VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

DETEKCE CESTY VE VENKOVNÍM PROSTŘEDÍ ZPRACOVÁNÍM OBRAZU
ROAD DETECTION IN OUTDOOR ENVIRONMENT USING IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Antónia Vrbičanová**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jiří Krejsa, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Studentka:	**Bc. Antónia Vrbičanová**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Jiří Krejsa, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Detekce cesty ve venkovním prostředí zpracováním obrazu

Stručná charakteristika problematiky úkolu:

V úloze navigace mobilních robotů ve venkovním prostředí hraje klíčovou úlohu schopnost detekce cesty ze senzorických dat robotu. Předmětem diplomové práce je realizovat detektor cesty pomocí metod zpracování obrazu z palubní kamery robotu. Detektor může být založen na klasických metodách zpracování obrazu nebo na metodách strojového učení. Předpokládá se využití předpřipravených softwarových nástrojů (OpenCV, TensorFlow). K dispozici je rozsáhlá množina dat získaných z mobilního robotu B2.

Cíle diplomové práce:

1. Prostudujte problematiku detekce cesty ve venkovním prostředí, zpracujte stručnou rešerši používaných metod.
2. Seznamte se se softwarovými balíky OpenCV a TensorFlow.
3. Vyberte vhodnou metodu detekce, implementujte a ověřte ji na reálných datech.
4. Vyhodnoťte úspěšnost metody.

Seznam doporučené literatury:

VERSCHAE R. et al: Object Detection: Current and Future Directions, Frontiers in Robotics and AI, vol 2, 2015, DOI=10.3389/frobt.2015.00029COMBS

NEETHU J. et.al: A Reliable Method for Detecting Road Regions from a Single Image Based on Color Distribution and Vanishing Point Location, Procedia Computer Science, Vol 58, 2015, pp 2-9, DOI:10.1016/j.procs.2015.08.002

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
Táto práca sa zaoberá detekciou cesty, ktorá sa nachádza vo vonkajšom prostredí. Na dosiahnutie tohto cieľa sú použité dva prístupy. Požaduje sa, aby boli oba odolné voči náhlym svetelným zmenám v obraze a použiteľné pre rôzne typy povrchu aj s malým znečistením. Prvý využíva klasické metódy spracovania obrazu. Výstupom tejto metódy sú vyznačené hranice cesty. Druhým zvoleným prístupom je použitie konvolučných neurónových sietí. V tomto prípade sa jedná o klasifikačnú úlohu a výstupom je odhad smeru cesty. Navrhlo sa niekoľko štruktúr sietí a po ich natrénovaní sa zvolila najvhodnejšia z nich. Dosiahnutá úspešnosť siete sa overila na novej testovacej množine. Obe metódy sú implementované v jazyku Python.

Summary
The Master's thesis deals with the issue of the road detection in the outdoor environment using image processing. It is highly required that the methods selected are robust to sudden light changes within the image and effective in detection of wide variety of road surfaces possibly comprising certain kinds of pollution. Two methods have been used in order to reach the desired goal. The initial method uses standard algorithms of the image processing. Main outcome of this method are highlighted road boundaries. The following method is based on convolutional neural networks. In this case we have classification task. The result of this method is the estimation of the road direction. In the whole process, several neural network structures have been designed. After the network training the most suitable one was selected. Eventually, the results have been retested using newly created test set. Both of these methods are implemented in programming language Python.

Kľúčové slová
detekcia cesty, spracovanie obrazu, konvolučné neurónové siete, strojové učenie, umelá inteligencia, Python

Keywords
path detection, image processing, convolutional neural network, machine learning, artificial intelligence, Python

VRBIČANOVÁ, A. *Detekce cesty ve venkovním prostředí zpracováním obrazu.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2020. 73 s. Vedoucí doc. Ing. Jíří Krejsa, Ph.D.

Prehlasujem, že som diplomovú prácu na tému „Detekce cesty ve venkovním prostředí zpracováním obrazu“ vypracovala samostatne s použitím literatúry uvedenej v zozname.

Bc. Antónia Vrbičanová

Ďakujem vedúcemu práce doc. Ing. Jiřímu Krejsovi, Ph.D. za ochotu a pomoc pri písaní práce. Taktiež by som rada poďakovala rodine a priateľovi za podporu počas celej doby štúdia.

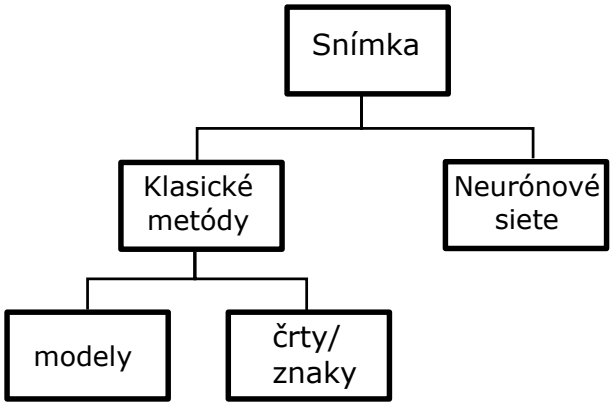
Bc. Antónia Vrbičanová

Obsah

Úvod

V súčasnosti sa v priemysle čoraz viac rozmáha práca s autonómnymi vozidlami. Jednoducho ich ide rozdeliť na dve kategórie a to na vozidlá určené na prácu vo vnútornom prostredí a na vozidlá určené na prácu vo vonkajšom prostredí. Aj keď sa to možno nejaví ako veľký rozdiel skutočnosť je iná. Zatiaľ čo vo vnútornom prostredí majú stále podmienky ako napríklad intenzitu osvetlenia, jednotnú farbu podlahy, presné hranice medzi cestou a okolím a pod., tak vo vonkajšom prostredí je potrebné zohľadňovať oveľa viac faktorov. V oboch prípadoch jednou z najzásadnejších vecí, bez ktorej nie sú schopní sa pohybovať bez ľudskej pomoci, je správne rozpoznávanie cesty. Jedným z riešení je vizuálne rozpoznávanie cesty, kedy sa zo snímok určujú v reálnom čase hranice cesty podľa vhodne zvolených algoritmov.

Rozpoznávanie cesty zo snímok je problém, ktorým sa zaoberá veľké množstvo ľudí už niekoľko desaťročí. Bolo vytvorených mnoho typov algoritmov, napísaných veľa článkov a kníh. Mnohé tieto riešenia je možné rozdeliť do niekoľkých základných kategórií, ktoré popisuje obr.č. 1. Na vstupe je snímka cesty. Je jedno či sa jedná o cestu s vodorovným dopravným značením alebo cestu neoznačenú, prístupy akým sa rieši nájdenie cesty je podobné v oboch prípadoch. Po načítaní snímky je možné cestu rozpoznávať buď pomocou strojového učenia (neurónové siete) alebo prostredníctvom klasických metód. Tieto klasické metódy ide najjednoduchšie rozdeliť na dve kategórie a to na algoritmy, ktoré používajú vo svojom riešení model (lineárny parabolický model, B - spline model, a pod.) a na algoritmy, ktoré svoje riešenie získajú na základe určitých charakteristických čŕt, ktoré snímka obsahuje (rozloženie farieb, orientácie textúr a pod.).



Obr. 1: Rozdelenie metód

Riešenia, ktoré sú založené na klasických metódach je možné zaradiť do oblasti spracovania obrazu, ktorej cieľom je zo snímok získavať nejaké vlastnosti v závislosti na type úlohy prostredníctvom vhodných metód. Riešenia, ktoré používajú umelé neurónové siete patria do oblasti strojového učenia, ktoré je podoblasťou umelej inteligencie. Do tohto oboru sa radia metódy, ktoré umožňujú programu sa učiť a následne adekvátne reagovať na rôzne vstupné hodnoty iba na základe informácií, ktoré sa naučil.

Postup práce je rozdelený na dva veľké celky pričom oba sú zamerané na nájdenie cesty prostredníctvom odlišných metód. Prvý spôsob používa na riešenie problému klasické metódy a jeho usporiadanie v práci je nasledovné. V prvej časti sa priblížia riešenia iných autorov. Následne sa daný problém analyzuje, navrhnú sa zjednodušenia a vytýčia

sa problémy, ktoré bude musieť navrhnutý algoritmus spĺňať. Potom sa pristúpi k voľbe vhodných metód. Na záver sa popíše a zhodnotí navrhnuté riešenie a dosiahnuté výsledky. Druhá časť práce bude venovaná neurónovým sieťam. Zásadnou vecou, ktorú je potrebné pred celým procesom učenia zvoliť je správna reprezentácia cesty. Na úvod sa popíšu neurónové siete a priblíži sa proces jej učenia. Následne je usporiadanie tejto časti obdobné ako pri použití klasických metód. V prvom kroku sa analyzuje problém. Následne sa zvolí vhodná reprezentácia cesty a pristúpi sa k roztriedeniu dát do konkrétnych kategórií. Potom je potrebné pre dosiahnutie vysokej presnosti po natrénovaní, zvoliť vhodnú štruktúru siete. Navrhne sa niekoľko typov, ktoré sa otestujú a vyberie sa z nich tá najlepšia. Na záver sa zhodnotia dosiahnuté výsledky.

1 Formulácia problému a ciele riešenia

Nájdenie cesty je problém, ktorý je zložitý z dôvodu premenlivosti veľkého počtu premenných, ktoré je potrebné pri návrhu takéhoto typu algoritmu zohľadniť. Práve z tohto dôvodu je nutné presne naformulovať problém a vytýčiť presné ciele, ku ktorým bude postupne celá práca smerovať.

Pred formuláciou problému je potrebné nadefinovať základný pojem a to je nájdenie cesty. Pod týmto slovným spojením sa rozumie vyznačenie buď celej alebo časti cesty, za predpokladu, že sa nachádza pred pozorovateľom, ktorý je taktiež na pozorovanej ceste.

Všeobecne je problém nájdenia cesty možné formulovať ako vytýčenie cesty na snímke ľubovoľným vhodným spôsobom pričom by mala byť použitá metóda dostatočne robustná.

Zvolili sa dva prístupy ako sa bude k danému problému pristupovať a to prostredníctvom klasických metód a použitím neurónových sietí. Cieľom v oboch prípadoch bude správne nájdenie a vyznačenie cesty. Vstupnými dátami pre oba prístupy budú snímky.

V prvom prípade, teda pri použití klasických algoritmov, bude správnemu nájdeniu cesty predchádzať vhodná voľba algoritmov spracovania obrazu, ktoré bude potrebné pred použitím preštudovať. Následne voľba metód je závislá na okolitom prostredí, v ktorom sa bude cesta nachádzať. Ako výsledok z tejto metódy sa očakávajú vyznačené hranice cesty.

Pri neurónových sieťach bude na dosiahnutie dobrých výsledkov potrebné zvoliť, vhodnú reprezentáciu cesty, typ siete a následne dosiahnuť dostatočne vysokú úspešnosť pri jej trénovaní.

2 Rozpoznanie cesty klasickými metódami

Cestné komunikácie sú vo všeobecnosti veľmi rozmanité. Pokiaľ sa autonómne vozidlo pohybuje po ceste s vodorovným dopravným značením je možné aby sa pohybovalo podľa tohto systému značenia. Existuje veľa druhov algoritmov, ktoré sa zaoberajú práve detekciou čiar. Je možné ich rozdeliť do dvoch základných kategórií, a to na metódy založené na modeli [46][18][16][45] a na metódy založené na určitých vlastnostiach obrazu [36] [20]. Jeden z najjednoduchších spôsobov ako rozpoznať značenie popisuje Surma [41]. Vo svojom riešení odstráni šum pomocou Gaussovho filtru, potom deteguje hrany. Na to aby určil, ktoré zo získaných hrán vyjadrujú cestu, použije masku tzv. oblasť záujmu. Pokiaľ sa čiary nachádzajú v tejto oblasti aplikuje Houghovu transformáciu. Riešenie je použiteľné len pri určitých podmienkach, jednotná intenzita svetla, žiadne iné vodorovné značenie na ceste a ani oproti idúce autá. Vo všetkých týchto prípadoch by použitie Houghovej transformácie zlyhalo a výsledok by bol nepoužiteľný. S iným riešením prišli Shi, Kong a Zheng [36], ktorých algoritmus je založený na charakteristických črtách. Podstatou je detekcia všetkých charakteristických čŕt z obrázka, následné posúdenie oblastí, v ktorých boli detegované čiary na základe získaných vlastností. Na záver sa aplikuje Houghova transformácia na vyznačenie výsledných čiar. Sun a spol. [40] vytvorili algoritmus, ktorý je použiteľný aj pri zmenách prostredia. Podstatou je extrakcia hrán založená na štatistike histogramu gradientu. Wang, Teoh a Shen [45] v práci vytvorili tzv. B-Snake model jazdného pruhu. Ide o model, ktorý použitím B-Splinu vytvorí ľubovoľný tvar z množiny kontrolných bodov. Tieto body sú určené pomocou algoritmu CHEVP (Canny/Hough Estimation of Vanishing Points), ktorý snímok rozdelí na niekoľko podoblastí. Pre každú oblasť sa určí tzv. bod zániku, pomocou ktorého sa následne vypočíta výsledný kontrolný bod pre danú oblasť. Táto metóda je zaujímavá v tom, že je dostatočne robustná voči šumu, tieňom na ceste a zmenám svetla.

Ako sa má vozidlo pohybovať vo chvíli, keď na ceste nie je žiadne značenie? Tento druh ciest pri rozpoznávaní zo snímkov prináša značné komplikácie. Cesty majú rôzne nedokonalosti, rôzne trhliny, jamy, neštandardné značenia alebo nápisy. Taktiež nemusia mať homogénny povrch a môžu mať rôzne druhy znečistenia ako listy, odpadky, lavičky a za určitý druh znečistenia sa môžu považovať dokonca aj ľudia. Práve z týchto dôvodov je detekcia cesty, ktorá nemá vodorovné dopravné značenie komplikovanejšia, ale aj napriek tomu vzniká mnoho algoritmov, ktoré sú robustné proti rôznym typom šumu.

Zhaozi Zu a spol.[49], na rozpoznanie vidieckej cesty využili tzv. bod zániku. Celé riešenie spočívalo v tom, že sa vypočítali orientácie textúr pomocou Sobelovho filtra. Zo získaných orientácií sa následne zvolil bod zániku. Overovali sa všetky čiary, ktoré týmto bodom prechádzali. Vybrala sa najdominantnejšia z nich a tá sa prehlásila za hranicu cesty. Body, z ktorých sa skladala prvá hranica cesty sa prehlásili za nové body zániku. Druhá hranica cesty sa určila obdobne ako prvá. Vybrali sa najdominantnejšie čiary, ktoré prechádzali novými bodmi zániku. Z týchto čiar sa následne určila požadovaná druhá hranica cesty. Nevýhodou tohto druhu riešenia je, že je použiteľný len pri fotografiách, kde cesta v diaľke zaniká.

Kong a spol [19] vo svojom článku prestavili metódu, ktorá pozostáva z dvoch krokov, odhadu bodu zániku a segmentáciu cesty. Základ celého výpočtu je postavený na tzv. Gabor wavelets[6][35][28], ktoré sa používajú na analýzu textúr a ich podstata je založená

na hľadaní špecifických frekvencií v rôznych smeroch v dopredu zvolenej oblasti záujmu. Na určenie bodu zániku sa vypočítajú orientácie textúr každého pixlu, ktoré sú rozšírené o váhu. Pixly s nízkou hodnotou váh sa nepovažujú za relevantné a tým sa zaistí, že výsledné body zániku na snímkach sú určené presnejšie ako pri čistom výpočte orientácie textúr pixlov. Detekcia hrán, ktoré vychádzajú z bodu zániku, je založená na OCR (Orientation Consistency Ratio). Následne je na segmentáciu cesty potrebné zvoliť najdominantnejšie hrany. Veľkou výhodou tejto metódy je použitie v rôznych typoch prostredia.

S iným riešením prišli Cristóforis a spol. [7]. Na rozpoznanie cesty využívajú segmentáciu obrazu. Ako prvé sa detegoval horizont. Oblasť, ktorá sa nachádzala pod ním sa zanechala a všetko nad sa odrezalo z dôvodu zjednodušenia celého obrazu. Následne sa na zredukovanie šumu použil mediánový filter. Získala sa tzv. mapa segmentov, ktorá pozostávala z pixlov rozriedených do skupín na základe podobných vlastností. Na rozpoznanie cesty sa použila tzv. metóda ROI (rectangular region of interest), ktorá z obrazu vyreže oblasť, kde sa s najväčšou pravdepodobnosťou cesta nachádza. Potom sa porovná zvolená oblasť s mapou segmentov a na základe podobností sa rozhodne, kde sa cesta nachádza a kde nie. Hranice cesty sa nakoniec vyznačia pomocou kontúr. Táto metóda sa zdá byť veľmi efektívna, ale s jednoznačnosťou to nie je možné povedať, pretože výsledky, ktoré autori prezentujú majú jednotnú intenzitu svetla. Je otázne, či by pri väčšom množstve tieňov, prípadne trhlín na ceste dávala stále dobré výsledky.

S ďalším riešením ako rozpoznať cestu bez vodorovného dopravného značenia popísali vo svojej práci Huang a spol.[15]. Zdrojom informácií o ceste je oblasť, ktorá sa nachádza pred kamerou, v spodnej časti snímky. Táto oblasť sa prevedie do HSV modelu. Následne sa získaná snímka postupne prechádza po riadkoch, ktoré sú rozdelené na niekoľko segmentov. Hraničné body týchto segmentov sa porovnajú s hodnotami predpokladanej cesty a pokiaľ hodnoty oboch koncových bodov odpovedajú hodnotám cesty, daný segment sa prehlási za časť cesty. Nakoniec sa pristúpi ku korekcii segmentov v jednotlivých riadkoch a získané body preložia krivkou.

V nasledujúcich kapitolách bude postupne popísaný vlastný návrh riešenia nájdenia cesty v obraze.

3 Analýza problému

V rešeršnej časti viď. 2 bolo popísaných viacero možných riešení nájdenia cesty v obraze. Z ich rôznorodosti je zjavné, že tomuto problému sa venuje množstvo ľudí a stále sa hľadá riešenie, ktoré by bolo natoľko robustné, aby ho bolo možné použiť na riadenie autonómnych vozidlách.

V tejto časti sa problém detekcie cesty analyzuje, navrhnú sa určité zjednodušenia a vytýčia sa problémy, ktoré bude potrebné do návrhu algoritmu zahrnúť. V prvom kroku je potrebné podrobnejšie analyzovať cestu a okolité prostredie a až na základe získaných poznatkov prijať určité zjednodušenia, ktoré budú následne použité pri návrhu algoritmu.

Všeobecne existuje nespočetne veľa ciest, a všetky sa od seba odlišujú. Základný rozdiel, tak ako bolo spomenuté už v rešeršnej časti, je v tom, či sa daná cesta nachádza vo vnútornom alebo vonkajšom prostredí. Pokiaľ vo vnútri tak intenzita osvetlenia je vo väčšine prípadov jednotná, cesta je jednoznačne oddelená od okolitého prostredia a jej farba je homogénna. No pri ceste nachádzajúcej sa vo vonkajšom prostredí sa všetko značne komplikuje. Osvetlenie cesty nie je ani zďaleka vo väčšine prípadov jednotné. Vplyvom okolitého prostredia vznikajú ostré prechody medzi miestami, kde sa tieň nachádza a kde nie. Taktiež povrch môže byť veľmi rôznorodý. Pokiaľ sa jedná o menšie chodníky, tie sú buď tvorené z kociek, štrku alebo čistej zeme, zatiaľ čo väčšie cesty z asfaltu. Ďalším problémom je vplyv ročného obdobia, zatiaľ čo v lete je cesta čistá a okolité prostredie je dosť odlišné, v jeseni alebo zime sa situácia znova dosť komplikuje. Vo vymenovávaní všetkých prípadov, ktoré je potrebné brať do úvahy, aby bol algoritmus nájdenia cesty dostatočne robustný je tak mnoho, že keby sa taký aj podarilo vytvoriť čas, ktorý by bol potrebný na jej detekciu by bol tak dlhý, že vo výsledku by ho nebolo možné použiť.

Z vyššie uvedených dôvodov sa pristúpilo k určitým zjednodušeniam, vďaka ktorým bude hľadanie cesty trochu menej náročné. V prvom rade sa vzalo v úvahu okolité prostredie. Keďže snímky, ktoré sú k dispozícií boli vytvorené v parku, bude sa predpokladať, že okolité prostredie bude tvorené prevažne stromami a trávou. Teda algoritmus nebude zahŕňať prípady, kde okolie bude tvorené budovami. Ďalej sa bude predpokladať, že snímky sú zaznamenané z miesta na ceste a pre celkové zjednodušenie obrazu sa odstráni niekoľko pixlov z hornej časti snímky.

Po prijatí zjednodušení je potrebné, pre úspešné nájdenie cesty, v algoritme vyriešiť problém náhlej zmeny intenzity svetla. Taktiež bude potrebné algoritmus navrhnúť tak, aby fungoval pri detekcií rôznorodých ciest, či už tvarovo alebo materiálovo. Taktiež by mal byť odolný voči malému znečisteniu na ceste. Pojem znečistenie zahŕňa odpadky, lístie prípadne označenie na ceste. Mal by byť dostatočne presný a čo najviac jednoduchý aby sa zaistila robustnosť.

