volt számos egyetem és autógyár részvételével. Maga a rendszer magas szintű gépi látásra támaszkodott: 18 kamera képét dolgozta fel a döntések meghozatalához [Shojania prezentáció]. Az önálló, vezető nélküli közlekedési eszközökkel kapcsolatos kísérletek lesznek valószínűleg a legfontosabb fejlesztési területek ebben a témában a jövőben.

A járművezetés automatizálásának egy megelőző lépcsőfokának tekinthetők a járművezetést támogató rendszerek fejlesztése, melyek ugyan még nem hoznak önálló döntéseket, de lehetőség szerint a legoptimálisabban támogatják annak meghozatalát.

A járművezetés automatizálásának legfontosabb elemei [Shojania] szerint:

- Az út, és ezen belül a sávok felismerése
- Akadályok felismerése
- Az elhaladó járművek érzékelése
- A gépjármű által megtett út nyomon követése
- A közlekedési jelzések érzékelése és értelmezése.

A fent említett felsorolás alapja lehet egy járművezetést támogató rendszernek is, mivel a beavatkozást teljesen különállónak tekinti. Mind az öt pont megvalósítása hatalmas feladat lenne, így a felsoroltak közül az utolsó megvalósítását tűztük ki a rendszer céljának. A projektünk, mivel kizárólag vizuális információk alapján dolgozik, a különböző feladatokhoz a kamera képét dolgozza fel, és ebből szűri ki a lényeges információt.

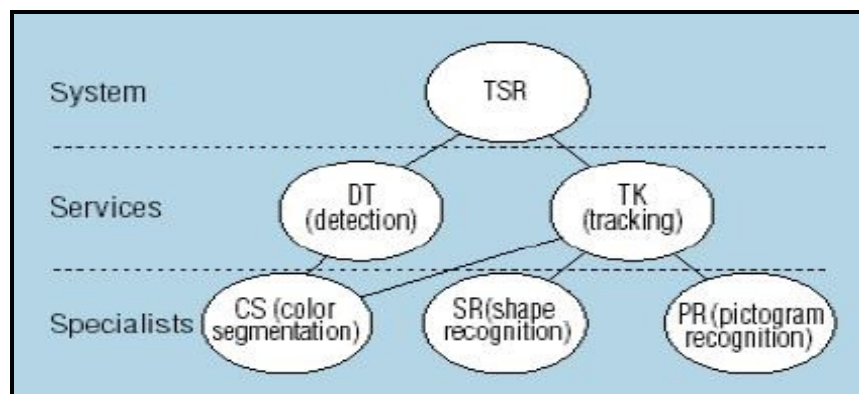
2.2 Képfeldolgozáson alapuló támogató rendszerek

Mivel a közlekedés-automatizálás manapság egyre népszerűbb kutatási terület, ezért a témában elmélyülve nagyon sok, a miénkhöz hasonló, képfeldolgozáson alapuló rendszert találhatunk. Létezik ezek között teljesen megvalósított, terv szinten felvázolt, specifikus hardvert igénylő, vagy általános eszközökkel működő rendszer is. Mindegyik bemutatása és elemzése nyilvánvalóan lehetetlen vállalkozás lenne, így ezek közül csupán néhány rendszer elvi működésének és felépítésének hasonlóságait nézzük meg, amelyek a projekt megtervezésében és megvalósításában segítséget nyújthatnak.

2.2.1 Prometheus TSR

A már az előző részben említett, 1986-ban indult pán-európai Prometheus projekt költségvetése meghaladta az egymilliárd dollárt. A teljes rendszert 18 kamera és 60 számítási csomópont alkotja, melyek eleinte Transputer, majd PowerPC601-es processzorokat alkalmaztak [Shojania prezentáció]. Nyilvánvaló, hogy ez a projekt teljesen más komplexitású szinten lévő célokat fogalmazott és valósított meg, mint a projektünk, de a jelzőtábla felismerő alrendszerének vázlatos működését mindenképpen érdemes szemügyre venni.

Ezt az ún. TSR (Traffic Sign Recognition) alrendszert a Daimler-Benz tervezte. Alapvető feladata a közlekedés irányítását ellátó objektumok felismerése. Ezen képfeldolgozó rendszer szoftverének háromszintes, egyszerű felépítési vázlatát a következő (2.1. ábra):



2.1. ábra

A Prometheus TSR alrendszere [Shojania prezentáció]

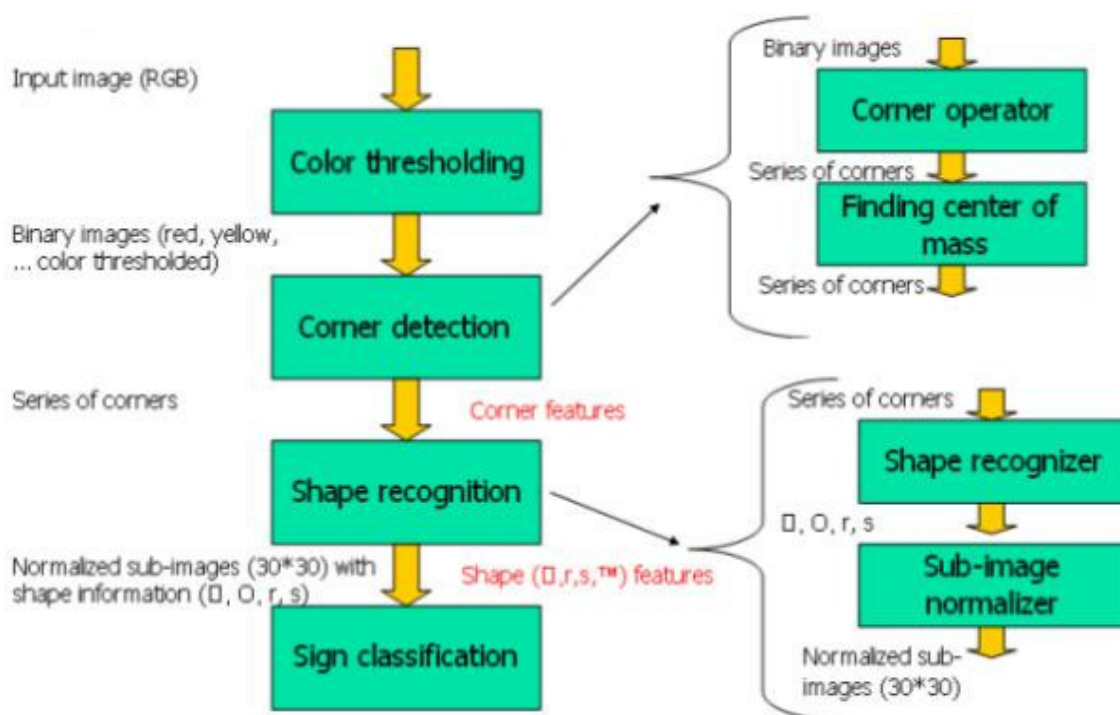
A rendszer az alábbi két fő modulból áll:

1. DT (detection): ez a folyamat vizsgálja át a képet a lehetséges jelzőtáblákra
2. TK (tracking): ez a folyamat ismeri fel a jelzőtáblát, és nyomon követi a következő képkockákon is

A DT és TK folyamatok három speciális modult használnak fel az észleléshez. Az első neve CS (color segmentation), azaz a forráskép felosztása színek alapján, és az összefüggő régiók meghatározása a potenciális jelzések felismerése végett. A második „specialista” az SR (shape recognition), ami a lehetséges jelzőtáblát az alakja alapján próbálja osztályozni. A harmadik modul, a PR (pictogram recognition) a már megfelelő színű és alakú jelzőtáblát próbálja meg besorolni annak piktogramja alapján.

2.2.2 Real-time Traffic Sign Detection

A [Shojania] által felvázolt mechanizmus négy részből áll. A szín-alapú szegmentálás lényege egy színmaszk alkalmazása, majd az eredmény küszöbölése úgy, hogy végül egy olyan bináris képet kapjunk, ahol a feltételezett jelzőtáblák kontúrjait jelölik a fehér pixelek. Az alakzat-felismerés egy sarokdetektáló konvolúciós maszk alkalmazásával kezdődik, amit az előző lépés eredményén hajtunk végre. Ezután geometriai transzformációkkal normalizáljuk a kapott eredményt egy fix pixel mátrixba. Ez a mátrix, mint normalizált képinformáció már átadható utolsó lépésként egy megfelelően betanított neurális hálónak, ami képes lesz osztályozni a jelzőtáblát. A teljes folyamatot összefoglalva láthatjuk a 2.2. ábrán.



2.2. ábra
Real-time Traffic Sign Recognition [Shojania]

Mivelhogy az első lépés, azaz a színek alapján történő küszöbölés nálunk is a feldolgozási folyamat alapját képezi, érdemes [Shojania] tanulmányának ezt a részét alaposabban is megvizsgálni, legfőképp mivel többféle megoldást is ajánl ennek végrehajtásához.

Az első lehetőség szerint a kép küszöbölhető a 2.3. ábrán látható kifejezés által, ahol a $g(x,y)$ az eredménykép egy pixele, k_1 és k_2 a felvett értékek, amelyek közül egy R/G/B minimum és maximum korlát-párhoz viszonyított $f_{r/g/b}(x,y)$ érték választ. Ez a függvény adja vissza az (x,y) koordinátájú képpont egyes összetevőit a forráskepből.

$$g(x,y) = \begin{cases} k_1 & \text{if } \begin{cases} R_{min} \leq f_r(x,y) \leq R_{max} \\ G_{min} \leq f_g(x,y) \leq G_{max} \\ B_{min} \leq f_b(x,y) \leq B_{max} \end{cases} \\ k_2 & \text{otherwise} \end{cases}$$

2.3. ábra
Egyszerű RGB küszöbölés [Shojania]

Ez az egyszerű, kis számításigényű küszöbölési módszer jól működik abban az esetben, amikor a fényviszonyok minden bejövő képnél, és azok teljes területén ugyanolyanok, azonban ez a valóságban még a legdrágább képrögzítő eszközök kiváló minőségű, automatikus színegyensúly-korrigált képei esetén sem garantálható.

Második lehetőségként a HSL/HSV színtér alkalmazását említi, amelyet nem fejt részletebben ki, csak azt írja le, hogy bár ez jó megoldás lehet a probléma kiküszöbölésére, és gyakran alkalmazott is, azonban hatalmas számítási teljesítményt igényel.

$$g(x, y) = \begin{cases} k_1 & \text{if } \begin{cases} R'_{min} \leq f_r(x, y) \leq R'_{max} \\ G'_{min} \leq \frac{f_g(x, y)}{f_r(x, y)} \leq G'_{max} \\ B'_{min} \leq \frac{f_b(x, y)}{f_r(x, y)} \leq B'_{max} \end{cases} \\ k_2 & \text{otherwise} \end{cases}$$

2.4. ábra

Javított RGB küszöbölés [Shojania]

Végül bemutat egy olyan megoldást, amely az első javított verziójának is tekinthető. A kiválasztó kifejezés 2.4. ábrán található módosításával az RGB alapú küszöbölés már változó fényviszonyok között is alkalmazható.

2.2.3 TSR for Intelligent Vehicle/Driver Assistance System

Ezen rendszer alapja, hogy a beérkező képkockából először létrehoz egy szürkeárnyaltos képet, amelyen azután egy simítást végez. Az eredményen éldetektálást hajt végre, amelynek kimenete egy olyan bináris kép, ami feltehetően kiemelve tartalmazza a jelzőtáblát, vagy jelzőtáblákat.

Erre a képre ezután lefuttat egy olyan algoritmust (OpenCV: cvFitEllipse), amely megkeresi azon az optimálisan elhelyezhető ellipsziseket. Ezzel feltételezhetően sikerült meghatározni azon objektumok pozícióját és kiterjedését, amelyek valószínűleg jelzőtáblák. Ezen részképek ezután átméretezésre kerülnek olyan 30x30-as ún. „blob” mátrixokba, amelyek alapján egy neurális hálózat végzi a táblák felismerését.



2.5. ábra

TSR for Intelligent Vehicle/Driver Assistance System [Lorsakul-Suthakorn]

[Lorsakul-Suthakorn] rendszerének tervén (2.5. ábra) láthatóak a főbb folyamatok, amelyek szükségesek a kép kiértékeléséhez. A feldolgozás előtt a beérkező videó jel átalakításra kerül, hogy előálljon az RGB kép. Ezután történik annak elő-feldolgozása, majd az objektum pozíciójának meghatározása. Ennek átméretezése után következhet a felismerés.

2.2.4 Automatic TSR in Digital Images

[Hatzidimos] tanulmányában arra világít rá, hogy milyen problémákkal szembesülhetünk, amennyiben egy digitális képen objektum-felismerést akarunk végezni. Elsősorban a lehetséges objektumok óriási mennyisége, és a közöttük lévő hasonlóság okozza a problémát.

Megoldást az jelenthet, hogyha a rendszer rendelkezik ún. „előzetes ismerettel”, azaz a felismerendő objektum lehetséges attribútumaival (szín, alak, méret), és a feldolgozás folyamata ehhez igazodik. Példaként felvázol egy lehetséges algoritmust a közlekedési táblák felismerésére.

A javasolt feldolgozás lépései:

I. Első fázis: a jelzés pozíciójának detektálása a képen

1. ROI (Region of Interest) szegmentáció bináris küszöböléssel
2. Éldetektálás és élvékonyítás
3. Régió meghatározás, és csoportosítás
4. Vonaldetektálás, kapcsolódási pontok meghatározása
5. Alakzat ellenőrzés a vonalak által bezárt szög alapján
- 6.1 Ha az alakzat sokszög, súlypont és csúcsok meghatározása
- 6.2 Ha az alakzat elliptikus, RHT (Randomized Hough Transform) alkalmazása a középpont és méret meghatározásához

II. Második fázis: a jelzés felismerése

- 1.1 Ha az alakzat sokszög, Affin transzformáció alkalmazása a mintaképeken, hogy összehasonlíthatóak legyenek az azonos koordinátarendszerben
- 1.2 Ha az alakzat elliptikus, hasonlósági transzformáció alkalmazható
2. Keresési terület meghatározása, azaz a tábla alakzatán kívüli, de a ROI-n belüli pixelek homogénné színezése
3. Kereszt-korrelációs egyeztetés alkalmazása a keresési terület pixelei, és a mintaképek között. Amelyik esetén a legnagyobb a korrelációs együttható, az a minta az eredmény, és így megtörtént a felismerés.

2.2.5 Road Sign Detection and Recognition

[Shneier] rendszerének alapötlete, hogy a beérkező képet többféle módon küszöböli a színinformációk alapján. A jelzőtáblákat a szín alapján osztályokba sorolja, melyek mindegyikére megállapítható egy logikai kimenetű egyenlőtlenség, amely alapján előáll egy bináris kép.

Például a figyelmeztető táblák esetén (fontos megjegyezni, hogy a rendszer az amerikai szabványokra íródott, ahol a figyelmeztető táblák sárgás színűek, és rombusz alakúak), a kimeneti képet az alábbi kifejezés határozza meg (az R , G , B értékek egy pixel színösszetevőinek intenzitását jelölik, az a , b , és c értékek pedig a jellemző együtthatók):

$$\frac{R}{G} > a \quad \wedge \quad \frac{R}{B} > b \quad \wedge \quad \frac{G}{B} > c$$



2.6. ábra Az arányokkal dolgozó küszöbölés eredménye [Shneier]

Minden jelzőtábla-osztályra meghatározható egy rá jellemző a , b , és c együttható hármassal, amivel megjeleníthetők a potenciálisan fontos régiók a képen (2.6. ábra).

Ezután a bináris kép kerül feldolgozásra, ahol első lépésként az esetlegesen „megszakadt” régiókat köti össze. A következő lépésben geometriai tulajdonságok (kiterjedés, alak) alapján az egyes régiókról eldönthető, hogy lehetnek-e jelzőtáblák, amennyiben igen, egy átméretezésre kerül sor. Végezetül a rendszer a lehetséges képrészletből a meghatározott tábla-osztály alakja alapján kiszűri a háttérrel, majd összehasonlítja a már eltárolt mintákkal.

2.2.6 A megvizsgált rendszerek összehasonlítása

A bemutatott rendszerek közös jellemzője, hogy egy előfeldolgozási fázis után, amelyben különböző szempontok alapján küszöböli a beérkező képet, próbálja meg meghatározni azon régiókat, amelyek potenciálisan tartalmazzák a felismerendő objektumokat.

A [Shojania] és [Shneier] által felvázolt módszer esetén a régiók meghatározásához a képpontok színjellemzőit használják fel, melyet egy alakzat-felismerés követ. [Shneier] módszere egyszerűbb, azonban hátránya, hogy az amerikai szabványt feltételezi, ahol jobban elkülönülnek a jelzőtábla-kategóriák jellemző színei. [Lorsakul-Suthakorn] és [Hatzidimos] módszere ezzel ellentétben elsődlegesen az éldetektálást használja fel, ami tisztább képet adhat különböző fényviszonyok esetén az alakzat-felismerés számára, azonban jóval nehezebb feladat meghatározni a küszöböt, és valószínűbb a hamis régiók felvétele.

[Lorsakul-Suthakorn] alakzat-felismerése csupán az elliptikus táblákra korlátozódik, míg [Hatzidimos] algoritmusai szélesebb körben alkalmazható a háromszög, téglalap, és kör alakok megkülönböztetése miatt. Azonban utóbbi hátránya, hogy nagyon sok és bonyolult műveletet végez, ami túl lassú lehet a valós-idejű feldolgozásra.

[Shojania] és [Lorsakul-Suthakorn] módszere a pozíció meghatározása után neurális hálót alkalmaz, melynek előnye, hogy gyorsabban reagál a bemenetre valós időben, és kevésbé érzékeny a zajokra, azonban bonyolultabb felépítést és hosszabb tanítási folyamatot követel meg, míg [Hatzidimos] és [Shneier] mintaillesztése jóval egyszerűbb, de lassabb, és pontosabb bemenetet is igényel.

Mindegyik megoldásnak vannak előnyei és hátrányai, az optimális megoldás megkeresését mindig az aktuális körülményekhez kell igazítani. A legfontosabb szempontok a következők: mekkora pontosság illetve mekkora feldolgozási sebesség szükséges, milyen speciális fényviszonyok állnak fenn, valamint milyen hardveres eszközök állnak rendelkezésünkre.

2.3 Összefoglalás a projekt szemszögéből

Mint láthattuk, a legtöbb jelzőtábla felismerő rendszer alapvetően hasonló felépítésű, a különbségek az egyes részműveletekben vannak. Vannak olyanok, amelyekben több feladatot kapnak a hagyományos képfeldolgozási algoritmusok, maszkolások, szűrők, küszöbölések, és vannak olyanok is, amelyekben a hangsúly a mesterséges intelligencia egyes eszközein, például a neurális hálózatokon van.

Összegezve elmondható, hogy általában szükség van mindkét eszközre, hiszen mindkettőnek vannak erősségeik és gyengeségeik. A képfeldolgozó algoritmusok nagyon megbízhatóak, sok tapasztalat gyűlt össze ezekkel kapcsolatban, és gyorsan képesek kezelni a nagyméretű képeket is. A neurális hálózatok ezzel szemben viszont adaptívak, képesek osztályozni a bemeneteket, és kevésbé érzékenyek a változásokra.

A 2.7. ábrán összefoglalhatjuk, hogy milyen részfolyamatok szükségesek ezen rendszerek számára, amelyek alapjául szolgálnak a mi projektünk számára is.



2.7. ábra
Jelzőtábla felismerő rendszerek jellemző fázisai

A beérkező videó jelekből a teljes feldolgozás sebességétől függően kell kiválasztani képkockákat, amelyek bemenetként szolgálnak a teljes folyamat számára.

A beérkező képnek át kell esnie egy előfeldolgozáson, amelyben egyrészt egyszerűsítésre kerül, másrészt lefutnak a szükséges algoritmusok (pl. szintelenítés, küszöbölés, éldetektálás).

Az eredményben előállt képet megfelelően szegmentálni kell, meg kell határozni azon régiókat, amelyek tartalmazhatnak a rendszer számára fontos objektumokat. Az itt alkalmazott módszer határozza meg leginkább az előfeldolgozás mikéntjét.

A meghatározott régiók, a felismeréshez szükséges átméretezése után jöhet maga a felismerés folyamata, amely történhet betanított neurális hálózattal, vagy előzetes minták alapján történő egyeztetéssel.

Előrevetítve a tervezési fázisra, a Visual Driving Assistant rendszer az előfeldolgozási fázisban a vörös illetve a zöld objektumokat kiemelő küszöbölést fog végezni, majd egy előre meghatározott régió belül – mivel az érdekelt objektumok ott helyezkedhetnek el – egy ún. kiértékelő egység fogja azt bejárni potenciálisan fontos jelzések után kutatva (ebből látható, hogy a programunk nem teljesen követi ezt a folyamatmodellt, ugyanis a szegmentálás és felismerési fázis valójában „összeolvad”).

3. A rendszer tervezése

3.1 A projekt céljából adódó sajátosságok

A szoftver feladata általánosítva tehát hasznos információk kinyerése forrásképek halmazából. Ezáltal lényegében a gépi látás témaköréhez tartozik, amely a számos jól definiált és kidolgozott módszer ellenére sokkal kevésbé determinisztikus, sokkal kevésbé jellemző rá a megoldandó feladat – egyértelműen legjobb algoritmus módszertan. Ezt tovább bonyolítja az, hogy sokszor, mint esetünkben is, nagyon fontos szempont a futási sebesség, az operatív tár elérési ideje és egyéb, a rendszer teljes teljesítményét befolyásoló tényezők.

A fentebb leírtak következtében a teljes szoftverkészítés folyamata semmiképpen sem követhet hagyományos, szekvenciális, (pl. vízesés) szoftverfejlesztési modellt, mivel akár minden egyes részfeladat implementálásánál és tesztelésénél olyan problémák merülhetnek fel, amelyek okán vissza kell térni a tervezés fázisához. Természetesen ez más részmodulok működését is jelentősen befolyásolhatja. Könnyen megtörténhet, hogy egy adott algoritmus, bár eleinte nagyon kézenfekvőnek és gyorsnak tűnik, sajnos nem vezet egyáltalán elfogadható eredményre, így teljesen át kell alakítani egy sokkal megbízhatóbb, ámde lassabb megoldásra. Ez azonban a teljes folyamatból jelentős plusz időt von el, így elkerülhetetlen egy másik programrész teljes átalakítása.

