BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

GÉPÉSZMÉRNÖKI KAR

Szabó Máté András

TUDOMÁNYOS DIÁKKÖRI KONFERENCIA DOLGOZAT

Detektált úthibák távolságának meghatározása sztereó kamera és LiDAR segítségével

Konzulens:

dr. Urbin Ágnes
adjunktus

Tizedes László
informatikus mérnök

Budapest, 2023.

Tartalomjegyzék

1. fejezet

Bevezetés

Az elmúlt években a járműipar egyre nagyobb hangsúlyt fektetett az önvezető autók kifejlesztésére. Az egyik legfontosabb feladata ezeknek az autonóm eszközöknek a környezetük azon tulajdonságainak az érzékelése, amiknek az ismerete elengedhetetlen a megfelelő irányítás kialakításához. Az útfelület hibái és a különböző akadályok fontos részei ennek a környezetnek. A személygépkocsik nagy sebességgel történő kátyúba vagy fekvőrendőrre hajtása komoly veszélyeket jelent mind az utas, mind a járókelők részére.

Úthibák detektálására már számos megoldás született, különböző módszerek felhasználásával. Az általam használt programot a Számítástechnikai és Automatizálási Kutatóintézet (SZTAKI) munkatársai fejlesztették ki. Ez vizuális alapon, gépi látás segítségével képes valós időben detektálni az előre meghatározott objektumokat. A kátyúk, fekvőrendőrök és egyéb akadályok észleléséhez felügyelt tanulással egy neurális hálózatot hoztam létre. Ennek segítségével az autó képes érzékelni ezeket és a sebesség redukálásával vagy a befutott pálya módosításával csökkenteni egy baleset veszélyét.

Az objektumok érzékelése azonban nem elegendő, hiszen az önvezető járműnek ismernie kell azok pozícióit is. Ezt elsősorban az autó és az akadály távolságának megmérésével lehet meghatározni. Erre a feladatra a munkám során két különböző megoldást teszteltem. Mivel a detekció egyébként is vizuális alapon működik, így használhatunk sztereó kamerát, ami ki tudja számítani az egyes képpontok távolságát a lencsétől. A másik lehetőség egy lézer alapú távérzékelő (LiDAR) alkalmazása, aminek a pontfelhőjét összefuzionálva a detektált objektumokkal meg tudjuk határozni azok távolságát.

A kutatásom fókuszában ezen két távolságmérési módszer összehasonlítása áll. Ehhez a két mérőprogramot szinkronizálni kell térben és időben is. Az időbeli szinkronizáció garantálja, hogy az egyes mérések ugyanazon pillanatban történjenek. Így elérhető, hogy az eredmények közötti eltérések csupán az érzékelés módjából adódjanak. A térbeli összehangolás a távolság meghatározásához használt képpont definiálását takarja. Ez

többféleképpen is történhet, például mediánnal, átlaggal, szélsőértékekkel vagy előre meghatározott pont kiválasztásával. Ezek összehasonlítása és közülük a legmegfelelőbb megoldás kiválasztása fontos elemei a dolgozatomnak.

A teszteket különböző eszközökkel végeztem el, majd az eredményeket számos szempont alapján értékeltem ki. Meghatároztam a sztereó látás és a lézeralapú távolságmérési módszerek egymáshoz viszonyított relatív hibáját. Feltérképeztem a hibák nagyságát az egyes távolságtartományokban, illetve az egyes objektumokra is különkülön. Megvizsgáltam, hogy a negatív vagy a pozitív tévedés-e a jelentősebb az egyes eszközöknél.

Kielemeztem a sztereó kamera karakterisztikáját és számításokkal bizonyított javaslatokat tettem annak javítására a pontosság növelése érdekében. Összehasonlítottam két LiDAR szkennelési metódusait is és megvizsgáltam, hogy azok eltérései milyen hatással vannak a távolságmérés pontosságára. Végül konklúzióval szolgáltam azt illetően, hogy melyik eljárás és milyen szempontok alapján alkalmasabb önvezető autók által detektált úthibák távolságmérésére.

A dolgozatom a témához kapcsolódó releváns irodalom, a használt mérőeszközök és a SZTAKI munkatársai által kifejlesztett valósidejű objektumdetektálásra alkalmas program bemutatásával kezdődik. Ezt követően a neurális hálózat létrehozását mutatom be, ami magába foglalja az adatok gyűjtését, illetve a hálózat tanítását és kiértékelését. A következő fejezet a mérőprogramok szinkronizálásáról, illetve a távolságmérési módszerek összehasonlítására használt program fejlesztéséről szól. Végül bemutatom és kiértékelem a különböző szempontok alapján mért eredményeket.

2. fejezet

Irodalomkutatás

2.1. Önvezető járművek

Az autonóm jármű egy többszenzoros, intelligens, döntéshozó és beavatkozó képességgel rendelkező rendszer [1]. Ezeknek a használata az elmúlt években az okos városok ötletével párhuzamosan fejlődtek ki [2]. Az önvezető jármű egyik legfontosabb feladata a környezetének az érzékelése. Az így nyert információk alapján tud ugyanis döntést hozni és annak megfelelően beavatkozni az autó sebességébe vagy haladási irányába.

A környezet érzékelése különböző szenzorok, leggyakrabban kamerák vagy lézer alapú távérzékelők (LiDAR) segítségével történik [3]. Az egyes szenzorok kalibrációja elengedhetetlen a megfelelő működéshez, hiszen ez biztosítja, hogy a környezetről valós képet kapjon a jármű.

A különböző objektumok detektálása a legfontosabb célja ezeknek az eszközöknek. Legyen szó emberekről, autókról, közlekedési táblákról vagy kátyúkról, az autónak ezek jelenlétét és pozícióját ismerve kell működnie. Az objektumok valósidejű detektálására az egyik leggyakoribb módszer a konvolúciós neurális hálózatok alkalmazása [4].

2.2. Kátyúdetektálás módszerei

Kátyúk detektálására számos alkalmazást fejlesztettek már ki. Ezek három kategóriába sorolhatóak az érzékelés módjától függően [5].

A vibráció alapú módszer [6] az autóba szerelt gyorsulásszenzor vertikális értékei alapján érzékeli, ha a jármű kereke belehajtott egy kátyúba. Ez a legkisebb költségű és legkevesebb számítási igényű megoldás. A módszer hátránya, hogy sem a kátyú méretét és alakját nem tudja megállapítani, sem az autó előtt elhelyezkedő objektumok detektálására nem alkalmas.

Egy másik megoldás a kátyúk detektálására a 3D rekonstrukció alapú módszer [7], ami az autó körülötti tér feltérképezésén alapul. Ehhez általában valamilyen sztereó kamerát vagy lézeres szenzort használnak. Ez a megoldás hatékony a kátyúk érzékelésében, azonban az eszközök magas költsége és a program nagy számítási igénye komoly hátrányt jelent. Szintén probléma az esővel, hóval vagy földel telített kátyúk érzékelése, melyre ez a módszer nem alkalmas.

A harmadik megoldás a vizuális alapú módszer [8], ami képfeldolgozási és deep learning algoritmusokkal működik. A kátyúk detektálása egy kamera által felvett képen egy neurális hálózat által történik. Ez a módszer költséghatékony és képes hatékonyan felismerni az objektumokat a jármű előtt is. Nagy hátránya azonban, hogy a kátyúk alakja és távolsága nem meghatározható, hiszen kétdimenziós képeket használ. Az időjárás viszontagságaira, árnyékokra és különböző megvilágításokra szintén érzékeny ez a módszer.

2.3. YOLOv4 neurális hálózat

A neurális hálózat egy olyan matematikai modell, ami az emberi agy működését veszi alapul [9]. A biológiai axonok, dendritek és szinapszisok gépi megfeleltetésével egy olyan rendszer építhető, ami számos komplex művelet elvégzésére képes. Ezek közül az egyik a különböző adatminták felismerése, aminek a képfeldolgozásban és a gépi látásban nagy hasznát vehetjük.

A gépi látásnak három fő fajtája létezik [10]. A klasszifikáció során a kép bemenetre csupán egy bináris kimenetet ad a neurális hálózat az alapján, hogy a keresett objektum szerepel-e rajta vagy nem. A detektálás során a neurális hálózat bekeretezi és a nevével felcímkézi az összes felismert objektumot. A szegmentálás hasonló a detekcióhoz, itt azonban nem egy befoglaló téglalapot, hanem az objektum felületét kapjuk meg formától függetlenül.

A konvolúciós neurális hálózat (CNN) [11] a hagyományos neurális hálózat egy továbbfejlesztett verziója. Ezek úgynevezett konvolúciós és pooling rétegekkel rendelkeznek, amik összessége képes a képek lényeges részeinek kiemelésére és így a detektálandó objektumok felismerésére.

A YOLO (You Only Look Once) [12] konvolúciós neurális hálózat kifejlesztése egy hatalmas ugrás volt a deep learning alapú gépi látás történetében. Ennek a lényege, ahogy a neve is sugallja, hogy a neurális hálózat csupán egyszer tekinti meg a képet. Így az objektumok detektálása valójában egy egyszerű regressziós problémára redukálódik, amiben a képpontokból egyből meghatározhatóak a befoglaló téglalapok és az osztálycímkék. A detekciós feladat egyszerűsítése következtében a YOLO neurális hálózat

sokkal gyorsabban, nagyobb FPS értékkel képes futni [13]. Ez a tulajdonsága lehetővé teszi a valósidejű alkalmazásokban való használatra. Összehasonlítva egy másik népszerű neurális hálózattal, a Fast R-CNN-el kimutatható, hogy a YOLO algoritmus 90-szer annyi futásra képes egységnyi idő alatt. A gyors felismerés azonban a pontosság rovására megy. A befoglaló téglalapok lokalizálása és a sok egymáshoz közel elhelyezkedő objektum detektálása kevésbé pontos, mint más neurális hálózatok esetében.

A YOLOv4 a bemutatott YOLO neurális hálózat egyik verziója. Ez az MS COCO adatbázison tesztelve 41,2%-os mean average precision-t (mAP) ért el. A YOLO algoritmus a Picasso adatbázison tesztelve 0,59-es F1 értéket ért el. Ezeknek a mérőszámoknak jelentését a 2.4 fejezetben mutatom be.