V nasledujúcich častiach budú postupne popísané metódy, ktoré je možné použiť pri návrhu vlastného algoritmu a následne sa pristúpi k samotnému riešeniu.

4 Voľba metód riešenia problému

Fotoaparáty, kamery a iné zariadenia na zachytávanie obrazu sa používajú už niekoľko desaťročí. S veľkým pokrokom vo výpočetnej technike je možné obraz, ktorý bol týmito zariadeniami získaný ďalej spracovávať a získavať z neho informácie, ktoré sú pre fungovanie niektorých zariadení zásadné. Vedná disciplína, ktorá sa snaží práve prostredníctvom počítačových prostriedkov napodobniť ľudské videnie a vnímanie sa nazýva počítačové videnie. Jej podstata spočíva v spracovaní obrazu (image processing), ktoré je možné podľa [38] rozdeliť na niekoľko krokov a to, snímanie, predspracovanie, segmentáciu, detekciu príznakov, klasifikáciu a rozpoznávanie objektov, porozumenie scény, rekonštrukciu a 3D zobrazenie. Prvé tri časti sa radia do tzv. nízko-úrovňového spracovania obrazu, výstupom ktorého sú extrahované znalosti. Zvyšné časti patria do tzv. vysoko-úrovňového spracovania obrazu.

Pre každý krok v spracovaní obrazu, bolo vytvorených veľké množstvo algoritmov, z ktorých je každý zameraný na nájdenie inej vlastnosti v obraze. Na dosiahnutie správneho výsledku je preto zásadná práve voľba metód, ktorej ale predchádza správne pochopenie riešenej sústavy.

Usporiadanie nasledujúcich podkapitol je spravené s ohľadom na všeobecný postup pri spracovaní obrazu a ich je obsah v stručnosti nasledovný. V prvej podkapitole sa popíše riešená sústava, v tomto prípade snímka. V ďalších častiach sa popíše snímanie obrazu a farebné modely, pomcou ktorých sa získa farebný obraz. Následne sa prejde k predspracovaniu, ktorého podstatou je odstránenie prípadne potlačenie informácií nežiadúcich pre ďalšie spracovanie. Do tejto podkapitoly z väčšej časti patria metódy zamerané na odstránenie rôznych druhov šumu. Potom sa prejde ku segmentácií, ktorej podstatou je rozdelenie obrazu na časti podľa spoločných vlastností. Na záver sa uvedie software,v ktorom sa celé riešenie realizovalo.

4.1 Riešená sústava

V tomto prípade bude riešenou sústavou snímka, ktorú je možné chápať ako sieť veľmi malých štvorcov, ktoré sa nazývajú pixly. Tieto pixly sú uložené tak, že vo výsledku tvoria dvojrozmernú maticu. Pričom počet pixlov v riadku vyjadruje šírku a počet pixlov v stĺpci výšku obrázka. Na to aby mohli reprodukovať obraz, ktorý vnímame zrakom je potrebné aby každá jednotka mala zakódované ďalšie parametre ako napr. farbu, jas a pod.

Všetky snímky, ktoré budú použité v práci sú rozmeru 480×360 pixlov. Taktiež na všetkých z nich je zaznamenaná cesta.

4.2 Snímanie

Obraz, ktorý vieme získať z fotoaparátu, kamery alebo iných senzorov je v súčasnosti niečo tak prirodzené, že väčšinu ľudí sa ani nenapadne zamyslieť nad tým ako obraz vlastne vzniká. Pri získavaní informácií z obrazu je táto znalosť potrebná či už z dôvodu správnej voľby snímača alebo porozumeniu toho z čoho je zložený a čo vyjadruje.

Signály ktoré vstupujú do zariadenia určeného na snímanie môžu byť rôzneho druhu, keďže je práca zameraná na získavanie informácií zo snímok, tak v tomto prípade je na

vstup snímača privádzané svetlo. Jeho intenzita je následne prevedená na elektrický signál prostredníctvom optických snímačov CMOS alebo CCD, ktoré sú zložené z veľkého počtu buniek, na ktoré dopadajú fotóny. Podrobnejšie vysvetlenie ako tieto snímače fungujú je popísané v [47]. Získaný analógový elektrický signál sa následne prevedie pomocou AD prevodníku na digitálny signál. Výsledkom celého procesu snímania je matica o rozmere $n \times m$. Počet prvkov matice odpovedá počtu buniek v optickom snímači a v získanom obraze je to jeho rozlíšenie. Hodnoty, ktoré môže nadobúdať každý prvok matice, závisia na rozlíšení AD prevodníku. Napríklad ak sa použije 8-bitový prevodník tak na výstupe je možné získať hodnoty v rozsahu $0 - 255$. Pokiaľ, výsledkom má byť farebný obraz tak každý prvok matice sa musí skladať z troch zložiek. Používa sa tzv. RGB farebný model, ktorý pozostáva z troch farebných zložiek (modrej, zelenej, červenej) a podrobnejšie je popísaný v nasledujúcej časti.

4.3 Interpretácia farieb

Veľká časť interpretácie nášho sveta, ako vyzerá, ako ho vidíme ako jednotlivci závisí na vnímaní farieb, ktoré nás každodenne obklopujú. Keď sa položí otázka, čo je farba nie všetci budú poznať odpoveď. V [12] je definovaná z fyzikálneho hľadiska ako vlnová dĺžka z viditeľného spektra, ktorá keď vstúpi do oka vyvolá reakciu, tzv. vnem. Teda farby sú len rôzne vlnové dĺžky svetla, ktoré dopadajú na sietnicu a mozog ich vie následne spracovať. Pri spracovaní obrazu je rozpoznanie farieb zásadné z dôvodu správnej interpretácie častí obrazu. Na vysvetlenie ako sa rozpoznávajú farby v digitálnom svete je potrebné ako prvé objasniť pojmy farebný model, farebný priestor a gamut.

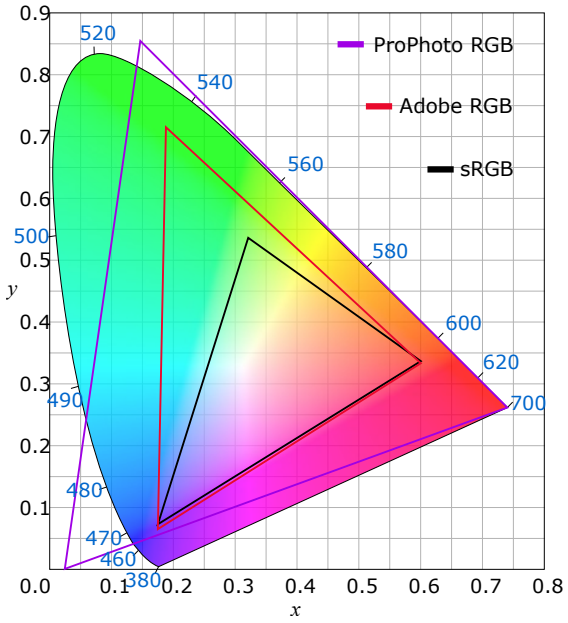
Farebný model: Abstraktný matematický model, ktorý je tvorený niekoľkými základnými farbami vyjadrenými číslami, z ktorých je možné správnou kombináciou vytvoriť iné farby.

Farebný priestor: Priestor, v ktorom je obsiahnutá každá farba ako diskrétny bod v zvolenom matematickom modele.

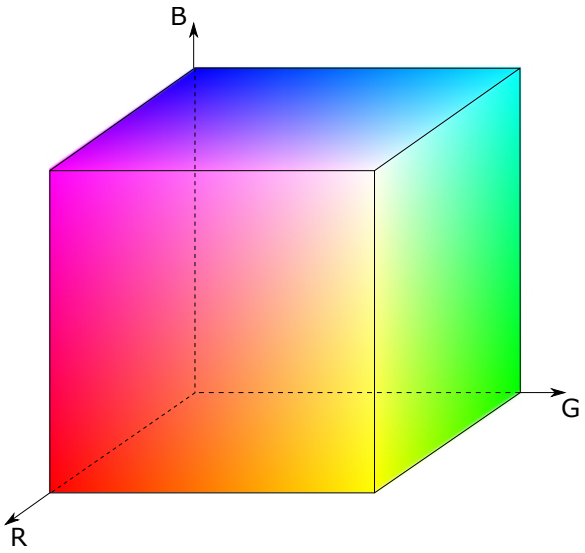
Gamut: Rozsah farieb, ktoré je možné vo vybranom priestore dosiahnuť.

Najväčším farebným priestorom, je priestor nášho viditeľného spektra, všetky ostatné, umelo vytvorené priestory, sú väčšinou jeho podmnožiny. Niektoré môžu zasahovať aj mimo viditeľné spektrum viď obr. č. 4.1. V tomto diagrame sú čisté chromatické farby prevedené do súradníc x, y. Výsledkom je tvar podkovy, tzv. spektrálny locus a čiara, ktorá spája koncové body podkovy sa nazýva purpurová línia. [39]

Jedným z najpoužívanejších modelov je práve RGB model, ktorý sa definuje ako aditívny farebný model, pozostávajúci z troch základných zložiek, a to z červenej (red), zo zelenej (green) a z modrej (blue). Tieto hodnoty je možné získať z chromatického diagramu prostredníctvom vhodnej transformácie. RGB model sa zobrazuje v priestore prostredníctvom kocky v kartézskom súradnicovom systéme, kde jednotlivé osi popisujú základné farby RGB modelu, obr. č. 4.2. Pokiaľ sa použije 24 bitové zobrazenie, teda 8 bitov na každý kanál, tak základné farby môžu nadobúdať hodnoty v rozsahu 0 - 255. Pre predstavu, takéto zobrazenie poskytuje približne 16,6 milióna rôznych farieb. Keď všetky zložky budú nulové (0, 0, 0), tak výsledkom bude čierna, a pokiaľ nadobudnú hodnoty (255, 255, 255) výsledkom bude biela. [30]



Obr. 4.1: Chromatický diagram



Obr. 4.2: Priestor RGB

4.4 Predspracovanie

V obraze sa nachádza veľa informácií, niektoré sú pre ďalšie spracovanie užitočné a niektoré je na dosiahnutie dobrých a správnych výsledkov potrebné potlačiť prípadne odstrániť. Medzi nepotrebné informácie patria vlastnosti v obraze, ktoré sú pre ďalšie spracovanie nadbytočné. Taktiež sem patria rôzne nedokonalosti v obraze, ktoré nemusia byť viditeľné ľudským okom, no pri ich ponechaní nie je možné pristúpiť k segmentácií obrazu. Väčšinou sa prejavujú vo forme šumu. Časť spracovania obrazu, ktorá sa zaoberá týmito problémami sa nazýva predspracovanie alebo tzv. preprocesing a jej hlavnými cieľmi sú teda odstránenie skreslenia, potlačenie šumu a zvýraznenie požadovaných rysov v obraze. Je dôležité si uvedomiť, že pri predspracovaní obrazu sa žiadne nové informácie nezískavajú, ale určité vlastnosti sa len potlačujú alebo odstraňujú aby boli zvýraznené iné.

Bolo vytvorených mnoho algoritmov, ktoré sa venujú tomuto problému a podľa typu úpravy je možné ich rozdeliť do viacerých kategórií:

1. Bodové jasové transformácie
2. Geometrické transformácie
3. Lokálne predspracovanie
4. Matematická morfológia

V nasledujúcich častiach budú postupne popísané metódy podľa kategórie, do ktorej patria. Informácie o jednotlivých metódach popísaných v častiach 4.4.1–4.4.4 boli čerpané z [38][14][42][9].

4.4.1 Bodové jasové transformácie

Do tejto časti predspracovania patria metódy, ktoré transformujú jediný vstupný bod na výstupný bod. Patria sem algoritmy jasová korekcia, transformácia jasovej stupnice a ekvalizácia histogramu. Prvé dva algoritmy sú zamerané na odstránenie chýb spôsobených snímacím zariadením, ktorého výstupom je nerovnomerne osvetlený obraz. Posledná spomenutá metóda ekvalizácia histogramu je zameraná na zvýraznenie pixlov popisujúcich objekt v snímke.

Jasová korekcia

Podstatou tejto metódy je korekcia obrazu, teda odstránenie systematickej chyby, ktorá vzniká pri snímaní obrazu snímacím zariadením. Nech i, j je poloha vstupného bodu obrazu $g(i, j)$ a nech $f(i, j)$ je výstupný skreslený bod obrazu. Potom hodnotu transformácie $e(i, j)$, ktorá vzniká pri procese snímania je možné vyjadriť ako

$$f(i, j) = e(i, j)g(i, j) \tag{4.1}$$

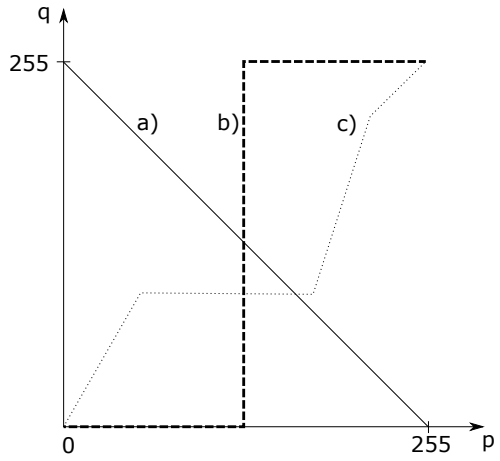
Hodnotu $e(i, j)$ je možné získať tak, že sa bude snímať plocha s konštantným jasom c. Potom sa pôvodný vzťah upraví do tvaru (4.2), z ktorého je možné získať hodnotu transformácie $e(i, j)$:

$$f_c(i, j) = e(i, j)c \tag{4.2}$$

Tento postup sa vykoná pre každý bod v obraze a tým sa získa tzv. korekčná transformačná matica. Pokiaľ sa podmienky (expozičný čas a teplota) pre snímanie zmenia je potrebné túto maticu vypočítať znova.

Transformácia jasovej stupnice

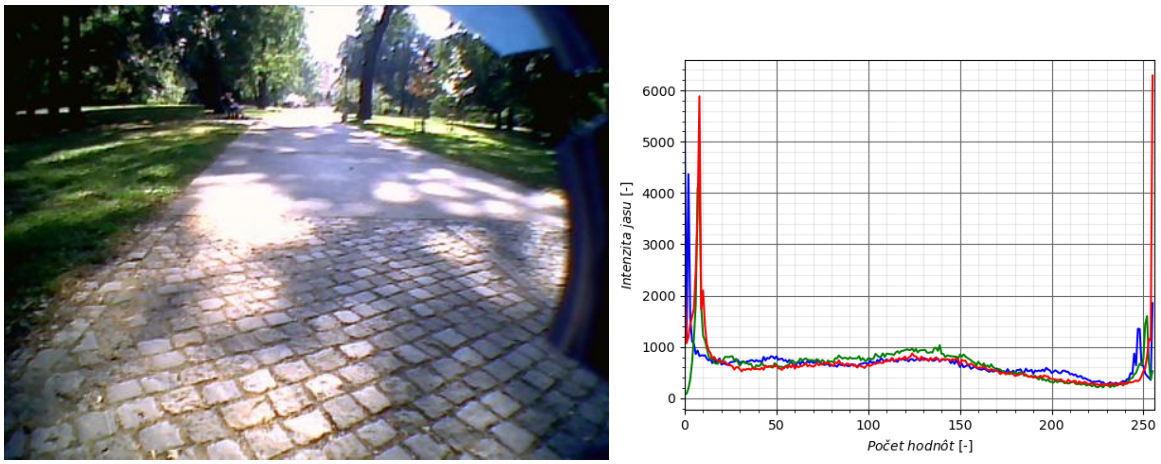
Táto metóda popisuje prevod vstupnej jasovej hodnoty na výstupnú prostredníctvom známej funkcie. Na rozdiel od jasovej korekcie, kde sa pre každý bod počítala transformačná hodnota $e(i, j)$ v tejto metóde je podľa jasovej hodnoty vstupného bodu a typu funkcie určený výstupný bod. Obr. č. 4.3 vyjadruje funkcie, ktoré sú bežne používané na transformáciu, os p vyjadruje vstupnú jasovú stupnicu a os q výstupnú jasovú stupnicu.



Obr. 4.3: Transformácia jasovej stupnice - a) Inverzná transformácia b) Zvyšovanie kontrastu c) Prahovanie

Vyrovnanie histogramu

Prvým krokom k vysvetleniu, čo je vyrovnanie inak nazývaná aj ekvalizácia histogramu, je objasnenie, čo je to histogram. Všeobecne sa dá definovať ako grafické znázornenie dát pomocou stĺpcového grafu, kde x-ová os vyjadruje počet zložiek a y-ová os vyjadruje počet hodnôt spadajúcich do daného intervalu (zložky). Pri fotografiách sa šírkou histogramu rozumie počet jasových úrovní. Keď sa vezme šedotónna snímka a spraví sa jej histogram tak pixely, ktoré majú hodnoty blížiace sa nule budú zastúpené v ľavej časti grafu zatiaľ, čo pixely, ktoré budú mať hodnoty blížiace sa bielej budú zastúpené v pravej časti grafu.



(a) Pôvodný obrázok (b) Histogram

Obr. 4.4: Histogram farebného obrázku

Problém vzniká vtedy, keď všetky pixly v obraze majú podobnú jasovú hodnotu. Objekty v snímke je ťažké rozoznať od okolia. Vtedy sa pristupuje k ekvalizácií histogramu. Ide o proces, v ktorom sa upraví kontrast obrazu.

Problém vznikajúci pri ekvalizácií je, že okrem zvýraznenia žiadúcich objektov môže dôjsť aj k zvýrazneniu nežiadúceho šumu a tým dôjde k potlačeniu informácií, ktoré sú pre ďalšie spracovanie dôležité .

4.4.2 Geometrické transformácie

Pri snímaní obrazu môžu vznikať geometrické deformácie, ktoré sa prejavujú formou skreslenia. Vznikajú vtedy, keď snímacie zariadenie nie je umiestnené kolmo na snímaný objekt a na ich odstránenie sa používa práve geometrická transformácia. Je definovaná tzv. transformačnými rovnicami, ktoré popisuje vzťah (4.3).

$$x_0 = T_x(x, y) \qquad y_0 = T_y(x, y) \tag{4.3}$$

Kde T_x a T_y sú transformačné matice, $x,\ y$ sú súradnice bodu vstupného obrazu a $x_0,\ y_0$ sú súradnice bodu transformovaného obrazu.

Všeobecne geometrická transformácia pozostáva z dvoch krokov a to z transformácie súradníc pixlov vstupného obrazu a jasovej interpolácie.

4.4.3 Lokálne predspracovanie

Cieľom tejto časti predspracovania obrazu je odstránenie šumu a nájdenie hrán. Metódy spadajúce do tejto kategórie sa teda delia na dve skupiny, a to na vyhladzovanie a detekciu hrán.

Podstatou vyhladzovania, známeho aj ako filtrovanie, je potlačenie vyšších frekvencií, čo má za následok odstránenie šumu v obraze, ale aj rozostrenie hrán. Existuje viacero typov šumu, napr. biely, aditívny, multiplikatívny, Gaussov, Salt and Pepper a iné. Podrobnejšie sú popísané v [5]. Každý z nich je vhodné odstraňovať iným typom filtru. Tie najvýznamnejšie a teda aj najčastejšie používané filtre budú popísané v nasledujúcom texte.

Podstatou detekcie hrán je nájdenie miest v obraze s veľkou veľkosťou gradientu. Na rozdiel od vyhladzovania, v tejto časti dôjde k zvýrazneniu vyšších frekvencií obrazovej funkcie (hrany) a časti, v ktorých dochádza k pomalým zmenám sa odstránia. Nevýhodou použitia metód tohto typu je, že môže dôjsť k zvýrazneniu šumu.

Filtrácia

Vyhladzovanie obrazu slúži na odstránenie šumu z obrazu, tak že hodnota jasu každého pixlu sa upraví na základe jeho okolia. Toto okolie má najčastejšie rozmer 3×3 alebo 5×5 pixlov.

Základnou metódou, ktorá sa používa je diskrétna konvolúcia, ktorá je popísaná rovnicou (4.4).

$$f(x,y) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} c(m,n) g(x-m, y-n) \tag{4.4}$$

Kde $f(x,y)$ je výstupný bod obrazu, $c(m,n)$ je konvolučná maska o rozmere $2M+1 \times 2N+1$, M, N vyjadruje rozmer a $g(x-m, y-n)$ je vstupný bod obrazu. Teda, hodnota jasu bodu výstupného obrazu sa vypočíta ako súčin koeficientov konvolučnej masky a hodnoty jasov bodov vstupného obrazu odpovedajúce okoliu. Podľa typu konvolučnej masky, inak povedané koeficientov, ktoré obsahuje sa rozlišuje viacero typov filtrov.

