

TDK dolgozat

Szerző: Dobay Tamás

Széchenyi István Egyetem
Gépészmérnöki, Informatikai és Villamosmérnöki Kar
Informatika Tanszék

Neurális hálózat alapú sávvonal detekció klaszterezése

Dobay Tamás
(I. éves villamosmérnöki MSc hallgató)

Konzulens:
Hollósi János
(egyetemi tanársegéd)

Tartalomjegyzék

1. Bevezetés.....	2
2. Mesterséges neurális hálózatok	4
2.1. Mesterséges neuronhálózatok – ANN	4
2.2. Konvolúciós neurális hálózatok – CNN.....	6
2.2.1. Konvolúciós réteg	6
2.2.2. Összevonó réteg	8
2.2.3. Teljesen összekapcsolt réteg.....	9
2.3. Mesterséges neurális hálózatok tanulása.....	9
2.3.1. Hiba visszaterjesztés	10
2.4. Felhasznált architektúra.....	12
2.4.1. YOLOP (You Only Look Once for Panoptic).....	12
3. Kamera-LiDAR kalibráció	14
3.1. Kamera kalibrációja	14
3.2. Kamera-LiDAR kalibrációja	15
4. A kiértékeléshez használt adatkészlet.....	18
5. Felhasznált klaszterezési módszerek	20
5.1. DBSCAN (Density - Based Spatial Clustering of Applications)	20
5.2. OPTICS (Ordering Points To Identify the Clustering Structure)	21
6. Klaszterezési mérőszámok	22
6.1. Homogenitás.....	22
6.2. Teljesség	22
6.3. V-mérték.....	22
6.4. Korrigált Rand pontszám	23
7. A kapott mérési eredmények bemutatása	24
8. Összefoglalás.....	33
9. Köszönetnyilvánítás	34
10. Irodalomjegyzék.....	35

Jelölések

$w_i^{(L)}$	az L rétegben i-edik neuron kapcsolat súlyértéke
$a_i^{(L)}$	az L rétegben az i-edik neuron kimeneti értéke
b	eltolássúly
ϕ	aktivációs függvény
s	konvolúciós kernel lépésköze
C	veszteség függvény
t	mérési alap értéke
K	konvolúciós kernel mátrixa

1. Bevezetés

A 20. század közepétől kezdve hatalmas fejlődésen mentek keresztül a gépi látás területén alkalmazott mesterséges neurális hálózatok. Az első konvolúciós neurális hálózatoktól, amelyek még csak egy képen szereplő objektumnak az éleit tudták meghatározni, mára már képesek pixelpontosan külön választani a különböző osztályokba tartozó objektumokat, és egyben lokalizálni is tudják azokat. Mivel már ilyen komplexitású problémát is megbírnak oldani, ezért különböző autonóm működésű berendezések és járművek szerves részeit képezik. Egy hasonló működéssel bíró képfeldolgozó szoftver pontosabb és részletesebb információt biztosít számukra, ezért bonyolultabb környezetekben is (mint például a közúti forgalom) pontosabb irányítást tudnak végrehajtani. Oldalirányú útvonal tervező algoritmusok számára egy sávsegmentáló neurális hálózat kimenetével megbízható bemenetet lehet szolgáltatni. Többször előfordul viszont, hogy a sávsegmentáló hálózatok kimenete nem szolgáltat adatot arról, hogy a szegmentált sávok a járműhöz viszonyítva melyik oldalt helyezkednek el, pedig ez egy elengedhetetlen információ az útvonal tervező megfelelő működéséhez.

Dolgozatom célja megvizsgálni, hogy a YOLOP [4] többfeladatos neurális hálózat sávval szegmentáló kimenetét fúzionálva LiDAR szenzorból érkező pontfelhővel mennyire pontosan lehet klaszterezni jobb, illetve baloldali sávok szerint. Ehhez több sűrűség alapú klaszterezési módszert fogok felhasználni, amiknek a predikciós pontosságát különböző mérőszámok alapján fogok kiértékelni.

Méréseimet a Járműipari Kutatóközpont Nissan Leaf járművével ROS [7] (Robot Operating System) keretrendszerbe felvett rosbagek fogják képezni. Egy rosbag tartalmazza a szenzorok mérési adatait időbélyegekkel ellátva, többször visszajátszható, és a ROS RViz nevezetű grafikus interfészen keresztül megjeleníthetők az egyes szenzorok kimenetei.

Az adatkészletemet az előbb említett mérésekből fogom előállítani, ami több általam címkézett pontfelhő lesz aszerint, hogy az adott pont zaj, bal oldali sávval vagy jobb oldali sávval reprezentál.

A YOLOP modellnek a gyakorlati felhasználását PyTorchon [13] keresztül fogom megvalósítani, ami a Python nyelvhez írt mély tanulási keretrendszer. A klaszterezéseket a Scikit-learn [14] Python könyvtár segítségével fogom implementálni, illetve kiértékelni, amit beépített függvényeinek köszönhetően egyszerűbben tudok majd végrehajtani az elkészített kiértékelő adatkészleten.

Dolgozatom eredményeként a kiértékelt mérési adatok nagyobb belátást adhatnak arra, hogy klaszterezési algoritmusokkal mennyire pontosan lehet szétválasztani a sávvonalakat, illetve az egyes módszerek mennyire pontosak egymáshoz viszonyítva.

2. Mesterséges neurális hálózatok

A mesterséges neurális hálókat az agyban található neuronok és azok összeköttetése ihlette. Ebben a fejezetben először a mesterséges neuronok és az előre-csatolt neuronhálózatok egy fajtáját, a többrétegű neuronhálózatok általános felépítését fogom bemutatni. Ezek után egy speciális esetüket, a konvolúciós neurális hálózatokat, végezetül pedig a feladatom szempontjából lényeges CNN (Convolutional Neural Network) architektúrákat és adatkészleteket.

2.1. Mesterséges neuronhálózatok – ANN

A mesterséges neuronhálózatok legfőbb elemei a mesterséges neuronok (perceptrons), másnéven a csomópontok (nodes) [1]. Ezek a neuronok az előre-csatolt többrétegű neuronhálózatoknál mind kapcsolódnak egymáshoz, és rétegekbe vannak rendezve. Két neuron közötti kapcsolat egy w súlyértékkel (weight) van ellátva. Ezzel a súlyértékkel van megszorozva az előző rétegből kapcsolt neuron kimenete, így tudja a háló mérlegelni az előjel és amplitúdó függvényében, hogy az a neuron kapcsolat az adott probléma megoldásában mennyire fontos. A csomópontokhoz egy b eltolássúly (bias) is hozzá van rendelve, az (1) -es képletből következően egy küszöbértéket képez a neuronok aktivitásának. Általánosságban rétegenként ugyanazzal az eltolássúllyal számolnak a neuronok. Az összes kapcsolaton keresztül érkező szorzatok összege bemenetét képzik az adott neuronhoz tartozó, úgynevezett ϕ aktivációs függvénynek, ami egy nemlineáris leképezést adja a bemenetnek az aktuális neuron kimenetén, ezzel biztosítva a hálózat nem linearitását. Ilyen függvény például a logisztikai szigmoid (2), vagy a ReLU (3). Egy többrétegű neuronhálózatban egy adott neuron kimeneti értékét az (1) -es képlet adja meg, grafikus szemléltetése a 2. ábrán látható.

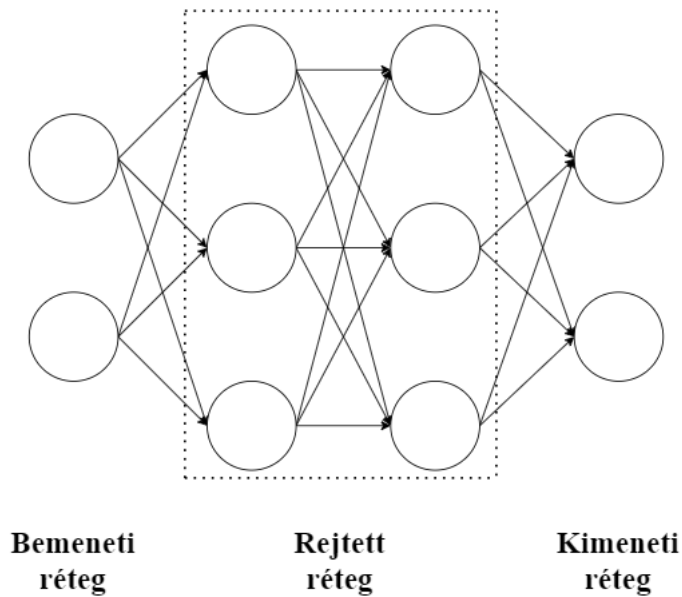
$$a_i^{(L)} = \phi \left(\sum_{i=1}^n w_i^{(L)} \cdot a_i^{(L-1)} + b^{(L)} \right) \quad (1)$$

ahol $a_i^{(L)}$ az adott neuron kimenete.

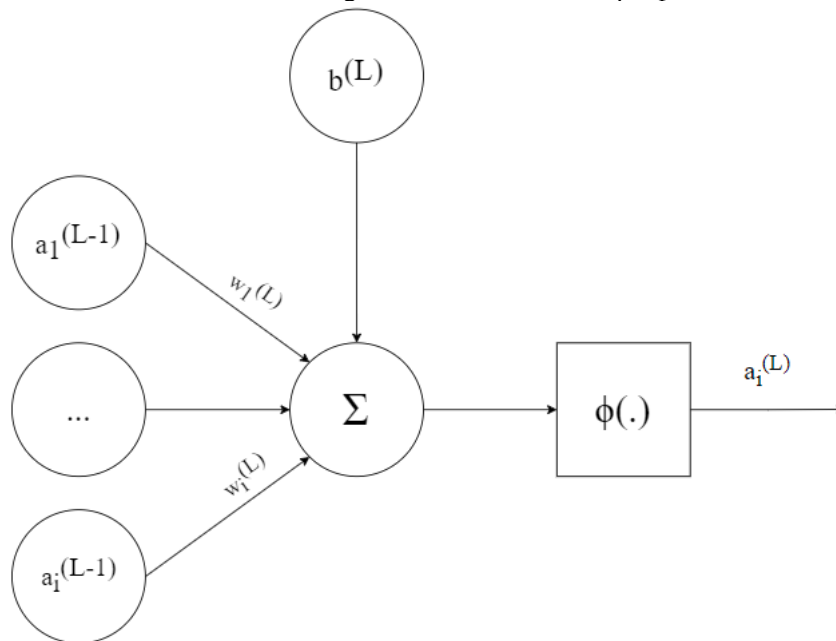
$$\phi(x) = \frac{e^x}{e^x + 1} \quad (2)$$

$$\phi(x) = \max(0, x) \quad (3)$$

A többrétegű neuronhálózatok esetében három réteg különböztethető meg: a bemeneti réteg, a rejtett réteg és a kimeneti réteg, ahogy az az 1. ábrán is látható.



1. ábra : Mesterséges neuronhálózat elvi felépítése



2. ábra : Egy mesterséges neuron művelet gráfrepresentációja

A korábbiakban leírtak alapján elmondható, hogy a rendszer, ami keletkezik, sok nem lineáris függvény szuperpozíciója [2], tehát ezek a hálózatok a bemenet nem lineáris leképezését adják a kimeneten. Ez a tulajdonságuk teszi lehetővé a mesterséges neuronhálók alkalmazását olyan erősen nem lineáris problémák megoldásához, mint egy adott képen való objektumok osztályozása.

