VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBJEKTŮ V LASEROVÝCH SKENECH POMOCÍ KONVOLUČNÍCH NEURONOVÝCH SÍTÍ

DETECTION OF OBJECTS IN LASER SCANS USING CONVOLUTIONAL NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. PETER MARKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN VEĽAS, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Marko Peter, Bc.**
Program: Informační technologie
Obor: Inteligentní systémy
Název: **Detekce objektů v laserových skenech pomocí konvolučních neuronových sítí**
Object Detection in the Laser Scans Using Convolutional Neural Networks
Kategorie: Umělá inteligence
Zadání:

1. Zorientujte se v problematice počítačového vidění v robotice a v základních technikách laserového skenování a senzoru Velodyne LiDAR.
2. Seznamte se s konceptem konvolučních neuronových síti se zaměřením na detekci objektů a sémantickou segmentaci.
3. Vyberte vhodné metody a nástroje, navrhněte neuronovou síť pro detekci vybrané skupiny objektů (např. přírodní objekty, silnice, vozidla, a pod.).
4. Navržené řešení implementujte s využitím existujících frameworků pro hluboké učení.
5. Experimentujte s vaší implementací a případně navrhněte vlastní modifikace metod.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Veľas Martin, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 30. října 2020

Abstrakt

Práca sa zaoberá detekciou čiary vodorovného dopravného značenia z mračna bodov, ktoré bolo získané laserovým mobilným mapovaním. Systém pracuje interaktívne v spolupráci s užívateľom, ktorý vyznačí počiatok čiary dopravného značenia. Program postupne deteguje zvyšné časti dopravného značenia a vytvorí ich vektorovú reprezentáciu. Na začiatku je mračno bodov premietnuté do vodorovnej roviny a výsledkom je 2D obrázok, ktorý je segmentovaný konvolučnou neurónovou sieťou U-Net. Segmentácia označuje jednu dopravnú čiaru. Segmentácia je prevedená na lomenú čiaru, ktorú je možné použiť v geo-informačnom systéme. Sieť U-Net pri testovaní dosiahla presnosť segmentácie 98,8%, špecificitu 99,5% a senzitivitu 72,9%. Odhadnutá lomená čiara dosiahla priemernú odchýlku 1,8cm.

Abstract

This thesis is aimed at detection of lines of horizontal road markings from a point cloud, which was obtained using mobile laser mapping. The system works interactively in cooperation with user, which marks the beginning of the traffic line. The program gradually detects the remaining parts of the traffic line and creates its vector representation. Initially, a point cloud is projected into a horizontal plane, crating a 2D image that is segmented by a U-Net convolutional neural network. Segmentation marks one traffic line. Segmentation is converted to a polyline, which can be used in a geo-information system. During testing, the U-Net achieved a segmentation accuracy of 98.8%, a specificity of 99.5% and a sensitivity of 72.9%. The estimated polyline reached an average deviation of 1.8cm.

Kľúčové slová

vodorovné dopravné značenie, lomená čiara, počítačové videnie, laserové skenovanie, velodyne LiDAR, mračno bodov, konvolučné neuronové siete, detekcia objektov, U-Net, sémantická segmentácia, hlboké učenie, PCL, QGIS, Keras, TensorFlow

Keywords

horizontal traffic signs, polyline, computer vision, laser scanning, Velodyne LiDAR, point cloud, convolutional neural networks, object detection, U-Net, semantic segmentation, deep learning, PCL, QGIS, Keras, TensorFlow

Citácia

MARKO, Peter. *Detekce objektů v laserových skenech pomocí konvolučních neuronových sítí.* Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Veľas, Ph.D.

Detekce objektů v laserových skenech pomocí konvolučních neuronových sítí

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pana Ing. Martina Veľasa, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Peter Marko
13. mája 2021

Poďakovanie

Chcem sa poďakovať mojej rodine za podporu a rady pri vypracovávaní diplomovej práce a vedúcemu diplomovej práce pánovi Ing. Martinovi Veľasovi, Ph.D. za konzultácie, odborné vedenie, trpezlivosť a podnetné rady k práci. Chcem sa tiež poďakovať firme Geodrom za poskytnuté dáta.

Obsah

Kapitola 1

Úvod

S veľkým rozvojom v oblasti autonómneho riadenia vozidiel, asistenčných a navigačných systémov nastáva výraznejší záujem o výskum v oblasti mapovania s veľkým rozlíšením. Takéto detailné mapy môžu poskytovať informácie o objektoch súvisiacich s dopravou, ako napríklad jazdné pruhy, okraje vozovky, rozdeľovanie jazdných pruhov, dopravné signalizačné zariadenia, chodníky a ostatné kritické dáta potrebné pre bezpečnú navigáciu po ceste. Vysoko presné mapy sa využívajú nielen v oblasti asistenčných systémov vozidiel, ale tiež ich používajú správcovia ciest na kontrolu a údržbu cestných komunikácií a značení.

Na zachytenie 3D geometrie toho, ako vyzerá okolie cesty, sa používajú skenovacie zariadenia založené na princípe merania vzdialenosti laserovým pulzom (LiDAR). Systém mobilného laserového skenovania (MLS) umožňuje priamy zber presných informácií v podobe 3D mračien bodov. Mobilné laserové skenovanie je menej náchylné na vplyv počasia a svetelných podmienok, než techniky založené na zbere obrazových dát pomocou kamery. Bližšie informácie o zbere dát a mapovaní sú popísané v kapitole 2.

Cieľ tejto práce je vytvoriť systém detekcie čiar na vozovke z mračna bodov. Zamerali sme sa najmä na využitie konvolučných neurónových sietí, ktoré vo všeobecnosti dosahujú veľmi dobré výsledky pri detekcii objektov na obrázkoch. Princípy konvolučných sietí sú spomenuté v kapitole 3. Konvolučné neurónové siete očakávajú vstup v podobe matice, alebo tenzoru. Na prevedenie trojrozmerného mračna bodov na obrázok sa používa vodorovná projekcia do plochy. Výsledný výrez obsahuje dve vrstvy, prvá určuje intenzitu odrazeného svetla od povrchu, a druhá vrstva znázorňuje hĺbku v danom bode.

Takto vytvorený výrez je privedený na vstup konvolučnej neurónovej siete U-Net, ktorá vykoná sémantickú segmentáciu vodorovného dopravného značenia (VDZ). Na vstupnom obrázku sa môže nachádzať viacero čiar, avšak výstupom je obrázok označujúci len jednu čiaru. Program sa v budúcnosti bude používať interaktívnou formou. To znamená, že užívateľ kliknutím zadá súradnice počiatku čiary a program vytvorí výrez okolia daného bodu a odhadne pozíciu čiary. Ďalej bude program čiaru podobným spôsobom predlžovať, kým je dostatočná istota, že čiara pokračuje. Podrobný popis architektúry použitého riešenia je v kapitole 5.

Presnosť programu bude vyhodnotená pomocou výpočtu plochy medzi čiarou odhadnutou Prediktorom a vzorovou čiarou. Plocha bude vypočítaná metódou numerickej integrácie, a potom bude normalizovaná dĺžkou čiary. Výpočtom presnosti a dosiahnutými výsledkami sa zaoberá kapitola 7.

Kapitola 2

Mobilné mapovanie vodorovného dopravného značenia

V tejto kapitole sa zameriame na popis základných pojmov. Ukážeme základné typy vodorovného dopravného značenia na vozovke. Vysvetlíme, čo je mračno bodov a predstavíme technológiu Velodyne LiDAR, ktorá slúži na vytváranie mračna bodov.

Mobilné laserové skenovanie (MLS) je systém ktorý nezávisle od svetelných podmienok je schopný vytvoriť veľmi presné 3D mračná bodov, kde každý bod okrem troch súradníc obsahuje aj intenzitu odrazeného svetla od povrchu [13].

2.1 Mračno bodov

Mračno bodov (anglicky point cloud) je množina neusporiadaných bodov v 3D priestore. Každému bodu sú priradené tri priestorové súradnice: X, Y a Z. Bodu v priestore môžu byť priradené aj iné atribúty ako napríklad intenzita, normála alebo farba. Rozdiel medzi mračnom bodov a obrázkom je veľmi výrazný. Obrázok má presne definovanú 2D mriežku pixelov a každému pixelu je pridelená hodnota. Naproti tomu mračno bodov nemá definovanú takúto mriežku, súradnice bodov sú vo všeobecnosti reálne čísla. Toto sa ukazuje ako nevýhoda pri použití konvolučných neurónových sietí, ktoré očakávajú vstup s presne definovanou mriežkou. Riešením tohoto problému môže byť konverzia mračna bodov do mriežky, či vytvorenie 2D projekcie mračna. Ďalšou možnosťou, ako tento problém vyriešiť je privedenie mračna priamo do neurónovej siete zvanej PointNet, ktorá využíva symetrickú funkciu na to, aby bol model nezávislý od poradia bodov na vstupe [6].

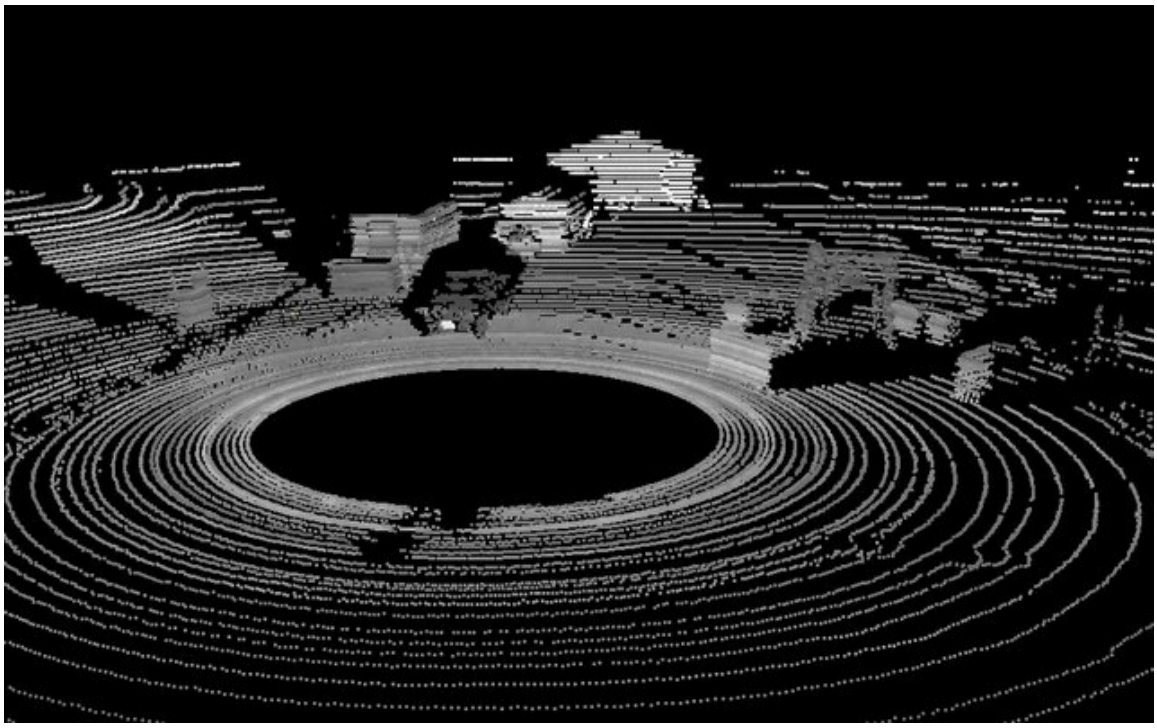
2.2 Velodyne LiDAR

LiDAR je skratka z anglického: light detection and ranging [2]. Niekedy sa tomu hovorí laserové skenovanie, alebo 3D skenovanie. Táto technológia využíva bezpečné laserové lúče na vytvorenie 3D modelu skúmaného prostredia. LiDAR sa používa v mnohých priemyselných odvetviach, vrátane automobilového priemyslu, nákladnej dopravy, bezpilotných lietadiel, priemyslu, mapovania a mnohých ďalších.

Typický LiDAR senzor vysiela pulzy svetla do prostredia. Tieto pulzy sa odrážajú od objektov v prostredí a časť svetla sa vráti do senzoru. Snímač počíta čas potrebný pre pulz svetla, aby sa po odraze vrátil. Na základe času potom vypočíta vzdialenosť, ktorú prešlo svetlo v priestore. Opakovaním tohoto procesu mnohokrát za sekundu dostávame 3D mapu

prostredia v aktuálnom čase [2]. Na obrázku 2.1 je zobrazené mračno bodov vytvorené senzorom Velodyne LiDAR.

Senzor LiDAR zachytáva aj intenzitu svetla ktoré sa vracia do snímača po odraze od objektu. Toto číslo závisí od odrazivosti povrchu na ktorý dopadol lúč. Malé číslo znamená malú odrazivosť zatiaľ čo veľké udáva veľkú odrazivosť povrchu. Intenzita laserového lúča môže byť tiež ovplyvnená uhlom vzhľadom k snímaču, vzdialenosťou, zložením povrchu, drsnosťou povrchu a vlhkosťou vzduchu. To znamená, že body pozdĺž okrajov majú nižšiu intenzitu ako ostatné, pretože sa znižuje energia odrazeného lúča do senzoru. Z týchto dôvodov nemusí byť hodnota intenzity získaná pomocou senzoru LiDAR vždy konzistentná, musí sa použiť ako relatívne meranie. Výhodou je že na rozdiel od pasívnych obrazových senzorov, ako sú napríklad kamery, skenovaním technológiou LiDAR nevznikajú tiene [8].

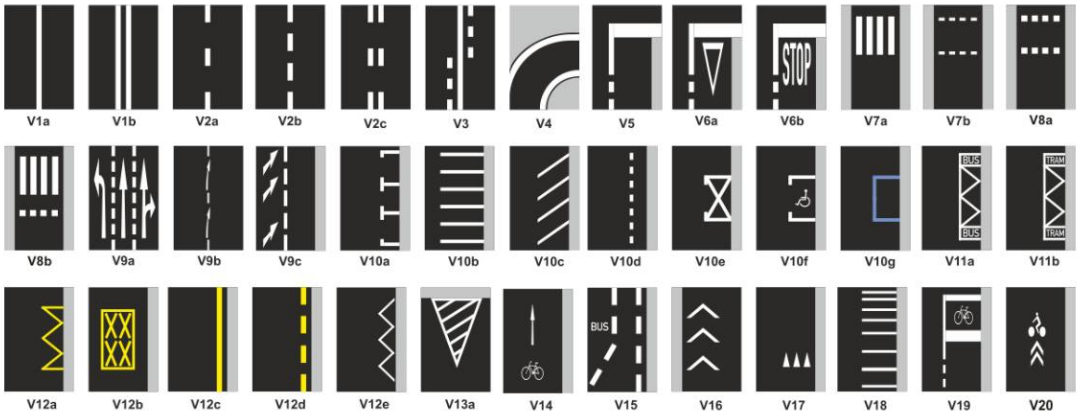


Obr. 2.1: Ukážka snímky dát zo senzoru Velodyne HDL. Technológia Velodyne HDL vytvára 360 stupňové 3D mračno v horizontálnom smere a vo vertikálnom smere je uhol pohľadu 24 stupňov. Vzdialenosti bodov korelujú s intenzitou pixelov. Medzi prekážkami možno vidieť budovy a iné objekty [4].

2.3 Vodorovné dopravné značenie

Z vyhlášky 294/2015 Sb. *úprava provozu na pozemních komunikacích* [18] vyberáme definície a typy vodorovného dopravného značenia v Českej republike. Vodorovné dopravné značky sa vyznačujú na vozovku, alebo inú spevnenú časť pozemnej komunikácie. Dočasná neplatnosť vodorovných dopravných značiek se vyznačuje prešikrtnutými žltými alebo oranžovými čiarami. Vodorovné dopravné značky se rozdeľujú na pozdĺžne čiary, priečne čiary, šípky, označenie státia a parkovísk, označenie zastávok a zákazov zastavenia a státia, os-

tatné vodorovné dopravné značky. Typy vodorovných dopravných značiek môžete vidieť na obrázku 2.2.



Obr. 2.2: Vodorovné dopravné značky používané v Českej republike [18]

2.4 Počítačové videnie

Cieľom tejto práce je vytvoriť program, ktorý bude rozumieť obrázkom vozovky, a bude schopný identifikovať čiaru na vozovke. Jedná sa teda o problém počítačového videnia. Počítačové videnie je oblasť vedeckého výskumu ktorá sa zaoberá spôsobom strojového pochopenia digitálneho obrazu a videa [21]. Rozpoznávanie objektov v obraze je zložitý problém, ktorý každý z nás rieši nevedomky, podvedome, akoby bez námahy. Keď človek vidí obraz, tak využíva predchádzajúce znalosti a skúsenosti na rozpoznanie objektov v obraze. Ľudská schopnosť rozmýšľať umožňuje použiť dlhodobo získavané znalosti na riešenie nových problémov [19]. Umelá inteligencia desaťročia pracovala na tom, aby počítačom poskytla schopnosť porozumieť pozorovaniam, zatiaľ čo pokrok bol obrovský, praktická schopnosť stroja porozumieť pozorovaniam zostáva veľmi obmedzená [19].

Z hľadiska matematickej logiky alebo lingvistiky možno interpretáciu obrazu považovať za mapovanie:

$$\text{interpretácia: } \textit{obrazové dáta} \rightarrow \textit{model}$$

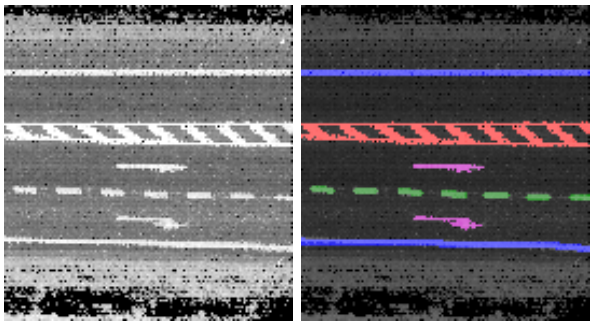
Logický model [19] predstavuje určitý svet, v ktorom pozorovania dávajú zmysel. Ak sú algoritmy rozpoznávania obrazu zamerané na konkrétnu doménu, tak je pozorovaný svet obmedzený. Potom je možné použiť automatickú analýzu aj na komplikované problémy.