Medzi najjednoduchší z nich patrí priemerovanie. Jedná sa o filter, ktorý výstupnú jasovú hodnotu pixlu vypočíta ako aritmetický priemer hodnôt jasu bodov, ktoré spadajú do zvoleného okolia. Konvolučná maska má v tomto prípade tvar jednotkovej matice, rovnica (4.5).

$$c = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{4.5}$$

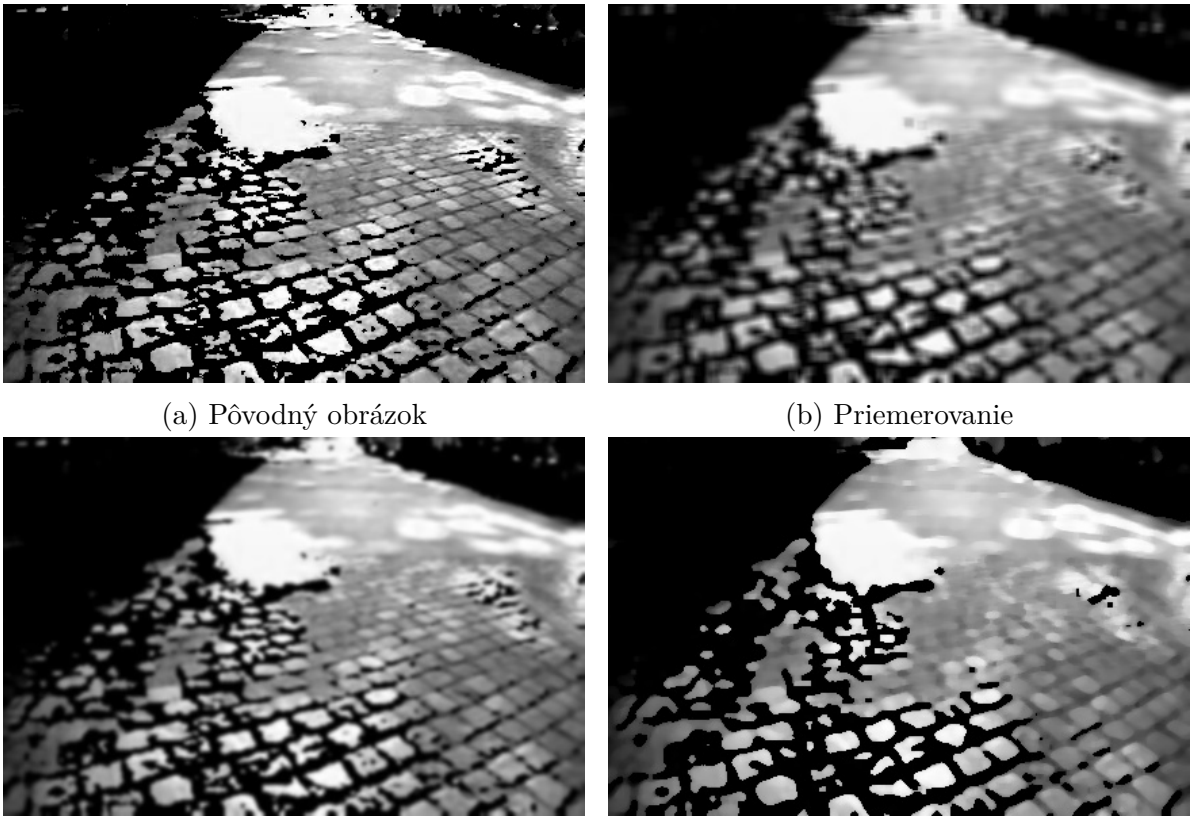
Priemerovanie sa využíva na odstránenie pixlov, ktoré v obraze majú rozličnú jasovú hodnotu ako ich okolie. Tento proces je síce efektívny, ale je potrené zdôrazniť, že pri väčšom rozmere obrázka môže byť časovo náročný.

Ďalším filtrom, ktorý slúži na vyhladenie obrazu je tzv. Gaussov filter. Na odstránenie šumu taktiež používa konvolučnú masku, ktorej hodnoty sú dané podľa Gaussovho rozdelenia, rovnica (4.6).

$$c = \frac{1}{273}\begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} \tag{4.6}$$

Pri použití Gaussovho filtra hodnoty jasu stredového bodu a jeho najbližšieho okolia (štyria susedia) majú oveľa vyššiu váhu ako ostatné, čo má za následok zvýraznenie stredového bodu, ale aj rozmazanie obrazu.

Mediánový filter usporiada hodnoty, ktoré spadajú do okolia aktuálne spracovávaného pixlu, podľa veľkosti, a následne sa z týchto hodnôt zvolí medián. Táto získaná jasová hodnota sa prehlási za výstupnú jasovú hodnotu spracovávaného pixlu. Výhodou tohto filtru je, že výsledná hodnota pixlu je stredná hodnota postupnosti a tým pádom extrémne hodnoty na okrajoch neovplyvnia výsledok na rozdiel od priemerovania. Samozrejme má to aj svoje nevýhody, pretože v prípade napr. tenkých čiar, dôjde k ich porušeniu.



(a) Pôvodný obrázok (b) Priemerovanie

(c) Gaussov filter (d) Mediánový filter

Obr. 4.5: Príklady vyššie popísaných filtrov. a) Pôvodný obrázok b) Priemerovanie, veľkosť použitej masky 7×7 c) S použitím Gaussovho filtra veľkosť použitej masky 7×7 d) Mediánový filter

Hľadanie hrán

Pri tvorbe tejto časti sa okrem literatúry uvedenej v úvode 4.4 čeralo aj z [13][11]. Aby bolo možné popísať postup ako sa detegujú hrany v obraze je potrebné najskôr vysvetliť pojmy gradient a hrana.

Gradient spojitej funkcie f je vektor, ktorého jednotlivé zložky sú tvorené parciálnymi deriváciami. Vyjadruje smer najväčšieho nárastu a jeho veľkosť a smer sa určia podľa rovníc (4.7, 4.8).

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{4.7}$$

$$\psi = tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right) \tag{4.8}$$

Kde $f(x,y)$ je obrazová funkcia.

Hrana je daná vlastnosťami pixlu a jeho okolia. Ide o vektor s dvomi zložkami, ktoré sa získajú ako parciálne derivácie obrazovej funkcie $f(x,y)$. Vyjadrujú veľkosť a smer, pričom veľkosť je totožná s veľkosťou gradientu a smer je kolmý na smer gradientu. Teda miesta v obraze, v ktorých dochádza k najväčšej zmene jasovej hodnoty sa dajú pokladať za miesta s najväčším nárastom obrazovej funkcie $f(x,y)$.

Keďže získaný obraz nie je spojitý, ale je tvorený jednotlivými pixlami je potrebné parciálne derivácie, ktoré sa využívajú na výpočet gradientu aproximovať, a to pomocou diferencií (rovnice 4.9 a 4.10).

$$\Delta_x f(x,y) = f(x,y) - f(x-1,y) \tag{4.9}$$

$$\Delta_y f(x,y) = f(x,y) - f(x,y-1) \tag{4.10}$$

Derivácia obrazovej funkcie sa dá aproximovať aj pomocou operátorov. Využije sa na to diskrétna konvolúcia, ktorej masky sa budú považovať za operátory aproximujúce prvú deriváciu obrazovej funkcie. Keďže operátory nie sú invariantné voči rotácií, tak každý operátor bude obsahovať toľko konvolučných masiek, koľko smerov rozlišuje. Výsledná celková odozva sa následne určí ako súčet všetkých natočení masiek.

Medzi najpoužívanejšie operátory patrí Robertsov operátor, ktorý využíva iba okolie bodu o veľkosti 2×2, vďaka čomu sa pokladá za výpočtovo nenáročný operátor. Konvolučné masky, ktoré používa vyjadruje (4.11)

$$k_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad k_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{4.11}$$

Z tvaru konvolučných masiek vyplýva, že hrany sa hľadajú na diagonálach. Keďže sa na nájdenie miest s najväčším nárastom obrazovej funkcie používa tak malé okolie, veľkou nevýhodou pri použití tohto operátoru je práve citlivosť na šum.

Ďalším operátorom, ktorý aproximuje prvé derivácie, je Sobelov operátor. Tento typ operátora využíva okolie bodu o veľkosti 3×3. Celkový počet používaných konvolučných masiek je osem. V (4.12) sú vyjadrené len prvé tri masky, pričom ostatné je možné získať ich pootočením. Najčastejšie sa používajú masky k_1 a k_3, pomocou ktorých sa hľadajú vodorovné a zvislé hrany.

$$k_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad k_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad k_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.12}$$

Nevýhodou použitia tohto operátoru je väčšia časová náročnosť na výpočet oproti Robortsovmu operátoru z dôvodu použitia väčšieho okolia. Medzi výhody patrí menšia citlivosť na šum.

Existuje niekoľko ďalších typov operátorov, napríklad Robinsonov, Prewitovej a iné, ktorých presnejší popis je možné nájsť v [11].

Druhá metóda, ktorá slúži na nájdenie hrán v obraze, je založená na nájdení nulovej hodnoty druhej derivácie obrazovej funkcie. Tieto miesta s nulovou hodnotou sú hľadané hrany. Na výpočet druhej derivácie sa používa tzv. Laplaceov gradientný operátor, ktorý sa označuje ∇^2. Ide o skalárnu veličinu, teda v porovnaní s gradientom sa prichádza o smer hrany, ale veľkosť je možné vypočítať podľa rovnice (4.13).

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{4.13}$$

Ako už bolo spomínané v predchádzajúcej metóde získaný obraz nie je spojitý, ale je tvorený jednotlivými pixlami. Parciálne derivácie, ktoré sú použité na výpočet veľkosti Laplaceovho operátoru je teda potrebné aproximovať. Znova sa na to využije diskrétna konvolúcia, ktorej konvolučné masky majú tvar (4.14).

$$k_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad k_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.14}$$

Ako je možné vidieť z ich tvaru, tak sú invariantné voči natočeniu. Nevýhodou použitia tohto operátoru je veľká citlivosť na šum, ale keďže vďaka strmosti prechodu je hľadanie priechodu hrany nulou oveľa spoľahlivejšie tak sa pristúpilo k riešeniu, v ktorom sa spraví konvolúcia obrazu s vyhladzovacím filtrom, aby sa zaistila robustnosť odhadu druhej derivácie. Filter podľa [4] musí spĺňať dve podmienky:

- Filter má byť hladký a vo frekvenčnom spektre má približne odpovedať pásmovej propusti, aby obmedzil možný počet frekvencií, pri ktorých môže dôjsť k priechodu nulou.
- Filter má reagovať iba na body z blízkeho okolia.

Popísaným podmienkam najlepšie odpovedá Gaussov vyhladzovací filter. LoG operátor využíva tento filter spolu s odhadom druhej derivácie pomocou Laplaceovho gradientného operátoru ∇^2. Pri prechode nulou pri LoG operátore potom platí (4.15).

$$\nabla^2 G(x,y,\sigma) * f(x,y) = (\nabla^2 G(x,y,\sigma)) * f(x,y) = 0 \tag{4.15}$$

Kde $G(x,y,\sigma)$ je použitý Gaussov filter a $f(x,y)$ je vstupná obrazová funkcia. Ako každý operátor aj LoG operátor má nevýhody. Medzi tie patrí vyhladenie ostrých tvarov a spájanie hrán do uzavretých kriviek.

4.4.4 Matematická morfológia

Ďalšou možnosťou ako získanú snímku pred použitím segmentačnej metódy upraviť, ale aj po nej, je pomocou matematickej morfológie. Do tejto časti patria dve základné metódy, a to erózia a dilatácia. Potom ich kombináciou vzniká tzv. otvorenie a uzavretie.

Erózia

Táto metóda matematickej morfológie funguje tak, že aktuálne spracovávanému pixlu priradí minimálnu hodnotu z jeho blízkeho okolia. Keďže sa pracuje v binárnom obraze, to znamená, že všetky pixly s hodnotou jedna (biela), ktoré susedia s pixlami s nulovou hodnotou (pozadie) sa zmenia na nulové pixly. Teda všetky objekty v obraze sa zmenšia z každej strany o jeden pixel.

Táto metóda sa najčastejšie používa na odstránenie šumu. Taktiež je možné pomocou tohto algoritmu oddeliť objekty, ktoré sa dotýkajú. Zaujímavým použitím je aj získanie hraníc tak, že sa odčíta erodovaný obraz od pôvodného.

Dilatácia

Ide o algoritmus, ktorý funguje obdobne ako erózia, len s jedným rozdielom. Aktuálne spracovanému pixlu nepriradí minimálnu hodnotu z jeho okolia ale maximálnu. To znamená, že všetky nulové pixly, ktoré susedia s pixlami s hodnotou jedna sa zmenia na pixly rovné jednej. V jednoduchosti to znamená, že hranice objektov sa zväčšia o jeden pixel.

Táto metóda sa používa na vyplnenie tenkých medzier v obraze, no nevýhodou je, že môže dôjsť aj k zvýrazneniu šumu v obraze.

Otvorenie

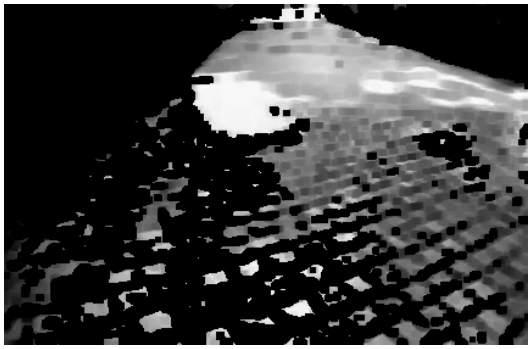
Táto metóda vznikla kombináciou dilatácie a erózie. Presnejšie erózia je nasledovaná dilatáciou. Použitím tohto algoritmu je možné z obrazu odstrániť malé oblasti alebo oddeliť objekty.

Uzavretie

Ide opäť o kombináciu dilatácie a erózie pričom v tomto prípade je dilatácia nasledovaná eróziou. Túto metódu je možné použiť pri spájaní objektov, ktoré sa nachádzajú blízko seba. Taktiež je možné docieliť vyhladenie obrysov v obraze.

Všeobecne metódy matematickej morfológie je vhodné použiť z dôvodu malej výpočetnej náročnosti, keďže pracujú s binárnym obrazom.

(a) Pôvodný obrázok



(b) Erózia

(c) Dilatácia

(d) Otvorenie

(e) Uzavretie

Obr. 4.6: Snímka upravená použitím rôznych metód matematickej morfológie, použité okolie 5×5

4.5 Segmentácia

Po aplikovaní metód z preprocesingu sa získal obraz, ktorý nie je skreslený, šum je potlačený a sú nájdené hrany v obraze. V ďalšom kroku je potrebné takto upravený obraz rozdeliť na niekoľko častí. Toto rozdelenie sa realizuje na základe spoločných vlastností pixlov, medzi ktoré patrí napríklad intenzita jasu, farba, textúra a pod.. Časť spracovania obrazu, ktorá sa práve týmto rozčlenením zaoberá sa nazýva segmentácia obrazu.

Podľa [9] je možné ju definovať nasledovne: Nech R označuje oblasť (celý obraz) a nech H je predikát pre dvojhodnotové ohodnotenie homogenity oblasti. Potom segmentácia je definovaná ako rozdelenie oblasti R do M podoblastí $R_1, R_2, , ..., R_M$, takých, že: Zjednotením všetkých disjuktných homogénnych podoblastí sa získa obraz R. Prienik

ľubovoľných dvoch podoblastí je prázdna množina. Každá z podoblastí musí spĺňať kritérium homogenity, ktoré by zjednotením dvoch alebo viacerých podoblastí bolo porušené. Pod pojmom kritérium homogenity sa rozumie úroveň jasu, farba a pod.

Segmentácia je proces, ktorý je ovplyvniteľný rôznymi faktormi ako napríklad kvalita vstupného obrazu, svetelné podmienky, pri ktorých bol obraz získaný, kvalita snímacieho zariadenia a v neposlednom rade aj vplyv okolitého prostredia.

Ako v prechádzajúcich častiach tak aj na segmentáciu obrazu bolo vytvorených mnoho metód. Medzi najznámejšie a aj najčastejšie používané patria:

- Prahovanie (Thresholding)
- Rozpoznanie hrán (Edge detection)
- Narastanie oblastí (Region growth)

V ďalších podkapitolách budú postupne jednotlivé metódy podrobnejšie popísané a vysvetlené.

Pri segmentácií tak isto ako pri predspracovaní je potrebné si pred začatím riešenia uvedomiť, že neexituje žiadny všeobecný postup, podľa ktorého sa získa požadované riešenie. Je potrebné metódy, ktoré sú v týchto kapitolách popísané, prípadne ďalšie, správne skombinovať a následne aplikovať tak aby, sa zo vstupného obrazu získali také informácie, ktoré odpovedajú čo najlepšiemu výsledku. Informácie o jednotlivých metódach boli opäť čerpané z [38][14][42][9].

4.5.1 Prahovanie

Najjednoduchšia metóda, ktorá sa používa na segmentovanie je prahovanie. Jedná sa o algoritmus, ktorý prostredníctvom prahu oddelí objekty od pozadia, pričom prah môže byť buď globálny pre celý obraz alebo lokálny. Základný princíp spočíva v tom, že na vstup sa privedie obraz, a následne sa prechádza každý pixel. Podľa hodnoty prahu sa rozhodne či daný pixel tento prah spĺňa alebo nie. Pokiaľ nie zaradí sa do skupiny, ktorá vyjadruje pozadie, inak do skupiny objektu. Ako prah sa najčastejšie volí úroveň jasu alebo farba.

Definuje sa teda ako priradenie (4.16).

$$f(i,j) = d \qquad\qquad t_{d-1} \leq f(i,j) < t_d;\ d = 1,2,...,D \tag{4.16}$$

Kde $f(i,j)$ je vstupný obraz a t_d je d-ty prah. Takto navrhnutá metóda sa vyznačuje svojou jednoduchosťou a výpočtovou nenáročnosťou. Prahovanie je možné rozdeliť na niekoľko typov algoritmov podľa prahu, aký daná metóda využíva, pričom sa môže určovať manuálne alebo automaticky napríklad použitím metódy p-podielu , metódy analýzy tvaru histogramu a optimálneho prahovania, ktoré sú podrobnejšie vysvetlené v [9].

Globálne prahovanie

Ide o metódu, ktorá je použiteľná na obrazy, v ktorých sa objekty od pozadia výrazne odlišujú. Keď sa vezme v úvahu vzťah (4.16) a $d = 1$, tak globálne prahovanie je možné definovať nasledovne:

$$g(i,j) = \begin{cases} 1 & f(i,j) \geq T \\ 0 & f(i,j) < 0 \end{cases} \tag{4.17}$$

Kde $f(i, j)$ je vstupný obraz, $g(i, j)$ výstupný binárny obraz a T je v tomto prípade prah. Globálne prahovanie je použiteľné len v malom množstve aplikácií práve z dôvodu použitia jedného prahu na celý obraz. Pokiaľ by sa v obraze nachádzali napríklad dva objekty, ktoré by mali inú jasovú úroveň, tak použitím iba jedného prahu by mohlo dôjsť k zániku celého objektu. Taktiež táto metóda nie je použiteľná pri obrazoch, ktoré sú nerovnomerne osvetlené alebo výrazne ovplyvnené šumom.

Prahovanie s hysteréziou

Tento typ prahovania pracuje s dvoma prahmi, tzv. horným (high) a dolným (low) prahom. Princíp použitia je nasledovný. Na vstup sa privedie obraz, postupne sa prechádza každý pixel. Hodnota aktuálne zvoleného pixlu sa porovná s oboma prahmi. Pokiaľ je hodnota pixlu vyššia ako horná prahová hodnota pixel sa prehlási za patriaci do hľadaného objektu a priradí sa mu hodnota najčastejšie sa používa 255, čo je hodnota maximálneho jasu. V prípade, že aktuálne spracovávaný pixel neprekročí spodný prah, tak sa prehlási za okolie a priradí sa mu nulová hodnota. Pokiaľ hodnota pixlu spadá do intervalu dvoch prahov, tak je potrebné vziať do úvahy okolie pixlu. Pokiaľ sa v jeho okolí nachádza aspoň jedna hodnota, ktorá presiahla hornú prahovú hodnotu, tak aktuálne spracovávaný pixel sa prehlási, taktiež za patriaci do hľadaného objektu a priradí sa mu znova hodnota maximálneho jasu, inak sa prehlási za nulový, teda za okolie. Hysterézne prahovanie sa používa pri Cannyho hranovom detektore, ktorý slúži na hľadanie hraníc v obraze.