2.2. Konvolúciós neurális hálózatok – CNN

A mesterséges neurális hálózatok egy speciális részét képezik a konvolúciós neurális hálózatok, amelyek manapság a gépi látás szerves részét képezik. Felépítésük annyiban különbözik az előbbieken bemutatott mesterséges neuronhálózat modelljétől, hogy nincsen mindegyik neuron összeköttetésben a konvolúciós rétegek között, tehát nem teljesen összekapcsolt hálózatok, és a súlyozások konvolúciós rétegenként megegyeznek. Ebben az alfejezetben fogom bemutatni a felépítésüket rejtett rétegük szerint, amit általánosságban konvolúciós rétegek, teljesen összekapcsolt rétegek és összevonó rétegek alkotnak.

2.2.1. Konvolúciós réteg

A konvolúciós neurális hálózatok legérzékenyebb rétege a konvolúciós réteg [3], neve a konvolúció műveletből származik. Segítségével a háló a bemeneti képről olyan hasznos tulajdonságokat tud leképezni (egy adott objektum éleitől kezdve teljes alakzatokig) amelyek felhasználásával képes pontos predikciót adni a kimenetén.

Egy ilyen réteg általában jellemezhető egy D csatorna számmal, egy K konvolúciós kernellel, aminek térbeli kiterjedését $n \times m$ alakban szokás megadni, egy s lépésközzel és egy p kitöltési tényezővel. Szerepeikre később külön-külön fogok kitérni. Bemenete általában vagy maga a bemeneti kép, vagy az előző konvolúciós réteg kimenete. Ezek adattípusa egy mátrix $H \times W \times D$ térbeli kiterjedéssel, ahol $H \times W$ a kép pixelszerinti magassága és szélessége, és D a csatornák száma (RGB kép esetén $D = 3$).

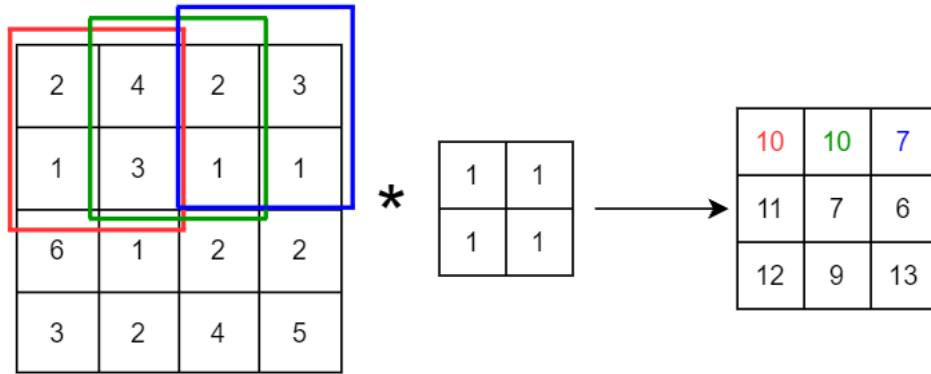
A konvolúció művelet bemutatásához a 3. ábra ad segítséget. Kimenetének a matematikai leírása háromdimenziós bemenet esetén a (4) -es egyenlet, ahol az egyszerűség kedvéért eltolássúllyal nem számolok, kitöltés nincs, $s = 1$.

$$y_{i^{(L+1)},j^{(L+1)},d} = \sum_{i=0}^H \sum_{j=0}^W \sum_{d^{(L)}=0}^{D^{(L)}} k_{i,j,d^{(L)},d} \cdot x_{i^{(L+1)}+i,j^{(L+1)}+j,d^{(L)}} \quad (4)$$

$$\text{ahol } y_{i^{(L+1)},j^{(L+1)},d} = a^{(L+1)}, \quad x_{i^{(L+1)}+i,j^{(L+1)}+j,d^{(L)}} = a^{(L)}, \quad k_{i,j,d^{(L)},d} = K$$

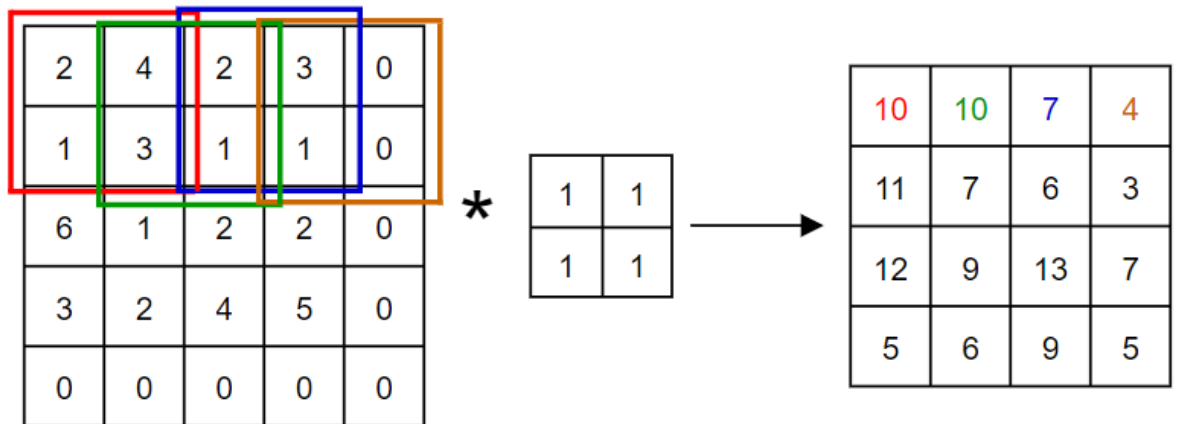
Veszek egy 2×2 -es térbeli kiterjedésű konvolúciós kernelt, aminek mindegyik eleme 1. Veszek egy $4 \times 4 \times 1$ -es bemeneti képet, amelynek elemei egy-egy pixelértéknek feleltethetők meg. Ha a konvolúciós kernelt a bemeneti kép felé helyezem, és minden azonos helyen elhelyezkedő kernel értéket és képmátrix értéket összeszorzok, majd ezeket összegezem, akkor végeredményként egy egész számot kapok. A 3. ábra alapján a kernel első pozíciójában ezt az alábbi módon kapom: $2 \cdot 1 + 4 \cdot 1 + 1 \cdot 1 + 3 \cdot 1 = 10$. Majd ezt egy pixelnyi lépésközzel ($s = 1$) végrehajtom a vízszintes és függőleges tengelyek mentén, amíg el nem érek a képünk

bal felső sarkától a jobb alsó sarkáig. Így kapom meg a művelet végeredményét, ami szintén egy mátrix.



3. ábra : A konvolúció illusztrációja

Észrevehető az ábra alapján, hogy a kimenet mérete a bemenethez képest változott, most már $(\frac{W-n}{s} + 1)$ sorral és $(\frac{H-m}{s} + 1)$ oszloppal rendelkezik, az én esetemben $(\frac{4-2}{1} + 1)$ sorral és $(\frac{4-2}{1} + 1)$ oszloppal, vagyis egy 3×3 -as kimenetet kapok a 4×4 -es bemenetből. Vannak olyan esetek, amikor azonos méretű kimeneti képre van szükség mint a bemeneti kép. Ekkor szokás használni kitöltést (p). A konvolúció előtt $\lfloor \frac{n-1}{2} \rfloor$ sort illesztnek a bemenet felső széléhez, $\lfloor \frac{n}{2} \rfloor$ sort az alsó széléhez, $\lfloor \frac{m-1}{2} \rfloor$ oszlopot a bal oldali széléhez és $\lfloor \frac{m}{2} \rfloor$ oszlopot a jobb oldali széléhez, ahogy az a 4. ábrán is látható. Elemeinek értéke általában 0, de természetesen lehet más értékeket is használni. Ennek hatására a kimeneti dimenzió az alábbi módon változik: $(\frac{H+p-n}{s} + 1) \times (\frac{W+p-m}{s} + 1)$ kiterjedésű lesz, ahol p értékét a fentebb leírtak alapján könnyen meg lehet határozni. Behelyettesítés után megkapom, hogy a kimenetünk a 4. ábrán egy 4×4 mátrix lesz, tehát nem történik méretbeli változás.



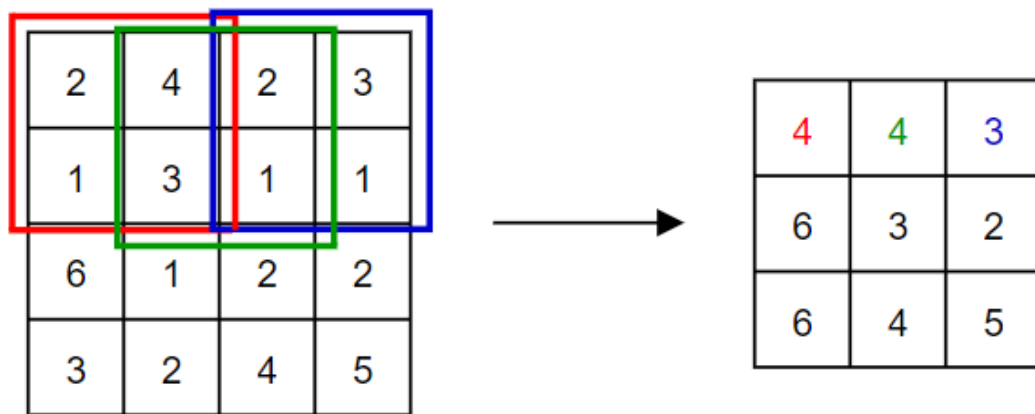
4. ábra : Konvolúció illusztrációja kitöltés alkalmazásával

Másik hasznos tulajdonsága a kitöltésnek, hogy a normál konvolúció során kevésbé érintett pixelek nagyobb szerepet kapnak. Amíg kitöltés nélkül a 3. ábrán a jobb alsó sarokban elhelyezkedő 5-ös értékű pixelt csak egyszer érinti a konvolúciós kernel, addig kitöltéssel már 4-szer. Így a kép szélein elhelyezkedő pixelértékek hasonló szerepet játszanak a tulajdonságtérkép leképezésében, mint a kép közepén elhelyezkedő pixelértékek.

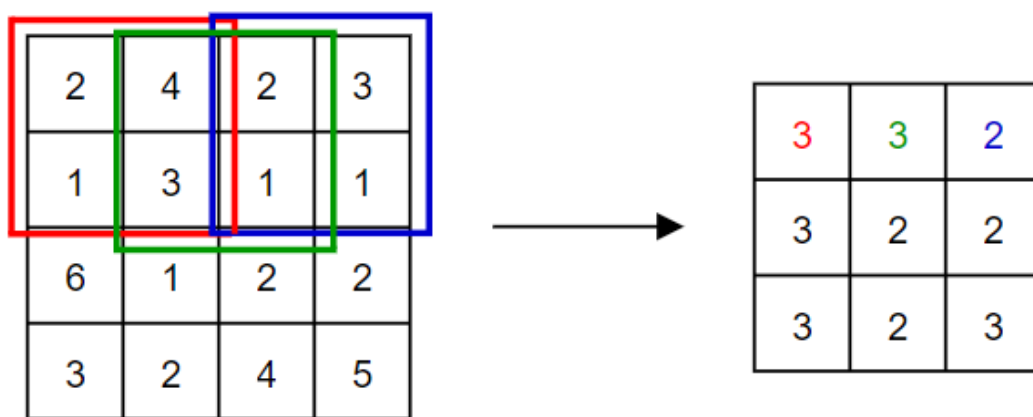
Egy konvolúciós réteg súlyai maguk a konvolúciós kernelek értékei. A rétegeken belül csatornánként azonos konvolúciós kernelt használ a hálózat a műveletekhez. Kimenetének csatornaszámát a rétegen belül felhasznált konvolúciós kernelek száma határozza meg. A rétegek kimeneteit tulajdonságtérképeknek szokás nevezni egy hálózaton belül. Minél több ilyen réteg van felhasználva egy modellben, annál mélyebb információkat lehet szerezni az adott bemeneti képről. A mélyebb rétegek jobban képesek „megtanulni” a tanulási fázis során, hogy csak bizonyos komplex mintákra legyenek aktívak. Egyre mélyebb rétegekben, ezek a minták már szemantikailag értelmes objektumrészek, vagy teljes objektumtípusok. Általánosságban elmondható, hogy minél több konvolúciós réteg van felhasználva egy CNN-ben, annál pontosabb predikciók érhetőek el, mondjuk egy adott objektum osztályozásában, viszont figyelembe kell venni, hogy egyre nagyobb számítási igénye lesz a hálózatnak a sok konvolúciós művelet miatt.