Az eredmények, a prototípusok tesztelése során a leírtaknak megfelelő visszatérés és újratervezés nagyon sokszor meg is történt a projekt elkészítése során. Ezt a megközelítést leginkább valósághűen az evolúciós fejlesztési módszertan írja le, így ezt próbáltuk alapul venni a fejlesztés folyamán. A gépi látás és az evolúciós modell olyan közel állnak egymáshoz, hogy már léteznek a genetikus algoritmusok ötletén alapuló automatikus szoftver evolúciós mechanizmusok is, ahogy azt [Ebner] bemutatja.

3.1.1 Az evolúciós fejlesztési modell

Mivel ez a módszertan alapvető szerepet töltött be a megvalósítás során, így röviden összefoglaljuk a modell elemeinek leírását. [Sommerville] vázlatos megfogalmazása alapján a modell két alapvető eszköze:

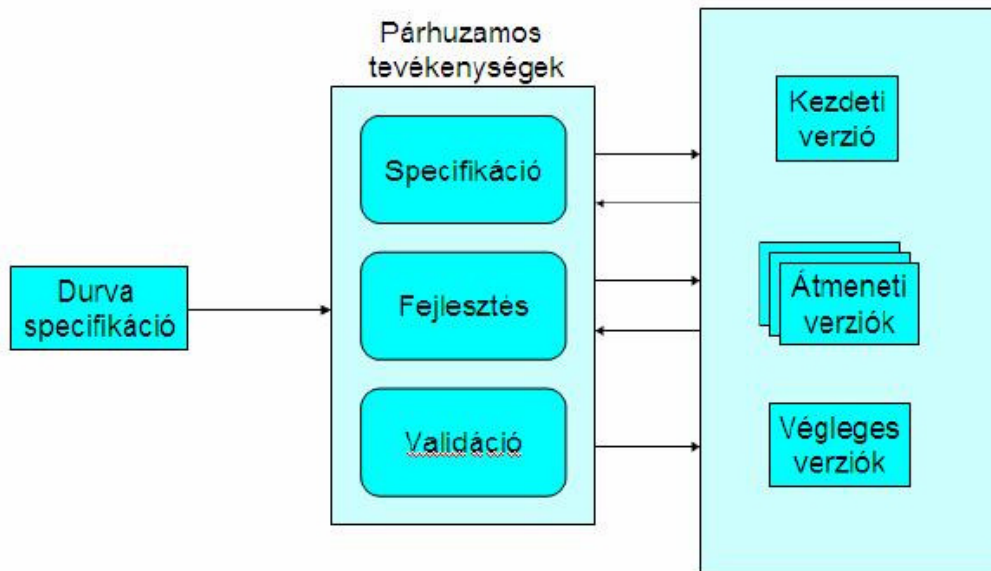
„Kísérletező fejlesztés:

Cél: a megrendelővel együtt egy kezdeti durva specifikációból a végleges rendszert kialakítani. A biztos követelményekből kiindulva a megrendelő igényei szerint újabb funkciókkal bővíthető a rendszer.”

Eldobható prototípus:

Cél: a homályos követelmények tisztázása. A legkevésbé kiforrott követelményekből indul, hogy tisztázza a valós igényeket.”

Esetünkben a megrendelő szerepébe az általunk felvázolt projekt-célok kerülnek, valamint kiemelt hangsúlyt kap az, hogy a szoftver valós időben működjön. A valós igények tisztázása pedig azt jelenti, hogy a rendszer jellemzőiből adódó korlátokat pontosan tisztázhatjuk, és a program pontosságának hangolását, valamint a kimenetek elemzését ezekhez optimálisan igazíthatjuk.



3.1. ábra
Az evolúciós fejlesztési modell [Sommerville]

A 3.1. ábrán láthatjuk a modell grafikus reprezentációját.

A célkitűzések tisztázásával az 1. fejezetben megtörtént a durva specifikáció felvázolása, valamint ezek részbeni finomítása is. A specifikáció finomítására példa a zöld lámpák pontos kezelése, azaz annak ellenőrzése, hogy egy érvényes kötelező elsőbbségadásra figyelmeztető tábla alatt hol helyezkedhet el egy, azt érvénytelenítő zöld lámpa, vagy az, hogy pontosan milyen távolságból oldható meg nagy bizonyossággal a felismerés. Miközben a validáció, azaz az értékelés egy részmodul elkészülte esetén megkezdődött, már párhuzamosan elindult a következő részfeladat megvalósítása, így annak tükrében vizsgálhattuk az elkészült kezdeti verzió, illetve a későbbi próbaverziók működését.

Ennek a modellnek azonban meg kell említeni a problémáit is, hogy megfelelően tudjuk kezelni, és vizsgálni a szoftvert ebből a szempontból is.

Az evolúciós modell problémái [Sommerville] megfogalmazásában:

- „• *A fejlesztés nem átlátható;*
- *A rendszerek gyakran rosszul strukturáltak;*
- *Speciális felkészültségre lehet szükség (pl. rapid prototyping nyelvek)."*

A felsoroltak kiküszöbölése, és a folyamatos kódellenőrzés, osztálystruktúrák szükséges átalakítása végig jelen volt a fejlesztés során.

3.1.2 A tesztelés és hangolás kiemelkedő szerepe

Az evolúciós fejlesztési modellben a validációs fázis mindannyiszor szükséges, ahányszor egy átmeneti verzió elkészítésre kerül. Ennek eredménye jelentősen befolyásolhatja a rendszer tervét, de még akár a pontos specifikáció is módosulhat. Ennél a módszertannál tehát döntő szerep jut a folyamatos teszteléseknek és azok megfelelő kiértékelésének, mely sokszor igen nehéz feladatnak bizonyul.

Egy, a gépi látás témakörébe tartozó program esetén gyakran előfordul az, hogy egy eleinte megfelelőnek talált feldolgozási folyamat komoly átalakításra, vagy akár teljes lecserélésre szorul, mivel vagy a kimenete nem elégséges, vagy az adott környezetben túlságosan lassúnak számít, ezt [Ebner] tanulmánya is külön kiemeli. Mindezekon felül a használt algoritmusok megfelelő paraméterezése, finomhangolása is időigényes feladat, és könnyen előfordulhat az is, hogy a módszer alkalmatlansága csak hosszas hangolások után derül ki egyértelműen. Ahogy ezt az elkövetkezendőkben látni fogjuk, ez a mi esetünkben sem volt másképp.

3.1.3 Az átmeneti verziók tesztelési szempontjai

Amikor elkészült egy prototípus, azt egészében, és modulokra bontva is meg kell vizsgálni a teljes elkészülendő rendszer tükrében. Ez elsősorban azt jelenti, hogy hiába működik első ránézésre elegendően gyorsan és kielégítően egy bizonyos egység, hogyha annak kimenete – még hogyha csak csekély mértékben is – eltér az elvárttól, és ez ez komolyan veszélyezteti a további, akár még nem implementált modulok helyes funkcionalitását.

A leírtakból következően, tehát sosem elegendő önmagában értékelni egy elkészült részegységet, azt mindig a jövőben elkészülő teljes rendszer szemszögéből kell megfelelő teszteléseknek alávetni. Ennek okán a validáció, és az ahhoz tartozó vizsgálati szempontok meghatározása hosszadalmas, az esetlegesen változó specifikációtól függő feladat, így a fejlesztési idő legnagyobb részét ez emészti fel.

3.1.4 Az evolúciós fejlesztésből adódó hátrányok kiküszöbölése

Egyértelműen következik az – ahogy azt [Sommerville] leírásából is láthattuk – a tény, hogy ennek a fejlesztési módszertannak számos komoly hátulütője, nehézsége is van, amelyeket mindenképpen valahogy kezelni kell, különben jelentősen hátráltatná a projekt előrehaladását.

Azonnal felmerül a kérdés, hogy az egymás után következő, sorozatos – sokszor a sürgető tesztelhetőség igénye miatt „gyorsan összedobott” – átalakításokon áteső átmeneti verziók struktúrája és forráskódja mennyire szabályos és átlátható. Sajnos a válasz az, hogy szinte semennyire. Ez egy mindenképpen kritikus probléma a projekt fejlesztésében, így ezt, amennyire csak lehetséges, korrigálni kell.

Az általunk alkalmazott megoldás lényege az, hogy néhány prototípus után elkészítettünk egy „mérőföldkőként” funkcionáló verziót, melynek mind az implementációját, mind a hozzá tartozó aktuális rendszertervet alaposan átvizsgáltuk, és a működést – lehetőség szerint nem megváltoztatva – újrastrukturáltuk. Ez leginkább a programkódot érintő kérdés volt, ahol az osztályok, azok metódusainak és változóinak összességét, azok elnevezéseit, kapcsolatait és láthatóságait kellett ellenőrizni és megfelelően újrászervezni. Ez egy hosszadalmas műveletnek tekinthető, mivel minden esetben a teljes, akár a nem megváltozott részeknél is szükséges volt rá a teljesen konzekvens forráskód elérése érdekében.

3.2 Előzetes rendszerterv

A Visual Driving Assistant rendszerben az objektumokat feltehetőleg tartalmazó részképek meghatározásánál két jellemzőt veszünk alapul. Az egyik az, hogy a célkitűzésben meghatározott objektumok jellemző színe a markáns, környezettől elütő piros, mint tiltószín, ezáltal a képpontokra olyan küszöbölés hajtható vége, amelynek eredménye egy bináris kép. Ezen valószínűleg láthatóan elkülönülnek azon területek, amelyeken a vörös összetevő dominál. Természetesen ezen megjelenik minden olyan objektum, amelyre ez a szín a domináns (3.2. ábra).

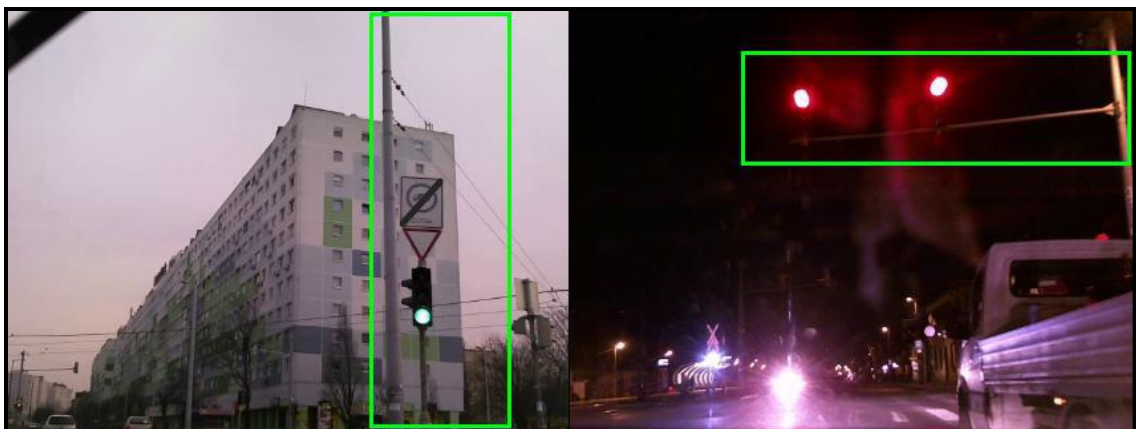


3.2. ábra

Forráskép és egy a vörös összetevőre küszöbölt kép RGB arányokkal

A küszöbölés történhet RGB színtérben az egyes összetevők arányainak vizsgálatával, ahogy azt [Shneier] esetében láthattuk. Ez kiegészülhet esetleg minimum/maximum értékek kizárásával csupán egy meghatározott intervallumon, de történhet a teljes, 0-tól 255-is terjedő skálán is. Szóba jöhet még ezen felül a forráskép konvertálása valamilyen színezettséget, telítettséget és fényerőt leíró HSV/HSL színtérbe is. Előbbi előnye a gyorsaság, mivel a beérkező képkocka maga is RGB ábrázolású, de itt nehezebb lehet az arányok hangolása. Utóbbinál a küszöbölési feltétel jelentősen leegyszerűsödik, viszont a konvertálás jelentős időt vehet igénybe.

A másik kihasználható tulajdonság, ami a közlekedésből származó bemeneti képekre jellemző, hogy a számunkra fontos objektumok, a jelzőtáblák és jelzőlámpák a bejövő kép két kitüntetett régiójában, a jobb szélső, illetve felső (3.3. ábra) sávjában tűnhetnek fel.



3.3. ábra

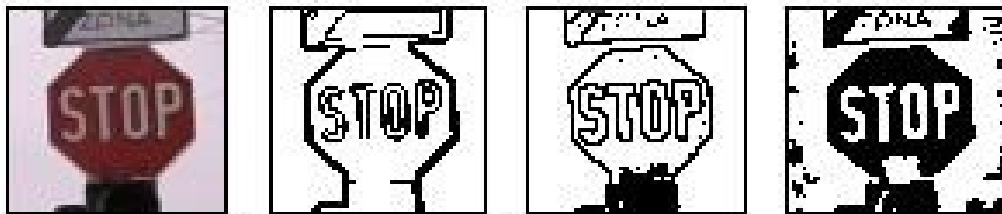
Az érdekelt régiók elhelyezkedése

Az érdekelt régiók, ROI-k (Region of Interest) pontos meghatározása a fejlesztés során nélkülözhetetlen. Szerepe valójában az, hogy gyorsítsa az objektumok keresését, hiszen kevesebb lehetőséget kell megvizsgálni, illetve hogy olyan, a küszöbölés után megmaradó nagyobb kiterjedésű gyakran feltűnő piros objektumokat, mint például egy piros autó, féklámpa stb. a rendszer ne vegyen figyelembe.

A lehetséges objektumok pozíciójának meghatározása tehát a vizsgált sávokba eső, a bináris képen megfelelő kiterjedésű, és oldalarányú közel négyzetes részeképek meghatározásával történik. Miután a régiók elhelyezkedését megkaptuk, az eredeti kép ezen területeinek tartalmából előkészítheti azt a felismerést végző modul számára.

Itt szintén több lehetőség közül kell kiválasztani az optimumot. Ezt a felismerő modul működéséhez és jellemzőihez kell igazítani, az szabja meg azt, hogy milyen méretű és milyen technikával legyen előkészítve az eredeti képből a kivágott rész.

Több lehetőség is kínálkozik, mint például egy Sobel-éldetektor alkalmazása a régión, amelyet egy küszöbölés követ, vagy egy hiszterézises küszöbölés, amelynek paramétereit a régió tulajdonságaiból határozza meg az előkészítő modul (3.4. ábra).



3.4. ábra

Különböző előfeldolgozások

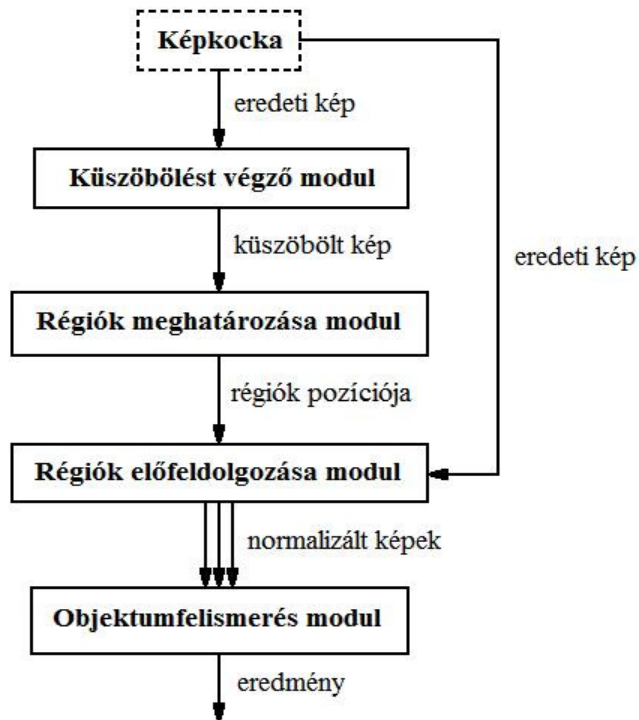
(jobbról balra: eredeti kép, Sobel éldetektor, hiszterézises küszöbölés, Niblack algoritmus)

Azt, hogy végeredményképpen melyik feldolgozási technika és mekkora méret bizonyul a legjobbnak, azt az implementálás teszteléssel összekötött fázisa határozza meg prototípusok alapján, ugyanúgy azt is, hogy a felismerő modul egy egyszerű egyrétegű előrecsatolt neurális hálózat, vagy egyfajta minta-összehasonlítási algoritmust használó objektum lesz-e, esetleg a kettő keveréke valamilyen módon.

3.2.1 A képelemző és objektumfelismerő alrendszer felépítése

Összefoglalva tehát láthatjuk, hogy a rendszer előzetes terve alapján a képelemző és objektumfelismerő alrendszer négy főbb modulra osztható: küszöbölést végző modul, fontos objektumokat potenciálisan tartalmazó régiókat meghatározó modul, ezen régiókat előfeldolgozó modul, és felismerést végző modul. A folyamatot összefoglalva láthatjuk a 3.5. ábrán.

A kezdeti rendszerterv szerint tehát egyetlen küszöbölt kép fog első lépésként előállni – a zöld lámpák ellenőrzéséhez szükséges bináris képet csak igény esetén állítjuk elő, ez nem szerepel az ábrán – majd ezen képet bejárva egy lehetőség szerint egyszerű algoritmussal keressük meg azon régiókat, amelyek célobjektumok küszöbölt képét tartalmazhatják. Végül ezen régiók átesnek egy külön előfeldolgozáson, ahol a megfelelő szűrések és az átméretezés megvalósul, majd ez szolgáltatja a bemenetet a felismerő modulnak. Ez a működési vázlat azonban a későbbiekben megváltozott, ahogy azt látni is fogjuk.

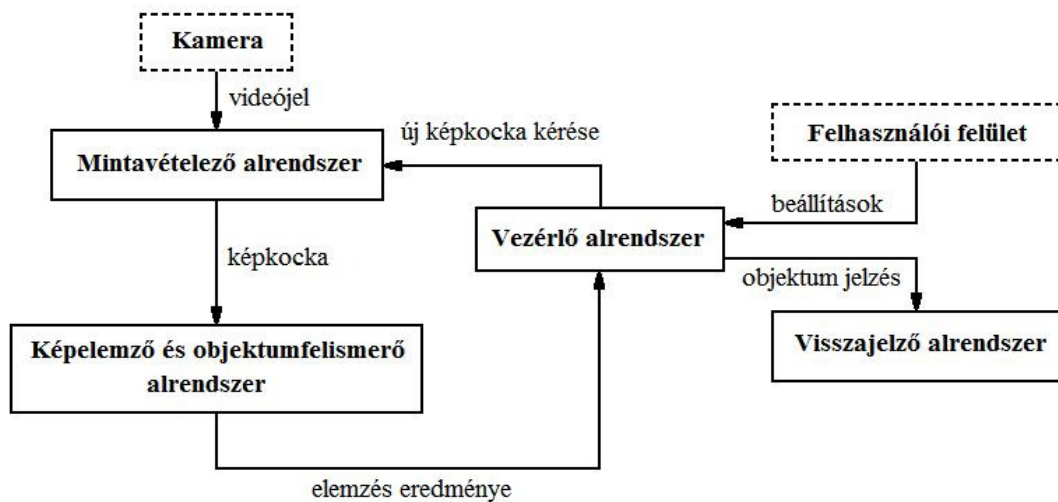


3.5. ábra

A képelemző és objektumfelismerő alrendszer vázlatos felépítése a kezdeti verzióban

3.2.2 A teljes szoftver-rendszer felépítése

A teljes rendszert tekintve meghatározhatjuk az alábbi főmodulokat: vezérlő, mintavételező, visszajelző, valamint a fent említett képelemző és objektumfelismerő. A vezérlő adja meg a mintavételezőnek, hogy mikor küldhet képet az elemzőnek, és annak kimenete alapján, tájékoztatja a visszajelző modult a felismert objektumokról. A program beállításához és finomhangolásához pedig egy grafikus felhasználó felület tartozik. Az említett alrendszereket és kapcsolataikat láthatjuk a 3.6. ábrán.



3.6. ábra

A teljes rendszer felépítése a kezdeti verzióban

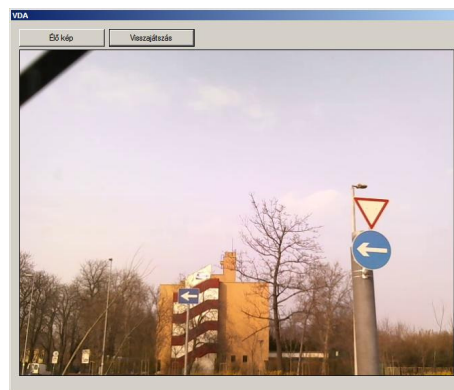
3.3 Az átmeneti verziók során alkalmazott változtatások

Ebben az alfejezetben megpróbáljuk összefoglalni az átmeneti verziók során szerzett tapasztalatainkat, és eredményeinket, amelyeknek visszahatása volt a tervezésre is. A fejlesztés, a modulok prototípusainak elkészítése során a tesztelés és a finomhangolás folyamatosan befolyásolta az eredményeinket, a későbbi módosításainkat.

3.3.1 A videó fájlok rögzítése

Mivel eleinte még nem álltak rendelkezésre „éles”, valódi rögzített felvételek, így a rendszer fejlesztésének ezek nélkül kellett elkezdődnie, de mivel nagyjából körvonalazódott az egyes alrendszerek feladata és algoritmusainak vázlatos felépítése, így ez eleinte nem okozott gondot. Azonban később hogy képesek legyünk a folyamatos tesztelési igényt kielégíteni, szükségünk volt egy olyan bemeneti forrásra, amivel bármikor előállíthattuk a program tesztelésének céljából a megfelelő inputot. Ha minden egyes alkalommal autóba kellett volna ülnünk ahhoz, hogy tesztelhessük a programot, az egész napot gépjárműben töltöttük volna. Az egyetlen alternatíva tesztfelvételek készítése volt.

Ehhez elkészítettünk egy különálló segédprogramot (3.7. ábra), amelynek feladata csupán a webkamerából beérkező képfolyam rögzítése volt egy videófájlba. Elsődleges szempont természetesen az volt, hogy semmi különbség ne legyen aközött, hogy a program felvett, vagy élő képből dolgozik. Az egyéni készítésű program ráadásul azt is biztosította, hogy a felvételeken semmiképpen sem lesznek más, elterjedt programok esetén általánosan használt szűrőhatások, csak a közvetlen hardveres korrekciók fognak működni, éppúgy, ahogy az élőkép esetén.



