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

TDK DOLGOZAT

GTA - MARADJ ÉBREN
GESTURES TO ALERT

SZERZŐK
HIRT KRISZTINA
KÁRPÁTI BENCE

KONZULENSEK
VAJDA LÓRÁNT
TÓTH ANDRÁS

BUDAPEST, 2015

Tartalomjegyzék

Összefoglaló

Napjainkban az informatikai rendszerek egyre inkább életünk részét képezik. Az intelligens mobil eszközök felhasználási lehetősége igen széleskörű. Szenzorok segítségével folyamatos monitorozás alatt tarthatják környezetüket, nincs ez másképp egy gépjármű vezető fülkéjében sem. Egy mobil készülék a vezető megfigyelésével és az OBD-II fedélzeti diagnosztikai rendszere által szolgáltatott adatok feldolgozásával egészségügyi támogatást nyújthat a vezető éberségi állapotának monitorozásában, ezzel komoly szerepet játszva az elalvás miatt bekövetkezett közúti balesetek megelőzésében.

Vezetés közben, legfontosabb a biztonsági szempontokat szem előtt tartani, így olyan alkalmazást kell tervezni, mely kezelése egyszerű, futás közben nem vonja el a figyelmet és közben kellő hatékonysággal határozza meg a vezető éberségi állapotát különböző vizuális gesztusok alkalmazásával, valamint a szenzor információk feldolgozásával.

A reakció mérése során, bizonyos időközönként, kézjeleket kér a vezetőtől, melyet a kamera felé mutatva detektál, így nem kell a képernyő megnyomására figyelni vezetés közben, mely tevékenység megint csak nagyban növelheti a baleset kockázatát. A felismerés során különböző képfeldolgozó és kézfelismerő algoritmusokat alkalmaztunk. Ennek segítségével először elkülönítjük a kézfejet a kamera által rögzített kép többi részétől, majd meghatározzuk a tenyeret, az ujjakat és a kart. Az így kapott információ alapján eldöntjük, hogy a vezető által mutatott kézjel megfelelő-e.

A reakcióidő és fedélzeti információk továbbításával és elemzésével meghatározható a fáradtság szintjének változása, így lehetőség nyílik a vezetőnek történő visszajelzésre, a viselkedésének módosítására, ezzel a baleset kockázatának csökkentésére.

Abstract

Nowadays, information technology is more and more present in our everyday life. Intelligent mobile devices are used in numerous ways. Such devices are able to constantly monitor their environment, for example while a vehicle is operated. A mobile device could watch the driver and process the data gathered by the OBD II on-board diagnostic tool and thus provide medical aid monitoring the vigilance of the driver. This could play an important role preventing accidents caused by falling asleep.

Safety has top priority while driving, therefore such an application should be easy to use, should not distract the driver and should provide reliable results, when it determines the level of tiredness of the driver, based on visual gestures and incoming sensor data.

To measure the driver's reaction time, the application periodically asks for a hand gesture, which has to be shown to the camera so it can be recognised. This way, the driver do not need to pay too much attention to the device and can concentrate on the road, which otherwise could highly increase the chance of an accident. Several image processing and recognition algorithms were implemented in the gesture recognition module of the application. With these, the hand is first isolated from the the rest of image, then the palm, the fingers and the arm is located. With that information, the application determines, if the shown gesture is the one asked for by the program.

By processing and forwarding the reaction time and the on-board diagnostic results, the change in the driver's level of tiredness can be determined. Then, it is possible to inform the driver, change his behaviour and therefore decrease the chance of an accident.

1 Bevezetés

A mai okos telefonok legnagyobb előnye, hogy rengeteg funkciót egyesítenek magukban. Mindig nálunk van, így sosem kell nélkülöznünk számos előnyét. Egyre gyakrabban kerülnek elő az élet minden területén, az egészségügyben is egyre több alkalmazás letölthető, például pulzusmérő, tünet-ellenőrző, alvásfigyelő, vagy különböző sport- és életmód- alkalmazások. Ezek mindegyike folyamatosan gyűjti az információkat, feldolgozza őket, és ezzel segít a helyes döntések meghozatalában. A készülék más szenzorokkal való összekapcsolásával és a környezet folyamatos monitorozásával akár baleset-megelőzésre is lehetőség nyílik. Különösen fontos lehet ez a gépjárművek esetében. Számos beépített alkalmazás segít a vezetés megkönnyítésében vagy biztonságossá tételében, például a gépjármű szenzorok felhasználásával. Sok esetben ezek korlátozottan elérhetőek szélesebb körben, így felmerül az igény egy olcsó és sokak számára elérhető megoldásra.

Minden évben a személyi sérüléssel járó balesetek akár 3-4%-a a vezető elalvása miatt következik be, ha a gépjármű vezetője elalszik, nem csak saját, de a mellette ülők, valamint a forgalom többi résztvevőjének testi épségét is veszélyezteti. A legtöbb hasonló jellegű baleset megelőzhető lett volna, ha a vezető tisztában van a kialvatlanság veszélyeivel. Fontos a tájékoztatás és megelőzés. Fáradtan volán mögé ülni legalább akkora kockázat, mint alkoholos állapotban vezetni. De míg az alkoholfogyasztás tekintetében vizsgálatokkal igazolható a véralkohol szint, a lebontási sebesség kiszámítható és a káros hatása megállapítható, addig sajnos az alváshiányból fakadó teljesítmény-csökkenés nehezebben kimutatható, de ugyanolyan képesség-romlással jár. Sajnos ezt az érintett személy éppúgy nem képes megítélni, mint az alkoholfogyasztás esetében. Nem egy termék foglalkozik a probléma megoldásával, némelyik a fej lecsuklását, vagy a vezetési tempót vizsgálja, vagy csak kék színnel világít, de általában vagy drágák, vagy hatástalanok.

Mindemellett a mobil eszközök nagy hátránya gépkocsivezetés közben, hogy mind a két kezünk foglalt, így a készülék nem egykönnyen hozzáférhető, valamint balesetveszélyes is lehet. A digitális kamerák és a számítástechnika fejlődésével lehetőség nyílik, hogy az elektronikai eszközeinket ne csak a hagyományos egér és billentyűzet páros, gombok, vagy érintőképernyő segítségével használhassuk, hanem

például arcmimikával, gondolatokkal, mozgással vagy kézjelekkel. Ilyen helyzetben egy olyan alkalmazás, mely a készülék érintése nélkül is megfelelően használható és sikeresen ébren tartja a vezetőt, komoly társadalmi problémát oldana meg.

A fejlesztés célja egy olyan android alapú alkalmazás megvalósítása, mely sikeresen felismeri a kézjeleket, és bizonyos időközönként ezek felmutatására kéri a vezetőt, tesztelve az éberségét, mérve a reakcióit, folyamatos megfigyelés alatt tartva megállapítja a vezető állapotát és kritikus szint alatt javaslatokat tesz a szükséges pihenőt illetően.

A dolgozat második fejezetében áttekintjük a vezetés éberségre gyakorolt hatásait és fellelhető baleset megelőző megoldásokat, végigvesszük a kéz felismerésére szolgáló algoritmusokat valamint adatelemző módszereket.

A harmadik fejezetben a tervezési fázist és a feladatok megosztását specifikáljuk.

A negyedik fejezetben az ébren tartó alkalmazás megvalósítását ismertetjük, a felhasználó felület és logika ismertetését, az adatok gyűjtését, mentését és szervernek történő továbbítását.

A dolgozat ötödik fejezetében különböző kézfelismerési algoritmusokat hasonlítottunk össze és valósítottunk meg, az érintés nélküli kezelés érdekében.

A hatodik fejezet a tesztadatok gyűjtését és részletes elemzését mutatja be, különböző statisztikai és elemző módszerek alkalmazásáról és kiértékeléséről.

Végül az összegzés során ismertetjük az elvégzett feladatokat és jövőbeli terveinket.

2 Irodalomkutatás

2.1 Baleset megelőzés

Amerikában több mint 40.000 ember szenved súlyos sérülést az 56.000 alvással kapcsolatos közúti balesetből évente. Kimutatták, hogy a monoton utakon történt balesetek 20%-áért felelős az álmosság [21]. Eközben Magyarországon évente átlagosan 2500-2800 baleset történik és ezek közül 1200-1300 halálos kimenetelű, melyek 3-4%-áért egyértelműen a sofőr elalvása tehető felelőssé. A 18-24 évesek 15%-ával történt már baleset amiatt, hogy túl fáradtan vezettek.[11]

A továbbiakban megvizsgálom milyen kimutatható élettani jellemzők vezetnek tragédiához, illetve milyen módszerek állnak rendelkezésre a megelőzéshez.

2.1.1 Vezetés éberségre gyakorolt élettani hatásai

A fáradt ember számára agyunk bizonyos időközönként újra meg újra felkínálja az alvás lehetőségét, ezért amikor a vezető elalszik, csak pár másodpercre, úgynevezett mikroalvásba csúszik bele, ez bármikor, bárkivel megtörténhet. A balesetek száma kiugró értékeket mutat a hajnali 1 és 9 óra közötti intervallumban, valamint a kora délutáni órákban. Ez a két időszak egybeesik azokkal a 24 órán belüli periódusokkal, amikor az ember kumulálódó alváskészsége a legnagyobb, és ezzel az elalvás veszélye is.[14]

2.1.1.1 Alvás és alvási fázisok

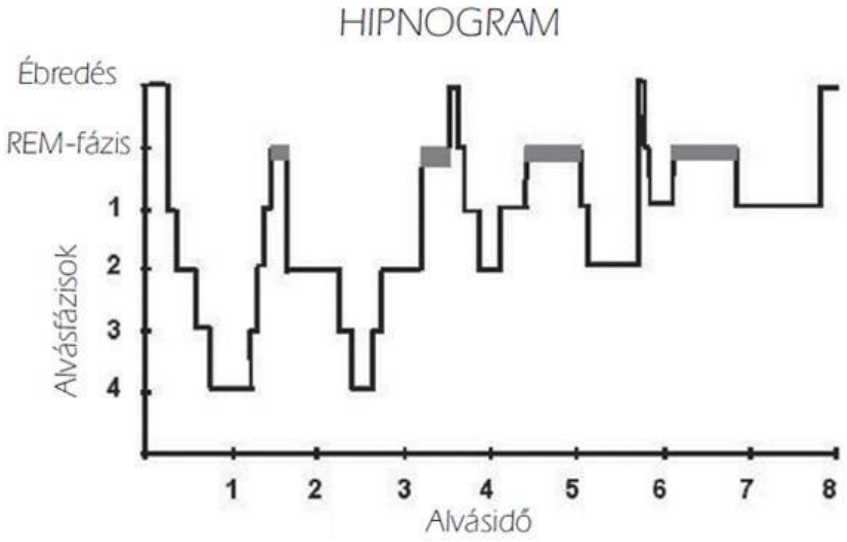
A vezetés közben történő elalvások megértéséhez szükséges áttekinteni, miért olyan nélkülözhetetlen része az alvás életünknek. Az alvás elengedhetetlen a test regenerálódásához, a napközben szerzett új ismeretek rendszerezéséhez. Megfelelő mennyiségű és minőségű alvás nélkül fáradtak leszünk, a koncentrációs képességünk romlik, reakcióidőnk csökken, ami különösen veszélyes vezetés közben. Nem leszünk képesek elhárítani a veszélyes helyzeteket, nem jutnak eszünkbe dolgok.[15]

A kimerültség az alkoholfogyasztással egyenértékű rizikófaktor vezetés közben. 17 órás ébrenlét után eléri az EU több országában megengedett 0.05% véralkohol szint hatását. 24 órás ébrenlét után ez meghaladja a 0.1%-ot, ami közepes alkoholos befolyásoltsághoz hasonlóan rontja a cselekvőképességet. Sajnos az alváshiányból adódó cselekvőképesség-csökkenés jóval nehezebben kimutatható és az érintett személy

a képességromlást ugyanúgy nem képes megítélni, mint alkoholos befolyásoltság esetén. A tartós alvásmegvonás a homloklebeny funkciózavarát okozza, amitől a személy agresszívvá, ingerlékennyé, féktelenné válhat, mielőtt a cselekvési zárlatot okozó mikroalvások megjelennének, ami bár segíthet az előrejelzésben, további baleseti kockázatot hordoz magában.[14]

Alvási ciklusunk illeszkedik a Föld forgásához kapcsolódó nappalok és éjszakák váltakozásához. Éjszaka kb. másfél órás ciklusokban alszunk, melyen belül megfigyelhetünk az éjszaka első felében jellemzően mély alvást, majd a második felében felületes, valamint álomalvást. Az ábrán látható 4 fázist különböztethetjük meg (2.1 ábra):

- szendergés (1. stádium),
- felszínes alvás (2. stádium),
- mély alvás (3-4. stádium),
- álomalvás (azaz REM alvás).[15]



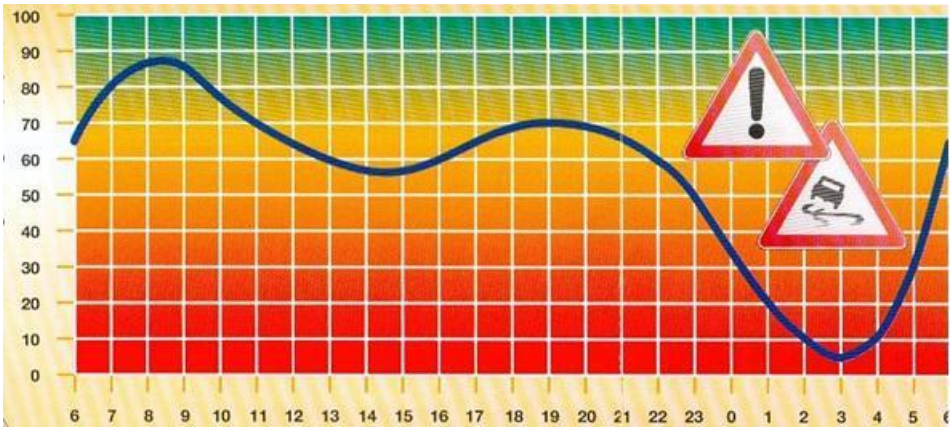
2.1. ábra: Alvási ciklus fázisai

2.1.1.2 Mikroalvások

A szervezet minden, nagyjából másfél órás szakasz után lehetőséget kap, hogy a test állapotából következtetve eldöntse szeretne-e váltani alvás és ébrenlét között. Amennyiben ebben az időszakban ébredünk, sokkal kipihentebbnek érezzük magunkat. Ezek a másfél órás ciklusok nappal is megfigyelhetőek, de a legtöbb esetben teljesen észrevétlenek, bár agyunk minden esetben „megkérdezi” akarunk-e aludni vagy ébren maradnánk, illetve fordítva. Az alváskényszer nagyságától függ milyen könnyen

leküzdhető. A ciklus végén átéljük a néhány perces álomkaput, ha ezt sikerül legyőznünk, a következő ciklusig ébren maradhatunk ugyan, de fáradtak leszünk és a reflexeink tompulnak. Az összegyűlt alváshiány csak két egymás utáni éjszakai alvással szüntethető meg.[16]

Egy nap során több időszakot különböztethetünk meg. A teljesítőképesség emelkedik reggel 3-tól 9 óráig, ahol is eléri a maximumot. 13-16 óra között visszaesik a koncentráció, majd 17-21 között újra éberebbek leszünk, végül 22 órától gyorsan zuhan (2.2 ábra).[17]



2.2. ábra: Teljesítőképesség napi változása

Ez a két lokális minimum a legveszélyeztetettebb időszak. Ekkor a másfél órás periódus végén erős fáradtság tör ránk, amikor szervezetünk felajánlja az alvás lehetőségét, ilyenkor következhetnek be mikroalvások, ami vezetés közben akár végzetes is lehet. Ez akár 30 másodpercig is eltarthat, a járművezető ugyan lát, de amit lát, nem képes tudatosítani, vagy reagálni az eseményekre. Az esetek nagy részében észre sem veszi őket. Ekkor a jármű irányítás nélkül sodródik [21]. Ezt az időszakot a legfontosabb felismerni és megállni pihenni, mozogni egy kicsit, könnyen felszívódó cukrot vagy kávét magunkhoz venni, de akár egy rövidebb 10 perces alvás is megoldhatja a problémát.[17]

A fáradtság alapvető jeleit a vezető maga is felismerheti: szemhéj nehézség, szájszárazság, szomjúság, hidegrázás, szem dörzsölés, kettős látás, alagútszerűen szűkülő látótér, érzékelési zavarok. Persze ezt sokkal egyszerűbb utasként észrevenni, jellemző lehet a vezetési stílus megváltozása (pontatlanul kuplungol, vált sebességet és hirtelen fékez), vezetés közben nincs gondolatai tudatában, jelentős tempónövekedés észlelhető, agresszív vezetési stílus, nehézség az úttartásnál.[17]

2.1.1.3 Reakcióidők vezetés közben

Felmérések során meghatározták a várható vezetés közbeni reakcióidőt kor és tapasztalat függvényében. Ez a 42-46 éves korosztálynak a legkisebb, akik már 17-21 éves vezetői tapasztalattal rendelkeznek. Ebből látszik, hogy bár idősebb korban a reakcióidő elméletileg növekszik, a vezetési tapasztalat mégis sokban segíthet a balesetek elkerülésében.[24]

Egy japán tanulmányban meghatározták, hogy egyrészt mennyi idő telik el az észleléstől a gáz lenyomásáig, álló helyzetből indulva, másrészt veszély észlelésétől a fékre taposásig mozgás közben. Mindezt valós és szimulált környezetben is, miközben olvasási és ismétlési feladatokat végeztetnek az alanyokkal. A legjobb fékre taposási eredmény is 700 ezredmásodpercnél nagyobb volt, a gáz megnyomásakor 800-ról 1700-ra is nőhetett a reakcióidő feladatmegoldás közben. Kimutatták, hogy a személyes tulajdonságok is nagyban befolyásolják a reakcióidőt.[25]

Egy 2015-ös kísérlet során megállapították, hogy a reakcióidő nem függ jelentősen a fék-gáz konfigurációtól, a kor előrehaladtával azonban jelentősen nő, és legrosszabb a mobilt használó idősebb nőknél.[26]

2.1.2 Elalvás megelőzést segítő technológiák

Gépjárművezetés közben történő elalvás problémájának megoldására több megoldás is született. Egyre gyakrabban találkozhatunk a különböző gépkocsikba épített vezető monitorozó rendszerekkel. Ezek nagy hátránya, hogy igen drágák, így nagyon szűk réteg számára érhetőek csak el. Találkozhatunk pár olcsóbb eszközzel is, de ezek sem nyújtanak minden esetben megfelelő eredményt. Az intelligens eszközök elterjedése komoly perspektívát nyit, hogy a biztonság elérhető legyen szélesebb rétegek számára is, valamint más eszközökkel, kamerával, pulzusmérővel, fedélzeti rendszerrel történő összekötésével egyszerűen megoldható a széleskörű információgyűjtés.