2.2.2. Összevonó réteg

Az összevonó rétegek [3] egy konvolúciós neurális hálózatban általában a konvolúciós rétegek között szoktak elhelyezkedni. Szerepe az, hogy redukálja a tulajdonságtérképek méreteit, ezzel csökken a hálózat paraméter száma, amivel a számítási igények is mérséklődnek. Egy ilyen réteg nulla tanulható paraméterszámmal rendelkezik, tehát nem vesznek részt a tanulási fázisban. Mint a konvolúciós réteg, az összevonó réteg is rendelkezik egy kernellel, amelynek mérete meghatározza, hogy mekkora régióként összegzi a bemenetként kapott tulajdonságtérkép értékeit. Ezen kívül jellemezhető egy lépésközzel, ami meghatározza mekkora a pixel szerinti léptéke a kernelnek, csak úgy, mint a konvolúciós rétegnél. Leggyakrabban maximum összevonó és átlagolt összevonó rétegről lehet beszélni. Az előbbi működése az 5. ábrán, az utóbbi a 6. ábrán látható.



5. ábra : A maximum összevonás illusztrációja



6. ábra : Az átlagolt összevonás illusztrációja

A réteg kimenetének mérete hasonlóképpen számolható ki, mint a konvolúciós réteg kimenete.

2.2.3. Teljesen összekapcsolt réteg

Teljesen összekapcsolt rétegnek [3] hívnak minden olyan réteget egy neurális hálózatban, amiről elmondható, hogy bármely elemének a kiszámításához szükség van az előző réteg összes elemére. Egy konvolúciós neurális hálózatban azért van szükség teljesen összekapcsolt rétegekre (általában a hálózat végén), mert hozzá képest az utolsó réteg a bemeneti kép elosztott reprezentációit tartalmazza, és az összes kinyert tulajdonságot érdemes felhasználni a predikcióhoz, vagyis a kimeneti réteghez.

2.3. Mesterséges neurális hálózatok tanulása

A mesterséges neurális hálózatok tanulása során a cél az, hogy a neuronok kapcsolatainak súlyértékei (és a rétegek súlyeltolásait) olyan módon legyenek módosítva minden végigfuttatott tanító adat/adatköteg után, hogy a hálózat veszteségfüggvényének (C) értéke minimális legyen.

A veszteségfüggvény általában az elvárt kimeneti érték és a kapott kimeneti érték különbségének a leképzése. A súlyozások frissítését az úgynevezett hiba visszaterjesztési algoritmussal szokás elvégezni. Még mielőtt erre részletesebben kitérnék, fontos először magáról a tanító adatkészletről és a tanítás menetéről néhány szót ejteni.

A tanításhoz használt adatkészletet három részre lehet bontani: egy tanító készletre, egy érvényesítő készletre és egy teszt készletre. Ezek felügyelt tanulás esetén mind címkézett adatok, tehát tartalmazzák az elvárt kimeneti értéket, amit később össze lehet hasonlítani a hálózat aktuális kimenetével. A tanítást korszakokra lehet lebontani, egy korszak akkor tekinthető befejezettnek, ha a rendelkezésre álló összes tanítási minta átment a hálózaton. A korszakok iterációkból állnak, aminek száma mintánként való tanítás esetén megegyezik a tanító készlet adatainak darabszámával. Általában nem mintákkal, hanem kötegekkel szokás tanítani a számítási igények csökkentésének érdekében. Ez azt jelenti, hogy egy minta helyett egyszerre több minta fut keresztül a hálón egy iteráció során, így a tanulható paramétereket (súlyokat) elég kötegenként frissíteni. Az érvényesítő készletet általában minden korszak végén szokás végig futtatni a hálón. Mivel az érvényesítő mintákkal a hálózat az adott korszak során nem találkozott, ezért a tanításból és az érvényesítésből kapott veszteség függvények összehasonlításával meg lehet határozni, hogy a modell mennyire volt képes elsajátítani egy általánosított megoldást az adott problémára. Ha ezek az értékek nagyban eltérnek egymástól (a tanítás során kapott veszteségfüggvény értéke jóval kisebb, mint az érvényesítés során kapotté), az azt jelenti, hogy a modell túlilleszkedik a tanító készletre, és nem volt képes általánosítani a problémát. Ezen a tanításhoz használt hiperparaméterek módosításával (tanulási ráta (η), köteg méret...stb.) lehet változtatni. Egy tanítást addig fut általában, amíg a validációs készletre adott veszteségfüggvény értéke el nem kezd nőni. Fontos megjegyezni, hogy az érvényesítő készlet nem vesz részt a hálózat paramétereinek frissítésében. A teszt készlettel lehet kiértékelni tanítás végén a modell predikciós pontosságát.

2.3.1. Hiba visszaterjesztés

Hiba visszaterjesztésnek [1] hívják azt a módszert, amivel a veszteség függvény értéke rétegenként vissza van terjesztve, és a hálózat tanulható paramétereit (súlyait) olyan irányba és mértékbe frissíti a gradiens ereszkedés módszerével (6), hogy a veszteség függvény értéke minimális legyen. Az alfejezetben ismertetett egyenletek mind többrétegű neuronhálózatok tanítására vonatkoznak.

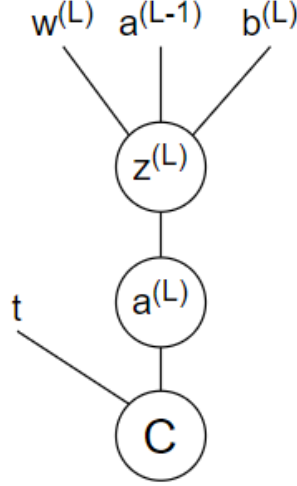
Minden iteráció végén meghatározható egy veszteségfüggvény érték. Ahhoz, hogy ezt minimalizálni lehessen, ahhoz először meg kell határozni $\frac{\partial C}{\partial w_i^{(L)}}$ értékét. Ezt a láncszabály szerint

egyszerűen meg lehet kapni (5).

$$\frac{\partial C}{\partial w_i^{(L)}} = \frac{\partial z_i^{(L)}}{\partial w_i^{(L)}} \cdot \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \cdot \frac{\partial C}{\partial a_i^{(L)}} \quad (5)$$

$$\text{ahol } z_i^{(L)} = w_i^{(L)} \cdot a_i^{(L-1)} + b^{(L)}.$$

Hogy miért alkalmazható a láncszabályt az a 7. ábrán látható.



7. ábra : Láncszabály a parciális deriválthoz

Az (5) -ös egyenlet bal oldala megmutatja, hogy az aktuális súlyérték változtatására mennyire érzékeny a veszteségfüggvény. Ha ez a művelet végre van hajtva az adott réteghez tartozó összes súlyhoz, megkapható a réteghez tartozó ∇C gradiens vektor, aminek (-1) -szeresével frissíteni lehet az adott réteghez tartozó súlyvektort (6) alapján (ha nem a (-1) -szeresével frissítené akkor nem csökkenne, hanem nőne a veszteségfüggvény értéke).

$$w^{(\tau+1)} = w^{(\tau)} - \eta \cdot \nabla C(w^{(\tau)}) \quad (6)$$

ahol $w^{(\tau+1)}$ a frissített súlyvektor az adott (L) rétegben.

Az (6) -ben szereplő η tanulási ráta értéke általában 0,01...0,0001, szerepe, hogy meghatározza a gradiens ereszkedés léptékét.

Az előző rétegre történő hiba visszaterjesztése (7) alapján történik, ahol szintén a 7. ábrán látható láncszabályt lehet alkalmazni.

$$\frac{\partial C}{\partial a_i^{(L-1)}} = \sum_{i=0}^{n^{(L-1)}} \frac{\partial z_i^{(L)}}{\partial a_i^{(L-1)}} \cdot \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \cdot \frac{\partial C}{\partial a_i^{(L)}} \quad (7)$$

A (7) -es egyenlet alapján elmondható, hogy ha ismerem, hogy az utolsó előtti réteg aktivitására mennyire érzékeny a veszteségfüggvény, akkor ezt a lépést meg lehet ismételni az összes korábbi rétegre, és (6) alapján frissíteni lehet a súlyértékeket.

A fejezet zárásául fontosnak tartom megjegyezni, hogy mivel általában a tanítás elején

véletlenszerűen kerülnek felvételre a súlyok kezdeti értékei (ezzel a veszteségfüggvény síkján is a kezdeti pontot), és a veszteségfüggvény sok változónak a függvénye, ezért valószínű, hogy a függvénynek nem a globális minimuma lesz megtalálva tanítás végére, hanem a kezdeti ponthoz legközelebb eső lokális minimum. Más módszerek is léteznek a gradiens ereszkedésen kívül a hálózat paramétereinek frissítésére, ezeket dolgozatomban külön nem fogom ismertetni, mivel egy általános ismertetés a célom.

2.4. Felhasznált architektúra

A kameraképen szereplő sávfelfestések meghatározásához egy szemantikus szegmentációs neurális hálózatnak a kimenetét használtam fel. A szemantikus szegmentálás az a folyamat, amikor a kép minden pixeléhez címkék vannak rendelve. Ugyanazon osztály több objektuma egyetlen entitásként van kezelve. Az általam felhasznált hálózatnak az architektúráját és működését fogom röviden bemutatni ebben a fejezetben.