2.4. Neurális hálózatok kiértékelésének mérőszámai

Neurális hálózatok kiértékelésére számos mérőszámot használnak [14]. Ezek közül a legfontosabb a konfúziós mátrix (2.1 táblázat), ami egy adott teszt adathalmazon számolja össze a detekciók eredményeit és azokat négy csoportba rendezi. A True Positives (TP) olyan detekciókat tartalmaz, ahol az objektum mind a valóságban, mind a neurális hálózat predikciójában jelen van. A True Negatives (TN) azokat az eseteket tartalmazza, amikor a valóságban nincs objektum és a neurális hálózat sem talál. A False Positives (FP) azokat az eseteket számolja össze, amikor a neurális hálózat tévesen jelzi, hogy egy objektum létezik, hiszen a valóságban ez nem így van. Ezt másik nevén egyes típusú hibának is nevezik. A False Negatives (FN) olyan detekciók csoportja, ahol a valóságban ugyan van objektum, azonban a modell ezt nem jelzi. Ezek a második típusú hibák. A négy értéket mátrixos elrendezésben, az egyes osztályokra külön-külön szokták ábrázolni.

	Pozitív valóság	Negatív valóság
Pozitív jelzés	TP	FP
Negatív jelzés	FN	TP

2.1. táblázat. A konfúziós mátrix

A Precision (P) és a Recall (R) egy adott osztályra jellemző mérőszámok. Előbbi az adott osztály TP értéke osztva az összes predikció, vagyis a TP és FP összegével. Utóbbi az adott osztály TP értéke osztva a összes valódi objektum számával, vagyis a TP és az FN összegével.

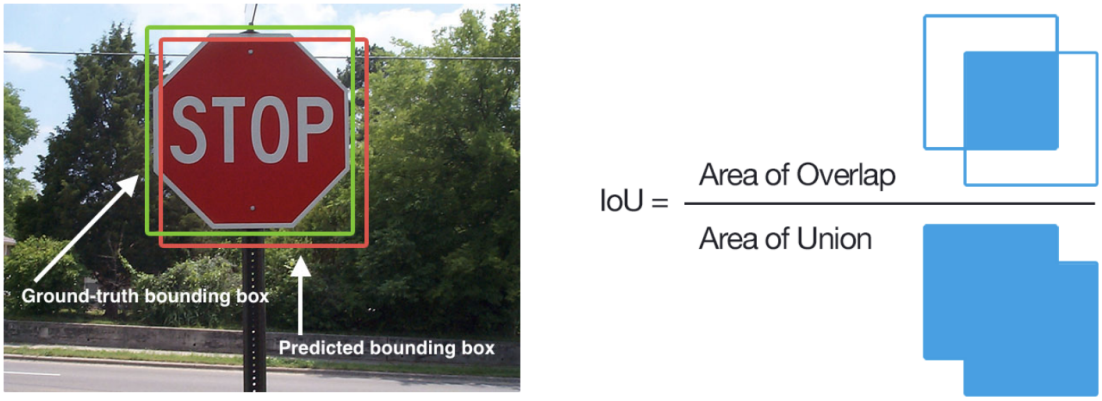
$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

A P és R értékek kapcsolatát azok harmonikus közepe írja le. Ez az úgynevezett F1 érték, ami a megadott képlet alapján számítható. Minél nagyobb ez az érték, annál jobb minőségű az osztályozó modellünk. Az egyes osztályokra vett értékekből átlagot vonva az egész neurális hálózatot is jellemezhetjük.

$$F1 = \frac{2 * P * R}{P + R}$$

Az Intersection over Union (IoU) egy detektálásról mondja el annak jóságát az eredeti címkézéssel összevetve. A mérőszám a két téglalap metszetének területe osztva azok uniójának területével (2.1 ábra).



2.1. ábra. Az IoU megegyezik az előre jelzett és a valós befoglaló téglalap metszetének és uniójának hányadosával [14]

Az Average Precision (AP) szintén egy adott osztály jellemzője. A bevezetése azért szükséges, mert a P és az R értékek nagyban függenek a beállított IoU határértéktől. Ezt kiküszöbölendő előállíthatjuk az összes 0-tól 1-ig lévő IoU értékre a P és az R értékét. Ezeket görbébe rendezve és megmérve az alattuk található területet megkapjuk az AP értéket. Ezt az integrálás helyett sokszor n darab pont felvételével és az ahhoz tartozó értékek átlagolásával helyettesítik.

$$AP = \int_0^1 P(R)\, dR$$

Az n darab osztály AP értékeiből átlagot vonva megkapjuk a Mean Average Precisiont (mAP), ami immár a teljes neurális hálózatunk általános jellemzője.

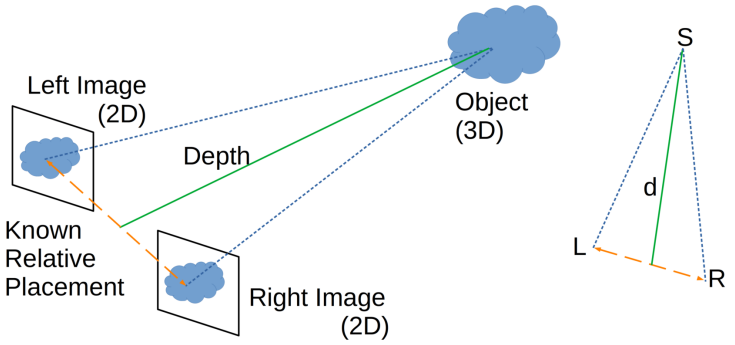
$$mAP = \frac{1}{n} * \sum_{k=1}^{n} AP_k$$

3. fejezet

Mérőeszközök

3.1. Sztereó kamera

A háromdimenziós tér feltérképezésére használt egyik leggyakoribb módszer a sztereoszkópia [15]. Ez az emberi látást imitálja, ahol a két szem helyett két, egy irányba néző kamera érzékeli a környezetet. A két egyidőben készített kép és a kamerák egymáshoz viszonyított helyzetének ismeretében háromszögeléssel kiszámítható az egyes képpontok távolsága (3.1 ábra).



3.1. ábra. A háromszögelés módszere a két kép és a kamerák közötti ismert távolság alapján kiszámítja az objektum távolságát a bázisvonaltól [16]

A sztereó kamera [17] egy olyan eszköz, ami ezt a két kamerát egy hardverbe ágyazva, sztereó látás segítségével érzékeli a környezetet. Ez több szempontból is előnyös. Egyrészt a két kamera egymáshoz viszonyított fix helyzete miatt általunk elvégzett kalibrációra már nincs szükség. Másrészt a magas számításigénnyel rendelkező háromszögelés műveletet nem egy külön számítógép, hanem maga az eszköz végzi.

A sztereó kamera egyik hátránya, hogy az érzékelés erősen függ a megvilágítási viszonyoktól. Ennek kiküszöbölésére gyakran aktív szenzort használnak, ami extra fényforrást biztosít. Másrészt összehasonlítva a többi 3D érzékelő módszerrel megállapít-

ható, hogy a távolságmérés kevésbé pontos ezeknél az eszközöknél. A sztereó kamerák nagy előnye viszont, hogy a berendezés viszonylag olcsó, illetve a kamerák képet is készítenek, amik felhasználhatóak további alkalmazásokra, például vizuális alapú objektumdetekcióra. A sztereó kamera az egyik leggyakoribb szenzortípus önvezető autók esetén.

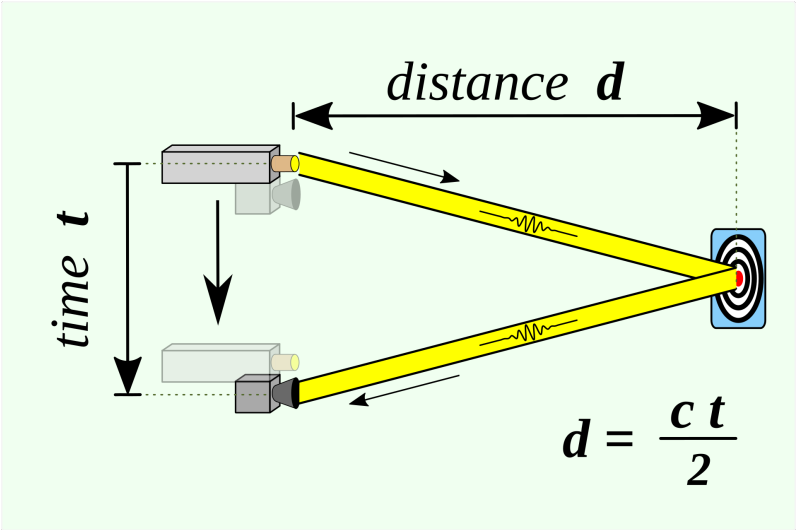
Én egy Stereolabs ZED 2 [18] sztereó kamerát használtam. Az eszköz nagy újítása, hogy neurális hálózat által segített távolságérzékelésre képes. A számomra lényeges gyári adatok a 3.1 táblázatban találhatóak. A távolságmérés pontossága előre beállított ideális környezetben mért érték, a valós alkalmazásokban ez eltérő lehet. A ZED 2 kamera a Robot Operating System (ROS) [19] keretrendszerbe is integrálható.

Minimális távolság	0,3 m
Maximális távolság	20 m
Pontosság	1% - 5%

3.1. táblázat. A ZED 2 sztereó kamera gyári adatai [18]

3.2. Lézer alapú távérzékelő (LiDAR)

A háromdimenziós tér feltérképezésének másik módszere az úgynevezett Time-of-Flight (ToF) metódus [20]. Ennek a lényege, hogy egy emitter lézerimpulzust bocsát ki, ami az útjába kerülő tárgyról visszaverődve egy detektorba jut. A kibocsátás és az érzékelés között eltelt idő és az elektromágneses hullám terjedési sebességének segítségével megkaphatjuk a megtett utat, ami a távolságnak a kétszerese lesz (3.2 ábra).