2.1.2.1 Gépjármű monitorozása

A modern autók fedélzeti rendszere rengeteg lehetőséget nyújt az adatgyűjtésre. Megfigyelhető az üzemanyag fogyasztás, a kormány mozgás és szögelfordulás, a sebesség és a gyorsulás változás, a gázpedál lenyomásának szintje, a fékezések, helyadatok. Ezen adatokból sok információ nyerhető a vezetési stílusról, a jármű

teljesítményről, valamint a vezetőről, többek között a fáradtsági és izgalmi szint is megbecsülhető.[27]

A Mercedes fáradtság figyelő rendszere az autós viselkedését figyeli, elsősorban a kormányoszlopba épített különleges, szögelfordulás-érzékelő segítségével. Amennyiben a kormánymozdulatok eltérnek a korábban megszokottól, hangjelzéssel és a műszerfalon megjelenő kávés csészével jelez (2.3. ábra).[19]

2.3. ábra: Mercedes kávés csésze

2.1.2.2 Külső környezet monitorozása

Általában a járműbe beépített külső kamerákkal és érzékelőkkel lehetséges, elsősorban sávelhagyást és követési távolságot mérnek. De fontosak lehetnek a lekérdezhető információk is, az időjárás, az útviszonyok és a forgalom is.

A Volvo elalvás figyelő rendszere az utat, vagyis a kontrollálatlan mozgást ismeri fel és jelez hangjelzéssel, valamint a kávés csésze ikonnal. A Citroën magasabb felszereltségű autói szintén sávelhagyást figyelnek és az autó hátuljának rezgetésével figyelmeztetnek. A Honda rendszerében sávelhagyás esetén a kormány nehézzé válik, így a rossz irányból visszaterel a megfelelő sávba. Sajnos ezek mind nagyon drága technológiák.[19]

2.1.2.3 Vezető megfigyelése

Egy, a balesetek okait felderítő felmérés szerint, mely 2000, 19-69 év közötti alannyal 19.9 éves átlagos vezetői tapasztalattal készült az utóbbi években elszenvedett baleseteikről, az első ok a sietség (22%), majd figyelemelterelés (21.9%), normál körülmények között (15%), tapasztalatlan vezető (8%), fáradtság (4%). Ebből látható milyen fontos az emberi tényező, a vezető monitorozása, annak megállapítása, hogy

mikor vonja el bármi a vezető figyelmét, legyen az beszélgetés, elmélázás vagy fáradtság. Lehetséges a külső látható jegyek megfigyelése, a belső biológiai jelek vizsgálata, valamint a vezető aktív tesztelése, a reakciók vizsgálata.[28]

2.1.2.3.1 Vezető külső megfigyelése

A vezető külső megfigyelése során általában a kamerás megfigyeléseken van a hangsúly, de érzékelők segítségével a mozgás is monitorozható. Egyértelműen fáradtságra utaló jegyek, a szem lecsukódása, a pupilla és az arckifejezés változása, a fej dőlésszöge illetve előrebukása, az ásítás. Ezek kamerák és érzékelők segítségével mind megfigyelhetőek. Az éjjeli megfigyelés pedig infra kamerákkal oldható meg.

Több autógyártó is telepített, kamerákból álló biztonsági rendszert a járműveibe. A német Fraunhofer Intézet olyan, több kamerából álló rendszert javasol telepíteni a gépjárműbe, mely jelzi a sofőrnek amennyiben a szem lecsukódása egy pillanatnál hosszabbra nyúlna. A japán Denso ezzel szemben a megelőzésre koncentrálna. Az arcizmokat 17 helyen figyelő és elemző program segítségével detektálná, ha a vezető túl közel kerülne az elalváshoz.[18]

A Lexus modelljeihez kérhetőek extra kamerák, melyek figyelik a vezetőt, és ha bóbiskolást érzékelnek, hangjelzéssel és biztonsági öv rángással figyelmeztetnek (2.4. ábra). Ha a külső kamera egyidejűleg akadályt lát, az autó vészfékez.

2.4. ábra: Lexus elalvás figyelő rendszer

Fellelhető egy olyan, fülre illeszthető szerkezet is, mely a fej előrebillenését vizsgálja és sípol, ha bólintunk. Bár olcsó, de komoly hátránya, hogy már csak a

legutolsó pillanatban jelez, valamint ha a kocsiban sikerül megtartani a fejünket, vagy hátrafele hanyatlik, úgy nem alkalmazható (2.5. ábra).

2.5. ábra: Bólintás vizsgáló

2.1.2.3.2 Vezető biológiai vizsgálata

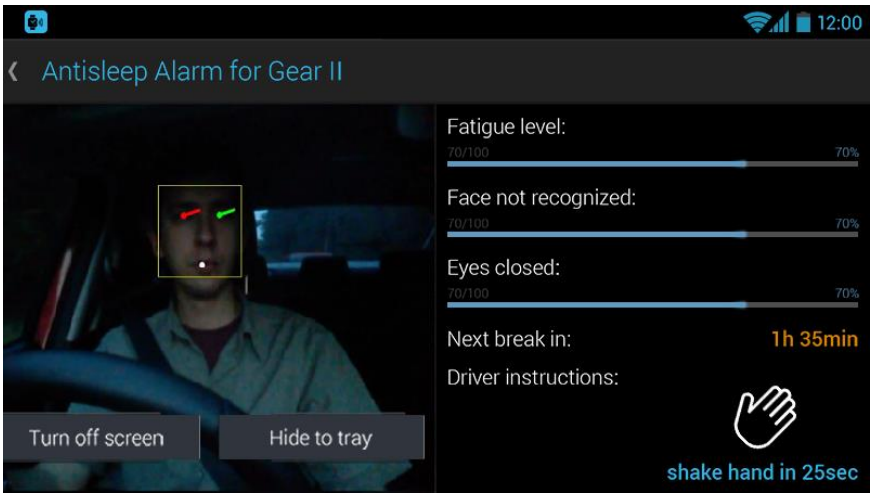
A Massachusetts-i Műszaki Egyetemen végzett tanulmány szerint elsődleges, hogy az adatokat több oldalról gyűjtsük és megvalósuljon mind a vezető, gépjármű és környezet megfigyelése. A korábbiakat egyesítve kiterjesztik a vezetőről gyűjtött biológiai adatokkal, és ezeket összevetve próbálják a fáradtsági és stressz szintet meghatározni és visszajelzéssel a helyes viselkedésre ösztönözi a vezetőt. Széleskörűen gyűjti a biometrikus adatokat: megfigyeli a szívritmust, a vérnyomást, a bőr vezetőképességét, a légzést, az agyhullámokat. Viszont a mérés nagyon körülményes a különböző elektródákkal és érzékelőkkel.[29]

Az Aichi japán egyetemen végzett kutatás a baleset-megelőzés során szintén az emberi tényezőre koncentrál, pontosan meghatározták, hogy a biológiai, külső és viselkedésbeli tényezők (fej dőlésszög, pupilla, pulzus, gáz nyomvatartása) mikor jeleznek fáradtságot, majd mintafelismeréssel és tanuló algoritmussal sorolják be a mintákat, és kapják meg a felismerés pontosságát. A 4 típusú minta együttes alkalmazásával sikerült 91%-os bizonyossággal sikeresen besorolniuk a vezető állapotát.[28]

2.1.2.3.3 Vezető reakciók tesztelése

A passzív megfigyelésen kívül, a vezető aktív tesztelése során a reakciókról is képet kaphatunk. Anti Sleep nevű Android alkalmazás szabálytalan időközönként 1 perces sípszót ad ki, amit a stop gomb megnyomásával lehet leállítani, közben méri a reakcióidőt. Folyamatos koncentrálást igénylő feladatoknál meglehetősen hasznos, de pont a járművezetők számára kevésbé, mivel akadályoztatva van a készülékhez való hozzáférés.[13]

Az Anti-Sleep Alarm egy Samsung Galaxy Gear típusú okos óra segítségével próbál ébren tartani. Kezdetben az alkalmazás feltesz egy sor kérdést a vezetőnek, mennyit aludt utoljára és milyen fáradtnak érzi magát. Ha telefonnal is rendelkezik, akkor az előlapi kamera a vezető arcát fogja monitorozni és a szemmozgásból próbálja megbecsülni a fáradtság mértékét (2.6. ábra). A vezetőnek időről időre meg kell ráznia a kezét, ha elfelejti, akkor az óra lágy rezgéssel figyelmezteti. Ezután a méréseket összegezve tanácsokat ad a vezető számára a további teendőket illetően.[22]



2.6. ábra: Anti-Sleep Alarm for Gear II

2.2 Érintés nélküli eszközkezelés

Mint látható, nagyon fontos, hogy a járművezetők éberségi szintjét fel tudjuk mérni, ám az is lényeges, hogy ezt valamilyen olcsó módszerrel tegyük, úgy, hogy közben nem akadályozzuk az alanyt a vezetésben. Ezek alapján a reakcióidő mérése egy kiváló megoldásnak tűnik. Hogy ezt ne törvénybe ütközően tegyük (jogszabályilag tilos hozzányúlni a telefonhoz vezetés közben Magyarországon és több másik országban is), érintés nélküli kommunikációval oldanánk meg a problémát, melyre a legjobb megoldást, az előre meghatározott kézjelek felismerése jelenti.

2.2.1 Kézdetektáló technológiák

A kézjelek felismerésére az évek során több módszert is kidolgoztak, ezek között vannak olyanok, melyek mára elavulttá váltak, de sokban hozzájárultak a terület fejlődéséhez. A kezdetekben még külső szenzorok segítségével próbálták meghatározni a kézjeleket, majd a kutatások szorosan hozzákapcsolódtak a digitális kamerák és egyéb digitális szenzorok fejlődéséhez.

2.2.1.1 Kesztyűs megoldás

A legelső gesztusfelismerő rendszerek valamilyen külső hardver segítségével működtek. Ez kezdetben egy szenzorokkal felszerelt kesztyű volt, ami a begyűjtött információk alapján próbálta meghatározni a kéz dőlésszögét, az ujjak ernyedtségét, helyzetét. Ezek a kesztyűk általában fény és fotocellák segítségével állapították meg az ujjak ernyedtségét (2.7. ábra). Minél jobban behajlott egy ujj, annál kevesebb fény jutott el a fotocellába. De akadtak olyanok is, amelyek nyomásérzékelőkkel voltak felszerelve. Az évek során az így nyert tapasztalatok, valamint a technológia fejlődésének segítségével jutottak el a tudósok a szenzor nélküli, színes kesztyűkig. Ezek kezdetben különböző színekkel jelölték a kézfej különböző részeit. Mivel az ilyen fajta vizuális gesztus felismerés sokkal kényelmesebb és olcsóbb is, kiszorította a nem vizuális felismerést és manapság ebben az irányban folynak tovább a kutatások. A technológia fejlődésének hála, már elég, ha egy egyszínű kesztyűt használnak a felismeréshez, így még kevesebb feltétel kell a rendszer fejlesztéséhez. (2.8. ábra) [7]

2.7. ábra: szenzoros kesztyű

2.8. ábra: színes kesztyű, kézjel felismeréshez

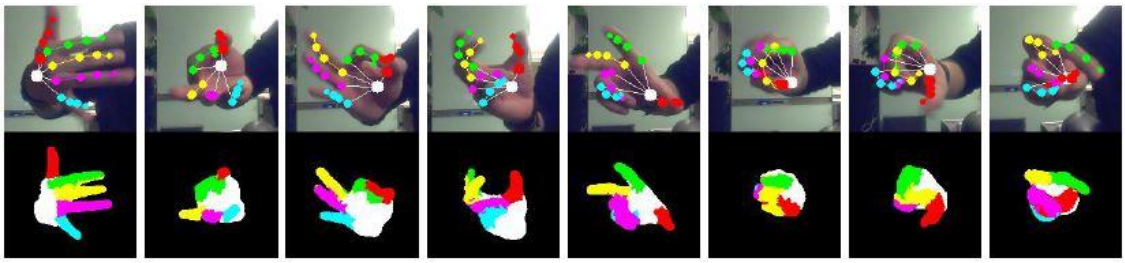
2.2.1.2 Szín szerinti szűrés

A színes kesztyűkkel történő kísérletezés megalapozta az utat, a kesztyűmentes, de továbbra is szín alapján történő felismerés előtt. A digitális kamerák fejlődésével ezek a programok egyre biztatóbb eredményeket mutatnak. A programok a bemeneti

képen megkeresik a bőrszínű területeket és ezeket különítik el, majd dolgozzák fel, hogy végül csak a kéz maradjon. Mivel a digitális kamerák nagy többsége az RGB színtér segítségével állítja elő a képet, ami érzékeny a fényerősség változásra, ezért az alkalmazások a kapott RGB kép értékeit egy másik, általában a HSV, színtérbe transzformálják át, így pedig már meg lehet határozni biztosabb határértékeket a bőrszín szűrésére.

2.2.1.3 Sztereó kamera – 3d-s kép

Az elmúlt években egyre több olyan eszközt fejlesztettek ki, amelyek egyszerű kamerán kívül más érzékelőt is tartalmaznak, például több kamerát vagy infravörös mélység érzékelőt. Ezek az extra információk sokkal könnyebbé teszik a kamera képének elemzését, valamint új lehetőségeket is megnyitnak a képfeldolgozáshoz. Két kamera esetén például a kamerák képeit összevetve kiszámítható a diszparitás, vagyis a két kép közötti eltérés, melynek segítségével mélységadatok nyerhetőek ki a képből. Ezek segítségével térben lehet modellezni a képen látható tárgyakat. Ez a módszer az emberi képalkotáshoz hasonlóan működik. Ha a kamera mellett egy mélység érzékelő szenzor is található, ennek segítségével egyszerűen rendelhetünk mélységadatokat a kamera által készített kép minden pixeljéhez. Egy mélységadatokkal rendelkező kép esetében már egyszerűbb dolgunk van a háttér és az előtér külön választásával, vagy az egyes tárgyak körülhatárolásával. Többek közt az emberi test részeinek meghatározása, így a kézé is, könnyebbé válik, ami sokkal pontosabb kézjel felismerést tesz lehetővé. (2.9. ábra) Ezek az eljárások nem igénylik semmilyen tárgy viseletét, valamint teljes mértékben függetlenek a képen látható színektől, ezért sokkal kényelmesebbek és jobb eredményeket nyújtanak mint a társaik.[8] [10]



2.9. ábra: Mélység érzékelővel ellátott kamera által követett kézjelek, az egyes ujjak külön színnel vannak jelölve

2.2.2 OpenCV

OpenCV (Open Source Computer Vision Library) egy nyílt forráskódú számítógépes könyvtár, amely a képfeldolgozásra és a gépi tanulásra lett kifejlesztve. Az OpenCV projektet az Intel kezdte el kifejleszteni, jelenleg pedig az Itseez felügyeli (2.10. ábra). A könyvtár használata széles körben elterjedt, egyéni fejlesztőktől kezdve a technológiai óriásokig, mint például a Google, a Toyota vagy a Microsoft, sok helyen használják. Ez többek közt a könyvtár BSD-licencének, valamint a platform támogatásnak köszönhető, ugyanis az OpenCV könyvtár használható Windows-on, Linux-on, Android-on és Mac OS-en, valamint támogatja a C++, C, Python, Java és MATLAB nyelveket. [12]

2.10. ábra: Az OpenCV és az Itseez logói

2.2.3 Algoritmusok

Az OpenCV keretrendszer több ezer képfeldolgozásra alkalmas függvényt tartalmaz. Ezek közül mutatok be most néhányat, melyeket az alkalmazás implementálása során használtam.