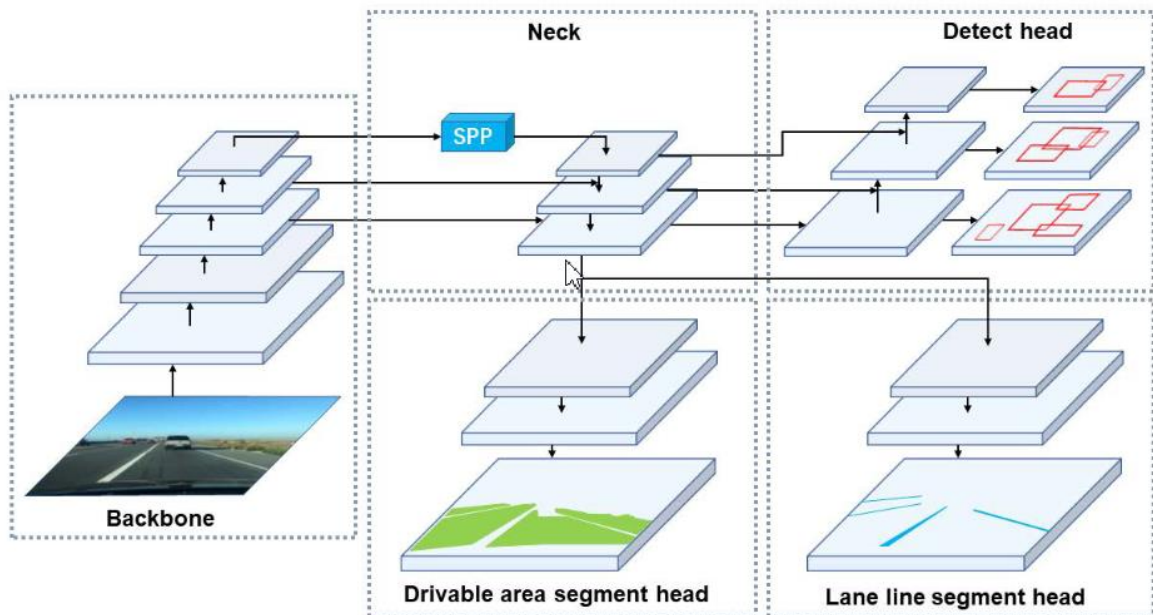
2.4.1. YOLOP (You Only Look Once for Panoptic)

A YOLOP [4] a YOLO (You Only Look Once) modell család korábbi verzióira épülő neurális hálózat, mely párhuzamosan három feladatot képes elvégezni egyszerre: objektumdetektálást, vezethető útfelület szegmentációját, valamint sávszélek szegmentálását. Készítőinek célja az volt, hogy egy olyan elejétől-végéig tanítható neurális hálózatot hozzanak létre, mely egy önvezető jármű számára a lehető legtöbb információt képes biztosítani a környezetéről, több feladat párhuzamosan történő megvalósításával. Mindezt valós időjű futási sebességgel, megfelelő pontossággal. A hálózat felépítése a 8. ábrán látható.

A képről történő tulajdonságok kinyeréséhez egy enkódert használtak fel, melynek kimenetét három dekódoló hálózat használja fel az egyes feladatok megvalósításához. Általánosságban elmondható, hogy ha a három feladatot külön-külön más neurális hálózatok látják el, akkor a számítási költségek jelentősen megnőnek, mivel az információ feldolgozása szekvenciálisan történik. Az autonóm járművek esetében az erőforrások jelentősen végesek, ezért fontos, hogy a szoftverek erőforrás igényét a fejlesztők minimalizálják.

A hálózat enkódere a CSPDarknet [5], amit a YOLOv4, illetve v5 verziójában is felhasználtak. Előnye, hogy megoldja a gradiens duplikáció problémáját, mivel a gradiens változások integrálva vannak a tulajdonságtérképekbe, ezzel csökkentve a hálózat paraméter számát, illetve a másodpercenként elvégzendő lebegőpontos számításokat (FLOPS). A gerinchálózatból érkező tulajdonságok egyesítésért az FPN [15] (Feature Pyramid Network) modulból és az SPP [16] (Spatial Pyramid Pooling) modulból álló nyakhálózat felelős. Az általa

generált tulajdonságok több szemantikai szintű információt tartalmaznak, mint az enkóder kimenete. Az első dekódoló a YOLOv4 fejhálózatahoz hasonlóképpen működik. PAN (Path Aggregation Network) [6] használatával pontosabb az objektumok lokalizációja és segít az alacsonyabb szintű tulajdonságok továbbításában a fokozott alulról-felfele történő információ áramlásának köszönhetően. A második és harmadik dekóder, amik a vezethető útfelület és sávvonal szegmentációjáért felelősek, ugyanazt a hálózati struktúrát alkalmazzák. A korábban említett FPN alsó rétege szolgáltatja a bemenetet, majd három mintavételezés után olyan tulajdonságtérképet szolgáltat, mely a bemenet minden egyes pixeléhez egy valószínűségi értéket rendel az alapján, hogy az adott pixel melyik osztályba tartozhat. Felmintavételezéshez dekonvolúció helyett a legközelebbi szomszéd interpolációt alkalmazták, ezzel tovább csökkentve a számítási költségeket.



8. ábra : YOLOP hálózat felépítése [4]

3. Kamera-LiDAR kalibráció

A kamerakép kétdimenziós pixeleinek háromdimenziós pontokra való konvertálására többféle módszer is létezik. Ebben a fejezetben azt fogom bemutatni, hogy LiDAR szenzor segítségével, hogyan lehet a pixeleket a háromdimenziós pontfelhőhöz hozzárendelni.

3.1. Kamera kalibrációja

Ahhoz, hogy síkbeli pixeleket térbeli pontokhoz lehessen hozzárendelni, először a kamera kalibrációjára [8] van szükség. Kamera kalibráció az a művelet amikor a torzítási és a kamera mátrixot kell meghatározni. A képeken általánosságban kétféle torzítás figyelhető meg: radiális, illetve tangenciális. Radiális torzítás a halszem effektusból ered, aminek következményeképpen a képen található egyenes vonalak az optikai középponttól távolodva egyre jobban elgörbülnek. A torzított pixelek x és y koordinátáit radiális torzítás esetében a (8) -as és a (9) -es egyenlet adja meg.

$$x_{torzított} = x \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad (8)$$

$$y_{torzított} = y \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad (9)$$

A tangenciális torzítás annak következtében jön létre, hogy a képalkotó lencse nem tökéletesen párhuzamos a képalkotó síkkal, ezért a kép néhány területe közelebbinek tűnhet. A torzított pixelek x és y koordinátáit tangenciális torzítás esetében a (10) -es és a (11) -es egyenlet adja meg.

$$x_{torzított} = x + (2 \cdot p_1 \cdot x \cdot y + p_2 \cdot (r^2 + 2 \cdot x^2)) \quad (10)$$

$$y_{torzított} = y + (p_1(r^2 + 2 \cdot y^2) + 2 \cdot p_2 \cdot x \cdot y) \quad (11)$$

Az ezekből az egyenletekből meghatározható együtthatók $\mathbf{d}$ sorvektorát nevezik torzítási mátrixnak (12).

$$\mathbf{d} = [k_1, k_2, p_1, p_2, k_3] \quad (12)$$

A $\mathbf{C}$ kamera mátrixot (13) az adott kamerára jellemző belső paraméterei alkotják: a gyújtótávolság és az optikai középpont pixel koordinátái.

$$\mathbf{C} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (13)$$

A kamera mátrix, illetve a torzítási mátrix meghatározásához gyakran használt módszer a sakktábla módszer. Egy sakktábla négyzetrács sarkainak pontjait térben és a kamerán pixel szerint meghatározva ki lehet számolni már a kamera, illetve torzítási mátrixot. Minél több a sarokpont, illetve mintaszám, annál pontosabbak lesznek az együtthatók értékei. Ezek után már

többféle algoritmus is rendelkezésre áll a kiegyenlített kamerakép előállításához. Torzított, illetve kiegyenlített képre példát a 9. ábrán lehet látni, ahol a Nissan Leaf-en lévő ZED kamerából érkező kép lett kiegyenlítve.



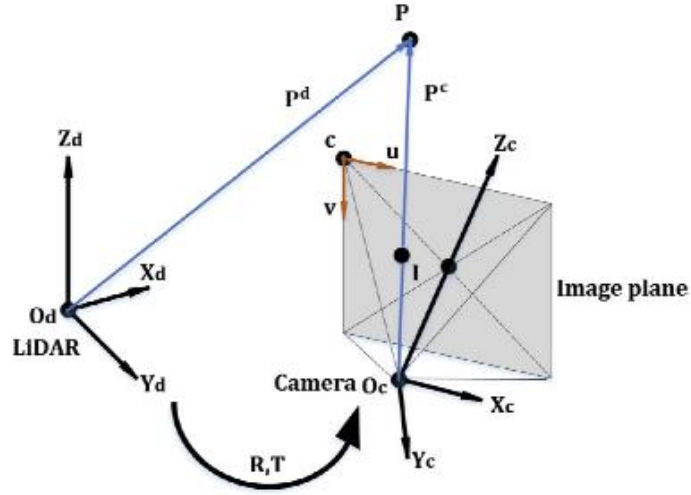
9. ábra : Nyers kép (bal), illetve a kiegyenlített kép (jobb)

3.2. Kamera-LiDAR kalibrációja

A fúzióhoz [9] szükséges belső paraméterek LiDAR esetében gyártó által meghatározottak, kamera esetében pedig a fentebb ismertett módszerrel meghatározható kamera mátrix formájában. Az kalibrációhoz szükséges továbbá egy vagy több kalibrációs objektum is, ezek lehetnek például sík sakktáblák.

Ezek ismeretében három koordináta-rendszert lehet definiálni: $O_{pi} - X_{pi}Y_{pi}Z_{pi}$, ahol O_{pi} az adott kalibrációs sakktáblának a bal felső sarokpontja; $O_d - X_dY_dZ_d$, ahol O_d a LiDAR pozíciója; $O_c - X_cY_cZ_c$, ahol O_c a kamera optikai középpontja, Z_c a kamera optikai tengelye, X_c, Y_c pedig a képsík vízszintes, illetve függőleges koordinátái.

A kamera-LiDAR rendszer külső paraméterei az $\mathbf{R}$ rotációs mátrix és a $\mathbf{T}$ transzformációs vektor. Ha van egy tetszőleges P pont a szabad térben, mint a 10. ábrán, ami a kamera által is látható, illetve a LiDAR által is mérhető, akkor $P^d = [X^d, Y^d, Z^d]^T$ P pont a LiDAR koordináta-rendszerében, $I = [u, v, 1]^T$ P pont a kamera koordináta-rendszerében.



10. ábra : 3D-2D pontok megfeleltetése [9]

Ha nincsen torzítás a kameraképen, akkor a 3D-2D pont megfeleltetését P^d -nek és I -nek a (14) -es egyenlet írja le.

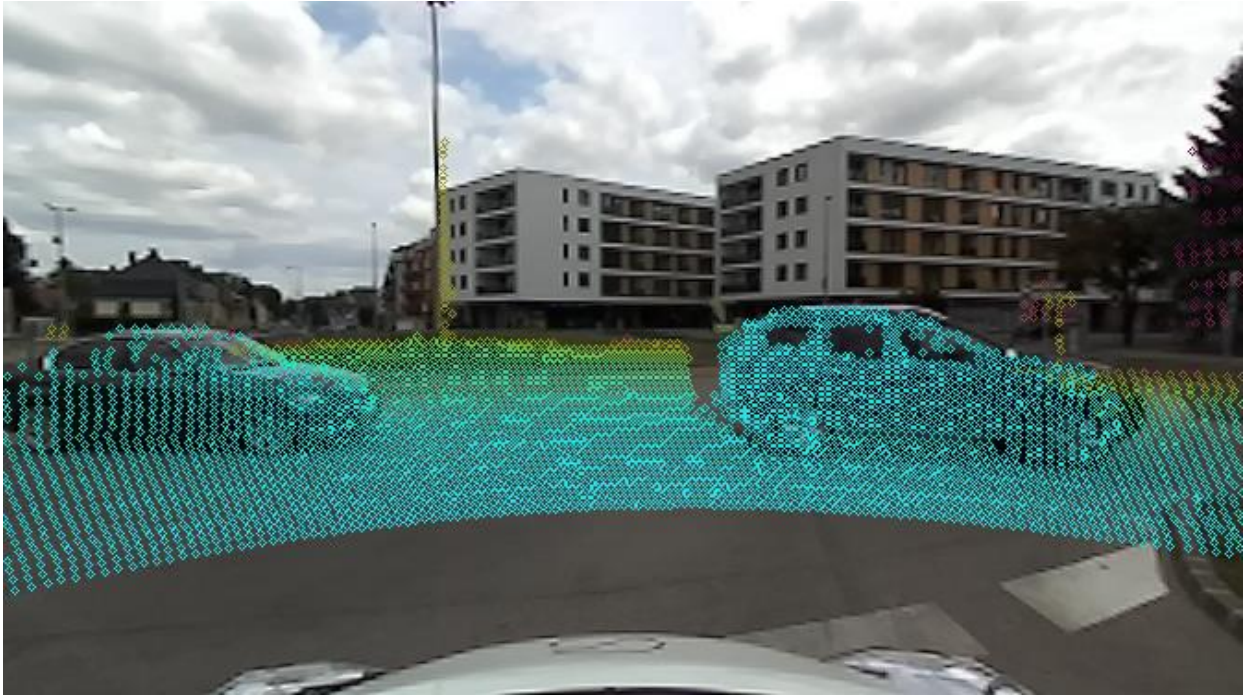
$$ZI = C \cdot (R \cdot P^d + T) \quad (14)$$

, ahol Z a (14) -es egyenlet jobb oldalán szereplő vektornak a harmadik eleme lesz, C pedig a kamera mátrix.