2.4.1 Zložitosť počítačového videnia

Aj keď pre človeka je pochopenie obrazu zdanlivo jednoduchá úloha, v skutočnosti ide o veľmi zložitý proces. Počítačové videnie je zložité čo sa týka pamäťovej a výpočtovej náročnosti, taktiež je zložité navrhnúť komplexné systémy na rozpoznávanie obrazu. Základné dôvody [19] prečo je počítačové videnie také náročné, sú nasledujúce:

- **Šum** – je neodmysliteľnou súčasťou každého merania. Jeho existencia si vyžaduje matematické nástroje, ktoré sú schopné pracovať s neistotou, príkladom je teória pravdepodobnosti. Zložitejšie nástroje samozrejme komplikujú analýzu obrazu v porovnaní so štandardnými deterministickými metódami.
- **Veľkosť dát.** – Obrázky sú veľké. Vďaka technickému pokroku sú požiadavky na procesor a pamäť oveľa menším problémom, ako boli kedysi, a veľa je možné vyriešiť bežne dostupnými výpočetnými prostriedkami. Efektívnosť riešenia problémov je však stále dôležitá a veľa aplikácií stále nie je použiteľných v interaktívnom režime.
- **Zachytená intenzita** – v obraze je daná komplikovanou fyzikou formovania obrazu. Intenzita obrazu závisí od typu zdroja svetla, intenzity a polohy svetla, polohy pozorovateľa, miestnej geometrie povrchu a vlastností povrchu.
- **Lokálne okno a globálny pohľad.** – Algoritmy na analýzu obrazu bežne analyzujú konkrétny bod uložený v operačnej pamäti (napr. Pixel v obraze) a jeho bezprostredné okolie. Počítač vidí obraz akoby cez kľúčovú dierku, a to sťažuje pochopenie globálnejších súvislostí.

2.4.2 Segmentácia obrazu



Obr. 2.3: Per pixel segmentácia obrázku vozovky

Segmentáciu zaraďujeme k metódam počítačového videnia a v tejto práci je použitá na označovanie bodov čiary na vozovke. Segmentácia je proces hľadania skupín pixelov, ktoré patria do jednej triedy [21]. Segmentácia vychádza zo štatistiky, kde existuje podobný problém – zhluková analýza t.j. priradenie podobných objektov do rovnakej triedy. Segmentácia pixelom, ktoré spolu súvisia, a teda patria do rovnakej triedy, priradí rovnaké označenie. Segment, alebo superpixel je zhluk pixelov s rovnakým označením. Cieľom segmentácie je rozdeliť obraz do disjunktných segmentov, ktoré pokrývajú celý obraz. Pomocou segmentov je možné ďalej jednoduchšie spracovať časti obrazu a agregovať podstatné informácie. Obrázok 2.3 zobrazuje segmentáciu vodorovného dopravného značenia z obrázku vozovky.

2.5 Problémy detekcie dopravného značenia z MLS

Tri hlavné problémy pri detekcii dopravného značenia z mračna bodov vytvoreného mobilným skenovaním vozovky sú nasledujúce:

2.5.1 Rozdiely intenzity a hustoty bodov

Mračná bodov sú zvyčajne vytvorené pomocou MLS systémov. Mračno je vytvorené tak, že skenovacie vozidlo udržuje určitý jazdný pruh a vytvára mračno bodov. Výsledné mračno bodov vznikne spojením mračien z viacerých jazdných pruhov. Typicky u mračien bodov vytvorených pomocou MLS nastáva, že hustota a intenzita bodov klesá so vzdialenosťou od skeneru. Samotné použitie technológie LiDAR spôsobuje, že vodorovné dopravné značenie bližšie k trajektórii vozidla má vyššiu intenzitu a hustotu bodov. Napríklad keď sú na vozovke dva pruhy a skenovacie vozidlo prešlo len niektorým z nich, tak rozdiel v intenzite môže byť 20% a v hustote bodov až 50% [22]. V takom prípade klasické metódy založené na prahovaní nemajú dobré výsledky.

2.5.2 Malý kontrast medzi značením a zbytkom vozovky

Veľmi často sa stáva, že vodorovné dopravné značenie je zodraté prípadne vyblednuté, a preto je málo viditeľné na vozovke. Zodraté značenie má zvyčajne nižšiu schopnosť odrážať svetlo než nové značenie. Preto v MLS mračnách bodov nastáva problém, že intenzita bodov značenia je veľmi podobná intenzite bodov vozovky t.j. malý kontrast značenia. Toto je výrazný problém pre niektoré techniky detekcie vodorovného značenia.

2.5.3 Nekonzistencia detekcie vodorovného značenia

Vplyvom rozdielov v intenzite bodov a malého kontrastu medzi značením a vozovkou sa môže stať, že niektoré časti značenia nie sú identifikované. Príkladom je detekcia súvislej vodiacej čiary na kraji vozovky, kde metóda vynechá určité miesta. Výsledkom metódy teda nie je súvislá čiara, ako by sme očakávali. Niektoré aplikácie vyžadujú presnú detekciu značenia bez vynechaných miest, príkladom sú mapy pre autonómne vozidlá. Avšak niektorým chybám detekcie sa vplyvom prekážok v zábere nedá úplne vyhnúť, a preto je potrebná manuálna kontrola detekcie. Bolo by vhodné teda vytvoriť systém, ktorý by bol schopný doplniť chýbajúce značenie podľa typu a pozície okolitého značenia. V článku [22] preto vytvorili neurónovú sieť cGAN, ktorá doplňa chýbajúce značenie.

Kapitola 3

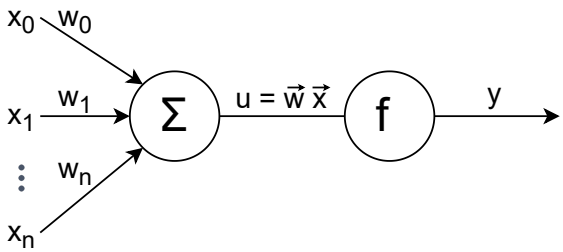
Umelé neurónové siete

Umelé neuronové siete (ANN – Artificial Neural Network) zaraďujeme do oblasti strojového učenia, to znamená že ide o algoritmus, ktorý sa automaticky zlepšuje na základe skúseností [14]. Algoritmy strojového učenia vytvárajú model založený na vzorových dátach, ktoré sa označujú ako trénovacia databáza. Cieľom je teda vytvoriť model bez toho, aby ho musel programátor explicitne naprogramovať. Model sa vytvára z trénovacej databázy, tak aby čo najlepšie pasoval na zadané dáta.

Základom umelých neurónových sietí je neurón, ktorý môžeme považovať za výrazne zjednodušený model biologického neurónu. Biologické neuróny sú veľmi špecializované nervové bunky schopné prijímať, spracovať a odpovedať na elektrické signály. Neuróny umožňujú organizmom spracovávať informácie z okolitého prostredia a reagovať na ne.

3.1 Model neurónu

Jedná sa o zjednodušený model biologického neurónu [14]. Model neurónu slúži ako základný stavebný blok neurónových sietí. Model neurónu obsahuje bázovú a aktivačnú funkciu. Najčastejšie sa ako bázová funkcia používa suma, ale aktivačné funkcie bývajú rôzne, záleží od aplikácie. V obrázku 3.1 je aktivačná funkcia znázornená písmenom f.



Obr. 3.1: Všeobecný model neurónu

Neurón prijíma vektor vstupov, kde každému vstupu je priradená váha, a potom vykonáva súčet všetkých vážených vstupov. Na základe tohto súčtu potom pomocou aktivačnej funkcie rozhodne akú hodnotu pošle na výstup. Funkciu neurónu popisuje vzťah 3.1, kde N je dimenzionalita vstupu, x je vstupný vektor a w je vektor váh. Výsledkom je vnútorný potenciál neurónu u.

$$u = \sum_{i=1}^{N} w_i x_i = \overrightarrow{w} * \overrightarrow{x} \tag{3.1}$$

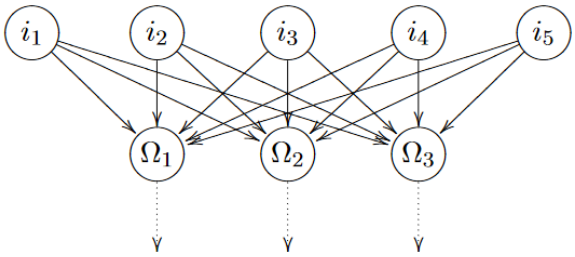
Hodnota vnútorného potenciálu u je použitá ako vstup aktivačnej funkcie, môže ísť napríklad o funkciu *sigmoid*, alebo skokovú funkciu [12]. Aktivačná funkcia potom vypočíta výstupnú hodnotu neurónu. Výstup neurónu môže byť pripojený ako vstup iných neurónov v neurónovej sieti. Hodnota výstupu sa odošle na všetky výstupy, a potom pokračuje spracovanie v neurónoch zapojených za aktuálny neurón.

Jeden neurón je schopný rozdeliť N-dimenzionálny priestor vstupných vektorov na dve časti pomocou takzvanej *hyper-roviny*. Táto vlastnosť je vhodná na vytvorenie klasifikátoru, teda modelu ktorý na základe vlastností objektu určuje, do ktorej triedy daný objekt patrí. Pre dvoj-dimenzionálny priestor to je priamka, ktorá rozdeľuje priestor na dve časti. Váhy vstupov neurónov potom tvoria koeficienty tejto priamky. Na to aby bolo možné rozdeliť priestor zložitejším spôsobom je potrebné použiť viac neurónov zapojených za sebou do siete.

3.2 Perceptronová sieť

Perceptronová sieť [12] je dopredná neurónová sieť, ktorá obsahuje jednu vstupnú vrstvu neurónov ktorým je pevne priradená váha, ale váhy všetkých ostatných vrstiev je možné zmeniť. Všetky neuróny nasledujúce za vstupnou vrstvou sú schopné rozpoznávať vzory. Binárny perceptron používa dve hodnoty ako výstup napríklad $\{0, 1\}$ alebo $\{-1, 1\}$, to znamená že používa binárnu prahovú funkciu ako aktivačnú funkciu. Funkcia určuje na základe prahu výstup neurónu.

Skokovú aktivačného funkciu je možné interpretovať ako podmienku z binárnej logiky, v ktorej je možné vyjadriť aj negáciu prostredníctvom záporných váh. Z toho vyplýva, že je možné použiť binárny perceptron na spracovanie informácií podobne, ako obvod spracováva binárne dáta.



Obr. 3.2: Jednovrstvová perceptronová sieť [12]

Na učenie jednovrstvovej perceptronovej siete existuje jednoduchý algoritmus, ktorý pre každý chybný výstup mierne upraví váhy výstupného neurónu a postupne konverguje k takému nastaveniu váh, aby oddelil body do dvoch tried rovinou. V prípade, že body dvoch tried nie je možné oddeliť rovinou, tak algoritmus nikdy neskončí, a bude sa stále snažiť vylepšovať riešenie. Algoritmus učenia perceptronu je možné zobecniť aj pre viacvrstvové siete.

Originálny algoritmus na učenie jednovrstvovej siete percentron je popísaný nižšie. Bolo dokázané, že algoritmus pre oddeliteľné triedy konverguje v konečnom čase. To znamená, že v konečnom čase sa perceptron naučí všetko, čo je schopný reprezentovať [12].

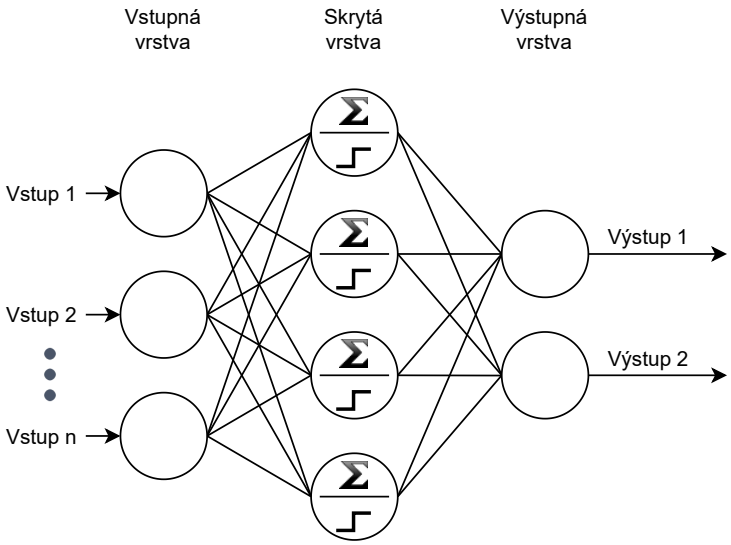
Význam značiek použitých v algoritme:

- x – Vstupný vektor

- X – Trénovacia množina
- y – Výstupný vektor
- Ω_1 – Prvý neurón výstupnej vrstvy
- i – Vstupný neurón
- o – Výstupný neurón

Algoritmus 1: Algoritmus úpravy váh perceptronu

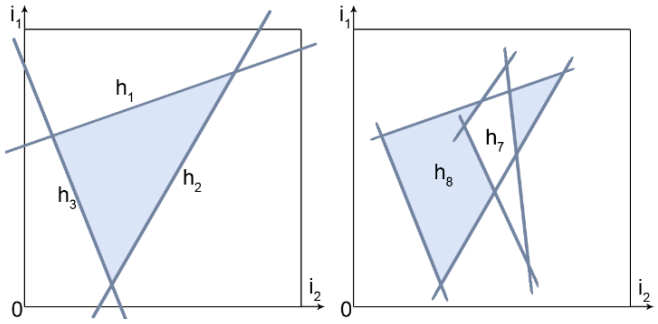
```
X ← Množina trénovacích vzorov;
while ∃x ∈ X a error je príliš veľký do
    y ← {Prilož vektor x na vstup siete a vypočítaj výstup siete};
    for všetky výstupné neuróny Ω do
        if y_Ω = 0 then
            for všetky vstupné neuróny i do
                w_iΩ = w_iΩ − o_i {Zvýšenie váhy smerom k neurónu Ω o výstup
                 neurónu i}
            end
        else
            for všetky vstupné neuróny i do
                w_iΩ = w_iΩ + o_i {Zníženie váhy smerom k neurónu Ω o výstup
                 neurónu i}
            end
        end
    end
end
```



Obr. 3.3: Príklad perceptronovej siete s jednou skrytou vrstvou a skokovou aktivačnou funkciou

Viacvrstvové siete sú schopné riešiť problémy, ktoré nie sú jednoducho oddeliteľné hyper-rovinou. To znamená, že zadané trénovacie dátové body nevieme oddeliť hyper-rovinou. Na obrázku 3.3 vidíme viacvrstvovú sieť perceptron, kde krúžok zobrazuje jeden neurón a spoje vyjadrujú váhy spojení. Skrytá vrstva obsahuje symbol Σ, to znamená že prebieha súčet váh spojení pre daný neurón, a potom je aplikovaná skoková prechodová funkcia.

Ako je spomenuté v časti 3.1, jednovrstvový percentron reprezentuje priamku. Pri dvoch trénovateľných vrstvách je možné vytvoriť viacero priamych čiar, ktoré ohraničujú polygón, alebo pri použití troch trénovateľných vrstiev je možné vytvoriť viacero prekrývajúcich sa polygónov 3.4. Nevýhodou perceptronu je, že potrebuje triedy, ktoré sú od seba oddeliteľné. Pokiaľ bude existovať trénovací vzor, ktorý nie je správne klasifikovaný, tak sa učenie nikdy neskončí.



Obr. 3.4: Príklad ako môže viacvrstvový perceptron rozdeliť priestor [12]

3.3 Spätné šírenie chyby

Aby bolo možné efektívne trénovať dopredné neuronové siete, bol vytvorený algoritmus spätného šírenia chyby [16]. Cieľom je stanoviť váhy siete tak, aby bola minimalizovaná chyba. Algoritmus spätného šírenia chyby, anglicky *backpropagation* je najznámejším a najpoužívanejším algoritmom trénovania acyklických dopredných neurónových sietí. Jedinou podmienkou pre jeho použitie je to, že všetky neuróny siete musia mať diferencovateľné aktivačné funkcie. Cieľom algoritmu spätného šírenia chyby je minimalizovať objektívnu funkciu, ktorá je funkciou všetkých váh, a ktorá vyjadruje odchýlku odozvy siete od požadovaných hodnôt.

Pôvodný algoritmus *backpropagation* [16] používa ako objektívnu funkciu polovicu súčtu kvadratických chýb, ktorá je pre m vstupných neurónov a jeden prvok p trénovacej množiny daná vzťahom 3.2, kde d_{pj} je očakávaný výstup výstupného neurónu, j a o_{pj} je skutočný výstup.

$$E_p = \frac{1}{2}\sum_{j=1}^{m}(d_{pj} - o_{pj})^2 \tag{3.2}$$

Túto chybu potom minimalizuje metódou gradientného zostupu 3.3, kde μ je parameter učenia a ∇ je symbol pre gradient. Gradient chyby ∇E_p je vektor parciálnych derivácií podľa jednotlivých dimenzií a vyjadruje smer najrýchlejšieho nárastu chyby.

$$\Delta\vec{w} = -\mu\nabla E_p \tag{3.3}$$

Pre požadovanú zmenu ľubovoľnej váhy w_{ji}, tj. váha i-tého vstupu j-tého neurónu potom platí 3.5. Kde u_j je vnútorný potenciál neurónu j a x_i je i-ty vstup neurónu.

$$\delta_j = -\frac{\partial E_p}{\partial u_j} = -\frac{\partial E_p}{\partial y_j} * \frac{\partial y_j}{\partial u_j} \tag{3.4}$$

$$\Delta w_{ji} = \mu \delta_j x_i \tag{3.5}$$

Ako vidíme z predchádzajúcich rovníc, veľkosť zmeny váhy je proporcionálna k veľkosti chyby a veľkosti daného vstupu. To znamená, že najviac sa zmenia váhy, ktoré majú najväčší vplyv na zmenu výsledku. Metóda spätného šírenia chyby počíta hodnoty δ_j spätným postupom. To znamená, že najprv vypočíta hodnoty pre poslednú vrstvu, ako rozdiel od očakávaného výsledku. Potom chybu propaguje do predchádzajúcej vrstvy a tak postupuje až k neurónom prvej vrstvy.