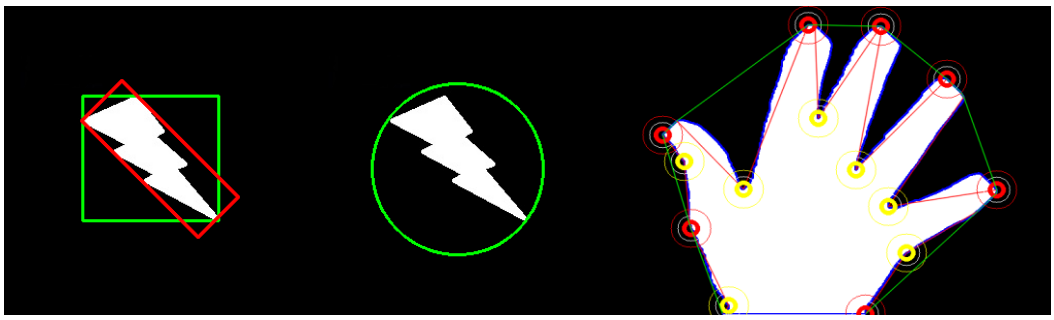
2.2.3.1 Otsu algoritmus

Az Otsu algoritmust a feltalálója, Nobuyuki Otsu után nevezték el. Az algoritmus szürkeárnyalatos képeket alakít binárissá (fekete-fehér), határérték számítás után. (2.11. ábra) A legtöbb thresholding függvény egy megadott érték alapján alakítja binárissá a képet, ezzel szemben az Otsu algoritmus megkeresi azt a határértéket, amelyet használva minimális a szórás az előtér (fehér) és a háttér (fekete) pixelei között. Ezt úgy éri el, hogy az összes lehetséges határérték esetén kiszámolja az előtér és a háttér közötti szórást, majd kiválasztja a legkisebb értékkel rendelkező határértéket és ez alapján végzi el a kép transzformációját. [20]

2.11. ábra: Az Otsu algoritmus a szürkeárnyalatos képet binárissá alakítja, a kiszámolt határérték alapján

2.2.3.2 Körvonal keresés

A „findContours” függvény megkeresi az alakzatok körvonalait egy bináris képen. Miután megtaláltuk a képen található objektumok kontúrját, lehetőségünk van több másik függvényt alkalmazni, melyek az egyes kontúrokat felhasználva szolgáltatnak információkat az adott objektumról. A függvények segítségével kiszámolhatjuk a legkisebb téglalapot, négyzetet vagy kört, amely körbehatárolja az alakzatot, kiszámíthatjuk az alakzat súlypontját, vagy vizsgálhatjuk az alakzat konvexitását, megkereshetjük a konkáv pontjait (2.12. ábra). [12]



2.12. ábra: Az alakzatok findContour függvény által megtalált kontúrjának felhasználásai: legkisebb körülölelő téglalap és négyzet kiszámítása (bal), legkisebb körülölelő kör kiszámítása (közép), egy kéz konvex alakzatának megkeresése és a hibapontok megtalálása (jobb)

2.2.3.3 Távolság transzformáció – Distance transform

A „distanceTransform” függvény egy bináris (fekete-fehér) képen kiszámolja a fehér pixelek távolságát a legközelebbi fekete pixeltől és minden pixel esetében egy 0 és 1 közötti számot ad eredményül. Minél nagyobb a távolság, annál közelebb van az adott érték 1-hez. Az egész képen lefuttatva a függvényt, egy két dimenziós mátrixot kapunk eredményül. Ha megkeressük a lokális szélsőértékeket a mátrixban, akkor megkapjuk a képen található alakzat „csontvázát/skeleton-ját” (2.13. ábra).[12]

2.13. ábra: A distanceTransform függvény eredménye egy kéz esetében

2.3 Éberségi szint meghatározás

A hatalmas információmennyiség leginkább annak köszönhető, hogy az elmúlt évtizedek informatikai fejlődésének hatására egyre jobban életünk része az automatizáltság, elektronikus úton való dokumentáltság. Gépjárműben is, az autó szenzorai és intelligens készülékek segítségével minden lépésünk nyomon követhető, fontossá válik a hatalmas adatmennyiség összegyűjtése, statisztikai elemzése, értékelése és az információ kinyerése. Az adatokból kinyert információnak pontosnak és statisztikailag szignifikánsnak kell lennie, a feltárt tudásnak az adott elemzés szempontjából hasznosnak is kell lennie. [6]

2.3.1 Adatgyűjtés

Az intelligens készülékek terjedésével egyre szélesebb körben nyílik lehetőség az adatgyűjtésre, az *Android* napjaink egyik legnépszerűbb, nyílt forráskódú, Linux-kernel alapú mobil-operációs rendszere. A rendszert futtató eszközök száma a mai napig növekszik. Népszerűségének oka az ingyenessége és rugalmas alkalmazkodása a legkülönfélébb eszközökhöz.[1]

2.3.1.1 Android platform

A fejlesztők számára két programozói felületet is biztosít, egy *Java* nyelvű menedzselt környezetet és egy C/C++ nyelvű natív környezetet. Az elkészült alkalmazások a *Dalvik* virtuális gépen futnak, melynek feladata, hogy egységes, platformfüggetlen környezetet biztosítson az Android alkalmazásoknak. Itt érhető el az operációs rendszer által nyújtott nagyszámú szolgáltatás. Az alkalmazásoknak több belépési pontja is lehet, valamint háttérbe kerülve futásuk felfüggesztődik, vagy akár

teljesen le is áll. Az alkalmazások kezelésének viszont jelentős erőforrás költségei vannak. A problémára az Android natív programozói felülete kínálja a megoldást.[2]

Az intelligens eszközöknek köszönhetően a vezetőről és vezetésről is számos információ nyerhető kamera, gyorsulásérzékelő, GPS segítségével.

2.3.1.1.1 Android NDK

Az Android alapú szoftverfejlesztés elsősorban a Java alapú fejlesztést célozza meg, de mivel a mobil eszközök erőforrásai igen korlátozottak, amennyiben szeretnénk olyan alkalmazást készíteni, melynek nagy az erőforrás igénye, akkor igénybe vehetjük az Android Native Development Kit (Android NDK) szoftvercsomag nyújtotta lehetőségeket.[2] Az OpenCV keretrendszer használatát Android-on, pontosan a lehetséges nagy erőforrás igény, valamint a speciális hozzáférések miatt, csak is az Android NDK segítségével lehet megoldani.

Az NDK nyújtotta környezetben C vagy C++ nyelven írhatunk különféle „shared library-ket” amiket a Java kódnyelvű programunkból érhetünk el. Ennek segítségével tudjuk a virtuális gép okozta felesleges terhelést megszüntetni.[4]

Használatához elengedhetetlen az Android Software Development Kit (Android SDK) megléte.

A natív rész egy önmagában fordítható egységet alkot, a fordítás eredménye egy „shared object”, vagyis .so fájl, ami egy dinamikusan betölthető bináris könyvtár, ezt tölti be az alkalmazás a Java kódból. A Java és natív részek közti kommunikációt a Java Native Interface (JNI) teszi lehetővé.[4]

2.3.1.1.2 Java Native Interfész

A JNI biztosítja a felületet, melyen keresztül a natív kód meg tudja szólítani a virtuális gépet. A Google a különféle szoftververziók kiadása során folyamatosan bővítette az NDK által nyújtott szolgáltatásokat, így már OpenGL-es támogatás is elérhető rajta. A billentyűzet kezelése, a fájlok elérése és a rajzolás feltétlenül natív hozzáférést igényelnek.[2]

Kezdetben csak a Java virtuális gép (JVM) tudta megszólítani, de mára az alkalmazások számára is elérhetővé vált. Java kódból a natív függvény egyszerűen meghívható, csak deklarálni kell és a *native* jelzőt elé írni. Innentől kezdve a rendszer felismeri és függvényhíváskor az implementációt a natív részben keresi. Nincs fordítási

idejű ellenőrzés, így elgépelés esetén csak a függvény első hívásakor, vagy futási időben derül ki a probléma.[2]

2.3.1.2 Jármű, mint adat forrás

A mai személygépjárművek nem egyszerű közlekedési eszközök, hanem folyamatosan nagy mennyiségű információt gyűjtenek belső és külső szenzoraik segítségével, többek között a környezetről és az utasokról, de elsősorban a vezetőről.

2.3.1.2.1 OBD-II

Az OBD-II (On-Board Diagnostics) fedélzeti diagnosztikai rendszer egy márkafüggetlen, szabványos protokollon alapuló hibakereső interfész, melyet a gépjárművek karbantartására fejlesztettek ki. 1996 óta minden autónak kötelező alap felszereltsége az USA területén, majd 2003 óta az EOBD (az OBD európai változata) Európában is. Bár ez soros port vezetékes, léteznek hozzá különböző Bluetooth adapterek, amelyek segítségével bármilyen eszközzel összekapcsolhatóak.[9]

Az OBD-II port segítségével a gépjármű számos paraméteréhez és szenzor információhoz férhetünk hozzá, többek között elérhetővé válik a motor fordulatszáma, káros anyag kibocsátás, belső és külső hőmérséklet, légnyomás, sebesség, fogyasztás, valamint a hibakódok sorozata. A VehicleICT Android alapú platform különböző fedélzeti diagnosztikai rendszerek kezelését könnyíti meg.[9]

2.3.1.2.2 VehicleICT Platform

A VehicleICT platform lehetővé teszi az android fejlesztőknek az autóhoz történő csatlakozást és számos különböző szolgáltatást, a szenzor információk kinyerését, a megjelenítést, háttér infrastruktúrát és adatfeldolgozást.[9]

A platform keretet biztosít a szenzor adatok eléréséhez, csatlakozik az OBD II vagy CAN buszhoz bluetooth kapcsolaton keresztül. A kliens platform a felhasználó eszközén fut a háttérben, ehhez csatlakozik a kliens alkalmazás, mely bizonyos időközönként megkapja a jármű által szolgáltatott adatokat.

2.3.2 Adatelemzési módszerek

Az adatelemzés során szükséges az adatok meghatározása, a nyers adatok tisztítása, számítások, statisztikák elvégzése, módszerek használata (valószínűségi eloszlások, idősorelemzés, klaszterezés, minták felderítése), eredmények megjelenítése

és értékelése, összehasonlítása a korábbi adatokkal. A cél az adatsor tartalmának olyan formában történő reprezentálása, mely megkönnyíti a vizsgált témakör szempontjából releváns részek felismerését. Ez történhet lényeg kiemeléssel (összesítő adatok), illetve formátum átalakítással (grafikus megjelenítés). A megfelelő elemzés releváns és kellően nagy adatmennyiségen nyugszik. Az elemzések rendszerint statisztikai alapokon állnak, a gyakorlatban több módszert is ki kell próbálni a helyes következtetés levonásához. A cél a jó döntés meghozatala bizonytalan információk mellett is.

2.3.2.1 Adatok kiválasztása

Statisztikai sokaságnak nevezzük a statisztika tárgyát képező egyedek összességét. Alapvetően két típusát szokás megkülönböztetni, beszélhetünk álló és mozgó sokaságról. Az álló sokaság állapotot fejez ki, élőlényekből, tárgyakból állhat egy bizonyos időpontra vonatkoztatva, a mozgó sokaság időtartamra értelmezhető. A megfigyelés a sokaságnak csak meghatározott egyedeire terjed ki, részleges megfigyelést végzünk. Reprezentatív megfigyelés során megpróbálok információt szerezni a meghatározott legfontosabb tényezők szerint, mint különböző alanyok, különböző időben (illetve fáradtsági szinten), valamint helyzetben. A részleges, de különösen a reprezentatív megfigyelés során célunk az alapsokaság egy részének, a mintasokaságnak a segítségével a teljes sokaságra vonatkozó következtetések megfogalmazása.

2.3.2.2 Statisztikák számítása

A statisztikai adatokhoz mérés vagy számlálás útján jutunk, megkülönböztetünk abszolút és származtatott adatokat. Utóbbihoz az abszolút adatokkal végzett műveletekkel jutunk, ami fontos az adatok összességének áttekintéséhez, összehasonlításához. A középértékek (nevezetes közepek, módusz és medián), szórás, átlagos eltérés az azonos fajta adatok tömegének számszerű jellemzője. A kvantilis értékek a mennyiségi ismérv értékeinek rendezésére szolgálnak, néhány számszerű információ alapján segítik az eligazodást. Ezzel tömör jellemzést adhatunk az adathalmazról, egyetlen számadatba sűrítve a jellemzőiket.[23]

2.3.2.3 Ismérvek kiválasztása, kapcsolatvizsgálatok

A statisztikai sokaságot statisztikai ismérvek szerint csoportosíthatjuk, ezek lehetnek mennyiségi, minőségi, időbeli ismérvek. Kisebb adathalmazok esetén, ha ismerjük az attribútumok közötti összefüggést, akkor manuálisan is kiválaszthatjuk a lényeges attribútumokat. Amennyiben az ismérvek között tendenciaszerűen érvényesülő

kapcsolatot észlelünk, tehát az egyik ismérvváltozathoz való tartozásból valószínűségi jelleggel következtethetünk a másik ismérvváltozathoz való tartozáshoz, sztochasztikus kapcsolatról beszélünk. Korrelációs kapcsolatról beszélhetünk, ha a kapcsolatot mennyiségi ismérvek közvetítik, asszociációsról, ha mindkét ismérv minőségi.[23]

Alternatív ismérvek esetén a kapcsolat mérésére alkalmazhatjuk az ún. Yule-féle mutatót, amely a kontingenciatáblában szereplő minőségi ismérvek gyakoriságainak keresztszorzatából állítható elő. Hátránya, ha bármely elem nulla, akkor determinisztikus kapcsolatot jelez, ha az egyébként nem áll fenn, valamint kettőnél több ismérvváltozat esetén más mérőszámot kell alkalmazni.

A Cramer együttható feloldja az alternatív ismérvek problémáját, ugyanakkor érzéketlen a 0 értékekkel szemben. Alapgondolata, ha a független viszonyt feltételező gyakoriságok és tényleges gyakoriságok között eltérést találunk, akkor sztochasztikus kapcsolat meglétére kell gondolnunk.[23]

Amennyiben boole-féle változókon kívül mennyiségi attribútumokat is tartalmaznak, akkor legkézenfekvőbb megoldás ezek leképezése, kategorizálása, majd az asszociációs szabályokat feltáró algoritmus alkalmazása. Ha a mennyiségi attribútumoknak több különböző értéke van, akkor az értékeket diszkretizáljuk, és intervallumokba soroljuk.[6]

2.3.2.4 Gyakorisági sorok

A mennyiségi ismérvek alapján végzett adatrendezés, adattömörítés legelterjedtebb módja a gyakorisági sorok képzése. Természetesen a gyakoriságokból számíthatunk megoszlási viszonyszámokat, amiket ún. relatív gyakoriságoknak nevezünk. Amennyiben nagyszámú ismérvértékkel rendelkezünk, célszerű olyan osztályközök kialakítása, amelyek jól tömörítik a jelenség információtartalmát, de ugyanakkor még nem eredményeznek számottevő információveszteséget. A gyakorisági sorok poligon alakja, formája fontos információkkal szolgálhat a vizsgált jelenségről. Az egymóduszú gyakorisági sorok lehetnek szimmetrikus vagy aszimmetrikus eloszlásúak. A szimmetrikus eloszlás a normális eloszlás gyakorlati megfelelője. Az aszimmetrikus eloszlások közül a jobb oldali aszimmetriát mutató eloszlás a gyakoribb, ez esetben a mennyiségi ismérvek alsó határát szigorúbb törvények szabják meg.[23]

2.3.2.5 Idősorok elemzése

A különféle jelenségek számszerűsíthető értékei lehetőséget nyújtanak az időbeni összehasonlításra, módot teremtenek az elmúlt időszakok tendenciájának, összefüggéseinek felismerésére.

A trend meghatározása az idősorok kisimítását jelenti, egyik fő módszere a mozgó átlagok számítása. A trendet az idősor speciális, dinamikus átlagaként állítja elő. A véletlen tényező, zaj tompítása az átlagolással kiküszöbölhető. Meghatározzuk az átlagolandó értékek tagszámát (k) és vesszük az idősor első (k) értékének átlagát, ez lesz a trendadat. A következő lépésben az első adatot elhagyjuk és helyette vesszük az idősor következő értékét, majd ismételjük a műveletet az utolsó értékig.

2.3.2.6 Osztályozási feladatok

Osztályozási feladatoknál a különböző mintákat előre definiált osztályokba soroljuk. Ez egy olyan összefüggés "tanulását" takarja, mely alkalmas az ismert változók alapján az egyes minta elemeket besorolni. A kiválasztott ismérvet osztálycímkének nevezzük. Az osztályozás során a mintákat valamilyen módon szeparáljuk, így lehetséges megállapítani, hogy a rendelkezésre álló paraméterek miként határozzák meg az adott elem hova kerül. Ennek eredményét megfogalmazhatjuk osztályozási szabályok formájában. [6]

Az osztályozási feladatok során az első lépés a szabálygenerálás, kiválasztjuk az osztálycímkét, vagyis mely attribútumok, mely értékei alapján szeretnénk osztályozni az adatokat. A modell elkészítéséhez tanuló mintákat használunk fel, a rendelkezésre álló adatok egy olyan halmazát, amire ismert az osztálycímke. Az osztályozási szabályok egyszerű ha-akkor típusú szabályokként fogalmazhatóak meg. Második lépés a modell ellenőrzése, melyben az elkészített modell pontosságát ellenőrizzük. A teszt minták esetén szintén ismert az osztálycímke értéke, ellenőrizzük, hogy jó osztályba sorolja-e be, illetve milyen arányban tudta megfelelően osztályozni. A tanulás és a tesztelés iteratív folyamat melynek során ellenőrizzük a modell helyességét, míg elfogadható eredményt nem kapunk. Harmadik lépés a modell felhasználása és előrejelzés.[6]

2.4 Piackutatás

Mint az a terület mélyebb tanulmányozásából látszik, a mai napig nincsen egy, kiválóan működő megoldás sem ebben a témakörben. Rengeteg megoldási kísérlet született már és valószínűleg van jelenleg is fejlesztés alatt. A legtöbb létező technikát

bemutattuk az irodalomkutatásban, a leírásokból látszik, hogy ezek nagy része vagy nagyon költséges, vagy csak nagyon körülményesen használható. Emellett az a tendencia is megfigyelhető, hogy a legtöbb megoldás nem a fáradtan vezetés megelőzésére helyezi a hangsúlyt, hanem az álmos vagy már alvó sofőrt próbálja ébren tartani és így megakadályozni a balesetet. Sok esetben ez már túl késő lehet. Épp ezért szükség van egy széles körben használható, a megelőzés fontosságát szem előtt tartó megoldásra.

3 Tervezés

A feladat tanulmányozását követően nyilvánvalóvá vált, hogy egy olyan alkalmazást kell létrehozni, mely a kézjeleket felismerve és különböző szenzor információk alapján, képes segítséget nyújtani autóvezetők számára az éberség megőrzésére. A feladat két részre bontható:

- kézjel felismerő modul, Kárpáti Bence megvalósításában,
- ébren tartó alkalmazás fejlesztése, reakció és szenzor információ rögzítés és elemzés, Hirt Krisztina feladata.

3.1 Ébren tartó alkalmazás

Célom egy olyan alkalmazás tervezése, mely a vezetővel együttműködve és nem ellene dolgozik, segíti a mindennapokat, növeli a biztonságot, megvalósítja a vezető és jármű monitorozását és az éberségi szint megállapítását.

A reakció mérő alkalmazást Android platformon valósítom meg. Elvárt főbb funkciók és feladatok:

- kézjelek felismerése,
- reakciók mérése,
- statisztika készítése és megjelenítése,
- VehicleICT platform szolgáltatásainak segítségével és az OBD-II port által küldött minták fogadása,
- gesztus és szenzor információk feldolgozása és továbbítása,
- adatok statisztikai elemzése, éberségi szint megállapítása.

3.1.1 Felhasználói felület tervek

A felület tervezésénél elsődleges volt a felhasználói igények kielégítése, az egyszerűség és letisztultság. A következő követelményeket támasztottam az alkalmazással szemben:

- felület megtervezése és kialakítása,
- a kézmozdulatok véletlenszerű időközönkénti megjelenítése,
- a következő ébresztésig hátralevő idő progress bar-on történő megjelenítése, időzítő, kezelő gombok,

- felismerés eredményének megjelenítése,
- adatok továbbítása az adatbázisnak,
- felhasználó kezelés kialakítása,
- teszt adatok gyűjtése.