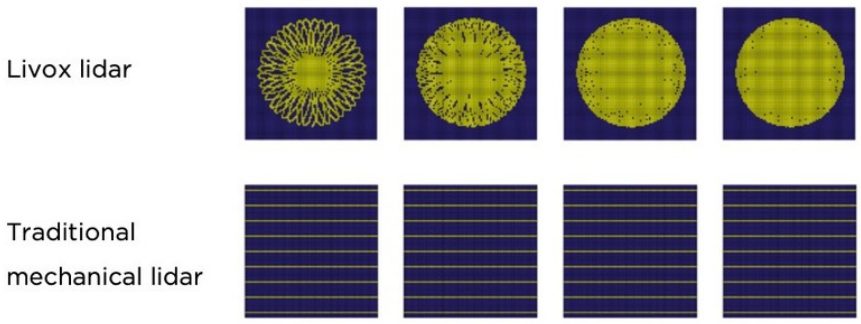


3.2. ábra. A Time-of-Flight módszer alapján az objektum távolságának kétszerese a lézerimpulzus emissziója és detekciója között eltelt idő és a terjedési sebességének a szorzata [21]

Ezt a módszert használják a lézer alapú távérzékelők (LiDAR) [22] is. Ezek az eszközök sok különböző irányba bocsátanak ki ilyen lézerimpulzusokat és a távolságokat meghatározva egy pontfelhőt készítenek a szenzor környezetéről.

Az ilyen eszközök egyik hátránya [23] az áruk, ami jóval magasabb, mint a korábban bemutatott sztereó kameráké. Cserébe azonban pontosabbak, illetve nagyobb területek is feltérképezhetőek vele. Egyrészt a maximálisan érzékelhető távolság is jóval nagyobb, másrészt bizonyos eszközök akár 360°-os látószöggel is rendelkezhetnek. Ezen tulajdonságai miatt elengedhetetlen kellékei az önvezető járműveknek [24].

Én két LiDAR-t, egy Ouster OS0-t [25] és egy Livox Avia-t [26] használtam. Az előbbi több lézeremitter körbeforgatásával tapogatja le a környezetét, ezáltal egy 360°-os pontfelhőt előállítva. Utóbbi ennél kisebb látószöggel, viszont különleges, nyolcas alakokat leíró szkennelési metódussal rendelkezik. Ezáltal a szenzor fókuszpontjában több, míg attól távolodva egyre kevesebb pont található a felhőben. Azonban, más szkennelési metódusokkal ellentétben az integrálási idő növelésével sűríthető a pontfelhő (3.3 ábra).



3.3. ábra. A Livox LiDAR és a hagyományos elven működő LiDAR-ok (például Ouster) szkennelési metódusai [27]

A két eszközre vonatkozó, számomra lényeges gyári adatok a 3.2 táblázatban találhatóak. A maximálisan mérhető távolság nagyban függ a tárgy fényvisszaverődésétől és a megvilágítástól. A távolságmérés pontossága előre beállított ideális környezetben mért érték, a valós alkalmazásokban ez eltérő lehet. Az Ouster és a Livox LiDAR is integrálható a ROS keretrendszerbe.

	Ouster OS0	Livox Avia
Minimális távolság	0,5 m	1 - 3 m
Maximális távolság	35 - 75 m	190 - 450 m
Pontosság	0,8 - 4 cm	2 cm

3.2. táblázat. Az Ouster OS0 és a Livox Avia gyári adatai [25, 26]

4. fejezet

Az úthibákat detektáló program bemutatása

4.1. A program működése

Az útfelületben lévő hibák detektálására a szakirodalom alapvetően három lehetséges megoldást mutat. A vibráció-, a 3D rekonstrukció- és a vizuális alapú módszerek a 2.2 fejezetben kerültek bemutatásra. A különböző metódusok előnyeit és hátrányait figyelembe véve megállapítható, hogy melyik megközelítés a legalkalmasabb valósidejű kátyúdetektáló algoritmus létrehozására. A vibráció alapú módszer csupán akkor képes érzékelni egy hibát az útfelületen, ha azon az autó már áthajtott. Emiatt ez nem alkalmas az irányítás kialakítására, hiszen annak az autó előtti objektumokat figyelembe véve kell döntenie. A 3D rekonstrukciós módszer használata szintén hátrányos lehet. A szenzor által feltérképezett pontfelhő kezelése nagy komplexitású számítást igényel, ami így lassítja a rendszerünket. Egy valósidejű alkalmazás esetén az idő kulcsfaktor, így ezt nem engedhetjük meg. A vizuális alapú módszer adja a legjobb megoldást objektumok detektálására, még ha a távolságmérés hiánya komoly problémát is jelent.

Ezeket a megfontolásokat figyelembe véve fejlesztettek ki a SZTAKI-ban egy programot, ami képes előre definiált objektumokat valós időben felismerni. Ez amellett, hogy a detekciót vizuális alapon végzi, amihez egy sztereó kamerát használ, annak segítségével az objektumok távolságát is meg tudja határozni.

A projekt a ROS keretrendszerben fut, ami egy köztes szoftver különböző eszközök irányítására, előre definiált ismétlődő funkciók implementálására és az egyes elemek közötti hatékony kommunikáció megvalósítására. Számunkra elsősorban az utóbbi funkció lesz lényeges, amivel meg tudjuk valósítani az adatok áramlását az eszközök között. Ezt úgynevezett node-ok segítségével érhetjük el, amik topic-okra tudnak publikálni, illetve onnan adatokat kiolvasni. A mérési összeállítás meglehetősen egyszerű, egy Stereolabs

ZED 2 sztereó kamerát és egy NVIDIA Jetson AGX Xavier [28] beágyazott rendszert tartalmaz. Előbbi az autó tetejére van rögzítve és egy node segítségével elküldi a rögzített képeket a számítógépnek.

A beágyazott rendszeren egy program fut, ami felelős az adatok fogadásáért, feldolgozásáért és a vezérlésnek való továbbküldéséért. A kód C++ nyelven lett megírva és nagyban támaszkodik az OpenCV-re [29]. Ez egy nyílt forráskódú függvénykönyvtár, aminek a segítségével gépi látást használó alkalmazások fejleszthetőek. A könyvtárból különösen nagy hasznát vesszük a DNN modulnak, ami neurális hálózatok kezelésére alkalmas.

Az algoritmus először kiolvassa a kamera által küldött négycsatornás képeket és átkonvertálja azokat BGR formátumúra. Ezután az éppen vizsgált képből egy úgynevezett blob-ot gyárt. Ez egy olyan adatstruktúra, ami a legtöbb neurális hálózat számára értelmezhető. Az átalakítás során a csatornák átlag kivonáson, normalizáláson és sorrendcserén esnek át. A blob struktúra már átadható a neurális hálózatnak bemenetként. A programban egy YOLOv4 konvolúciós neurális hálózatot használunk az objektumok detektálására. Ennek a tulajdonságait és használhatóságát a 2.3 fejezetben mutattam be.

A neurális hálózat eredményeként egy adott objektumra több, akár különböző osztályokba tartozó befoglaló téglalapot is kaphatunk. A helyes detektálások elérésének érdekében ezeket még posztprocesszálni kell, aminek keretében egy Non-maximum suppression algoritmust futtatunk rajtuk. Ez kiválasztja az egymást fedő téglalapok közül a legmegfelelőbbet, illetve az egyes osztályokra kapott konfidenciaszint alapján meghatározza az objektum típusát is.

Az adott képhez meghatározott összes objektum befoglaló téglalapját és osztályát a program visszamásolja az eredeti képre. Az így létrehozott, immár a detekciókkal jelölt képet megjeleníthetjük a ROS RViz nevű vizuális segédeszközével, lementhetjük egy előre definiált mappába, illetve továbbadhatjuk az vezérlés számára, mint hasznos információ. A program egészen addig végzi ciklikusan ezt a feladatot, amíg a sztereó kamera adatokat szolgáltat vagy amíg le nem állítjuk a futtatást.

A programot nem csak valós időben, hanem előre felvett *bag* fájlok alapján is tudjuk futtatni. Ez egy olyan ROS adatstruktúra, ami tartalmazza a felvétel idején az összes node által az összes topic-ra publikált adatot. Ilyen például a LiDAR által rögzített pontfelhő, a program által elkészített, a detektált objektumok befoglaló téglalapjait is tartalmazó kép vagy a ROS által létrehozott időbélyeg. A felvétel visszajátszható és a program közben futtatható az adatokon.

4.2. A detektált objektumok távolságmérése sztereó kamera segítségével

A SZTAKI-ban kifejlesztett program nem csupán előre definiált objektumok detektálására képes, hanem azok távolságának megmérésére is. Ez egy rendkívül hasznos funkció, hiszen egy önvezető autó esetében nem elég ismerni a környezetében lévő objektumok jelenlétét, azok pozíciója is fontos információval szolgál. Úthibák detektálásakor például nem feltétlenül kell lelassítani, amennyiben egy kátyú viszonylag távol van tőlünk. Abban az esetben, ha ez közvetlenül az autó előtt van már jóval fontosabb valamilyen beavatkozás.

A felhasznált program egy sztereó kamera segítségével tudja mérni az egyes objektumok távolságát. Ennek a működését a 3.1 fejezetben fejtettem ki. Az algoritmus a képet tartalmazó ROS topic-on kívül egy depth map-et is kiolvas. Ez egy olyan mátrix, aminek minden egyes eleme az adott képpont távolsága double formátumban, méterben mérve.

A Stereolabs ZED 2 kamera használatára számos beállítás létezik, melyek ismerete és konfigurálása elengedhetetlen a helyes eredmény eléréséhez. Számunkra az egyes képpontok távolságmérésének metódusa a lényeges ezek közül. Standard módban a depth map a valós, mért távolságokat tartalmazza. Amennyiben egy képpont esetén valamilyen optikai hiba vagy nem mérhető távolság lép fel, akkor ott a double szám helyett egy hibára figyelmeztető enum érték lesz. Ezt a kis számítási igénye miatt gyakrabban használják valósidejű rendszerekben, például önvezető autókhoz. A másik a Fill mód, ahol nincsenek nem értelmezett távolságok a mátrixban. Amennyiben egy képpontban a valós érték nem mérhető le, akkor azt a szomszédos távolságokból számolja ki a rendszer. Ezzel egy teljesen kitöltött depth map-et kapunk, ami azonban nagyobb számítási igénye miatt csak kisebb FPS értékkel tud frissülni. A mi alkalmazásunkban egyértelműen a Standard mód a megfelelő a gyors feldolgozás érdekében.

A program az egyes objektumokhoz valamelyik képpontjuk alapján távolságot társít. A képpont kiválasztása külön megoldandó probléma volt, amivel a 6.3 fejezetben foglalkozom. Az így kapott távolságértékeket a kimeneten, a detekciókat tartalmazó képeken is megjelenítjük.