Trénovanie siete potom prebieha takto: na začiatku sú váhy inicializované náhodne potom sa na vstup priloží trénovací vektor a vypočíta sa výstup. To sa vykoná pre všetky dáta trénovacej sady, potom sa vypočíta chyba neurónovej siete a propaguje sa naspäť pomocou algoritmu spätného šírenia chyby. Proces sa opakuje, až dokým nie je dosiahnutá ukončujúca podmienka.

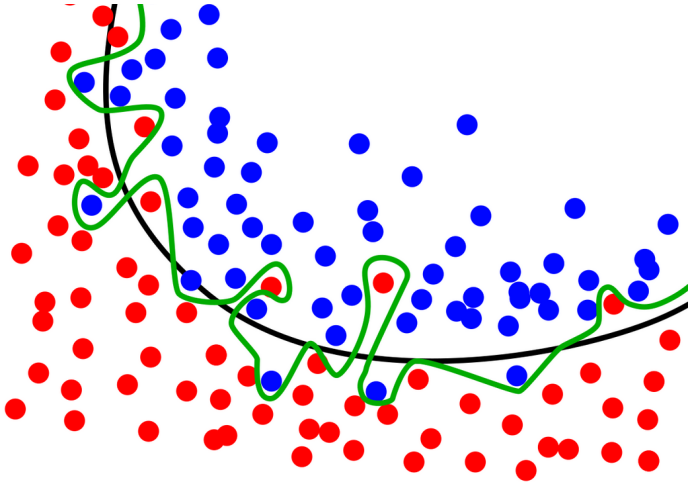
V praxi sa ale používa upravená varianta algoritmu, kde netreba počítať hodnotu objektívnej funkcie zo všetkých trénovacích vzorov. To by bol príliš pomalý algoritmus na to, aby sa dal prakticky použiť. Preto sa objektívna funkcia počíta len z malej podmnožiny všetkých trénovacích vzorov, táto podmnožina sa nazýva *mini-batch*. Výsledná hodnota objektívnej funkcie je potom menej presná, ale výpočet bol výrazne rýchlejší. Môžeme si to predstaviť tak, že sieť miesto pomalých, ale presne vypočítaných krokov úpravy váh, používa rýchlejšie, ale menej presné kroky v približne správnom smere. Celkovo teda bude potrebné viackrát vykonať úpravu váh (viacej krokov), ale výpočet bude rýchlejší, lebo sa ušetril čas tým, že sa počítala iba približná chyba siete. Táto metóda sa nazýva *stochastic gradient descent*.

3.3.1 Problém s pretrénovaním

Pretrénovanie (anglicky overfitting) nastáva, keď sa model príliš prispôsobí trénovacej sade dát a nie je schopný generalizovať, alebo predikovať budúce pozorovania [1]. Model vytvorí veľmi komplexnú hranicu, tak aby čo najlepšie oddelil body z rôznych tried. Môže dosiahnuť prakticky 100% presnosť na trénovacích dátach, ako je vidieť na obrázku 3.5. Keď ale takýto model spustíme na testovacej množine dát, tak bude mať výrazne horšie výsledky, než jednoduchší model.

Aby sa predišlo pretrénovaniu, tak je možné použiť jednoduchší model s menším množstvom parametrov, alebo zastaviť učenie skôr, než sa model pretrénuje. Včasné zastavenie učenia je spôsob, keď priebežne po niekoľkých krokoch učenia nastane testovanie modelu na dátach, ktoré sa nenachádzajú v trénovacej množine [20]. Pokiaľ sa model prestane zlepšovať na testovacích dátach, tak sa učenie zastaví, aby sa predišlo pretrénovaniu.

Problém pretrénovania je možné tiež riešiť pomocou techniky zvanej *dropout* [20]. Tento termín vyjadruje, že niektoré časti siete sú dočasne odstránené. Dočasným vyhodením jednotky zo siete, odstránime tiež všetky vstupné aj výstupné spoje danej jednotky. Výber jednotky, ktorá bude odstránená je v najjednoduchšom prípade náhodný, s určitou fixnou pravdepodobnosťou sa daná jednotka zo siete odstráni. Technika dropout výrazne znižuje pravdepodobnosť pretrénovania hlbokých neurónových sietí s veľkým množstvom paramet-



Obr. 3.5: Porovnanie pretrénovaného modelu (zelená rozhodovacia hranica) a modelu, ktorý správne generalizuje (čierna rozhodovacia hranica) [23]

rov. Nevýhodou techniky je, že trénovací čas sa zväčší dvoj až trojnásobne v porovnaní so sieťou, ktorá nepoužíva dropout.

Pretrénovanie môže nastať aj tým, že trénovacia sada dát je príliš malá, a preto model nieje schopný nájsť správne vzory, ktoré odlišujú dáta do požadovaných tried. V takom prípade je potrebné rozšíriť trénovaciu databázu, aby obsahovala viac dátových bodov. Keď nie je možné získať viac skutočných dát, tak je možné použiť techniky na vytvorenie umelých dátových bodov podľa bodov v databáze. Tieto techniky sa nazývajú *data augmentation*. Nové dáta sú vytvorené pridaním šumu, deformovaním vstupov, alebo aplikovaním invariantných transformácií.

3.4 Konvolučné neurónové siete

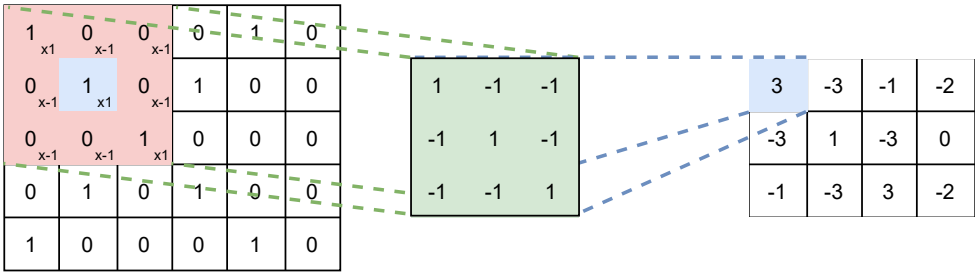
Konvolučné neurónové siete (CNN) [15] sú analogické s tradičnými neurónovými sieťami v tom, že pozostávajú z neurónov, ktoré sa samy optimalizujú učením. Každý neurón dostane vstup a vykoná operáciu, rovnako ako tomu je u všeobecných ANN sietí. Jediný významný rozdiel medzi CNN a ANN sieťami je ten, že CNN sú optimalizované na použitie v oblasti rozpoznávania vzorov v obrazoch. CNN má optimálnejší počet parametrov preto, že neuróny rovnakej konvolučnej vrstvy majú zdieľané váhy, t.j. jedno konvolučné jadro. Konvolučné siete využívajú špecifické vlastnosti obrazu. Optimalizácia siete znižuje počet potrebných parametrov, a tak aj pravdepodobnosť pretrénovania.

Architektúra CNN pozostáva zo štyroch základných typov vrstiev. Tieto vrstvy sú: konvolučná vrstva, pooling vrstva (podvzorkovanie), aktivačná vrstva a plne prepojená vrstva.

3.4.1 Konvolučná vrstva

Ako názov napovedá, táto vrstva je základom CNN. Parametre vrstvy sú zamerané na použitie konvolučných jadier, ktorých parametre sa sieť učí na základe trénovacej množiny. Konvolučná vrstva využíva konvolučné jadro, vstupný obrázok a operáciu konvolúcie na výpočet novej hodnoty pre daný pixel v novom obrázku. Konvolučné jadro môžeme považovať

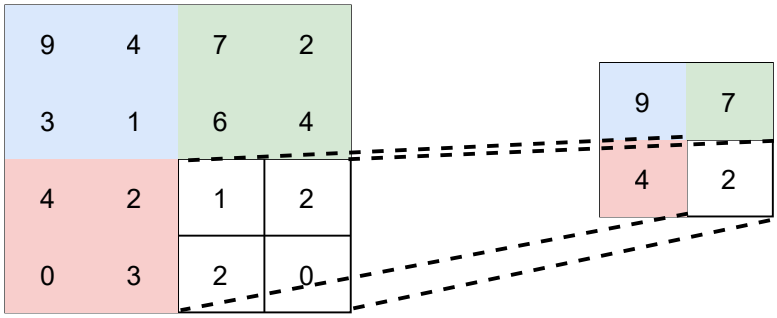
za štvorcovú maticu váh, kde váhy určujú vplyv okolitých pixelov na novú hodnotu pixelu. Konvolúcia funguje tak, že priloží konvolučné jadro na určitú pozíciu vstupného obrázku, potom vynásobí hodnotu zo vstupného obrázku príslušnou hodnotou konvolučného jadra a výsledkom je suma všetkých týchto hodnôt, viď obrázok 3.6.



Obr. 3.6: Príklad konvolúcie: na ľavej strane je pôvodný obrázok, uprostred konvolučné jadro a napravo je výstupný menši obrázok. Zmenšenie nastalo, lebo konvolúciu nebolo možné aplikovať na okrajové časti pôvodného obrázku. Aby rozmery výsledného obrázku boli rovnaké ako pôvodný obrázok, tak je možné použiť tzv. padding, ktorý okolie pôvodného obrázku doplní nulami.

3.4.2 Vrstva na podvzorkovanie

Cieľom tzv. pooling vrstvy je postupne znižovať dimenzionalitu dát, a tým ďalej znižovať počet parametrov a výpočtovú zložitosť modelu. Pooling vrstva je zvyčajne umiestnená za konvolučnou vrstvou a znižuje dimenzionalitu pomocou nejakej funkcie. Väčšinou sa používa funkcia maximum, potom sa taká vrstv nazýva *max-pooling layer*. Pri použití jadier o rozmeroch 2×2 s odstupom 2 nastane redukcia obrázku na 25% pôvodnej veľkosti. Môžeme si to predstaviť tak, že štyri susedné body obrázku zlúčime do jedného tak, že vyberieme pixel s maximálnou hodnotou. Použitie väčšieho jadra ako 2 x 2 sa nedoporučuje, z dôvodu veľkej straty informácie.

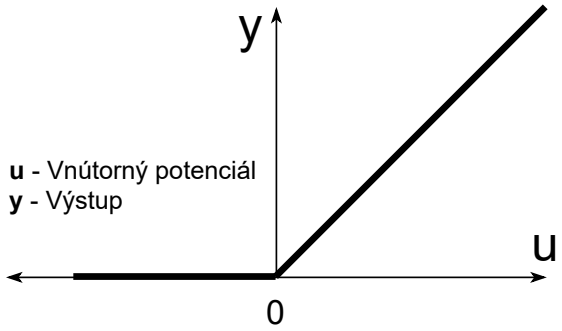


Obr. 3.7: Podvzorkovanie obrazu filtrom o veľkosti 2x2 pixely. Konkrétne ide o Max-pooling, t.j. filter vyberie maximum z každého okna.

3.4.3 Aktivačná vrstva

Filter v konvolučnej vrstve si možno ľahko predstaviť ako bázovú funkciu neurónu. Aktivačná vrstva je vrstva aktivačných funkcií. Aktivačná funkcia pracuje rovnako ako tomu

bolo v modele neurónu. Obvykle aktivačné funkcie nebývajú lineárne, takmer výlučne sa používa funkcia ReLU (Rectified Linear Unit), ktorá je zobrazená na obrázku 3.8.



Obr. 3.8: Aktivačná funkcia ReLU určuje závislosť výstupu y od vnútorného potenciálu neurónu u

3.4.4 Plne prepojená vrstva

Plne prepojená vrstva obsahuje neuróny, ktoré sú priamo spojené s neurónmi v dvoch susedných vrstvách bez toho, aby boli spojené s akýmikoľvek neurónom v ich vrstve. Každý neurón predchádzajúcej vrstvy je pripojený na vstup každého neurónu nasledujúcej vrstvy neurónov. Ide teda o prepojenie každého neurónu s každým, a preto sa to volá plne prepojená vrstva.

Kapitola 4

Stav technológií rozpoznávania VDZ

Cieľom tejto kapitoly je objasniť rôzne technológie a prístupy používané na rozpoznávanie vodorovného dopravného značenia. Štúdium podobných metód z odbornej literatúry je veľmi prínosné, lebo na ich základe bol vytvorený návrh nášho riešenia.

4.1 Metódy využívajúce kamerový záznam

Mnoho odborných článkov sa zaoberá detekciou vodorovného dopravného značenia na vozovke s využitím kamerového záznamu, alebo statických obrázkov. Tieto metódy často používajú konvolučné neurónové siete, všeobecný popis takýchto sietí je v kapitole 3. Obraz vozovky je vytvorený videokamerou, ktorá zachytáva svetlo z okolia. Formát vstupu niektorých metód je odlišný od nášho vstupu, očakáva sa obraz vytvorený kamerou a nie mračno bodov. Avšak mračno bodov je možné previesť na čiernobiely obrázok, a tak by sme mohli určité metódy na spracovanie obrazu využiť v našom programe.

Jedným spôsobom detekcie čiar je časopriestorové zhlukovanie [11], ktoré bolo použité na detekciu jazdných pruhov a čiar na ceste s využitím kamerového záznamu vytvoreného videokamerou umiestnenou na palubnej doske vozidla. Cieľom je v reálnom čase vytvoriť vektorovú reprezentáciu čiar na vozovke. Na spracovanie obrazu vytvorili algoritmus časopriestorového zhlukovania, ktorý vytvára zhluky obrazových bodov a v čase ich doplňuje o nové body. Ide o učenie bez učiteľa, to znamená že neexistuje trénovacia množina dát s vyznačenými čiarami. Výsledné zhluky potom prevedú na vektorovú reprezentáciu preložením krivky.

Zložitejší algoritmus [5] využíva kamerový záznam z dvoch kamier, pričom každá má inú pozíciu a uhol pohľadu. Pohľady z kamier potom transformuje na pohľad zhora. Inverzná perspektívna transformácia je zobrazená na obrázku 4.1. Takto transformované pohľady tvoria vstup konvolučnej neurónovej siete DVCNN (Dual-View Convolutional Neural Network). Neurónová sieť pozostáva z dvoch konvolučných neurónových sietí, pre každý pohľad jedna, ktoré sú prepojené pomocou plne prepojenej vrstvy. Výstupom siete je odhad pozície čiar a ich pravdepodobnosť. Na záver prebehne globálna optimalizácia, ktorá vyberie len najpravdepodobnejšie čiary. Použitie dvoch pohľadov prinieslo kvalitnejšie výsledky, lebo jeden pohľad bol zameraný na čiary blízko vozidla a druhý bol pohľad do diaľky.

Obr. 4.1: Príklad inverznej perspektívnej transformácie obrazu z prednej kamery. Na výslednom pohľade zhora je možné vidieť, že čiary sú rovnobežné, ale okoloidúce vozidlá sú deformované.

Algoritmus na detekciu značiek a nápisov na ceste [24] je založený na porovnávaní aktuálneho obrázku so vzorom. Výsledok porovnania je vektor príznakov podľa ktorého je možné určiť podobnosť so vzorom.

4.2 Metódy založené na MLS

Nevýhodou všetkých techník založených na snímaní vozovky pomocou kamery, ktorá sníma obraz okolia, je vplyv svetelných podmienok na kvalitu výslednej detekcie. Tiene, horšia viditeľnosť, prípadne slnko nízko nad obzorom výrazne zhoršujú kvalitu detekcie. Ďalšia zásadná nevýhoda je, že kamerový záznam neposkytuje presnú pozíciu bodov v priestore. Preto je výrazne zložitejšie previesť kamerový záznam na vektorovú mapu v geografických súradniciach.

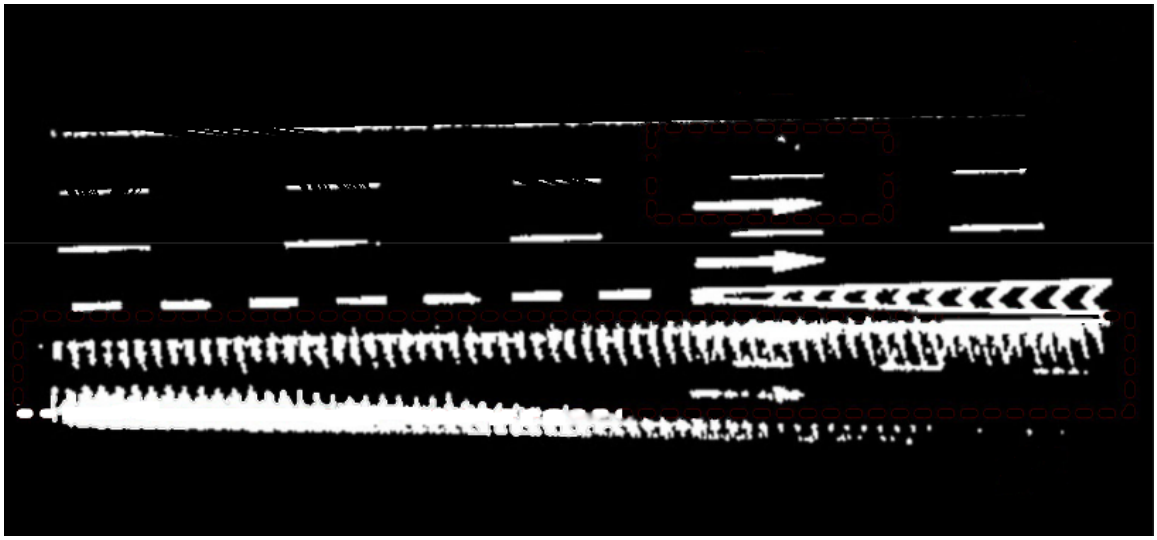
Problémy kamerového snímania je možné odstrániť použitím skeneru Velodyne LiDAR, ktorý vytvára mračno bodov s presnou pozíciou. Dáta vytvorené laserovým snímaním sú výrazne menej poznačené aktuálnymi svetelnými podmienkami ako kamerové záznamy. Navyše pri takomto skenovaní nevznikajú tiene od objektov blízko vozovky. Metódy založené na mobilnom laserovom skenovaní väčšinou pracujú v troch fázach.