A tervezés során felmerült a nappali és éjszakai mód különböző stílusban történő megjelenítése, gyakori az éjjeli vezetés és ez esetben szükséges a kéz megvilágítása a képernyő által, erre a narancs színt választottam.(3.1. ábra)

3.1. ábra Képernyőtervek

3.2 A kézjel felismerő modul

Korábban egy kézjel felismerő modul már elkészült, ám a gesztus felismerés nem az elvárások szerint működött, ezért került most újraírásra. A modul vagy nagyon lassan, vagy rosszul ismerte fel a mutatott kézjeleket. Ez több okra is visszavezethető. Egyrészt, a natív kódot Windows-on fejlesztették, OpenCV segítségével, majd ez lett Android-ra portolva, ám a portolás során hibák maradtak a kódban, amik nem kerültek javításra. Másrészt, a gesztus felismerő eljárás erősen támaszkodott a színekre, ami az intenzív tesztelés során komoly problémának bizonyult. Nyugodt környezetben, ez nem jelent gondot, de vezetés közben gyakran változnak a fényviszonyok, ami nagyban torzítja az észlelt színeket. Ezen hibák miatt, döntöttünk úgy, hogy az alapoktól kezdve újraírjuk a gesztus felismerő modult. Az új modullal szemben a következő elvárásokat támasztottuk:

- egy olyan felismerő modul létrehozása, amely egyáltalán nem, vagy csak elhanyagolható mértékben támaszkodik a feldolgozandó kép színeire,
- fel kell térképezni minden lehetséges megoldást, és ezek közül kell kiválasztani a legjobban működőt.

Portolás esetén mindenképpen tüzetesen át kell nézni a kapott kódot, mielőtt integrálnánk az Android-os alkalmazással. Ez azért is fontos, mert az Android NDK segítségével futtatott natív kód esetén csak futás közben derül ki, ha hibás.

4 Ébren tartó alkalmazás fejlesztése

4.1 Fejlesztőkörnyezet beállítása

A fejlesztés Eclipse Classic 4.2.2 Juno termékcsomaggal történt, Windows 8 operációs rendszer alatt. Android 2.2, 4.0, 4.2.2 és 4.4.2 SDK-t telepítettem.

Az Android NDK Revision 9b (Október 2013) verzióját használtam, ez tetszőleges mappába tehető. Az eclipse-ben beállítottam a *Window/Peferences/Android/NDK* alatt az *NDK Location*-t az NDK-t tartalmazó mappára [3].

A projekt további környezeti beállításokat igényelt, az *Android.mk* fájlban megadott *OpenCV-2.4.9-android-sdk*-ra van szükség, melyet letöltöttem, importáltam a worspace-be és lefordítottam [5].

A VehicleICT platform használatához mindenképpen szükséges a készülékre letölteni és telepíteni a kliens 1.1-es verzióját (*vehicleict-client-v1-1.apk)*. Ezután a VehicleICT platform könyvtárat kell csatolni az alkalmazáshoz, jelenlegi legfrissebb verzió vehicleict-platform-lib.aar (1.1 verzió) kiterjesztésben érhető el.

4.2 Implementálás

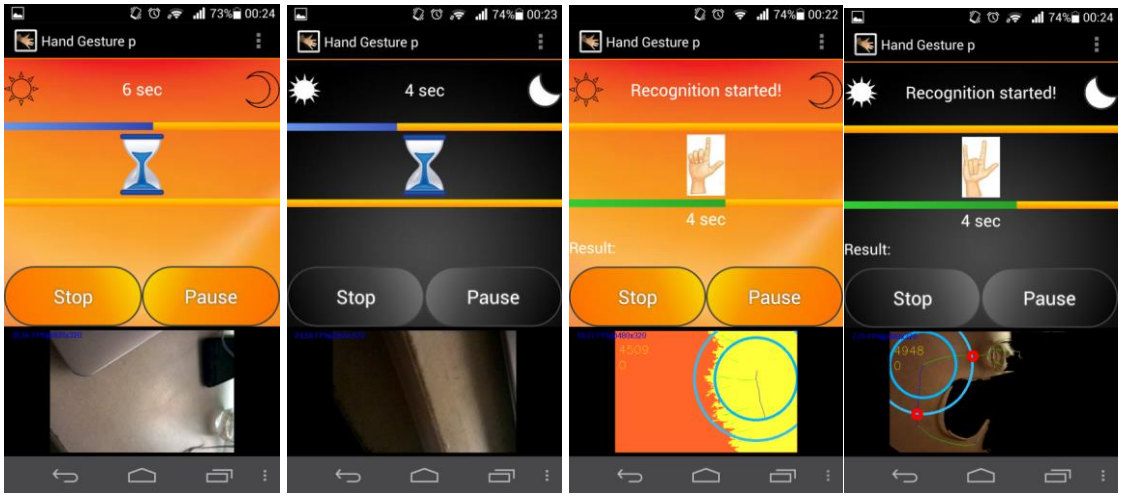
A megvalósítás során törekedtem az egyszerűségre, letisztult felületre és könnyű használhatóságra, akár változatos körülmények között, a megfelelő funkcionalitás biztosítása mellett.

4.2.1 Felhasználói felület és logika

A nappali és éjjeli módok közötti váltást a nap és a hold biztosítja a bal és jobb felső sarokban. Megterveztem a különböző stílusokat, nappal sötét a képernyő és a felismerés is, este a megvilágítás miatt világos a képernyő.

Implementáltam a megjelenítésért felelős logikát, a *Start* gomb megnyomása után elindul az akár pár perces visszaszámláló, egy random értékkel, mely egy *progress bar* segítségével jeleníti meg a hátralevő időt és egy homokóra képét (4.1. ábra). Ekkor még mindig sima kamera képet láthatunk. A visszaszámlálót a *CountDownTimer* osztály segítségével valósítottam meg, ennek a szálnak a befejezése triggereli a

felismerés kezdetét, majd a felismerés végén indítunk újat. A *progress bar* megjelenítéséért külön szál felelős, a kék másodpercenként számol vissza, míg a zöld másodpercenként többször frissül. Ha lejárt az idő megkezdődik az aktív felismerő fázis, megjelenik a kívánt kézmozdulat a képernyőn és a pár másodperc hosszú visszaszámlálás, ennyi időt kapunk a felismerésre. Világos stíluson az éjjeli felismerés látható, ekkor meg kell világítani a kezet a képernyő fényével, míg a sötéten a nappali.



4.1. ábra: A visszaszámlálás és felismerés fázis a két stílusban

Sikeres kézmozdulat esetén elölről kezdődik a pár perces visszaszámlálás a következő jelzésig. Sikertelenség esetén kiírjuk ennek tényét, mely fényében a vezető konstatálhatja, hogy sajnos kifutott az időből, lassul a reakcióideje. Természetesen a futás bármikor szüneteltethető, ekkor visszavált a sima kamera képre.

Ha elnavigálunk a visszaszámlálóról és visszaszámlálás van folyamatban, az onPause meghívásakor elindít egy service-t, amely folytatja a visszaszámlálást és a lejárta előtt egy másodperccel előtérbe helyezi az Activity-t. Ha felismerési időszak alatt fut az onPause, akkor a visszaszámláló kezdetekor indul a service.

Start és Stop gomb megnyomása között megakadályozza a készülék alvó módba lépését és a képernyő kikapcsolását.

A kép alapértelmezettként az első kamerával indul, amennyiben a készülék rendelkezik vele. Lekérdezi a készüléken található kamerákhoz tartozó információkat és megállapítja van-e köztük előlapi. Ha nincs, az alapértelmezett kamerával operál a továbbiakban. A kameraváltás a *Switch Camera* opcióval elérhető.

4.2.2 Felhasználó kezelés

A Parse, ingyenes felhő alapú szolgáltatás, mely a háttér adatbázisunkat is futtatja, támogatja a felhasználó kezelés megvalósítását [32]. Az alkalmazás, amennyiben van aktív felhasználó bejelentkezve, átirányítja a főoldalra, ahol lehetőség lesz kijelentkezésre is. Ha nincs aktív felhasználó az üdvözlő oldalra irányít, ahol lehetőség van a bejelentkezésre, vagy regisztrációra.

4.2.3 Hibakezelés

A több készüléken jelentkező összeomlásokat is jelezni kellene a fejlesztőnek. Erre az ACRA Android könyvtárat használtam, mely segítségével többféle módon is automatikusan elküldhető a Crash Riport.[30]

Az ACRÁ-t az Application osztályban kell inicializálni, összeomláskor felajánlja a hiba log e-mailben történő továbbítását a fejlesztőnek. A későbbiekben számos problémát sikerült felderíteni a segítségével.

4.2.4 Reakciók mentése

Ahhoz, hogy hálózati kapcsolat, vagy szerver elérés nélkül is tárolva legyen, illetve el lehessen őket érni, a gesztusok és reakciók SQLite adatbázisba lesznek perzisztensen mentve. Így azok nem vesznek el, ha a felhasználó elnavigál az alkalmazásunkból.

A gesztus és reakció információk periódusonként vannak tárolva. Egy periódus a *Start* és *Stop* gomb megnyomása közötti időszak, ennek során minden hosszabb visszaszámlálás után történik egy rövidebb felismerés időszak, ennek a reakció idejét kell mérni. Több ilyen visszaszámlálás-felismerés tartozik egy logikai egységbe és periódusba, akár több órán keresztül.

A periódusokat egy listában jelenítem meg, az eredmények a főoldalról érhetőek el a *Results* menüpont megnyomásával. A lista törlés esetén automatikusan frissül.

4.2.5 Eredmények megjelenítése

Az adatbázisba mentett periódusok megjeleníthetőek. A lista elemeire kattintva a részletek jelennek meg egy grafikon formájában, melyre *MPAndroidChart* könyvtárat használtam, ebben lehetőség van a megjelenítendő adatok megfelelő skálázására, így a kevesebb és a nagy mennyiségű adat is átlátható lesz [31]. A felismerési idők a későbbi

besorolás szerint oszlop és kördiagramon is megjeleníthetőek. Jelenleg is lehetőség van a kiugró értékeket, melyek esetében lejárt az idő, a grafikonról eltávolítani (4.2. ábra).

4.2. ábra: Eredmények megjelenítése

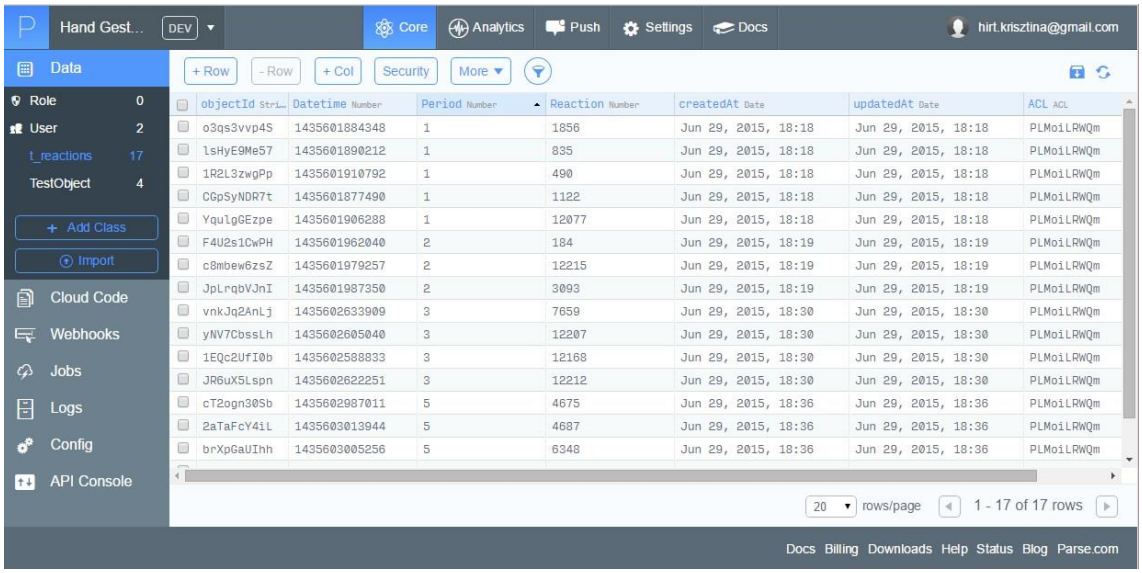
4.2.6 Jármű által szolgáltatott adatok

A jármű által biztosított szenzor információk a *VehicleICT* platform segítségével elérhetőek az OBD-II porthoz való csatlakozás után.

A *SampleListener* interfész implementálása és a platform felkonfigurálása után elindítható a mintavételezés. Showcase üzemmód elindításakor egy előre rögzített minta sorozat lejátszását indítja el, így valódi eszköz nélkül is tesztelhető. Állítható továbbá a szükséges mezők típusa, az üzemmód és a mintavételezés gyakorisága is, ami jelenleg 10 másodperc. A *receiveSample* felüldefiniálásával feldolgozhatóak a minták. Az OBD II *showcase* küldi a fogyasztási információkat, GPS adatokat, motor fordulatszámot, fojtószelep helyzetet, sebességet és egy hibakódot.

4.2.7 Mérések továbbítása

Egy ingyenes, saját magunk által kezelt felhőszolgáltatást választottunk a mérések tárolására, a Parse.com szolgáltatását [32]. A projekteket több platform is támogatja, többek között az Android is. Minden felismerési fázis végén ebbe az adatbázisba tölti fel az időbélyeget, az aktuális periódust és a reakcióidőt, amennyiben be van állítva a szervernek történő továbbítás (4.3. ábra).



4.3. ábra: Felismerési adatok a Parse-on

Az OBD II-es port által küldött adatok és GPS információk is továbbításra kerülnek. Az adatbázis tartalma exportálható JSON formátumban.

Megvalósítottam a még fel nem töltött adatok szinkronizálását. Ez manuálisan is kényszeríthető a főmenüből, illetve amennyiben rendelkezik internet kapcsolattal a fő activity megjelenítésekor automatikusan elvégzi.

5 Gesztus felismerés

5.1 Fejlesztői környezet beállítása

Mivel a kezdetekben nem tudtuk, hogy a végső megoldás milyen elemekből fog összeállni, a cél az volt a fejlesztés elején, hogy minél több megoldási lehetőséget próbáljunk ki, minél gyorsabban. Ezért a fejlesztői környezetet Windows 7 operációs rendszer alatt állítottam össze. Az OpenCV 3.0-t Microsoft Visual Studio 2013-al integráltam, mivel a hivatalos honlap ezt a programot javasolja.

Mivel Android rendszerre fejlesztjük az alkalmazást, ezért az OpenCV C++-os verzióját töltöttem le GitHub-ról. Ezt a későbbiekben sokkal egyszerűbb átalakítani Androidon is használhatóvá, hiszen az OpenCV Android-os változatához is C++-ban kell megírni a programot, mivel az Android NDK csak a C-ben és C++-ban megírt programot tudja futtatni. Az ilyen programokat már egyszerűen lehet integrálni bármilyen Android-os alkalmazásba. Miután a forráskód letöltődött, a CMake nevű program segítségével létrehoztam egy Visual Studio projektet. Ennek során először kiválasztottam a használni kívánt Visual Studio verziót (2013, 64-bit) valamint a használni kívánt C és C++ compiler-eket (a Visual Studio 2013 alapértelmezett compiler-ei). Ezek után hozzáadtam a projekthez az OpenCV modulokat, majd elkészítettem a projektet.

A projekt elkészülte után, kapunk egy OpenCV.sln nevű Visual Studio fájlt, melyet megnyitva, és az INSTALL nevű projektet kiválasztva build-elhetjük az OpenCV projektünket. Miután ezzel készen voltam, regisztráltam az OpenCV environment variable-öket, valamint létrehoztam egy PropertySheet-et Visual Studio segítségével, mellyel könnyen importálhatóvá tettem az OpenCV könyvtárakat, bármilyen Visual Studio projektbe.

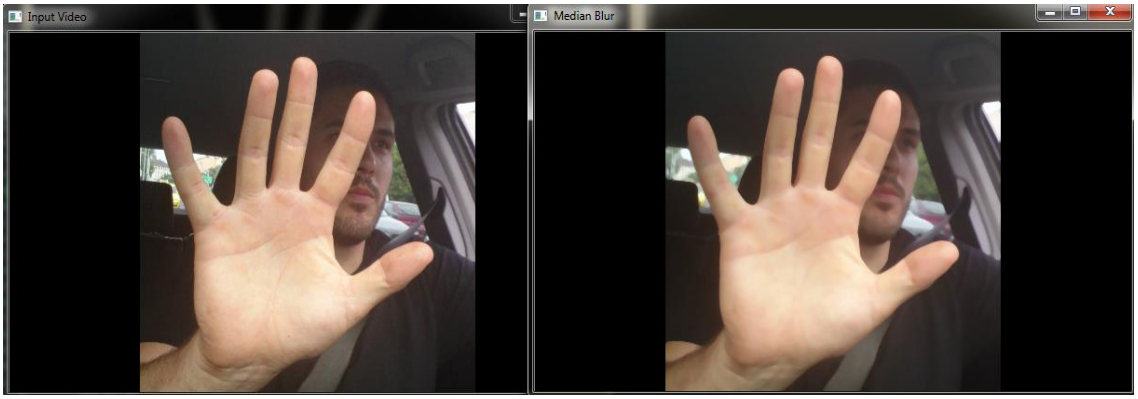
5.2 Implementálás

Az implementálás során két részre bontottam a modult. A program első fele felelős a kéz elkülönítéséért a kép többi részétől, a program második fele pedig a mutatott kézjel felismeréséért. A feldolgozás során a kamera által rögzített képeket egyesével dolgozom fel, egymástól függetlenül. A különböző lépések egymást lineárisan követik, abban a sorrendben, ahogy bemutatásra kerülnek.