3.7. ábra
Videórögzítő segédprogram



3.8. ábra
Képkockák a felvételekből

Összesen 222 videofelvétel készült – több mint 3 GB terjedelemmel – ügyelve arra, hogy minden lehetséges esetről legyen több felvétel is minden napszakon belül. A felvételek elkészítése nagyjából 10 órát vett igénybe, amikor is Budapest számos részét bejárva próbáltunk változatos körülményekre törekedni. Mire a próbafelvételeket (3.8. ábra) elkészítettük, már rendelkezésünkre állt egy olyan prototípus, melynek segítségével a képfeldolgozás első fázisát, nevezetesen a küszöbölést tesztelhettük.

3.3.2 A célkitűzések áttekintése a felvételek tapasztalatai által

Vázzuk fel az eredeti célkitűzéseinket az eredeti feladat kiírás „tervezett funkciói” alapján, és a tesztelések során szerzett tapasztalatok során felmerült gondolatainkat:

- *„Az „elsőbbségadás kötelező”, és a „stop, elsőbbségadás kötelező” közúti jelzőtáblák felismerése, mivel ezek kiemelt fontosságúak, elmulasztásuknak súlyos következményei lehetnek. Kiemelkedő szerepüknek köszönhető, hogy ezen tábláknak az egész világon többé-kevésbé megegyező, szabványos, egyedi alakjuk, és kinézetük van.”*

Tapasztalataink alapján az elsőbbségadás kötelező táblák a küszöbölt képen emberi szemmel is nyilvánvalóan elkülöníthetőek, a csúcsára állított háromszög a legtöbb esetben jól látható és felismerhető. A stop táblák esetén viszont annak szabályos nyolcszög alakja miatt távolról nehezebben megkülönböztető a kör alakúaktól.

- *„Közúti jelzőlámpák, és vasúti átkelők tilos jelzéseinek felismerése, figyelembe véve az esetleges bonyolultabb keresztezések esetén szintén látszódo, de egyértelműen nem a járművezetőre vonatkozó jelzések szűrését.”*

A jelzőlámpáknak jól megkülönböztethető „fánk-szerű” alakjuk van a binarizált képen nappali fényviszonyok között. Azonban éjszaka, mint azt az 1. fejezetben már megmutattuk, sajnos a webkamerák tipikusan másmilyen képet adnak vissza ezekről, jellemzően középen hófehér (nagyon magas, szinte egyenlő RGB értékek) területet, mely körül egy rózsaszínes-magentás elmosódott folt látható, így ezek észlelése – legfőképpen amikor több olyan is látható, amelyek nem a járművezetőnek szólnak – különösen nehéz feladatnak bizonyul. A vasúti átkelők tilos jelzéseinek megvalósításánál a helyzet hasonló, ugyanis ott is a vasúti lámpa piros jelzését kívánjuk felismerni. A villogásuk, mely azt a célt szolgálja, hogy még figyelemfelkeltőbb legyen, nem okoz gondot relatíve alacsony frekvenciája miatt.

- *„Zöld lámpák felismerése abban az esetben, amikor a felettük elhelyezkedő táblák csupán a lámpa működésképtelensége esetén szükségesek.”*

A zöld lámpáknak szerencsére a környezettől erősen elütő színük van, így megfelelő paraméterekkel a piroshoz képest sokkal könnyebb küszöbölni. A cél megfogalmazásából kiderül már, hogy ezen objektumok figyelése csak akkor szükséges, amennyiben már történt kötelező elsőbbségadásra figyelmeztető tábla felismerés. Egyéb esetben csak fölöslegesen lassítaná a rendszert.

3.3.3 Az azonosítási folyamatterv problémája

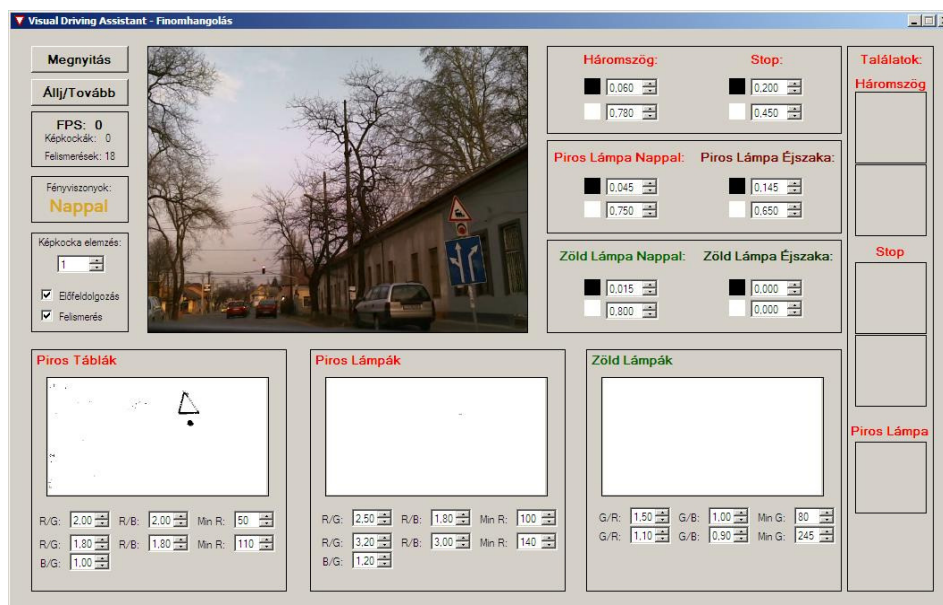
Az átmeneti verziók fejlesztése során szerzett tapasztalataink alapján több ponton is szükséges volt módosítani az eredeti rendszertervet és a teljes képfeldolgozási folyamat működését. Elsősorban az a folyamatmodell változott meg gyökeresen, hogy a beérkező kép küszöbölése után külön meghatározzunk olyan részeket a képen, melyek tartalmazhatnak fontos objektumokat, majd ezeket egyenként külön ellenőrizzük egy felismerő modullal. Ezzel a legnagyobb probléma az, hogy az olyan algoritmusok számítási igénye óriási, amelyekkel a küszöbölt képen hatékonyan megtalálhatjuk az érdekelt részeket, és mindezen felül még külön is ellenőrizni kell azokat egy szintén nem túl gyors hasonlító eljárással.

3.4 A végleges verzió rendszerterve

Miután a szoftver több átmeneti verziót is megért, és sikerült teljesen tisztázni az egyes módszerek előnyeit, hátrányait, időigényét, valamint a rendszer pontos korlátait, elkezdődhetett a végleges verzió megtervezése. Mivel nagyvonalakban már felvázoltuk a kezdeti verzió tervét, és megemlítettük a prototípusok döntő következtetéseit, ezért azok által mutatjuk meg a végleges rendszertervet.

3.4.1 Objektumcsoportok

A célobjektumok csoportosításával kapcsolatban arra a fontos következtetésre jutottunk, hogy több, más paraméterekkel küszöbölt bináris képre van szükség párhuzamosan. Mivel a táblák másodlagos fényforrások, ezért színük és annak eloszlása általában valamennyire eltér a közlekedési lámpákétól, amelyek ezzel szemben elsődleges források, így ez a két kategória, bár mindkettőben a piros szín kiemelése a cél, máshogyan küszöbölt képet igényel. A lámpák alapvető tulajdonságából – azaz, hogy saját fényük van – következik az is, hogy éjszakai fényviszonyok között másmilyenek, mint nappal. Mindezek tükrében arra jutottunk, hogy ezen objektumok esetén különböző paraméterek szükségesek nappal és éjszaka, amelyeknek bár nem kell párhuzamosan előállniuk, mégis külön vizsgálni kell az aktuális fényviszonyokat, amely természetesen szintén plusz terhelést jelent a rendszer számára. A zöld lámpák küszöböléséhez egyértelműen más értékek szükségesek, ezek képezik az objektumok harmadik csoportját. Érdekesség azonban, hogy ezeknek annyira sajátos színük van, hogy küszöbölésük jelentősen egyszerűbb, mint a piros objektumok esetén (de az éjszakai lámpa problémája itt is felmerül). Bár a zöld lámpák ellenőrzése csak akkor szükséges, ha érvényes táblát találtunk, sokkal komplikáltabb lenne külön csak akkor lefuttatni az ehhez tartozó küszöbölést, amikor ez a helyzet fennáll, ráadásul ennek folyamatos futása nem jelent komoly pluszterhelést a rendszernek. A három objektumcsoport és a nekik megfelelő paraméterek beállításának szükségessége jól látható a program hangolási ablakán (3.9 ábra).



3.9. ábra
Képernyőkép egy kései prototípusból
(az objektumcsoportok és hangolásuk módja már nem változott)

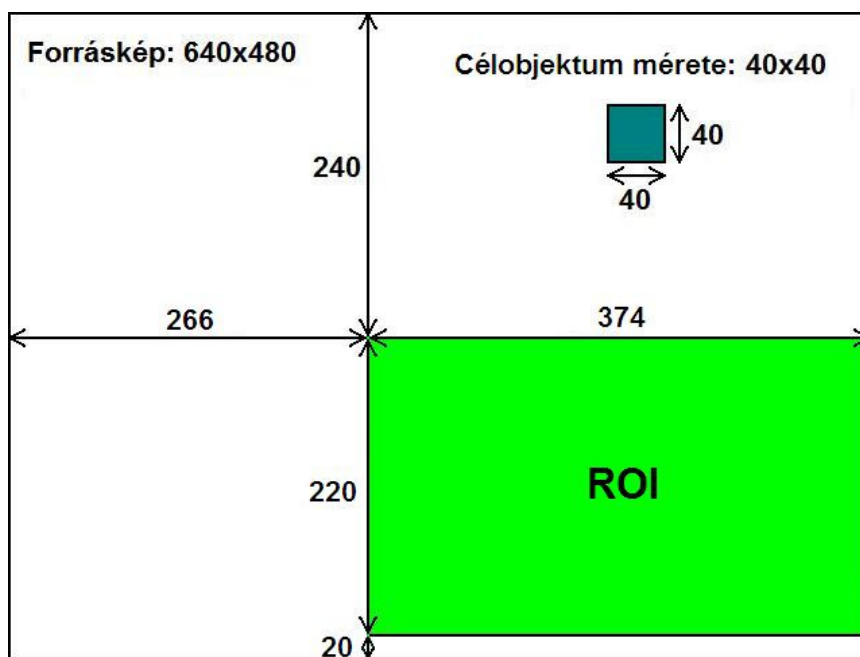
3.4.2 Az előre definiált érdekelt régiók

Az előzetes rendszertervben említésre került az, hogy a beérkező képen a célobjektumokat főleg és félrevezető is lenne a teljes képen keresni, ugyanis azok a kép felső, illetve jobb oldalán helyezkedhetnek el, így két érdekelt régiót, azaz ROI-t határozhatunk meg.

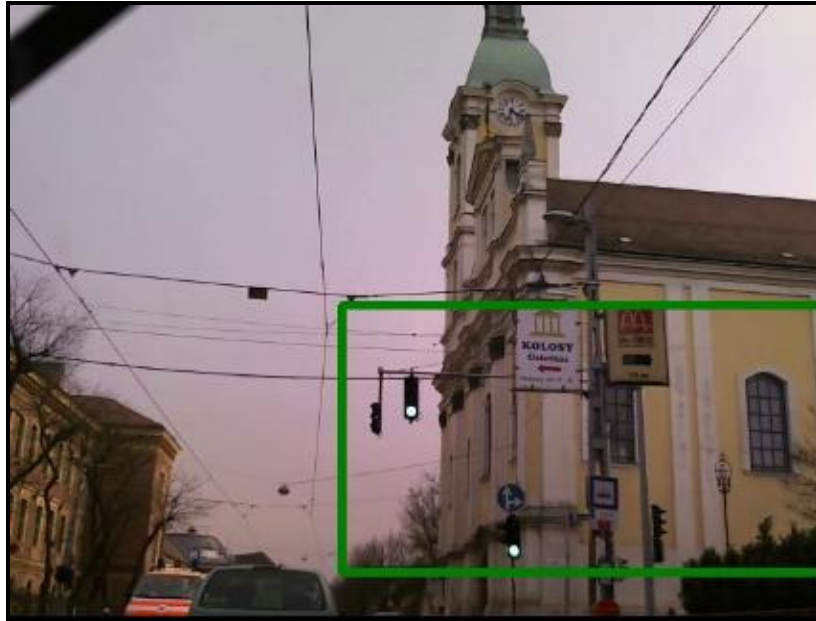
A prototípusok fejlesztése során azonban egyértelművé vált az, hogy felesleges két külön ROI-t definiálni, mivel abban a távolságban, ahol a felismerés szükséges, a járművezetőre vonatkozó felső lámpa (és annak működésképtelensége esetén alárendelt úton a fölötte lévő tábla) bőven a jobboldali ROI-ban helyezkedik el, annak csekély kiszélesítése esetén. Mindezek felül felső lámpák esetén nagyon gyakori az oldalra forduló (jobboldali közlekedés esetén a balra forduló) sáv külön szabályozása is, amelynek tilos jelzése téves felismeréshez vezethet. Mindemellett szól az az érv is, hogy az esetek döntő többségében amennyiben van felső lámpa, van oldalsó is, így a legszélső sávban, amikor előfordulhat, hogy a nagyon mélyre benyúló felső lámpa nem kerül be a régióba, a szélső mindenképpen be fog. Nagyon ritka (főleg sűrű építésű házak közötti szűk egyirányú utcákon fordul csak elő) az, hogy ez a szélső hiányzik, de mivel ezek egysávos utcák, a szükséges távolságban a ROI részét fogja képezni a felismerendő lámpa.

Ezek után felmerülhet a kérdés, hogy ha egyetlen, ráadásul megközelítőleg 4:3-as arányú régiót használunk csak, miért is nem állítjuk úgy be a webkamerát, hogy a teljes kép ezt a területet mutassa. A válasz egyszerű: ahhoz, hogy az eszköz látószögével és a célobjektumok felismerésének szükséges távolságával számolva a kamera pozíciója olyan helyre – az utastéren kívülre, valahol a motorháztető fölé nyúlva – kerülne, ahol nemcsak annak rögzítése nehézkes, hanem már zavaró is lenne a járművezető számára.

A VGA felbontáshoz és a megfelelően pozícionált képhez igazított régió elhelyezkedését a 3.10. ábrán szemléltetjük, majd egy példát is láthatunk a 3.11. ábrán egy közlekedési helyzet képen annak elhelyezkedésére.



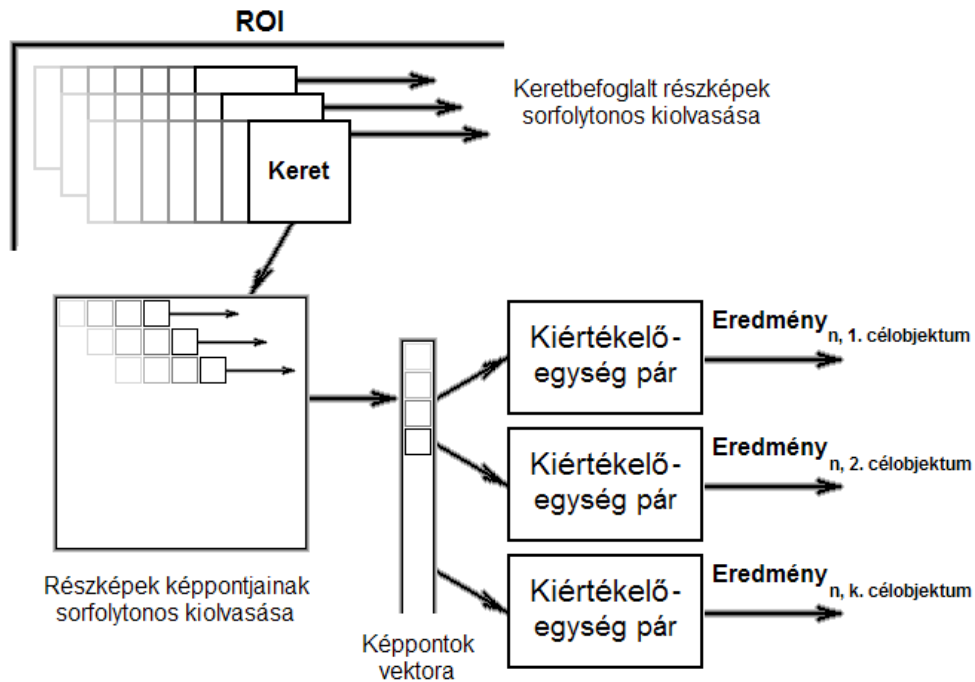
3.10. ábra
Az érdekelt régió elhelyezkedése a VGA felbontású képen



3.11. ábra
Egy képkocka, rajta zöld téglalappal megjelölve a ROI

3.4.3 A felismerés mechanizmusának felvázolása

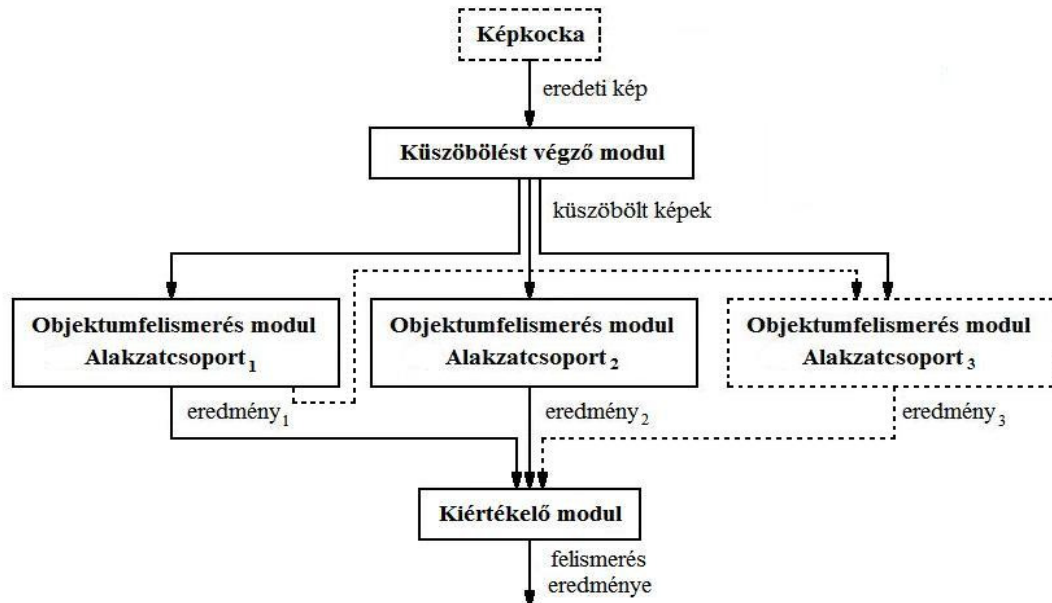
Az objektumdetektálás módszerén is – ahogy azt az átmeneti verziókkal kapcsolatos rész végén már felvázoltuk – változtattunk, ugyanis eredetileg az előfeldolgozott ROI-n belüli szegmentáláshoz nem találtunk elegendően gyors módszert annak fényében, hogy a kapott részképeket újból feldolgozva ismertessük fel a kiértékelő egységek által. Végeredményképpen tehát amellettt döntöttünk, hogy a ROI-t a már küszöbölt képen „járjuk be” megadott méretű kerettel, amely a bemenetét képezi egy „fekete-fehér” kiértékelő-egység párnak (3.12. ábra).



3.12. ábra
Az előfeldolgozott ROI bejárása, és a kiértékelő-egység párok bemenetének előállítása

3.4.4 A képelemző és objektumfelismerő alrendszer végleges modellje

Az előzőleg leírtaknak megfelelően az alrendszer működésében a legfontosabb különbség az, hogy a beérkező képet csak egyszer használjuk fel a három küszöbölt kép előállításához, melyek közvetlen bemenetként szolgálnak a felismerő modulok számára (3.13. ábra).

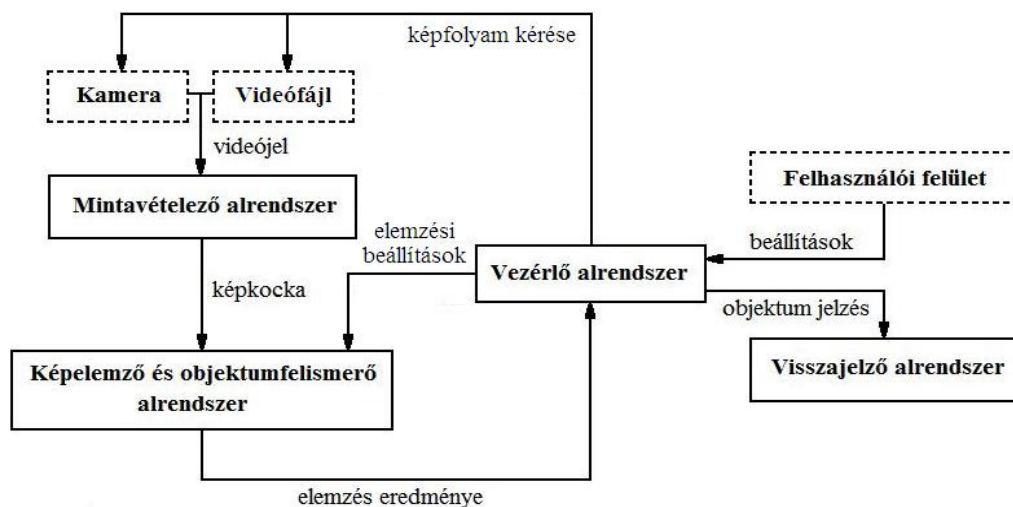


3.13. ábra

A képelemző és objektumfelismerő alrendszer végleges modellje
(a szaggatott vonal jelzi azt, hogy a harmadik felismerő modul (zöld lámpák) csak akkor fut le, ha az első modul (táblák) eredménye pozitív)

3.4.5 A teljes rendszer végleges modellje

A teljes rendszer modellje végeredményképpen annyiban változott, hogy a végleges verzióban lehetőség van a képelemző alrendszer finomhangolására, és a működés lehetséges előzetesen rögzített videófájlok esetén is (3.14. ábra).