1. **Extrakcia vozovky** – Na základe vlastností tvaru vozovky sa algoritmicky určí, ktoré body patria vozovke a ktoré nie

2. **Extrakcia dopravného značenia** – Výsledkom je čiernobiely obrázok, kde čierna znamená, že tam nie je značenie a biela znamená, že tam je značenie

3. **Vytvorenie finálnej reprezentácie** – Doplnenie chýbajúcich bodov, prípadne výpočet vektorovej reprezentácie

Najjednoduchšou metódou na extrakciu dopravného značenia z mračna bodov je použiť globálnu prahovaciu metódu [7]. Obecne platí, že vodorovné dopravné značenie odrazí viac svetla ako povrch vozovky. Táto metóda na základe intenzity odrazeného lúča LiDAR skeneru určuje, či sa na danom mieste nachádza dopravné značenie. Metóda podľa určitej hodnoty prahu rozdelí body do dvoch skupín. Body z nižšou intenzitou nereprezentujú dopravné značenie, naopak body z vyššou intenzitou patria k dopravnému značeniu. V skutočnosti môže ale intenzita kolísať podľa vzdialenosti objektu od skeneru a uhlu vzhľadom



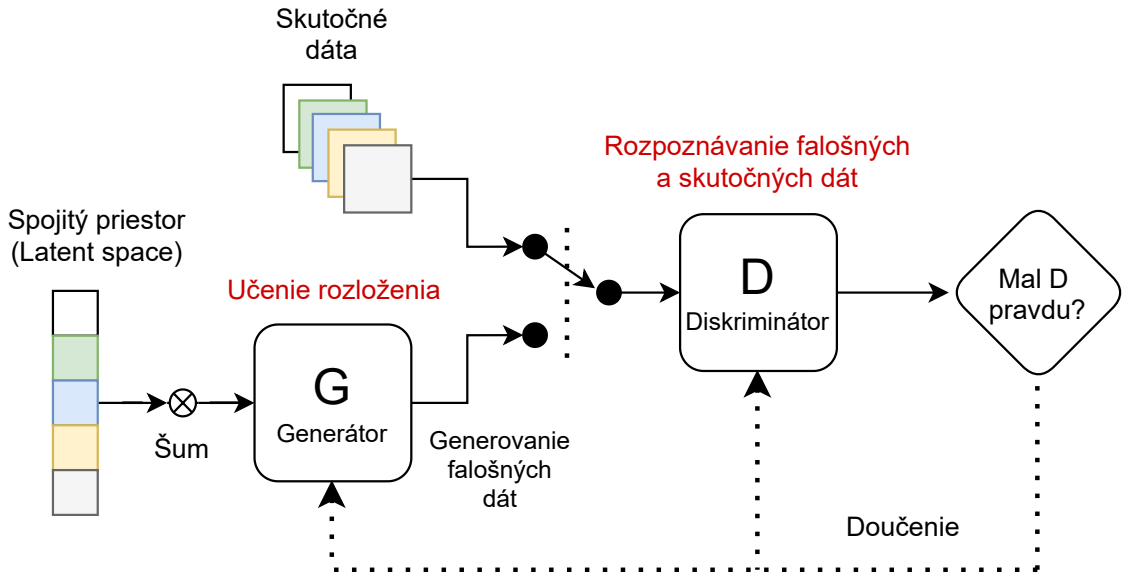
Obr. 4.2: Problém globálnej prahovacej funkcie, napriek tomu, že hodnota prahu je zvolená správne, tak niektoré body sú klasifikované nesprávne [25].

ku skeneru. Avšak táto metóda môže zlyhať aj keď je mračno bodov kvalitné, príklad zlyhania je na obrázku 4.2.

Metóda [13] využíva prahovanie založené na vzdialenosti. To znamená, že hodnota prahu, ktorý oddeľuje body dopravného značenia od ostatných, klesá so vzdialenosťou od skeneru. To znamená že hodnota intenzity je normalizovaná a nezávisí od vzdialenosti ku skeneru. Tento algoritmus vytvára dva typy obrázkov. Prvý obrázok obsahuje intenzitu a druhý zachytáva vzdialenosť. Po vykonaní prahovania aplikuje morfologické operácie, ktoré využívajú známe vlastnosti o dopravnom značení. Tieto operácie odstránia šum a doplnia chýbajúce miesta značenia. Nevýhodou je, že normalizačné hodnoty sa menia podľa typu scény. Globálne prahovacie metódy nemusia pracovať vždy správne, hlavne ak odrazivosť bodov vozovky má neočakávané rozloženie.

Existujú aj metódy založené na lokálnom prahovaní [10], kde sú body rozdelené do skupín a prah sa počíta dynamicky pre každú skupinu. Nevýhodou týchto metód je citlivosť na šum a kvalitu vstupných dát, problém je tiež že metódy sú špeciálne určené na detekciu vodorovného dopravného značenia a nie je možné ich upraviť tak, aby detekovali iné objekty v okolí cesty ako napríklad zvodidlá, chodníky, alebo trávnaté plochy.

Problém ako oddeliť body dopravného značenia a body vozovky rieši metóda [22] pomocou konvolučnej neurónovej siete U-Net. Táto neurónová sieť je schopná vytvoriť segmentáciu na úrovni pixelu. To znamená že pre každý pixel určí do akej triedy patrí. Názov siete vychádza z jej topológie, ktorá pripomína písmeno U. Sieť U-Net sa delí na dve časti, kodér a dekodér. Pri kódovaní nastáva znižovanie rozlíšenia obrázku a prebieha extrakcia príznakov t.j. objektov ktoré sa na obrázku nachádzajú. Pri prechode dekódovaciou časťou sa tieto rysy použijú na zvýšenie rozlíšenia obrázku. Viac informácií o sieti U-Net bude uvedených v časti 4.3. Výhoda tohoto prístupu je, že sieť je schopná okrem rozpoznania intenzity značenia, rozpoznať aj konkrétny typ značenia na základe hrúbky či tvaru značky. Sieť U-Net pre každý pixel určí triedu ku ktorej daný pixel patri. Veľmi zaujímavá časť článku [22] je tiež použitie siete cGAN (conditional generative adversarial network) na doplnenie chýbajúcich častí značenia na základe značenia, ktoré bolo správne označené. Sieť cGAN vychádza zo siete GAN, ktorá je znázornená na obrázku 4.3. Táto metóda je teda



Obr. 4.3: Základnou myšlienkou sietí GAN [9] je vytvoriť hru medzi dvoma hráčmi. Jeden z nich sa nazýva generátor. Generátor vytvára vzorky, ktoré majú pochádzať z rovnakého rozloženia, ako trénovacie dáta. Druhý hráč je diskriminátor. Diskriminátor skúma vzorky, aby určil, či sú skutočné alebo falošné. Diskriminátor sa učí pomocou tradičných metód učenia s učiteľom, pričom vstupy rozdeľuje na dve triedy (skutočné a falošné). Cieľom generátoru je teda oklamať diskriminátor. Môžeme si predstaviť, že generátor je falšovateľ, ktorý falšuje peniaze, a diskriminátor je policajt, ktorý sa snaží odhaliť falošné peniaze a prepustiť legitímne peniaze. Aby bol falšovateľ úspešný, tak sa musí naučiť vytvárať peniaze, ktoré sú neodlíšiteľné od pravých peňazí. Analogicky sa generátorová sieť musí naučiť vytvárať vzorky, ktoré sú z rovnakého rozloženia, ako trénovacie dáta.

schopná veľmi presne klasifikovať rôzne objekty vozovky a prípadne doplniť ich očakávaný tvar. Je veľmi robustná vzhľadom k šumu, artefaktom a zodratému značeniu na vozovke.

4.3 U-Net

U-Net je konvolučná neurónová sieť určená pre segmentáciu biomedicínskych obrázkov vyvinutá na katedre informatiky na univerzite vo Freiburgu v Nemecku [17]. Táto sieť bola tiež použitá na detekciu vodorovného dopravného značenia v práci, ktorou sme sa inšpirovali [22]. Sieť U-Net pozostáva z kontraktačnej cesty a expanzívnej cesty, ktoré jej dávajú architektúru v tvare písmena U. Kontraktačná cesta je typická konvolučná sieť. Pozostáva z opakovanej aplikácie konvolúcií, striedajú sa tu konvolučné a max-pooling vrstvy. Počas kontrakcie sa priestorová informácia zmenšuje, zatiaľ čo vektor príznakov sa zväčšuje. Expanzívna cesta kombinuje priestorové informácie a vektor príznakov prostredníctvom sekvencie vrstiev zvýšenia rozlíšenia obrázku. Kontraktívna cesta využíva pooling operátor a expanzívna cesta používa opačný tzv. upsampling operátor. Upsampling (up-conv) operátor je zložený z nasledujúcich dvoch častí:

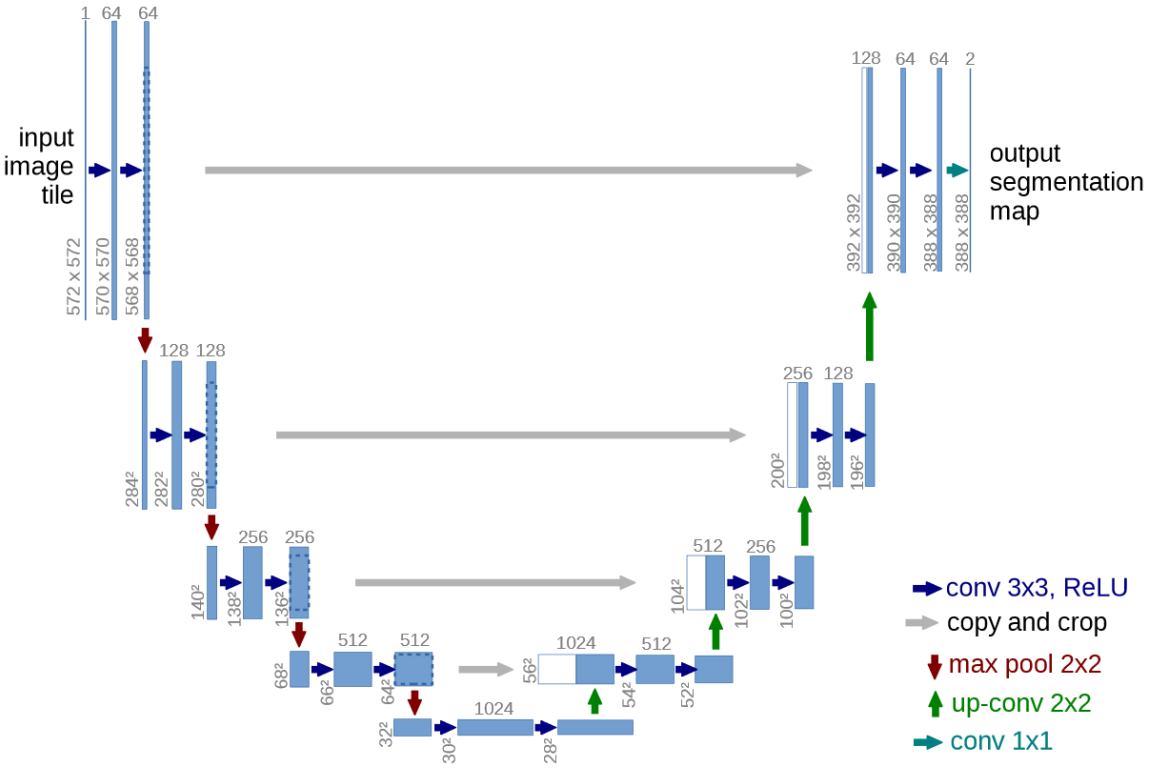
1. **Zväčšenie rozlíšenia** – Vytvorenie mapy príznakov, s dvojnásobným rozlíšením takým spôsobom, že jeden príznak rozdelíme pomocou mriežky 2x2 na štyri rovnaké príznaky. Rozlíšenie sa zväčší, avšak susedné príznaky budú mať rovnaké hodnoty.

2. **Konvolúcia 2x2** – Použitie tradičnej konvolučnej vrstvy, ktorá bola vysvetlená na obrázku 3.6. Veľkosť jadra je rovnaká, ako v predchádzajúcej vrstve 2x2.

Pri zväčšovaní rozlíšenia sa najprv použije upsampling operátor a potom, sa výsledná mapa príznakov spojí s mapou príznakov z kontrakčenj cesty. Vznikne mapa príznakov s dvojnásobným počtom kanálov. Takto je možné posúvať informáciu o type objektu na obrázku do vrstvy s vyšším rozlíšením. Dôsledkom je, že expanzívna cesta je prakticky symetrická s kontrakčnou cestou, a tak vzniká architektúra v tvare písmena U. Sieť neobsahuje žiadne plne prepojené vrstvy, segmentačná mapa obsahuje iba pixely, kde je známy celkový kontext vstupného obrázku. Aby sa kvôli konvolúcii nezmenšoval vstup, tak sa v okrajových častiach obrázku používa zrkadlenie obrazu za hranicu. To umožňuje kvalitnejšiu predikciu triedy na okrajoch obrázku. Táto funkcia je dôležitá pre veľké obrázky, ktoré treba rozdeliť do menších dlaždíc.

Sieť je trénovaná pomocou trénovacej databázy, ktorá obsahuje vstupné obrázky a očakávané segmentácie. Očakávaná segmentácia je obrazok, kde hodnota pixelu udáva, triedu, do ktorej daný pixel patrí. Hlavná výhoda siete U-Net je, že množstvo obrázkov potrebných na trénovanie siete je výrazne nižšie ako u starších metód. U-Net pracuje vždy na štvorcových obrázkoch, preto je potrebné prispôsobiť formát vstupu.

Sieť U-Net podporuje tzv. data ugmentation. To znamená, že vstupné obrázky sú mierne pozmenené npr. pomocou deformácie či natiahnutia obrazu. Táto metóda má dve výhody zabráni pretrénovaniu siete a vytvorí väčšiu trénovaciu databázu.



Obr. 4.4: Architektúra siete U-Net [17] s najnižším rozlíšení 32x32 pixelov. Každý modrý obĺžnik reprezentuje viackanálovú mapu príznakov. Počet kanálov je zapísaný na vrchu obdĺžnika. Na ľavej spodnej strane je x-y veľkosť mapy. Biele obdĺžniky znázorňujú prekopírované mapy príznakov. Operácia crop oreže okrajové hodnoty mapy príznakov, je to dôležité kvôli strate okrajových pixelov pri konvolúcii.

Kapitola 5

Návrh

5.1 Popis problému

Cieľom nášho programu je spracovať mračno bodov, ktoré zachytáva trojrozmerný obraz cesty a vytvoriť lomenú čiaru, ktorá odpovedá čiare na vozovke. Vytvorená lomená čiara je uložená vo vektorovej podobe, ako tabuľka bodov. Proces postupného spracovania dát rozdeľujeme do nasledujúcich krokov:

1. **Príprava dát** – Ide o rozdelenie veľkého mračna bodov na menšie a zmenšenie rozlíšnia pomocou voxelizácie viac 5.2.
2. **Vytvorenie databázy výrezov** – Načítanie mračien bodov a generovanie pohľadov zhora na vozovku.
3. **Odhad pozície čiary vo výreze** – Vytvorenie segmentácie výrezu, každý pixel výstupného obrázku určuje, aký objekt sa na danom mieste nachádza.
4. **Vytvorenie vektorového popisu** – Na záver je pravdepodobnostná mapa algoritmicky prevedená na vektorový popis, ktorý sa uloží do geografického informačného systému.

Program bude pracovať v interaktívnom režime. To znamená, že užívateľ bude interagovať s programom. Užívateľ zadá počiatok čiary, ktorú chce označiť a program ju bude predlžovať. Po vrátení čiary môže užívateľ zadať počiatok inej čiary a proces sa opakuje. Predmetom práce nie je vytvoriť grafické užívateľské rozhranie, a preto program očakáva súradnice počiatočného bodu v textovej podobe.

5.2 Príprava dát

Cieľom tejto časti je urýchliť načítanie mračien bodov z pamäti počítača tak, aby bola výsledná aplikácia použiteľná v interaktívnom režime. Problém je, že mračná bodov sú veľmi pamäťovo náročné, a to hlavne z dôvodu veľkého počtu bodov v súbore. Súbor obsahujúci mračno bodov môže mať stovky megabajtov. Cesta dlhá niekoľko kilometrov je zachytená vo veľkom počte mračien bodov. Z toho vyplýva, že pre označenie kilometer dlhej čiary by bolo potrebné načítať rádovo gigabyte dát z pamäti. Na zmenšenie pamäťovej náročnosti programov používame dva spôsoby redukcie množstva načítaných dát:

1. **Zníženie rozlíšenia** – je zníženie hustoty bodov, pri ktorej dochádza k nevratnej strate informácie. Určitá množina bodov sa nahradí jedným bodom, ktorý dobre reprezentuje danú množinu. Často sa používa priemerovanie bodov v danej množine.

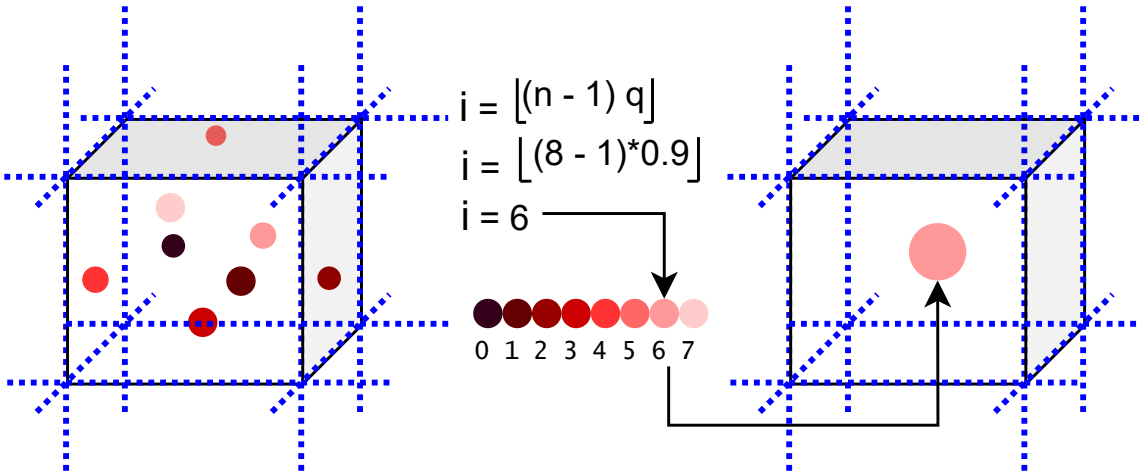
2. **Rozdelenie mračien.** – Mračná bodov, ktoré máme k dispozícii obsahujú veľké množstvo nezaujímavých dát. Ide o vzdialené objekty stromy vedľa cesty a podobne. Riešením je rozdeliť veľké mračno bodov na malé, a neskôr načítavať len tie mračná, ktoré sú nevyhnutne potrebné.