5.2.1 A kéz elkülönítése

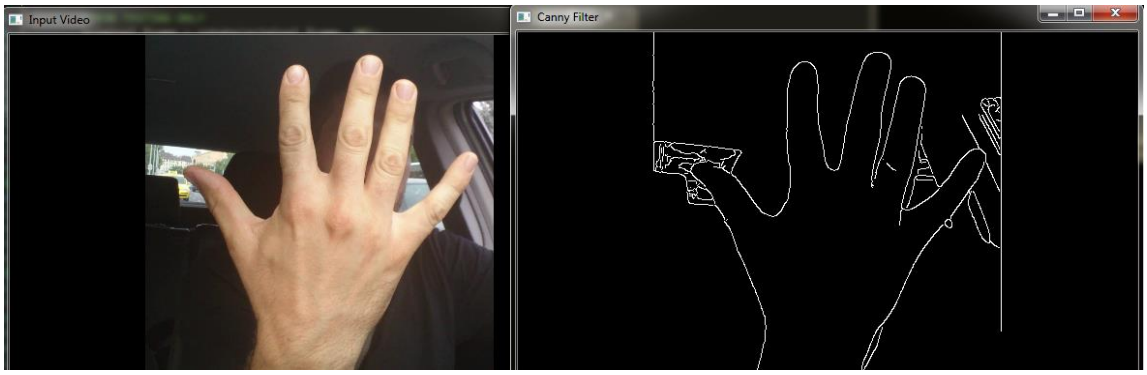
A kéz elkülönítése a legfontosabb és legösszetettebb lépése a programnak. Mint az irodalomkutatásban bemutattam, az évek során több módszert is kifejlesztettek, amely ezt a problémát hivatott megoldani, ezek sajnos a mi esetünkben nem használhatóak, több okból sem. Egyrészt el akarjuk kerülni, hogy egy okostelefonon kívül bármilyen más segédeszköz keljen az alkalmazás használatához, másrészt a mai okostelefonok specifikációi jelentősen behatárolják, hogy milyen technikákat használhatunk. Ezek alapján kizártuk a színes kesztyűs megoldásokat, valamint a mélység érzékelős és sztereó kamerás megoldásokat, mivel a legtöbb okostelefon nem rendelkezik az ezekhez szükség alkatrészekkel. Ezen kívül kizártuk a bőrszín alapú felismerést is, mivel az egyik célunk, a színek használatának elkerülése. Ezek után a kamerától kapott képet az alábbiakban leírt módon dolgozom fel.

A kép beolvasása után, első lépésként, alkalmazom a “medianBlur” függvényt, 5-ös mask size-ot használva. Ennek segítségével egységesebbé válik a kép, mivel eltűnik sok egy-két pixeles hiba a képről. (5.1. ábra)

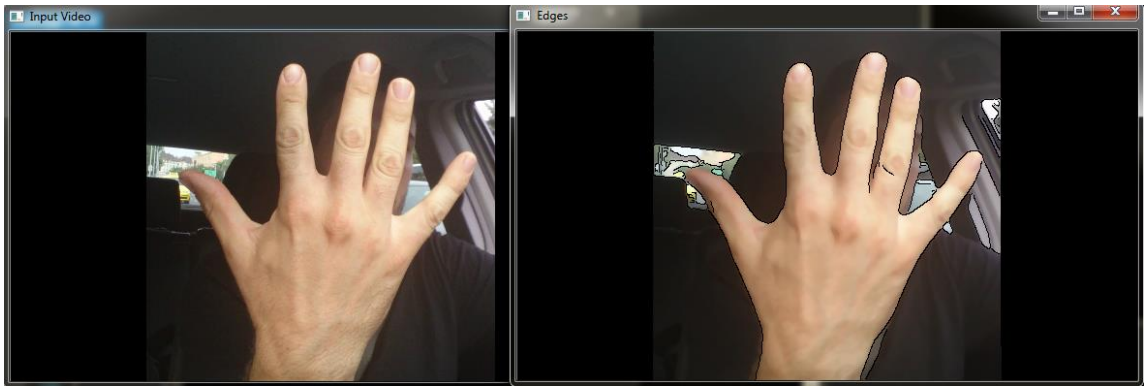


5.1 ábra: a “medianBlur” függvény használata után a kép kevésbé részletes, ezáltal egységesebb lesz

A következő lépés az élek megkeresése és kiemelése. Erre a Canny él kereső algoritmust használtam. Azért is fontos a blur függvény alkalmazása, hogy utána az él keresés során az algoritmus ne vegye számításba ezeket a pár pixelnyi foltokat. Ezáltal sokkal jobb és megbízhatóbb eredményt tudunk elérni. A megtalált éleket azután fekete színnel visszarajzolom a képre, így felerősítve őket. Fontos a fekete szín használata, mert ez biztosítja, hogy a későbbiekben is kiemelve maradjanak a megtalált élek. (5.2. és 5.3. ábrák)

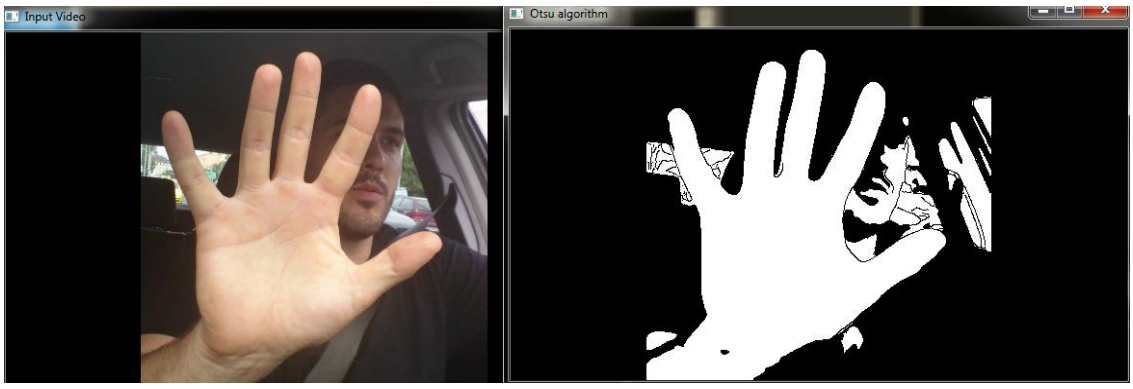


5.2. ábra: a Canny él kereső algoritmus eredménye, a képen megtalált élek fehérrel látszanak



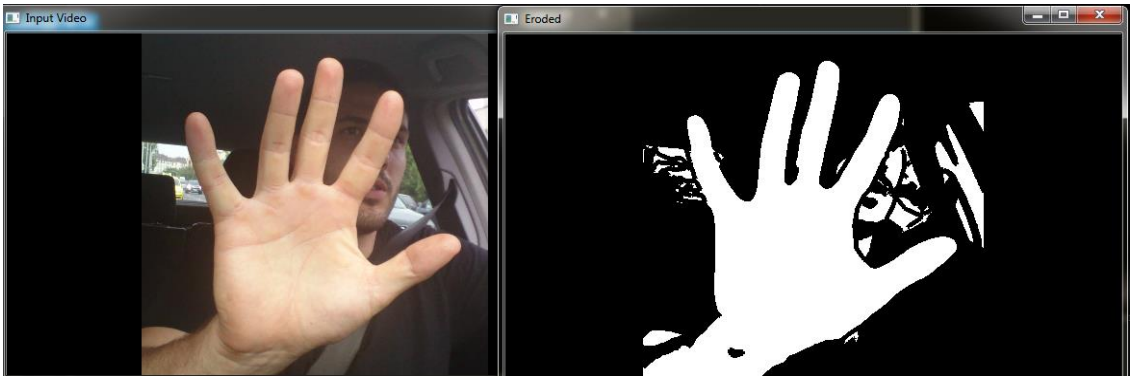
5.3. ábra: a Canny algoritmus által megtalált élek feketével vannak kiemelve az eredeti képen

Az így kapott egységesebb és kiemelt élekkel rendelkező képet átkonvertálom szürkeárnyalatossá, majd lefuttatom az Otsu algoritmust. Az Otsu algoritmus világosság szerint előtérre és háttérre osztja a képet. (5.4. ábra) Mivel a mi esetünkben a kéz a rögzített kép jelentős részét fogja elfoglalni, valamint világosabb, mint a háttér és a környezetének nagy része, ezért szinte mindig az előtérhez sorolódik és jól körülhatárolható lesz. A tesztek azt mutatták, hogy ez a gyakorlatban is így van, mind éjjel, mind nappal.



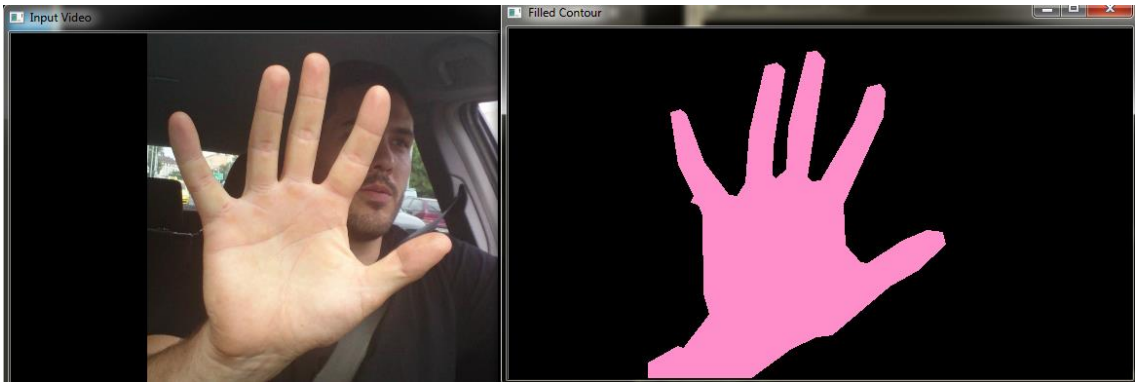
5.4. ábra: az Otsu algoritmus alkalmazása után egy bináris képet kapunk, fehérrel látszanak az előtérhez sorolt részek

A kapott bináris képen nagy valószínűséggel nem csak a kéz lesz megtalálható, hanem más egyéb világosabb képrészek is. Ezen alakzatok között a határokat az “erode” függvény segítségével emelem ki. Ebben a lépésben is 5-ös masksize-t alkalmazok. (5.5. ábra)



5.5. ábra: az “erode” függvény alkalmazása után a kép részei sokkal jobban elválnak egymástól

Miután megkaptam a bináris képet, jól elkülöníthető alakzatokkal, a “findContour” függvény használatával meghatározom a képen található alakzatok kontúrját. Ezek után pedig az “approxPolyDP” függvénnyel körbehatárolom az alakzatokat a kontúrjuk mentén, így kapva egy listát, melyben minden alakzat külön-külön elérhető. Mivel az alkalmazás használata során a kéz közel lesz a kamerához, így feltételezhető, hogy a kéz lesz a legnagyobb, egybefüggő alakzat, amit így kapunk. (5.6. ábra) Hogy a kezet megtaláljuk, minden kapott alakzat esetében megkeressük a legkisebb határoló téglalapot a “boundingRect” függvény segítségével, majd ezek közül kiválasztjuk a legnagyobb területtel rendelkezőt. Ha esetleg ez mégsem a kéz volna, akkor a program a későbbiekben ezt észlelni fogja.



5.6. ábra: A legnagyobb területű határoló téglalap kiválasztása után megkapjuk a kéz kontúrját

5.2.2 A kézjel felismerése

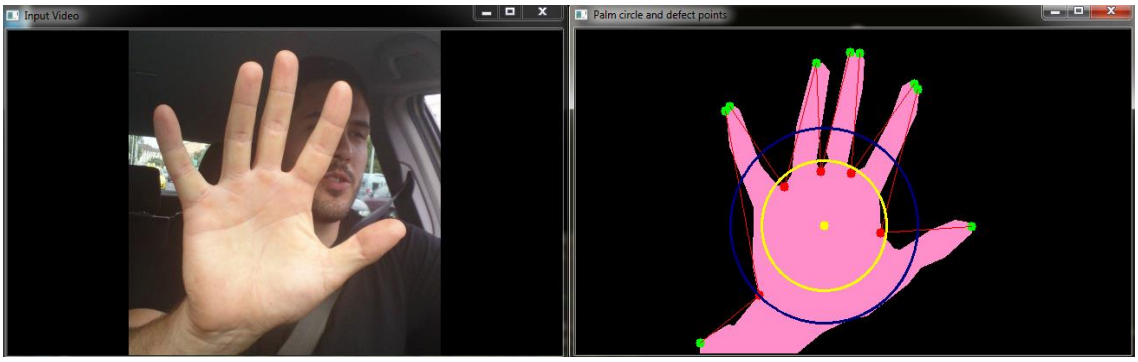
Miután elkülönítettük a kezet, meg kell határoznunk a felmutatott kézjelet. Hogy ezt megtehessük, először is meg kell keresnünk a képen az ujjakat és meg kell határoznunk a helyzetüket.

Első lépésként hasznos információkat nyerek ki a kéz kontúrjából. A kontúr alapján kiszámolható az alakzat tömegközéppontja. Ehhez a “moments” függvényt kell használni. Ezzel kiszámítom az alakzat első és nulla rendű momentumait, ezek hányadosa pedig megadja a tömegközéppontot (5.7. ábra).

A “convexHull” függvény segítségével meghatározom a kezet körbefogó legkisebb konvex alakzatot. Ez az alakzat többek közt az ujjak végénél fog csúcsokkal rendelkezni. Sajnos ez önmagában nem elég, hogy meghatározzuk az ujjakat, mivel több csúcs is lesz, amik nem az ujjaknál találhatóak (5.7. ábra).

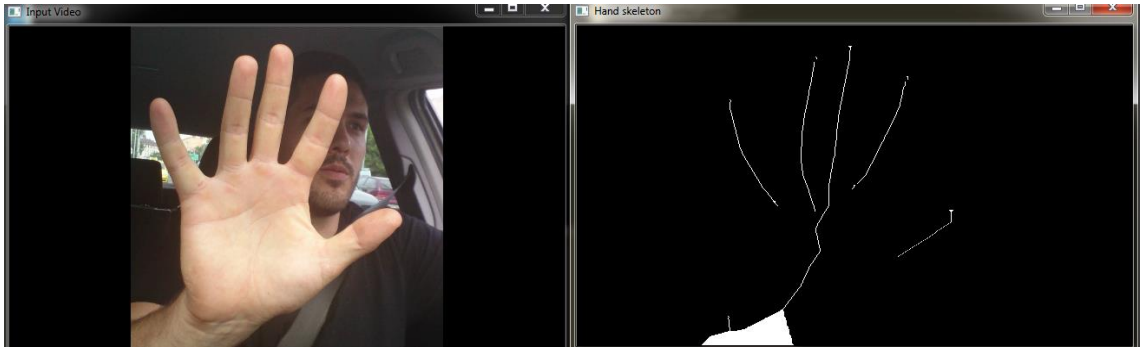
A konvex alakzat és a “convexityDefects” függvény segítségével meghatározom azokat a pontokat, ahol a kéz kontúrja a legjobban eltér az őt körülölelő konvex alakzattól, vagyis megkapom a kontúr konkáv pontjait. Ezek többek közt az ujjak között, valamint a tenyér és a csukló találkozásánál helyezkednek el általában (5.7. ábra).

Ha kiszámoljuk ezen konkáv pontok átlagos távolságát a tömegközépponttól, majd ezzel a távolsággal kört rajzolunk a tömegközéppont köré, akkor egy olyan kört kapunk, ami nagyjából meghatározza a tenyeret, mivel a legtöbb ilyen pont a tenyér szélén helyezkedik el. Ha egy ennél másfélszer nagyobb sugarú kört is felveszünk a képre, akkor az már biztos, hogy el metszi az ujjakat (5.7. ábra).



5.7. ábra: a jobb oldali képen zölden látszanak a kéz kontúrját körülölelő alakzat csúcsai, pirosan látszanak a kontúr konkáv pontjai, a kéz közepén látható sárga pont a tömegközéppont, a sárga kör, pedig a tenyeret jelképezi, kék színnel látható az ujjakat metsző kör

A következő lépés, a kéz skeleton-jának megkeresése. Erre a “distanceTransform” függvényt használom. A függvény kiszámolja a távolságot a kéz alakzatának minden pixelje és az adott pixelhez legközelebbi háttér pixel között, majd ennek alapján ad minden pixelnek egy nulla és egy közötti értéket. Minél távolabb van az adott pixel a háttértől, vagyis az alakzat szélétől, annál magasabb értéket kap. A “distanceTransform” lefutása után, a kapott mátrixon elvégzek egy szélsőérték keresést, amivel megkeresem a lokális maximumokat a mátrixban, így megkapva az alakzat skeleton-ját (5.8. ábra).



5.8. ábra: A “distanceTransform” függvény alapján elkészített skeleton a kézhez

5.2.3 A kézjel felismerő modul befejezése

Sajnos az idő rövidsége és a feladat összetettsége miatt nem sikerült teljesen befejezni a kézjel felismerő modult. Az alábbiakban vázlatosan szemléltetem, hogyan tervezem befejezni a programot. Mivel ezen módszerek legtöbbjét még nem volt alkalmam a gyakorlatban kipróbálni, ezért elképzelhető, hogy egyes helyeken változtatni fogok, valamint, hogy akadhatnak pontatlanságok a leírásokban.

A skeleton megtalálása után megkeresem azokat a köröket, melyek középpontja a tömegközéppont, majd ezek közül kiválasztom a nagyobb sugarúval való metszéspontokat. Ezektől a pontoktól kifele illetve befele helyezkednek el az ujjak skeletonjainak a részei. Először kifele lépkedek végig a skeleton vonalakon, bizonyos időközönként elmentve az éppen aktuális pontot. Ha ezzel végeztem, akkor ugyanezt elvégezem a körívtől befele is, amíg el nem jutok a tenyeret meghatározó, kisebb sugarú körig. Mivel a tenyér maga nem hordoz semmilyen lényeges információt a kézjel szempontjából, ezért azt nem kell vizsgálni. Az így elmentett pontok segítségével illesztek majd egyenest az ujjakra. Figyelni kell arra is, hogy a csuklót kiszűrjük az ujjak mellől, hiszen a skeleton képen az ujjakhoz hasonlóan található meg.

Ha megtaláltuk az ujjakat, akkor a felismerni kívánt kézjelek közül kiválasztjuk azt, amire a legjobban hasonlít. Ha esetleg minden kézjeltől jelentősen eltér a kapott eredmény, például azért, mert nincs kéz a képen, hanem a fej, vagy egyéb világos alakzat került kiválasztásra, akkor megismételjük az eljárást. Ha beazonosítottuk a mutatott kézjelet, akkor elmentjük az eredményt, majd még néhányszor lefuttatjuk a programot.

Miután összegyűjtöttük a szükséges mennyiségű kézjelet, kiválasztjuk az eredmények közül a leggyakoribbat, és ezt összehasonlítjuk az alkalmazás által kért kézjellel, majd visszaküldjük az eredményét az alkalmazásnak.