4.3. A detektált objektumok távolságmérése LiDAR segítségével

Az előzőleg bemutatott megoldásban az egyetlen szenzora az önvezető autónak, amivel képes a környezetét érzékelni egy sztereó kamera volt. Ez rendkívül kényelmes megoldás, hiszen mind az úthibák és akadályok detekciója, mind a távolságuknak a mérése egy eszközzel történik. Ez nem csak költséghatékony, de az egyes elemek közötti transzformációtól és kommunikációtól is megkímél minket. Azonban a távolságmérés pontossága sztereó kamera segítségével igencsak kérdéses. A 7 fejezetben be fogom mutatni ennek a tényleges számszerűsített eredményeit is. A sztereó kamerával történő távolságmérés pontatlansága miatt érdemes más eszközökhöz is fordulnunk.

Erre a problémára válaszul született a SZTAKI-ban az eddig bemutatott projektnek egy továbbfejlesztése, amiben az úthiba-detekciókat továbbra is vizuális módszerrel, egy sztereó kamera képe alapján végezzük, azonban a távolságmérés már egy LiDAR-ral történik. Ez egy lézersugár úthosszmérésével működő szenzor, aminek a működését már a 3.2 fejezetben bemutattam. A mérés eredménye egy háromdimenziós pontfelhő, aminek az egyes elemei egy-egy távolságot jelölnek.

A távolságmérés átállítása sztereó kameráról LiDAR-ra a hardveres oldalon csupán még egy eszköz felszerelését jelenti az autóra. A szoftveres oldalon azonban jóval bonyolultabb feladat a probléma megoldása. A detektált objektumok távolságának méréséhez a detekciókat fuzionálni kell a pontfelhővel annak érdekében, hogy az egyes befoglaló téglalapokhoz valamilyen távolság értéket párosíthassunk. A háromdimenziós tér pontjait először át kell konvertálni a kép kétdimenziós síkjába. Ehhez a program az OpenCV egyik projektáló függvényét használja, ami paraméterként különböző kalibrációs mátrixokat vár. Ezek közül a kamerára a kamera mátrix és a torzítási koefficiensek vektora, a kamera és a LiDAR egymáshoz képesti elhelyezkedésére pedig a transzlációs és rotációs vektorok a jellemzőek. Utóbbiak határozzák meg a bázistranszformációt a két eszköz koordináta-rendszere között. Az így megkapott kétdimenziós síkban elhelyezkedő pontok közül már ki lehet választani azokat, amik egy adott befoglaló téglalapban találhatóak. A távolságként definiált képpont kiválasztását a 6.3 fejezetben tárgyalom. A program végén a meghatározott távolság megjelenítése a kimeneten, a detekciókat tartalmazó képen az előző fejezetben tárgyalt megoldással megegyező.

A SZTAKI-ban kifejlesztett program képes egyébként a sztereó kamera és a LiDAR közötti fúzió fordítottjára is. Ebben az alkalmazásban a pontfelhő kiszínezése történik a képpontok színe alapján. Erre a funkcióra az én kutatásom során nem volt szükség. A program egy Ouster OS0 és egy Livox Avia LiDAR-ral lett kipróbálva. Ezek működését a 3.2 fejezetben mutattam be. Mindkét szenzor a Stereolabs ZED 2 kamerával van összekalibrálva.

5. fejezet

Neurális hálózat

5.1. A neurális hálózat jellemzői

Az objektumok felismerésére neurális hálózatot kellett tanítanom. Erre a feladatra a YOLOv4 konvolúciós neurális hálózatot választottam, aminek a tulajdonságai a 2.3 fejezetében be lettek mutatva. Olyan valósidejű alkalmazások számára, mint az önvezető autó programja elengedhetetlen a gyors futás. Ennek a kritériumnak tökéletesen megfelel a YOLOv4, ami az egyik legnagyobb FPS érték elérésére képes a többi neurális hálózattal szemben. A hátránya, miszerint sok kicsi, egymáshoz közel eső objektumot nehezen érzékel külön-külön nem jelentős, hiszen az útfelület eltérései általában nem így helyezkednek el. Mint ahogy az irodalomkutatásban is be lett mutatva, a gyors kiértékelés a befoglaló téglalapok pontosságának kárára történik. Ezt a hibát sajnos muszáj elfogadnunk, igaz a távolság mérésére ez sem lesz jelentős hatással.

Az interneten számos előre tanított neurális hálózat elérhető, ami különféle objektumok felismerésére képes. A Pascal VOC adatbázison tanított 20, míg a COCO adatbázison tanított 80 osztályt különböztet meg [30]. Ezek számomra nem voltak felhasználhatóak, hiszen egyedi, ezek között nem szereplő objektumokat kellett detektálni. Ezeket az objektumokat öt osztályba soroltam be. Az első kettő a fekvőrendőr, mint akadály az úton és a kátyú, mint az útfelület eltérése. A maradék három a csatornalefolyó, a csatornafedő és a bicikliút jelzés. Ezekre azért volt szükség, mert az alakjuk és az útfelületen elfoglalt pozíciójuk miatt gyakran hibásan kátyúként voltak érzékelve. A használatukkal a hibás detekciók száma jelentősen csökkenthető.

5.2. Az adatbázis létrehozása

Az általunk definiált osztályokat felismerni képes YOLOv4 konvolúciós neurális hálózatot felügyelt tanítással lehet létrehozni. Ennek a módszernek a lényege, hogy nagy számú annotált adatot adunk a hálózatnak, ami a tanítási folyamat végén képes lesz a nyers adatokat is helyesen felcímkézni. A YOLOv4 esetében a nagy számú adat olyan képeket takar, amin a felismerendő objektumok szerepelnek. Ezeket egyrészt nyílt hozzáférésű, interneten felérhető forrásokból [31, 32], másrészt a saját méréseink alapján szereztem be. Utóbbi során egy autót a megfelelő szenzorokkal felszerelve vittünk el adatokat gyűjteni. A felépített adatbázis 1609 képet tartalmaz, melyeken különböző számú, nagyságú és fajtájú objektumok szerepelnek. A reprezentativitást a kamerák, azok pozíciói, az útfelületek típusai és az időjárási viszonyok sokfélesége is biztosítja.

A felügyelt tanuláshoz ezeket az adatokat annotálni kell. Ez a képeken lévő objektumok bekeretezését és címkézését takarja. Ennek az információnak a segítségével tudja a neurális hálózat azonosítani, hogy melyik képrész milyen objektumot tartalmaz. Az annotáláshoz az OpenLabeling [33] alkalmazást használtam. Ez egy olyan GUI-t biztosít, amivel az egyes képeken lévő objektumok bekeretezhetőek és az eredményeket Pascal VOC és YOLO típusú fájlokban adja vissza. Számomra az utóbbi volt hasznos. Ez egy szöveges fájl, ami egy adott képhez tartozik és mindegyik sora egy objektumot reprezentál. Az első szám az objektum osztályát jelenti. A következő kettő a befoglaló téglalap középpontjának két koordinátáját, míg az utolsó kettő a szélességét és a magasságát relatív értékben a teljes kép méreteihez viszonyítva adja meg. Ezek a fájlok a képekkel együtt olyan adatbázist alkotnak, ami a neurális hálózat tanításának alapjául szolgálnak.

5.3. A neurális hálózat tanítása

Az adatbázis felépítését követően annak alapján tanítani lehet a neurális hálózatot. Ehhez a Darknet [34] nevű programgyűjteményt használtam. Ez egy olyan keretrendszer, ami képes a YOLOv4 neurális hálózat tanítására, futtatására és tesztelésére.

A tanítási folyamatot speciális fájlok határozzák meg, amik helyes létrehozása elengedhetetlen a megfelelő működéshez. Ezek közül a legfontosabb a konfigurációs fájl. Ebben az osztályok számát, a neurális hálózat rétegeinek tulajdonságait és a tanítás hosszát tudjuk beállítani. Egy másik fájl tartalmazza az osztályok neveit sorban egymás alá írva.

A felépített adatbázist három részre kell osztani. A legnagyobb csoportot a tanításhoz használt képek alkotják. Ezeknek a jellemzőit és mintáit elemzi és tanulja meg a

neurális hálózat. A második csoportba olyan képek tartoznak, amiket az egyes paraméterek finomhangolására használhatunk. Ezekből ugyan sosem tanul a neurális hálózat, azonban indirekten hatással vannak a működésére. Emiatt kell a harmadik csoport, amibe a teszt képek tartoznak. Ezeken csak egyszer futtatjuk a neurális hálózatot, abból a célból, hogy kiértékeljük annak a jóságát.

A tanítási folyamathoz szükséges egy súly fájl is. Építhetnénk új neurális hálózatot a nulláról is, azonban ez jelentősen idő- és erőforrás-igényesebb lenne. Érdemesebb helyette transzfer tanulást használni, aminek a lényege, hogy egy korábban már bizonyos szintig előre tanított hálózaton folytatjuk a munkát. Ehhez egy nyílt hozzáférésű, 137 konvolúciós réteggel rendelkező hálózatot használtam [35].

Ha mindezek megvannak, akkor el lehet indítani a neurális hálózat tanítását. Ez egy rendkívül idő- és számítási kapacitás igényes feladat. A tanító program előnye, hogy az aktuális állapot gyakori mentésével lehetővé teszi a folyamat bármikori megszakítását. Ez egy ilyen nagy időigényű feladatnál kifejezetten hasznos. A tízezer iteráció végére érve eredményül kapjuk az immár tanított súlyfájlt, amit később az objektumok detektálására használhatunk.

5.4. A neurális hálózat kiértékelése

A tanított neurális hálózatot a 2.4 fejezetben bemutatott mérőszámok alapján értékelhetjük ki. Az általam létrehozott modell F1 értéke 0,56, míg az mAP 68,7%. Ezeket össze lehet vetni a 2.3 fejezetben bemutatott nagyobb adatbázisokon elvégzett tesztekkel. A YOLO algoritmus ezeken hasonló F1 értéket ért el, azonban az én modellem több, mint 25%-al nagyobb mAP értéket produkált.

Az egyes osztályok, mint fekvőrendőr, kátyú, csatornalefolyó, csatornafedő és bicikliút jelzés AP értékei a 5.1 táblázatban láthatóak. Megfigyelhető, hogy a neurális hálózat legnehezebben a kátyúkat ismeri fel. Ez azért van, mert ezek nagyon nehezen definiálható és körülhatárolható, valamint sok egyéb útfelület eltéréssel összetéveszthető objektumok.