5.2.1 Zníženie rozlíšenia

Na zníženie rozlíšenia bola použitá tzv. voxelizácia. Voxel (volumetric pixel) je kocka v trojdimenzionálnom priestore. Voxelizácia je proces, ktorý body pre každý voxel nahradí jedným reprezentujúcim bodom. V jednom voxele sa môžu vyskytovať stovky bodov, a preto voxelizácia prináša značnú pamäťovú úsporu. V našom prípade bola pre výpočet reprezentujúceho bodu voxelu zvolená relatívne komplikovanú funkciu:

Algoritmus 2: Algoritmus agregácie jedného voxelu

Input: v – Vektor bodov voxelu
Input: n – Počet bodov voxelu
Input: q – Kvantil
Output: Výsledný bod voxelu
Function `Aggregation`(v, n, q)**:**
    $v \leftarrow$ zoraď v podľa intenzity;
    $i \leftarrow \lfloor (n-1) * q \rfloor$;
    return $v(i)$;



Obr. 5.1: Ukážka, ako funguje voxelizácia. Na lavej strane je pôvodné mračno bodov, ktoré môže byť veľmi rozsiahle, preto sa zameriame len na jednu kocku v priestore tzv. voxel (volumetric pixel). Voxel obsahuje viacero bodov. Farba bodu znázorňuje intenzitu, svetlejšia farba znamená vyššia intenzita. V strednej časti je výpočet, kde n je počet bodov, q je kvantil a i je výsledný index. Napravo je výsledný bod, ktorého súradnice vždy odpovedajú stredu voxelu.

Táto funkcia vyžaduje, aby bol pre každý voxel vytvorený vektor všetkých bodov vo vnútri. Je teda potrebné najprv prejsť celé mračno a vytvoriť vektory. Výhoda je, že funkcia nerobí len jednoduché priemerovanie hodnôt, ale vyberá hodnoty s vyššou intenzitou na základe kvantilu. Výsledná hodnota potom s vysokou pravdepodobnosťou nebude nejaká odľahlá hodnota, ktorá sa vyskytuje len zriedkavo. Dôvod, prečo vyberáme vyššie hodnoty je, aby sa zvýšil kontrast medzi vodorovným dopravným značením a vozovkou. Obrázok 5.1 vizuálne znázorňuje algoritmus.

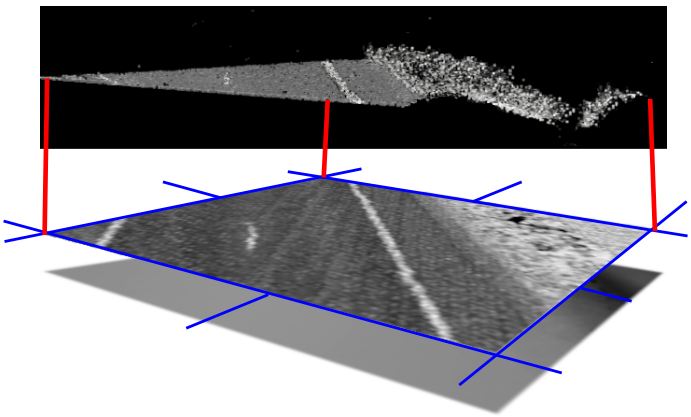
5.2.2 Rozdelenie mračien

Ide o jednoduchú operáciu, ktorá na základe hodnôt x-ovej a y-ovej súradnice rozdelí body do viacerých mračien. Každé mračno potom uloží do samostatného súboru. Výsledné mračno zaberá veľmi malú oblasť napr. 15x15 metrov. Pôvodné mračno bodov môže mať napr. 100x100 metrov. Vďaka tomuto riešeniu potom postupné prechádzanie po ceste nevyžaduje načítavanie veľkého objemu dát, ktoré nie sú potrebné. Načítajú sa len dáta v bezprostrednej blízkosti a z nich sa neskôr vytvorí výrez.

5.3 Vytvorenie databázy anotovaných výrezov

Problémom pri práci s mračnom bodov je, že body nie sú v mračne priestorovo usporiadané a môžu mať na rôznych miestach rôznu hustotu. To znamená, že je veľmi zložité takýto typ dát priamo transformovať do normalizovanej matice, ktorú potrebujeme ako vstup konvolučnej neurónovej siete. Z toho dôvodu sú mračná bodov transformované do dvojdimenzionálnych výrezov. Pri transformácii nastáva určitá strata informácie a presnosti, ale výsledkom sú obrázky, ktoré je potom možné omnoho jednoduchšie spracovať prostredníctvom konvolučnej neurónovej siete.

Generátor výrezu



Obr. 5.2: Vytváranie výrezu premietnutím mračna bodov do plochy. Vzniká intenzitový a hĺbkový obrázok.

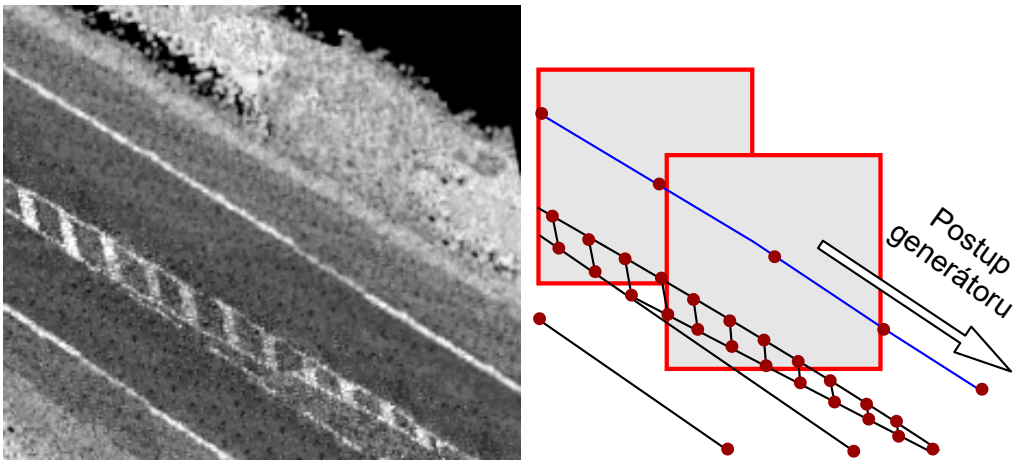
Generátor výrezov najprv spája najviac štyri mračná bodov, ktoré sú v zábere, a až potom generuje výrez. Z toho dôvodu sa používa buffer štyroch posledne použitých mračien, ktoré je v prípade potreby možné spojiť a vygenerovať výrez. Pri načítaní ďalšieho mračna

sa nepotrebné mračno odstráni z pamäti. Týmto spôsobom sa znižuje počet pomalých prístupov na disk.

Na základe viacerých odborných článkov 4, ktoré často používali projekciu do vodorovnej plochy, bol ako spôsob zobrazenia mračna zvolený pohľad zhora. Vytvorenie pohľadu zhora je znázornené na obrázku 5.2. Ide o jednoduchý spôsob zobrazenia mračna bodov, kde ale dochádza k stratám informácie aj presnosti. Výhodou je, že výsledný výrez je výrazne menší než pôvodné mračno bodov a dáta sú uložené v normalizovanej mriežke. Vďaka tomu je možné pre detekciu objektov použiť klasickú 2D konvolučnú sieť. Výrez sa skladá z dvoch druhov obrázkov:

1. **Obraz intenzity** – zachytáva intenzitu odrazeného svetla od povrchu vozovky.
2. **Obraz hĺbky** – zachytáva hĺbku vo zvislom smere. Každý pixel tohoto obrázku je možné previesť na nadmorskú výšku.

Generátor anotovanej databázy



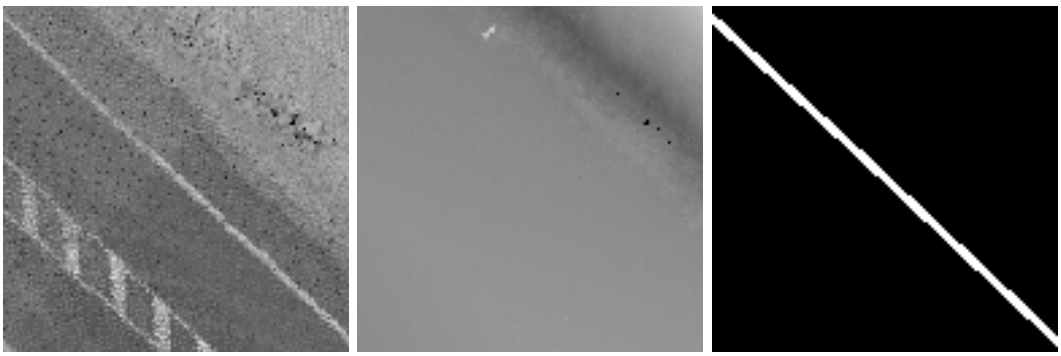
Obr. 5.3: Postup generovania databázy výrezov. Modrou a čiernymi čiarami je znázornená časť vodorovného dopravného značenia, ktoré bolo dopredu vytvorené ručne. Modrou farbou je označená aktuálna čiara, cez ktorú program prechádza s určitým krokom. Červený štvorec znázorňuje oblasť, z ktorej sa vytvorí výrez. Pre červený štvorec vždy platí, že čiara záujmu prechádza cez stred štvorca.

Pri generovaní databázy anotovaných výrezov sa vytvárajú výrezy a odpovedajúce anotácie. Anotácia je obrázok, v ktorom hodnota 0 označuje pozadie a hodnota 255 označuje čiaru prechádzajúcu stredom obrázku. Žiadna iná hodnota sa v anotačnom obrázku nenachádza. V zábere obrázku môžu byť viaceré čiary, avšak vyznačená je len tá, ktorá ide cez stred obrázku. Dôvodom je, že obrázky s vyznačením jedinej čiary sú výrazne jednoduchšie na neskoršie spracovanie.

Generátor anotovanej databázy výrezov potrebuje ako vstup mračná bodov a súbor obsahujúci vzorové vodorovné dopravné značenie. Postup generovania zobrazený na obrázku 5.3 je nasledujúci:

1. Pokiaľ existuje ďalšia lomená čiara zo súboru vzorového VDZ, tak ju načítaj do premennej l, inak ukonči.

2. Navzorkuj lomenú čiaru l krokom o veľkosti k. Vznikne množina bodov P, ktoré patria čiare l.
3. Pokiaľ $P \neq \emptyset$, tak vyber bod $S \in P$, inak pokračuj bodom 1.
4. Vygeneruj výrez mračna (obr. 5.2) a anotáciu čiary l tak, že bod S je stredom.
5. Odstráň bod S z množiny P a pokračuj bodom 3.



Obr. 5.4: Jedna vzorka trénovacej databázy. Naľavo je intenzitový obrázok, v strede hĺbkový a napravo je anotácia čiary.

5.4 Odhad pozície čiary vo výreze

Pre odhad pozície čiary sa používajú výrezy vozovky privedené na vstup siete U-Net. Cieľom je vytvoriť per pixel segmentáciu čiary, ktorá je blízko stredu výrezu. Dôležité je, aby použitý model bol schopný vyjadriť neistotu rozhodnutia. Pokiaľ sa vo výreze čiara vôbec nenachádza, tak aby model vyjadril, že si nie je istý pozíciou čiary. Môže sa stať aj to, že prerušovaná čiara prechádza na plnú. V takom prípade by bolo vhodné získať informáciu, kde končí prerušovaná čiara.

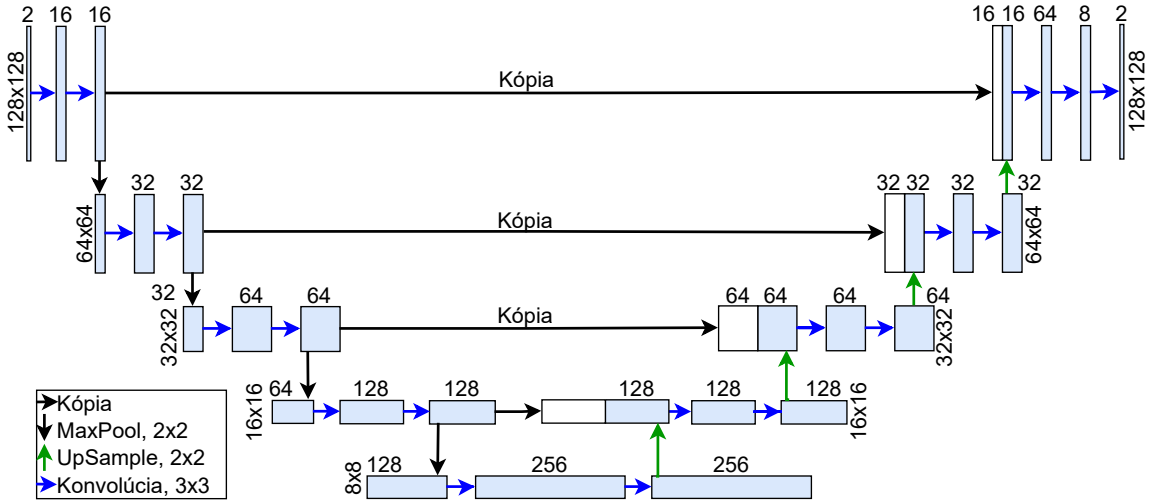
Sieť U-Net je vhodná na riešenie takto definovanej úlohy. Je robustná vzhľadom k šumu a dokáže vyjadriť neistotu. Táto sieť vytvára tzv. per pixel segmentáciu, čiže čiernobiely obrázok, v ktorom hodnota pixelu určuje či daný pixel patrí, alebo nepatrí do hľadanej čiary. Príklad vozorvej segmentácie je na obrázku 5.4. Vstup pozostáva z dvoch kanálov. To znamená, že na vstup neurónovej siete je privedený výrez, čiže dvakrát obrázok tej istej vozovky. Avšak jeden obrázok zachytáva intenzitu a druhý hĺbku.

Výstupom siete bude obrázok s dvomi kanálmi, každý kanál si môžme predstaviť ako samostatný obrázok. Jeden kanál zachytáva pravdepodobnosť výskytu čiary a druhý znázorňuje pravdepodobnosť pozadia. Pre každý pixel obrázku je potom možné určiť s akou pravdepodobnosťou sa tam nachádza čiara a s akou je tam pozadie. Segmentácia je spojenie týchto dvoch kanálov tak, že pre každý pixel vyberieme triedu (čiara, vozovka), ktorá má vyššiu pravdepodobnosť.

5.4.1 Architektúra siete

Sieť U-Net pozostáva z konvolučných vrstiev, kde veľkosť používaného konvolučného jadra je buď 2x2, alebo 3x3. Ďalej sú tu aktivačné vrstvy, ktoré používajú funkciu ReLU. Max

pooling je vrstva , ktorá zníži rozlíšenie obrázka tak, že zo štyroch susedných pixelov vyberie ten, ktorý má maximálnu hodnotu. Up samplig pracuje opačne, vezme mapu príznakov z nižším rozlíšením a každý pixel rozdelí na štyri pixely s rovnakou hodnotou. Up sampling je na obrázku 5.5 zelenou šípkou. Potom sa takto zväčšená mapa príznakov spojí s mapou príznakov z kontrakčnej cesty a aplikujú sa konvolúcie s jadrom 3x3.



Obr. 5.5: Návrh siete U-Net, ktorá generuje segmentácie výrezov vodorovného dopravného značenia. Veľkosť vstupného a výstupného obrázka je rovnaká. Vstupný aj výstupný obrázok obsahujú dva kanály. Sieť sa skladá z modrých blokov. Na ľavej strane bloku vidíme rozlíšenie a na hornej strane počet kanálov mapy príznakov.

5.4.2 Trénovanie

Na trénovanie siete bude potrebná databáza vstupných výrezov vytvorených z mračien bodov, postup vytvárania databázy bol popísaný vyššie 5.3. Každý vstupný výrez bude obsahovať dva kanály, jeden pre intenzitu a druhý pre hĺbku. Ďalej bude potrebná databáza anotovaných obrázkov. Od siete budeme požadovať, aby sa naučila zo vstupných výrezov vytvárať očakávanú segmentáciu.

Databáza anotovaných obrázkov sa vytvára z dopredu označeného vodorovného dopravného značenia. Vodorovné dopravné značenie je uložené vo vektorovej podobe, kde každá lomená čiara odpovedá jednej čiare na vozovke. Vodorovné dopravné značenie sa tradične označuje ručne, a to je potrebné automatizovať týmto systémom.

Sieť U-Net nevyžaduje veľmi veľké množstvo trénovacích vzorov, približne 1000 obrázkov stačí na trénovanie. Po trénovaní na takejto vzorke je už vidieť, že model pochopil čo má hľadať, ale so zriedkavo sa vyskytujúcimi situáciami má problém. Väčšie množstvo obrázkov bolo vytváraných technikou data augmentation, čo znamená vytvorenie umelých trénovacich dát úpravou existujúcich dát. Technika data augmentation je popísaná v nasledujúcej časti 5.4.3.

Na výpočet chyby modelu sa rovnako, ako v pôvodnej implementácii [17], bude používať funkcia soft max, ktorá normalizuje výstupy siete tak, aby suma všetkých výstupov bola rovná jednej, kombinovaná s funkciou krížovej entropie. Sieť sa potom učí (upravuje váhy spojení) algoritmom stochastic gradient descent založenom na spätnom šírení chyby.