3.14. ábra

A teljes rendszer végleges modellje

4. A rendszer megvalósítása

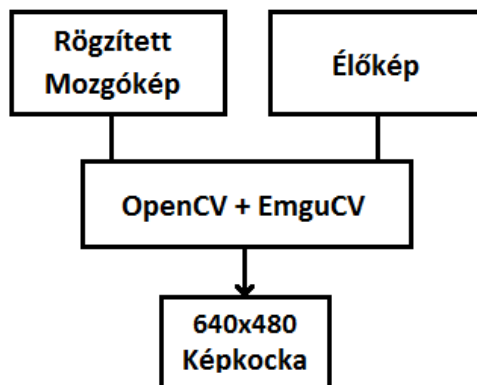
A projekt, mivel témájából következően leginkább az evolúciós fejlesztési módszertannak felel meg, a vázlatos tervezés, az egyes funkciók részletes megtervezése, az implementáció és a tesztelés nehezen elválasztható részfeladatok. Ebben a fejezetben igyekszünk összefoglalni mindazt, ami összességében meghatározza a szoftver egészének működését, kitérve az egyes alrendszerek funkciójára, az azokban alkalmazott technikák, algoritmusok, azok optimalizálási lehetőségeinek bemutatására, megemlítve a prototípusokban alkalmazott, de a végleges verzióból kikerült, vagy másmilyen formában alkalmazottakat is.

A rendszer megvalósítása során jelentős könnyebbséget hozott annak moduláris természete. A különböző lépések jól különálló folyamatként is leírhatóak, ennek mind a tesztelésben, mind az optimalizálásban való előnyös szerepe megkérdőjelezhetetlen. Hátránya ellenben az, hogy egy esetlegesen rosszul, vagy lassan működő modul a többi ráépülő részegységben is nem várt problémákat okozhat. A továbbiakban részegységekre bontva mutatjuk be a program megvalósítását.

4.1 A mintavételező modul

4.1.1 A modul megvalósítása

A bemenet alatt a feldolgozandó nyers forrásképet értjük. Ez lehet az élőkép, illetve egy előre rögzített felvétel aktuális képkockája. Egy olyan megvalósításra volt szükségünk, ami ezt egy uniformizált interfészen keresztül képes kezelni, a képfolyam formátumától és annak élő, vagy rögzített mivoltától függetlenül. Erre a feladatra használtuk fel az OpenCV függvénykönyvtárat, illetve az azt .NET alatt elérhetővé tévő EmguCV wrappert (4.1 ábra). Egy objektum metódusának segítségével így lekérhetővé vált a következő képkocka a forrástól függetlenül, így ezután a feladatunk már csak a bemenet megfelelő lejátszási sebességének, azaz a mintavételezési frekvenciának a biztosítása volt. A használt felbontás a már meghatározott 640x480-as VGA, a beérkező lejátszási sebesség pedig 30 képkocka másodpercenként (erre bár nem minden webkamera képes, nem jelent problémát, mivel az ezt biztosítani nem tudó eszközök esetén csak annyi történik, hogy ugyanazt a képkockát többször is lekéri a program).



4.1. ábra
A mintavételező modul felépítése

4.2 Az előfeldolgozó modul

4.2.1 A modul megvalósítása

A végleges megvalósításban az előfeldolgozó modul feladata a három küszöbölt bináris kép előállítás a fényviszonyoknak megfelelően. Végeredményképpen a [Shneier] által bemutatott RGB aránypáros küszöbölést alkalmazzuk, mivel ez szolgáltatta a legmegfelelőbb eredményt. A felhasznált feltétel a következő egyenlőtlenségek logikai „és” kapcsolata:

1, *Piros táblák és lámpák esetén (az utolsóra csak éjszakai körülmények között a piros objektumoknál van szükség, a városi közvilágítási náatriumgőz lámpák erős narancssárga fényének kiküszöbölése végett):*

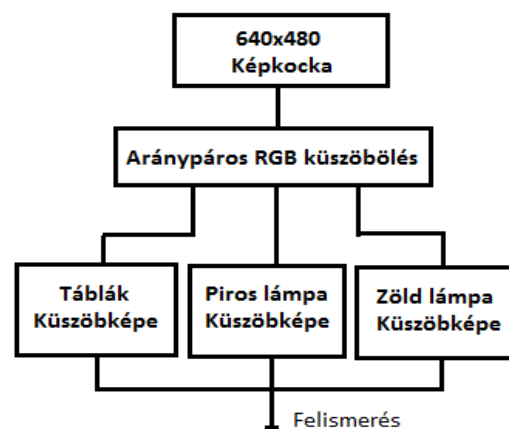
$$\frac{R}{G} \geq C_1 \quad \frac{R}{B} \geq C_2 \quad R \geq C_3 \quad \left(\frac{B}{G} \geq C_4 \right)$$

2, *Zöld lámpák esetén pedig:*

$$\frac{G}{R} \geq C_1 \quad \frac{G}{B} \geq C_2 \quad G \geq C_3$$

Az eredeti képből három különböző küszöbölt képre van szükség, három eltérő C_1 C_2 C_3 (C_4) konstans értékek mellett a megfelelő mennyiségű információ kinyeréséhez, mivel – ahogy azt már az előző fejezetben is leírtuk – mind a stop és elsőbbségadás kötelező, mind a piros lámpák és a zöld lámpák is eltérő aránypárokat igényelnek. A fényviszonyok megállapításához azt a módszert alkalmazzuk, hogy a forráskép pixelértékeit összegezzük, és egy előre definiált korláthoz igazítjuk. Ez bár első olvasatra túlságosan egyszerűnek tűnhet, de teljes mértékben megfelel a feladatra.

A küszöbölés végeredményeként három, ROI méretű bináris képet kapunk (mivel a kép többi részén felesleges a küszöbölést végrehajtani), ahol a feltételeknek megfelelő képpontokhoz fekete, a nem megfelelő képpontokhoz fehér színérték kerül hozzárendelésre. A következő, felismerő modulok bemeneteként szolgálnak ezen binarizált képek (4.2. ábra).



4.2. ábra
Az előfeldolgozó modul működése

A program egyik prototípusában egyértelműen látszik a három ugyanolyan módszerrel, de más paraméterekkel küszöbölt kép különbsége (4.3. ábra).



4.3. ábra

A forráskép három küszöbölt képe

(látszik, hogy ez eredmény közel sem tökéletes, és az is, hogy a zöld küszöbölés sokkal tisztább)

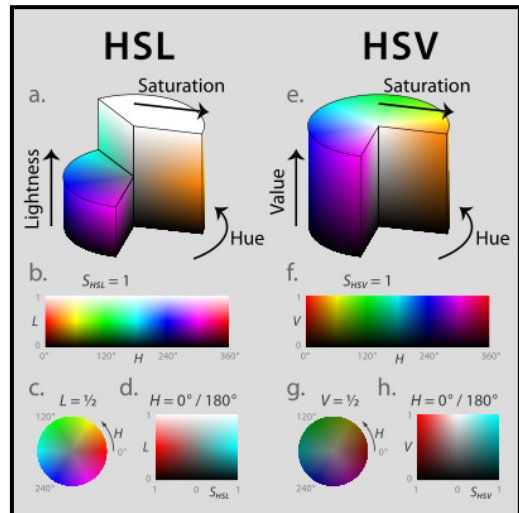
4.2.2 Optimalizálás

A három különálló képet érdemes egy cikluson belül elkészíteni, mivel mindhárom azonos képpontokból dolgozik, és kimenetük is azonos, ROI méretű. Az aránypáros küszöbölés arányok összehasonlítására alapuló mivoltának köszönhetően sok osztási művelettel járna minden egyes képpont meghatározása. Kihasználva azt, hogy az mind az osztó, mind az osztandó érték, csak 0 és 255 közötti egész értéket vehet fel, egy LUT (LookUp Table) készítésével megspórolható a futási időben történő rengeteg számításigényes osztási művelet. A LUT nem más, mint adott értékek előre kiszámítása egy olyan mátrixba, amelyből megfelelő indexeléssel sokkal gyorsabban kinyerhetők az indításkor előre kiszámított számításigényes adatok.

4.2.3 Elvetett ötletek

Mielőtt a végső verzióban megállapodtunk az RGB aránypáros küszöbölésnél, nagyon sok olyan módszer kipróbálásra került, amelyek végül nem bizonyultak megfelelőnek. Mivel a projekt célja a valós időben történő felismerés, így elsődleges szempont a modul sebessége volt, hiszen ha már az első részegység nem képes tartani a lépést az ideális másodpercenkénti 30 képkockával, a program nem lenne képes valós időben elegendő mennyiségű információt kinyerni a környezetből, és mivel minden kihagyott képkocka lényegesen rontana a projekt megbízhatóságán, ez semmiképpen sem megengedhető.

A HSL/HSV (Hue, Saturation, Lightness/Value) (4.4. ábra)¹ színtérre való konverzió gyakori eljárás a közlekedési táblafelismerések megvalósításánál, ugyanis különböző fényerősségű és telítettségű, de hasonló színezetű képpontok kiemelése eléggé egyszerű. Viszont sajnos a bonyolult konverziós eljárás még az OpenCV mély absztrakciós szinten megvalósított, metódusával sem bizonyult elegendően gyorsnak a valós időben való feldolgozásra, ráadásul az 1. fejezetben említett, a webkamerák automatikus fehéregyensúly beállítása miatt, az eredménye sem volt számottevően jobb az RGB arányos küszöbölésnél.

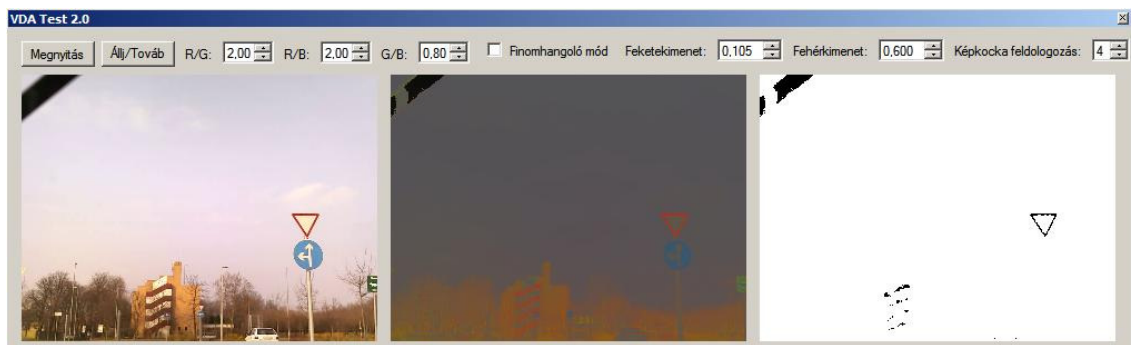


4.4. ábra¹
A HSL/HSV színtér

Ennél sokkal gyorsabb konverzió az ún. RGB normalizálás. Ennek lényege a színek kiemelése az alábbi összefüggésekkel:

$$R := \frac{R \times 255}{\sqrt{R^2 \times G^2 \times B^2}} \quad G := \frac{G \times 255}{\sqrt{R^2 \times G^2 \times B^2}} \quad B := \frac{B \times 255}{\sqrt{R^2 \times G^2 \times B^2}}$$

Egy kezdeti verzióban helyet is kapott ennek az implementálása (4.5 ábra), amely eleinte szép eredményekkel kecsegtetett, azonban nem várt anomáliák tűntek fel a küszöbölt képen. Az élénk piros táblákat valóban felerősítette, azonban más, sokkal kevésbé vörös domináns objektumokat is túlerősített, amelyek több kárt okoztak, mint hasznat, így ezt a megoldást is elvetettük.

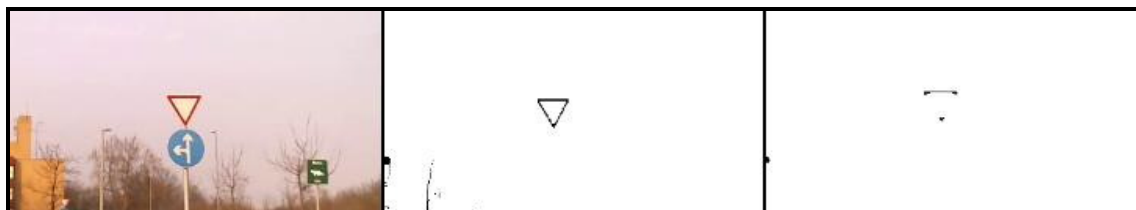


4.5. ábra
Az RGB reprezentáció színkiemelése
(ezen a képkockán éppen jól funkcionált)

Az RGB aránypáros küszöbölés mellett egy másik, dinamikus küszöbölés is kipróbálásra került, melynek működését [Yang] írja le. Azonban ez, pont dinamikus mivolta miatt, a piros lámpák képbe kerülése esetén minden más objektumot kiszűrt, illetve rosszabb esetben akár egy épület is képes volt elnyomni a táblákat, így hiába adott jó eredményeket az esetek túlnyomó többségében, túl megbízhatatlannak bizonyult helyenként ahhoz, hogy ez kerüljön a végleges rendszerbe.

¹ Wikipédia ábra a HSL/HSV színtéréről, upload.wikimedia.org/wikipedia/commons/thumb/a/a0/Hsl-hsv_models.svg/400px-Hsl-hsv_models.svg.png

Miután megállapodtunk a végleges küszöbölés mellett, annak hibáinak kiszűrésére többféle módszert is kipróbáltunk. Legelőször egy Gauss-simítást¹ próbáltunk ki az eredeti képen, hátha így a kisebb zajok nem fognak előtűnni, ami működött is, de sajnos a táblák élei is teljesen elmosódtak (4.6. ábra). Következő lehetőségként egy mediánszűrést próbáltunk ki a már küszöbölt bináris képen, de sajnos akkora maszkkal, amely képes volt szűrni a zajokat, szintén tönkretette a felismerni kívánt alakzatokat. Végül egy egyszerű dilatációs módszer implementálásával próbálkoztunk, hogy sokkal karakteresebbé váljanak a jól különálló objektumok. Ezt a szerepet el is látta ez a módszer, azonban a küszöbölésen átjutó kisebb zajokat is annyira felerősítette, hogy esetenként már ellehetetlenítette a felismerést.



4.6. ábra

A forráskép, az abból küszöbölt bináris kép, és a Gauss-életlenített¹ forráskép küszöbölt képe

4.3 A felismerő modul

A felismerő modul tartalmazza a rendszer legfontosabb, legtöbb hangolást igénylő, és a végső verziót tekintve leglassabb és legkritikusabb részét. Az irodalomkutatás során egyértelmű volt, hogy szinte minden rendszer ezen végső lépésként vagy valamilyen neurális hálózatot, vagy eltárolt mintákkal való valamilyen (pl. kereszt-korrelációs) összehasonlítási algoritmust alkalmaz.

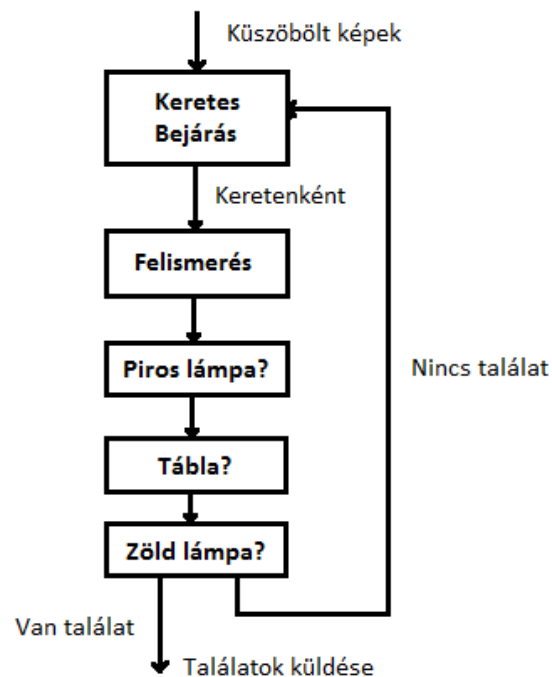
Mivel a legjobb eredményeket a tesztelések során azzal értük el, hogy az érdekelt régiót a lehető leggyorsabban előfeldolgozva, egy kerettel bejárva minden elegendően sok pozitív képpontot tartalmazó részképet átadtunk a felismerő objektumnak, így annak működését, felépítését is végső soron ehhez igazítottuk. Előre vetítve egy olyan kiértékelő egység-párt alkalmazunk, amely működésében a neurális hálók egy neuronjához hasonlít, de a súlyozása egy segédprogram által konvertált, előre definiált minták alapján történik, és nem alkot komplex neurális hálózatot.

A folyamatot tekintve tehát első lépésként a küszöbölő modultól vett képet, egy adott méretű kerettel be kell járni, és minden alapvetően érvényeset átadni a kiértékelő egységnek. Azt, hogy mi alapvetően érvényes egyszerűen az dönti el, hogy tartalmaz-e elegendő mennyiségű pozitív képpontot az adott objektumcsoportra vonatkoztatva. Amennyiben nem, felesleges átadni azt felismerésre. A keret méretét és ezáltal az egység bemeneteinek számát tapasztalataink alapján 40x40-esnek választottuk, mivel minden célobjektum egy ekkora négyzetbe foglalható megfelelő – nem túl kis – távolságból a 640x480-as forrásképen. Természetesen ez valamennyire változhat az alkalmazott webkamera beállított látószögének függvényében, de ez a változás a figyelmeztetési távolság arányában csekély, így nem okoz gondot.

¹ Gauss-simítás: Gauss függvény által meghatározott értékekkel történő szűrés, mely a forrásképet életleníti, homályosabbá, kevésbé zajossá teszi

4.3.1 A régió bejárása

Ez az egyszerű folyamat egyetlen óriási problémát rejt: valós időben kell működnie. Egy kétdimenziós kép bejárása egy nála kisebb kétdimenziós kerettel négy ciklussal valósítható meg. A számítási igény hatalmas tud lenni, ezért szükségessé vált ennek a műveletnek egy külön futási szála való helyezése. Erre a C# egy egyszerű megoldást nyújt. A külön szál nagy előnye még, hogy egy előre rögzített mozgókép esetén nem lassítja annak visszajátszását, így teljesen valós körülmények között tud működni a program, így az eredetileg, egy szálon előállt különbség az élő és rögzített kép között semmissé vált. A kiértékelő objektumok meghívásának sorrendjét a 4.7. ábra mutatja (az egyszerűség, és a prioritási sorrend hangsúlyozása végett nem jelöltük külön azt, hogy piros lámpa esetén a többi ellenőrzésre nem kerül sor).



4.7. ábra
A felismerő modul működése

A projekt széleskörű használhatóságához elengedhetetlen a régebbi, gyengébb processzorok esetén történő megbízható futás. Mivel a processzor sebességén múlik a feldolgozott képkockák száma, többször jelentős átvizsgáláson esett át a projekt ezen a kritikus részen, és több optimalizálás is történt, míg megszületett a végleges megoldás.

Az első átvizsgálás során több változó is típusváltáson esett át, hogy a lehető legkisebb helyet foglalják, illetve sokszor végrehajtott műveleteknél gyorsulást érjünk el. A kiértékelő egységeknek való adatátvitelnél a manuálisan kasztolt *Clone()* eljárás helyett az *Array.Copyto()* hatékonyabbnak bizonyult, mivel az ezt automatikusan végzi el. Emellett az első tesztek bizonyították, hogy a felismerő modul ROI bejáró kereténél történő folyamatos nagymértékű értékadás az egyik legnagyobb rendszerigényű művelet. Ez a keret csak 0 illetve 1 értéket tartalmazhat, ezért ha a keretet minden bejárás előtt egy sokkal hatékonyabb eljárással nullázzuk, minden 0 értékadást megspórolhatunk. Erre az *Array.Clear()* metódus bizonyult a leggyorsabbnak, ami a C# tömb osztályának egy beépített, nagyon effektív eljárása.

Az értékadások jelentős mértékű csökkentése meghozta eredményét, a felismerés gyorsabbá vált, a sűrűbb mintavételezés megbízhatóbb felismeréssel járt. Azonban a felismerés továbbra se érte el az általunk kiszabott elégséges teljesítményt, de volt még egy kihasználható lehetőség, amivel nagymértékben csökkenthető lett az értékadások száma.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

22	23	24	25	32	33	34	35	42	43	44	45	52	53	54	55
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

4.8. ábra
Az n . lépés és a hozzátartozó feldolgozó vektor

A keretet egy $kerethossz \times kerethossz$ méretű egydimenziós tömbben tároljuk (4.8. ábra). Átvizsgálás után azonban a keret hosszának csak egy töredékével mozdul odébb vízszintes irányban ez a keret. Ez nagy mennyiségű redundáns adatot jelent n és $n+1$ lépés között, feltéve, hogy függőlegesen nem mozdult a keret, illetve nem értünk egy sor végére. Azonban mivel a pozíciója is számít ezeknek az adatoknak, szükségünk van az adat $kerethossz$ méretű eltolására. Ezt a leggyorsabban a C# `Array.Copy()` eljárása biztosítja managed környezetben. Ezután csak a bejárt új $lépéshossz$ szélességű adatmennyiséget kell beírni (4.9. ábra). Az eddigi $kerethossz \times kerethossz$ mennyiségű értékadás $lépéshossz \times kerethossz$ -ra csökkent, ami a jelenlegi implementációban ($kerethossz = 40$, $lépéshossz = 4$) tized annyi értékadást jelent, és egy viszonylag gyors másoló eljárást eredményez.

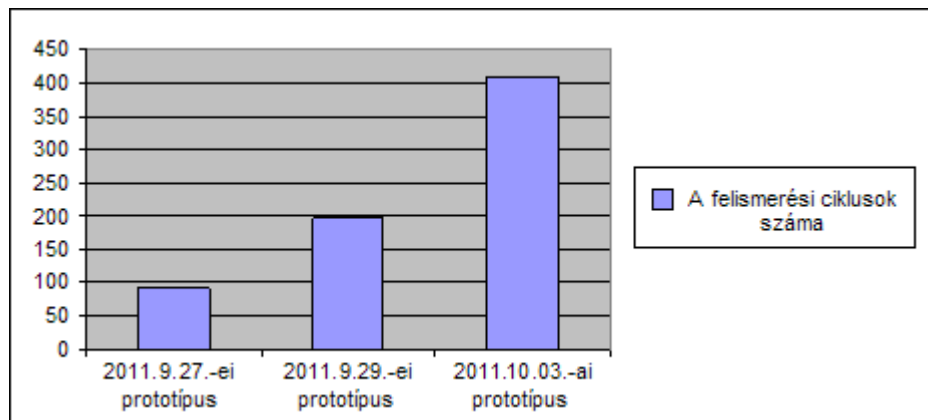
0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

22	23	24	25	26	27	34	35	36	37	44	45	46	47	54	55	56	57
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

←-----

4.9. ábra
Az $n+1$. lépés: csak a vörös értékeket kell beírni a többi redundáns

A teszteredmények alapján körülbelül 400%-os gyorsulást sikerült elérnünk (4.10. ábra), így már egy napjainkban gyengébbnek minősülő processzoron is elégséges mennyiségű felismerési ciklus fut le ahhoz, hogy élő képen is működőképes legyen a program.