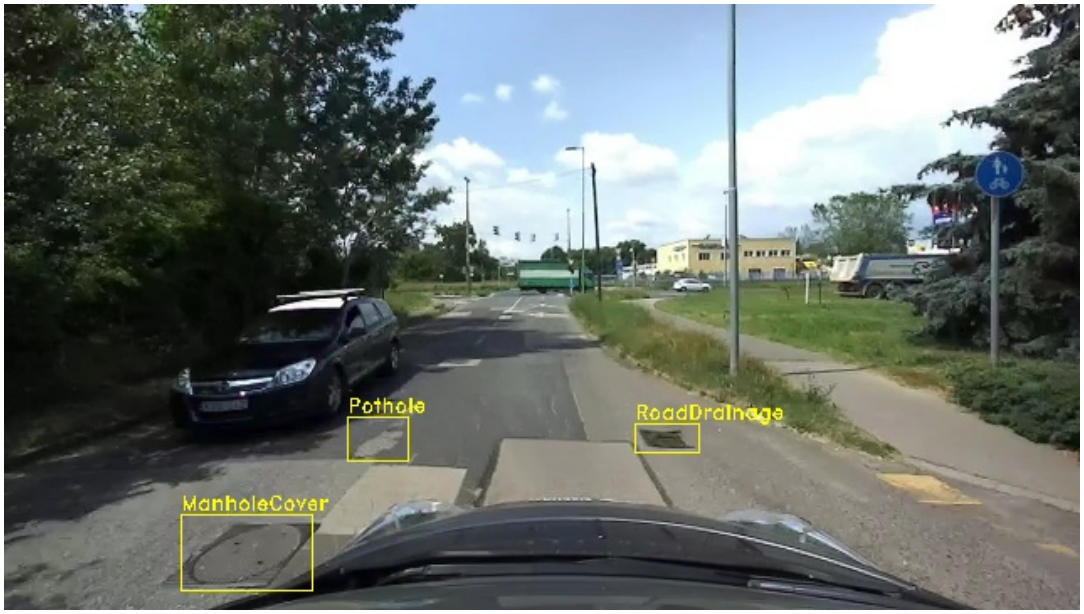
Fekvőrendőr	69%
Kátyú	52%
Csatornalefolyó	69%
Csatornafedő	87%
Bicikliút jelzés	66%

5.1. táblázat. Az egyes osztályok AP értékei

A neurális hálózat P és R értéke rendre 0,45 és 0,73. Ez szerencsésnek mondható, ugyanis a magas R érték azt jelzi, hogy ha van objektum a képen, akkor a modell

az esetek többségében visszajelez. A P alacsonyabb értéke, vagyis, hogy a detektálás sokszor nem valódi objektumra történik még számos módszerrel javítható. A kép egy adott részének figyelésével például kiszűrhetőek az útfelületen kívül felismert tárgyak, objektumkövető algoritmus implementálásával pedig az egy pillanatra megjelenő, hibás detekciók kerülhetőek el. Ezekkel kiegészítve a programot jelentősen növelni lehet a P értéket is.

A kiértékelésre használt mérőszámok értékei és az 5.1 ábra alapján kijelenthető, hogy az általam tanított neurális hálózat alkalmas önvezető autókban az öt előre definiált osztály detektálására.



5.1. ábra. A neurális hálózat detekciói valós környezetben

6. fejezet

Az úthibákat detektáló programok szinkronizálása

6.1. A távolságmérési módszerek összehasonlításainak feltételei

Az általam használt, a SZTAKI-ban kifejlesztett két program a 4 fejezetben kerültek bemutatásra. Ezek az úthibákat egy sztereó kamera képe alapján detektálják, azonban az objektumok távolságának meghatározását az egyiknél ez az eszköz, a másiknál egy LiDAR végzi. A kutatásom fókuszában ennek a két távolságmérő módszernek az összehasonlítása áll, így ezek a programok tökéletesek az adatok gyűjtésére.

Ahhoz, hogy ezeket a mérési adatokat összehasonlítsam előbb szinkronizálnom kellett a két rendszert térben és időben. Az időbeli szinkronizáció garantálja, hogy az egyes mérések ugyanazon pillanatban történjenek. Ennek köszönhetően elérhető, hogy az eredmények közötti eltérések csupán az érzékelés módjából adódjanak. A programok térbeli összehangolása az adott objektum távolságának definiálását takarja. Erre több megoldás is létezik, mint a képpontok átlaga, mediánja, szélsőértéke vagy egy előre meghatározott képpont távolsága. A kutatásom fontos feladata ezeknek a módszereknek az összehasonlítása és közülük a legmegfelelőbb kiválasztása.

6.2. A térbeli szinkronizálás módszerei

Egy detektált objektum távolságát annak képpontjai alapján sokféle módon definiálhatjuk. Mivel az én kutatásom a sztereó kamera és a LiDAR segítségével történő távolságmérés összehasonlításáról szól, így elengedhetetlen, hogy a két program távolság-definíciója egységes legyen. Így garantálható, hogy a mérési adatok közötti eltérések nem ebből,

hanem az érzékelési módszerek közötti különbségekből adódnak. Az SZTAKI-ban kifejlesztett két program távolság-definíciói nem voltak egységesek, így ezek szinkronizálására mindenképpen szükség volt. Másodlagos feladatként érdemes megvizsgálni a kiterjedt objektumok távolságának mérésére alkalmas módszerek eredményességét is és az eredmények alapján kiválasztani a számunkra legmegfelelőbb megoldást.

A távolság definiálására négy módszert vizsgáltam meg. Az első a detektált objektum befoglaló téglalapjának képpontjai közül valamely szélsőérték kiválasztása. Ez autonóm járművek esetén a biztonság irányába való tévedés elve alapján logikusan a legközelebbi pont. Ez azonban több szempontból sem megfelelő. Az objektum egy része takarásban lehet, akár a 6.1 ábrán látható módon az autó motorháztetője által is. Ilyen esetben a mért távolság nem a valóságot fogja tükrözni. Az objektum egy részének takarása nélkül sem megfelelő ez a megoldás, hiszen a mérési hibákból adódó kiugró értékekre nagyon érzékeny ez a módszer.

6.1. ábra. Egy részlegesen takart objektum detektálása

A másik lehetőség egy konkrét, előre definiált pont távolságmérése. Ez a detektált objektum befoglaló téglalapjának a középpontja. Ezt a módszert már csak nagyon ritka esetekben zavarja meg az objektum egy részének a takarása. Adódik azonban egy másik probléma, miszerint a téglalap középső pontjában nem feltétlenül vesz fel értelmes értéket a sztereó kamera depth map-je, illetve a LiDAR pontfelhőjének lehet, hogy nincs is ott pontja. Ennek kiküszöbölésére a sztereó kamera esetén a középponthoz legközelebb eső 25, míg a LiDAR esetén a legközelebb eső 5 pont mediánját vettem.

A harmadik módszer a detektált objektum befoglaló téglalapjába eső képpontok távolságainak az átlaga, míg a negyedik azok mediánja. Ezek abból a szempontból

előnyösek, hogy a mérésből és a takarásból adódó hibákat nagyban elnyomják, bár a befoglaló téglalap nagyobb takarása esetén ez az átlagolásra már nem feltétlenül igaz.

6.3. A térbeli szinkronizálás megvalósítása

Ahhoz, hogy kiválasszam a megfelelő módszert ki kellett értékelnem ezeket. Ezt az Ouster és a Livox LiDAR távolságértékeinek összehasonlítása alapján tettem. Mivel mind a két szenzor nagy precizitással rendelkezik, ezért az volt a feltételezésem, hogy a legpontosabb módszer esetén a legkisebb eltérést kapom a két eszköz között. Az általam vizsgált három módszer a befoglaló téglalap középpontjának, az átlagos távolságnak, illetve a medián távolságnak a mérése. A szélsőérték mérése a sok hiányossága miatt nem megfelelő módszer autonóm járművek számára, így azt elvetettem.

Az Ouster és a Livox LiDAR távolságmérése között fellépő relatív hibákat a három módszer esetén a 6.1 táblázat tartalmazza. Ezek alapján a legkisebb hibával és így a legpontosabb módszerrel a medián számítása rendelkezik.

Befoglaló téglalap középpontja	12%
Átlagos távolság	15%
Medián távolság	7%

6.1. táblázat. A távolságdefiníciók relatív hibái

A módszer előnye, hogy a kiugró értékekre, illetve az objektum nagy mértékű, akár közel 50%-os takarására sem érzékeny. A medián mérése kevés pont esetén is megfelelő eredményt ad, ami egy fontos tulajdonság a LiDAR pontfelhőjének változó pontsűrűségét figyelembe véve. A módszer hátránya, hogy a távolságot statisztikai úton közelíti meg, így az nem garantálható, hogy a különböző eszközök ugyanazt a képpontot mérik.

Fontos megvizsgálni ennek a pontatlanságnak a hatását az eredményre. Ezt a befoglaló téglalap szélsőértékeinek különbségével, tehát az objektum távolságtartományával és a befoglaló téglalapba eső pontok számával lehet meghatározni. A kiugró értékek kiszűrésére mind a kettőnek csupán a 80 százalékával számoltam. A pontok egyenletes eloszlását feltételezve a teljes tartományban az átlagos különbség két pont között a távolságtartomány osztva a pontok számával. Ennek az értéknek a fele az, amivel maximálisan eltérhet az adathalmaz mediánja az elméleti középponttól. Ezt kiszámolva két eszközre majd az értékeket összeadva és valamelyik távolságtartományra normalizálva megkaphatjuk, a két medián mérés közötti maximális, relatív eltérést. Ezeket az értékeket a 6.2 táblázat tartalmazza. Megfigyelhető, hogy a medián mérés pontatlansága

csak nagyon kicsi, az eszközök közötti különbségekhez képest elhanyagolható nagyságú hibát okoz.

	Medián mérés pontatlansága
ZED - Ouster	0,8%
ZED - Livox	0,2%
Ouster - Livox	0,9%

6.2. táblázat. A medián mérés pontatlansága két eszköz összehasonlítása esetén

6.4. Az időben való szinkronizálás

A két projekt időben való szinkronizálása azért szükséges, hogy a távolságmérések ugyanazon időpillanatban történjenek. Ezt a programok ciklusidejének összehangolásával meg lehetne oldani. Ez azonban körülményes és legalább az egyik mérési módszert lassítani fogja. Ehelyett érdemes a távolságméréseket időbélyeggel ellátni, majd ezek alapján összehasonlítani az azonos bélyeggel rendelkezőket. Ebből a szempontból a programok felépítése előnyös is, hiszen mindkét távolságértéket a sztereó kamera detektálása alapján mérjük. Ezáltal a képfrissítési periódusidőn belül mindig lesznek egyező időbélyegű távolságértékeink.