5.4.3 Technika data augmentation

Technika data augmentation je proces, ktorého cieľom je zväčšiť množstvo trénovacích dát pridaním mierne upravených kópií dát z databázy, alebo umelo vytvorených dát. Pridaním nových dát sa znižuje pravdepodobnosť pretrénovania.

V našom prípade boli vytvorené upravené kópie existujúcich výrezov a odpovedajúcich anotácií. Boli použité úpravy ako: otočenie, prevrátenie, zmena jasu a deformácie obrázkov. Vďaka tomu sa veľkosť trénovacej databázy zväčšila približne päťnásobne, a model bolo možné kvalitnejšie natrénovať.

5.5 Generovanie vektorového popisu

Generátor vektorového popisu je algoritmus, ktorého vstupom sú mračná bodov a počiatočný bod (prvý bod čiary na vozovke) a výstupom je lomená čiara odpovedajúca čiare na vozovke. Tento proces je možné rozdeliť na dve časti:

1. **Odhad lokálnej čiary** – proces, ktorého vstupom je pozícia v priestore a mračno bodov a výstupom je lomená čiara prechádzajúca v blízkosti zadanej pozície. Na odhad čiary vo výreze sa používa sieť U-Net. Podrobný postup je rozpísaný nižšie v časti 5.5.1.

2. **Spájanie čiar do globálnej čiary.** – Vstupom sú dve čiary: globálna a lokálna. Globálna čiara je taká lomená čiara, ktorá už v predchádzajúcich krokoch vznikla spojením lokálnych čiar. A výstupom je nová globálna čiara, ktorá spája tieto dve čiary. Postup je podrobne vysvetlený nižšie v časti 5.5.2.

5.5.1 Odhad lokálnej čiary

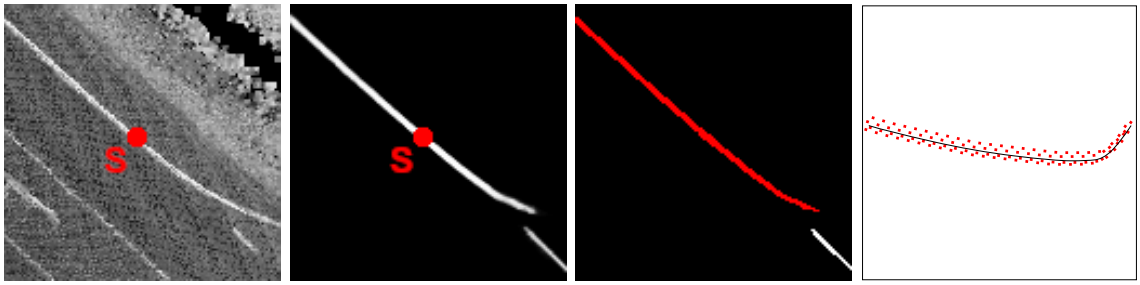
Cieľom tohoto programu je získať vektorovú lomenú čiaru z danej oblasti priestoru. Oblasť je definovaná jedným bodom, ktorý má tri priestorové súradnice $S = (x, y, z)$. Tento bod je stred kocky (voxelu) ktorej dĺžka hrany je implicitne definovaná v hlavičkovom súbore. Program hľadá čiaru v blízkosti bodu S. Na začiatku je bod S zadaný užívateľom, ktorý tým definuje čiaru záujmu. Na odhad lokálnej čiary sa používa generátor výrezu 5.3, ktorý vytvorí pohľad zhora a U-Net 5.4, ktorý vytvára segmentáciu výrezu. Odhad je realizovaný, ako postupnosť nasledovných krokov, ktoré sú tiež ilustrované na obrázku 5.6:

1. Sieť U-Net podľa výrezu okolia bodu S, vytvorí odhad, ktorý pre každý pixel udáva pravdepodobnosť výskytu čiary.

2. Z odhadu sa vyberú body, ktoré pravdepodobne obsahujú čiaru na vozovke. Potom tieto body vstupujú do zhlukovej analýzy, ktorá vráti len zhluk v blízkosti stredu obrázka.

3. Body stredového zhluku sú ďalej použité na analýzu hlavných komponentov. Za pomoci tejto metódy získame tzv. vlastné hodnoty a vlastné čísla. Vlastnému vektoru u odpovedá najväčšie vlastné číslo a udáva smer najväčšieho rozptylu bodov. Z vektoru u sa vypočíta uhoľ ϑ.

4. Stredový zhluk sa otočí o uhoľ $-\vartheta$. Otočenie je dôležité. Bez otočenia môže nastať situácia, že požadovaná krivka nie je funkcia, a teda jednej súradnici x odpovedá



Obr. 5.6: Odhad lokálnej čiary. Bod S označuje vstup užívateľa, alebo predošlého kroku. Spracovanie prebieha zľava. Prvý obrázok je intenzita odrazeného svetla. Druhý obrázok je predikcia čiary blízko stredu obrázka pomocou siete U-Net. Na treťom obrázku sú dva zhluky, červený označuje zhluk najbližší stredu. Posledný obrázok ukazuje otočenie bodov čiary a preloženie polynómom tretieho stupňa.

viacero súradníc y. Ak by dáta neboli otočené, tak by odhad polynomiálnej funkcie mohol vrátiť nezmyselné výsledky.

5. Preloženie bodov stredového zhluku polynomiálnou krivkou tretieho stupňa. Ďalej sa dosadením bodov do funkcie vytvorí lomená čiara.

6. Spätné otočenie lomenej čiary o uhoľ ϑ.

5.5.2 Spájanie čiar do globálnej čiary

Cieľom programu na spájanie čiar je vytvoriť jednu globálnu lomenú čiaru, ktorá by spájala všetky lokálne čiary. V skutočnosti ale nikdy nie sú k dispozícii všetky lokálne čiary, ale generujú sa postupne. Na začiatku je dostupný len jeden bod, ktorý určuje počiatok čiary. Kroky algoritmu 5,6 a 7 sú zobrazené na obrázku 5.7. Algoritmus pracuje nasledovne:

1. Lokálna čiara sa z bodu S odhadne pomocou algoritmu odhadu lokálnej čiary.

2. Pokiaľ je dĺžka lokálnej čiary dostatočná, tak pokračuj, inak skonči.

3. Pokiaľ je globálna čiara neinicializovaná, tak lokálna čiara sa stáva globálnou.

4. Vezmi posledný bod globálnej čiary a bod globálnej čiary vzdialený o veľkosť okna od konca.

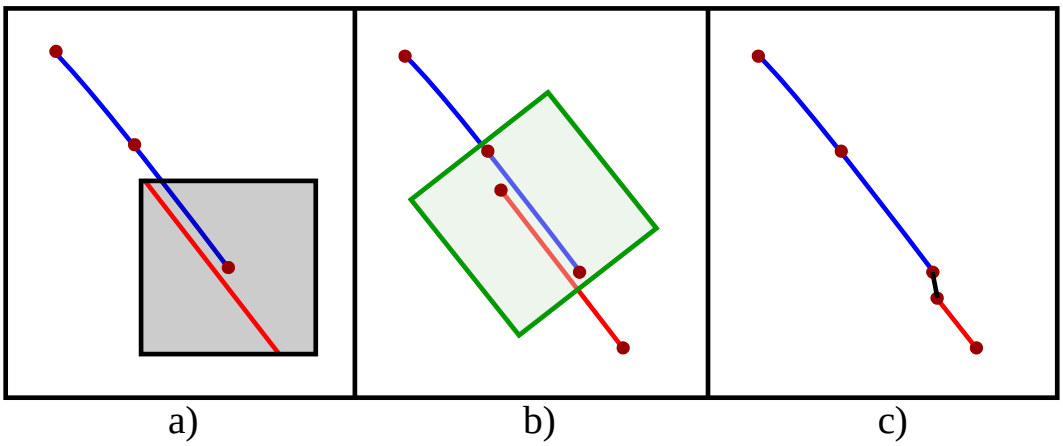
5. Vytvor obdĺžnik O nad týmito dvomi bodmi tak, že body budú ležať na osi obdĺžnika (obr. 5.7a).

6. Od lokálnej čiary odčítaj obdĺžnik O (obr. 5.7b).

7. Vytvor novú globálnu čiaru zapojením lokálnej čiary za globálnu (obr. 5.7c).

8. Ulož novovytvorený koniec globálnej čiary, do bodu S. Pokračuj krokom 1.

Tento algoritmus spájania čiar bol zvolený preto, že je implementačne veľmi jednoduchý. Na obrázku 5.7 sú chyby medzi čiarami zámerne zvýraznené, aby bol algoritmus pochopiteľný. V skutočnosti sú chyby maximálne v jednotkách centimetrov a nenastáva problém pri spájaní čiar.



Obr. 5.7: Postup spojenia čiary vytvorenej v predchádzajúcich krokoch a aktuálnej čiary. Modrá farba zobrazuje globálnu čiaru vytvorenú v predošlých krokoch a aktuálna, lokálna čiara je červená. Obrázok a) ukazuje, že na konci globálnej čiary sa zavolá funkcia na predikciu lokálnej čiary. V kroku b) sa na konci globálnej čiary vytvorí zelený obdĺžnik O na orezanie lokálnej čiary. Obdĺžnik O sa vytvorí tak, aby mierne presahoval za koniec globálnej čiary. Potom sa lokálna čiara oreže tak, že zostane len časť mimo zelený obdĺžnik. Obrázok c) zobrazuje výsledok, ktorý vznikne prepojením segmentov.

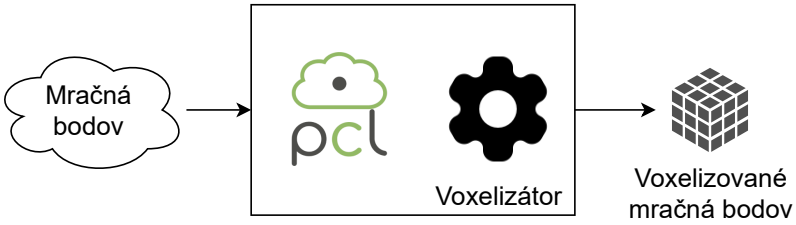
Kapitola 6

Implementácia

Táto kapitola popisuje implementačné detaily. Nachádza sa tu popis použitých knižníc a formátov ktoré sa v programe využívajú. Implementácia je rozdelená na dve časti, backend je napísaný v jazyku C++ a frontend je v jazyku Python. Generátor výrezov predstavuje backend a Prediktor je frontend, ktorý sa priebežne dotazuje na nový výrez.

6.1 Voxelizátor

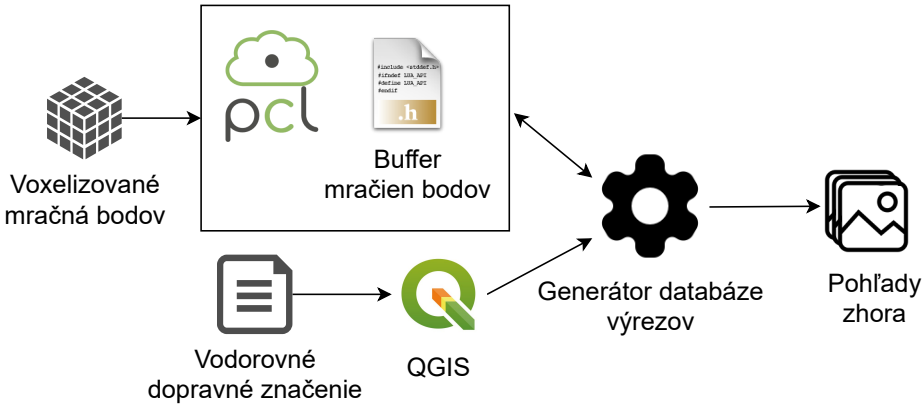
Základom celého systému na detekciu čiary na vozovke sú dva programy. Prvý voxelizuje mračná bodov a druhý z voxelizovaných mračien vytvára trénovaciu databázu výrezov. Mračná bodov sú raz zmenšené pomocou voxelizátora, a potom všetky ďalšie nástroje využívajú voxelizované mračná. Je to preto, aby ostatné nástroje načítavali menej dát z disku a boli schopné pracovať interaktívne. Samotná voxelizácia je dosť pomalá, ale vykoná sa len raz pre celý projekt. Voxelizátor je implementovaný v jazyku `C++` a vyžíva knižnicu PCL, ako je vidieť na obrázku 6.1.



Obr. 6.1: Architektúra voxelizátora, ktorý znižuje pamäťovú náročnosť mračien bodov.

6.2 Generátor výrezov

Generátor výrezov je implementovaný v jazyku C++ a využíva pre svoju funkciu hlavne knižnice QGIS a PCL. Potrebuje tiež databázu voxelizovaných mračien bodov, ktoré vytvoril voxelizátor 6.1 . Z mračien chceme získať výrezy vozovky. Výsledkom je potom výrez obsahujúci intezitu a hĺbku, a obrázok vzorovej segmentácie. Architektúra generátora výrezov je zobrazená na obrázku 6.2.



Obr. 6.2: Architektúra generátora databázy anotovaných výrezov.

- **PCL** – z anglického Point Cloud Library[1] je komplexná voľne šíriteľná knižnica distribuovaná pod BSD licenciou. Slúži na spracovanie a manipuláciu s mračnami bodov. PCL používame načítanie mračna bodov a následné filtrovanie podstatných častí mračna. Mračno obsahuje veľké množstvo bodov, ktoré pre nás nie sú zaujímavé a sú vzdialené od miesta skenovania. Preto mračno presúvame, filtrujeme a až potom premietame do roviny.
- **QGIS** – voľne šíriteľný geo-informačný systém[2] distribuovaný pod GNU licenciou. Slúži na prácu s geografickými dátami, umožňuje užívateľom zobrazovať a manipulovať s geografickými dátami. Knižnicu QGIS používame na čítanie súboru, v ktorom je zachytené vzorvé označenie vodorovného dopravného značenia. Generátor výrezov s určitým krokom prechádza čiarami vodorovného dopravného značenia. Podľa pozície je možné filtrovať mračno bodov a zachovať iba tie body, ktoré sú v dostatočnej blízkosti a sú pre nás najzaujímavejšie. Získame tak iba obraz jednej čiary a jej bezprostredného okolia.

6.3 Formát vstupu

Program na generovanie výrezov z mračien bodov očakáva na vstupe zložku obsahujúcu mračná bodov vo formáte `.pcd`. K spracovaniu pozície mračien je potrebný aj projektový súbor s príponou `.prj`. Veľmi dôležitý vstupný súbor je tiež súbor obsahujúci vzorové označenie vodorovného dopravného značenia.

6.3.1 Projektový súbor

Projektový súbor pre každé mračno bodov definuje polygón v priestore, ktorým je dané mračno ohraničené. Tento súbor sa používa pri generovaní výrezov. Program najprv načíta projektový súbor a vytvorí si vnútorné mapovanie. Na základe toho je program schopný rýchlo vyhľadať mračno bodov, ktoré odpovedá určitej pozícii v priestore.

Projektový súbor sa skladá z dvoch častí: z hlavičky a popisu jednotlivých blokov. Blok môžeme chápať ako dlaždicu v priestore, ktorá obsahuje mračno bodov a určuje jeho hranice.

[1] https://pointclouds.org/
[2] https://qgis.org

V hlavičke projektového súboru je dôležitá iba informácia veľkosti bloku ktorá udáva, aká je dĺžka a šírka jedného mračna bodov v metroch.

Na prvom riadku príkladu bloku (obr. 6.3) je názov bloku, ďalej sú tu položky GroupFirst a GroupCount, ktoré ale nevyužívame a na nasledujúcich piatich riadkoch sú body popisujúce polygón ohraničujúci daný blok. Prvý a posledný bod polygónu musia byť zhodné. V našom prípade sú bloky vždy štvorce so stranou 100 metrov.

```
Block I-21_04_000001.laz
GroupFirst=1000000
GroupCount=1000000
-868200.0000 -1043900.0000
-868200.0000 -1043800.0000
-868100.0000 -1043800.0000
-868100.0000 -1043900.0000
-868200.0000 -1043900.0000
```

Obr. 6.3: Formát bloku projektového súboru, ktorý popisuje jenu dlaždicu priestoru.

6.3.2 Mračná bodov

V úvode kapitoly sme spomínali, že generátor výrezov potrebuje na svoju činnosť databázu `.pcd` súborov tieto súbory obsahujú mračná bodov, ktoré generátor využíva na vytváranie výrezov cesty.

Z príkladu formátu bloku 6.3.1 vidíme, že názov súboru obsahuje príponu `.laz` ale program očakáva mračno bodov vo formáte `.pcd`. Súbor v `.pcd` formáte je možné načítať pomocou knižnice `PCL`. Preto je potrebné mračno bodov najprv skonvertovať z formátu `.laz` na `.pcd`. Na konverziu mračna bodov bola použitá knižnica `pdal`, ktorá obsahuje modul `translate` na prevod medzi viacerými formátmi mračien bodov.

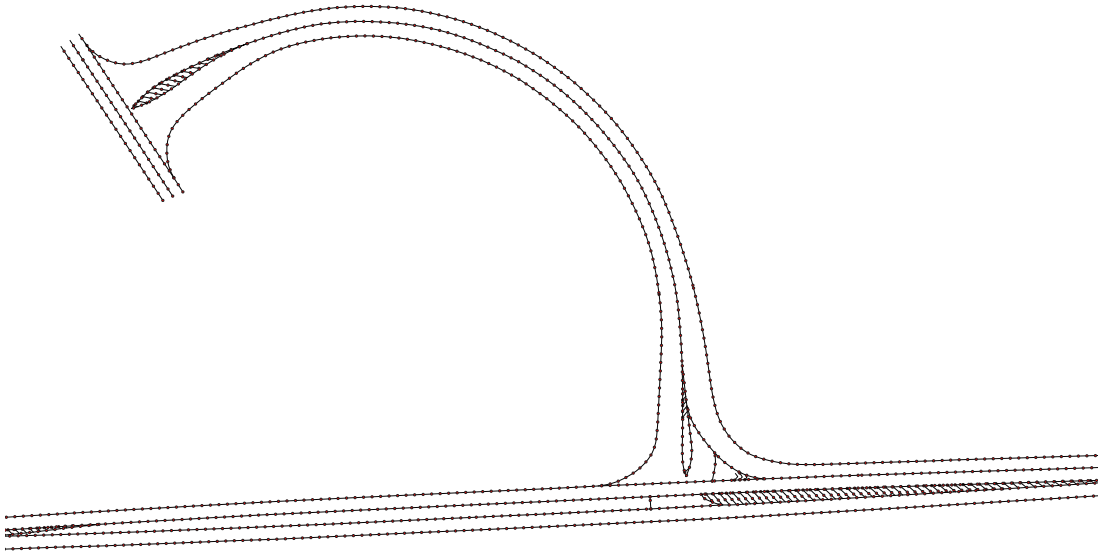
6.3.3 Súbor vzorového VDZ

Súbor vzorového vodorovného dopravného značenia je geografická databáza, do ktorej sú uložené informácie o pozíciách čiar na vozovke. Dáta vzorového súboru boli získané ručným označovaním dopravného značenia. Súbor obsahuje dve vrstvy: vrstvu bodov a vrstvu lomených čiar. Generátor vzorovej databázy výrezov používa iba vrstvu lomených čiar. Pohľad na časť vrstvy lomených čiar je na obrázku 6.4.