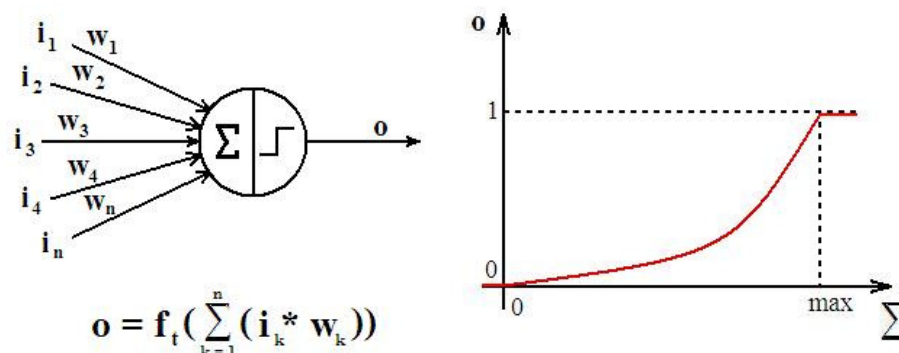
4.10. ábra

A gyorsulás mértéke egy példa felvételen a két gyorsítási átvizsgálás után

Még egy fontos tapasztalatunk az, hogy többszörösen beágyazott ciklusok esetén, az általános elvekkel szemben máig kíváncsú a *goto* (illetve az annak megfelelő ciklusmegszakító *break*) utasítás használata, ugyanis egy esetleges tábla találat esetén egyből odaugorhatunk a zöld lámpa ellenőrzéséhez, ezzel nagyon sok fölösleges ciklusfeltétel-kiértékelést megspórolva.

4.3.2 A kiértékelő egységek működése

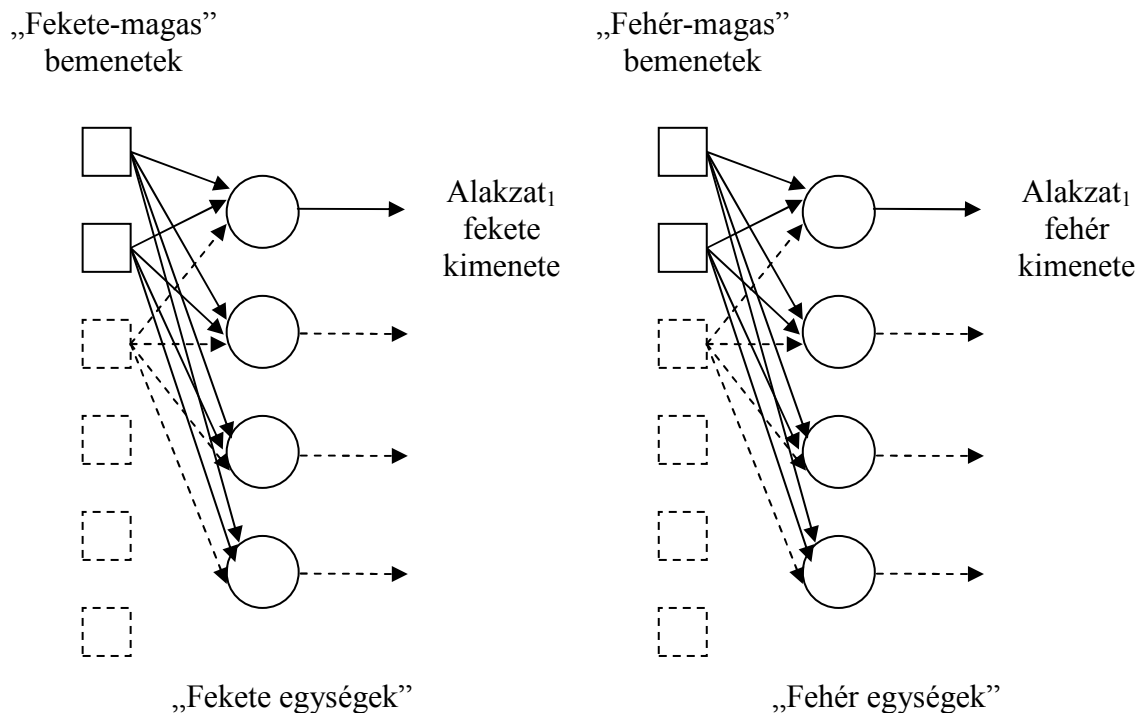
A felismerés alapja a már említésre került kiértékelő egységek. Ezek működése a mesterséges neurális hálózatok objektumaihoz, az egyes neuronokéhoz hasonlít, amely megadott számú bemenetet meghatározott súlyozással összegez, majd ennek eredményéhez tartozó értéket - melyet egy transzferfüggvény határoz meg – adja a kimeneten (4.11. ábra). Esetünkben az 1600 bemenetet az egyes 40x40-es bináris részképek határozzák meg a bal felső sarokból kiindulva sorfolytonosan, még hozzá úgy, hogy a pozitív képpontok értéke 1, a negatívaké 0, és a súlyok a $[0;1]$ zárt intervallumból vehetnek fel értékeket. Az általunk a projektben alkalmazott transzferfüggvény egy transzformált, korlátos másodfokú polinom függvény.



4.11. ábra

A mesterséges neuronok működése és a szoftverben használt transzferfüggvény közelítő ábrázolása

A felismerés működésének lényege az, hogy ha egy alakzathoz tartozó mindkét – egy „fekete-magas” és egy „fehér-magas” – kiértékelő egység kimenete elér egy bizonyos küszöbértéket – ezt természetesen célobjektumonként hangolni kell –, akkor az nagy valószínűséggel a keresett alakzat. Ahogy azt a 4.12. ábrán is láthatjuk egyik egység bemenetén a fekete pixelek, azaz a pozitív képpontok jelennek meg magas értéken és a fehérek alacsonyan, míg a másikon pont fordítva, és a súlyai is invertáltak. Ennek azért van jelentősége, mert egy nagyon sok feketét tartalmazó részkép magas kimenetet eredményezne a fekete egységen – mivel tartalmazza a felismerendő fekete alakzatot is – így az egység tévesen helyesnek fogadná el az eredményt. Ezzel a megoldással azonban ez a probléma kiküszöbölhető.



4.12. ábra
A kiértékelés elvi működése

Bár látszik a hasonlóság, a hagyományos neurális paradigmával szemben azonban ezeket az egységeket nem tanítjuk, hanem a súlymátrixaikat manuálisan állítjuk be, „festjük”, így egy minta alakzatot felvázolva. Erre a célra készítettünk egy külön segédprogramot (4.13. ábra), amely átkonvertál egy szürkeárnyaltos képet az elvárt súlyformátumba. Ennek pontos funkciója az, hogy a maszk-képet sorfolytonosan bejárva, pixelenként 255-ből vonja ki annak intenzitását, és az eredményt ossza el 255-tel. Ezzel a [0;1] zárt intervallumba eső lebegőpontos értékeket kap, amelyeket folytonosan kiír egy bináris fájlba. A program kiértékelő egységei ezt a fájlt olvassák be létrejöttükkor – a „fehér-magas” neuronok természetesen invertálva –, és ezt használják fel a detektálás során. Minden egyes felismerendő alakzathoz szükséges annak maszkját „megfesteni”, és elmenteni az azt leíró fájlba.



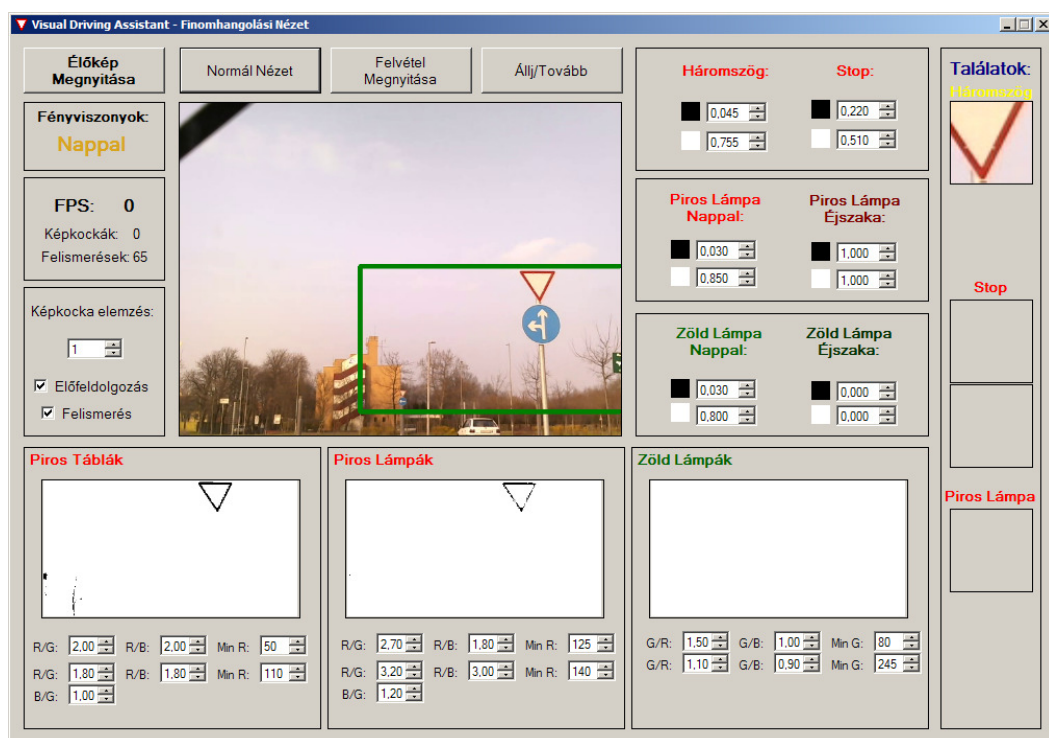
4.13. ábra
A Súlymátrix Konverter
segédprogram

4.3.3 Az eredmények értékelése

Miután tehát meghatározott kerettel bejárjuk a képet, és minden egyes részképet átadunk a kiértékelő-egység pároknak, visszakapjuk azon pozíciók koordinátáit, ahol valószínűleg felismerendő objektum van. Ez ebben a formában azonban még nem elég.

Egyrésztől mindhárom bináris képet végig bejárni fölösleges és lassú, mivel például zöld lámpát csak akkor kell ellenőrizni, amikor a program talált már a képkockán háromszög vagy stop táblát – ráadásul csupán a felismert objektum alatt egy bizonyos régióban számít érvénytelenítőnek –, de egy piros lámpa detektálása esetén is szükségtelen tovább folytatni a keresést a táblák után, mivel mindenképpen a tilos jelzés élvez prioritást. Ennek okán a bejárás során lefutó felismerések függenek egymástól, egyik pozitív eredménye befolyásolja a másik végrehajtását oly módon, hogy az vagy csak akkor fut le, hogyha nincsen azt érvénytelenítő eredmény, azaz a piros lámpák esetén szükségtelen táblát keresni, de egy detektált háromszög vagy stop tábla esetén is fölösleges a másikat ellenőrizni, mivel nincs olyan közlekedési helyzet, amikor mindkettő egyszerre érvényes lehetne. Zöld lámpák keresésének szüksége pedig csak detektált táblák esetén merül fel. Így jelentősen csökkenthető a futási idő, legalábbis kettő vagy több olyan képkocka között, amelyek mindegyike tartalmazza ugyanazt a felismerendő objektumot (ez fontos és mindenképpen előnyös hozomány a kiértékeléshez).

A kiértékelés valójában nagyon egyszerű ötleten alapszik. Mivel csupán kijelezni és végleges kimenetként kezelni képkockánként a felismert objektumokat nem elég megbízható – sokszor előfordulhat „bevillanó” téves detektálás – ezért azt csak akkor fogadjuk el, hogyha a megelőző képkockákon is történt azonos típusú objektum detektálása legalább n (kategória-függő) alkalommal. Csak ez kerülhet ki a végleges kimenetre (4.14. ábra).



4.14. ábra
A végleges verzió finomhangolási nézete
(a találatjelző panelen látszik a kijövő eredmény)

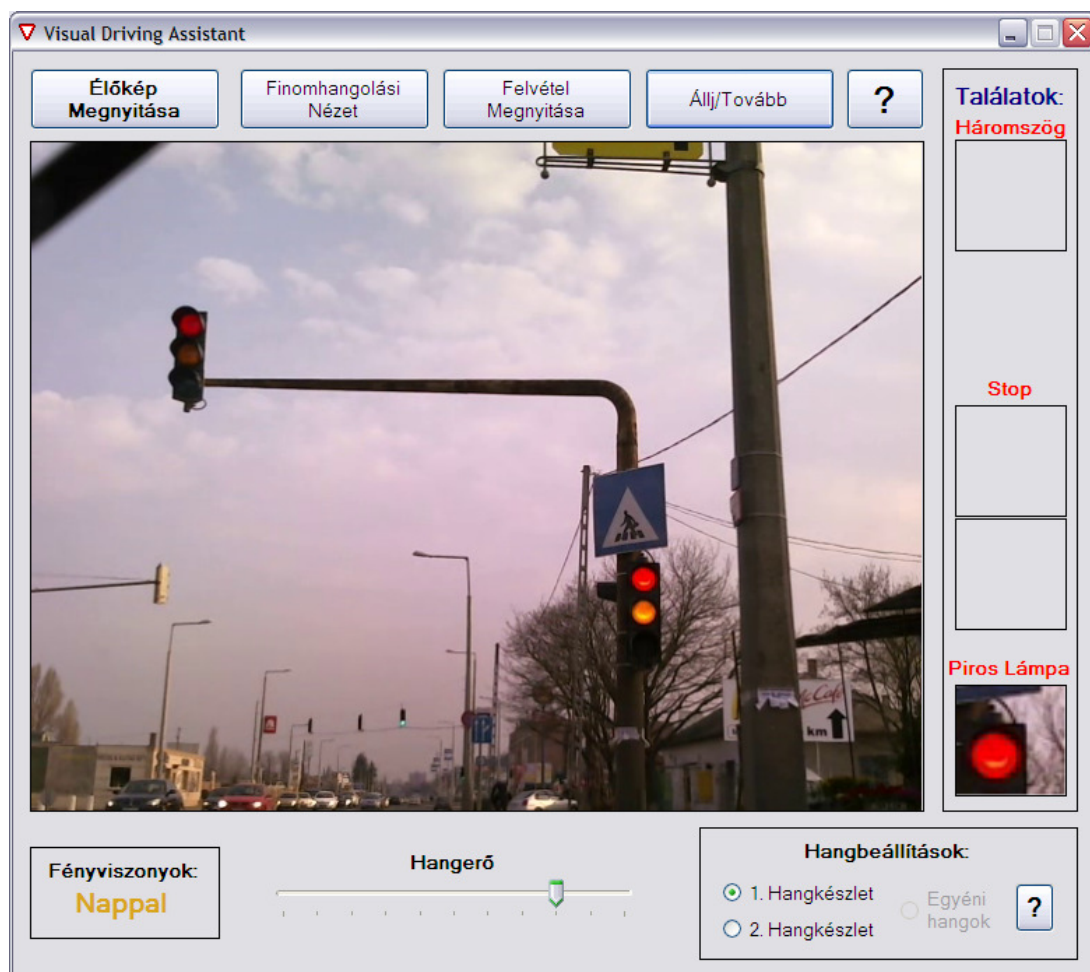
4.4 A felhasználói felület

A végleges felhasználó felület kialakításánál – lévén hogy a program célja futás közben a megfelelő hangfájlok lejátszása – az egyszerűsége törekedtünk. Bár a kijelző figyelése nem szükséges, sőt kifejezetten tilos vezetés közben, a bejövő kép, a szoftver által meghatározott fényviszonyok, és a találatjelző ablak megjelenítésre kerül, de kizárólag parkoló helyzetben történő kalibrálás, és esetleg menet közben egy utas általi megfigyelés céljából.

A felhasználónak lehetősége van beállítani a szükséges hangerőt, valamint kiválasztani azt a hangjelzés-készletet, amelyik számára a legmegfelelőbb. A program két beépített ilyen készlettel rendelkezik, de amennyiben a felhasználó saját jelzéseket kíván használni, arra is lehetősége van azok bemásolásával a *User Sounds* könyvtárba. Ezek megfelelő elnevezéséről és formátumáról egy üzenetablakban kaphat tájékoztatást.

Végezetül úgy döntöttünk, hogy a végső verzióban maradjon meg a finomhangolási nézet, illetve a rögzített videók lejátszásának lehetősége a felhasználó számára is, így szükség esetén módosíthatja és ellenőrizheti a működési paramétereket. A két nézet között egy gomb segítségével lehet egyszerűen átváltani.

A végleges felhasználói felületet a 4.15. ábrán láthatjuk.



4.15. ábra
A végleges felhasználói felület Windows XP x86 alatt

5. Tesztelés

5.1 A tesztelések szerepe

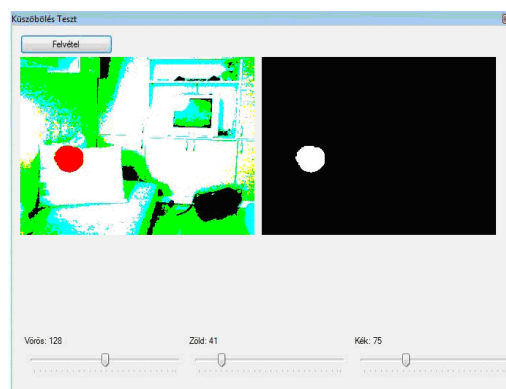
Tesztelés alatt a projekt esetében elsősorban nem annyira a hibakeresés és javítás hagyományos programozás-technikai fogalmát értjük, mint inkább a modulok közötti átvitel, a különféle bemenetekhez tartozó megfelelő kimenetek optimális értékeinek vizsgálatát, valamint az ezekhez tartozó paraméterek megfelelő behangolását. Természetesen valódi programhibák is előfordultak a szoftver készítése közben, ami miatt szükség volt klasszikus értelemben vett tesztelésre is, de ez csupán töredék időt emésztett fel. A teljes fejlesztés legnagyobb részét ez a fajta „tesztelés-hangolás” tette ki, amely végigkísérte az egész folyamatot az elejétől a végéig.

5.1.1 A folyamatos tesztelés igénye

A projekt kezdetétől fogva egyértelmű volt, hogy az csak akkor lehet eredményes, ha a jól elszeparált modulokat egyenként folyamatos vizsgálatoknak vetjük alá mind a hatékonyság, mind a futási sebesség szempontjából, azokat külön-külön a lehető legoptimálisabbra készítjük el. Az egymásra épülő, egymástól függő egységek egyértelmű hátránya az, hogy ha már a legelső hibásan, vagy túl lassan működik, annak nem megfelelő kimenete továbbterjed a ráépülő többi modulra is, ráadásul jelentős futási időt is vonhat el azoktól. Mindezekon felül a sok apró hiba – ideértve a magas időigényt is – összeadódik a kimenetek során, amelyek szinte exponenciálisan képesek csökkenteni a megbízható, megfelelő időben megjelenő kimenet előállításának esélyét. Lényegében tehát nagy szükség volt a folyamatos és gyakran hosszadalmas tesztelésekre, azok eredményeinek elemzésére, és a modul megfelelő áthangolására már akkor is, amikor a teljes rendszernek még csupán egy töredéke készült el.

5.1.2 A részeredmények felhasználása

A kezdeti verzió fejlesztése – mint azt a rendszer tervezése fejezetben már említettük – már azelőtt elkezdődhetett mielőtt rögzítettük volna a tesztfelvételeket, ugyanis a moduláris felépítésnek köszönhetően lehetőségünk volt bizonyos részegységek előzetes megtervezésére és implementálására. A videófájlok elkészültének idejére már az előfeldolgozó egység kezdeti megvalósítása rendelkezésünkre állt, ahogy ezt az 5.1.-es ábrán is láthatjuk.

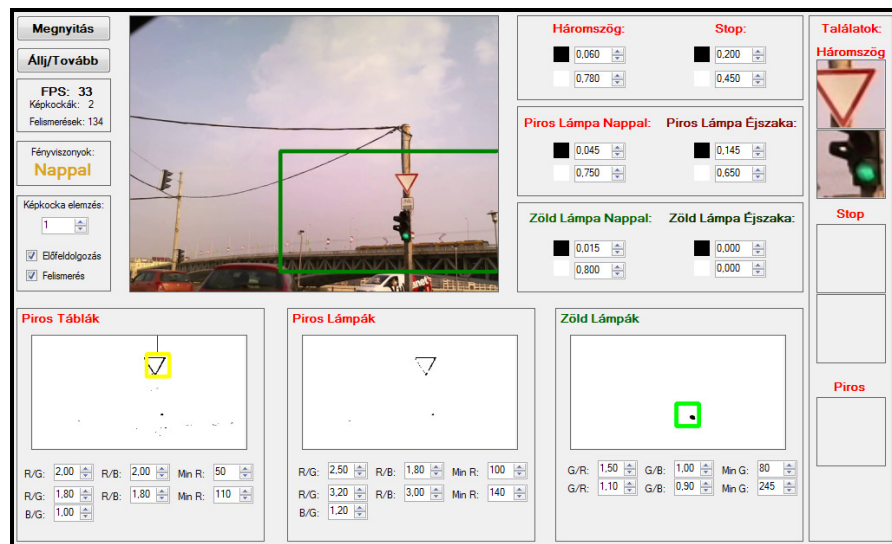


5.1. ábra
Előfeldolgozás a kezdeti verzióban
(jelen esetben Windows 7 x64 alatt)

Az átmeneti verziók segítségével jól elszigetelve tudtuk ellenőrizni az egyes modulok kimeneteit, és ennek megfelelően a részeredményeket a további fejlesztések tükrében értékelni. Az általunk követett evolúciós fejlesztési modell sajátossága az is, hogy meglehetősen sokszor kényszerültünk rá bizonyos egységek újratervezésére és újrainplementálására azok teljesítményének fényében. Természetesen az efféle beavatkozás mindig újabb teszteléseket vont maga után.