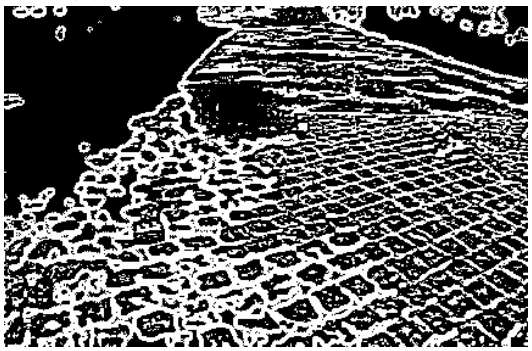
Adaptívne prahovanie

Pokiaľ sa v obraze nachádza viac oblastí s odlišnou úrovňou jasu, tak globálne prahovanie ako už bolo napísané v predchádzajúcej časti by bolo neefektívne. Je preto potrebné zvoliť metódu, ktorá získaný obraz rozdelí na niekoľko podoblastí pričom pre každú z nich je prah určený samostatne. Takéto riešenie poskytuje práve algoritmus adaptívneho prahovania. Prah sa v tomto prípade volí automaticky na základe vlastnosti obrazu v danom mieste. Využíva na to buď priemer pixlov, alebo Gaussovské rozdelenie hodnôt jasu na danom lokálnom mieste.

(a) Pôvodný obrázok

(b) Globálne prahovanie, $t = 100$

(c) Prahovanie s hysteréziou $t_1 = 100, t_2 = 150$



(d) Adaptívne prahovanie s použitím Gaussovho rozdelenia na blízkom okolí veľkosti 9×9

Obr. 4.7: Príklady vyššie popísaných typov prahovania

4.5.2 Rozpoznanie hrán

V obraze sa zvyčajne nachádzajú rôzne objekty a pozadie. Prechody medzi nimi sú pokladané za dôležité miesta, pretože obsahujú viacej informácií ako iné miesta v snímke. Je preto dôležité aby boli tieto miesta nájdené.

V 4.4.3 bolo popísané hľadanie hrán v obraze prostredníctvom rôznych operátorov. Okrem hrán sa v spracovaní obrazu zavádza aj pojem hranica. Hranica je výstup segmentácie a vyjadruje obrys segmentovaného regiónu. Teda rozdiel medzi hranou a hranicou je, že zatiaľ čo hrana musí spĺňať len podmienku vysokého gradientu, hranica musí spĺňať aj ďalšie podmienky ako napríklad stanovenú hodnotu jasu a iné.

Sledovanie vnútornej hranice objektov

Aby bolo možné použiť túto segmentačnú metódu, je potrebné v prvom kroku vytvoriť binárny obraz. Následne sa aplikuje postup, ktorého výsledkom sú hranice objektov.

Postup nájdenia hranice je nasledovný. Najskôr sa začne prehľadávať obraz. Tento proces trvá pokiaľ sa nenájde bod, ktorý patrí do inej oblasti. V binárnom obraze sa inou oblasťou rozumie hrana. Následne sa prehľadá okolie nájdeného bodu, ktorého rozmer je 3×3. Prvý nájdený bod sa stáva novým bodom hranice. Treba poznamenať, že v každom kroku sa uchováva poloha predchádzajúceho bodu. Algoritmus sa zastaví vo chvíli, keď sa vráti do počiatočného bodu.

Prahovanie gradientného obrazu

Ide o jednu z najjednoduchších segmentačných metód, ktorá sa radí to detekcie hrán. Algoritmy, ktoré spadajú do tejto časti rozpoznania hrán v obraze sa delia na dve kategórie a to na:

1. Detektory založené na aproximovaní maxím prvých derivácií
2. Detektory založené na hľadaní priechodov nulou pri druhej derivácií

Použitím operátorov vysvetlených v časti 4.4.3 a prahovaním je možné získať hľadaný obraz so zvýraznenými hranicami.

Cannyho hranový detektor

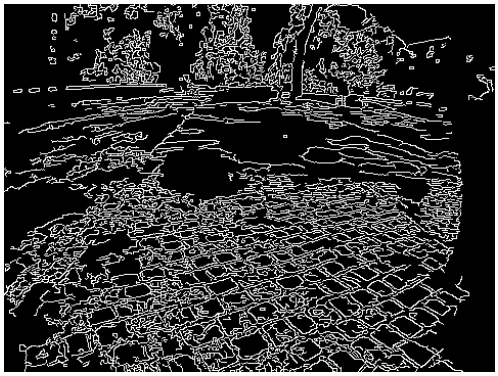
Posledným detektorom, ktorý sa využíva na nájdenie hrán v obraze je tzv. Cannyho hranový detektor. Ide o metódu, ktorej cieľom je nájsť skokové hrany prostredníctvom filtru. Jeho myšlienka je založená na troch kritériách:

- Detekčné kritérium: Musia byť detegované všetky hrany a zároveň miesta, kde sa hrany nenachádzajú, sa nesmú prehlásiť za miesta s hranami.
- Lokolizačné kritérium: Rozdiel medzi reálnou a nájdenou hranou musí byť minimálny.
- Požiadavka jednej odozvy: Každá detegovaná hrana musí byť nájdená len raz, teda na jednu hranu musí byť maximálne jedna odozva.

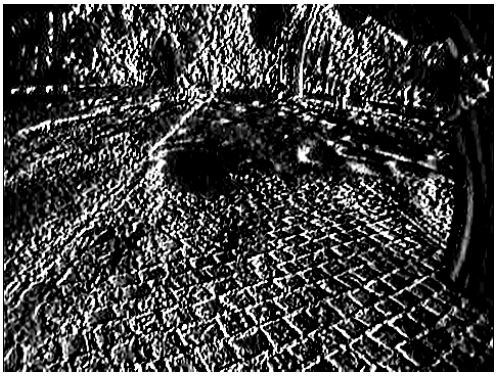
Na realizáciu týchto kritérií sa vytvoril postup, ktorý je možné popísať v niekoľkých krokoch.

1. Filtrácia obrazu prostredníctvom Gaussovho vyhladzovacieho filtra. V tomto kroku sa eliminuje šum, získaný obraz sa vyhladí.
2. Určenie gradientu. Keďže obraz je nespojitý, tak na aproximáciu prvej derivácie sa použije konvolúcia so Sobelovým operátorom.
3. Nájdenie lokálnych maxím derivácií.
4. Potlačenie odoziev mimo maxima (non-maximal suppression). Potlačia sa hodnoty, ktoré nie sú maximálne a tým dôjde k stenčeniu hrán.
5. Prahovanie s hysteréziou. Postup tejto metódy bol popísaný v 4.5.1.

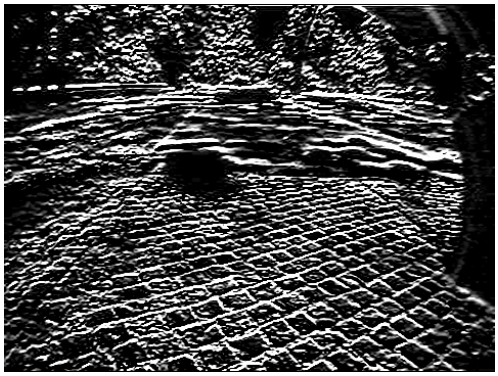
(a) Pôvodný obrázok



(b) Cannyho hranový detektor



(c) S použitím Sobelovho operátoru v smere x



(d) S použitím Sobelovho operátoru v smere y

(e) S použitím Laplaceovho operátora

Obr. 4.8: Príklady vyššie popísaných typov detektorov

4.5.3 Narastanie oblastí

V prechádzajúcej časti boli popísané metódy, ktorých podstatou bolo nájdenie hrán v obraze. Táto metóda je navrhnutá úplne inak. Ide o algoritmus, ktorý sa v obraze snaží nájsť oblasti s homogénnymi vlastnosťami, teda podstata metódy vychádza z klasickej definície segmentácie, ktorá bola popísaná v úvode tejto kapitoly. Za homogénne vlastnosti sa môžu považovať intenzita jasu, farba, textúra a iné. Pri hľadaní oblastí sa používa práve homogenita ako kritérium. Príkladom kritéria homogenity je stredná hodnota intenzity jasu.

Princíp algoritmu je nasledovný. Ako prvé je potrebné v snímke vytvoriť počiatočné regióny. Následne sa budú testovať body, ktoré susedia s týmto regiónom. Keďže ide o iteratívny proces, tak v každom kroku sa otestuje susedný bod narastajúcej oblasti. Pokiaľ vybraný bod bude spĺňať zvolené kritérium homogenity, tak sa priradí do oblasti. Pokiaľ bod patrí už do inej oblasti, tak sa tieto oblasti zlúčia. Následne po priradení je potrebné upraviť kritérium a určiť nové body, ktoré susedia s narastajúcou oblasťou. Pokiaľ, ale testovaný bod kritérium nesplní tak môžu nastať dve situácie. Prvou je, že bod sa prehlási za nový región. Druhou je, že tento bod sa nezaradí nikam a bude sa pokračovať v pôvodnom procese ďalej. Teda obraz bude mať len toľko regiónov, koľko ich bolo vytvorených na začiatku.

Výhodou tejto metódy je, že je odolná voči šumu, no jej veľkou nevýhodou je práve výpočtová náročnosť.

4.6 Python

Za programovací jazyk, v ktorom bude celé riešenie realizované, sa zvolil Python. Je to vysoko úrovňový, dynamický, interpretovaný programovací jazyk. Medzi jeho výhody jednoznačne patrí, že sa jedná o open-source softvér, teda voľne dostupný pre každého. Taktiež pri písaní kódu umožňuje používať okrem objektovo orientovanej paradigmy aj procedurálnu a funkcionálnu. Medzi veľké výhody patrí aj jeho čitateľnosť a krátkosť v porovnaní s inými programovacími jazykmi.

Celkovo boli vyvinuté tri verzie. Aktuálne je aktívna a bežne používaná práve verzia 3.x. Má v sebe zabudované veľké množstvo knižníc a pokiaľ je potrebná iná tak je možné ju kedykoľvek dodatočne nainštalovať. [29] Na realizáciu riešenia bude potrebné niektoré doinštalovať. Použité knižnice:

- Numpy
- OpenCV

Numpy ja základný balík pre vedecké výpočty v jayku Python. Podporuje mnoho matematických funkcí, generátory náhodných čísel, Fourierove transformácie a mnoho iných. Rovnako podporuje prácu s rozsiahlymi n-dimenzionálnymi poľami. Jeho jadro tvorí dobre optimalizovaný kód v jazyku C.[26]

OpenCV (Open Source Computer Vision Library) je knižnica pre počítačové videnie a strojové učenie s podporou v jazykoch C++, Python a Java. Obsahuje viac než 2500 optimalizovaných algoritmov medzi, ktoré paria filtre a segemtačné metódy používané v tejto páci.[27]

5 Postup riešenia

V predchádzajúcej kapitole boli priblížené metódy spracovania obrazu, pomocou ktorých by bolo možné nájsť cestu v obraze. V nasledujúcich častiach sa popíše riešenie, ktoré bude tvorené kombináciou práve vyššie popísaných algoritmov.

Podstatnou vecou pri vytváraní riešenia je to, aby navrhnutý algoritmus spĺňal podmienky, ktoré boli vytýčené v časti 3. Od metódy sa požaduje aby bola odolná voči náhlym zmenám intenzity jasu. Taktiež je potrebné aby ju bolo možné použiť na rôznych typoch cesty a aby bola odolná voči znečisteniu, ktoré sa môže na ceste nachádzať.

Riešenie je rozdelené na niekoľko častí, ktoré budú podrobnejšie popísané v nasledujúcich podkapitolách. V prvej časti sa popíše a zdôvodní odstránenie nežiaduceho okolia. V druhej časti sa analyzuje farebnosť cesty a následne sa pristúpi k rozloženiu celej snímky na jednotlivé RGB zložky. V treťom kroku sa zrealizuje potrebná korekcia obrazu. Následne sa pristúpi ku samotnej segmentácií. Chybám v obraze a ich odstráneniu je venovaná piata časť, v ktorej je popísané využitie, aj prípadné vytvorenie uzavretých kriviek a ich následné odstránenie na základe veľkosti plôch, ktoré obopínajú. V poslednej časti sa vyznačia získané hranice cesty.

Vstupnými dátami, ktoré sú potrebné pre riešenie sú fotografie získané z mobilného robota B2, ktoré boli vytvorené v parku Lužánky - Brno. Na každej snímke je zaznamenaná cesta s okolitým prostredím. Rozmer každej získanej fotografie je 480×360 pixlov.

5.1 Zjednodušenie obrazu

Získanú snímku je vhodné upraviť skôr ako sa pristúpi k samotnej detekcií. Je veľké množstvo faktorov, ktoré môžu ovplyvniť správne nájdenie cesty. Niektoré z nich je možné odstrániť ešte pred aplikovaním algoritmov na rozpoznanie cesty. Na obr. č. 5.1 sú uvedené pôvodné fotografie.

Obr. 5.1: Pôvodné fotografie

Ako je možné vidieť na všetkých snímkach sa nachádzajú kryty z robota - pravá časť obrazu. Tieto časti z robota, ktoré boli zachytené pri snímaní cesty, nenesú žiadnu informáciu o tom, kde sa cesta nachádza a pre ďalšie spracovanie sú nepotrebné, preto sa pristúpilo k ich odstráneniu. Odrezalo sa 50 pixlov z pravej strany každej snímky.

Ďalšou časťou obrazu, ktorá môže komplikovať detekciu cesty sú oblasti nad tzv. horizontom cesty (miesto, kde cesta zaniká). V tejto časti obrazu sa väčšinou nachádza obloha, stromy, budovy a pod., ktoré buď svojou farbou alebo tvarom môžu výrazne

ovplyvniť výsledky. Jednou z možností bola metóda detekcie horizontu, ktorá bola použitá v prácach spomínaných v rešeršnej časti. Keďže sa ale predpokladá, že cesta je bez výrazného stúpania pristúpilo sa k radikálnejšiemu riešeniu, z každej snímky sa odrezalo 20 %. Teda zo snímky s rozmerom 480 × 360 pixlov sa vytvorila snímka s rozmerom 480 × 280 pixlov. Pokaľ by sa cesta už nachádzala aj v odrezanej časti, tak informácie o jej tvare a polohe sa nestratia. Je to z toho dôvodu, že robot zaznamenával cestu podstatne rýchlejšie v porovnaní s jeho pohybom. Teda informácie budú obsiahnuté v nasledujúcej snímke. Výsledné, orezané fotografie, ktoré sa použili na rozpoznanie cesty vyjadruje obr. č. 5.2.

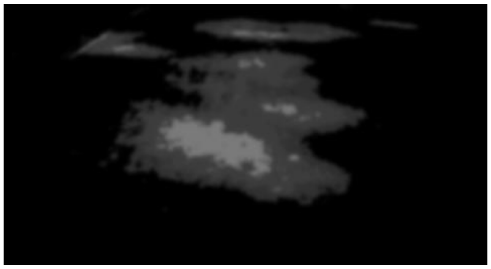
Obr. 5.2: Orezané fotografie

5.2 Rozdelenie na zložky RGB

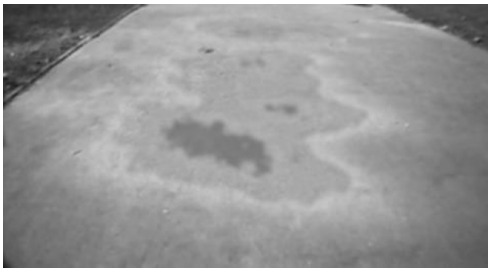
V tejto časti je potrebné získanú snímku upraviť tak, aby po následnej filtrácií a segmentácií bolo možné cestu v obraze vyznačiť. Preto je potrebné navrhnúť také riešenie, ktoré bude zohľadňovať všetky podmienky popísané v úvode tejto kapitoly. Pracovalo sa s niekoľkými myšlienkami, pomocou ktorých by bolo možné riešenie zrealizovať.

Prvou bolo, že sa na miesta s vysokou hodnotu jasu spraví „záplata“. Postup by bol nasledovný. Najskôr by sa snímka previedla na šedotónnu. Následne by sa z obrázku spravil histogram, z ktorého by sa vybralo 5% najjasnejších pixlov. Potom by sa učila minimálna hodnota z týchto 5% a tá by sa odčítala od pôvodného histogramu. Výsledkom by bola maska na čiernom pozadí. Následne by sa obrázok upravil použitím masky. Navrhnutý postup sa aj zrealizoval viď. obr. č. 5.3, no pri ostrých prechodoch svetlo-tieň je daná metóda nepoužiteľná.

(a) Pôvodný obrázok



(b) Maska

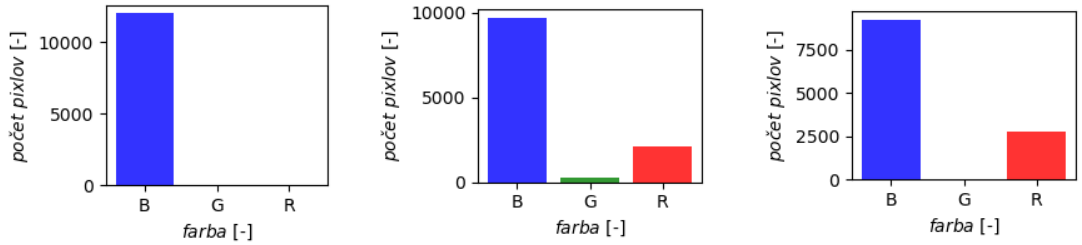


(c) Po pridaní masky

Obr. 5.3: „Záplata“

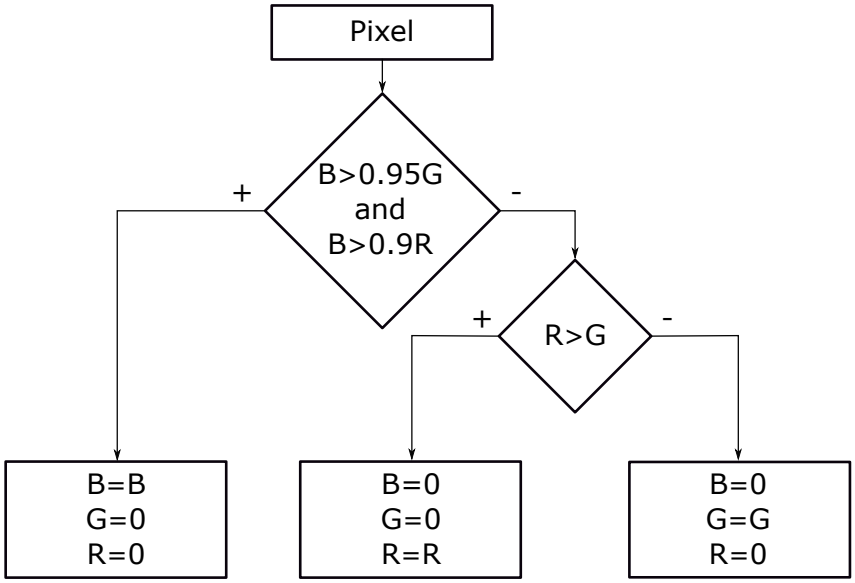
Ďalšou možnosťou bola zmena farebného priestoru. Na taký, ktorý je voči zmene intenzite svetla invariantný. Zvažovalo sa použitie modelu HSV. Pri odstraňovaní miest s vysokou hodnotou jasu sa vyskytol problém. Tieto miesta, ktoré sú veľmi blízko bielej môžu mať v tomto priestore praktický ľubovoľnú farbu a pri znížení saturácie na týchto miestach vznikli oblasti s náhodnými farbami (najčastejšie ružová, purpurová, tyrkysová).

Poslednou zvažovanou možnosťou, bolo zistiť, ktorej základnej farbe najviac odpovedá cesta v RGB modeli. Postup bol nasledovný. Z viacerých snímok sa vyrezalo miesto, na ktorom sa nachádzala cesta. Výrez bol veľkosti 200×80 pixlov. V každom pixle sa zistila, ktorá zložka nadobúda maximálnu hodnotu. Získané výsledky sú vyjadrené grafmi na obr. č.5.4.



Obr. 5.4: Grafy vyjadrujúce počet pixlov v závislosti na farbe

Keďže sa v snímkach nachádzajú miesta s vysokou intenzitou jasu tak namiesto štandardného rozkladu na farebné zložky (modrú, zelenú, červenú) sa aplikuje postup, ktorý popisuje obr. č. 5.5.

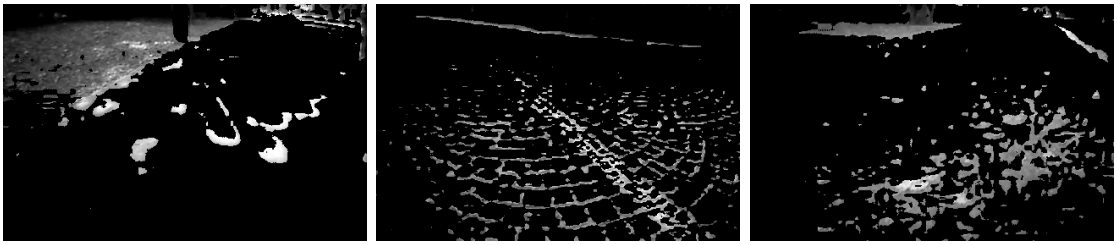


Obr. 5.5: Postup určenia farby pixlu

Na vstupe sa nachádza aktuálne zvolený pixel, ktorý sa skladá zo spomínaných troch zložiek - modrej(B), zelenej(G) a červenej(R). Následne sa vyhodnotí, ktorá zložka má najvyššiu hodnotu a zvyšné dve sa prehlásia za nulové. Výnimku tvoria pixly, ktoré majú všetky farebné zložky takmer rovné. Keďže modrá zložka je dominantná na ceste uplatnila sa nasledujúca podmienka. Pokiaľ je modrá zložka väčšia ako 95% zelenej alebo 90% červenej zložky tak za najvyššiu hodnotu sa považuje modrá časť a zvyšné dve sa prehlásia za nulové. Tento postup sa aplikuje na všetky pixly v snímke a výsledkom sú tri obrázky, ktoré popisujú obr. č. 5.6 - 5.8.