Ha P koordinátái kamera koordináta-rendszerben P^c , akkor a 3D-3D pont megfeleltetését P^c -nek és P^d -nek a (15) -ös egyenlet írja le.

$$P^c = R \cdot P^d + T \quad (15)$$

Ennek a két egyenletnek az ismeretében, különböző algoritmusokkal meglehet határozni mind a rotációs, mind a transzformációs mátrixot, amik segítségével hozzá lehet rendelni a kamerakép pixeleit a LiDAR-ból érkező pontfelhőhöz. Erre példát a 11. ábrán lehet látni, ahol a Nissan Leaf 64 csatornás LiDAR pontjai lettek fúzionálva a kamera pixeleivel.



11. ábra : Pontfelhő pontjai a kameraképen kamera-LiDAR kalibráció után

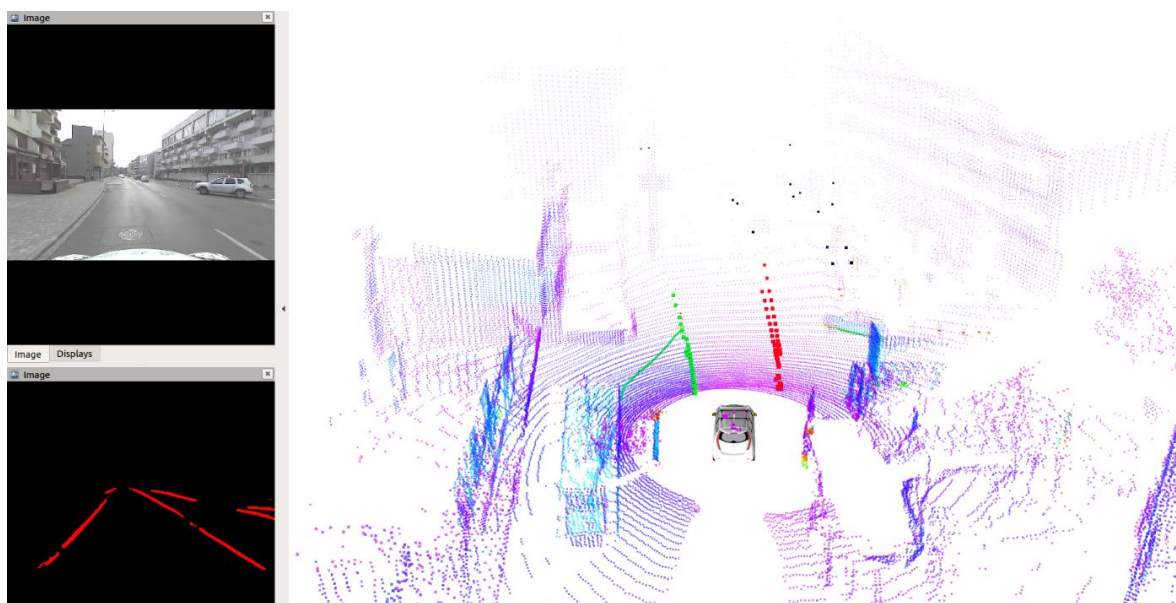
4. A kiértékeléshez használt adatkészlet

Mivel nincsen elérhető címkézett adatkészlet sávvonalakat reprezentáló pontfelhőről, ezért magamnak kellett létrehoznom egy kiértékelő adatkészletet. Több különböző forgalmi szituációban felvett rosbag alapján készítettem el, a pontfelhőt zaj, saját bal oldali sávvonal, illetve saját jobb oldali sávvonalak alapján címkéztem fel, mivel a felhasznált klaszterezési módszerek is elkülönítik az egyik klaszterbe se tartozó kívül eső pontokat, valamint a módszerem csak a saját sávokhoz tartozó pontokat csoportosítja. A szituációk mintaszámait azonos eloszlást követve címkéztem fel, ahogy azt az 1. táblázat is szemlélteti.

1. táblázat : Az adatkészlet mintáinak eloszlása forgalmi szituációkra bontva

Forgalmi szituáció sorszáma	Forgalmi szituáció megnevezése	Minták darabszáma
1.	Sávváltás 1x1 sávos úton	10
2.	Kanyarodás	10
3.	Egyenes haladás forgalom mellett	10
4.	Sávváltás 3x3 sávos úton	10
5.	Egyenes haladás forgalom nélkül	10

A címkézés előtt a pontfelhő fuzionálva lett a neurális hálózat kimenetével, ami maszkolva tartalmazza a sávvonal felfestéseket. Amelyik pont a kalibráció után beleesett a maszkolt pixelekre azok lettek címkézve, az adatkészlet egyik mintája a 12. ábrán látható.



12. ábra : Példa egy címkézett mintára az adatkészletből

A piros pontok reprezentálják a jobboldali sávot, a zöld pont a baloldali sávot, a fekete pontok pedig a zajt.

Mivel sajátkezűleg címkéztem a pontokat egyenként, ezért 50 db minta elkészítésére volt lehetőségem. Ezek a minták tartalmazznak sávváltást, kanyarodást, illetve többsávos utakat is.

5. Felhasznált klaszterezési módszerek

Klaszterezés feladata az előre nem címkézett adatok csoportosítása különböző tulajdonságaik alapján. Az esetben előre nem tudható, hogy mely fuzionált pontok tartoznak a bal, illetve jobb oldali sávvonalakhoz, mivel az egyszerű koordináta alapú szűréssel kanyarokban már nem határozhatóak meg az osztályok.

Többféle klaszterezési módszer is létezik, mint például a hierarchikus, sűrűség alapú és középpont alapú módszerek. Dolgozatomban csak a sűrűség alapú klaszterezésekkel fogok foglalkozni, mert a fent említett klaszterezési típusokból többet is megnéztem, és a probléma megoldásának szempontjából ez bizonyult a legrelevánsabbnak. A következőkben fogom bemutatni a felhasznált algoritmusokat, illetve a kiértékelésükre szolgáló mérőszámokat.

5.1. DBSCAN (Density - Based Spatial Clustering of Applications)

A DBSCAN [10] módszer arra a feltételezésre épül, hogy az egyes klaszterek sűrűbb régiók, amik el vannak különülve a kevésbé sűrűbb régióktól. Két állítható paraméterrel rendelkezik: ϵ és a minimum pontok száma.

Az ϵ megadja, hogy az adathalmaz pontjai körül mekkora rádiuszba vizsgálja az algoritmus a sűrűséget. A minimum pontok száma megadja, hogy minimum hány pontnak kell lennie az adott ponthoz tartozó ϵ rádiuszon belül ahhoz, hogy a pontot alappontnak lehessen tekinteni. Ezek alapján a DBSCAN a pontokat háromféleképpen definiálja: vannak a fent említett alappontok, vannak a határpontok és a zaj. Határpontnak számítanak azok a pontok, amelyeknek az ϵ rádiuszán belül kevesebb pont szerepel, mint a meghatározott küszöbérték. Zajnak számítanak azok a pontok, amelyeknek a rádiuszán belül egy pont se szerepel. A pontok definiálásához csak egyszer kell végig iterálni az adott adathalmazon, ami jelentősen csökkenti a számítási költségeket például a középpont alapú megoldásokkal szemben.

A klaszterezés folyamata előtt fontos a közvetlen sűrűséggel elérhetőséget, a sűrűséggel elérhetőséget, és a sűrűséggel összakapcsoltságot definiálni. P pont közvetlen sűrűséggel elérhető Q pontból, ha P pont a Q pont ϵ sugarán belülre esik, illetve Q pont alappont. Ez a tulajdonság nem szimmetrikus, mivel, ha P pont nem alappont, akkor Q pont nem közvetlen sűrűséggel elérhető P pontból.

P pont sűrűségében elérhető Q pontból, ha van egy olyan pontlánc, aminek első eleme P, utolsó eleme Q, valamint a köztük elhelyezkedő pontokra igaz, hogy egymás után közvetlen sűrűséggel elérhetőek.

P pont sűrűséggel összekapcsolt Q ponttal, ha van egy olyan O pont, ami mind P-vel, mind Q-val sűrűségében elérhető. Ez a tulajdonság szimmetrikusnak számít.

Ezek alapján egy sűrűség alapú klaszter olyan pontok halmaza, amelyek sűrűség elérhetőségük szempontjából maximálisak.

5.2. OPTICS (Ordering Points To Identify the Clustering Structure)

Az OPTICS [11] algoritmus a DBSCAN-re épül, és jobban teljesít változó sűrűségű adatok klaszterezésében. A módszer két új fogalommal van kiegészítve: magtávolság és az elérhetőségi távolság. Magtávolság az a minimum távolság, ami az adott pontnak szükséges ahhoz, hogy teljesítse a minimum pont küszöbértéket, ezzel alapponttá válva. Maximális értéke maga az ϵ . Ha nem alappont az adott pont, akkor nincsen hozzá magtávolság definiálva. Elérhetőségi távolság P és Q pontok között a P maximális magtávolsága, illetve P és Q euklideszi távolsága. Ha Q pont nem alappont, akkor P és Q között nem lehet elérhetőségi távolságot definiálni.

Az algoritmus az alábbiak alapján működik. Egy adott P ponthoz ki van számolva a magtávolsága, illetve elérhetőségi távolságai. Ezek után az elérhető pontok között feldolgozási sorrend van felállítva távolságuk alapján. Amelyik alappont legközelebbi szomszédja a P pontnak az lesz a következő feldolgozandó pont. Amikor P pontnak nincsen több szomszédja, az algoritmus a következő eddig nem látogatott pontot veszi P pontnak és kezdi előlről a fent leírt folyamatot. Minden pontnál el van tárolva az elérhetőségi távolsága, ami alapján válogatva lesznek a pontok, majd klaszterezve.

6. Klaszterezési mérőszámok

Különböző klaszterezési algoritmusok számára a gépi tanulás létezése óta több mérőszámot is definiáltak a kutatók, hogy jobban összetudják hasonlítani az egyes algoritmusok teljesítményét egy adott probléma megoldásában. Ebben a fejezetben fogom bemutatni az általam használt mérőszámokat.

6.1. Homogenitás

Egy tökéletesen homogén klaszterezésnél [12] minden klaszterbe azonos címkézésekkel rendelkező adatok szerepelnek. Ennél a mérőszámnál nem számít, hogy a klaszter és a címkézett osztály megegyezik-e, csak az, hogy az összetartozó címkéjű adatok egy klaszterbe kerültek-e. Számítása (16) alapján történik.

$$h = 1 - \frac{H(C|K)}{H(C)} \quad (16)$$

ahol $H(C|K)$ a feltételes entrópiája az osztályoknak a klaszterek alapján, $H(C)$ pedig az osztályok entrópiája.