5.2 A tesztkörnyezet

A tesztkörnyezetet valójában maguk az egyes átmeneti verziók jelentették. Az evolúciós fejlesztésnél ezek a prototípusok formálták folyamatosan a teljes rendszertervet azáltal, hogy lehetőséget biztosítottak a készítéskor felmerülő, elméletben tökéletesen megfelelő, de a gyakorlati megvalósítás során komoly problémákat előhozó módszerek gyors kipróbálására. Nélkülözhetetlen volt egy, a szoftver működését részletesen ismerő felhasználó részére – így saját magunk számára is – egy hatékony és átlátható kezelőfelület létrehozása. Az így megszületett finomhangolási felületen (5.2. ábra) fellelhetőek a képelemző és objektumfelismerő alrendszer rendszertervben felvázolt moduljai, és az azokat meghatározó pontos hangolást igénylő paramétereinek összessége. Mivel a fejezet későbbi részében bemutatásra kerülő teszteredmények is ebben a környezetben jöttek létre, így ezt részletesebben is bemutatjuk.

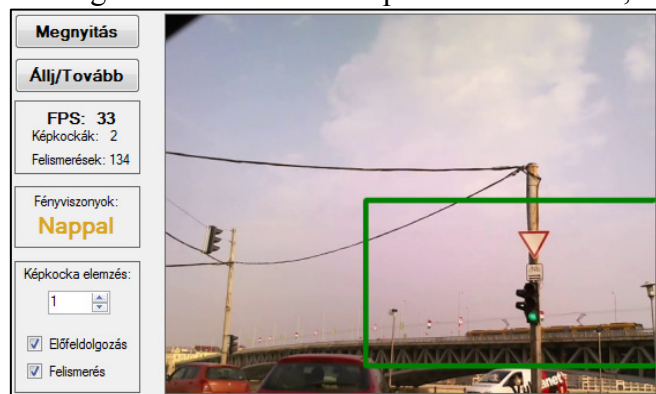


5.2. ábra

A Visual Driving Assistant finomhangolási felülete egy kései átmeneti verzióban

5.2.1 A bemeneti egység

A felület ezen része (5.3. ábra) a bejövő képfolyamot jeleníti meg. Itt van lehetőségünk a felvétel megnyitására – a végső verzióban az élőkép behozatalára is –, és itt látszanak olyan egyéb adatok, mint például a másodpercenként megjelenített, illetve felismerésre elküldött képkockák száma, valamint az aktuális fényviszonyok megállapításának eredménye. Külön a felismerést illetve az előfeldolgozást is ki lehet kapcsolni, ezzel ellenőrizve az egyes egységek sebességét. A „Képkocka elemzés” számlálójával csak minden n -edik beérkező képkockát dolgoz fel a program.

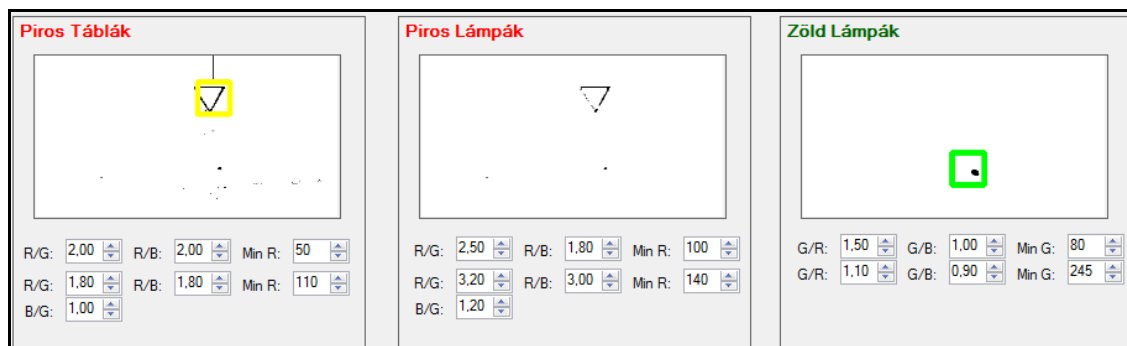


5.3. ábra

A bemenet eredményének megjelenítése
(a zöld téglalap a ROI-t jelöli)

5.2.2 Az előfeldolgozó egység

Ez a rész (5.4 ábra) az előfeldolgozó modul – amely a bemeneti egység aktuális képkockájának érdekelt régiójából állítja elő a három bináris képet – paramétereinek beállítását teszi lehetővé, és megjeleníti az eredményeket. Mindegyik képnél egyszerűen, akár futás közben is állíthatóak a hozzájuk tartozó aránypárok határértékei. A megjelenítés által azonnali visszajelzés kapható a változtatás hatásairól.



5.4. ábra

A program előfeldolgozást vizualizáló része
(ebben a verzióban a program által megjelölésre kerülnek a felismert objektumok)

5.2.3 A felismerő egység

A felismerő egységhez eleinte nem tartozott külön vizuális megjelenítés mivel mind az előfeldolgozott képeken, mind a bemeneti képen megjelenítésre kerültek a találatok. Pontos finomhangolási igénye miatt azonban a különböző küszöbértékek szintén futás közben állíthatóak be (5.5. ábra), így ez egy új, pontosabb súlyfájl esetén (5.6. ábra) jelentősen meggyorsította az újrahangolási folyamatot. Helyet kapott még egy, a felismerést visszaigazoló felület a program jobb szélén, ahol az aktuális legutolsó találat kategóriánként megjelenik, és az azt semlegesítő zöld lámpa is – persze csak ha van ilyen – az adott táblához tartozó csoportban kerül ábrázolásra.



5.5. ábra

A felismerő modulhoz tartozó
beállítási és találati felület



5.6. ábra

A végleges súlymátrix fájlok vizualizálva balról jobbra az alábbi sorrendben:
Háromszög, piros lámpa este, STOP, piros lámpa nappal,
zöld lámpa nappal, zöld lámpa este

5.3 A tesztelés folyamata

A tesztelések során az adott bejövő kép eseményeit kellett figyelembe vennünk, ugyanis ezek összessége határozza meg a program helyes működését. Először is azt kell ellenőrizni, hogy felismerésre került-e a specifikációban meghatározott figyelmeztető jelzés. Ha ez nem történt meg, egységenként kivizsgálásra kerültek a paraméterek. Az előfeldolgozás hibája a megjelenített bináris képeken ránézésre is megállapítható. Amennyiben ott nem találtunk problémát, akkor a felismerő súlyértékei szorultak hangolásra. Amennyiben megfelelő hangolás után a jelzés felismerésre került, akkor megoldódott a probléma, azonban ha így se történt felismerés, akkor vagy a feldolgozás lassúsága volt okolható, vagy a súlymátrix szorult javításra. A feldolgozási sebességből fakadó hibák a program késői verzióiban már az előző fejezetben tárgyalt gyorsításoknak köszönhetően ki lettek küszöbölve, és a súlymátrixok is többször újrarajzolásra kerültek. Hosszas finomhangolás és a végleges verzió implementálása után már készen álltunk arra a feladatra, hogy a tesztek eredményeit megvizsgáljuk, és konklúziót vonhassunk le a projekt eredményességét illetően.

5.4 Teszteredmények

A tesztek eredményeinek táblázatában az alábbi szavak alatt a következőket értjük:

- Felvétel: az általunk készített videófelvétel sorszáma
- Ciklusok: a lefutott felismerési ciklusok száma a felvételen
- Felismerendő: a felvételen megtalálható, és megfelelő távolságban lévő célobjektumok száma (képkockánként összegezve)
- Felismert: a célobjektumok közül a program által helyesen felismertek mennyisége (képkockánként összegezve)
- Hibás: a téves felismerések száma, a jelzett területen nincs, vagy nem a jelzett célobjektum van (képkockánként összegezve)
- Hatásfok: a felismerendő és felismert objektumok aránya
- Hibaszázalék: [Shneider] méréseinek bemutatását alapul véve, a felismert tényleges objektumokhoz képest a hibás felismerések aránya.

5.4.1 Délben készült felvételek eredményei

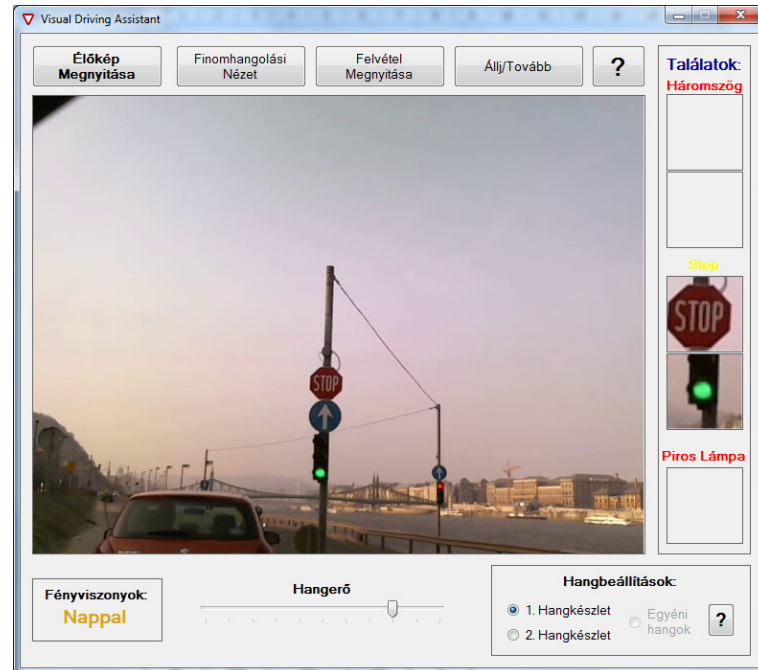
Az 1. táblázaton lévő felvételek 13:00-14:00 között készültek jó látási körülmények között. Nagyobb sebességű utak jellemzik, így ideális volt gyorsabb városi közlekedési tempó mellett történő felismerések tesztelésére (5.7. ábra).

Felvétel	Ciklusok	Felismerendő	Felismert	Hibás
5	386	27	27	0
6	425	22	22	0
7	335	0	0	0
10	393	0	0	0
14	505	0	0	11
19	272	4	0	0
21	254	30	30	0

Összesen: 83-ból:
79 felismerve, 11 hiba
Hatásfok: 95,1%
Hibaszázalék: 13,9%

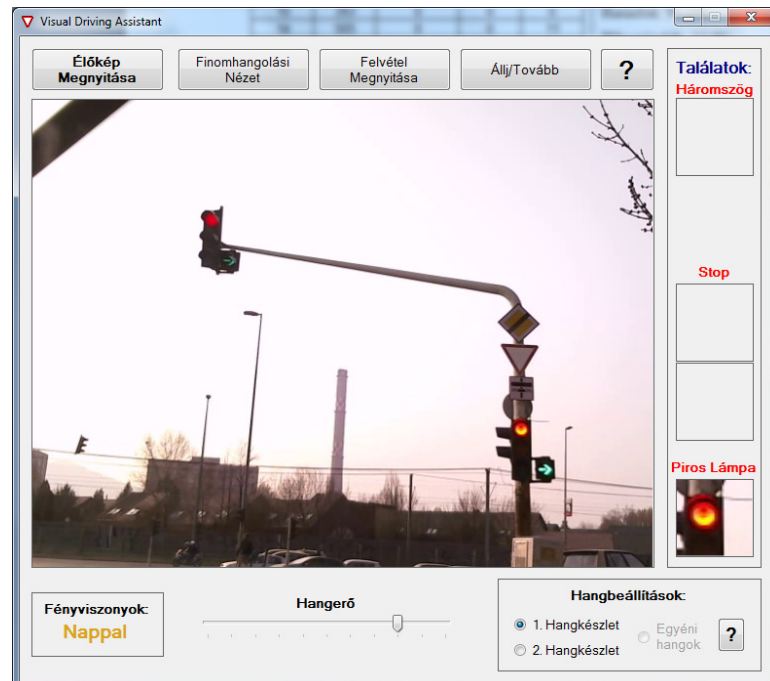
1. táblázat
Déli felvételek kiértékelése

A hatásfokon látszik, hogy nappali körülmények között a program felismerési aránya megfelelően magas (5.8. ábra). A hibaszázalék egyetlen felvételtől származik, a 14-es felvétel piros-fehér csíkos magasságmérő oszlop piros lámpának hitt téves érzékeléséből.



5.7. ábra

A jó felismerések egyike délben, a STOP táblát semlegesíti az alatta található zöld lámpa



5.8. ábra

Jól megfigyelhető a piros lámpák egyenletlen színe, amit a súlymátrix létrehozásakor is figyelembe vettünk

5.4.2 Kora délután készült felvételek eredményei

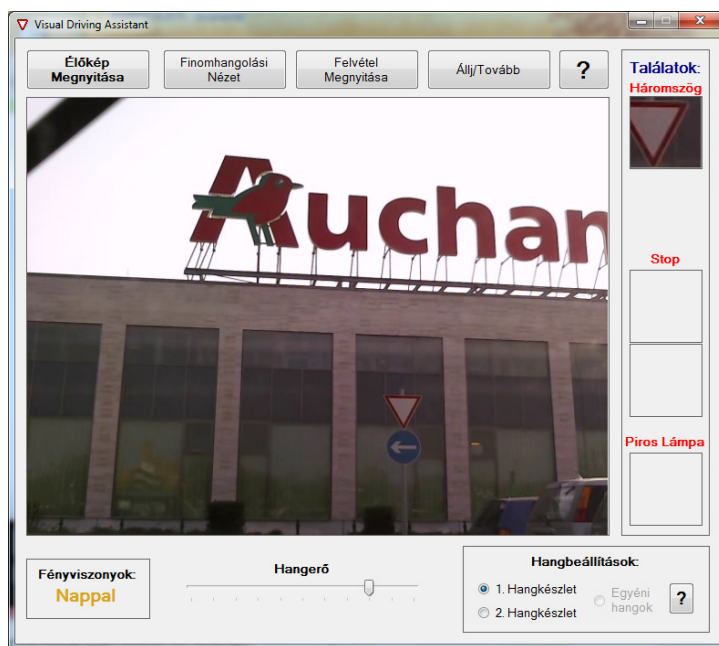
A 2. táblázaton megtalálható kora délutáni felvételek 14:00 és 15:00 között készültek. A déli felvételekhez képest lassabb tempójúak, így hosszabb ideig képen maradó felismerendő objektumok, és ebből adódóan erősebb hibalehetőségek is jellemzik. A napszak azonban továbbra is kedvező.

Felvétel	Ciklusok	Felismerendő	Felismert	Hibás
22	531	22	19	0
23	174	6	6	0
24	209	4	4	0
26	155	8	8	0
28	392	0	0	13
32	94	0	0	1
36	555	10	10	0
42	156	5	5	1
46	329	27	27	0
47	622	5	2	1
48	152	3	3	0
49	425	8	8	0

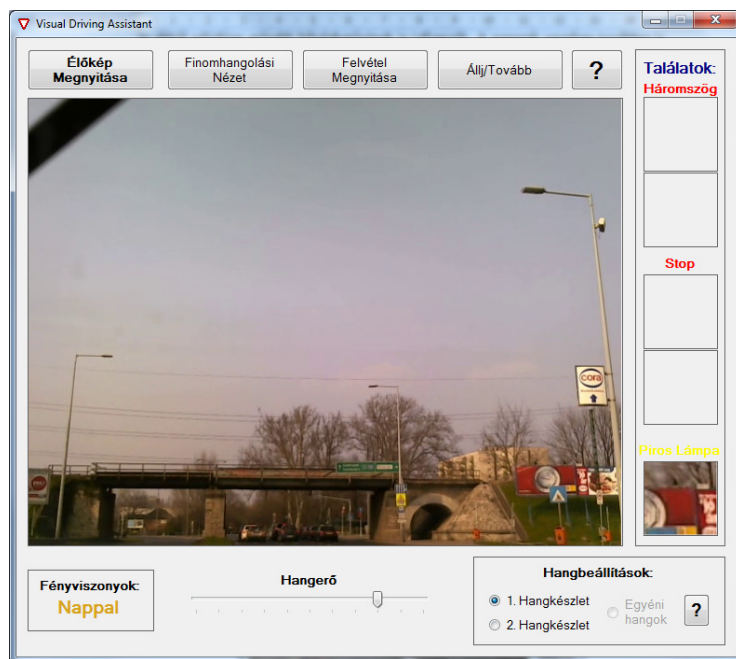
Összesen: 98-ból:
92 felismerve, 16 hiba
Hatásfok: 93,8%
Hibaszázalék: 17,3%

2. táblázat
Kora délutáni felvételek kiértékelése

A program itt is hasonlóan magas hatásfokkal dolgozik (5.9. ábra), de sajnos itt már több felvételen elszórtan is keletkeztek hibák, amit a nappali környezetben sűrűn előforduló vöröses épületek, reklámtáblák okoztak. A 16 hibából 13-at így is a 28-as felvételen egy nagyon rossz pozícióban lévő élenkpiros reklámtábla (5.10. ábra) okozta, amit táblának hitt a rendszer és a hibák 81%-át teszi ki összességében.



5.9. ábra
A sötétebb háttér se okoz problémát jó fényviszonyok között



5.10. ábra

Az élénk piros négyzetes alakú objektumok nagyon meg tudják tévesztetni a programot, ezért nehéz csak színinformációk alapján felismerni

5.4.3 Délután készült felvételek eredményei

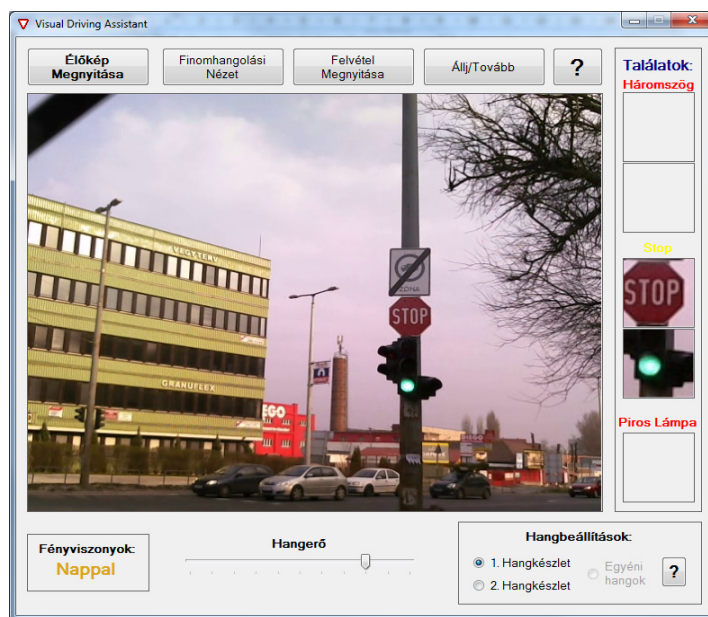
A 3. táblázaton összefoglalásra került felvételek 15:00 és 16:00 között készültek. Itt már a napszaktól adódó megvilágítási körülmények kezdenek problémákat okozni, megjelennek rosszul kontrasztolt, szinte szintelen táblák, melyeket a rendszer nem képes felismerni. Összességében azonban az elért eredmények itt még megfelelőnek mondhatóak.

Felvétel	Ciklusok	Felismerendő	Felismert	Hibás
54	148	4	4	0
55	147	11	11	0
56	278	1	1	0
57	161	6	6	0
58	161	0	0	1
64	383	4	0	2
70	358	2	0	0
73	378	2	0	1
82	311	10	10	1
88	305	3	0	1
94	314	75	72	0
96	397	5	3	0
97	451	12	12	0

Összesen: 135-ből:
119 felismerve, 6 hiba
Hatásfok: 88,1%
Hibasázalék: 5%

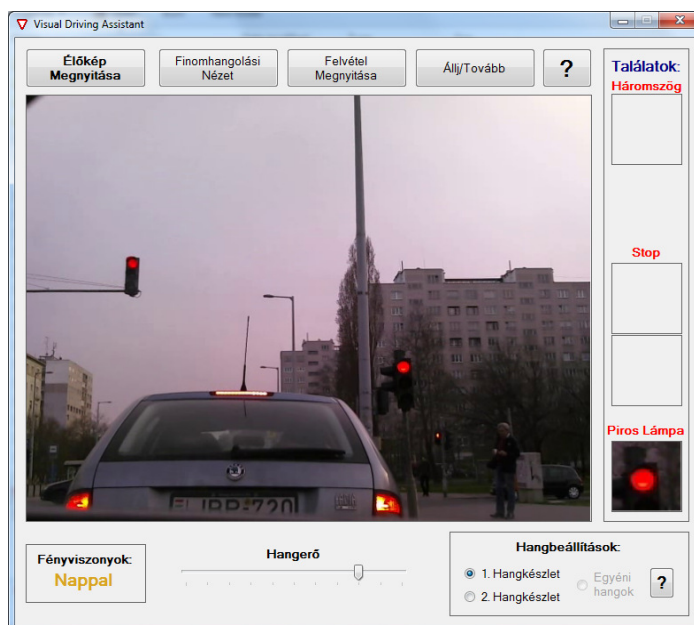
3. táblázat
Délutáni felvételek kiértékelése

Jelenlegi beállítások mellett tehát itt kezd el visszaesni a hatékonyság, helyenként előtérbe kerül a célkitűzésekből eredő legsúlyosabb probléma, a webkamerák képességeinek korlátai, de ez a későbbi napszakokban sokkal jobban befolyásolja majd az eredményeket. Ez még az elfogadható korlátokon belül mozog (5.11. ábra). A hibaszázalékot a felismerések mennyisége jelentősen elnyomja, a 94-es felvételen található hosszú piros lámpa (5.12. ábra) a felismerések 60%-át teszi ki. Nagymértékű hiba azonban egyik felvételen sem tapasztalható.