Az időbélyeget szolgáltathatná egy beépített függvény is, azonban ez a két program különböző inicializálási feladatainak eltérő futásideje miatt pontatlanságot okozna. A legjobb megoldás, ha az időbélyeget az összehasonlításra felvett adatokat tartalmazó *bag* fájl szolgáltatja. Ez ugyanis pontosan azt az időpillanatot tartalmazza nanoszekundum pontossággal, amikor az adott távolságmérés történt.

Az előre felvett *bag* fájlokat visszajátszva elvégezhetjük a távolságméréseket valamelyik eszköz alapján. Ezeket a mérésadatokat a hozzájuk tartozó időbélyeggel együtt egy CSV fájlba írja ki a program. Ez egy olyan fájltípus, amiben egy táblázatos formához hasonlóan, csak sortöréssekkel és vesszőkkel elválasztva tárolhatunk adatokat. Ezt a formátumot előszeretettel használják mérésadatok tárolására.

A CSV fájlba a mért távolságon és az időbélyegen kívül még a befoglaló téglalap pontjainak minimális és maximális értéke, a pontok száma, illetve a téglalap vízszintes és függőleges irányú koordinátája is bekerül. Előbbiek a 6.3 fejezetben bemutatott medián mérés hibájának kiértékeléséhez, utóbbiak pedig a pozíciófüggő kiértékelésekhez szükségesek.

6.5. A kiértékelő program létrehozása

Miután mind a két program módosítva lett az előző fejezetekben bemutatott szempontok alapján, akkor az összes *bag* fájlon, az összes eszközzel elvégezhetjük a távolságméréseket. Eredményképpen olyan CSV fájlokat kapunk, ami egy adott *bag* fájl és egy adott eszköz mérésadatait tartalmazza.

Ezeknek a kiértékelésére egy kétlépcsős programot készítettem Python nyelven. Ez először végigmegy a CSV fájlokon és string műveletek segítségével kiszedi a benne található adatokat. A két különböző eszköz által mért távolságokat az időbélyeg és az azonos objektum osztály alapján összekapcsolja és elmenti egy új CSV fájlba. Ezek a fájlok már nem egy, hanem kettő eszköz mérési adatait tartalmazzák. A program ezeken is végigmegy és különböző szempontok alapján kiértékeli a két távolságértéket. A szempontok a 7.1 fejezetben lesznek bemutatva.

A program a futás végén kiírja az eredményeket, amik a két eszköz összehasonlítását jellemzik. Bizonyos értékeket kettő és háromdimenziós grafikonokon is megjelenít a könnyebb vizualizáció érdekében. Fontos fejlesztési feladat volt, hogy a program kiértékelési szempontjai könnyen és széles körben változtathatóak legyenek.

7. fejezet

A távolságmérési módszerek összehasonlítása

7.1. Az összehasonlítás szempontjai

A távolságméréseket három különböző eszközzel végeztem el. Sztereó kamerából egy Stereolabs ZED 2 kamerát használtam, LiDAR-ból pedig egy Ouster OS0-át és egy Livox Avia-t.

Elsődleges feladatként a sztereó kamera pontosságát elemeztem ki. Ennek érdekében összehasonlítottam a távolságértékeit mind a két LiDAR eredményeivel. Mivel a LiDAR alapú távolságmérés viszonylag pontos módszer, így ezzel értékelni lehet a sztereó kamera teljesítményét. Megvizsgáltam az eszközök közötti relatív hibát és kielemeztem a távolság- és objektumfüggő eltéréseket is.

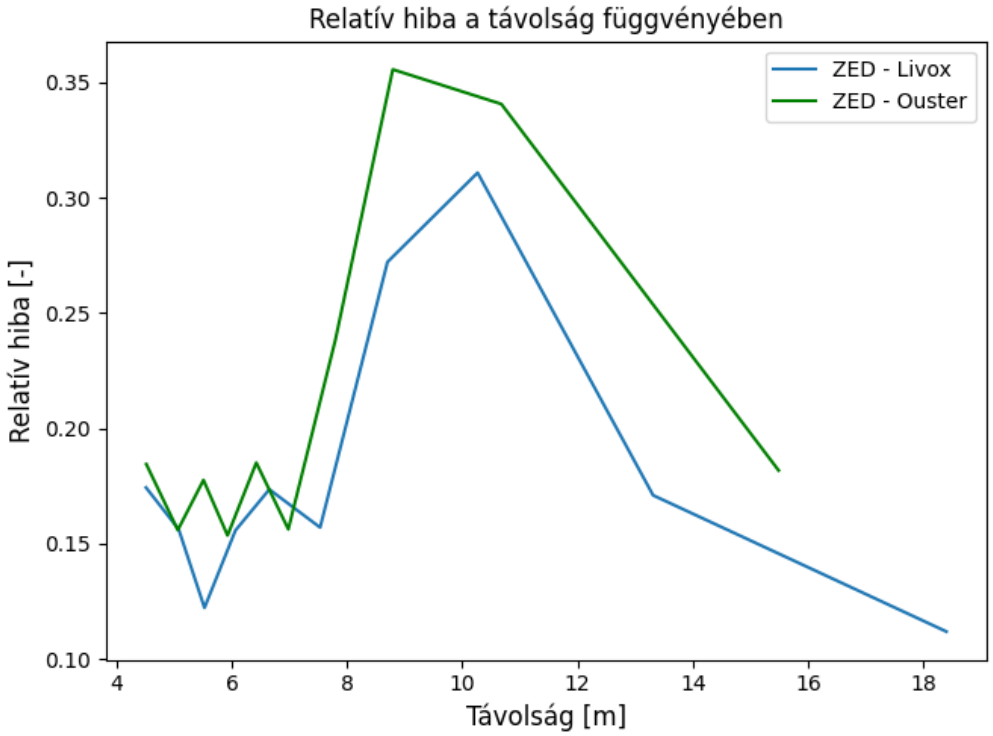
A hiba az eszközök működési módjainak eltérésein kívül adódhat a sztereó kamera karakterisztikájának pontatlanságából is. Megpróbáltam ezen is javítani és feltárni, hogy a távolságszámítás milyen módosításával kapható minimális hiba.

Másodlagos feladatként összehasonlítottam a két LiDAR mérésadatait is. Itt nem az egymástól való eltérést vizsgáltam, hiszen az jóval kisebb, mint a sztereó kamerával összehasonlítva. Az eltérések a két LiDAR között nem a távolságmérési módszerüknek, hanem a szkennelési metódusuknak köszönhető. Ahogy a 3.2 fejezetben is bemutattam az Ouster egymással párhuzamos vonalakban, 360 fokban körbefordulva alkotja meg a pontfelhőjét, míg a Livox nyolcas pályákat leírva szkenneli az előtte lévő teret. Ezen eltérés miatt a Livox pontfelhője a kép szélén kisebb sűrűséggel rendelkezik, mint a közepére beállított fókuszpontban. Érdemes megvizsgálni, hogy a pontfelhő sűrűségének ilyen változása jelentős hatással van-e a távolságmérés hibájára.

7.2. A sztereó kamera összehasonlítása LiDAR-ral

A ZED 2 sztereó kamera távolságmérését 648, illetve 642 detektált objektumon vetettem össze az Ouster és a Livox LiDAR eredményeivel. A vizsgált objektumok osztályra, pozícióra és távolságra nézve is reprezentatív mintát képeznek. A detektálásokra a sztereó kamera átlagos relatív hibája az Ouster-t etalonnak véve 21%-os, míg a Livox-ot etalonnak véve 18%-os. Érdemes azonban megvizsgálni ezeknek a hibáknak a jellegét is. A detekciók nagyjából kétharmadában a sztereó kamera pozitív irányba téved, tehát távolabbit mér, mint a valóság. Ezekben az esetekben a relatív hiba is jóval nagyobb, 26%-os, illetve 24%-os a negatív tévedés 12%-ához, illetve 8%-ához képest.

A hibákat érdemes a távolság függvényében is megvizsgálni. Természetes, hogy az abszolút hiba egyre nagyobb a távolság növekedésével, így itt is a relatív hibát érdemes nézni. A 7.1 ábrán látszik, hogy közeli objektum esetén a tévedés az átlagosnál kisebb, azonban egy ponton drasztikusan megnő. Ez az ugrás nagyjából 8 méteres távolság esetén lép fel. A grafikonok vége a relatív hiba mérséklődését mutatja, ez azonban a ZED 2 kamera 20 méteres maximális méréshatárának és az emiatt bekövetkező telítődésnek köszönhető. Releváns adatokat a grafikonok 10 méter alatti tartománya mutat.



7.1. ábra. A sztereó kamera relatív hibája a távolság függvényében a két LiDAR-hoz képest

Érdemes megvizsgálni a hibák mértékét objektum osztályok szerint is. Ezekben az Ouster-t etalonnak véve kisebb, míg a Livox-ot etalonnak véve jelentősebb eltérések mutatkoznak. Az eredményeket összefoglalva a 7.1 táblázat tartalmazza.