Obr. 5.6: Modrá zložka

Obr. 5.7: Zelená zložka



Obr. 5.8: Červená zložka

V získaných výsledkoch je možné si všimnúť, že cestu je možné vyjadriť aj opačne, práve prostredníctvom, zelenej, ktorá sa na ceste nenachádza. Tento postup by sa mohol použiť, ale celkové riešenie by sa zúžilo len, na fotografie, v ktorých cestu obopína zelené okolie. V opačnom prípade, ako je možné vidieť aj na obr. č. 5.7 v ľavo cesta zanikne s okolím a jej identifikácia by nebola možná. Aj z tohto dôvodu sa pre ďalšie spracovanie použili výsledky prezentované na obr. č. 5.6, teda modrej časti.

5.3 Korekcia obrazu

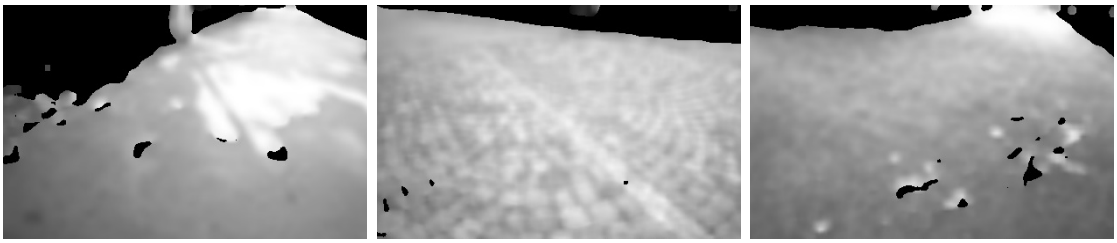
V obraze sa nachádza veľa nedokonalostí, ktoré je potrebné odstraňovať rôznymi metódami, ktoré boli popísané v časti 4.4. Po použití rozkladu na farebné zložky ani tieto obrázky nie sú výnimkou. Vznikli v nich rôzne nedokonalosti, či už vplyvom veľkej intenzity svetla, rôznych objektov na ceste alebo typom povrchu cesty. Tieto nedokonalosti sa prejavujú formou čiernych pixlov, ktoré nezapadajú medzi okolité, tak ako je to možné vidieť na obr. č. 5.6. Na prvý pohľad sa môže zdať, že tieto snímky, už nie je možné upraviť do takého stavu, aby cesta bola jednoliata a bola možná jej identifikácia, no opak je pravdou. Na korekciu celého obrazu sa vytvoril algoritmus, ktorý je podobný priemerovaniu popísaného v časti 4.4.3.

Vytvorila sa maska rozmeru 3×3, ktorá postupne prechádzala celú snímku. Pri každom posune sa overovalo, koľko čiernych pixlov sa aktuálne nachádza v maske. Pokiaľ ich hodnota presiahla 75% všetkých pixlov v rámci masky, tak sa aktuálne vyhodnocovaný pixel - pixel v strede masky prehlásil za čierny. Pokiaľ hodnota všetkých pixlov nepresiahla spomínanú percentuálnu hodnotu, tak sa spravila priemerná hodnota nenulových pixlov. Výsledok sa znova zapísal do aktuálne vyhodnocovaného pixlu. Tento postup sa zopakoval na každej snímke dva- krát. Výsledok tohto procesu je možné vidieť na obr. č. 5.9.

Obr. 5.9: Snímky po korekcií s maskou 3×3

Vyššie popísaný postup nedokonalosti v obraze zmenšil, ale nie natoľko, aby bolo možné aplikovať hľadanie uzavretých kriviek v obraze, preto sa na finálnu úpravu použila

maska s väčším rozmerom 5×5 a znova sa celý postup aplikoval na každú snímku. Na obr. č. 5.10 je zobrazený výsledok po tomto procese.

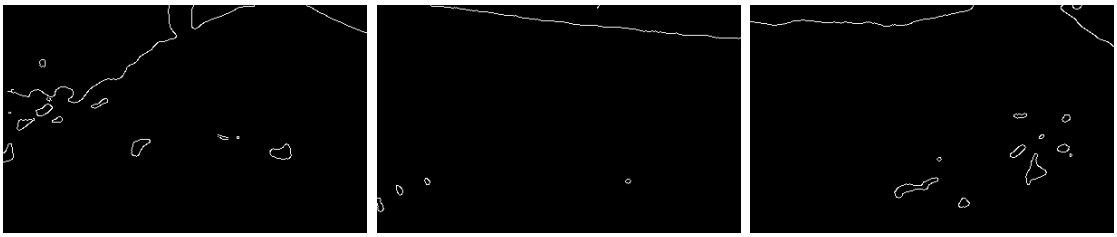


Obr. 5.10: Snímky po korekcií s maskou 5×5

5.4 Nájdenie hrán v obraze

V rešeršnej časti bolo popísaných viacero segmentačných metód. Prvou z nich bolo prahovanie, viď. 4.8. Tento algoritmus sa nepoužil z jednoduchého dôvodu. Touto metódou by sa nič nezískalo, keďže pixly okolia majú nulovú hodnotu. Ďalšou metódou, ktorá bola vysvetlená v časti 4.5.3 je metóda narastania oblastí. Keďže sa nakoniec zvolil postup rozdelenia obrazu na jednotlivé zložky aj táto metóda je nepoužiteľná z rovnakého dôvodu ako prahovanie. Preto sa pristúpilo k poslednej vysvetlenej metóde, a to detekcií hrán. Na nájdenie hrán v obraze je možné použiť viacero filtrov, ako napríklad Sobelov a pod. Pretože za optimálnu metódu sa považuje Cannyho hranový detektor použila sa práve táto metóda. Jej teória bola popísaná v časti 4.5.2. V skratke ide o metódu, ktorá v sebe zahrňuje viacero krokov, a to filtráciu Gaussovým filtrom, následné použitie Sobelovho filtra a na rozhodnutie či sa jedná o hranu alebo nie sa použije prahovanie s hysteréziou.

V práci sa použila funkcia z knižnice OpenCV prostredníctvom, ktorej sa metóda realizovala. Nazýva sa `Canny` a je do nej potrebné nastaviť niekoľko argumentov. Ako prvý sa požaduje vstupný obraz, druhý a tretí argument vyjadrujú minimálnu a maximálnu hodnotu prahovania s hysteréziou, teda ktoré hrany sa naozaj pokladajú za hrany a ktoré nie. Štvrtý argument vyjadruje veľkosť Sobelovho kernelu. Pokiaľ sa hodnota nenastaví, východzia hodnota je nastavená na veľkosť tri. Po nastavení všetkých parametrov `cv.Canny(img, 220, 250)` je získaný výsledok zobrazený na obr. č. 5.11.

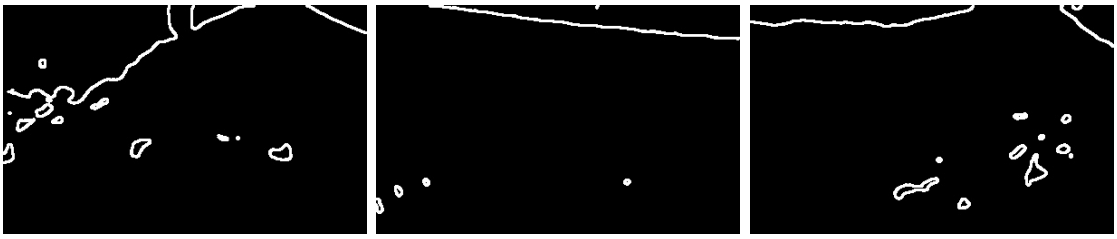


Obr. 5.11: Nájdené hrany v obraze

5.5 Uzavreté krivky

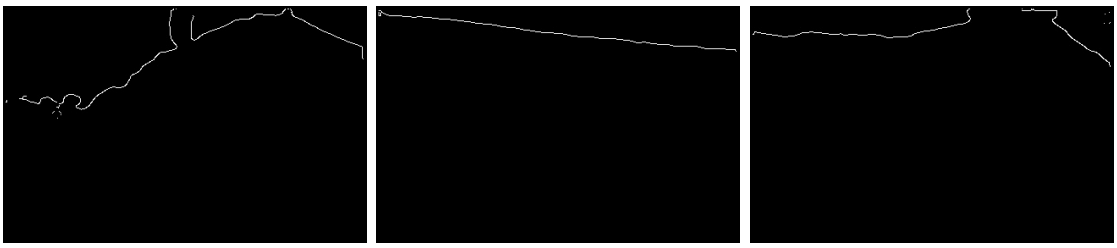
Na obr. č. 5.11 sú vyznačené hranice cesty, ale aj všetky nedokonalosti, ktoré sa v procese predspracovania nepodarilo odstrániť. Táto časť sa bude zaoberať práve ich nájdením a následnou elimináciou.

V prvom kroku sa budú hľadať v obraze uzavreté krivky. Keďže pri použití Cannyho hranového detektora v krivkách vznikli nespojitosti, bolo potrebné ich hrúbku zväčšiť, čím sa tento problém vyriešil. Použila sa na to metóda z matematickej morfológie - dilatácia. Výsledok je zobrazený na obr. č. 5.12.



Obr. 5.12: Zvýraznené hrany v obraze

Následne sa na detekciu týchto uzavretých kriviek použila funkcia `findContours`. Problém pri jej použití bol, že všetky krivky pokladala za uzavreté. Bolo potrebné zaviesť minimálnu veľkosť oblasti závislú na dĺžke krivky, ktorú majú obopínať. Následne po ich nájdení boli odstránené z pôvodného obr. č. 5.11.



Obr. 5.13: Obrázky po odstránení hrán pomocou uzavretých kriviek

Týmto postupom sa neodstránili krivky na okrajoch obrazu. Bolo ich potrebné uzavrieť, čo sa docielilo pridaním bieleho okraja. Všetky krivky, ktoré v obraze zostali sa považujú za hranice cesty. Získaný výsledok je možné vidieť na obr. č. 5.13.

6 Zhodnotenie dosiahnutých výsledkov

V prechádzajúcej kapitole bol popísaný postup riešenia, v ktorom sa vysvetlili dôvody použitia jednotlivých metód. Cieľom tejto kapitoly bude vyhodnotiť úspešnosť daného algoritmu. Taktiež budú popísané výhody a nevýhody navrhnutej metódy a jej časová náročnosť.

Ako prvé je potrebné definovať, ktoré výsledky sa považujú za správne a ktoré za nesprávne. Následne je možné vyhodnotiť úspešnosť zvolenej metódy. Za úspešne nájdenú cestu sa pokladajú nasledujúce prípady:

- Hranice cesty nájdené po celej dĺžke na oboch stranách.
- Hranica cesty nájdená po celej dĺžke na jednej strane, na druhej aspoň čiastočne.
- Hranice cesty nájdené zhruba do polovice dĺžky celej cesty na oboch stranách.
- Hranica cesty sa pokladá za nájdenú aj v prípade, že sú v obraze ďalšie čiary, ktoré sa nachádzajú mimo nej.

Vo všetkých ostatných prípadoch sa cesta považuje za nenájdenú. Sú to prípady ako napríklad, nenájdené žiadne hranice cesty alebo nájdenie hraníc v strede cesty a pod. Takýto prípad je zobrazený na obr. č. 6.2 v pravo dole. Úspešnosť metódy podľa predchádzajúcich kritérií je zaznamenaná v tabuľke č. 6.1.

Celkový počet snímok	Nájdené hranice cesty	Nenájdené hranice cesty	Percentuálna úspešnosť
260	235	25	90,39

Tabuľka 6.1: Tabuľka vyjadrujúca úspešnosť zvolenej metódy

Dosiahnutá percentuálna úspešnosť bola neočakávaná, keďže sa predpokladala malá robustnosť tejto metódy. Za veľké pozitívum sa pokladá okrem úspešnej nezávislosti na intenzite svetla aj možnosť použitia algoritmu na všetkých typoch povrchu vrátane prechodov medzi nimi.

Je potrebné spomenúť aj nevýhody tejto metódy. Prvou z nich je časová náročnosť. Pri pôvodnej veľkosti obrázka, 430×360 pixlov, bol výpočetný čas potrebný na nájdenie hraníc cesty v jednej snímke 180 sekúnd. Čo je pre prípad aplikácie v reálnom čase nepoužiteľné. Preto sa pristúpilo k nasledovnému riešeniu. Vstupný obrázok sa zmenšil o polovicu, teda nový rozmer bol 215 × 140 pixlov. V zmenšenej snímke nebolo potrebné vykonať jej korekciu dva-krát, ale len raz. Po týchto úpravách sa podarilo výpočetný čas znížiť na 6 sekúnd. Pre real-time aplikácie je to stále nepoužiteľné, ale pre ostatné aplikácie ako napríklad príprava do neurónovej siete je to vhodné a aj dostatočne rýchle riešenie.

Obr. 6.2: Vyznačené hranice cesty v pôvodných obrázkoch

Ďalšie výsledky možno nájsť v prílohe A.

6.1 Možné modifikácie a ďalšia práca

V prípade ďalšej práce by bolo potrebné vylepšiť vypočetný čas navrhnutej metódy. Riešenie je v optimalizácií tej časti algoritmu, ktorá sa venuje korekcií obrazu. Jej vylepšenie by mohlo spočívať v použití rôzneho kroku pri prechádzaní snímky alebo návrhu iného a lepšieho algoritmu.

Taktiež by bolo možné algoritmus rozšíriť na typy cesty, ktoré sú reprezentované inou základnou farbou. Riešenie by mohlo spočívať v určovaní farby v oblasti záujmu pri každej snímke.

Ďalšou prácou by mohlo byť rozšírenie navrhnutej metódy, tak aby bola použiteľná aj v mestskom prostredí.

7 Rozpoznanie cesty prostredníctvom neurónových sietí

Rozpoznanie objektov na fotografiách, zlepšenie vyhľadávania obrázkov, vytváranie umeleckých diel, hranie hier. Toto všetko bolo donedávna doménou človeka a záhadou pre svet programovania. Ľudský mozog bol v tomto smere neprekonateľný, no v súčasnosti sa začali používať umelé neurónové siete, ktoré vznikli na základe poznatkov získaných práve z pozorovania neurónovej siete v ľudskom mozgu. Tieto umelé neurónové siete patria do oboru umelej inteligencie a existuje pre ne pomenovanie deep learning (hlboké učenie). Cieľom sietí je v získaných dátach nájsť hlbší význam a následne porozumieť abstraktným pojmom a reagovať na rôzne úlohy a situácie ako človek.

Existuje veľa druhov neurónových sietí a každá z nich je vytvorená pre iný účel. Napríklad na rozpoznávanie reči sa používa rekurentná neurónová sieť [10]. Pomocou nich je taktiež možné hranie hier ako napríklad šach [44] alebo gomoku [37].

Výnimkou nie sú ani autonómne vozidlá. V predchádzajúcich častiach 2 bolo priblížených viacero prístupov, ktoré sa na rozpoznávanie cesty používajú. Dokonca sa vytvoril aj vlastný algoritmus na detekciu cesty. Síce výsledky, ktoré všetky články aj vlastná práca poskytujú sú dostatočné, dalo by sa povedať že nie je dostatočne robustné. Stačí pri každej metóde zaviesť do okolia šum, na ktorý daná metóda nie je navrhnutá a výsledkom určite nebude správne nájdená cesta. Preto sa v posledných rokoch aj na rozpoznávanie cesty začali používať neurónové siete.

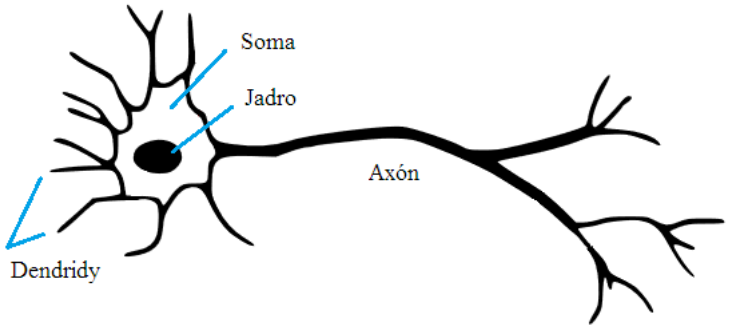
Malik, Ahmed a Kunwar vo svojej práci [22] uviedli metódu, ktorá je schopná rozpoznať cestu pri rozličných vonkajších podmienkach. Jej podstatou je klasifikácia podľa charakteristických čŕt a použitie Kohenovej samo organizačnej mapy (SOM) na nájdenie cesty. Algoritmus celkovo pozostáva z piatich krokov. Ako prvé bolo potrebné vstupný obraz upraviť a extrahovať vlastnosti. Následne sa použije samo organizačná mapa, ktorej výstupom je odlíšenie miest, kde sa nachádza cesta od ostatných. Teda výstup je buď nulový alebo rovný jednej pre každý pixel. Získa sa binárny obraz. Na záver sa aplikuje algoritmus na detekciu hrán a Houghova transformácia. Autori v práci prezentujú výsledky, pri ktorých zvolená metóda dosahuje až 99 percentnú úspešnosť. Je potrebné ale podotknúť, že navrhnutá sieť bola trénovaná a testovaná len na veľmi malom množstve obrázkov.

S inou, ale podobnou metódou prišli Wuerch a Wallance [48]. V ich riešení je na vstup siete privedený celý obraz a výstupom sú opäť binárne hodnoty, ktoré vyjadrujú či sa v danom mieste v obraze nachádza cesta alebo nie. Potom sa tieto nájdené miesta vyznačia v pôvodnom obraze. Výsledky, ktoré autori prezentujú dosahujú presnosť takmer 95%. Trénovacia množina, ktorú použili obsahovala okolo desať tisíc obrázkov. Nevýhodou tejto metódy je jednoznačne veľkosť výstupu, z dôvodu modifikácie veľkého množstva parametrov v sieti.

Predtým ako sa pristúpi k navrhnutiu a aplikovaniu vlastnej konvolučnej neurónovej siete je potrebné vysvetliť základné pojmy, ktorých porozumenie je neoddeliteľnou súčasťou celého procesu.

7.1 Neurón

Ako prvé je potrebné popísať základnú stavebnú jednotku neurónovej siete – neurón. Z biologického hľadiska sa jedná o bunku, ktorá je určená k prenosu, spracovaniu a uchovávaniu informácií, ktoré sú potrebné pre funkčnosť organizmu. Neurón pozostáva zo somatu, dendridov a axiónu (obr. č. 7.1).



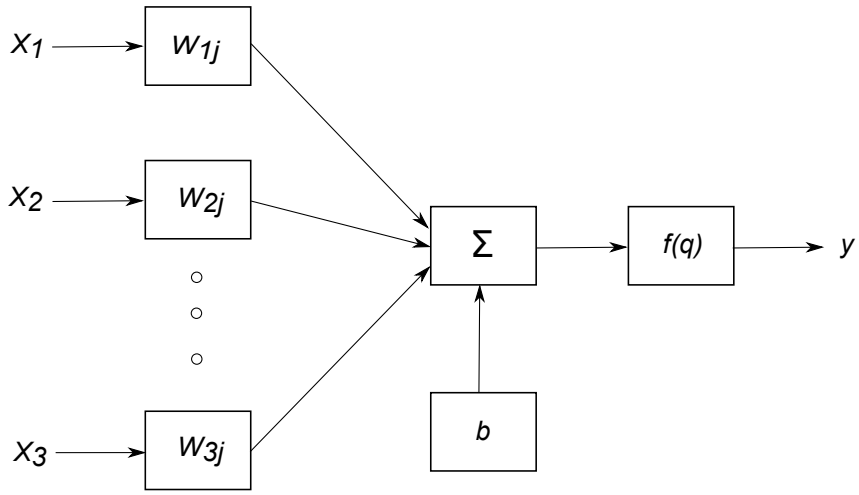
Obr. 7.1: Biologický neurón

V umelých neurónových sieťach bol na základe biologického neurónu vytvorený tzv. formálny neurón. Matematicky sa môže vyjadriť nasledovným vzťahom:

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{7.1}$$

Tento formálny neurón, má n reálnych vstupov $x_1, ..., x_n$. Tieto vstupy sú ohodnotené synaptickými váhami $w_1, .., w_n$, ktoré vyjadrujú priepustnosť, ktorá sa pri prechode signálu mení a tým sa neurónová sieť učí. Suma súčinu reálnych vstupov a synaptických váh vyjadruje vnútorný potenciál neurónu ξ, ku ktorému sa dodatočne pripočíta premenná b, ktorá označuje špeciálny neurón tzv. prah (bias). Bias vyjadruje hodnotu potrebnú pre ovplyvnenie aktivačnej funkcie neurónu. Výstup y z neurónu sa vypočíta na základe vnútorného potenciálu ξ a aktivačnej funkcie f, ktorá slúži pre prahovanie vnútorného potenciálu.[4]

V jednoduchosti je možné vyššie popísaný postup vysvetliť nasledovne. Neurón je možné chápať ako aproximátor, ktorý funguje ako vypínač. Na vstup neurónu sú privedené výstupy z iných neurónov. Zo vstupov je potrebné vytvoriť jeden signál. Slúži na to funkcia nazývaná aktivačná funkcia. Následne je takto vytvorený signál potrebné spracovať. O to sa postará tzv. signálová funkcia. Keďže výstup, ktorý je možné dostať z neurónu je buď rovný nule, teda do okolia nie sú vysielané žiadne signály alebo rovný jednej, tak signálovu funkciu je možné chápať ako funkciu, ktorá realizuje vypínač.