5.3 Kipróbált módszerek

Mint említettem, az is célunk volt, hogy minél több lehetséges megoldást kipróbáljunk. Egyrészt azért, mert ebben a témakörben még senki sem tudott egy konkrét, jól működő megoldást felmutatni, így mi sem tudtuk, hogyan érhetjük el a célunkat, másrészt azért, hogy megtaláljuk a legjobban teljesítő módszert, amivel megoldható a feladat. Az alábbiakban bemutatom azokat a módszereket, melyeket kipróbáltam a fejlesztés során, de valamilyen oknál fogva alkalmatlannak találtam a végső programban való használatra.

5.3.1 A kéz elkülönítése

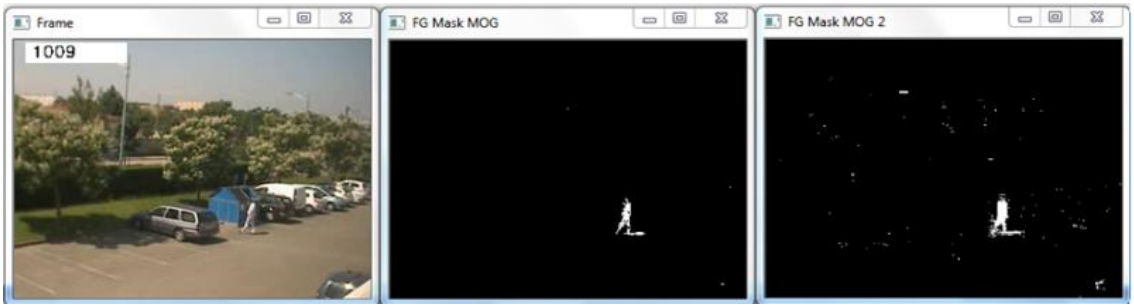
Több módszert is kipróbáltam a fejlesztés során, ezek két csoportra bonthatóak, melyek között a fő különbség, hogy az egyes képkockákat magukban, vagy pedig együtt vizsgálják-e. Mindkét csoportnak vannak erősségei és gyengeségei.

5.3.1.1 Képkockák együttes vizsgálata

A háttér kiszűrésére a képkockák együttes vizsgálata a legjobb. Az ilyen módszerek nagy hátránya, hogy nagy számítási igényűek, mivel sok képkockára van szükségük a működéshez, így a jelenlegi okostelefonok többségén használhatatlanok lennének. A technikai fejlődési sebességét látva azonban valószínű, hogy a közeljövőben nagyobb szerephez jutnak.

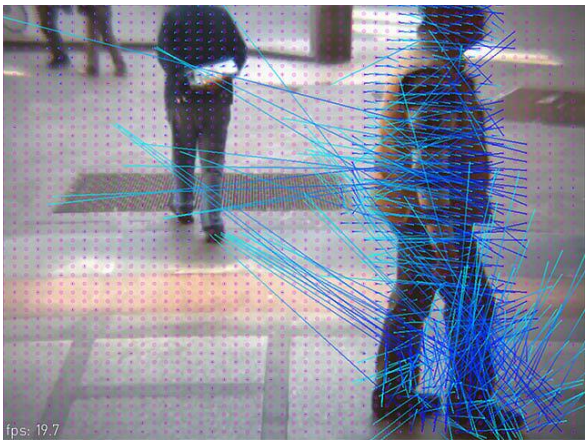
Az OpenCV környezet több függvénnyel is rendelkezik, amelyek az előtér elkülönítését végzik el egy videó alapján. Ezek a függvények a “backgroundSubtractorMOG”, a “backgroundSubtractorMOG2” és a “backgroundSubtractorGMG”. A függvények valószínűség számítás segítségével

különítik el a statikus és a mozgó pixeleket egymástól a videón. A három függvény közül a mi esetünkben a “backgroundSubtractorMOG2”-t lehetne alkalmazni. Ez a függvény jobban kezeli a fényváltozásokat, ellentétben a “backgroundSubtractorMOG”-al, valamint nincs szüksége pár másodperces várakozásra, mielőtt elkezdené a mozgó pixelek azonosítását, mint a ”backgroundSubtractorGMG”-nek. Sajnos azonban ezek az alkalmazás folyamatos mozgás követésére vannak kitalálva, ezért a mi esetünkben nem eléggé megbízhatóan működnek. A tesztek során a függvény által visszaadott képeken a kéz hiányos volt, ami annak köszönhető, hogy a kezdeti mozgás után viszonylagos nyugalomban van a kéz, így a függvény nem találja meg tökéletesen. (5.9. ábra)



5.9. ábra: a videón egyedül az ember mozog, ez és az egyéb kisebb hibák fehérrel láthatóak a függvények alkalmazása után, középen a “backgroundSubtractorMOG”, balra pedig a “backgroundSubtractorMOG2” látható

Egy másik hasonló módszer az OpticalFlow elemzése. Ez a módszer ugyancsak az egyes pixelek mozgását követi. Két képkocka alapján minden pixel esetén kiszámol egy elmozdulás vektort. Ennek segítségével meghatározhatóak a mozgó pixelek, valamint meg lehet határozni a mozgás nagyságát, így ki lehet szűrni a kisebb rezdüléseket a képről és csak a nagyobb, fontos mozgásokra koncentrálni. Sajnos az OpticalFlow használata nagy számítási igényű, hiszen a kép minden egyes pixele esetében kiszámolja az elmozdulási vektort, így jelenleg mobiltelefonokon ezt a módszert sem lehet alkalmazni. (5.10. ábra)



5.10. ábra: a képen kékkel láthatóak a mozgó pixelek elmozdulási vektorjai, rózsaszínnel pedig egy pont rendszer, ami segíti az elemzést

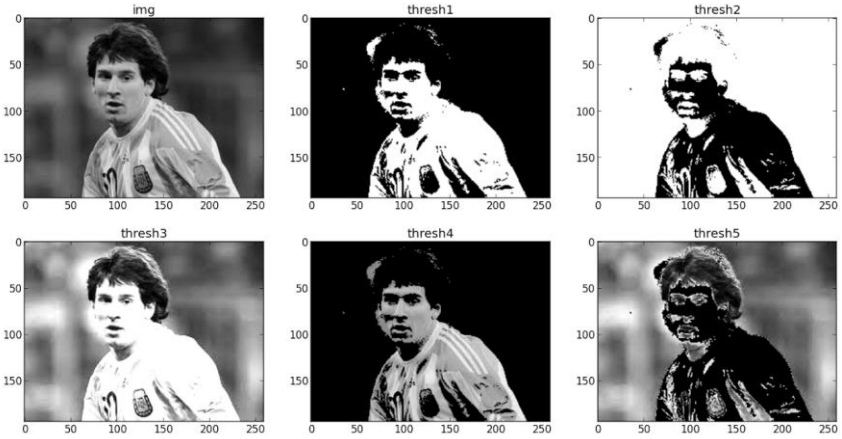
5.3.1.2 A képkockák egymástól függetlenül történő vizsgálata

A háttér kiszűrésének másik módja, ha a kamera által rögzített képkockákat egyesével, egymástól függetlenül dolgozzuk fel. Ennek során különböző módszerek segítségével lehet megpróbálni elkülöníteni a kezet a kép többi részétől. A cél az, hogy a képet több részre osszuk fel, majd ezek közül kiválasszuk azt, ami tartalmazza a kezet. Ehhez a legjobb megoldás a kép feldarabolása a rajta látható kontúrok alapján. Ezt a már fentebb említett “findContours” nevű függvényével tudjuk megtenni. A függvény használatához egy bináris (fekete-fehér) kép szükséges. Ezt több féle módon is elő tudjuk állítani.

A legegyszerűbb megoldás, a bőrszín alapú szűrés. Ennek során, azok a pixelek a képen, melyek előre meghatározott határértékek közé esnek, fehérek lesznek, a többi pixel pedig fekete. Mivel nem akarunk szín alapú szűrést az algoritmusban, ezért ezt az eljárást el kellett vetni.(5.11. ábra)

5.11. ábra: a bal oldali képen látható bőrszínű területek fehéren jelennek meg a jobb oldali képen, látható hogy már egy nagyon kis megvilágításbeli eltérés is rengeteg hibát okoz

Egy másik lehetőség bináris kép készítésére valamilyen thresholding függvény alkalmazása. Ezek a függvények egy szürkeárnyalatos képet transzformálnak binárissá, úgy, hogy azok a pixelek, melyek értéke a megadott határérték felett van, fehérek lesznek, a többi fekete. Sajnos a legtöbb thresholding függvény nem tud megbirkózni azzal, ha egy képen belül változik a fényerősség a kép részei között. Így ezek esetünkben nem alkalmazhatóak. (5.12. ábra)



5.12. ábra: a képen balról jobbra, fentről lefele a következő thresholding eljárások láthatóak (127-es pixel értéket használva határértékként): eredeti (szürkeárnyalatos) kép, bináris, inverz bináris, truncate (ha az adott pixel értéke nagyobb mint a határérték, akkor fehérré változik, ha nem akkor megtartja eredeti értékét), threshold nullához (ha pixel értéke nagyobb mint a határérték, akkor érintetlen marad, egyébként nullává változik), inverz threshold nullához

6 Tesztelés és adatelemzés

Az alkalmazás és a kézjel felismerés fejlesztésével a célunk a fáradtság detektálása vezetés közben, a reakciók és jármű adatok monitorozásával. Ehhez szükséges megfelelő mennyiségű teszt alany és minta gyűjtése, majd különböző technikák segítségével az összegyűjtött adatok elemzése, hogy minél pontosabban meghatározhatóak legyenek a vezetés körülményei, az elalvás és a figyelemzavar valószínűsége.

Elsődleges feladat meghatározni mennyire éber a vezető egy hosszú monoton út során. Ezeket az értékeket el kell különíteni a zavaró jelektől, például mikor nagyobb forgalmú területre ér és nem képes a mérés elvégzésére a kormányzás miatt.

6.1 Adatgyűjtés

A reakció mérő alkalmazás a jelenlegi szinten alkalmas a mérési adatok tömeges gyűjtésére. A tesztidőszak alatt a kamera egy teszt gombbal lett helyettesítve, a felismerést egy gombnyomás szimulálja. A Start és Stop gomb lenyomásakor az alanyoknak lehetőségük nyílik egy szöveges üzenet elküldésére, melyben leírják a mérés körülményeit, ezt szintén továbbítom a szerverre. Jelenleg 5 alany mérési eredményei állnak rendelkezésre, melyeket 3 héten keresztül a legkülönfélébb időpontokban és körülmények között végeztek, melynek során több mint 1000 reakcióidő rögzítés történt.

A feladat az alkalmazás telepítése és regisztráció után, a mérés kezdetekor, a körülmények tömör megfogalmazása, elsősorban

- mennyire érzi fáradtnak magát vagy mennyit aludt az éjjel, mikor szokott lefeküdni,
- milyen helyzetben méri, egyéb külső körülmények.

Ezután elindul a 3-6 perces visszaszámlálás, melynek végén következik reakció mérési időszak, melyet figyelmeztető hanggal jelez, az alanynak meg kell nyomnia a teszt gombot, melyre 15 másodperce van jelenleg, ezzel folytatódik a visszaszámlálás. Ha nem nyúl hozzá az idő lejárta után, szintén a visszaszámlálással folytatódik. Ha elnavigál a képernyőről az idő lejárta előtt automatikusan felugrik az ablak, így nem okoz gondot a telefon használata mérések alatt. A méréseket nem kötelező gépjárműben

készíteni, de előny. Munka vagy szórakozás közben is lehetséges a feltétel a hosszabb monoton egy helyben tartózkodás és a telefon kartávolságban tartása. A tesztek szerint a 3-6 perc bőven elegendő, hogy az alany megfeledkezzen a feladatról, így legtöbb esetben releváns méréseket produkál. A végén szintén fontos a mérés közben történtek leírása a Stop gomb megnyomásakor,

- milyen zavaró körülményeket tapasztalt
- fáradtabbnak érzi-e magát.

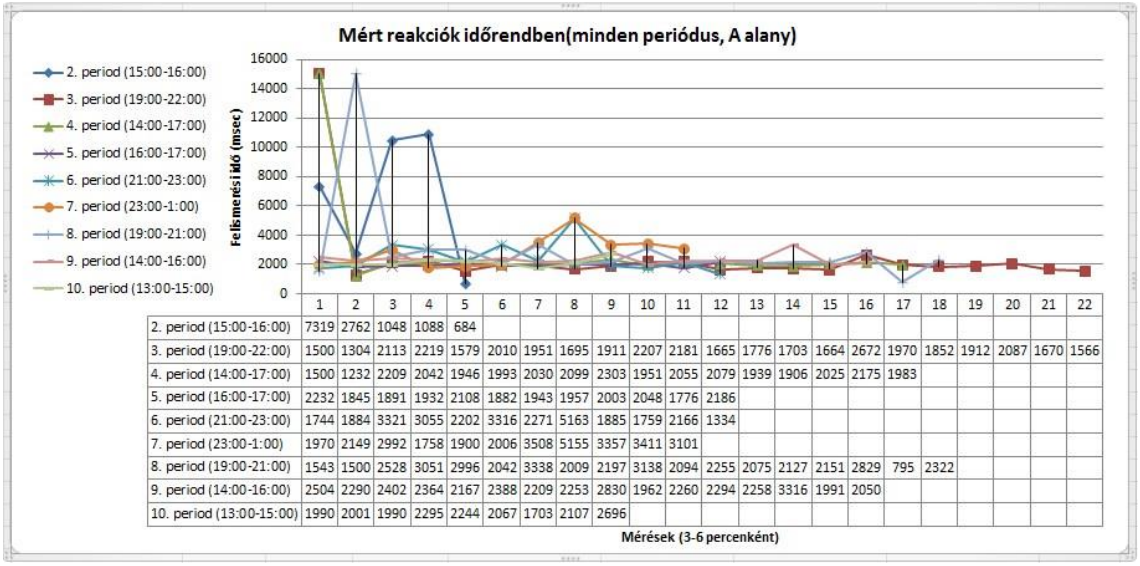
A mérések reggel 8 és hajnali 4 között történtek, különböző fáradtsági szinteken, vezetés közben, nappal illetve este munka közben, asztalnál ülve, illetve este szórakozás, film, számítógép előtt ülve, nyugodt és zavaró körülmények között is, 20 és 50 év közötti férfiak és nők közreműködésével.

A mérések a Parse.com-on levő adatbázisba lettek továbbítva, melyen JSON formátumban elérhetőek és exportálhatóak, az Excelbe történő importálás után Excel és Matlab segítségével már feldolgozhatóak.

6.2 Elemzés

A beérkezett több mint 1000 mérési eredményt megvizsgáltam alanyra, periódusra, időszakra vonatkozólag. Az eredmények áttekintésére először meghatároztam az átlagot, szórást, mediánt, minimumot, maximumot, 10%-onként a percentilis értékeket majd periódusonként összehasonlítottam.

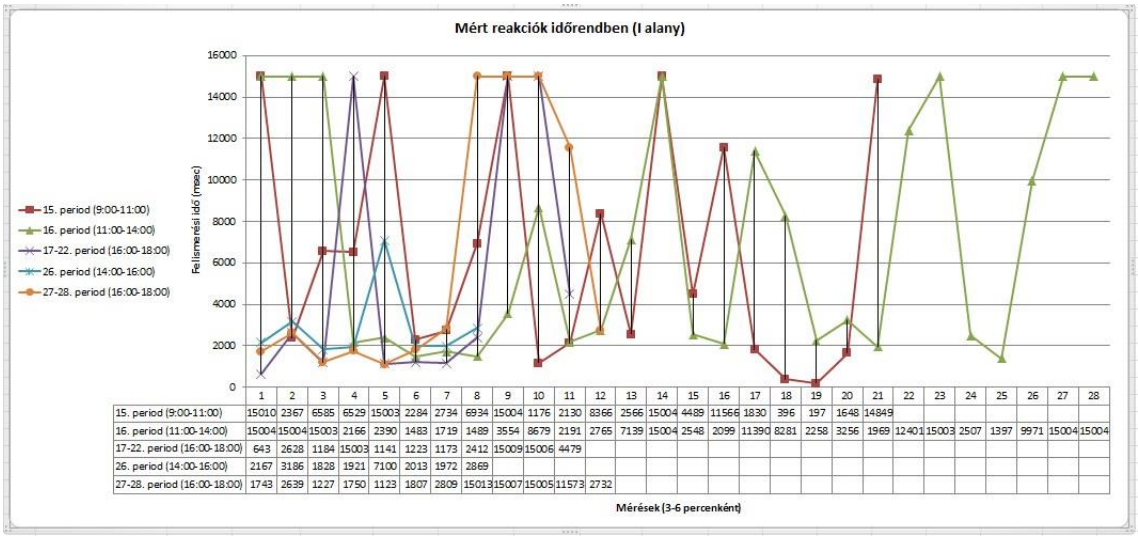
Egy perióduson belüli reakcióidők igen változatosak 150 és 15000 ezredmásodperc közötti értékek, a függvények alakja változatos, gyakori kiugró értékekkel. A következő 6.1. ábrán az első alany eredményei láthatóak néhány jellemző függvénnyel, a 2-10. periódusig, alatta a mért értékek ezredmásodpercben megadva. Megfigyelhetőek a 15 másodperces kiugrások, ebben az esetben nem történt mérés, a többi alanynál ez gyakoribb. Látható, hogy 2 másodperc körül tömörülnek az alacsony eredmények. A megfelelő lelkiállapot, éberség és zavarásmentes időszakokban mindig a kisimult függvény a jellemző. Zavarás, mérés kihagyás vagy álmosság esetén az értékek szórása nagyban megnőtt.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2. period (15:00-16:00)	7319	2762	1048	1088	684																	
3. period (19:00-22:00)	1500	1304	2113	2219	1579	2010	1951	1695	1911	2207	2181	1665	1776	1703	1664	2672	1970	1852	1912	2087	1670	1566
4. period (14:00-17:00)	1500	1232	2209	2042	1946	1993	2030	2099	2303	1951	2055	2079	1939	1906	2025	2175	1983					
5. period (16:00-17:00)	2232	1845	1891	1932	2108	1882	1943	1957	2003	2048	1776	2186										
6. period (21:00-23:00)	1744	1884	3321	3055	2202	3316	2271	5163	1885	1759	2166	1334										
7. period (23:00-1:00)	1970	2149	2992	1758	1900	2006	3508	5155	3357	3411	3101											
8. period (19:00-21:00)	1543	1500	2528	3051	2996	2042	3338	2009	2197	3138	2094	2255	2075	2127	2151	2829	795	2322				
9. period (14:00-16:00)	2504	2290	2402	2364	2167	2388	2209	2253	2830	1962	2260	2294	2258	3316	1991	2050						
10. period (13:00-15:00)	1990	2001	1990	2295	2244	2067	1703	2107	2696													

6.1. ábra: Mért reakcióidők (A alany)

Mérések során a másik gyakori eset, a gyakori zavarás, vagy tényleges figyelemhiány miatt bekövetkezett kiugró értékek, nagy szórás, mely a következő ábrán figyelhető meg. Ez esetben ki kell szűrni a mérésre alkalmas részeket, a többi időszakban pedig megállapítani mennyire vehetőek figyelembe a mérési eredmények. A további függvények képei is megegyező jelleget mutattak ezekkel a típusokkal.