	ZED 2 - Ouster	ZED 2 - Livox
Detekciók száma	648	642
Átlagos relatív hiba	21%	18%
Pozitív relatív hibák száma	423	423
Negatív relatív hibák száma	225	219
Pozitív relatív hibák átlaga	26%	24%
Negatív relatív hibák átlaga	12%	8%
Fekvőrendőrök relatív hibáinak átlaga	22%	15%
Kátyúk relatív hibáinak átlaga	20%	21%
Csatornalefolyók relatív hibáinak átlaga	16%	13%
Csatornafedők relatív hibáinak átlaga	22%	22%
Bicikliút jelzések relatív hibáinak átlaga	27%	6%

7.1. táblázat. A sztereó kamera összehasonlítása a két LiDAR-ral

7.3. A karakterisztikus hibák kiküszöbölése

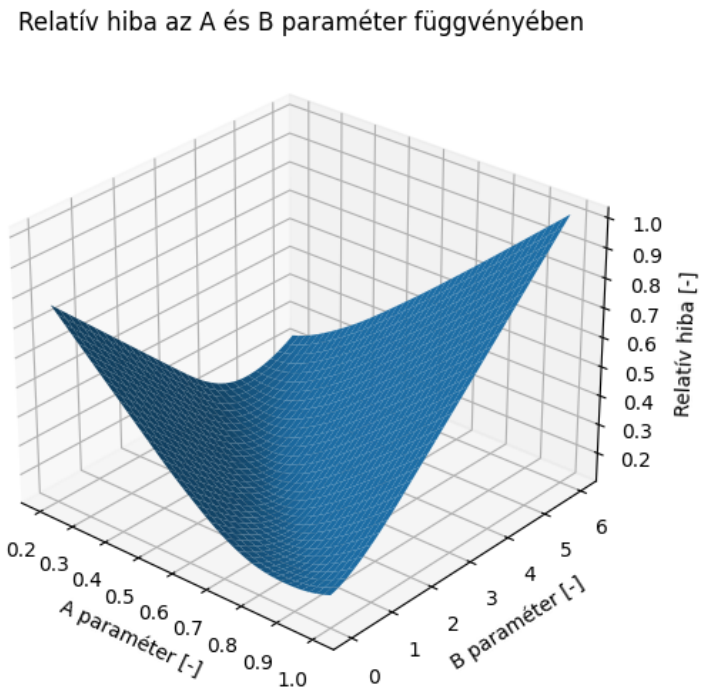
A 7.2 fejezetben láthattuk, hogy a sztereó kamera távolságmérése a legtöbb esetben pozitív irányban téved. Ebből sejthető, hogy a hibák nem csupán az eszközök különböző működéséből, hanem karakterisztikus pontatlanságokból is adódnak. Ezt okozhatja a mérés elején lévő kalibráció, valamint a kamera belső működése is. A sztereó kamera és az etalonnak vett LiDAR távolságértéke között lineáris kapcsolatot feltételezve két fajta hiba képzelhető el. Az egyik az eltolódást okozó ofszet hiba, míg a másik a meredekséget befolyásoló kalibrációs hiba [36]. Ezeket a hibákat figyelembe véve a valós távolságot az alábbi képlet alapján tudjuk kiszámítani, ahol az A paraméter a kalibrációs hiba (hibamentes állapotban az értéke 1), a B paraméter az ofszet hiba (hibamentes állapotban az értéke 0), az Y a valós távolság és az X a mért távolság.

$$Y = A * X + B$$

Az A és B paraméterek változtatásával az etalonhoz viszonyított relatív hiba is módosulni fog. Érdemes megállapítani, hogy a paraméterek milyen értékénél lesz minimális a hiba, hiszen ezeket beállítva a mérés pontosságán is jelentősen javítani tudunk.

Ahogy a 7.2 ábrán is látszik a relatív hiba a két paraméter függvényében széles határok között változik. Minimális értéket akkor kapunk, ha az A paraméter értéke 0,62, míg a B paraméter értéke 2,1. A valós távolság a sztereó kamera által kiszámolt érték függvényében a következő egyenlet alapján adható meg a legpontosabban.

$$Y = 0,62 * X + 2,1$$



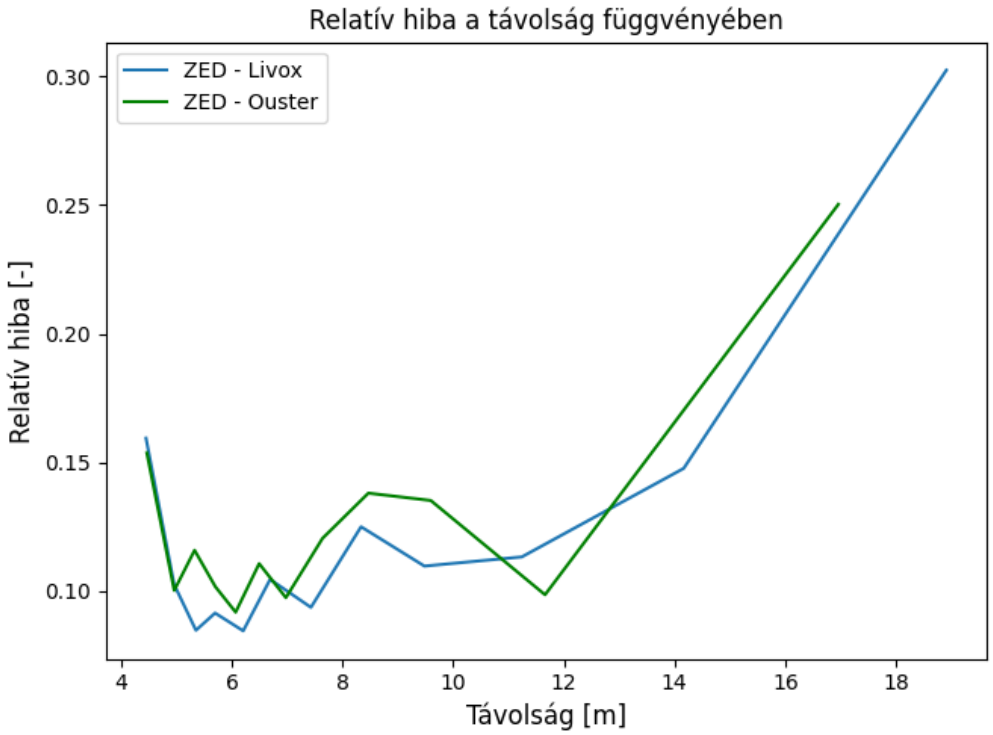
7.2. ábra. A relatív hiba változása a két paraméter függvényében

Ebben az esetben a sztereó kamera távolságmérésének relatív hibája mindkét LiDAR-ral összehasonlítva 12%. Az előző fejezetben bemutatott további értékeket a módosított karakterisztikájú mérésre a 7.2 táblázat tartalmazza.

	ZED 2 - Ouster	ZED 2 - Livox
Detekciók száma	648	642
Átlagos relatív hiba	12%	12%
Pozitív relatív hibák száma	305	294
Negatív relatív hibák száma	343	348
Pozitív relatív hibák átlaga	14%	13%
Negatív relatív hibák átlaga	11%	12%
Fekvőrendőrök relatív hibáinak átlaga	13%	14%
Kátyúk relatív hibáinak átlaga	13%	10%
Csatornalefolyók relatív hibáinak átlaga	10%	9%
Csatornafedők relatív hibáinak átlaga	13%	12%
Bicikliút jelzések relatív hibáinak átlaga	23%	16%

7.2. táblázat. A módosított karakterisztikájú sztereó kamera összehasonlítása a két LiDAR-ral

A távolságfüggő relatív hibát a 7.3 ábrán lehet látni. Megfigyelhetjük, hogy a karakterisztika paramétereinek helyes beállításával a relatív hiba ugrásának helye is kitolódott 15 méterre.



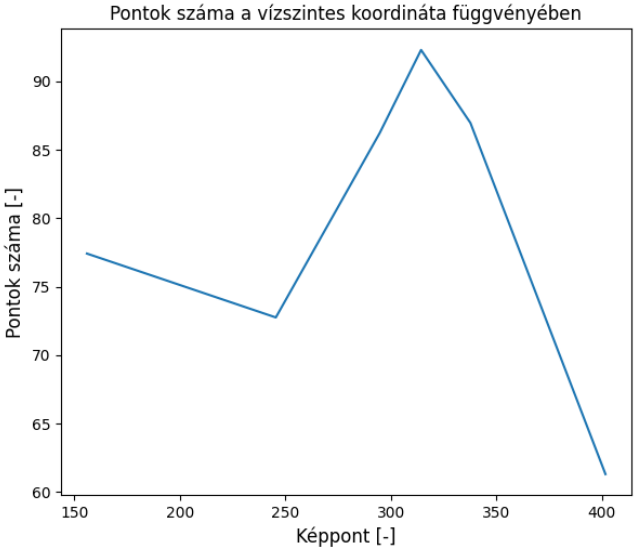
7.3. ábra. A karakterisztika korrigálásának hatása a sztereó kamera és a két LiDAR közötti relatív hibára a távolság függvényében

7.4. Az Ouster és a Livox LiDAR összehasonlítása

Az Ouster és a Livox LiDAR mérési elve ugyanúgy lézeralapú, azonban a környezet feltérképezésére használt szkennelési mintájuk különbözik. Míg az Ouster pontfelhője azonos távolságban homogén sűrűségű, addig a Livox a nyolcasokat leíró szkennelési minta miatt a fókuszponttól távolabb kevesebb pontot tud felvenni. Érdemes megvizsgálni, hogy ez a különbség okoz-e jelentős pozíciófüggő relatív hibát a két eszköz között.

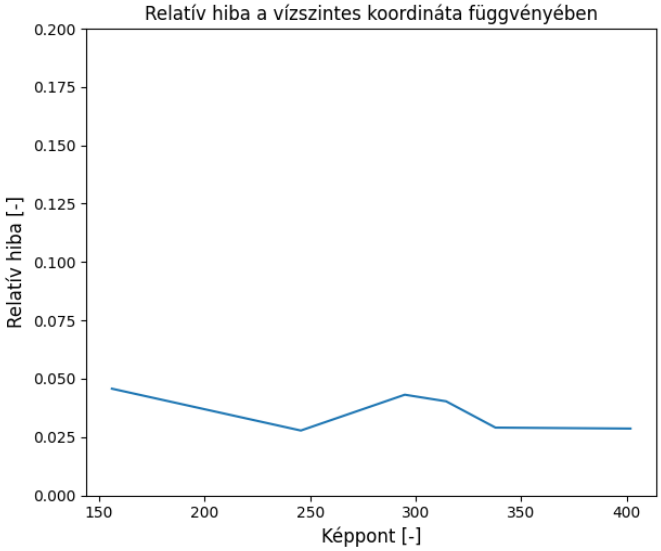
Az objektumok útfelületen való elhelyezkedésük miatt a kép függőleges tengelyén minimális az eltérés a pozíciójukban. A vízszintes tengelyen ennél jóval nagyobb tartományt fednek le, így ezen érdemes megvizsgálni a pozíciófüggő eltéréseket. Erre a mérésre kivettem a fekvőrendőrök osztályt az adatok közül. Ezek az objektumok nagy és széles kialakításuk miatt mindig a kép középső részén helyezkednek el, azonban viszonylag távolabbról történő detektálásuk miatt jóval nagyobb hibával rendelkeznek. Ez torzítja az eredményeket, így a kivételük szükséges a helyes kiértékelés elvégzéséhez.