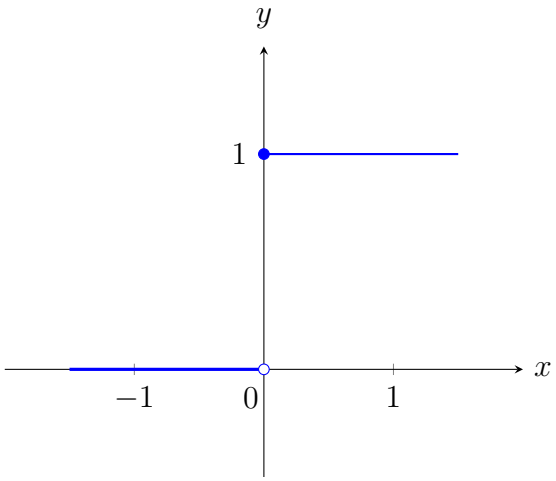


Obr. 7.2: Model formálneho neurónu

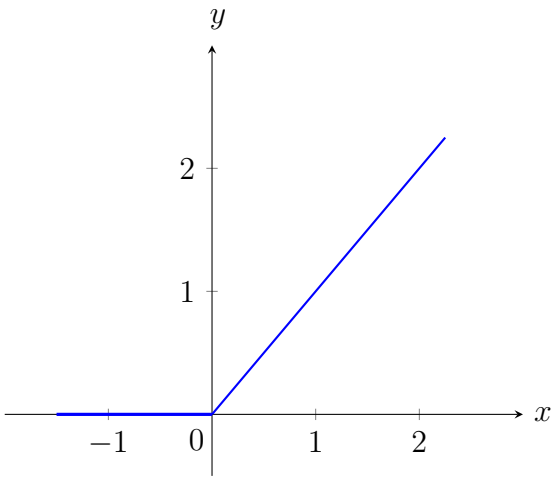
7.2 Aktivačné funkcie

Existuje viacero typov aktivačných funkcií, ktoré sa využívajú v neurónových sieťach. Základnou vecou, ktorú musia všetky používané funkcie spĺňať je schopnosť transformovať reálne hodnoty zo vstupov do intervalu $(0, 1)$ alebo $(-1, 1)$. Ďalej je potrebné klásť veľký dôraz na tvar použitých funkcií. Používajú sa také, pri ktorých funkcia prudko stúpa v okolí nuly a pri vyšších hodnotách len veľmi málo. Najpoužívanejšími predstaviteľmi takýchto funkcií sú sigmoida alebo hyperbolický tangens. Obidve sa vyznačujú diferencovateľnosťou, čo umožňuje pri učení neurónových sietí použiť algoritmy založené na gradientových metódach.

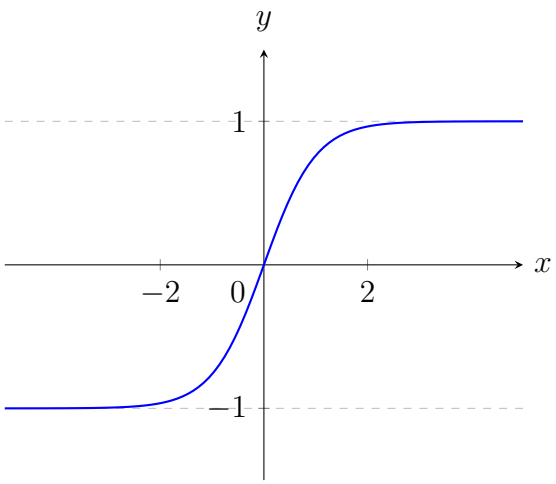
Medzi veľmi používané funkcie patria aj Relu alebo ostrá nelinearita. Ide o jednoduchšie funkcie, ktoré sa diferencovateľnou nevyznačujú a pri ich použití je potrebné použiť iné metódy ako gradientové na nastavovanie parametrov v sieťach. Priebehy vyššie spomínaných funkcií sú zobrazené na obr. č. 7.3.[21][1]



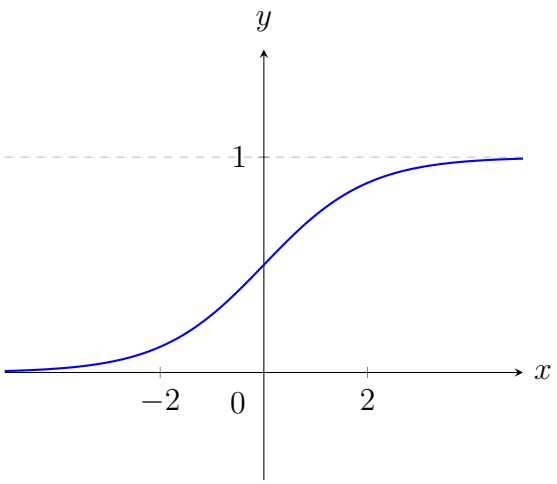
(a) Ostrá nelinearita



(b) Relu



(c) hyperbolický tangens



(d) sigmoida

Obr. 7.3: Príklady aktivačných funkcií

7.3 Neurónové siete

Neurónové siete vznikajú zložením neurónov, ktoré sú navzájom medzi sebou prepojené. Je možné ju definovať ako výpočtový graf, ktorý obsahuje uzly (výpočtové jednotky) a orientované hrany prenášajúce numerickú informáciu cez jednotlivé uzly. Každý neurón je schopný vyhodnotiť na vstupe jednoduchú funkciu. Teda neurónová sieť reprezentuje reťazec kompozícií funkcií, ktorý transformuje vstupný vektor na výstupný.

Je zložená z veľkého množstva neurónov, ktoré je potrebné usporiadať do niekoľkých častí, ktorými bude postupne prechádzať informácia zo vstupu až na výstup. Tieto časti sa nazývajú vrstvy neurónovej siete a delia sa na:

- *vstupná vrstva* - Na vstup je privedená informácia z okolitého sveta a výstup z neurónu je súčasne vstupom ďalšieho neurónu.
- *skrytá vrstva* - Na vstup je privedená výstupná informácia z predchádzajúceho neurónu a výstup pokračuje ďalej do neurónovej siete, teda sa stáva vstupom ďalšieho neurónu.

- *výstupná vrstva* - Na vstup znova prichádza informácia z predchádzajúceho neurónu, ale výstup je konečný. Získala sa informácia, ktorá sa od požadovala.

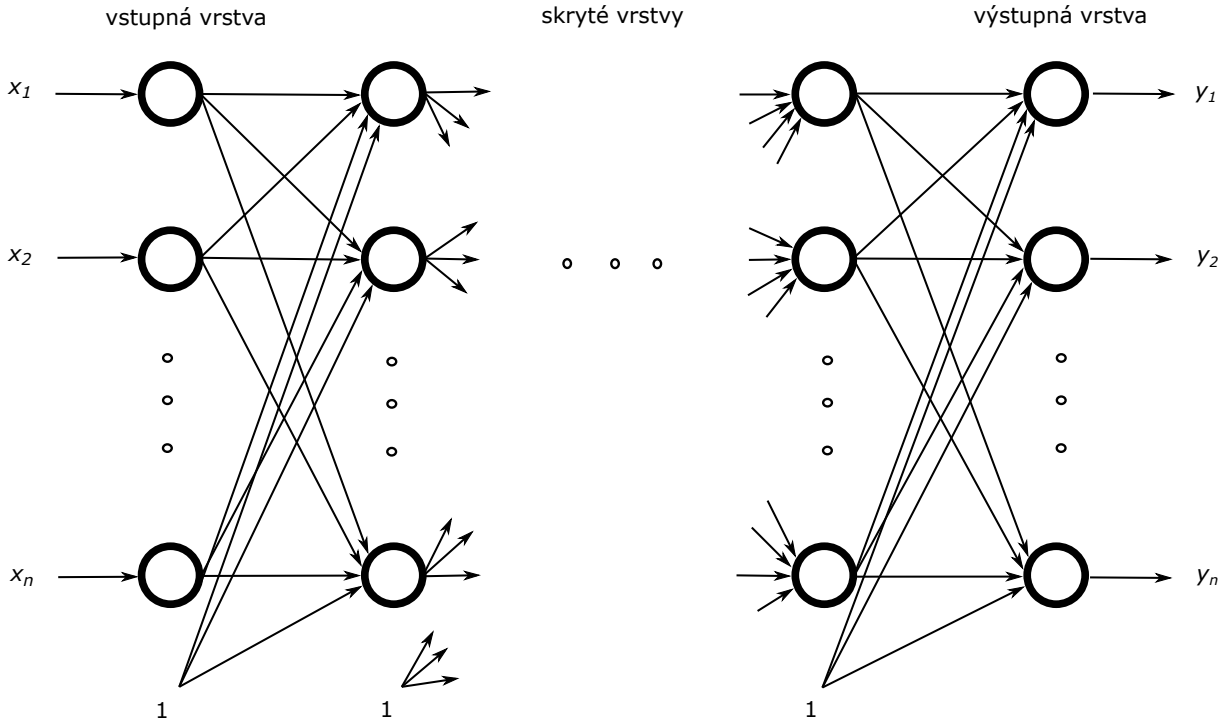
Tieto vrstvy sú vyobrazené na obr.č. 7.4.

Neurónové siete sú veľmi zložité a možno si kladiete otázku, či by nebolo lepšie použiť iné metódy, ktorými by sa dosiahol podobný výsledok. Možné to je, ale je potrebné si uvedomiť, že pomocou sietí je možné sledovať... Neurónové siete je možné rozdeliť do dvoch základných skupín a to na:

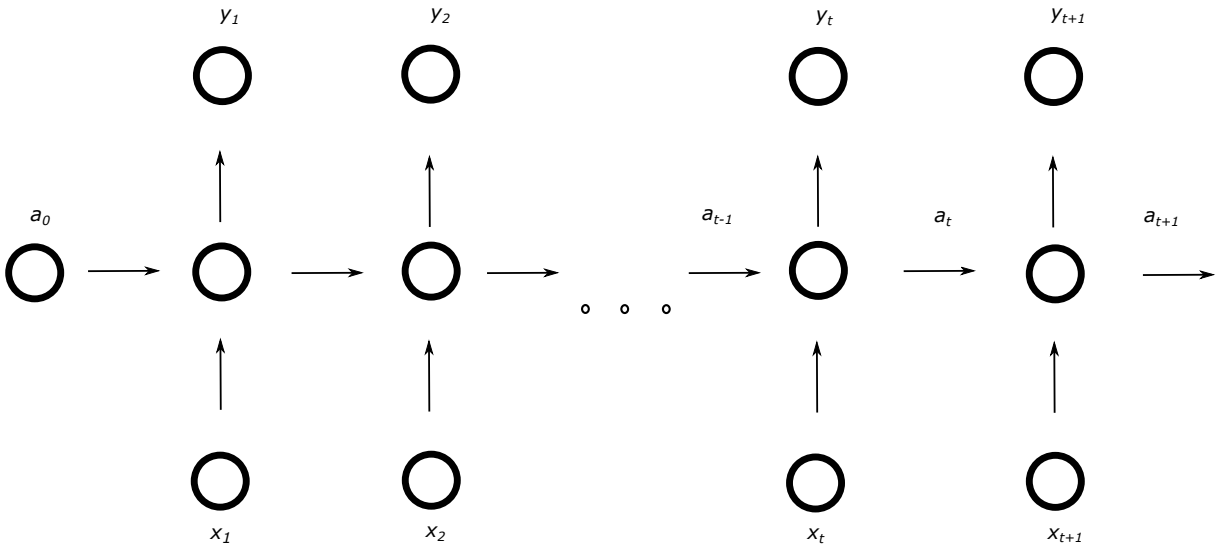
1. Dopredné neurónové siete (Feed-forward neural network)
   Sú to siete, ktoré neobsahujú spätné väzby v prepojení neurónov, teda neexistuje spojenie, ktoré by sa vracalo smerom ku vstupu.

2. Rekurentné neurónové siete (Recurrent neural network)
   Ide o siete, ktoré obsahujú spätné väzby.[3]

Príklady oboch sietí sú na obr č. 7.4 a 7.5. Je medzi nimi zásadný rozdiel a to, že rekurentná sieť je schopná postihnúť dynamiku, zatiaľ čo dopredná nie. No súčasne je aj komplikovanejšia, a preto sa v praxi častejšie využívajú práve dopredné siete a sú spravené tak, aby boli schopné aproximovať aj nejaké dynamické deje. Okrem týchto dvoch základných skupín existujú aj špecifické typy neurónových sietí, ako napríklad Kohonenove siete alebo Hopfieldove siete, ktoré sú podrobnejšie vysvetlené v [31].



Obr. 7.4: Topológia doprednej neurónovej siete



Obr. 7.5: Topológia rekurentnej neurónovej siete

Činnosť neurónových sietí sa vo všeobecnosti môže rozdeliť do dvoch fáz.

1. Fáza učenia
2. Aktívna fáza

V prvej fáze sa sieť trénuje. Ide o proces, v ktorom sa upravujú váhy medzi jednotlivými neurónmi tak, aby sa minimalizovala chybová funkcia. Učenie je možné chápať ako adaptáciu neurónovej siete, na konci ktorej bude sieť obsahovať váhy, použitím ktorých by sa mal dosahovať najlepší výsledok. Druhá fáza, teda aktívna fáza siete, vezme natrénovanú sieť aj hodnoty váh a použije ju na riešenie nejakého reálneho problému.

Veľmi podstatnou vlastnosťou neurónových sietí je práve spomínané učenie. V dobách pred využívaním neurónových sietí bolo potrebné na riešenie problému vytvoriť algoritmus, na základe ktorého sa daný problém vypočítal. Avšak, pri neurónových sieťach je navrhnutý všeobecný algoritmus a ten daná sieť používa vo fáze učenia. Správnosť a vhodnosť tohto algoritmu určuje kvalitu a rýchlosť učenia. V nasledujúcej časti bude podrobnejšie popísané učenie siete.[8][31]

7.4 Učenie

Pri tvorbe textov sa čerpalo [31][32]. Učenie neurónovej siete je jej základná vlastnosť, pri ktorom dochádza k zmene parametrov. Cieľom učenia je dosiahnutie žiadanej hodnoty medzi výstupom z neurónovej siete a požadovaným výstupom. Pri procese učenia ide teda o postupné upravovanie váh medzi jednotlivými neurónmi. Medzi nastaviteľné parametre okrem váh patria aj zmena štruktúry neurónovej siete alebo zmena strmosti aktivačných funkcií. Učenie môže teda prebiehať:

- **Bez učiteľa**
  Učenie bez učiteľa je prístup, v ktorom nie sú presne určené triedy výstupov, ale algoritmus má vo vstupných dátach nájsť určité vlastnosti medzi, ktorými existujú súvislosti a následne podľa nich vytvoriť jednotlivé triedy. Tento druh učenia funguje na základe zhlukovej analýzy.

Vysvetlenie ako sa získajú jednotlivé zhluky, inak nazývané clustery, je nasledovný. Dáta sa rozdelia do niekoľkých častí, clusterov, podľa určitých spoločných vlastností. Z každej časti sa vezmú funkčné hodnoty, prostredníctvom ktorých sa vypočíta reprezentatívna hodnota každého clusteru. Touto hodnotou môže byť napríklad ťažisko. Následne sa táto hodnota prehlási za hodnotu reprezentujúcu celý cluster a bude trvať až do ďalšieho clusteru. Takýto postup sa aplikuje v smere x aj y. Výstupom sú teda dáta, ktoré sú rozdelené do samostatných množín.

- **S učiteľom**
  Na rozdiel od učenia bez učiteľa pri pri učení s učiteľom existuje množina tzv. vzorov s klasifikáciami, ktoré sú potrebné pre určenie chybovej funkcie, ktorá je zásadným prvkom tohto učenia. Hodnota tejto funkcie sa získa ako rozdiel medzi aktuálnym vzorom a výstupom zo siete a vyjadruje chybu o koľko sa výstup z neurónovej siete odlišuje od vzoru. Následne sa podľa tejto chyby a prostredníctvom algoritmu spätného šírenia upravia váhy a celý proces sa zopakuje od začiatku až kým sa na výstupe nezíska hodnota, ktorá odpovedá požadovanej minimálnej chybe.

  Tento typ učenia sa využíva pri aplikáciách, kedy je možné vytvoriť alebo nejakým spôsobom získať množinu vzorov, ktorá sa použije ako tréningová množina neurónovej siete.

Po použití či už učenia s učiteľom alebo bez neho by mal byť systém schopný priradiť neznáme vzory triede, ktorá najlepšie odpovedá rozpoznaným vlastnostiam. Táto fáza sa nazýva rozpoznávanie. A celý proces učenia je známy aj pod pojmom automatická klasifikácia vzorov.

Keďže cieľom práce je detegovať cestu na základe vopred pripravených fotografií tak učenie neurónovej siete bude spravené na základe učenia s učiteľom. V ďalšej časti sa teda podrobnejšie vysvetlí tento prístup.

7.4.1 Učenie s učiteľom

Informácie v nasledujúcej časti sa čerpali z [31]. Podstata metódy učenia s učiteľom, už bola vysvetlená v úvode. Pre podrobnejšie vysvetlenie je potrebné ako prvé zaviesť určité predpoklady. Tie sú nasledovné:

- Predpokladá sa dopredná sieť s n vstupnými a m výstupnými jednotkami. Skrytých jednotiek môže obsahovať ľubovoľné množstvo.
- Tréningová množina $\{(x_1, t_1), ..., (x_p, t_p)\}$, pozostávajúca z p usporiadaných dvojíc n a m dimenzionálnych vektorov. Nazývajú sa vstupné a výstupné vzory.
- Aktivačná funkcia f, ktorú obsahuje každý neurón (výpočetná jednotka) musí byť spojitá a diferencovateľná.
- V prvej iterácií váhy predstavujú náhodne vybrané reálne čísla.

Potom postup učenia neurónovej siete je možné popísať nasledovne. Ako prvé sa na vstup privedie z tréningovej množiny vstupný vzor x_i. Na základe tohto vstupu sieť vypočíta výstup, ktorý sa označí o_i. Tento výstup zo siete je rozdielny od výstupného vzoru t_i. Pre všetky $i = 1, ..., p$ sa požaduje aby bol každý výstup o_i dostatočne blízky

výstupnému vzoru t_i. Následne je potrebné dosiahnuť minimalizáciu chybovej funkcie, pričom učiteľom neurónovej site sa rozumie práve táto chybová funkcia. Na to, aby bola dosiahnutá požadovaná minimalizácia tejto funkcie, sa na vstup privedie ďalší vstupný vzor a predpokladá sa, že neurónová sieť rozpozná, či je nový vstup podobný tomu učenému alebo nie. K nájdeniu lokálneho minima sa používa algoritmus spätného šírenia, ktorý na zníženie chybovej funkcie využíva gradient.

Keď sa neurónová sieť rozšíri o chybovú funkciu tak ako je to zobrazené na obr č. 7.6, tak výstupom je celková chybová funkcia E. Váhy v takejto sieti je potom možné modifikovať tak, aby bola kvadratická odchýlka E čo najnižšia. Keď sa vezme v úvahu tvar rozšírenej siete, tak hodnotu E je možné prehlásiť za spojitú a diferencovateľnú funkciu, pretože sa počíta iba pomocou skladania funkcií neurónov. Následne je možné potom túto chybovú funkciu minimalizovať použitím iteratívneho procesu gradientného zostupu.