Pomocou knižnice QGIS je možné čítať súbor vzorového VDZ a získať tak jednotlivé lomené čiary. Každá lomená čiara s dostatočnou dĺžkou je použitá na vytvorenie databázy vzorových výrezov, ktorá sa neskôr použije na trénovanie modelu. Podrobný postup bol popísaný v časti 5.3.

6.4 Vytváranie lomenej čiary

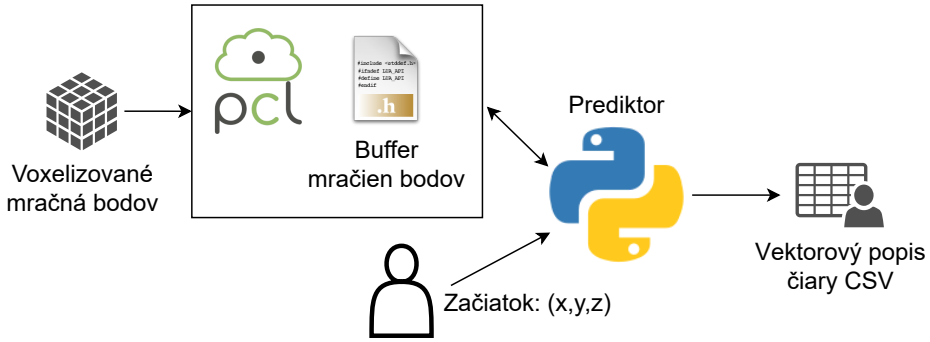
Spôsob postupného generovania lomenej čiary je podrobne vysvetlený v kapitole 5.5. Algoritmus je implementovaný v programovacom jazyku `Python`. Na obrázku 6.5 je vidieť, že Prediktor, ktorý je napísaný v jazyku `Python` vyžaduje pre svoju funkciu buffer mračien bodov, ktorý udržuje posledné mračná bodov a z nich je schopný vytvoriť výrez. Buffer mračien je implementovaný v `C++` a na to, aby k nemu mohol Prediktor pristupovať je potrebný



Obr. 6.4: Vrstva lomených čiar v súbore vzorového VDZ

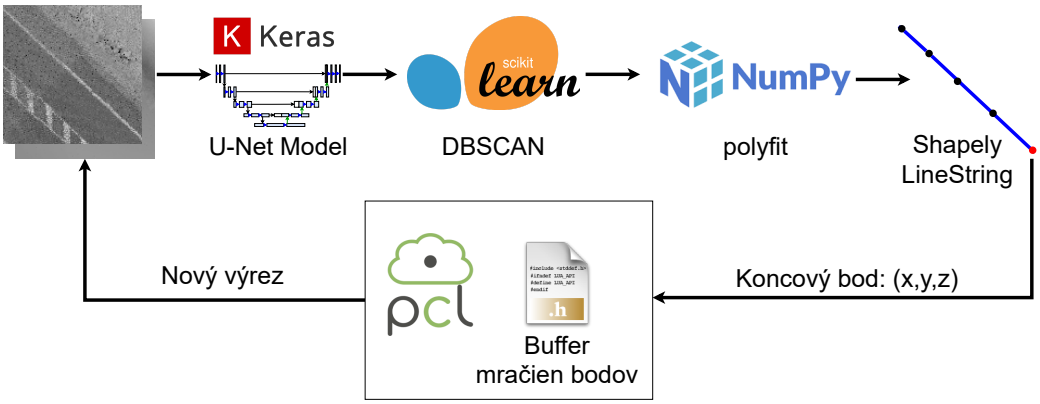
tzv. wrapper. Wrapper prevedie knižnicu v jazyku `C++` na statický objekt. Statický objekt je binárny súbor s príponou `.so`, ktorý je možné importovať v prostredí jazyka `Python`.

Prevod `C++` knižnice na statický objekt zabezpečuje nástroj SWIG. Tento nástroj na základe hlavičkového súboru, definície rozhrania a externých závislostí vytvorí statický objekt a triedu v jazyku `Python`. Potom je už možné triedu importovať a používať jej metódy.



Obr. 6.5: Architektúra predikcie lomenej čiary. Na začiatku užívateľ zadá počiatočnú pozíciu. Na základe pozície Prediktor pomocou siete U-Net predikuje čiaru, ktorú postupne predlžuje. Výrezy sa vytvárajú priebežne z voxelizovaných mračien. Výsledkom je tabuľka, kde na každom riadku je jeden bod odhadnutej lomenej čiary. Podrobnejší obrázok vnútornej architektúry Prediktoru je 6.6.

Architektúra vnútra Prediktoru je na obrázku 6.6. Na začiatku procesu predikcie je výrez, z ktorého sa pomocou siete U-Net, uloženej ako Keras model, odhadne segmentácia čiary. Potom sa použije funkcia DBSCAN z knižnice scikit-learn. Funkcia DBSCAN vytvorí zhluky na základe hustoty bodov. Vyberie sa zhluk najbližší stredu výrezu a pomocou funkcie polyfit sa preloží polynómom. Na záver sa vytvorí lomená čiara ako objekt knižnice Shapely a posledný bod čiary sa použije na vytvorenie nového výrezu.



Obr. 6.6: Vnútorná architektúra Prediktoru.

6.4.1 SWIG

Názov SWIG[3] vychádza z anglických slov *Simplified Wrapper and Interface Generator*. SWIG je nástroj na vývoj softvéru, ktorý prepája programy napísané v jazykoch C a C++ s rôznymi programovacími jazykmi na vysokej úrovni. Program SWIG podporuje rôzne typy cieľových jazykov vrátane bežných skriptovacích jazykov, ako sú Javascript, Perl, PHP, Python, Tcl a Ruby.

Softvér SWIG sa zvyčajne používa na analýzu rozhraní C / C ++ a na generovanie tzv. lepiaceho kódu požadovaného cieľovým jazykom. SWIG je voľne šíriteľný softvér a kód, ktorý generuje, je kompatibilný s komerčným aj nekomerčným využitím.

6.4.2 Keras

Keras je užívateľsky prívetivé aplikačné rozhranie postavené na knižnici TensorFlow. Tensorflow je často používaná voľne šíriteľná knižnica na strojové učenie. TensorFlow dokáže pracovať na procesore, na grafickej karte ako aj na mobilných zariadeniach [26]. Vďaka hlbokému prepojeniu s knižnicou TensorFlow Keras poskytuje jednoduché a prehľadné použitie, bez redukcie flexibility. Umožňuje užívateľovi upraviť hocijakú časť funkcionality.

Hlavnou výhodou rozhrania Keras je, že je jednoduchý na naučenie a tiež jednoduchý na použitie. S viac ako 400 000 používateľmi začiatkom roku 2021 má Keras silné uplatnenie v priemysle aj vo výskumnej komunite [3]. Keras podporuje paralelný výpočet na viacerých grafických kartách. Je tiež možné jednoducho previesť program, ktorý predtým bežal na procesore na veľký zhluk grafických kariet. Prostredníctvom masívneho paralelizmu je možné využiť ohromný výpočtový výkon.

V tejto diplomovej práci bol Keras využitý na implementáciu a trénovanie konvolučnej siete U-Net. Model siete je možné po trénovaní uložiť do súboru. Neskôr, keď bude potrebné sieť použiť na predikciu, tak sa načíta z uloženého súboru.

Na trénovanie siete bola použitá online služba Google CoLab, ktorá zdarma poskytuje grafické karty na trénovanie neurónových sietí. Pred zahájením trénovania bolo potrebné nahrať trénovaciu databázu na disk Google. CoLab je potom možné spojiť s diskom, a tak pristúpiť k databáze.

[3] http://www.swig.org/

Kapitola 7

Experimenty a výsledky

V tejto kapitole popíšeme experimenty vykonané na programoch. Ďalej tu budú definované metriky chyby programu. Cieľom je zistiť ako veľmi sa výsledky programu líšia od skutočnosti. Zaoberali sme sa dvomi typmi chyby: chybou modelu U-Net a chybou výslednej vektorovej reprezentácie.

7.1 Príprava testovacích dát

Jedná sa o vytvorenie databázy anotovaných výrezov z mračien bodov. Na vytvorenie databázy boli dostupné dva úseky cesty. Na výpočet výkonnosti bol použitý úsek dĺžky 5,56 km. Na väčšine dĺžky úseku má cesta dva jazdné pruhy a na niektorých miestach sú tiež pripájacie a odpájacie pruhy.

Na testovanie bol použitý notebook HP ENVY s procesorom Intel Core i5-7200U a SSD diskom. Okrem toho programy boli spustené vo virtuálnom prostredí WSL 2 (Windows Subsystem for Linux). To znamená, že namerané doby behu sú skôr vyššie, ako na natívnom Linuxovom zariadení.

V tabuľke 7.1 je rozpísaná doba behu jednotlivých programov. Na začiatku program PDAL translate prevedie komprimované mračná bodov do formátu PCD, voxelizátor zníži množstvo bodov a v poslednom kroku sa vytvára databáza anotovaných výrezov. Voxelizácia je časovo náročná, ale môže bežať offline bez zásahu používateľa. Výsledkom prípravy dát je databáza anotovaných výrezov, ktorú je možné použiť na trénovanie konvolučnej siete U-Net.

Program	**Doba behu**	**Veľkosť vstupu**	**Veľkosť výstupu**
Voxelizátor	44m 50s	8,52 GB	1,28 GB
Generátor DB	2m 38s	1,28 GB	27,6 MB

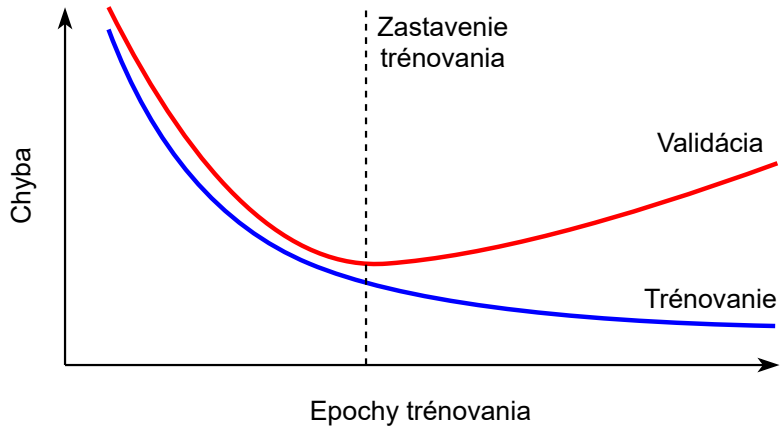
Tabuľka 7.1: Pamäťová a časová náročnosti vytvorenia databázy anotovaných výrezov z 5,56 km dlhého úseku cesty.

7.2 Trénovanie modelu U-Net

Trénovanie modelu je proces nastavovania vnútorných váh modelu tak, aby sa minimalizovala chyba na výstupe modelu. Model sa učí na trénovacej sade. Trénovacia sada je malá podmnožina všetkých možných pozorovaní. Cieľom trénovania je, aby bol model schopný

porozumieť všetkým pozorovaniam na základe trénovacej databázy. To znamená, že chceme, aby bol model schopný generalizovať. Na skoré ukončenie trénovania a vyhodnotenie kvality modelu bola databáza všetkých pozorovaní rozdelená na tri časti:

1. **Trénovacia sada** – Databáza použitá na učenie modelu.
2. **Validačná sada** – Databáza použitá na skoré zastavenie učenia, aby sa zabránilo pretrénovaniu. Keď chyba modelu na validačných dátach začne rásť, tak je potrebné zastaviť trénovanie (Obr. 7.1).
3. **Testovacia sada** – Databáza použitá na celkové vyhodnotenie kvality modelu.



Obr. 7.1: Chyba pre trénovacie dáta klesá až na nulu, ale pre testovacie dáta sa od určitého bodu začne znova zvyšovať, preto treba trénovanie včas zastaviť.

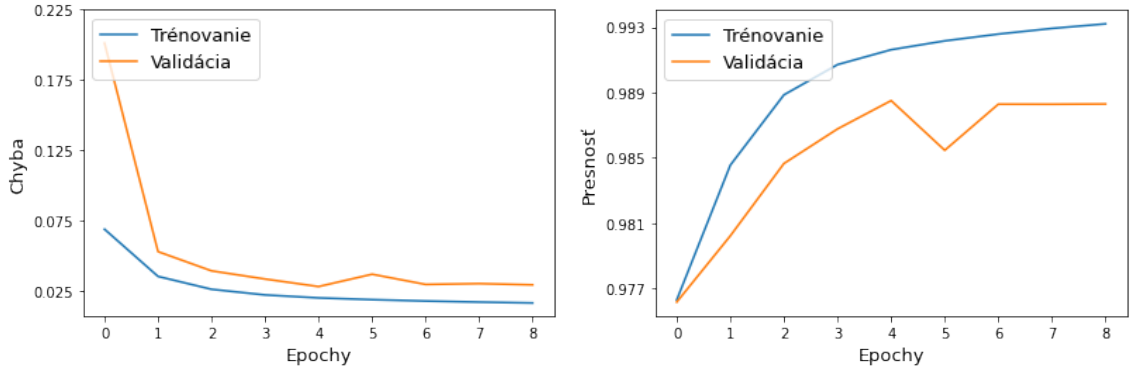
Dátová sada	Počet dát	Relatívny počet
Trénovacia	4575	75,32%
Validačná	750	12,35%
Testovacia	749	12,33%

Tabuľka 7.2: Rozdelenie dátových bodov na dátové sady.

7.2.1 Chyba a presnosť modelu

Chyba bola po každej epoche overená na trénovacej aj validačnej sade. Pri zhoršovaní validačnej chyby sa vykoná skoré zastavenie učenia. Zastavenie nastane, až keď zhoršenie chyby pretrváva štyri epochy učenia. Na výpočet chyby bola použitá funkcia binárnej krížovej entropie, ktorá je rozpísaná nižšie 7.1. Kde m udáva počet riadkov obrázka, n počet stĺpcov, y_{ij} je očakávaná hodnotu na danej pozícii a $\hat{y}_{ij}$ je hodnota predikcie na pozícii (i, j). Priebeh chyby je na obrázku 7.2.

$$Loss = \frac{1}{m * n} * \sum_{i=0}^{m} \sum_{j=0}^{n} (y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij})) \tag{7.1}$$



Obr. 7.2: Priebeh trénovacej a validačnej chyby a presnosti.

		Klasifikované hodnoty	
		Pozitívne	Negatívne
Skutočné hodnoty	Pozitívne	TP = 1,98%	FN = 0,738%
	Negatívne	FP = 0,46%	TN = 96,81%

Tabuľka 7.3: Chybová matica klasifikácie každého pixelu testovacej databázy.

Na výpočet presnosti predikcie jedného výrezu sa používa funkcia 7.3. Kde funkcia $f(x)$ 7.2 vráti hodnotu 1 pokiaľ sú triedy rovnaké, inak vráti nulu. Výsledná presnosť je potom desatinné číslo v intervale $< 0, 1 >$.

$$f(x) = \begin{cases} 1, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{7.2}$$

$$Accuracy = \frac{1}{m * n} * \sum_{i=0}^{m} \sum_{j=0}^{n} f(y_{ij} - \hat{y}_{ij}) \tag{7.3}$$

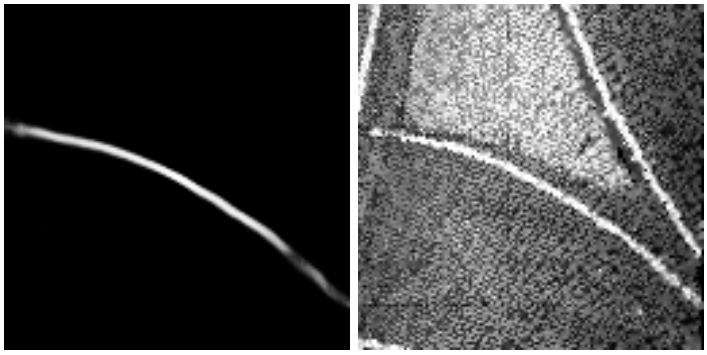
Priebeh presnosti je na obrázku 7.2. Väčšina plochy obrázka je tvorená pozadím a len jednotky percent tvorí čiara. Model teda môže dosiahnuť presnosť 90%, keď každý pixel klasifikuje, ako pozadie.

$$Specificity = \frac{TN}{TN + FP} = \frac{11880982}{11880982 + 56695} = 99,5\% \tag{7.4}$$

$$Sensitivity = \frac{TP}{TP + FN} = \frac{243330}{243330 + 90609} = 72,9\% \tag{7.5}$$

Nakoniec bol model otestovaný na testovacej sade obsahujúcej 749 vzorových výrezov. Priemerná dosiahnutá presnosť bola 98,8%. Hodnota chybovej funkcie na testovacej sade bola 0,03. Chybová matica modelu sa nachádza v tabuľke 7.3. Špecificita 7.4 je metrika, ktorá udáva pomer správne klasifikovaných bodov pozadia. Senzitivita 7.5 udáva pomer správne identifikovaných pixelov čiary.