Degenerált esetben, amikor $H(C) = 0$ (vagyis egy osztály létezik csak), akkor h értékéhez 1 van hozzárendelve.

6.2. Teljesség

Egy tökéletesen teljes klaszterezés [12] esetében az összes adat, amely ugyanazzal az osztállyal lett felcímkézve ugyanazokba a klaszterekbe kerülnek besorolásba. Ennél a mérőszámnál már számít, hogy a ponthoz rendelt klaszter megegyezik-e az eredeti címkézett osztállyal. Számítása (17) alapján történik.

$$c = 1 - \frac{H(K|C)}{H(K)} \quad (17)$$

Amint látható az egyenlet jobb oldala a tükörképe a (16) -os egyenlet jobb oldalának.

Degenerált esetben, amikor $H(K) = 0$, vagyis minden pont egy klaszterhez lett hozzárendelve, $c = 1$.

6.3. V-mérték

A V-mérték [12] a homogenitás és teljesség közötti harmonikus átlag, számítása (18) alapján történik.

$$v = \frac{(1 + \beta) \cdot h \cdot c}{(\beta \cdot h + c)} \quad (18)$$

ahol β egy súlyérték, aminek értéke alapértelmezetten 1.

A képlet alapján jól látszik, ha $\beta > 1$, akkor nagyobb súllyal lesz figyelembe véve a teljesség értéke, ellenkező esetben pedig a homogenitás értéke.

6.4. Korrigált Rand pontszám

A korrigált Rand pontszám hasonlóságot néz két klaszter között, úgy, hogy veszi az összes mintapárt és megszámlolja az azonos, illetve külön klaszterekbe tartozó mintapárok számait (19), majd ez kerül korrigálásra (20).

$$RI = \frac{a + b}{C_2^{n_{samples}}} \quad (19)$$

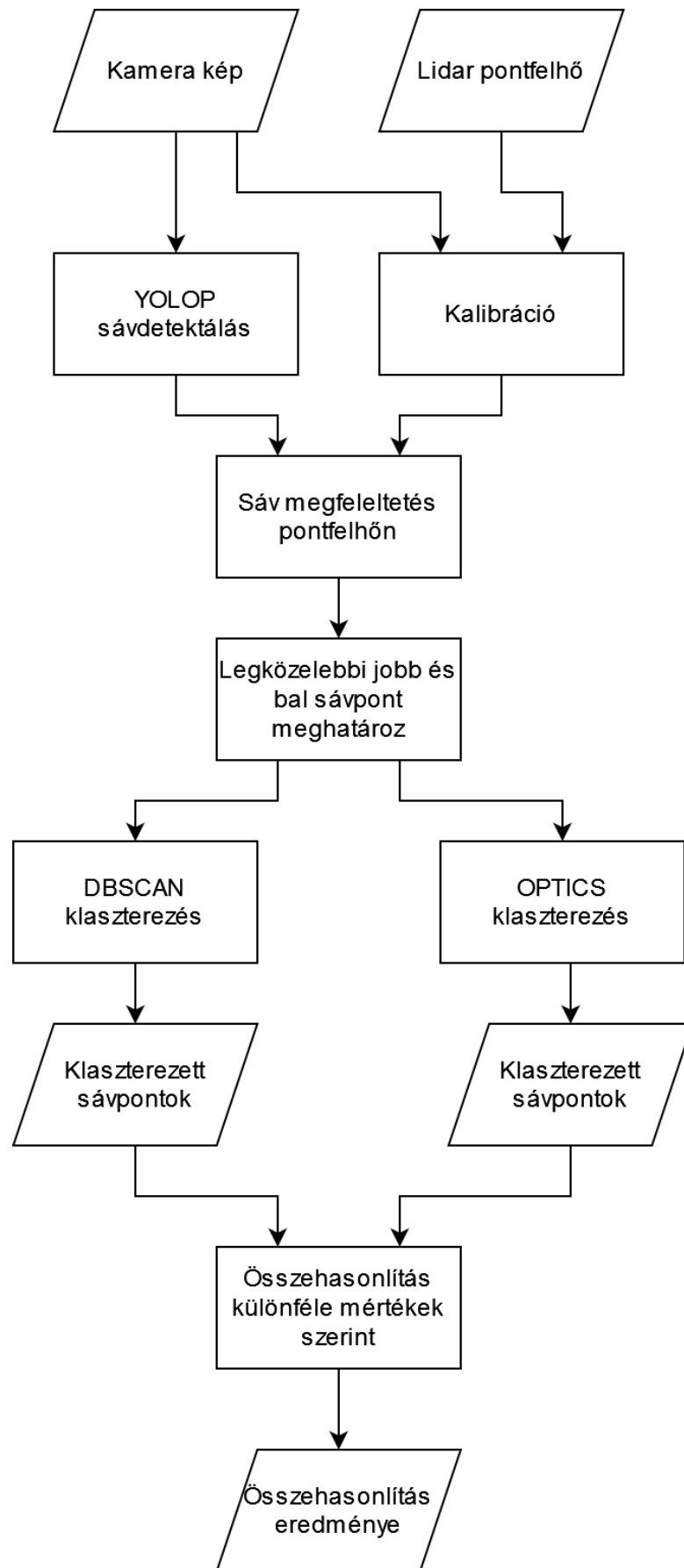
ahol a az azonos címkézett osztályba és prediktált klaszterbe tartozó mintapárok száma, b a különböző klaszterekbe tartozó mintapárok száma, $C_2^{n_{samples}}$ a lehetséges mintapárok száma az adatkészletbe.

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)} \quad (20)$$

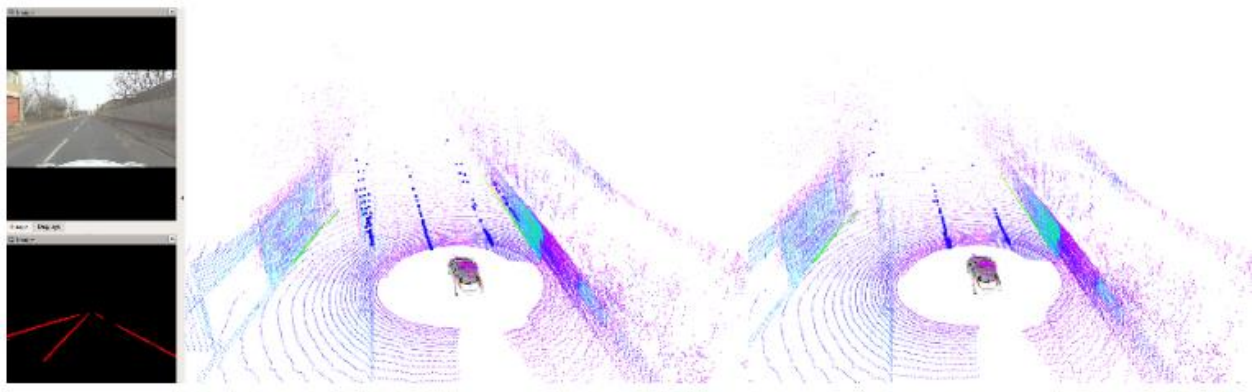
7. A kapott mérési eredmények bemutatása

A felhasznált klaszterezések kiértékelésére és összehasonlítására a korábban részletezett homogenitási és teljességi mérőszámokat, a V-mértéket és a korrigált Rand pontszámot vettem alapul. Mielőtt a kapott mérési eredményeket kiértékelem, bemutatom a használt módszer elvét, aminek a folyamatábrája a 13. ábrán látható, illetve szemléltetek néhány klaszterezési kimenetet az 1. táblázatban definiált forgalmi szituációkban.

Az algoritmus a kamera-LiDAR kalibráció után a pontfelhőből csak azokat a pontokat vette figyelembe melyek a szegmentált sávvonalak pixeleivel egybe estek. Ezután a megmaradt pontok a kétfelé lettek osztva Y koordinátájuk alapján a jármű koordináta-rendszeréhez viszonyítva. Az egyik ponthalmaz tartalmazta a negatív Y koordinátájú pontokat, a másik a pozitív Y koordinátájú pontokat tartalmazta. Mindkét ponthalmaznál megvizsgáltam, hogy melyik ponthoz rendelt helyvektornak az euklideszi normája volt a legkisebb. Így megkaptam a járműhöz viszonyított legközelebbi bal, illetve jobb oldali sávpontokat. Klaszterezés után csak azokat a pontokat továbbítottam a kimenetre, amelyek címkéje megegyezett a két legközelebbi sávpont címkéjével. Ezzel megakadályoztam, hogy 2-nél több klaszter kerüljön a kimenetre, tehát így biztosan csak a saját sávok lesznek klaszterezve. A klaszter címkék folyamatos változása is ki van küszöbölve ezzel a módszerrel. A 14. és 15. ábrán láthatóak a nyersen fuzionált pontok pontfelhője kézzel, illetve az előbb említett módszerrel kiszűrt pontok szintén kék színnel megjelenítve.



13. ábra : A módszer blokkdiagramja



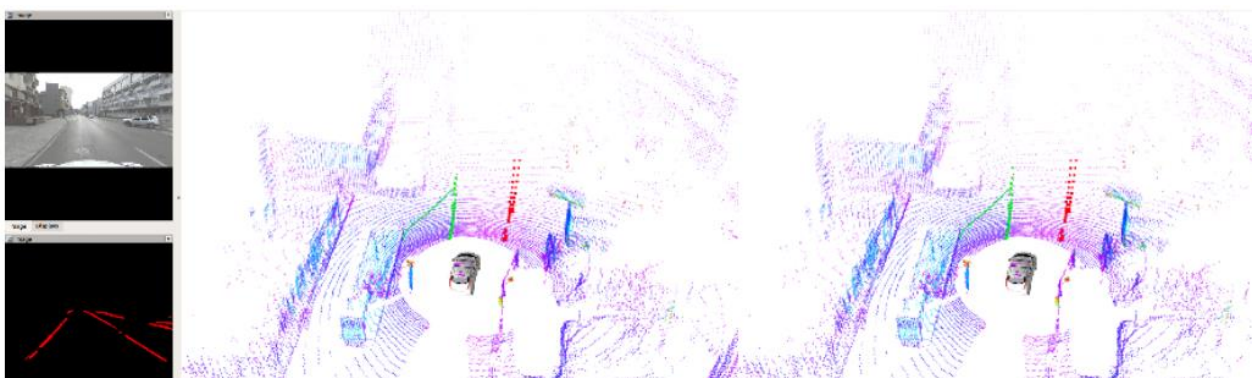
14. ábra : Fúzionált pontok az ötödik forgalmi szituációban nyersen (bal), és szűrve (jobb)



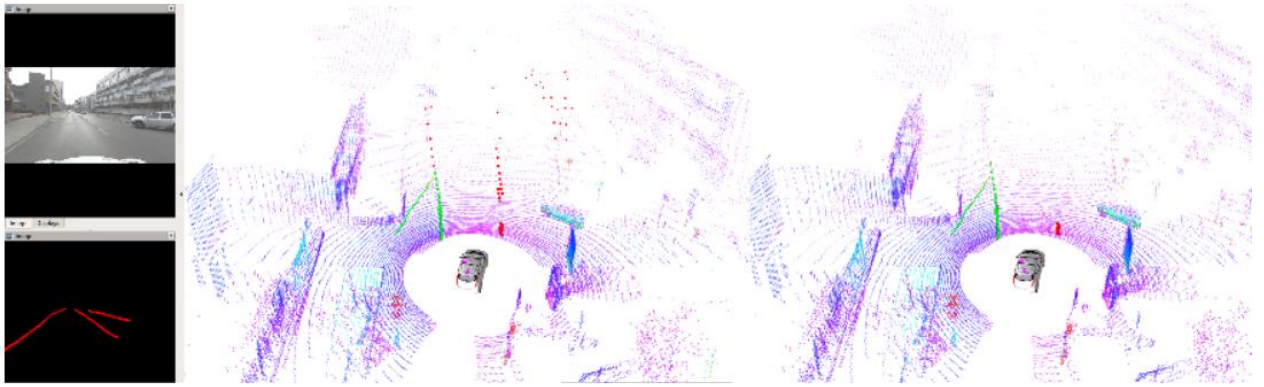
15. ábra : Fúzionált pontok az első forgalmi szituációban nyersen (bal), és szűrve (jobb)

Látható, hogy a szűrt pontok tartalmazzák még a klaszterezés által zajnak ítélt pontokat, ezek könnyen eltávolíthatók a kimenetről, mivel mind a DBSCAN, mind az OPTICS algoritmus egyedi, -1-es címkével látja el a zajpontokat. Módszerek optimális paramétereinek megválasztása empirikus módon történt.