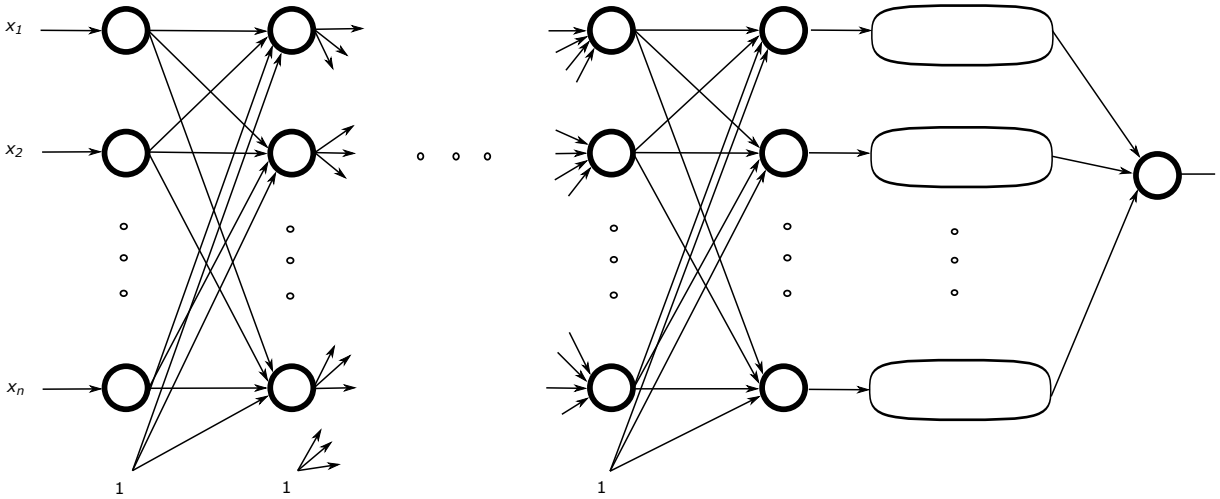
$$\nabla E = \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_k} \tag{7.2}$$

Kde $w_1, ..., w_k$ sú váhy siete. Potom každá váha je modifikovaná podľa rovnice (7.3).

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i}, \text{ pre } i = 1, ..., k \tag{7.3}$$

Kde γ je tzv. konštanta učenia.

Pokiaľ sa na výpočet gradientu funkcie siete vzhľadom k jej váham použije vhodná metóda, tak potom je možné váhy v sieti nastavovať pri každom prechode. Toto by mal zaručiť algoritmus spätného šírenia inak známy aj ako back propagation, ktorý je vysvetlený v [31] [25].



Obr. 7.6: Neurónová sieť rozšírená o chybovú funkciu

7.5 Konvolučné neurónové siete

Konvolučné neurónové site sa radia medzi špeciálne neurónové siete, ktoré na vstupe očakávajú obrazové dáta. Tieto siete aj keď boli vymyslené už v 80-tych rokoch minulého storočia sa do popredia záujmu dostali len nedávno. Prelom nastal v súťaži Imagenet Visual Recognition Challenge (ILSVRC), ktorá je zameraná na klasifikáciu obrazu

použitím rôznych algoritmov. V tejto súťaži sa prvý krát ukázalo, že použitie konvolučných neurónových sietí na rozpoznávanie obrazov s minimálnym predspracovaním je možné s oveľa lepšími výsledkami ako pri použití iných metód. Od vtedy sa začali tieto siete masovo používať vo všetkých odvetviach.[24]

Ide o typ sietí, ktoré taktiež ako neurónové siete popísané v prechádzajúcom texte, obsahujú parametre, ktoré sa v procese učenia modifikujú. Ich štruktúra taktiež obsahuje vrstvy s neurónmi pričom každý môže byť aktívny alebo nie v závislosti na funkcií. Zásadným prvkom, ktorým tento typ sietí odlišuje od iných neurónových sietí je použitie konvolučnej vrstvy.

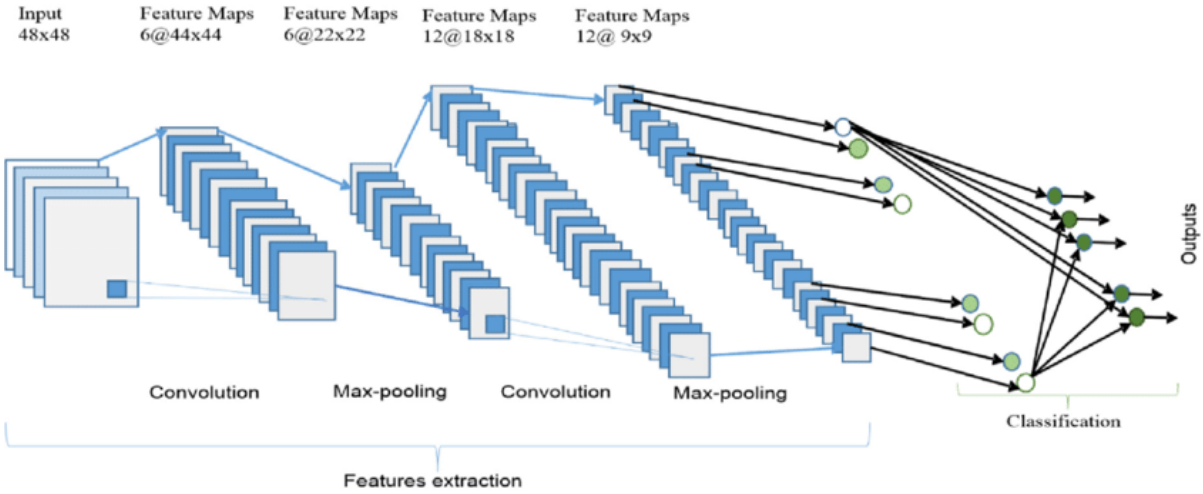
Konvolučné neurónové siete sa vyznačujú týmito charakteristickými črtami:

- lokálna konektivita
  Neuróny, do ktorých majú prichádzať vstupné informácie nie sú spojené s celým vstupom, ale len jeho malou časťou.

- zdieľanie váh
  Neuróny, ktoré sú uložené v jednej vrstve zdieľajú rovnaké hodnoty parametrov, čím sa zmenšuje ich celkový počet, ktorý je potrebný počas trénovania ukladať a následne modifikovať.

- odolnosť voči posunutiu
  Vyplýva z princípu fungovania výberu maximálnej hodnoty v zvolenej oblasti.

Príklad takéhoto typu siete je zobrazený na obr.č.7.7. Vo všeobecnosti je tvorená niekoľkými typmi vrstiev a to konvolučnej, aktivačnej, poolingovej a plne prepojenej. Ako je možné si všimnúť jednotlivé vrstvy sa striedajú a to nasledujúcim spôsobom. Na vstupe je obraz, ktorého informácie sú privedené do prvej konvolučnej, za ňou nasleduje aktivačná a poolingová vrstva. Tieto tri vrstvy je možné niekoľko krát striedať podľa toho, aké príznaky sa požaduje v sieti detegovať. Následne sa použije plne prepojená vrstva, z ktorej sa získa konečný výstup.



Obr. 7.7: Architektúra konvolučnej neurónovej siete [2]

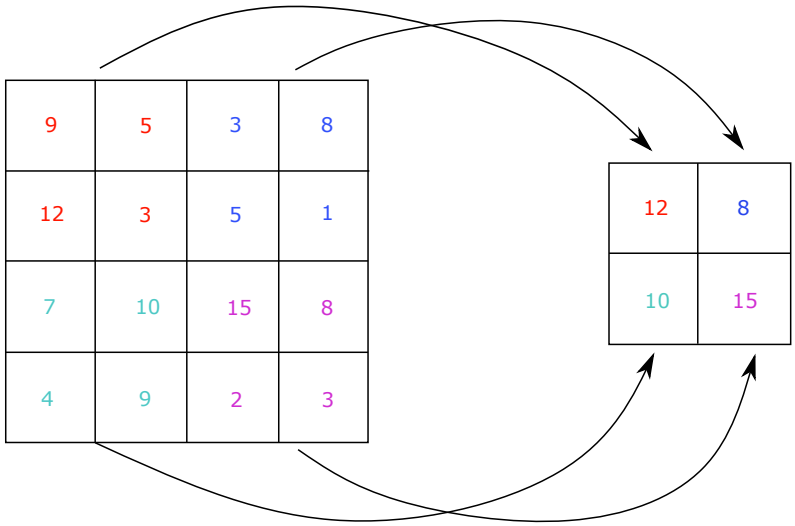
Konvolučná vrstva

Konvolučná vrstva je najdôležitejšou časťou celej siete a slúži na detekciu príznakov vo vstupnom obraze. Jej podstatou je použitie diskrétnej konvolúcie, ktorá bola definovaná v časti 4.4.3. Jej presnejšie fungovanie je popísané v [34][8].

Je dôležité poznamenať, že v rámci jednej konvolučnej vrstvy sa nerealizuje len jedna konvolúcia, ale hneď niekoľko. Teda každá vrstva tohto typu obsahuje niekoľko filtrov, pričom každý filter deteguje iný príznak v celom obraze. Čo je zobrazené aj na obr. č. 7.7, ktorý popisuje architektúru siete.

Poolingová vrstva

Ďalšou časťou konvolučnej neurónovej siete je tzv. poolingová vrstva. Na rozdiel od konvolučnej vrstvy, ktorej úlohou je detegovať príznaky v obraze cieľom tejto časti je zmenšenie rozmerov vstupu. Toto zmenšenie je realizované prostredníctvom hľadania maximálnych hodnôt tzv. max pooling. Princíp fungovania je zobrazený na obr. č. 7.8.



Obr. 7.8: Max pool

Veľkosť výstupu závisí na dĺžke použitého kroku a jeho hodnota na veľkosti použitého filtra. Podrobnejší postup fungovania je popísaný v [34].

Medzi výhody použitia poolingovej vrstvy patrí nezávislosť na posunutí vstupu. Ďalšou výhodou je, že je možné ju použiť ako prvú vrstvu siete pokiaľ na vstup prichádzajú snímky s rôznymi rozmermi. Táto vrstva zaistí aby sa ich rozmer zjednotil. Parametre vrstvy sa v tomto prípade volia dynamicky v závislosti na veľkosti vstupu. Pokiaľ sa použije ako skrytá vrstva siete, teda po konvolučnej vrstve, tak na vstup je privedená aktivačná mapa. Nájdenie maxima sa pre každú aktivačnú mapu vykoná samostatne. Teda počet máp sa zachová.

Plne prepojená vrstva

Poslednou vrstvou, ktorá patrí do architektúry konvolučných neurónových sietí je práve plne prepojená vrstva. Je charakteristická pre dopredné neurónové siete. Pre

pripomenutie, vstupom do neurónu sú teda všetky výstupy neurónov z prechádzajúcej vrstvy ale vrámci jednej vrstvy sú neuróny oddelené, nemajú medzi sebou žiadne spojenia. Môže byť buď skrytou alebo výstupnou vrstvou. Najčastejšie sa používa niekoľko skrytých vrstiev, ktoré sú zakončené výstupnou vrstvou, tak ako je to na obr. č. 7.7.

Cieľom plne prepojenej vrstvy je vziať výsledky po procese konvolúcie a poolingu, previesť ich do podoby vektoru a následne ich použiť na klasifikáciu obrazu. Teda výstupom konvolučnej neurónovej siete bude priradenie vstupného obrazu do triedy kam by mal podľa siete patriť. Taktiež ako v doprednej neurónovej sieti aj tu je na úpravu váh použitý algoritmus spätného šírenia.[34][8].

8 Analýza problému

V rešeršnej časti 7 bolo popísaných viacero prístupov, ako rôzni autori pristupovali k detekcií cesty prostredníctvom neurónových sietí. Všetky spomenuté prístupy dosahovali pozoruhodne dobré výsledky. Zarážajúcou vecou vo väčšine prác bolo, že vo väčšine z nich sa vo výsledkoch uviedla len výsledná dosiahnutá percentuálna úspešnosť. No pre správne natrénovanie siete je potrebné vykresliť dosiahnuté výsledky úspešnosti aj chybovosti počas celého učenia aj testovania, a to z dôvodu pretrénovania alebo podtrénovania siete. Taktiež, v niektorých prácach trénovacia množina bola veľmi malá a teda sieť by bolo možné použiť iba pre konkrétne prípady. Z uvedených prác vyplýva, že existuje viacero vhodných prístupov riešenia.

V tejto časti sa teda problém nájdenia cesty pomocou neurónových sietí analyzuje, vymedzia sa hranice a príjmu sa potrebné zjednodušenia aby sa mohlo pristúpiť k návrhu vlastného riešenia a následne k jeho realizácií.

Keďže výstupom siete má byť nájdená cesta je ako prvé potrebné zvoliť vhodnú stratégiu riešenia a reprezentáciu výsledkov. Následne, aby sa zamedzilo spomínaným nežiaducim výsledkom (pretrénovanie, podtrénovanie) je potrebné mať na učenie siete pripravenú dostatočne veľkú trénovaciu množinu, správne pripravené a roztriedené dáta, taktiež správne navrhnutú štruktúru siete a jej jednotlivé vrstvy. Výsledky siete by mali buď priamo alebo po následnom spracovaní nejakým spôsobom reprezentovať nájdenú cestu.[32]

Pri návrhu algoritmu prostredníctvom klasických metód bolo potrebné brať do úvahy veľa faktorov, ktoré by mohli ovplyvniť dosiahnuté výsledky. Medzi ne patrili napríklad typ cesty, osvetlenie cesty, okolité prostredie, objekty na ceste a mnoho iných. Pri návrhu neurónovej siete je potrebné tieto okolnosti taktiež zohľadniť, ale iným spôsobom. Zatiaľ čo pri klasický metódach sa pre každý faktor musel algoritmus upraviť, pri neurónových sieťach je potrebné, čo najviac rôznych príkladov cesty zahrnúť do trénovacej množiny. Čím väčšia táto množina bude, tým by mala byť sieť pri detekcii novej cesty, ktorá nebola zahrnutá v trénovacej množine, robustnejšia.

Pre neurónové siete je charakteristická veľká časová a pamäťová náročnosť pri trénovaní. Keďže na trénovanie sa bude používať bežne dostupná počítačová zostava je potrebné aby vstupné dáta, veľkosť siete aj počet výstupov neboli veľmi veľké. Z dôvodu obmedzeného množstva výpočetných zdrojov bude potrebné snímky prichádzajúce na vstup siete zmenšiť, štruktúru siete navrhnúť tak, aby počet nastavovaných parametrov bol čo najmenší, ale úspešnosť siete bola čo najvyššia a nedochádzalo k spomínanému pretrénovaniu alebo podtrénovaniu siete.

9 Návrh vlastného riešenia

Detekcia cesty vo vonkajšom prostredí patrí medzi komplikovanejšie úlohy, z dôvodu rôznorodosti cesty aj okolia. O tom sa presvedčilo už v prvej časti práce, ktorá sa venovala použitiu klasických metód. Algoritmus, ktorý sa použil, obsahoval veľa rôznych filtrov, segmentačných metód a úprav. Problémom okrem časovej náročnosti bola nakoniec aj robustnosť algoritmu, ktorá sa začala strácať s použitím viacerých metód.

Pristúpilo sa teda k hľadaniu iného riešenia. Keďže už v rozdelení popísanom v úvode práce, sa hľadanie cesty delilo na klasické metódy a neurónové siete, tak táto časť práce sa bude venovať práve neurónovým sieťam. V nasledujúcich častiach bude postupne vysvetlené aké riešenie na nájdenie cesty sa zvolilo, ako prebiehala príprava dát a aká štruktúra siete sa zvolila. Následne sa pristúpi k implementácií a prezentácií dosiahnutých výsledkov.

9.1 Voľba stratégie riešenia

Ako prvé bolo potrebné vybrať vhodný typ neurónovej siete. V častiach 7.3 a 7.5 boli popísané dopredané, rekurentné a konvolučné neurónové siete. Keďže je k dispozícií veľké množstvo snímok cesty, pre riešenie sa zvolila práve konvolučná neurónová sieť a získané snímky sa použijú ako trénovacie dáta.

V rešeršnej časti 7 bolo popísaných niekoľko možností, ako by bolo možné reprezentovať cestu. Prvou zvažovanou možnosťou bolo vytvorenie výstupnej vrstvy tak, že každý neurón odpovedá práve jednému pixlu na snímke. Pixel obsahuje cestu ak jeho odpovedajúci neurón je rovný jednej v opačnom prípade tento pixel prislúcha okoliu. Toto riešenie sa nepoužilo z dôvodu malého množstva pamäte. Druhou zvažovanou možnosťou bolo nájdenie hraničných bodov v obraze, prípadne bodov zániku a rozdelenie obrázka na oblasti. Pri tomto návrhu vznikol problém s vhodným určením kategórií.

Ak sa uvažuje, že snímka je rozdelená do 6×4 podoblastí a každý zo 4 hraničných bodov sa môže nachádzať v ľubovoľnej oblasti obrázka, celkový počet možných kategórií by bol 24^4. Konečným návrhom bolo opäť rozdelenie snímky na niekoľko oblastí avšak iným spôsobom. Vychádzalo sa z predpokladu, že robot zaznamenávajúci cestu sa nachádza na nej. Potom sa pristúpilo k vytvoreniu oblastí tak ako je to vyobrazené na obr. č 9.1.

Obr. 9.1: Obrázok s vyznačenými oblasťami

Z obrázku je možné vidieť, že všetky oblasti majú počiatok v jednom bode a to stred spodnej hrany. Tento bod je najbližšie skutočnej polohe robota a súčasne je aj vrcholom uhlu. Jednotlivé oblasti odpovedajú výseku s uhlom 15°. Aktuálna snímka sa zaradí do danej kategórie ak sa v odpovedajúcej oblasti nachádza bod zániku, teda miesto, kde cesta zaniká. V tomto prípade by bol obrázok zaradený do piatej kategórie.

Celý proces riešenia bude teda nasledovný. Ako prvé sa dáta roztriedia do príslušných kategórií. Následne sa vytvorí vhodná konvolučná neurónová sieť, na vstup ktorej budú privedené snímky s klasifikáciou. Výstupná vrstva siete bude navrhnutá tak, aby obsahovala práve jedenásť neurónov pričom každý odpovedá jednej kategórií. Výstup zo siete bude vektor, s hodnotami v rozmedzí $\langle 0, 1 \rangle$. Snímka patrí do príslušnej kategórie v prípade, že hodnota odpovedajúceho neurónu je najvyššia zo všetkých. Podrobne bude celé riešenie postupne popísané v nasledujúcich častiach.

9.2 Implementácia v jazyku Python

Na realizáciu neurónovej siete je možné použiť viacero programovacích jazykov. Používané sú napríklad Java, Python, C++ a iné. V tejto práci sa zvolil práve Python, z dôvodu predošlých skúseností. Python už bol popísaný v časti 4.6. Na realizáciu neurónovej siete bolo potrebné nainštalovať niekoľko prídavných knižníc.

Použité knižnice:

- Numpy
- OpenCV
- TensorFlow
- Matplotlib

Numpy je knižnica učená pre matematické operácie, ktorá umožňuje prácu s viacrozmernými poľami. V práci sa využije na vytvorenie poľa obrázkov, ktoré sa použijú na trénovanie siete.[26]

V knižnici OpenCV je obsiahnutých mnoho funkcií na spracovanie obrazu. V predchádzajúcom riešení z nej bolo použitých hneď niekoľko.[27]

Na tvorbu neurónových sietí sa používa knižnica TensorFlow, ktorej súčasťou je framework Keras, ktorý poskytuje na vytvorenie neurónovej siete užívateľsky prehľadné prostredie. Medzi jej veľké výhody patrí možnosť vykonávať výpočty ako na procesore, tak aj na grafickej karte a to pomocou CUDA jadier. Táto knižnica patrí medzi najpoužívanejšie a bude využitá aj v tejto práci.[43]

Na záver získané výsledky zo siete je potrebné vykresliť. Pre tento účel bola zvolená knižnica Matplotlib.[23]

Na písanie programov v Pythone existuje veľké množstvo editorov a IDE (Integrated Development Environment). Medzi najpoužívanejšie patria PyCharm, Visual Studio Code a mnoho iných. Pri vytvorení neurónovej siete sa v práci použila verzia Pythonu (3.6.2), Numpy (1.17.4), OpenCV (4.4.2.30), TensorFlow (2.0.0), Keras (2.2.4), Matplotlib (3.1.3) a editor Visual Studio Code.

9.3 Príprava dát

Vychádza sa z predpokladu vytvorenia konvolučnej neurónovej siete, ktorá sa bude učiť na základe metódy s učiteľom, ktorá bola popísaná v časti 7.4. Pre pripomenutie podstatou tohto typu učenia je predkladanie vzorov s klasifikáciou, pričom cieľom je určiť chybovú funkciu. Následne prostredníctvom nej nastavovať parametre siete tak aby dochádzalo k minimalizácií tejto funkcie. Keďže sa jedná o konvolučnú neurónovú sieť, tak sa na vstup budú privádzať snímky. V prvom kroku je potrebné vytvoriť trénovaciu a testovaciu množinu, ktoré budú obsahovať dáta roztriedené do jednotlivých kategórií. Je vhodné pripraviť, čo najväčšie množstvo dát a zároveň je dôležité klásť dôraz aj na ich rôznorodosť. Jednoduchšie povedané je dobré získať čo najviac snímok cesty, ktoré sa od seba odlišujú napr. textúrou, farbou, natočením, osvetlením a pod.

Dáta pre túto prácu boli získané z mobilného robota B2, ktorý zaznamenával cestu v parku Lužánky. Ide o kolekciu dát zozbieraných z niekoľkých rokov. Všetky obrázky sú rozmeru 480×360 pixlov. Na vytvorenie potrebných množín sa použila väčšina z nich s výnimkou tých snímok, na ktorých sa cesta nenachádzala. Prípad rozpoznávania, či sa cesta na obrázku nachádza alebo nie, nebude do trénovania neurónovej siete zahrnutý.