	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
15. period (9:00-11:00)	15010	2367	6585	6529	15003	2284	2734	6934	15004	1176	2130	8366	2566	15004	4489	11566	1830	396	197	1648	14849							
16. period (11:00-14:00)	15004	15004	15003	2166	2390	1483	1719	1489	3554	8679	2191	2765	7139	15004	2548	2099	11390	8281	2258	3256	1969	12401	15003	2507	1397	9971	15004	15004
17-22. period (16:00-18:00)	643	2628	1184	15003	1141	1223	1173	2412	15009	15006	4479																	
26. period (14:00-16:00)	2167	3186	1828	1921	7100	2013	1972	2869																				
27-28. period (16:00-18:00)	1743	2639	1227	1750	1123	1807	2809	15013	15007	15005	11573	2732																

6.2. ábra: Mért reakciók (I alany)

A percentilis értékek és az átlagok vizsgálatával is nyilvánvalóvá vált, hogy a magas mérési eredmények komoly zajt jelentenek a normális reakcióidők megállapításában. Minden felhasználónál szűrtem az eredmények alsó 10%-át, ami véletlen esemény hatására lehetett kiemelkedően alacsony. Majd a nagy torzítás miatt az

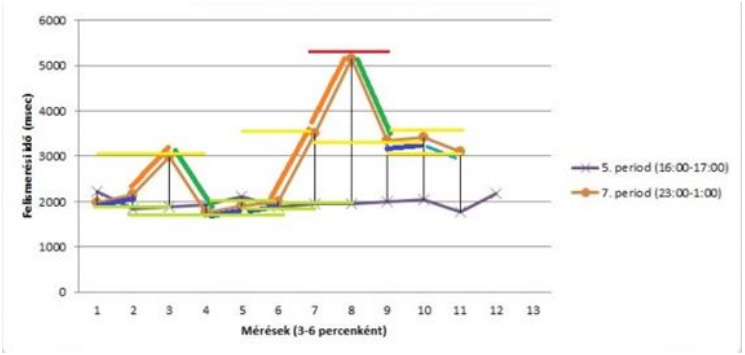
átlagon felüli értékeket is, így minden esetben közel azonosan 1300 és 5500 ezredmásodperc közötti értékeket kaptam.

Megvizsgáltam a teljesen nyugodt állapotban mért értékek átlagát a különböző felhasználók és periódusok esetében, 15 sorozat után az átlagok 1995-2083 ezredmásodperc között mozogtak, ezzel megállapítható a normális reakcióidő szintje, ami ezzel az alkalmazással elérhető.

6.2.1 Reakciók sorrendjének vizsgálata

Az eddigi megfigyelések alapján felmerül a kérdés, hogy lehetséges-e a reakció időket különböző szintek szerint osztályozni. A nagyon alacsony eltérések tekinthetőek-e egyenesnek. A mérések sorozatát elemezve megfigyelhető, hogy milyen alapelemekből épül fel.

A függvények felírhatóak különböző típusú vonalak sorozataként. A különböző alapelemek és azok elhelyezkedése jellemző lehet bizonyos helyzetre. Egy teljesen egyenes sorozat 2 másodperces átlagidőkkel, még egy esetleges kiugrás esetén is éber állapotot jelez. Egy-egy csoportot 3 mérés alkot, melyek állhatnak egyenesekből, növekvő, illetve csökkenő vonalakból. Az 6.3. ábrán egy ilyen elképzelés látható, egy bizonyítottan kiegyensúlyozott és egy elalvást jelző függvényen. Bizonyos szinten található minimum és maximum értékekkel (zöld, sárga, piros), valamint a különböző típusú vonalakkal, egyenes (kék), emelkedő (sárga), csökkenő (zöld). Ezek egy bizonyos sorrendje meghatározhatja az éberség szintjét, zavarás valószínűségét.



6.3. ábra: Reakció változás és szintek besorolása

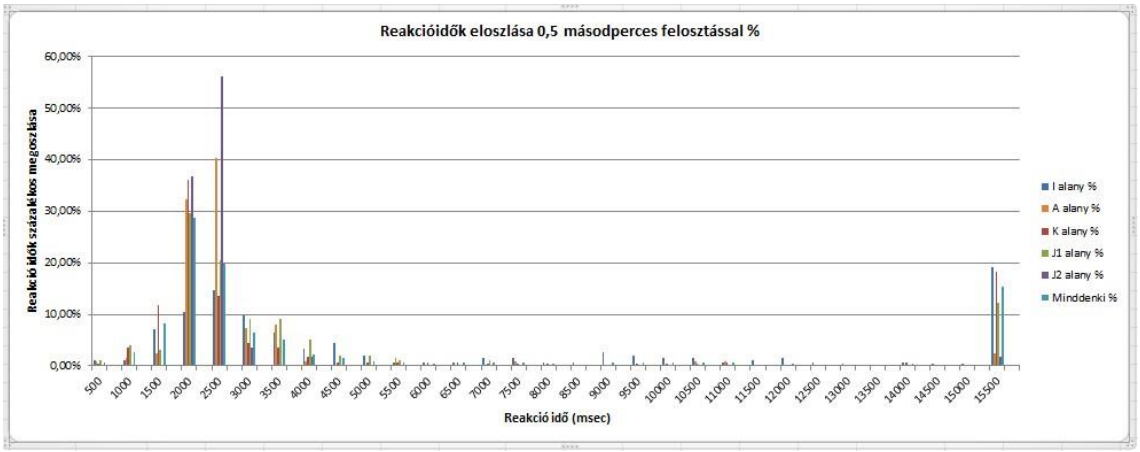
Fontos kérdés a szintek helyes megállapítása, alkalmazható-e a különböző felhasználók mindegyikére? Valamint a pontosság meghatározása, ki lehet-e terjeszteni hosszabb sorozatokra, vagy lefedni egy mérőszámmal ennek érdekében? Vannak-e bizonyos feltételek, amelyek interpretálnak egy meghatározott eredményt?

6.2.2 Reakciók eloszlása

Elvégeztem a mennyiségi ismérvek alapján végzett adattömörítést, gyakoriségi sorok képzésével, többféle szempont szerint.

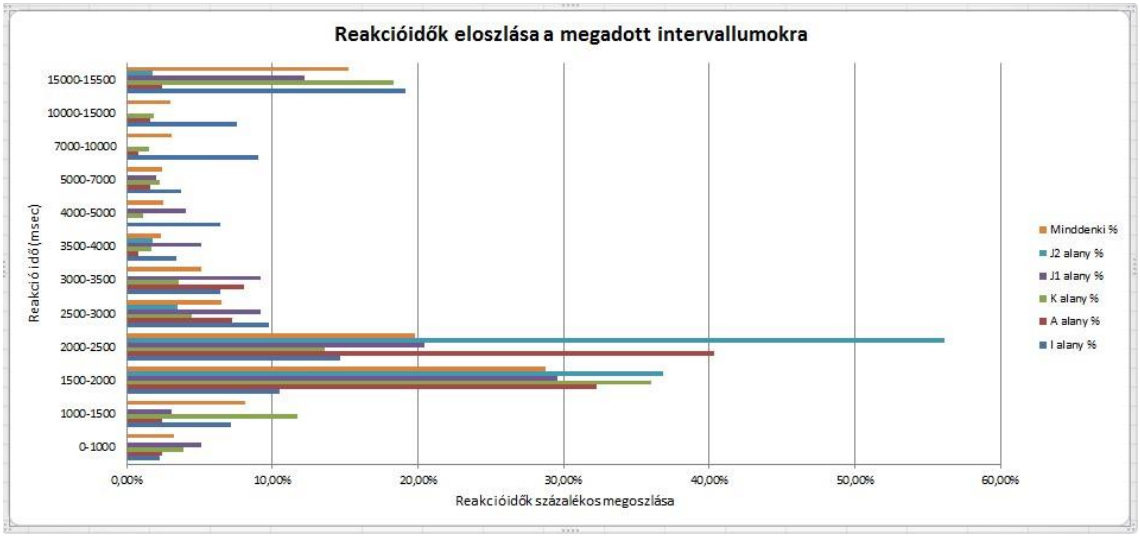
6.2.2.1 Reakció idő szerinti eloszlás

Először a 0-15000 ezredmásodperces időtartamot fél másodperces egységekre bontottam. A következő 6.4. ábrán látható, hogy az összes mintát alapul véve 1000-1500 között található a mérések 8%-a, majd egy nagy ugrással 1500-2000 között a mérések 29%-a, 2000-2500 között 20% látható. Ezután 2500-3500 között 5-6% körül mozog, végül folyamatos csillapodást követően 5500-től visszaesik 0.5%-ra. 15000 ezredmásodperc fölött láthatóak a kiugró értékek mikor a mérés kifutott az időből, az előfordulás ismét 15% körüli.



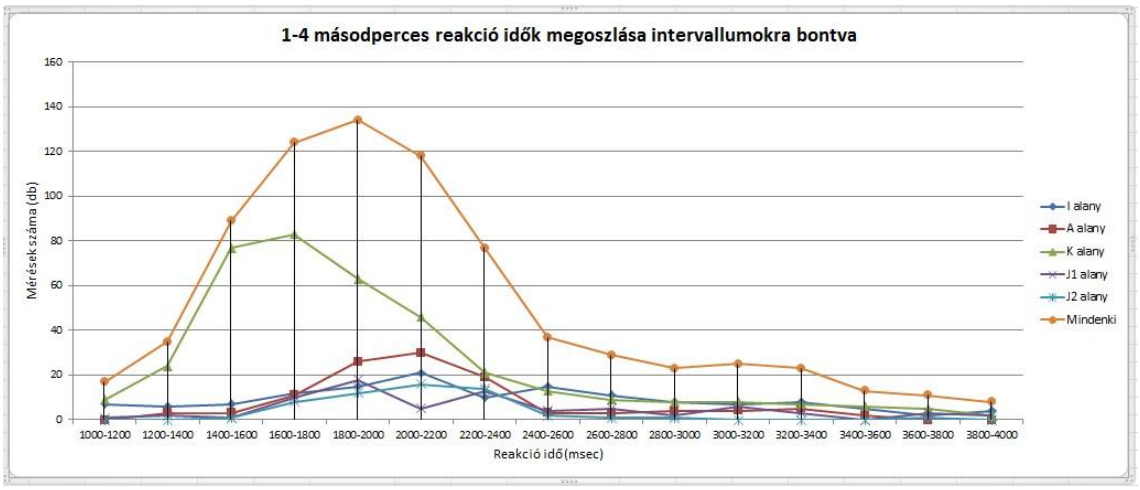
6.4. ábra: Reakciók százalékos eloszlása 0,5 másodperces felosztással

Kevésbé részletes felosztás szerint látható, hogy a leggyakoribb értékek egyénenként változnak (6.5. ábra). A magasabb reakcióidők eloszlása megegyezik az adatok előzetes elemzése során kapott értékekkel és a mérés közben leírt tapasztalatokkal. Akit többet zavartak, esetleg gyakrabban volt figyelmetlen, vagy álmos, a magasabb értékek is sűrűbben fordulnak elő. J2 és A alany esetén kiugróan nagy számban vannak alacsony eredmények. Ez megerősíti, hogy sokszor végeztek nyugodt méréseket a többiekhez képest. I alany esetében jóval magasabb arányban fordul elő magasabb mérési eredmény, az ő sorozatai jóval zajosabbak.



6.5. ábra: Reakció idők eloszlása

Értékelhető eredmény az alacsonyabb értékek eloszlásánál figyelhető meg részletesebb felbontásban. A 6.6. ábrán 0,2 másodperces felbontásban egymóduszú szimmetrikus és aszimmetrikus gyakorisági sorok poligonjai láthatóak. K alany mérési száma a legnagyobb és jobb oldali aszimmetriát mutat, vagyis az alsó határ szigorúbb. Láthatóak az egyes felhasználók közötti különbségek minimum teljesítményt illetően. Magas mérési szám esetén a függvények kisimultak. K alany esetében a leggyakoribb értékek 1400 és 1600 között fordulnak elő, míg a többieknél 1800-2400 közöttiek. Az alsó határ 1200.



6.6. ábra: 1-4 másodperces reakciók eloszlása

Mivel a készülékek teljesítménye közel azonos és kor szerint sincs nagy eltérés, a mérések a teljes napot felölelik a legtöbb mérő esetében, így fennáll a lehetőség, hogy tanulással vagy megszokással javítható az eredmény. Alacsonyabb mérési szám esetén I

és J1 alanynál is törés figyelhető meg, ez a különböző mérési helyzetekből adódó differencia. Megállapítható, hogy alapbeállításként mindenkinél azonos feltételekkel kezdhető a mérés, de szükségessé válhat az eredmények alapján későbbi módosítás. A függvények hasonló képet mutatnak korra, nemre és a készülék minőségére való tekintet nélkül.

6.2.2.2 Napszak szerinti eloszlás

Normális helyzetben, amennyiben nem történik semmi rendkívüli a nap folyamán, éberségi ciklusunk periodikus, reggeltől délutánig nő, majd egy helyi visszaesés után 22 órától hajnali 3-4-ig meredeken zuhan. A legtöbb mérést végző K alany 379 mintáját alapul véve, lebontottam az eloszlást órákra bontva, majd a hasonló eredményeket összevontam a következő grafikonon. A 6.7. ábrán látható, hogy a leggyakoribb reakcióidők 15-22 óráig 1400-1600 ezredmásodperc, majd 23-tól hajnali 2-ig 1600-2000-re majd 1800-2200-ra, végül 3-tól 2000-2400 közötti eredményekre nőttek. Bár az alany lefekvési ideje általában hajnali egyre tehető, a fáradtság bizonyított, már akkor észrevehető a növekedés. A mérések szerint a reakció idő átlagosan 0,7 másodpercet növekedhet már közvetlenül a szokásos lefekvési idő után.

K alany	10-13-ig	15-17-ig	19-20-ig	21-22-ig	23	0	1	2	3-4-ig	Összesen
1000-1200	2	0	1	0	2	2	0	1	1	9
1200-1400	1	6	3	4	2	4	0	2	3	25
1400-1600	3	21	13	24	4	4	6	6	1	82
1600-1800	7	14	5	10	18	12	8	5	1	80
1800-2000	4	8	4	8	11	12	5	10	1	63
2000-2200	1	7	0	6	4	2	6	11	6	43
2200-2400	0	3	0	4	2	3	1	3	6	22
2400-2600	1	1	1	0	2	4	0	3	0	12
2600-2800	0	2	1	0	1	1	2	0	0	7
2800-3000	0	2	0	2	0	1	0	2	0	7
3000-3200	1	2	1	0	0	1	1	2	0	8
3200-3400	0	0	0	4	0	2	0	1	2	9
3400-3600	0	1	0	0	0	2	1	2	0	6
3600-3800	2	1	0	0	1	1	0	0	0	5
3800-4000	0	0	0	0	0	1	0	0	0	1
Összesen	22	68	29	62	47	52	30	48	21	379

6.7. ábra: Reakciók eloszlása a különböző napszakokban

6.3 Szabályrendszer kialakítása

A továbbiakban a legfontosabb, hogy már a mérések elvégzése közben megállapítható legyen a fáradtság illetve a zavarás szintje.

A 4 hasonló eloszlást mutató alany értékeiből meghatároztam 4 alapvető szintet:

- 0-2900 msec: zöld, jó eredmény, eddig tart a függvények csökkenése,
- 2900-4000 msec: sárga, közepes eredmény, romló teljesítmény

- 4000-14000 msec: piros, magas, e fölötti értékek vagy zavarásra utalnak, vagy nagyon erős reakciócsökkenésre,
- 14000 msec fölött nincs jelen a mérő, fekete.

Néhány pontosan ismert eredményű függvényt alapul véve kialakítottam egy szabályrendszert az utolsó 3 érték sorozatára. Zöld, sárga, piros és fekete típusú értékek mindegyikére meghatároztam mennyiben csökkentik vagy növelik az éberségi és jelenléti szintek valószínűségét. Figyelembe kell venni, amennyiben a szabályok túl szigorúak, fals pozitív eredményre juthatok, amennyiben túl megengedőek nem lehet észrevenni az éberség csökkenést. A jelenléti szintet csökkenti az összes zavaró tényező, ami nem számít reakciócsökkenésnek, valamint a mérések hiánya. Magas éberségi és magas jelenléti szint mellett minden rendben van. Magas éberségi és alacsony jelenléti szint mellett nincs lehetőség a készülékhez nyúlni valamilyen okból. Alacsony éberség, magas jelenlét mellett a legnagyobb a kockázata az éberség csökkenésének. Alacsony éberség, alacsony jelenlét esete kialakulhat, ha mindig hasonlóan magas szintű a zavarás, vagy extrém a reakciócsökkenés. Nagy bizonytalanság esetén lehetőség van a mérés fél percen belüli megismétlésére. Az eddigi tesztek azt igazolták, amennyiben tényleg zavarásról van szó, 4 másodpercen belüli reakciót szinte lehetetlen produkálni.