A továbbiakban a két módszer kimenetei láthatóak 4 különböző forgalmi szituációkban. Az ábrákon látható jelenetek a mérések ugyanazon időbéljegyeihez tartoznak.

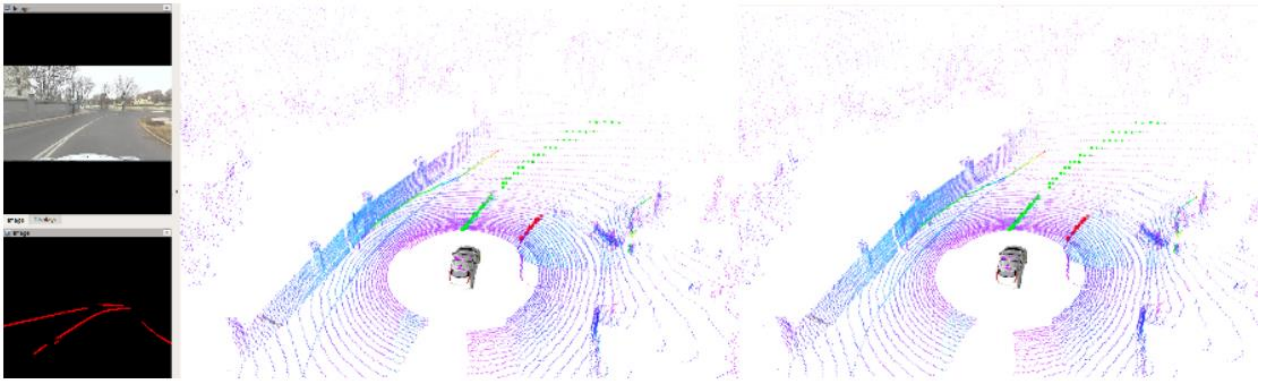


16. ábra : OPTICS (bal), DBSCAN (jobb) által alkotott klaszterek az első forgalmi szituációban, 1

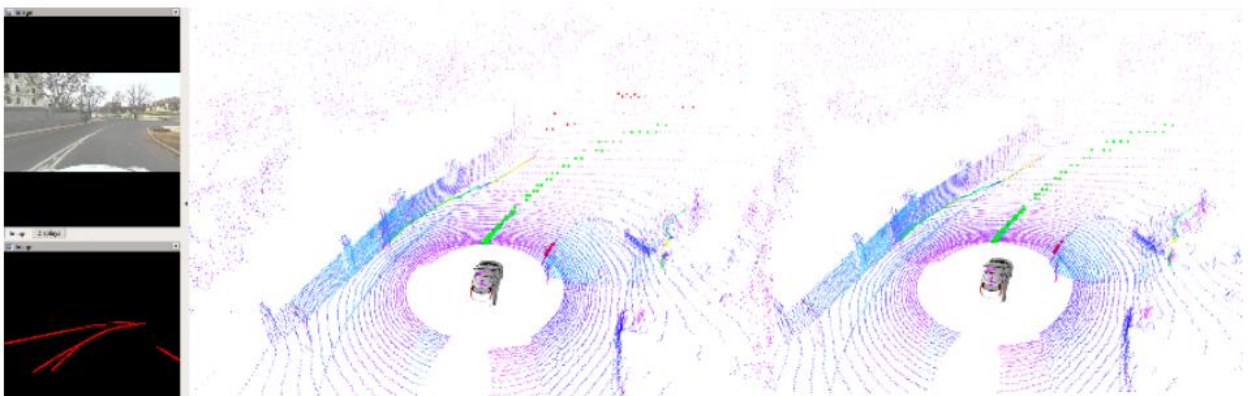


17. ábra : OPTICS (bal), DBSCAN (jobb) által alkotott klaszterek az első forgalmi szituációban, 2

A 16., 17. ábrákon látható, hogy mindkettő módszer a sávváltás előtt szinte azonos kimenetet ad, viszont közeledve a jobb oldali sávhoz az OPTICS hajlamos a DBSCAN által zajnak ítélt pontokat is klaszterbe venni.



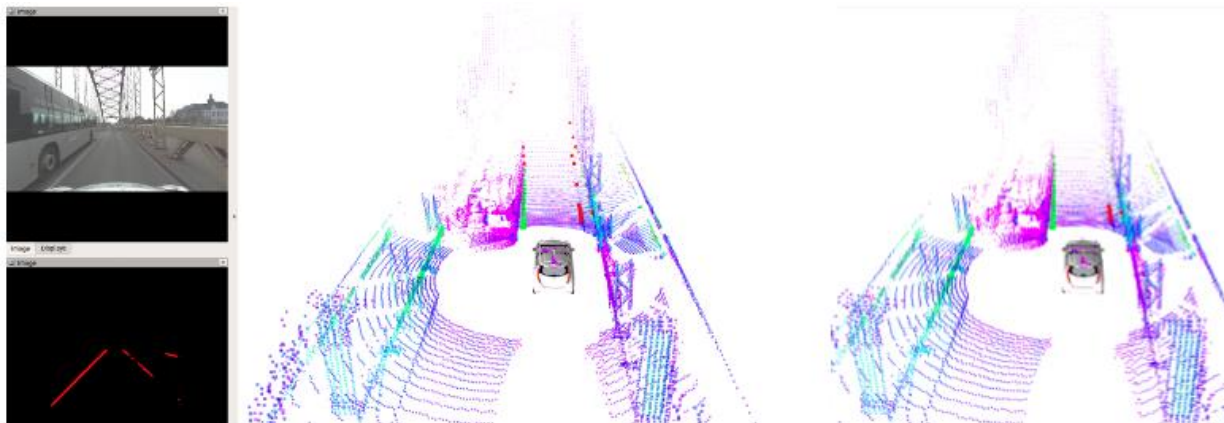
18. ábra : OPTICS (bal), DBSCAN (jobb) által alkotott klaszterek a második forgalmi szituációban, 1



19. ábra : OPTICS (bal), DBSCAN (jobb) által alkotott klaszterek a második forgalmi szituációban, 2

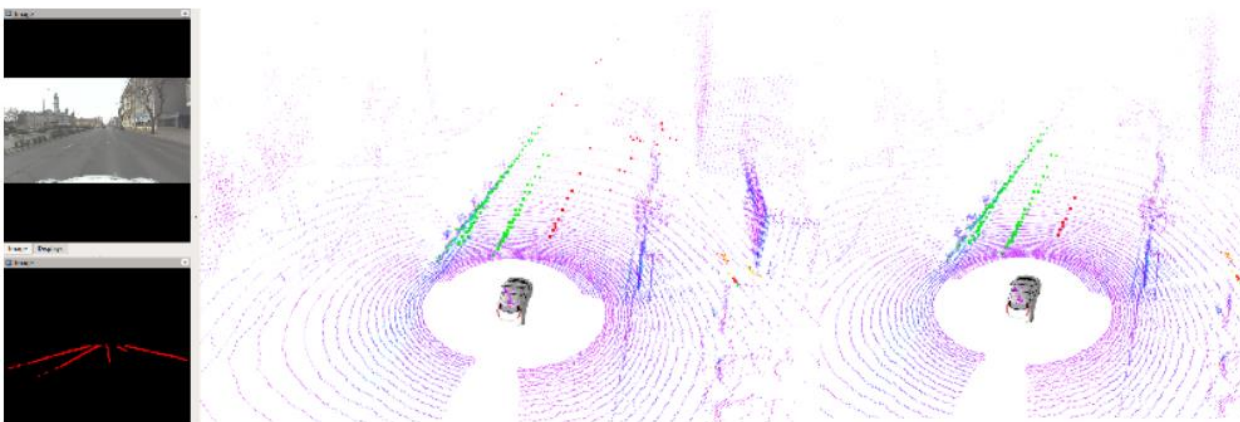
A 18. és 19. ábrákon látható, hogy az algoritmusok milyen kimenetet szolgáltatnak kanyarodás esetén.

Ugyanúgy elmondható, mint fentebb, hogy a két algoritmus hasonló kimenetet biztosít azzal a különbséggel, hogy az OPTICS itt is hajlamos néhány zajpontot klaszterpontnak venni.



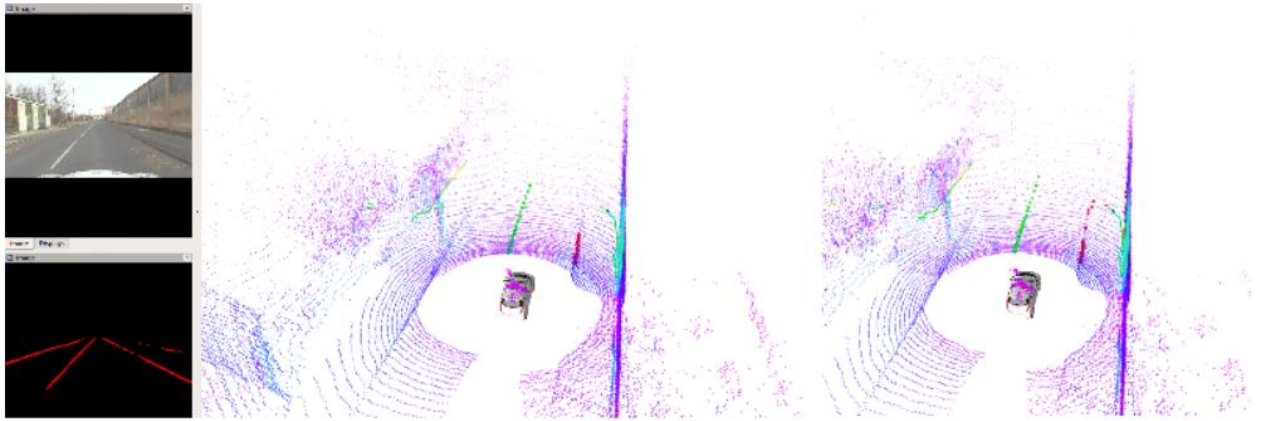
20. ábra : OPTICS (bal), DBSCAN (jobb) által alkotott klaszterek a harmadik forgalmi szituációban

20. ábra látszik, hogy a DBSCAN statikus ϵ paraméterének köszönhetően jelentősen rövidebb sávval információkat szolgáltat a jobb oldalról, mint a másik módszer, cserébe zajmentes a kimenete.