5.11. ábra

A színek még ebben a napszakban is elfogadhatóak néhány esettől eltekintve



5.12. ábra

A féklámpák nem zavarják meg a rendszert nappal, a jó ROI ráadásul nagy részét ki is szűri

5.4.4 Késő délután készült felvételek eredményei

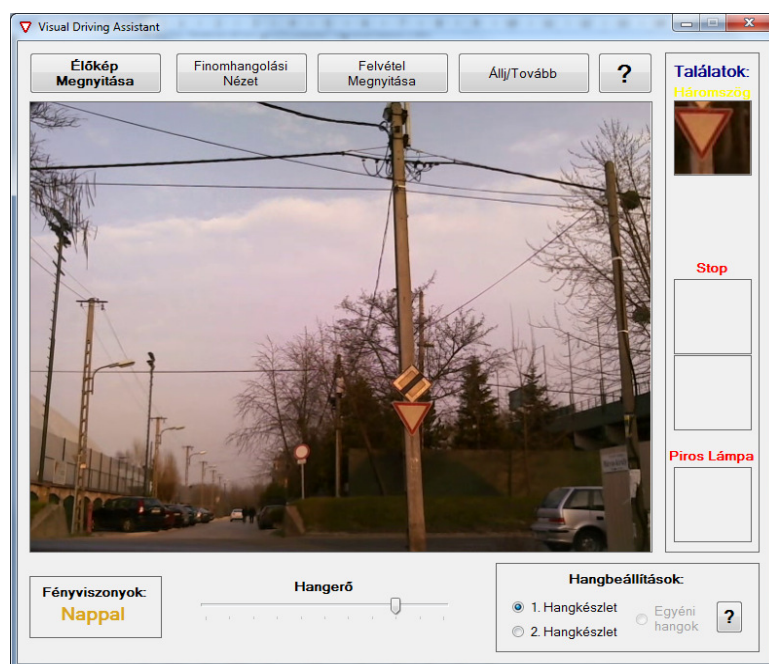
A 16:00 után készített felvételek tartoznak ide, a naplemente itt már valós probléma, ahogy azt az 4. táblázat eredményei is mutatják. Ezeken a felvételeken már szemmel láthatóan súlyos színeltérési problémák tapasztalhatóak.

Felvétel	Ciklusok	Felismerendő	Felismert	Hibás
101	83	1	0	4
107	273	9	9	4
113	311	2	0	0
119	58	6	6	0
127	255	4	4	3
128	261	2	1	0
134	180	3	3	0
137	213	10	10	5

Összesen: 37-ből:
33 felismerve, 16 hiba
Hatásfok: 89,1%
Hibaszázalék: 48,4%

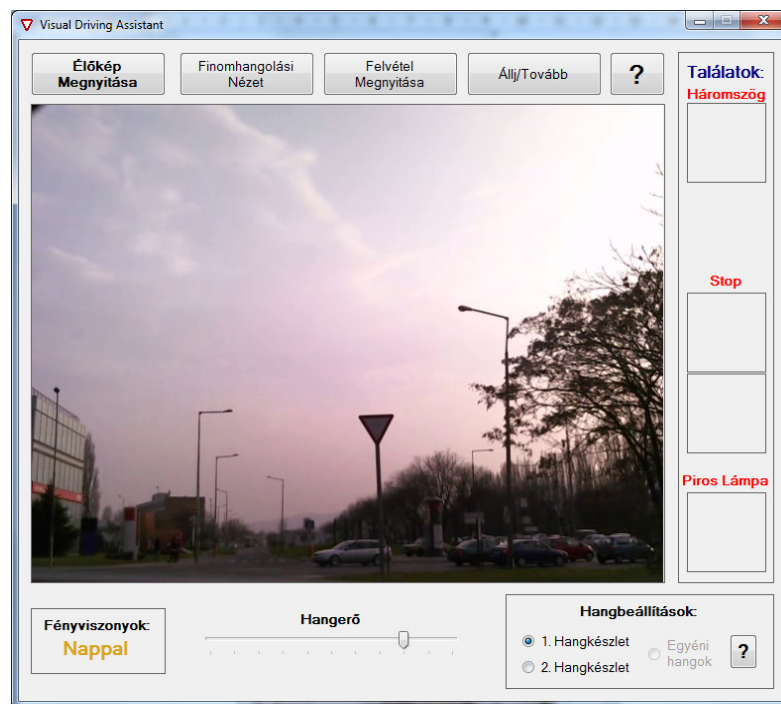
4. táblázat
Késő délutáni felvételek kiértékelése

A késő délutáni napszakban, a legtöbb hiba és a felismerések hiánya is a kamerával szembesítő nappal hozható összefüggésbe. Sajnos a webkamera automatikus fehéregyensúly- és kontrasztbeállítása ilyen körülmények között nagyon rossz képet eredményez, amelyen a teljesen eltorzult színintenzitások nagyon erős hibákat okoznak, ebből adódóan jóval magasabb a hibaszázalék. Az általunk kitűzött körülmények között ez a hatásfok még így is elégségesnek számít (5.13. ábra), mivel egy koromfekete táblával, ami még bele is olvad a mögöttes feketeségbe (egyedül szinte csak a kékes-rózsaszínes égbolt látható) szinte lehetetlen boldogulni, akármilyen színteret vagy algoritmust is válasszunk (5.14. ábra).



5.13. ábra

A problémák ellenére, ha nem süt szembe a nap, nincs gond a felismerésekkel



5.14. ábra

Az ilyen esetekkel már nem lehet mit kezdeni amíg csak színinformációkra támaszkodunk

5.4.5 Éjszaka készült felvételek eredményei

Az éjszakai táblafelismerés nehézségeit már az első fejezetekben is megemlítettük, és az 5. táblázaton is látható ennek befolyása az elért eredményekre. Az általunk készített felvételekkel történő tesztelés során, már a beérkező képkockán is szemmel láthatóan egyértelművé váltak a jelentkező problémák.

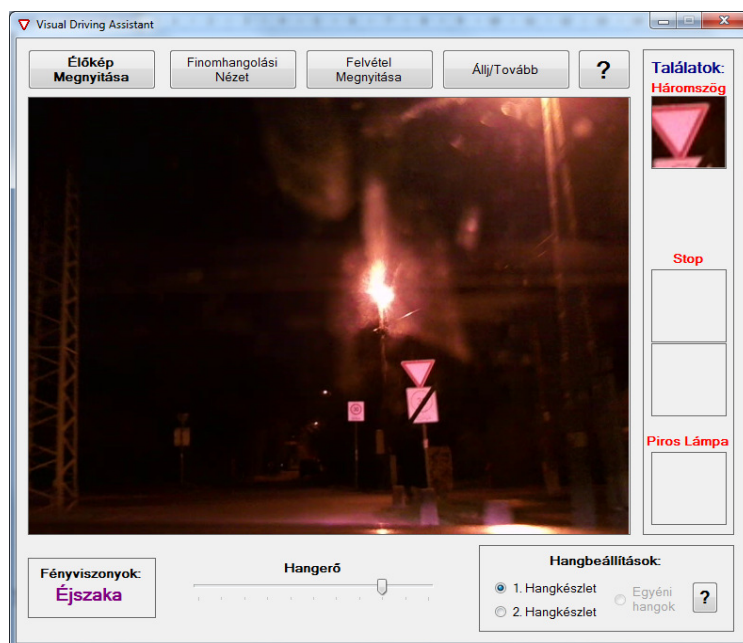
Felvétel	Ciklusok	Felismerendő	Felismert	Hibás
143	149	7	7	0
152	136	2	0	1
163	137	3	3	0
173	99	4	4	0
183	154	7	7	7
187	225	3	3	0
188	63	2	0	0
190	77	6	6	0
193	94	4	0	0
197	94	16	16	0
214	41	1	0	0
222	100	12	8	6

Összesen: 67-ből:
54 felismerve, 14 hiba
Hatásfok: 80,6%
Hibaszázalék: 25,9%

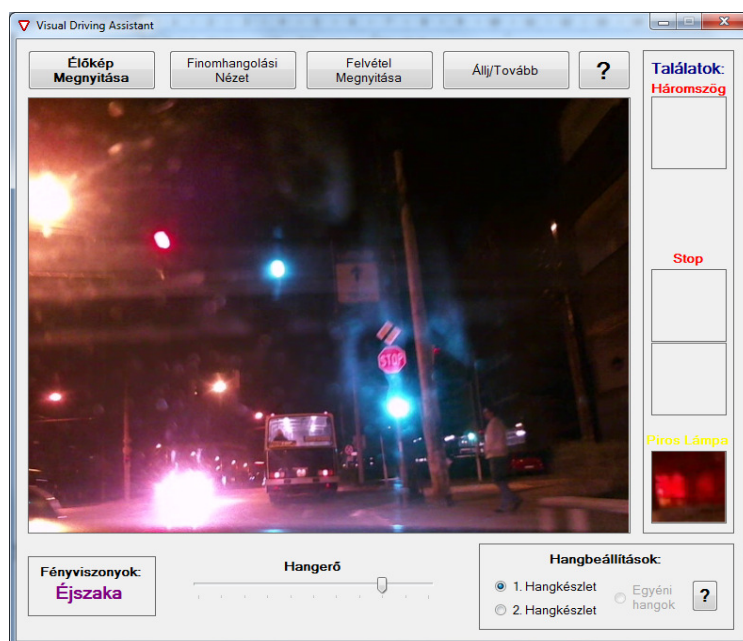
5. táblázat
Éjszakai felvételek kiértékelése

Éjszaka a zavaró tényezők száma összemérhetetlen módon megnő akármilyen más napszakban tapasztalhatóakéhoz képest (5.16. ábra). Külön problémát jelent a városi közvilágításban alkalmazott erős narancssárga színű nátriumgőz lámpák fénye is, melyeket nagyon nehéz kiszűrni és jelentősen megzavarják a piros objektumok küszöbölését (ezért is jelenik meg egy plusz feltétel éjszaka ezen objektumoknál).

Ilyen körülmények mellett minden egyes tábla felismerése sikernek számít már egy olyan egyszerű eszközzel, mint egy VGA-felbontású webkamera (5.15. ábra). Összességében elégedettek vagyunk még ezzel a hatásfokkal is, az éjszakai módra váltásnak köszönhetően a kimondottan éjszakára hangolt küszöb- és súlyértékeknek köszönhetően így is sikerült elérni egy elfogadható szintű megbízhatóságot.



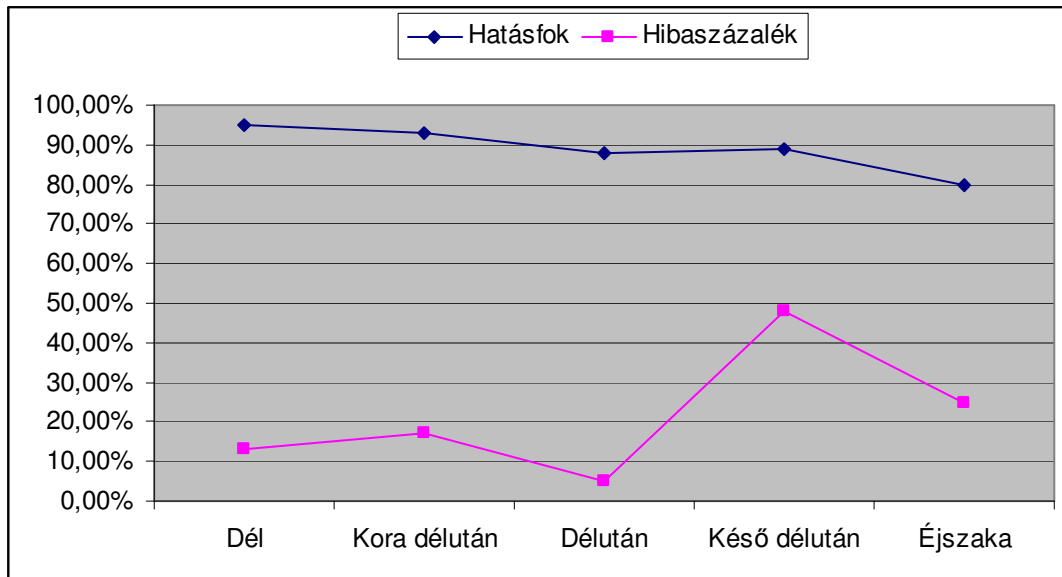
5.15. ábra
Éjszaka a rendszer teljesen ki van szolgáltatva a jó megvilágítási viszonyoknak



5.16. ábra
A különböző éjszakai zajok sokféle problémához vezethetnek

5.5 Az eredmények értékelése

Ahogy az összesített diagram (5.17. ábra) is mutatja, jól kivehető az, hogy mi a probléma forrása, és hol van az a határ, ahonnan a használt képfelvevő eszköz már megbízhatatlan bemeneti forrásnak számít.



5.17. ábra
A különböző napszakokban elért eredmények

A webkamera beépített kompenzációs eljárásai egyértelműen más környezetre lettek optimalizálva, ami érthető is, lévén nem tartozik az alkalmazási körei közé a közlekedési táblafelismerés. Az eredmények alapján megállapíthatjuk, hogy habár a tesztelés során felhasznált küszöbölési, illetve súly értékek még hónapok munkájával tökéletesíthetőek lehettek volna, vannak olyan körülmények, ahol a definiált célkitűzések mellett egy ilyen egyszerű eszközzel nem lehetséges már a megbízható felismerés. Az első három napszakban azonban kielégítő eredményeket produkál a rendszer, magas hatásfok mellett csak néhány hiba lép fel kedvezőtlen esetekben.

A hibaszázalék látványosan a késő délutáni órákban tetőzik, amikor is a lenyugvó nap vagy közvetlenül a kamerába süt, vagy pedig a megvilágítás szögéből adódóan már nem jut elegendő fény a közúti objektumok számára. Ez sajnos teljesen megzavarja az eszköz képalkotási mechanizmusát, szinte mindent csak egy fekete sziluetté alakítva. A hatásfok csökkenése a napszakok előrehaladtával jól megfigyelhető, a legideálisabb déli fényviszonyoktól kiindulva, egészen a legnehezebbnek mondható éjszakai körülményekig.

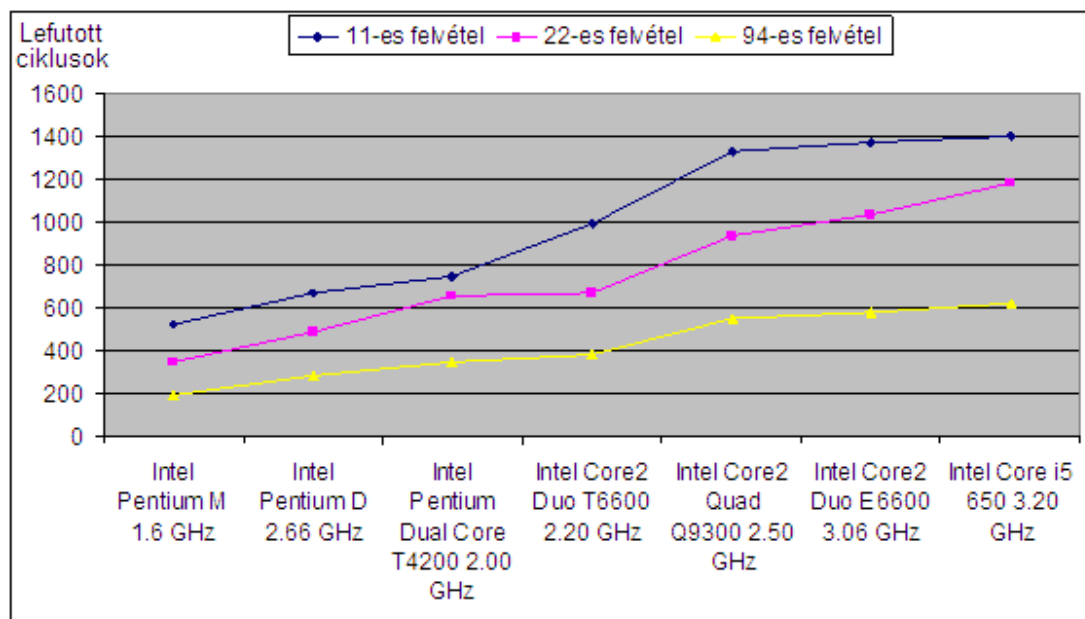
Természetesen a használt algoritmus is jelentősen befolyásolja az eredményt, nyilvánvalóan létezhet ennél optimálisabb mechanizmus is a megadott hardveres környezetben. Igyekeztünk minél többet kipróbálni a lehetséges módszerek, finomítási technikák közül, azonban a megbízhatóság, és az ellene dolgozó futási sebesség között egy optimális középútnak bizonyult az általunk felvázolt és végső soron alkalmazott megoldás. Az esetleges továbbfejlesztési lehetőségről és az algoritmus megbízhatóságának esetleges növeléséről még szót ejtünk az utolsó fejezetben.

5.6 Futási sebesség különböző konfigurációkon

A futási sebesség mérésére három különböző felvételt használtunk. A 11-es számú felvétel egy 47 másodperces hosszú monoton felvétel, ahol a felismerő egységnek szinte semmi dolga nem akad. A 22-es felvétel egy 43 másodperces, hosszú felvétel, azonban nagyon mozgalmas, és a felismerő modul számára sok objektumot biztosít. Az utolsó 94-es számú felvételen egy hosszú piros lámpának köszönhetően sok felismerés található. Így három különböző jellegű videón tudtuk bemutatni a program teljesítményét a különféle erősségű processzorokon (5.18. ábra).

Összesen hét processzoron futtattuk le a teszteket, az általunk elérhető asztali és hordozható számítógépeken egyaránt. Az eredmények nem meglepőek, a program futásában egyértelműen szűk keresztmetszetként funkcionál a CPU által nyújtott számítási teljesítmény. Az utolsó két helyen teljesítő processzor között egy asztali és egy hordozható eszközökben funkcionáló is megtalálható. Mindkettő, ma már új számítógépekben nem alkalmazott, elavultnak tekinthető architektúra, ezek kevesebb, mint fele annyi felismerési ciklust produkálnak, mint amennyit a legjobbakon mértünk. Az első, már teljesen elfogadható eredményt, az első kettőnél modernebbnek számító, de már szintén kifutóban lévő T4200-as hozza, ami a manapság használatban lévő néhány éves hordozható számítógépekben gyakran előforduló processzor. Itt a mozgalmasabb felvételen érezhető a jobb teljesítmény. Ennél egy fokkal jobbat produkál egy újabb, mai hordozható számítógépekbe szánt T6600-as. Az előbb említett két feldolgozó egység elegendő a program olyan szintű futtatásához, hogy a sebesség ne okozzon hibákat a felismerések során. A legjobb három processzor már valós időben képes volt minden képkockát feldolgozni a monoton 11-es felvételen, azonban a mozgalmasabb 22-es, illetve a sok felismerést produkáló 94-es videók esetén is egyenes arányos növekedés volt megfigyelhető a számítási teljesítmény növekedésével.

Végző soron tehát megállapíthatjuk, hogy a célkitűzésnek megfelelően, a program működőképes sebességgel képes futni a napjainkban használatos hordozható számítógépeken, megfelelő eredményeket produkálva.



5.18. ábra
Lefutott felismerési ciklusok száma az egyes processzorokon

6. Összefoglalás

6.1 Kitűzött és elért célok

Mivel a rendszer alapvetően a gépi látás problémakörébe sorolható, sokkal nehezebb feladat pontosan értékelni és meghatározni a végeredmény specifikációkhoz mért relatív eredményességét, mint egy olyan rendszer esetében, amely a klasszikus, egyértelműen meghatározott módszereket felhasználva látja el funkcióját. Valójában ez szinte minden magas hibatűrést, alkalmazkodóképességet, dinamikus adatfeldolgozást igénylő rendszer esetén igaz, legyen az akár beszédfelismerés vagy robotika.

A végleges verziót, annak funkcióit és eredményeit tekintve elmondhatjuk azt, hogy a projekt megfelelő mértékben megközelíti – a fejlesztések során szerzett tapasztalatokhoz mérten – az eredetileg felvázolt célkitűzéseket. A hardveres és szoftveres környezetben nem történtek változtatások, végig elsődleges szempont volt az egyszerűség, és az elterjedt eszközök használata, bár ez sok esetben az eredetileg várt korlátokon felül is további nehézségeket okozott. A helyes és téves felismerések aránya kielégítő, ahogy a fényviszonyok kezelése is, melyhez bár egyszerű módszert alkalmazunk, összességében megfelelőnek mondható.

6.1.1 Fejlesztési tapasztalatok

A projekt fejlesztése – melyhez elválaszthatatlanul hozzá tartozik a folyamatos tesztelés – nagyon sok olyan tapasztalat gyűjt össze, mely döntően befolyásolta a tervezés és implementálás menetét. A legfontosabb programozás-technikai tapasztalataink közé tartoznak a .NET platformmal, és annak felépítését a leginkább hűen visszaadó, általunk is használt C# magas szintű programozási nyelv sebességével és optimalizálási lehetőségeivel kapcsolatosak.

Egy kép feldolgozásához alapvetően szükséges elsősorban azt a memóriában valamilyen objektumként eltárolni, valamint annak intenzitásértékeit kiolvasni, azokon a megölelő műveleteket végrehajtani, és végül visszaírni. A tárolást közvetlen módon .NET alatt a *System.Drawing* névtér *Bitmap* osztályának egy példányával tehetjük meg. Ennek metódusai között pedig megtalálhatjuk a *GetPixel()* és *SetPixel()* nevezetűeket, melyekkel egyszerűen, és kényelmesen megoldható mind az olvasás, mind az írás. A probléma azonban azonnal szembetűnik, mihelyt akár a legegyszerűbb képfeldolgozási algoritmust is lefuttatjuk rajta: nagyon-nagyon lassú, a gyakorlatban teljességgel használhatatlan, a magas szintű objektum-orientáltság kényelmének és biztonságának itt bizony túlságosan magas ára van.

A megoldás többféleképpen is lehetséges. A feldolgozási algoritmusok szemszögéből a legegyszerűbb, és egyben a leggyorsabb is, hogyha mátrix, vagy sorfolytonos vektor ábrázolásban férünk hozzá a pixelek értékeinek halmazához. Erre több megoldás is kínálkozik, például *unsafe* blokk használata, melyben lehetőség nyílik nem felügyelt kód futtatására, így mutatók is felhasználhatóak a gyorsabb adatolvasás és visszaírás végett, illetve a *System.Runtime.InteropServices.Marshal* osztály, mely valójában ugyanezt hajtja végre metódusaival anélkül, hogy a programozónak ki kellene lépnie a felügyelt környezetből.