A szabályok lehetőséget adnak a jelenlét hiányából és a zavarásból adódó hibák kiküszöbölésére is, néhány jellemző szabály:

- utolsó két zöld érték esetén az éberségi és jelenléti szint is maximális,
- utolsó fekete esetén a jelenlét drasztikusan csökken, de az éberséget már az előző értékektől függően csökkenti, de általában minimálisan, vagy 0 mivel ekkor az éberség nem változik vagy nem meghatározható.
- jelentősége van a magas-alacsony-magas, valamint az alacsony-magas-alacsony értékhármasoknak is, hogy lokális hiba, vagyis zavarásként értelmezhető-e, ha zöld értékek vesznek közre egy magasabbat, akkor szinte mindig hibáról beszélhetünk,
- különbséget kell tenni, ha több piros vagy sárga érték szerepel, mert míg a sárgák esetén a jelenlét esélye szinte maximális, piros esetén már a zavarás esélye is megnő.

Ezután lefuttattam a szabályokat szintén ismert minta csoportra és összehasonlítottam a várt eredménnyel. Amennyiben nem volt megfelelő,

következetesen módosítottam a szabályokon. Mivel nem vettem figyelembe, hogy a reakciók időnként extrém magasra nőnek, így a piros intervallum további finomítást igényelt. A tapasztalat és beszámolók szerint nagy fáradtságnál több másodpercig is eltart, míg az agyunk realizálja a helyzetet. Ekkor olyan viselkedés tapasztalható mintha nem is lennénk tisztában vele hol vagyunk és mi történik. Még ha nem is ismeretlen számunkra a feladat, akkor is jó ideig eltarthat, míg kapcsolunk és megnyomjuk a gombot. Extrém esetekben ez 4-9 másodpercig is eltarthat. Felbontottam a 10 másodpercet 1, 2, 3, 4 másodperc hosszú intervallumokra. Az utolsó 3 érték átlagától függően a következőképpen módosul:

- az éberségi szint jó átlag esetén nő, közepes esetén fokozatosan csökken, majd nem változik, mivel ekkor már zavarás gyanúja állhat fenn,
- a jelenléti szint változás a jó értéktől fokozatosan csökken.

Kiegészítettem az utolsó érték figyelembe vételével, mely szerint:

- az éberség zöld és fekete érték esetén is nő, mivel ekkor vagy minden rendben, vagy az esélye csökken a jelenlét hiányával
- a jelenléti szint az átlaghoz hasonló módon változik.

Kiértékelésnél figyelembe veszi volt-e már a periódus során értékelhető zöld sorozat.

A továbbiakban meghatároztam a mérés során sárga és zöld típusú, vagyis alacsony értékek mozgó átlagát, a véletlen kiugró értékek kiküszöbölése érdekében. De sajnos 1-1 magasabb érték még így is kiugróan módosítja és a tendencia csak hosszútávon figyelhető meg, így nem bizonyult alkalmasnak.

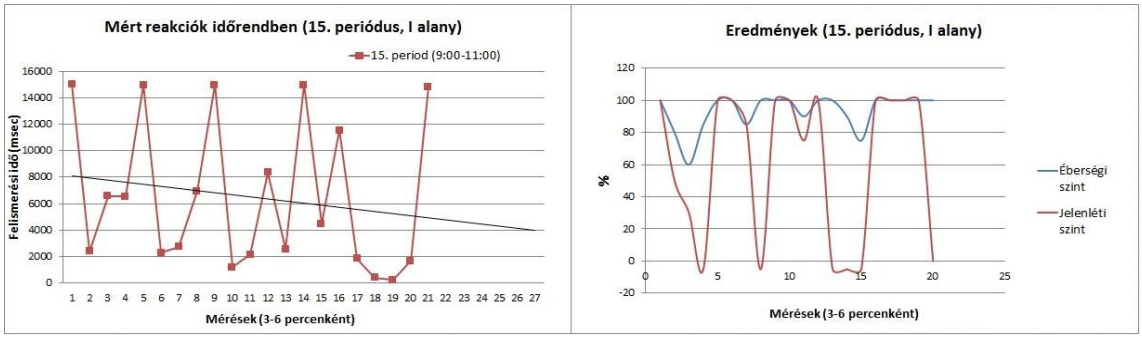
A periódusra jellemző gyakoriság vizsgálat a mérések alacsony száma miatt bukik el, így is az első eredmény csak a 3. mérés.

6.4 Eredmények értékelése

A mérések során az utolsó 3 eredmény sorozatából, ezek átlagából, az utolsó eredmény értékéből, a teljes periódusra vonatkozó jó és közepes eredményekből és ezek átlagából próbálok helyes következtetésre jutni és megállapítani az éberségi és jelenléti szintet. A szabályokat lefuttattam az összes ismert mintán.

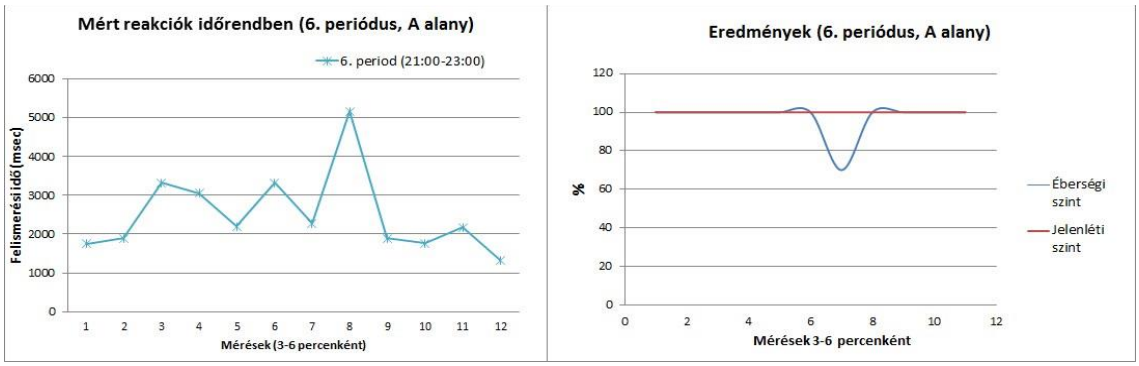
Az eredmények értékelése során normál körülmények között jó reakció idővel triviálisan meghatározható mindkét érték 100%-ra, nincs hibás felismerés.

A következő 6.8. ábrán I alany 15. periódusa látható, nagyon zavaró körülmények alacsony és egyre alacsonyabb reakció időkkel, tehát a figyelem teljes. És bár az elején nagy esélyt adott a reakció problémának, a jelenléti szint alacsonyabb. A jelenléti problémát minden esetben felismerte. Jelen követelmények mellett az éberség valószínűségi szintje mindig a jelenlét fölött maradt, nem produkált hibás riasztást.



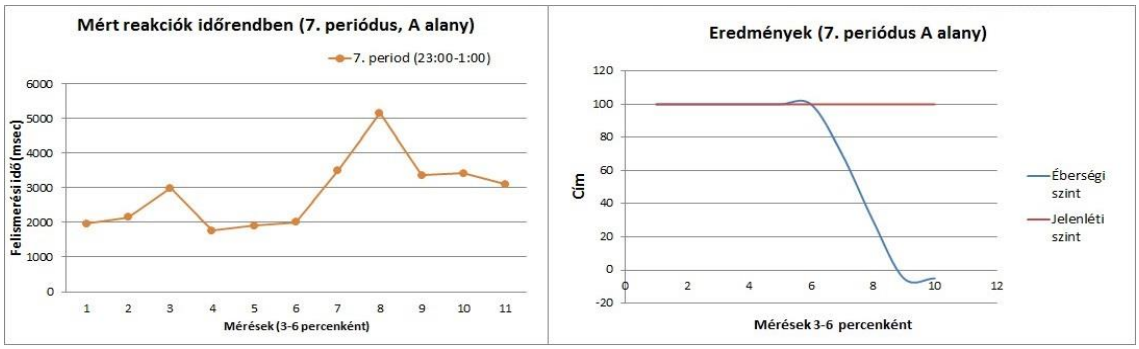
6.8. ábra: I alany, 15. periódus (9:00-11:00) éberségi és jelenléti szint zavaró körülmények között

Hasonló jellegű függvény, de a hiba szintje alacsonyabb, ebben az esetben nem a jelenléti szintnek kell csökkennie, hanem maximum az éberségi szintnek, ha nem akadályozza meg valamely más paraméter. Bár a reakciók szórása nagyobb, a nagyszámú, visszatérő jó eredmény megakadályozza, hogy reakció csökkenésként azonosítsa. Pillanatnyi megingás a magasabb, de mégis relatíve alacsony érték miatt látható (6.9. ábra).



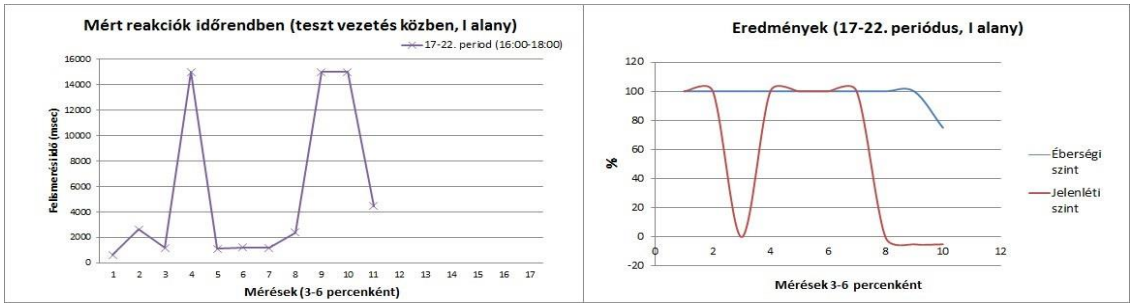
6.9. ábra: A alany, 6. periódus (21:00-23:00) éberségi és jelenléti szint, lokális hiba

Ebben az esetben konkrétan átesett az alany egy elalvási kapun, lefekvés előtt végezte a mérést. Egyértelműen meghatározható a reakció csökkenés egy hiba után, bár a várt 0,7 másodpercet meghaladja (6.10. ábra). Sajnos erre nem volt túl sok konkrét példa.



6.10. ábra: A alany, 7. periódus (23:00-1:00) éberségi és jelenléti szint, elalvás detektálás

A következő példa abból a szempontból érdekes, hogy a vezetés közben mért periódusok egyike. Látható hogy az alakja megegyezik a más esetben mért függvények alakjával. A 4. mérésnél a zavarás egyértelműen detektált. Az éberségi szint egyenletes, bár a végén, ha hasonló tendenciával emelkedne tovább, nagy eséllyel hamarosan riasztana (6.11. ábra).



6.11. ábra: I alany, 17-22. periódus (16:00-18:00) éberségi és jelenléti szint, vezetés közben

Ezzel a módszerrel a jelenléti szint nagy bizonyossággal megállapítható, valamint leolvasható egyértelmű fáradtság. A probléma nem a szembetűnő esetekkel van, hanem a hosszabb távú, nagyon kismértékű éberség csökkenéssel lehet, melynek vége a holtpont átlépése.

7 Összegzés

A feladat egy olyan Android alapú alkalmazás létrehozás volt, amely a vezető reakcióit vizsgálva segít a fáradtságból bekövetkező balesetek megelőzésében, mindezt biztonságos módon, a képernyő érintése nélkül.

Az érintés nélküli kezelés megvalósításához az alkalmazás gesztus felismerő része újraírásra került, mivel a korábbi verzióban nem működött elég megbízhatóan. Az új verzióban törekedtünk arra, hogy a felismerés minél kevésbé támaszkodjon a színekre, hogy ezáltal pontosabb eredményeket tudjunk elérni, változó körülmények esetén is. A gesztus felismerő modul, OpenCV segítségével, C++ nyelven lett megírva, így ki tudja használni a rendelkezésre álló erőforrásokat, hála az Android natív keretrendszerének. Sajnos az alkalmazás ezen részét nem sikerült teljesen befejezni, így az utolsó lépés, a gesztus felismerése, még nem került implementálásra. Mivel nem tudtam még egészben tesztelni a programot, ezért csak részleges teszteredmények állnak rendelkezésre, de ezek alapján az látszik, hogy megbízhatóan azonosítható a kéz, azon eseteket leszámítva, amikor a háttér világosabb (például fehér, világos barna), mint a kéz a képen.

Megvalósult egy, a kézfelismerő technikát intergálni képes, Android alkalmazás, melynek feladata a vezető és a jármű monitorozása. Visszaszámlálás után bizonyos időközönként kézjeleket kér a vezetőtől és rögzíti a reakcióidőt. Megvalósult a felhasználókezelés, az eredmények mentése, és megjelenítése grafikonok formájában, az éjjeli és nappali mód, valamint a kamerák közötti váltás, és a háttérben futtatás, így nem zavarja más alkalmazások egyidejű használatát. Összegyűjti és továbbítja a reakció időket, jármű és helyadatokat a szerverre további feldolgozásra.

A reakció idők vizsgálata során megvalósítottam a tesztelési környezetet és a széleskörű adatgyűjtést, több tesztalanytól. Megvizsgáltam a különböző időben és körülmények között végzett mérési adatok eloszlását, a függvények alakját. Levontam a következtetéseket és kialakítottam egy szabályrendszert melynek segítségével kellő pontossággal meghatároztam az éberségi és zavarási szint valószínűségét.

A továbbiakban tervezzük a natív modul és az Android-os alkalmazás egyesítését, miután mindkettő elkészült. Ha ez megtörtént, akkor intenzív tesztelésnek kell alávetni az alkalmazást, majd az így megtalált hibákat és gyengeségeket ki kell javítani. A tesztelésnek része lehet, hogy az alkalmazás egy egyszerűbb változatát ingyenesen elérhetővé tesszük a Google Play-en, béta üzemmódban. Ha sikerült a hibákat kijavítani és az alkalmazás szinte fennakadás nélkül működik, akkor megjeleníthető egy teljes verzió, valamint el lehet kezdeni dolgozni más platformok támogatásán is.

Irodalomjegyzék

[1] Ekler Péter, Fehér Marcell, Frostner Bertalan, Kelényi Impre: *Android-alapú szoftverfejlesztés,* Szak Kiadó 2012

[2] Kékesi Péter: *Az Android Platform natív programozói felületének vizsgálata és teljesítmény mérése,* BME-AUT Szakdolgozat 2013

[3] Android NDK (2014. december)
https://developer.android.com/tools/sdk/ndk/index.html#revisions

[4] Teljes C és C++ támogatás androidon – cikk (2011 január)
http://androidhungary.com/2011/01/teljes-c-es-c-tamogatas-androidon/

[5] OpenCV for Android SDK (2014. december)
http://opencv.org/platforms/android.html

[6] Dr. Abonyi János: *Adatbányászat a hatékonyság eszköze*, ComputerBooks 2006

[7] Prashan Premaratne: *Human Computer Interaction Using Hand Gestures*, Springer Science & Business Media 2014

[8] Microsoft Research Asia, Chinese University of Hong Kong: *Realtime and Robust Hand Tracking from Depth*, 2014

[9] Balogh Tamás, Ujj Tamás István: *ICT keretrendszer hálózatba kapcsolt jármű környezetben*, (TDK 2014)

[10] Háromdimenziós képalkotás a NI LabVIEW segítségével, 2013
http://www.ni.com/white-paper/14103/hu/

[11] Veszélyes Autóvezetés: alvás a volánnál (2015 május)
http://tudaskozpont.allianz.hu/mobilitas/?2348/Veszelyes-autovezetes-alvas-a-volannal

[12] OpenCV hivatalos weboldalak (2015 szeptember)
http://itseez.com/OpenCV/
http://opencv.org/

[13] Anti Sleep alkalmazás – Google Play (2014. december)
https://play.google.com/store/apps/details?id=em.software.antisleep

[14] Alvásterápia - Előzzük meg a baleseteket (2015 május)
http://alvasterapia.hu/hirek/elotildezzuk_meg_a_baleseteket9

[15] Az alvás és alvási fázisok (2015 május)
http://www.webbeteg.hu/cikkek/apnoe/13816/alvas-alvasi-fazisok

[16] Vezetés fáradtan: a kialvatlanság veszélyei (2015 május)
http://hegylakok.hu/2011/03/cikkek/vezetes-faradtan-a-kialvatlansag-veszelyei

[17] Közlekedésbiztonsági szemle – Emberi tényező (2015 május)
http://www.kozszemle.hu/index.php?o=emberi_tenyezo&cikk=97

[18] Arcizomfigyelő kamera előzheti meg az elalvást (2015 május)
http://www.stop.hu/tudomany/arcizomfigyelo-kamera-elozheti-meg-az-elalvast/971757/

[19] Hogyan maradjunk frissek a volán mögött (2015 május)
http://www.vezess.hu/vezetunk/hogyan_maradjunk_frissek_volan/25800/

[20] Otsu algoritmus (2015 szeptember)
http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

[21] Hang-Bong Kang: *Various Approaches for Driver and Driving Behavior Monitoring: A Review*, IEEE 2013

[22] Drive safe with Anti-Sleep Alarm
http://intersections.challengepost.com/submissions/26488-anti-sleep-alarm

[23] Ács Pongrác, Pintér József: *Bevezetés a sportstatisztikába*, Dialóg Campus Kiadó 2011

[24] Khashbat.J, Tsevegjav.Ts, Myagmarjav.J, Bazarragchaa.I, Erdenetuya.A, Munkhzul.N: *Determining the Driver's Reaction Time in the Stationary and Real-Life Environments*, IEEE 2013

[25] Zhong Zhang, Yuki Asakawa, Takashi Imamura, Tetsou Miyake: *Experiment designfor measuringdriver reaction time in driving situation*, IEEE, 2013

[26] Qingwan Xue, Yuting Zhang, Xuedong Yan, Bin Wang: *Analysis of Reaction Time During Car-following Process Based on Driving Simulation Test*, IEEE 2015

[27] Yi Lu Murphey, Robert Milton, Leonidas Kiliaris: *Driver's Style Classification Using Jerk Analysis*, IEEE, 2009

[28] Dr Masahiro MIYAJI, Dr Haruki KAWANAKA, Dr Koji OGURI*: Internet-Based Survey on Driver's Psychosomatic State for the Creation of Driver Monitor Function*, Aichi Prefectural University, IEEE

[29] Joseph F. Coughlin, Bryan Reimer, Bruce Mehler: *Monitoring, Managing, and Motivating Driver Safety and Well-Being*, Massachusetts Institute of Technology, IEEE 2011

[30] Application Crash Report for Android (2015 október)
http://github.com/ACRA/acra

[31] Chart library for Android (2015 október)
http://github.com/PhilJay/MPAndroidChart

[32] Parse felhő alapú szolgáltatások (2015 október)
http://www.parse.com/