A többi osztály objektumai pozícióra és távolságra nézve reprezentatív eloszlásúak. Ahogy a 7.4 ábra is mutatja, a Livox pontfelhőjének sűrűsége, vagyis az egy befoglaló téglalapba eső pontok száma a vízszintes tengely két szélén lecsökken.



7.4. ábra. A Livox pontfelhő pontjainak száma a kép vízszintes tengelye mentén

A 7.5 ábrán a relatív hibák értékét láthatjuk a vízszintes tengely mentén. Megfigyelhetjük, hogy bár a pontok száma a kép szélén lecsökken, ez a relatív hiba mértékére nincs jelentős hatással, így pozíciófüggő hiba nincs.



7.5. ábra. A Livox relatív hibája az Ouster-hez képest a kép vízszintes tengelye mentén

7.5. Konklúzió

A sztereó kamerát két referenciával összehasonlítva hasonló értékeket és karakterisztikákat kaptam. Ez alapján kijelenthető, hogy az eredmények valóban a sztereó kamerára és nem a kiválasztott etalonra jellemzőek.

Az eredmények alapján látható, hogy a ZED 2 sztereó kamera közeli objektum távolságmérése esetén viszonylag pontos tud lenni. Ezt a pontosságát egy adott határ után jelentős mértékben elveszíti. Mind a pontosság mértéke, mind ennek a határnak a kitolása lehetséges a karakterisztika megfelelő beállításával.

Természetesen a sztereó látás alapú távolságmérés nem tud annyira pontos lenni, mint a lézeralapú. Amennyiben az itt bemutatott eredmények nem kielégítőek egy autonóm járműveket tervező vállalat számára, akkor a távolság mérését érdemesebb a a kamerakép és egy LiDAR fúziójával végezni. Ez már csak azért is előnyös megoldás lehet, mert ahogy a 2.1 fejezetben is be lett mutatva, sok önvezető autó már egyébként is rendelkezik ilyen típusú szenzorral.

A Livox LiDAR vizsgálatából kijelenthető, hogy a pontfelhő sűrűségének változása nincs jelentős hatással a relatív hiba változására. Ennek az oka elsősorban az objektumok távolságának definíciójában rejlik. Ahogy a 6.3 fejezetben bemutattam, a legmegfelelőbb megoldás a programok térbeli szinkronizálásához a medián módszer használata. Ennek a hatása itt is megfigyelhető, hiszen ennek a módszernek a pontosságát nem befolyásolja a befoglaló téglalapon belül található pontok alacsony száma.

Ezek alapján megállapítható, hogy ha egy autonóm járműveket gyártó vállalat a LiDAR-ral történő távolságmérést preferálja, akkor a Livox Avia-t is használhatja az Ouster OS0 helyett, pozíciófüggő hiba fellépése nélkül. A két eszköz közötti jelentős árkülönbség miatt pedig ez a döntés jelentősen csökkentheti az autonóm jármű gyártásának költségeit.

8. fejezet

Összefoglalás

A manapság egyre inkább elterjedő autonóm járművek egyik legfontosabb feladata a környezetükben található objektumok észlelése. A kátyúk, fekvőrendőrök és egyéb akadályok detektálása kulcsfontosságú egy önvezető autó biztonságos működése szempontjából.

Ezen objektumok detektálására felügyelt tanulással neurális hálózatot hoztam létre. Ehhez nagyszámú adat gyűjtésére, azok annotálására és a hálózat tanítására volt szükség. A kiértékelésnél látható, hogy az általam létrehozott modell alkalmas kátyúk, fekvőrendőrök, csatornafedők, csatornalefolyók és bicikliút jelzések detektálására.

Az előre definiált objektumok észlelésén túl azonban fontos azok pozícióinak is a meghatározása. Ez egy autonóm jármű esetén elengedhetetlen a megfelelő irányítás kialakításához. Erre a feladatra két módszert, a sztereó látást és a lézeres távmérést teszteltem.

Ehhez a két mérőprogramot térben és időben szinkronizálnom kellett. A térbeli szinkronizáció során összehasonlítottam és kielemeztem az objektumok detektálása esetén lehetséges távolságdefiníciókat is. Az eredmények kiértékelésére külön programot hoztam létre, ami előre meghatározott, de dinamikusan változtatható szempontok alapján hasonlítja össze az egyes eszközök távolságméréseinek értékeit.

A sztereó kamera és a két LiDAR összehasonlítása során látható volt, hogy előbbi eszköz mérsékelt pontossággal rendelkezik, ami egy bizonyos távolsági határ után jelentősen csökken. A dolgozatomban kielemeztem a sztereó kamera karakterisztikáját is és annak kompenzálásával jelentős javulást értem el az eredményekben. A két LiDAR szkennelési metódusainak elemzése során megállapítottam, hogy a medián alapú távolságmérés miatt a pontfelhő sűrűségének csökkenése nem jár a pontosság változásával.

A dolgozatom és a benne bemutatott programok, módszerek és mérések alkalmasak egy autonóm járműveket tervező vállalat számára is. Az általam fejlesztett neurális hálózat képes bizonyos úthibák és akadályok észlelésére, az elvégzett kutatás pedig alapot adhat egy konkrét termék szenzorainak és mérési módszereinek felülvizsgálatára is.

Budapest, 2023. november 2.

Szabó Máté András

Irodalomjegyzék

[1] Pan Zhao és tsai. "Design of a control system for an autonomous vehicle based on adaptive-pid". *International Journal of Advanced Robotic Systems* 9.2 (2012), 44. old.

[2] Asif Faisal és tsai. "Understanding autonomous vehicles". *Journal of transport and land use* 12.1 (2019), 45–72. old.

[3] Jelena Kocić, Nenad Jovičić és Vujo Drndarević. "Sensors and sensor fusion in autonomous vehicles". *2018 26th Telecommunications Forum (TELFOR)*. IEEE. 2018, 420–425. old.

[4] Wang Zhiqiang és Liu Jun. "A review of object detection based on convolutional neural network". *2017 36th Chinese control conference (CCC)*. IEEE. 2017, 11104–11109. old.

[5] Young-Mok Kim és tsai. "Review of recent automated pothole-detection methods". *Applied Sciences* 12.11 (2022), 5320. old.

[6] Ronghua Du és tsai. "Abnormal road surface recognition based on smartphone acceleration sensor". *Sensors* 20.2 (2020), 451. old.

[7] Muhammad Uzair Ul Haq és tsai. "Stereo-based 3D reconstruction of potholes by a hybrid, dense matching scheme". *IEEE Sensors Journal* 19.10 (2019), 3807–3817. old.

[8] Amita Dhiman és Reinhard Klette. "Pothole detection using computer vision and learning". *IEEE Transactions on Intelligent Transportation Systems* 21.8 (2019), 3536–3550. old.

[9] Neha Gupta és tsai. "Artificial neural network". *Network and Complex Systems* 3.1 (2013), 24–28. old.

[10] Pulkit Sharma. *Image Classification vs. Object Detection vs. Image Segmentation*. `https://medium.com/`. 2023.

[11] Zewen Li és tsai. "A survey of convolutional neural networks: analysis, applications, and prospects". *IEEE transactions on neural networks and learning systems* (2021).

[12] Joseph Redmon és tsai. "You only look once: Unified, real-time object detection". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 779–788. old.

[13] Alexey Bochkovskiy, Chien-Yao Wang és Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". *arXiv preprint arXiv:2004.10934* (2020).

[14] Deval Shah. *Mean Average Precision (mAP) Explained: Everything You Need to Know*. `https://www.v7labs.com/`. 2023.

[15] Stephen T Barnard és Martin A Fischler. "Computational stereo". *ACM Computing Surveys (CSUR)* 14.4 (1982), 553–572. old.

[16] DrMax. *Computer Vision: Stereo 3D Vision*. `https://www.baeldung.com/cs/`. 2023.

[17] Clearview. *Stereo Vision for 3D Machine Vision Applications*. `https://www.clearview-imaging.com/en/`. 2023.

[18] Stereolabs. *Stereolabs ZED 2*. https://www.stereolabs.com/zed-2/. 2023.

[19] Morgan Quigley és tsai. "ROS: an open-source Robot Operating System". *ICRA workshop on open source software*. 3. köt. 3.2. Kobe, Japan. 2009, 5. old.

[20] Larry Li és tsai. "Time-of-flight camera—An introduction". *Technical white paper* SLOA190B (2014).

[21] Wikipédia. *Time-of-flight camera*. https://www.wikipedia.org/. 2023.

[22] Joe Liadsky. "Introduction to LIDAR". *NPS Lidar Workshop*. 2007, 1–41. old.

[23] Synopsys. *What is LiDAR?* https://www.synopsys.com/. 2023.

[24] You Li és Javier Ibanez-Guzman. "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems". *IEEE Signal Processing Magazine* 37.4 (2020), 50–61. old.

[25] Ouster. *Ouster OS0*. https://ouster.com/products/hardware/os0-lidar-sensor. 2023.

[26] Livox. *Livox Avia*. https://www.livoxtech.com/avia. 2023.

[27] Livox. *Livox Introduces High Performance, Low Cost, Mass Market Lidar Sensors For L3/L4 Autonomous Driving Applications*. https://www.livoxtech.com/. 2023.

[28] Auvidea. "X221 for Nvidia Jetson AGX Xavier". (2023). URL: https://auvidea.eu/product/70881/.

[29] G. Bradski. "The OpenCV Library". *Dr. Dobb's Journal of Software Tools* (2000).

[30] Marko Bjelonic. *YOLO ROS: Real-Time Object Detection for ROS*. https://github.com/leggedrobotics/darknet_ros. 2016–2018.

[31] Bianka Tallita Passos és tsai. "Cracks and potholes in road images". *Mendeley Data* 4 (2020), 2020. old.

[32] Oshada Jayasinghe és tsai. "CeyMo: See More on Roads - A Novel Benchmark Dataset for Road Marking Detection". *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, 3104–3113. old.

[33] J. Cartucho, R. Ventura és M. Veloso. "Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots". *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, 2336–2341. old.

[34] Joseph Redmon és Ali Farhadi. "YOLOv3: An Incremental Improvement". *arXiv* (2018).

[35] Techzizou. *Train a custom YOLOv4 object detector on Linux*. https://medium.com/. 2023.

[36] Gerzson Miklós. *Mérési hibák*. Pannon Egyetem, Villamosmérnöki és Információs Rendszerek Tanszék. 2020.