Eleinte ez utóbbi megoldást alkalmaztuk, de mivel a videó és a kamera kezeléséhez felhasználjuk az OpenCV-t, kézenfekvő megoldás annak képobjektumát alkalmazni. Ennek egy módszerével gyorsan másolható ki-és vissza a képet reprezentáló vektor, és nagyon egyszerűen használható. Az OpenCV képfeldolgozó eszközök egész tárházát biztosítja a kezeléstől a megjelenítésig. Tárolásra az *Emgu.CV.Image<Tcolor, Tdepth>* osztály állt rendelkezésünkre, ami generikus mivoltának köszönhetően igényeink szerint testre szabható volt, a szintér és a színmélység is közvetlenül a webkamera által szolgáltatott RGB szintér, csatornánként 8 bites színmélységgel. Megjelenítésre az *Emgu.CV.UI.Imagebox* osztályt alkalmaztuk, ami azért említésre méltó, mert a C# maga is rendelkezik egy hasonló elnevezésű eszközzel, azonban ezzel is konverziós időt spórolhatunk meg, mivel az OpenCV-s *Imagebox* értelemszerűen közös interfésszel rendelkezik a szintén ebből a könyvtárból használt *Image* osztállyal.

A C# nyelvet magas absztrakciós szintje és felügyelt mivolta miatt sokan lassú nyelvnek tartják, ami nem fedí az igazságot teljes valójában, ugyanis nagyon sok olyan eszköz és módszer áll rendelkezésünkre, amelyek segítségével minimális teljesítménybeli különbség érhető el egy hardver közelebbi nyelvhez képest, emellett mégis kényelmesebb fejlesztőkörnyezetet biztosít. Megvalósításunk esetén kiemelkedő szerephez jut az *Array* osztály különféle tömbkezelő eljárásainak használata. Az *Array.Clear()*, *Array.Copy()* és *Array.CopyTo()* eljárásoknak köszönhetően a 4.3.1.-es fejezetben részletesen is kifejtett módszerekkel sikerült elsősorban gyorsítást elérnünk, amelyek nagyon sok ciklusidőt spóroltak meg futás közben.

A ciklusok nyelvtől függetlenül a leglassabb kódrészek közé tartoznak. Ezért azok kerülése, de legalább minimalizálása mindig kívánatos. Már említettük az előbbieken, hogy a *goto* eljárás használata manapság szinte tilosnak mondható, de ez nem teljes mértékben helytálló, ugyanis nagyon mélyen beágyazott ciklusok esetén, ha tudjuk, hogy abban bármikor jöhet olyan találat, amikor ki kell tudnunk jutni abból a mélységből minél hamarabb, ez az eljárás a leggyorsabb megoldás. Így azt mondhatjuk, hogy ha a teljesítményoptimalizálás fontos szerepet tölt be egy program esetén, az egyszerű ugrás még mindig szükséges lehet, természetesen csekély mennyiségben, és jól megfontolt kereteken belül, ugyanis joggal nem tartozik az általánosan elfogadott megoldások közé.

A fejlesztés során az is hamar realizálódott, hogy a felismerő egység párhuzamossá tételével sokat tudunk gyorsítani a program működésén. Ennek megvalósítására a C# szintén egyszerű megoldást nyújt, egy külön eszközt biztosít annak kezelésére. Azonban a legegyszerűbb párhuzamosítás esetén is már nehezen kezelhető problémák léphetnek fel. A programok a Neumann-architektúrából eredően soros működést feltételeznek, amit a párhuzamos futás teljesen felborít. A mi esetünkben szerencsére a párhuzamosítás megtervezése egyszerű volt a moduláris felépítésnek köszönhetően, azonban még így is megtapasztaltuk, hogy miért is ez a modern programozás-technika egyik legnagyobb kihívása. Egy egyszerű, általunk alkalmazott példa erre, hogy a találati képernyő elvileg mindig az aktuálisan lekért képkockáról (amin a felismerés történt) vágja ki a kis 40x40-es méretű képet ahol a felismerés vissza lett igazolva. Azonban gyakran előfordul, hogy mire ez megtörténik, a párhuzamosan futó szál már lekérte a következő képet feldolgozásra, ahol a gyors mozgás következtében már jelentősen elmozdult a tábla pozíciója, és mire az ábrázolásra kerülne sor, a tábla alig található meg azon a helyen. Mivel csak egy megjelenítő funkcióról van szó, ezért elhanyagolható ennek fontossága, azonban ez a példa remekül felvázolja azt, hogy a legapróbb részletekig újra kell gondolni egy program működését, amint párhuzamosítani kívánjuk azt.

6.1.2 A korlátok behatárolása

Amikor a projekt kezdetén meghatároztuk a célkitűzéseket, tisztában voltunk azzal, hogy ilyen korlátozások mellett nem lehet tökéletes rendszert alkotni. Azonban pont az jelentette a kihívást, hogy a megvalósítás során képesek legyünk a legjobbat kihozni ezekből az eszközökből az elvárásoknak megfelelően úgy, hogy mégis egy működő rendszert kapjunk végeredményül. A projekt végére sikeresen elkészítettünk egy olyan programot, ami alkalmas a specifikációban meghatározott körülmények között működni, de ami ennél is fontosabb volt, hogy képesek voltunk meghatározni ennek korlátait, és jól körülhatárolni azok miéttjeit. Így két jól izolált problémát sikerült meghatároznunk, ami miatt ez a rendszer korlátosnak nevezhető.

Az első a késő délután mért teszteredményeknél tapasztalható magas hibaszázalék problémája. A dinamikus kontrasztjavítás, ami ezekben az egyszerű készülékekben található nem elégséges arra, hogy az alacsonyan lévő nap által okozott fényviszony változásokat rendesen legyen képes kezelni. Ha már maga a forráskép rossz minőségű (6.1. ábra), a színintenzitások kijavítására még akkor se lenne lehetőség, ha elvetnénk a valós időben történő futás igényét.



6.1. ábra
Egy tipikus forráskép késődélutáni körülmények között

A másik az éjszakai teszteredményeknél tapasztalt érezhető csökkenés a felismerések terén. Az éjszaka megjelenő zavaró tényezők, amiket az egyszerű bemeneti eszköz szintén nem képes kezelni, olyan képi anomáliákat okoznak, amik kiszűréséhez komplex algoritmusokra lenne szükség, amivel borulna az általunk felállított követelmények egyik legfontosabbika, a valós időben futás. Mindezekon felül éjszaka már megjelennek a szélvédőn található apró foltok – melyek mindig jelen vannak, ezek keletkezésének kivédése lehetetlen – által okozott elmosódások, főleg erős közvilágítás esetén. A gyengébb eredmények tehát ebben az esetben is nagyrészt a forráskép hibáira vezethetőek vissza.

A rendszer megfelelő működésének tehát elsődlegesen a sötétebb fényviszonyok szabnak határt, ugyanis az összességében nem rendelkezik ezt a problémát kiküszöbölő, akár hardveres, akár szoftveres megoldásokkal, mivel a célkitűzések által megszabott korlátok ezek alkalmazását nem teszik lehetővé.

6.2 Összehasonlítás hasonló rendszerekkel

Gépi látásról lévén szó – a fejezet bevezetésében már kifejtésre került, hogy ez a tudományág még ma is mennyire képlékeny – az abszolút eredményesség nehezen számszerűsíthető. Mivel ezek a rendszerek csupán a beérkező intenzitásinformációk sorozata alapján, statikus mechanizmusokkal próbálják felismerni az objektumokat, óriási hátrányban vannak az emberrel szemben, akinek a „képfeldolgozása” dinamikus és adaptív, köszönhetően az ezt lehetővé tevő komplex központi idegrendszernek. A hatásfok kérdését ezért mindenképpen relatívan kell kezelni, ennek megfelelően tehát mindig a hasonló rendszerekkel történő összehasonlítás képezi az értékelés alapját.

A következőkben ennek megfelelően megnézzük néhány, a miénkhez hasonló jelzőtábla felismerő rendszer hatékonyságát. Bár léteznek speciális, célhardveres megvalósítások is erre a célra, sőt már olyanok is, amelyek alapvető részét képezik egy-egy gépjárműtípusnak, ezek eredményessége nyilvánvalóan felülmúlja az általános célú számítógépekre szánt megoldásokét. Előnyük a magas hatásfok, azonban ezek a megoldások csupán az adott gépjárműben alkalmazhatóak, így egyértelműen más céllal készültek, mint a projektünk, illetve a miénkhez hasonló rendszerek. Ennek megfelelően vizsgálatunk ezen utóbbiakra szorítkozik. Annyit azért mindenképpen érdemes megemlíteni, hogy egy ilyen egyszerű rendszer, mint a miénk is képes bizonyos körülmények között megbízható eredményeket szolgáltatni.

6.2.1 Néhány rendszer eredményeinek vizsgálata

A [Shneier] által bemutatott rendszer nagyban hasonlít a mi projektünkhöz, így mindenképpen érdemes megvizsgálni, hogy milyen eredményeket is ért el. A második táblázat (6.2. ábra) *Percent Recognized* oszlopában látható, hogy a felismert célobjektumok százaléka egy kicsivel alacsonyabb, mint a mi esetünkben, azonban ahogy azt a *False Recognitions* oszlop mutatja, a téves felismerések terén jobban teljesít a rendszerünknel. Ez a két érték – ahogyan az mi is tapasztaltuk – sajnos nagyon egymás ellen dolgozik, ezt [Shneier] tanulmányának tesztelését bemutató részében is olvashatjuk. Említést érdemel még, hogy [Shneier] bemeneti forrása egy, a gépjármű tetejére helyezett videokamera volt, így sokkal jobb minőségű felvételekkel tudott dolgozni, ráadásul a szélvédő által okozott képi zavarok se jöhettek szóba.

Run	No. Frames	No. Signs	No. Detected	No. Recognized	False Detections	False Recognitions
1	10,924	39	33	29	19	1
2	9,439	45	43	38	34	5
3	3,274	8	5	5	0	0
1, 2, 3	23,637	92	81	72	53	6

Table 1 Performance of Sign Detection and Recognition

Run	No. Frames	No. Signs	Percent Detected	Percent Recognized	False Detections	False Recognitions
1	10,924	39	96 %	84 %	76 %	11 %
2	9,439	45	95.4 %	86 %	75 %	7 %
3	3,274	8	63 %	63 %	0 %	0 %
1, 2, 3	23,637	92	88 %	78 %	58 %	6.5 %

Table 2 Percentage results for Sign Detection and Recognition

6.2. ábra

[Shneier] által felvázolt rendszer tesztelési eredményeket összefoglaló táblázata

[Shojania] rendszere működésében jelentősen eltér az általunk felvázolttól. Ahogy az irodalomkutatásban is láthattuk, az általa alkalmazott felismerési folyamat komplexebb, több összetett algoritmust is felhasznál annak során. Ebből adódik a nagyobb megbízhatóság, de mint azt ki is emeli az összefoglaló részben, ennek hátránya a lassú feldolgozás, így mindenképpen nagyon erős hardveres környezetet, vagy speciális architektúrát igényelhet éles körülmények közötti használatra.

Ennek megfelelően [Shojania] pontos, számszerűsített tesztelési adatokat sem közöl, azonban az általa bemutatott elemzési folyamat – pontosan annak jelentősen bonyolultabb mivolta miatt – számos hasznos következtetéssel, problémaértékeléssel segítette munkánkat, és a rendszerünk eredményeinek értékelését. Külön érdekesség a kör alakú piros objektumok között egyértelműen és megbízhatóan különbséget tevő algoritmus megtalálásának nehézsége, mivel amennyiben ilyen objektumok tűnnek fel a képen, még egy ilyen összetett képelemzési folyamat is sokszor hibázik (6.3. ábra). Ez a probléma esetünkben is előbukkan, leggyakrabban akkor, amikor féklámpák kerülnek be az érdekelt régióba.



6.3. ábra

A [Shojania] rendszerében is fellelhető probléma kör alakú piros objektumok esetén (ezt a módszert a mentő jelzőfényének szintén piros körkerülete zavarja meg)

A [Mühlbauer] által bemutatott robot, az ACE is alkalmaz közlekedési jelzéseket felismerő modult, ami a szempontunkból azért érdekes, mert a leírásban utal arra is, hogy a téves felismerések nagy része csak „bevillan”, legtöbbször csupán csak egy képkocka erejéig, míg a valódi objektumok esetén ez több képkockán is nagy valószínűséggel megjelenik. Ahogy a működésnél leírtuk, mi is ezt a jelenséget használjuk ki a kiértékelésnél.

Az ACE robot ezen moduljának eredményei elég jónak mondhatóak, ugyanis a sikeres felismerések aránya 79%-88% közé, míg a téveseké csupán 2%-6% közé esik átlagosan. Azonban ennek megvan az ára: egy képkocka teljes átvizsgálásának időigénye 1,2 másodperc, azaz a rendszer nagyon lassúnak mondható. Ez a robot esetében nem akadály, hiszen annak sebessége csupán 1,8 km/óra, de nyilvánvalóan gépjárművek esetén ez a rendszer nem jöhetne szóba.

6.2.2 A rendszer előnyei és hátrányai

Néhány hasonló rendszer igényeinek és eredményeinek vizsgálata tükrében több következtetést is levonhatunk a saját rendszerünkkel kapcsolatban, megállapíthatjuk annak néhány előnyös illetve hátrányos jellemzőjét. Bár a megvizsgált rendszerek esetén is a specifikáció által felvázolt követelmények és igények, valamint azok részletessége is széles skálán mozog, az alapvető funkciójuk megegyezik.

A Visual Driving Assistant szoftver legfontosabb előnye az, hogy nem igényel semmilyen speciális hardvert, különleges beruházást, vagy átalakítást a járműben, a rendszer követelményei olyan általános célú eszközökre korlátozódnak, melyek széles körben elterjedtek. A program számítási teljesítmény igénye manapság a hordozható számítógépek tekintetében átlagosnak mondható (bár fontos megjegyezni azt, hogy gyorsabb processzoron a program hatásfoka javul, mivel adott idő alatt több felismerési ciklus fut le, magasabb lehet így a mintavételezés). A felhasznált eszközök egyszerűségének ellenére a rendszer által nyújtott teljesítmény is elfogadható.

Az előnyökkel szemben azonban – mint minden rendszernek – hátrányai is vannak, melyek között megtalálhatunk olyat is, ami közös forrásból ered egy pozitív tulajdonsággal, csak éppen „az érem másik oldala”. Az olcsó és egyszerű hardver, elsősorban a képrögzítő eszköz sajátosságai nagyban befolyásolták a szoftver működésének folyamatosan változó – és ennek eredményeképp kialakult végső – tervét így magát a végleges verziót is. Elsősorban ez azt jelenti, hogy a kialakított mérőszámok, a régió méretei, a keresett célobjektumok várható kiterjedései is ehhez igazodtak, így a szoftver hatásfoka, annak átalakítása nélkül nem növelhető jelentősen egy sokkal komolyabb képalkotó eszköz használata által sem. A második komolyabb hátránya a rendszernek szintén a webkamerák korlátosságából adódik, nevezetesen a nagyon rossz hatásfok késő délutáni, illetve éjszakai fényviszonyok között. Ez, mint azt meg is mutattuk, sajnos nagyon nehezen kezelhető probléma. Mivel nem volt lehetőségünk tesztelni speciális időjárási viszonyok között, pontos információink nincsenek arról, hogy ilyen esetekben hogyan teljesít a program, így csak sejthető az, hogy például esős, borús idő esetén is valamilyen szinten előjöhet a fényviszonyokból eredő hatásfok-csökkenés.

6.3 Továbbfejlesztési lehetőségek

A projekt – bár a kitűzött célokat elérte – sok továbbfejlesztési lehetőséget is rejt magában. Bár sokat foglalkoztunk az optimalizálással és a megfelelő működéshez szükség finomhangolásokkal, ezen a téren lehetségesek olyan átalakítások, melyekkel növelhető a program hatékonysága. Természetesen igyekeztünk a lehető leghatékonyabb megoldást megtalálni és az ahhoz tartó paramétereket helyesen beállítani, de lévén hogy a gépi látásban a körülmények legkisebb változása is szembetűnő eltéréseket okozhat, az alkalmazható algoritmusok száma óriási, így egyértelműen sosem állapítható meg, hogy az adott feladat végrehajtásához melyik a legmegfelelőbb, azaz amivel a rendszer bármilyen előálló helyzetben a tökéletesen funkcionál.

Említettük, hogy a program sebességében a CPU jelenti a legszűkebb keresztmetszetet, így a hatékonyság növelhető a processzor által nyújtott lehetőségek minél szélesebb körű kihasználásával. Ilyenek például az utasításkészlet egyes kibővítései, elsősorban az SSE – és ennek újabb verziói –, melyek felhasználásával jelentősen gyorsíthatóak a nagy mennyiségű adaton végrehajtandó műveletek.

Manapság már egyértelművé vált az a tény is, hogy a többmagos processzorok jelentik a teljesítménynövelés egyetlen járható útját, így egyre nagyobb szerepet kap a párhuzamos adatfeldolgozás, amellyel kifejezetten az ehhez hasonló feladatok esetén jelentős gyorsulást lehet elérni. Az egyre magasabb szinten párhuzamos architektúrák által nyújtott lehetőségek kiaknázásához az egyes futtató-, és a hozzájuk tartozó fejlesztőkörnyezetek, így a .NET is újabb és újabb lehetőségeket kínál. A *Parallel.For()* metódus például a hagyományos számláló ciklus párhuzamos végrehajtását valósítja meg, amennyiben az lehetséges.

6.4 Konklúzió

Összefoglalva tehát sikerült megalkotnunk egy olyan rendszert, mely – bizonyos korlátok között – megfelel a specifikációban felvázoltaknak, azaz képes vizuális információk alapján, valós időben, hangjelzések segítségével tájékoztatni a járművezetőt az aktuális helyzetben a legfontosabb közlekedési jelzésekről. A tervezés és fejlesztés folyamata evolúciós módszertant követett a feladat sajátosságai miatt, így valójában a kezdeti verzió folyamatos módosításával, a megszerzett tapasztalatok tükrében alkalmazott változtatásokkal jutott el végleges formájába.

A témában való kutatásaink, és a rendszer elkészítése során arra a következtetésre jutottunk, hogy bár a gépi látás az informatika egy fiatal ágának tekinthető, mégis mára már komoly irodalommal, kifejlesztett algoritmusokkal, és eredményekkel büszkélkedhet. Mindezek ellenére azonban hétköznapi hardveres eszközök használata esetén – melyeknél az általános célú adatfeldolgozásra szánt architektúrák dominálnak – elérhető képfeldolgozási teljesítmény még nagyon szigorú megkövetelésnek számít, főleg a napi szinten fejlődő és egyre komplexebb képfeldolgozási mechanizmusok mellett.

Azonban ahogy a projektünk is megmutatja, már nem járunk messze attól, hogy a hétköznapiokban használatos hardveres eszközeink is képesek legyenek akár komolyabb gépi látáshoz kapcsolódó feladatok végrehajtására. Természetesen láttuk azt is, hogy ahhoz még, ezen általános célra szánt eszközök nem elegendőek, hogy komolyabb gyakorlati haszonnal kecsegtető eredményeket megbízhatóan és megfelelő sebességgel szolgáltatassanak. Ebben azonban komoly változást hozhatnak az egyre inkább elterjedőben lévő általános célú grafikai processzorok, melyek kifejezetten a nagy mennyiségű, de egymás közötti függőségi kapcsolattal nem rendelkező adatok gyors feldolgozásában – így például egy forráskép keretes bejárásán alapuló kiértékelésében – a szükséges műveletek magas szintű párhuzamosításával jelentős sebességnövekedést nyújthatnak. Itt azonban ma még problémát jelent a gyártók termékeinek különbsége azok programozása esetén, de ebben is konvergencia várható a jövőben, így a gépi látás nyújtotta lehetőségek belátható időn belül mindennapjaink részévé válhatnak, legyen szó biztonsági rendszerekről, háztartási robotokról, vagy éppen jelzőtábla felismerésről.

Irodalomjegyzék

- [Ebner] Marc Ebner, Engineering of Computer Vision Algorithms Using Evolutionary Algorithms, Eberhard Karls Universität Tübingen Wilhelm-Schickard-Institut für Informatik Abt. Rechnerarchitektur, Sand 1, 72076 Tübingen, Genetic Programming: Proceedings of the 12th European Conference, 2009, page(s) 1, <http://www.ra.cs.uni-tuebingen.de/mitarb/ebner/research/publications/uniTu2/EvoCVengineering.pdf>
- [Freeman] William T Freeman, Where computer vision needs help from computer science, ACM-SIAM Symposium on Discrete Algorithms (SODA), January, 2011, invited talk, page 1, <http://people.csail.mit.edu/billf/papers/cscvFreemanW.pdf>
- [Hatzidimos] John Hatzidimos, Automatic Traffic Sign Recognition in Digital Images, Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics, 2004, Thessaloniki, Greece, www.math.ethz.ch/EMIS/journals/AUA/acta8/Hatzidimos.pdf
- [Lorsakul-Suthakorn] Auranuch Lorsakul - Jackrit Suthakorn, Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV, The 4th International Conference on Ubiquitous Robots and Ambient Intelligence, 2007, www.bartlab.org/Dr.%20Jackrit%27s%20Papers/ney/3.KRS036_Final_Submission.pdf
- [Mühlbauer] Quirin Mühlbauer, Stefan Sosnowski, Tingting Xu, Tianguang Zhang, Kolja Kühnlenz, Martin Buss, Navigation through Urban Environments by Visual Perception and Interaction, Institute of Automatic Control Engineering Technische Universität München D-80290 Munich, Germany, International Conference on Robotics and Automation, 2009, page(s) 5, <http://www.lsr.ei.tum.de/fileadmin/publications/mbauer2009ICRA.pdf>
- [Shneier] Michael Shneier, Road Sign Detection and Recognition, IEEE Computer Society, International Conference on Computer Vision and Pattern Recognition, June 2005, www.isd.mel.nist.gov/documents/shneier/Road_Sign_Detection.pdf
- [Shojania] Hassan Shojania, Real-time traffic sign detection, CISC854: Computer Graphics, 2003, hassan.shojania.com/pdf/TrafficSignDetection-Paper.pdf

- [Shojania prezentáció] Hassan Shojania, Real-time traffic sign detection, CISC854: Computer Graphics, 2003, hassan.shojania.com/pdf/Real-time%20traffic%20sign%20detection-Presentation.pdf
- [Sommerville] Ian Sommerville, Software Engineering, 7th edition, School of Computer Science, St Andrews University, ST ANDREWS, UK 2004
- [Yang] Hsiu-Ming Yang, Chao-Lin Liu, Kun-Hao Liu, Shang-Ming Huang Traffic Sign Recognition in Disturbing Environments Department of Computer Science, National Chengchi University Taipei 11605, Taiwan Proc. of the 14th Intl. Symp. on Methodologies for Intelligent Systems, 2003, <http://www.cs.nccu.edu.tw/~chaolin/papers/ismis031.pdf>