21. ábra : OPTICS (bal), DBSCAN (jobb) által alkotott klaszterek a negyedik forgalmi szituációban

A 21. ábrán látható forgalmi szituáció alapján a következő mondható el: A DBSCAN nem csak a saját sávval pontjainak bal oldalát klaszterezi, hanem a mellette levő sáv bal oldalát is beleveszi ugyanabba a klaszter csoportba. Az OPTICS algoritmusnál ez nem csak a bal oldalon, hanem jobb oldalon is megfigyelhető, valamint a zajszűrése se annyira effektív, mint a másik módszer esetében.



22. ábra : OPTICS (bal), DBSCAN (jobb) által alkotott klaszterek az ötödik forgalmi szituációban

A 22. ábrán szinte hasonló kimenetek figyelhetők meg mindkét módszer esetében, azzal a különbséggel, hogy a DBSCAN sávvonal végpontjai távolabb helyezkednek el, mint az OPTICS esetében.

A következőkben a mérőszámok alapján fogom összehasonlítani a két módszert forgalmi szituációkra felosztva, illetve összességében.

2. táblázat : Klaszterezési algoritmusok teljesítménye az első forgalmi szituációban

DBSCAN	OPTICS	
0,796	0,487	Homogenitás
0,795	0,535	Teljesség
0,795	0,508	V-mérték
0,798	0,475	Korrigált Rand pontszám

Az 2. táblázat alapján a DBSCAN teljesítménye sávváltás esetén egyértelműen jellemezhető mivel mindegyik mérték pontszáma szinte azonosan jó. Az OPTICS a kapott eredmények alapján jóval bizonytalanabb, főleg a klaszterek felosztásában, ahogy azt a homogenitási pontszáma is mutatja, feltehetően a korábban említett zajkezelési problémái miatt bizonyos szituációkban.

3. táblázat : Klaszterezési algoritmusok teljesítménye a második forgalmi szituációban

DBSCAN	OPTICS	
0,911	0,386	Homogenitás
0,862	0,885	Teljesség
0,883	0,538	V-mérték
0,901	0,347	Korrigált Rand pontszám

A 3. táblázat alapján elmondható, hogy a DBSCAN kanyarokban is jobban teljesít, mint az OPTICS algoritmus, mivel mindegyik mérőszám alapján magas pontszámot szerzett az adatkészleten. Az OPTICS esetében ki szeretném emelni, hogy a teljességi pontszáma jelentősen magasabb, mint a többi érték. Ez arra utal, hogy a pontok túlnyomó többségében megfelelő klaszterekbe kerültek az adatkészlet címkézései alapján, csak maguknak a klasztereknek az alakja különbözik a címkézett adatokétól.

4. táblázat : Klaszterezési algoritmusok teljesítménye a harmadik forgalmi szituációban

DBSCAN	OPTICS	
0,752	0,272	Homogenitás
0,793	0,423	Teljesség
0,765	0,331	V-mérték
0,789	0,212	Korrigált Rand pontszám

A 4. táblázat alapján az OPTICS rossz teljesítménye szintén annak köszönhető, hogy az algoritmus hajlamos a zajt is valamelyik klaszterhez hozzárendelni.

5. táblázat : Klaszterezési algoritmusok teljesítménye a negyedik forgalmi szituációban

DBSCAN	OPTICS	
0,353	0,219	Homogenitás
0,511	0,271	Teljesség
0,411	0,239	V-mérték
0,283	0,146	Korrigált Rand pontszám

A 5. táblázatban a DBSCAN értékei is jelentősen csökkentek az előző forgalmi szituációkhoz képest. Feltehetően a sávok ϵ paraméternél közelebb helyezkedtek el egymáshoz, ezért más sávok is klaszterekbe kerültek. Ennek köszönhetően más lett a klaszterek szerkezete, amit az alacsony homogenitási és alacsony korrigált véletlenszerű pontszám is igazol.

6. táblázat : Klaszterezési algoritmusok teljesítménye az ötödik forgalmi szituációban

DBSCAN	OPTICS	
0,824	0,533	Homogenitás
0,811	0,502	Teljesség
0,816	0,516	V-mérték
0,857	0,508	Korrigált Rand pontszám

Az 6. táblázat adatai alapján a következőket lehet megállapítani: a DBSCAN egy olyan forgalmi szituációban is, mint az egyszerű előrehaladás, jóval precízebben képes klaszterezni a sávvonalakat, az OPTICS-hoz tartozó magasabb értékek pedig a szituáció bonyolultságának köszönhetőek.

7. táblázat *Klaszterezési algoritmusok teljesítménye összességében*

DBSCAN	OPTICS	
0,766	0,405	Homogenitás
0,783	0,518	Teljesség
0,774	0,454	V-mérték
0,779	0,367	Korrigált Rand pontszám

A 7. táblázat tartalmazza az előző öt forgalmi szituáció pontszámait átlagolva. Mind a kettő módszer a klaszterezés teljességi mértékéhez kapta a legnagyobb pontszámot. Ez várható volt, mivel a fejezet elején leírt módszer folyamatosan megtartja a klaszterekhez tartozó címkéket mind a bal, mind a jobb oldali sávvonalakhoz. Az OPTICS algoritmus a kapott mérési eredmények alapján kevésbé alkalmas sávvonalakat reprezentáló pontok klaszterezésére, mint a DBSCAN, ami az algoritmus elődje.

8. Összefoglalás

A feladatomban az volt, hogy egy sávvonal szegmentáló neurális hálózat kimenetét kamera-LiDAR kalibráció után klaszterezzem több módszerrel, valamint ezeket a módszereket különböző klaszterezési mérőszámok alapján hasonlítsam össze. Ezt a vizsgálatot azért tartottam fontosnak mivel szemantikus szegmentáció módszeréből fakadóan nincsen információ a hálózat kimenetén arról, hogy a sávvonalak a járműhöz viszonyítva melyik oldalhoz tartoznak. Ez pedig egy útvonal tervező algoritmus számára elengedhetetlen bemeneti információ. Az alábbiakban fogom részletezni a tapasztalataimat.

Sávvonalak klaszterezéséhez többféle típusú módszert is kipróbáltam, a probléma jellegéhez a sűrűség alapú klaszterezések álltak legközelebb. Egyrészt azért, mert a hierarchikus, illetve középpont alapú módszerekkel ellentétben a sűrűség alapú módszerek képesek zaj kezelésére, másrészt a LiDAR szenzor kimenetének jellege miatt. Kamera-LiDAR kalibráció során előfordul, hogy az adott pixelnek nem csak a számunkra hasznos objektumhoz tartozó pontok feleltethetőek meg, hanem távolabb eső pontok is. Ezeknek a kiszűrése fontos feladat.

Dolgozatomhoz az OPTICS és DBSCAN nevezetű algoritmusokat használtam fel, melyek közül az adatkészletemen a DBSCAN teljesített jelentősen jobban. Véleményem szerint azért, mert a statikus epszilon paraméterrel jobban lehet kezelni a LiDAR pontokból álló klasztereket. Mivel a pontok sűrűsége a szenzortól távolodva egyre jobban csökken, ezért ezeket a pontokat az OPTICS algoritmus jellegéből következően több klaszterként is letudja kezelni, ami egy alapvetően előnyös tulajdonsága, esetemben nem hasznos.

Összességében elmondható, hogy ha a kamera képen levő információ LiDAR pontokkal van fűzionálva, és ezeket az adatokat klaszterezni szeretnénk, érdemes sűrűség alapú módszert felhasználni. Továbbá sávvonalak elhelyezkedésének meghatározására is felhasználhatóak a klaszterezési algoritmusok, mivel a DBSCAN az adatkészletemen, bonyolultabb forgalmi szituációkban is megfelelően teljesített az adott klaszterezési mérőszámok szerint.

9. Köszönetnyilvánítás

A dolgozat kutatásaihoz az Új Széchenyi Terv keretein belül a „Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási technológiák területén (EFOP-3.6.3-VEKOP-16-2017-00001)” projekt és a Széchenyi István Egyetem biztosított forrást. A kutatás az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

10. Irodalomjegyzék

- [1] Bishop, C. M.: *Neural Networks for Pattern Recognition*, „Könyv”: Clarendon Press, Oxford, pp. 77-380, 1995
- [2] Gardner, M.W.; Dorling, S. R.: *Artificial neural networks (the multilayer perceptron) – a review of applications in the atmospheric sciences*, Atmos. Environ., 32 (14-15), pp. 2627-2636, 1998
- [3] Jianxin, W.: *Introduction to Convolutional Neural Networks*, „Egyetemi jegyzet”: Nanjing University, National Key Lab for Novel Software Technology, pp. 5-25, 2017
- [4] Wu, D.; Liao, M.; Zhang, W.; Wang, X.; Bai, X.; Cheng, W.; Liu, W.: *YOLOP: You Only Look Once for Panoptic Driving Perception*, „Konferenciakiadvány”: arXiv:2108.11250v7 [cs.CV]
- [5] Wang, C. Y.; Liao, M., H. Y.; Wu, Y. H.; Chen, P. Y.; Hsieh, J. W.; Yeh, I. H.: *CSPNet: A new backbone that can enhance learning capability of cnn*, „Konferenciakiadvány”: Conference on Computer Vision and Pattern Recognition (CVPR 2020), Washington, DC, USA pp. 390-391, 2020
- [6] Wang, K.; Liew, J. H.; Zou, Y.; Zhou, D.; Feng, J. Panet: *Few-shot image semantic segmentation with prototype alignment*, „Konferenciakiadvány”: International Conference on Computer Vision (ICCV 2019), Seoul, Korea, pp. 9197-9206, 2019
- [7] Stanford Artificial Intelligence Laboratory et al.(2018): *Robotic Operating System*. <https://www.ros.org>, (Letöltés ideje: 2022. 04. 11.)
- [8] Zhang, Z.: *A Flexible New Technique for Camera Calibration*, „Konferenciakiadvány”: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1330-1334, 2000
- [9] An, P.; Ma T.; Yu, K.; Fang, B.; Zhang, J.; Fu, W.; Ma, J.: *Geometric calibration for LiDAR-camera system fusing 3D-2D and 3D-3D point correspondences*, „Folyóirat”: Optics Express, Vol. 28, No. 2/20, Január, 2020
- [10] Ester, M.; Kriegel, H. P.; Sander, J.; Xu, X.: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, „Konferenciakiadvány”: KDD '96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 226-231, 1996
- [11] Ankerst, M.; Breunig, M. B.; Kriegel, H. P.; Sander, J.: *OPTICS: Ordering Points To Identify the Clustering Structure*, „Konferenciakiadvány”: ACM SIGMOD '99:

International Conference on Management of Data, Philadelphia, 1999

- [12] Rosenberg, A.; Hirschberg, J.: *V-Measure: A conditional entropy-based external cluster evaluation measure*, „Konferenciakiadvány”: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, pp. 410-420, 2007
- [13] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, „Konferenciakiadvány”: 33rd Conference on Neural Information Processing Systems, Vancouver, Canada pp. 2-9, 2019
- [14] Pedragosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dobourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E.: *Scikit-learn: Machine Learning in Python*, „Folyóirat”: Journal of Machine Learning Research, Vol. 12, pp. 2825-2830, 2011
- [15] Lin, T. Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S.: *Feature pyramid networks for object detection*, „Konferenciakiadvány”: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117-2125, 2017
- [16] He, K.; Zhang, X.; Ren, S.; Sun, J.: *Spatial pyramid pooling in deep convolutional networks for visual recognition*, „Konferenciakiadvány”: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1904-1916, 2015