Nižšia hodnota senzitivity oproti špecificite udáva, že model niekedy neoznačí čiaru, aj keď sa na danom mieste čiara nachádza. Opačný problém, že model označí čiaru na mieste pozadia je dosť zriedkavý. Model sa naučil porozumieť pozorovaniam a vracia dobré výsledky, ale na okrajoch obrázkov je neistý. V oblasti blízko stredu je model výrazne istejší, lebo sa naučil, že hľadaná čiara prechádza blízko stredu. To je vidno aj na obrázku 7.3, ktorý



Obr. 7.3: Problém predikcie čiary na okrajoch obrázku.

ukazuje menej kvalitnú predikciu. Ďalší dôvod nižšej senzitivity je, že čiara je široká len dva pixle a miernym posunutím odhadu čiary vzniká výrazná chyba.

7.3 Presnosť odhadnutej lomenej čiary

Cieľom je vypočítať presnosť lomenej čiary, ktorá je získaná spojením viacerých lokálnych lomených čiar. Lokálna lomená čiara je získaná z jedného obrázku vozovky pomocou siete U-Net a preloženia bodov polynomiálnou funkciou.

Pri spájaní viacerých lokálnych čiar môže nastávať chyba. Na základe metriky presnosti sme schopní merať celkovú kvalitu Prediktoru. Chyba je zapríčinená povahou problému. Samotné dáta obsahujú neurčitosť a nepresnosť. Model je preto schopný odhadnúť výsledok len s určitou presnosťou.

7.3.1 Metrika presnosti

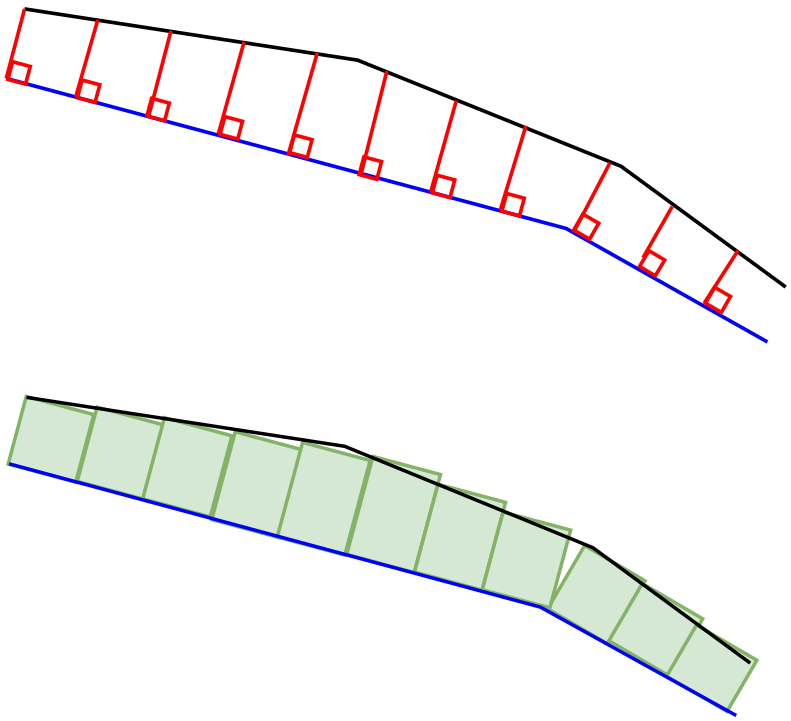
Na výpočet presnosti čiary vytvorenej modelom bola použitá veľkosť priemernej odchýlky medzi vzorovou a odhadovanou lomenou čiarou. Tento výpočet je len približný, lebo sa používa aproximácia odchýlky pomocou určitého počtu bodov. Algoritmus výpočtu presnosti prebieha v dvoch krokoch:

1. Odhadovaná lomená čiara sa navzorkuje s krokom 10 centimetrov. Vznikne veľké množstvo bodov a pre každý bod sa vypočíta najkratšia (kolmá) vzdialenosť k vzorovej lomenej čiare. Tento proces je zobrazený na obrázku 7.4.

2. Priemerná odchýlka sa počíta podľa plochy obdĺžnikov, kde jedna strana je veľkosť kroku a druhá vzdialenosť ku vzorovej lomenej čiare. Plocha je vyznačená na spodnom obrázku 7.4. Veľkosť odchýlky je potom normalizovaná na jeden centimeter čiary.

7.3.2 Dosiahnuté výsledky

Program odhadu lomenej čiary bol otestovaný na troch typoch čiar na vozovke. Výsledky testovania sú v tabuľke 7.4. V niekoľkých prípadoch sa program odhadu čiary úplne zastavil a bolo potrebné ho pustiť znovu na na nasledujúcom úseku. Zastavenie môže nastať, keď je program príliš neistý odhadom čiary. Plocha chyby je približne taká, ako plocha čiary. Väčšie problémy mal program s prerušovanou čiarou. Dôvodom je pravdepodobne to, že program musel odhadnúť čiaru aj v priestore prerušenia čiary.

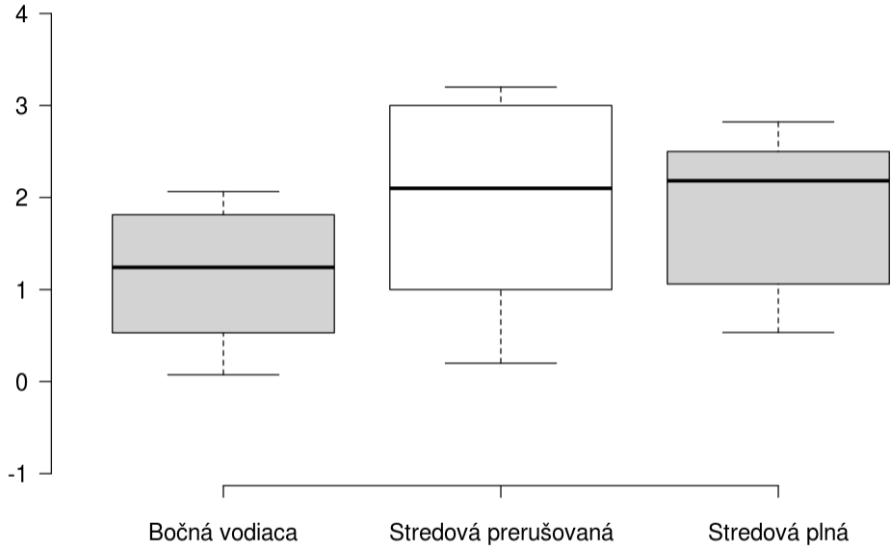


Obr. 7.4: Vzorkovanie výslednej, odhadnutej čiary. Modrou farbou je zakreslená odhadnutá čiara a čiernou vzorová čiara. Červené čiary ukazujú vzdialenosť dvoch čiar pre daný bod. Zelenou farbou je označená odhadnutá plocha medzi dvomi lomenými čiarami.

Typ čiary	**Priemerná odchýlka**	**Dĺžka čiary**	**Doba inferencií**	**Počet inferencií**	**Počet zastavení**
Bočná vodiaca	1,68cm	4350m	3m 44s	698	2
Stredová prerušovaná	2,24cm	890m	1m 23s	210	3
Stredová plná	2,03cm	430m	26,8s	38	0

Tabuľka 7.4: Vyhodnotenie celkových presností modelu a času vykonávania programu bez GPU akcelerátora.

Na obrázku 7.5 je zobrazený krabicový graf pre tri behy programu. Jedná sa o behy, ktorých štatistiky boli znázornená v tabuľke 7.4. Graf ukazuje, že maximálna normalizovaná odchýlka bola približne 3,3 centimetra.



Obr. 7.5: Krabicový graf zobrazujúcu chybu modelu pre jednotlivé typy čiar.

Kapitola 8

Záver

Diplomová práca sa zaoberá detekciou čiar vodorovného dopravného značenia na vozovke zo zachytených mračien bodov. Mračná bodov boli získané pomocou mobilnbého laserového skenovania technológiou Velodyne LiDAR. Cieľom je odhadnúť jednu lomenú čiaru, ktorá odpovedá čiare na ceste. Systém je navrhnutý tak, aby ho užívateľ mohol použiť v interaktívnom režime. Na začiatku užívateľ označí jeden bod čiary, ktorú chce označiť. Potom program predlžuje zvolenú čiaru, až kým čiara neskončí, výsledkom sú body lomenej čiary.

Na základe dostupnej literatúry sme vytvorili návrh nášho systému, ktorý vychádza z existujúcich systémov a využíva existujúce prostriedky a knižnice pre prácu s geo-informačnými dátami, mračnami bodov a konvolučnými neurónovými sieťami. Veľké množstvo odborných článkov využíva na detekciu značenia konvolučné neurónové siete, preto sme sa ich aj my rozhodli využiť.

Mračná bodov sú vysoko pamäťovo náročné súbory, preto sa mračná rozdeľujú na menšie a následne sa znižuje ich kvalita použitím Voxelizátora. Na zníženie pamäťovej náročnosti a prípravu dát na trénovanie konvolučnej siete sa používa premietnutie mračna do vodorovnej plochy. Tým vznikajú hĺbkové a intenzitové obrázky vozovky. Hlavnou výhodou použitia obrázkov je, že ich je možné spracovať tradičnými konvolučnými neurónovými sieťami.

Bol vytvorený návrh systému na odhad pozície čiary z obrázku. Využíva sa na to konvolučná neurónová sieť U-Net, ktorá vytvorí segmentáciu čiary v danom výreze. Segmentácia bola prevedená na vektorovú informáciu, ktorú bude možné vložiť do geo-informačného systému 5.5. Implementácia bola rozdelená na dve časti, backend je napísaný v jazyku C++ a frontend je v jazyku Python. Model siete U-Net je implementovaný pomocou rozhrania Keras, ktoré využíva knižnicu TensorFlow.

Model U-Net dosiahol na testovacej databáze obsahujúcej 749 dátových bodov presnosť 98,8%. So všetkých pixelov testovacej databázy boli vypočítané metriky: špecificita a senzitivita. Špecificita dosiahla hodnotu 99,5% a senzitivita 72,9%. Nižšia senzitivita je zapríčinená hlavne problémami detekcie na okrajoch obrázku a malou hrúbkou čiary. Existuje veľký priestor na ďalšie vylepšenie práce najmä vylepšenie kvality prediktora a spájania čiar 5.7. Model U-Net je potrebné doladiť tak, aby dosahoval maximálnu presnosť. Ďalej je možné skvalitniť a zväčšiť trénovaciu databázu.

Odhad vektorovej lomenej čiary bol otestovaný na troch rôznych typoch čiar. Najlepšie bola odhadnutá bočná vodiaca čiara z priemernou odchýlkou 1,68cm na centimeter dĺžky čiary. Stredová plná čiara dosiahla priemernú odchýlku 2,03cm. Najhoršie dopadol odhad stredovej prerušenej čiary s priemernou chybou 2,24cm.

Pri implementácii práce som sa stretol s problémom nízkeho kontrastu medzi vozovkou a čiarou. Problém sa mi podarilo zmenšiť, ale nie úplne odstrániť. Experimentovaním

s rôznymi nastaveniami parametrov je možné dostať lepšie výsledky. Práca s pamäťovo náročnými mračnami bodov je veľmi zdĺhavá a často trvá hodiny, z toho dôvodu boli vykonané len jednotky experimentov s rôznymi parametrami.

Výsledkom tejto práce je program odhadu čiary na vozovke, avšak zatiaľ chýba grafické užívateľské rozhranie, ktoré nebolo predmetom tejto práce. V grafickom rozhraní by užívateľ mohol označovať čiaru, ktorá ho zaujíma a program by ju predlžoval. Užívateľ by mohol kontrolovať, ako program postupuje a aká veľká je chyba. Taký program by ušetril veľké množstvo pracného manuálneho označovania čiar na vozovke.

Literatúra

[1] *Oxford Dictionary*. Oxford University Press [cit. 2021-01-18]. Dostupné z: https://www.lexico.com/definition/overfitting.

[2] *Velodyne Lidar* [online]. [cit. 2021-01-18]. Dostupné z: http://velodynelidar.com/.

[3] *Why choose Keras?* [online]. Keras documentation [cit. 2021-01-18]. Dostupné z: https://keras.io/why_keras/.

[4] Beeson, P., O'Quin, J., Gillan, B., Nimmagadda, T., Ristroph, M. et al. Multiagent Interactions in Urban Driving. *Journal of Physical Agents*. Január 2008. DOI: 10.14198/JoPha.2008.2.1.03.

[5] Bei He, Rui Ai, Yang Yan a Xianpeng Lang. Accurate and robust lane detection based on Dual-View Convolutional Neutral Network. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016, s. 1041–1046. DOI: 10.1109/IVS.2016.7535517.

[6] Charles, R. Q., Su, H., Kaichun, M. a Guibas, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, s. 77–85. DOI: 10.1109/CVPR.2017.16.

[7] Cheng, M., Zhang, H., Wang, C. a Li, J. Extraction and Classification of Road Markings Using Mobile Laser Scanning Point Clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*. 2017, zv. 10, č. 3, s. 1182–1196. DOI: 10.1109/JSTARS.2016.2606507.

[8] Geodetics. *LiDAR Intensity: What is it and What are it's applications?* [online]. Geodetics, Inc. [cit. 2021-01-01]. Dostupné z: https://geodetics.com/lidar-intensity-applications/.

[9] Goodfellow, I. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017.

[10] Guan, H., Li, J., Yu, Y., Wang, C., Chapman, M. et al. Using mobile laser scanning data for automated extraction of road markings. *ISPRS Journal of Photogrammetry and Remote Sensing*. 2014, zv. 87, s. 93 – 107. DOI: https://doi.org/10.1016/j.isprsjprs.2013.11.005. ISSN 0924-2716. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0924271613002657.

[11] Gupta, A. a Choudhary, A. A Framework for Camera-Based Real-Time Lane and Road Surface Marking Detection and Recognition. *IEEE Transactions on Intelligent Vehicles*. 2018, zv. 3, č. 4, s. 476–485. DOI: 10.1109/TIV.2018.2873902.

[12] Kriesel, D. *A Brief Introduction to Neural Networks*. 2007. Dostupné z: availableathttp://www.dkriesel.com.

[13] Kumar, P., McElhinney, C. P., Lewis, P. a McCarthy, T. Automated road markings extraction from mobile laser scanning data. *International Journal of Applied Earth Observation and Geoinformation*. 2014, zv. 32, s. 125 – 137. DOI: https://doi.org/10.1016/j.jag.2014.03.023. ISSN 0303-2434. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0303243414000816.

[14] Mitchell, T. *Machine Learning*. McGraw Hill, 1997. ISBN 0-07-042807-7.

[15] O'Shea, K. a Nash, R. *An Introduction to Convolutional Neural Networks*. 2015.

[16] Rojas, R. *The Backpropagation Algorithm*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. 149–182 s. ISBN 978-3-642-61068-4. Dostupné z: https://doi.org/10.1007/978-3-642-61068-4_7.

[17] Ronneberger, O., Fischer, P. a Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

[18] sněmovna, P. *Vyhláška, 294/2015 Sb.* November 2015. Dostupné z: https://www.psp.cz/sqw/sbirka.sqw?cz=294&r=2015.

[19] Sonka, M., Hlava, V. a Boyle, R. *Image processing, Analysis and MachineVison*. 4. vyd. Academia, 2013. ISBN 978-1-133-59360-7.

[20] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* JMLR.org. január 2014, zv. 15, č. 1, s. 1929–1958. ISSN 1532-4435.

[21] Szeliski, R. *Computer Vision: Algorithms and Applications*. Springer London, 2010. Texts in Computer Science. ISBN 9781848829350. Dostupné z: https://books.google.sk/books?id=bXzAlkODwa8C.

[22] Wen, C., Sun, X., Li, J., Wang, C., Guo, Y. et al. A deep learning framework for road marking extraction, classification and completion from mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*. 2019, zv. 147, s. 178 – 192. DOI: https://doi.org/10.1016/j.isprsjprs.2018.10.007. ISSN 0924-2716. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0924271618302855.

[23] Wikipedia. *Overfitting* [online]. [cit. 2021-01-18]. Dostupné z: https://en.wikipedia.org/wiki/Overfitting.

[24] Wu, T. a Ranganathan, A. A practical system for road marking detection and recognition. In: *2012 IEEE Intelligent Vehicles Symposium*. 2012, s. 25–30. DOI: 10.1109/IVS.2012.6232144.

[25] Yang, R., Li, Q., Tan, J., Li, S. a Chen, X. Accurate Road Marking Detection from Noisy Point Clouds Acquired by Low-Cost Mobile LiDAR Systems. *ISPRS International Journal of Geo-Information*. 2020, zv. 9, č. 10. DOI: 10.3390/ijgi9100608. ISSN 2220-9964. Dostupné z: https://www.mdpi.com/2220-9964/9/10/608.

[26] Yegulalp, S. *What is TensorFlow? The machine learning library explained* [online]. [cit. 2021-01-18]. Dostupné z: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.

Príloha A

Obsah pamäťového média

Popis adresárov a súborov uložených na pamäťovom médiu, ktoré je priložené k tejto práci. Nachádzajú sa tu zdrojové kódy, dokumentácia a vzorové dáta.

A.1 Adresárová štruktúra pamäťového média

- **doc** – Latex dokumentácia diplomovej práce a práca vo formáte pdf.
- **data** – Vzorka voxelizovaných dát, projektový súbor a vzorové VDZ.
- **src** – Všetky zdrojové súbory tejto práce.

A.2 Obsah adresára src

- **CMakeLists.txt** – Skript na preklad a linkovanie binárnych súborov.
- **README.md** – Návod na použitie programov.
- **Common.h** – Spoločné parametre všetkých programov.
- **Voxelizer.*** – C++ modul voxelizátora.
- **voxelizerMain.cpp** – Hlavný súbor voxelizátora.
- **Segmentator.*** – C++ modul na vytváranie výrezov.
- **PcdBuf.*** – C++ modul buffer mračien bodov
- **main.cpp** – Hlavný súbor generátoru anotovanej databázy výrezov.
- **Setup.py** – Skript na vytvorenie knižnice PcdBuf pre Python.
- **trainUnet.py** – Skript na trénovanie modelu U-Net.
- **predictSingleLine.py** – Skript na predikciu lomenej čiary.
- **UNetLineModel.h5** – Vopred natrénovaný model Keras siete U-Net.