V nasledujúcich podkapitolách bude bližšie popísané ako sa vytvorili jednotlivé kategórie, následné rozdelenie na trénovaciu a testovaciu množinu a ich doplnenie upravenými dátami.

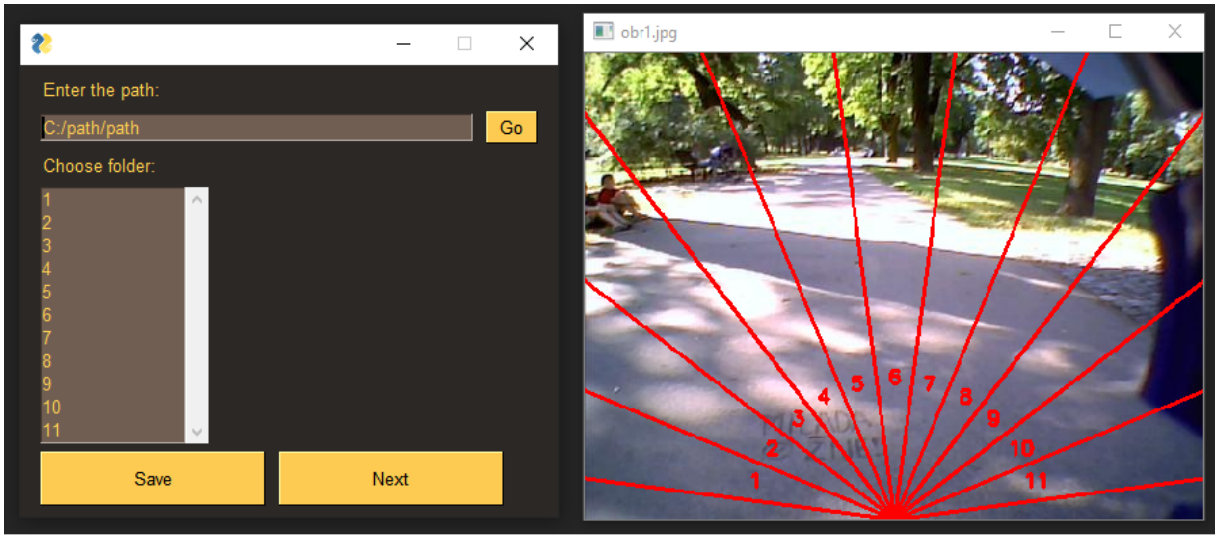
9.3.1 GUI

Pred procesom trénovania siete je potrebné získané snímky rozdeliť do príslušných kategórií. Ide o časovo náročnú činnosť, a preto sa bralo v úvahu viacero možností ako získané dáta roztriediť.

Prvou možnosťou bolo manuálne riešenie. Postupne každú snímku otvoriť, rozhodnúť do ktorej kategórie patrí a následne ju priradiť do príslušnej zložky. Pri prvom návrhu neurónovej siete sa takýto proces uplatnil, ale ukázal sa ako časovo veľmi neefektívny. Počet snímok, ktoré sa spracovali za hodinu, bolo približne tristo a to išlo iba o čisté roztriedenie do zložiek. Následné rozdelenie na trénovaciu a testovaciu množinu, prípadne ďalšie úpravy v tomto procese neboli zahrnuté.

Druhou možnosťou bolo vytvorenie jednoduchého poloautomatického GUI (Graphical User Interface), v ktorom sa pomocou niekoľkých klikov načítajú všetky obrázky, užívateľ len rozhodne o tom, do ktorej zložky chce aktuálne otvorenú snímku priradiť a systém všetko ostatné spraví za neho.

Keďže druhá možnosť sa javila ako rýchlejšia a použiteľná aj v prípade ďalšieho rozšírenia trénovacej a testovacej množiny zvolila sa práve táto možnosť. Vytvorilo sa jednoduché GUI, tak ako je možné vidieť na obr. č. 9.2. Princíp fungovania je nasledovný. V prvom kroku sa zadá cesta, v ktorej sa nachádzajú dáta na roztriedenie do kategórií. Následne po načítaní dát sa zobrazí prvá snímka. V ďalšom kroku je potrebné zvoliť, do ktorej kategórie daná snímka patrí a stlačiť tlačítko „save“. Nakoniec je možné prejsť na novú snímku pomocou tlačítka „next“.



Obr. 9.2: GUI

Na obr. č. 9.2 je možné vidieť snímku cesty, do ktorej sú nakreslené jednotlivé oblasti. V tomto prípade cesta zaniká v oblasti č. 5 a podľa toho sa zvolí aj príslušný priečinok, do ktorého sa táto snímka uloží.

Pokiaľ načítaná snímka neobsahuje cestu alebo z nejakého iného dôvodu nemá byť v nasledujúcom procese použitá jednoducho sa stlačí tlačítko „next“ a prejde sa na ďalšiu snímku. Po stlačí tlačítka „save“ sa má aktuálna snímka uložiť do zvoleného priečinku. Pokiaľ priečinok neexistuje vytvorí sa automaticky nový s príslušným názvom.

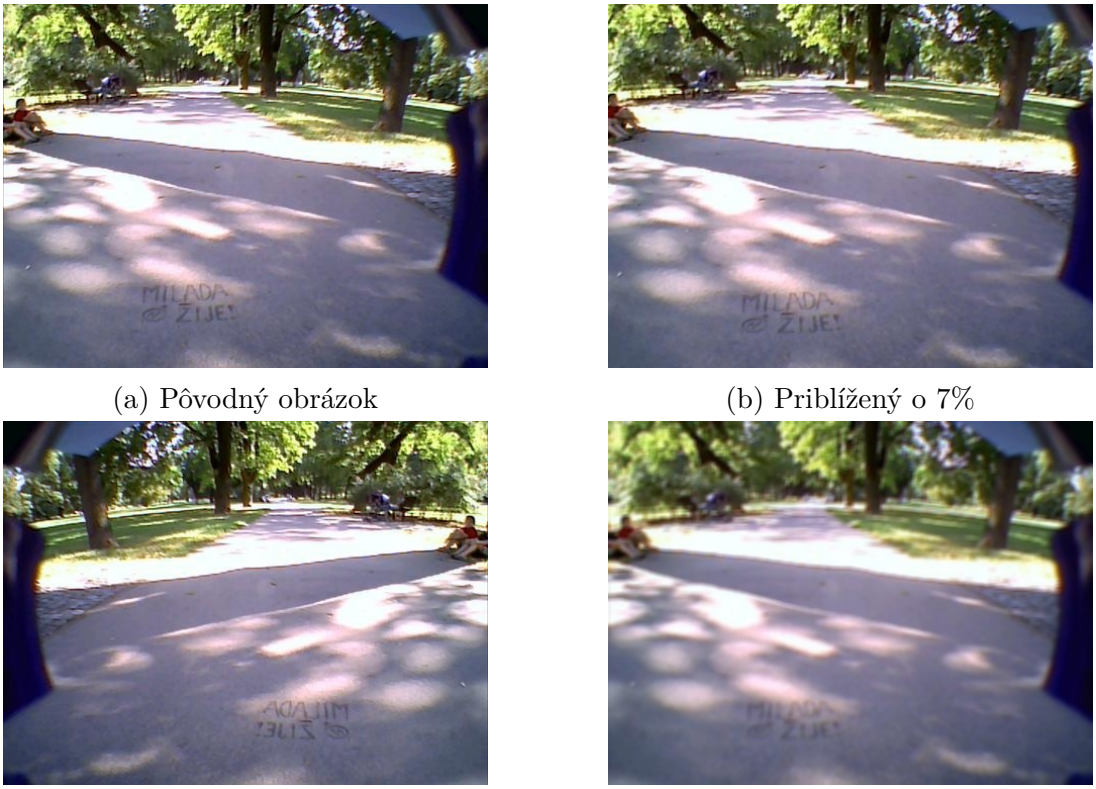
Po roztriedení všetkých snímok je potrebné vytvorené kategórie rozdeliť na trénovacie a testovacie dáta. Doporučuje sa, že je vhodné použiť pomer 80/20. Teda 80 percent zo všetkých dát by malo byť obsiahnutých v trénovacej množine a zvyšných 20 percent v testovacej množine. Po roztriedení dát sa automaticky všetky zložky v príslušnej ceste rozdelia vo zvolenom pomere na trénovacie sa testovacie dáta. Výsledkom sú teda dve zložky - dáta na trénovanie a dáta na testovanie. Vo vnútri každej z nich sa nachádzajú snímky roztriedené do príslušných kategórií.

9.3.2 Doplnenie trénovacej a testovacej množiny

Po vyššie popísanom roztriedení by bolo možné už pristúpiť k návrhu štruktúry neurónovej siete. Odporúča sa roztriedené dáta ešte upraviť, trochu pozmeniť. Tento krok je vhodný z viacerých dôvodov. Prvým a aj hlavným dôvodom rozšírenia množín je zväčšenie počtu snímok v kategóriách, v ktorých ich nie je obsiahnutých veľké množstvo snímok. Jedná sa zväčša o okrajové kategórie (01, 02, 03, 09, 10, 11). Druhým je, že výsledná natrénovaná sieť bude robustnejšia.

Opäť ako pri triedení obrázkov aj tento krok bol naprogramovaný tak, aby sa všetko vykonalo automaticky. V prvom kroku sa roztriedené dáta načítajú. Následne podľa toho, o ktorú kategóriu sa jedná, sa zvolí koľko percent zo všetkých snímok nachádzajúcich sa vo zvolenej zložke sa upraví. Okrajové kategórie budú obsahovať viac upravených snímok ako tie prostredné. Potom sa pristúpi k samotnej úprave. Zvolilo sa hneď niekoľko zmien. Prvou z nich je priblíženie. Pôvodná snímka sa priblíži o 7%, pričom zvolená hodnota sa určila s ohľadom na možnú zmenu zaradenia pri väčšom priblížení. Ďalšou vykonanou zmenou je prevrátenie snímky horizontálne. Pri tomto type úpravy bolo potrebné výsledný obrázok

uložiť do zložky, ktorá aktuálne odpovedá miestu zániku cesty. Poslednou úpravou, ktorá bola použitá bolo rozmazanie snímky prostredníctvom Gaussovho filtru. Zvolená maska bola rozmeru 5×5. Všetky vyššie popísané úpravy spolu s pôvodným obrázkom sú vyobrazené na obr. č. 9.3.



(a) Pôvodný obrázok

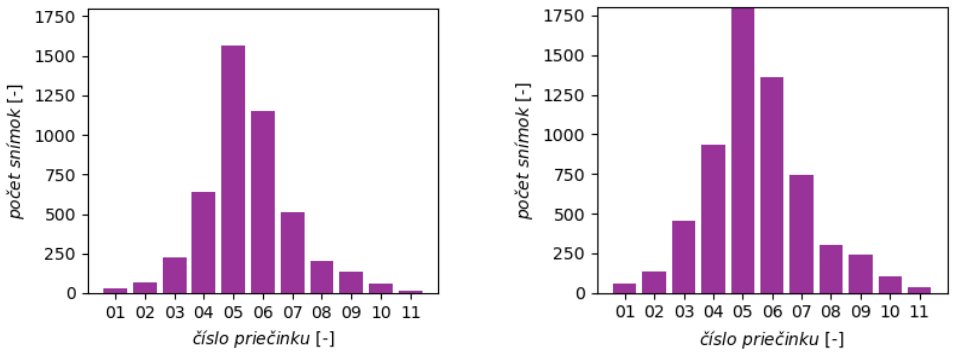
(b) Priblížený o 7%

(c) Stranovo prevrátený

(d) S Gaussovým filtrom

Obr. 9.3: Zvolené úpravy snímok

Pomocou vyššie spomínaných úprav sa množiny dostatočne rozšírili. Z pôvodných 6032 obrázkov v trénovacej množine po úpravách vzniklo 6514 obrázkov. Grafy na obr. č. 9.4 znázorňujú množstvo snímok v jednotlivých kategóriách pred a po doplnení.



(a) Pred doplnením

(b) Po doplnení

Obr. 9.4: Grafy vyjadrujúce rozloženie dát v jednotlivých kategóriách. (a) Pôvodné kategórie (b) Kategórie rozšírené o upravené dáta

Z grafov je jednoznačne vidieť, že došlo k značnému navýšeniu dát v okrajových kategóriách, čo bolo aj hlavným cieľom. S takouto trénovacou, ale aj testovacou množinou je možné pristúpiť k tomu najdôležitejšiemu, návrhu neurónovej siete.

9.4 Návrh štruktúry siete

Pri návrhu konvolučnej neurónovej siete nie je možné na začiatku presne povedať, z koľkých konvolučných, poolingových a plne prepojených vrstiev sa bude skladať. Celý návrh štruktúry sa odvíja od týchto nasledujúcich podmienok.

- Sieť by mala byť navrhnutá tak, aby v nej nedochádzalo k overfittingu a underfittingu.
- Mala by dosahovať čo najvyššiu presnosť.
- Hodnota chyby by mala byť čo najnižšia.

Aj keď sa splnenie tých podmienok môže zdať ako jednoduchá záležitosť opak je pravdou. Okrem nájdenia vhodného počtu vrstiev je potrebné správne nastaviť aj množstvo parametrov v sieti.

Prvá testovaná štruktúra siete

Pred návrhom prvej štruktúry bolo spravených niekoľko testov na malej množine trénovacích dát (350 snímok). Zoznámilo sa s použitím jednotlivých vrstiev, nastavením jednotlivých parametrov a overilo sa, či je možné dostať zo siete výstup taký ako sa požaduje.

Následne sa prešlo na väčšiu množinu dát (6000 snímok), rozdelených na trénovaciu a testovaciu množinu, tak ako to bolo popísané v kapitole 9.3. Na vstup siete sa priviedli zmenšené dáta (120x90 px). Toto zmenšenie bolo nutné z dôvodu malého množstva pamäte. Navrhla sa prvá štruktúra neurónovej siete. Počet vrstiev a jednotlivé parametre, ktoré boli nastavené zobrazuje tabuľka č. 9.1.

Typ vrstvy	Počet filtrov	Veľkosť jadra	Výstupný tvar	Počet parametrov
Konvolučná	32	3x3	90x120x32	896
Poolingová	-	2x2	45x60x32	0
Konvolučná	32	3x3	45x60x32	9248
Poolingová	-	2x2	22x30x32	0
Vyrovnávacia	-	-	21120x1	0
Plne prepojená	-	-	700x1	14784700
Plne prepojená	-	-	60x1	42060
Výstupná	-	-	11x1	671
$\sum$				14837575

Tabuľka 9.1: Navrhnutá štruktúra siete

Zvolila sa sieť s tromi konvolučnými, dvomi poolingovými, dvomi skrytými plne prepojenými vrstvami a výstupnou vrstvou. V konvolučnej vrstve sa zamedzilo zmenšeniu výstupu a ako aktivačná funkcia sa zvolila ReLu.

O takto navrhnutej štruktúre nie je možné povedať nič konkrétnejšie, kým sa nepristúpi k jej otestovaniu. Až na základe získaných priebehov chyby a presnosti v závislosti na epoche, je možné zhodnotiť, ktoré parametre je potrebné do siete pridať prípadne pozmeniť.

```
with tf.device('/gpu:0'):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), padding="same", activation='relu',
                            input_shape=(90, 120, 3))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(32, (3, 3), padding="same", activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(700, activation='relu'))
    model.add(layers.Dense(60, activation='relu'))
    model.add(layers.Dense(11, activation='softmax'))
    model.summary()
```

Obr. 9.5: Navrhnutá štruktúra napísaná v programe Python

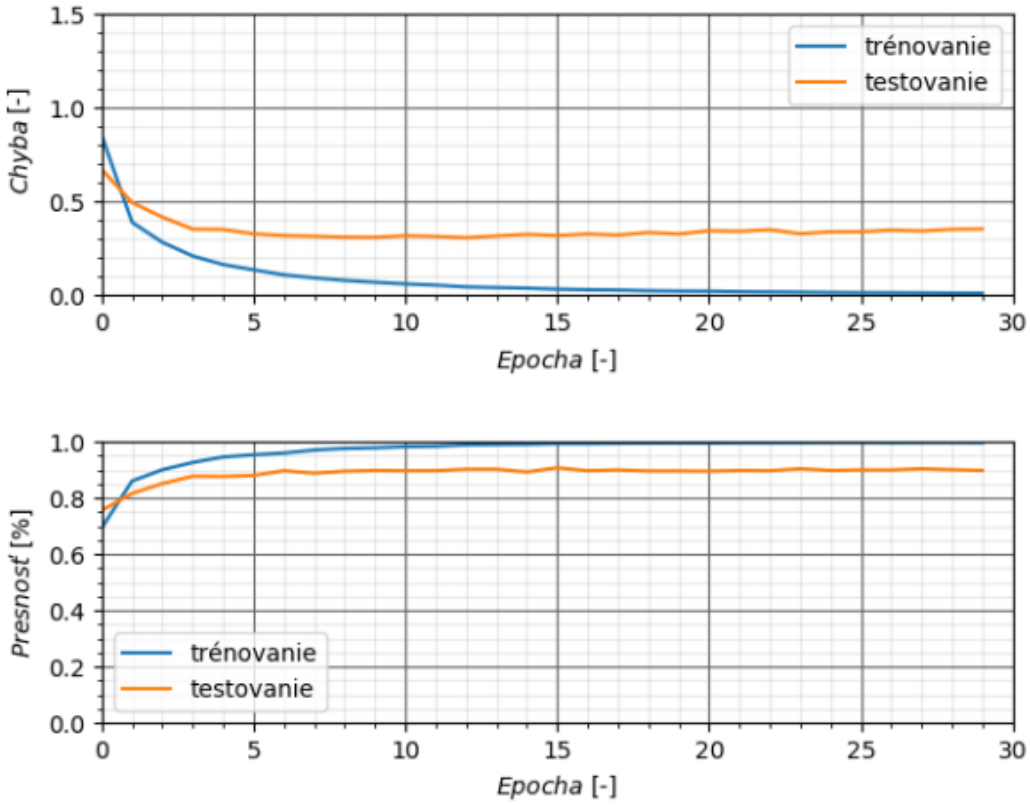
Trénovanie siete

Tak isto ako návrh siete aj tréning siete bude zrealizovaný prostredníctvom programu Python a prídavných knižníc, ktoré boli popísané v časti 9.2. Pred zahájením trénovania bolo potrebné zvoliť vhodný optimalizačný algoritmus, typ chybovej funkcie. taktiež počet epoch a veľkosť batchu. Zvolené hodnoty, ktoré sa na použili sú v tabuľke č. 9.2.

Optimalizačný algoritmus	Typ chybovej funkcie	Počet epoch	Batch
Adam	categorical crossentropy	30	64

Tabuľka 9.2: Hodnoty nastavené pri procese trénovania

Podrobnejšie sú jednotlivé parametre a aj možnosti ich nastavenia popísané v [17]. V ďalšom kroku sa mohlo pristúpiť k trénovaniu siete. Celkovo sa použilo 6000 snímok rozdelených na trénovaciu a testovaciu množinu. Hodnota chyby aj aktuálne dosiahnutej presnosti siete sa ukladala po každej epoche. Po skončení celého procesu sa tieto ukladané dáta vykreslili do grafov, ktoré sú zobrazené obr. č. 9.6.



Obr. 9.6: Grafy zobrazujúce priebeh chyby a presnosť v závislosti na epoche

Z grafov jednoznačne vyplýva, že došlo k značnému pretrénovaniu siete. Prejavilo sa to tak, že hodnota chyby pri testovaní sa postupne zvyšovala, zatiaľ čo pri tréningu postupne klesala k nule. Ďalšou indikáciou pretrénovania je dosiahnutie takmer sto percentnej presnosti učenia. Je viacero možností ako potlačiť pretrénovanie siete a to:

- Zjednodušenie modelu.
- Skoré zastavenie trénovania siete.
- Rozšírenie vstupných dát.
- Použitie regularizácie (Dropout, L1, L2 regularization).

Keďže vstupné dáta už boli rozšírené v 9.3.2, tak táto možnosť sa zatiaľ vylúčila. Taktiež zmena hodnoty počtu epoch by v tomto prípade bola neúčinná, keďže k pretrénovaniu začína dochádzať už po prvej epoche. K úprave štruktúry modelu sa zatiaľ pristupovať nechcelo, teda v ďalšom kroku sa bude navrhnutá štruktúra optimalizovať pridaním a zmenou viacerých parametrov.

Optimalizácia parametrov

V tejto časti sa postupne vysvetlí aké parametre je vhodné do navrhnutej siete pridať aby sa predišlo pretrénovaniu a zároveň sa dosiahla čo najmenšia chyba. Ako prvé sa popíšu parametre pridané do siete. Patrí medzi ne:

- Dropout
  Jeho podstatou je, že vo fáze učenia určitú časť neurónov neuvažuje. Podrobnejšie je vysvetlený v [17]. Oporúča sa hodnotu nastavovať v rozmedzí 0.2-0.5. V sieti sa medzi každú skrytú plne prepojenú pridal s hodnotou 0.4.

- L1, L2 regularization
  Sú to parametre, ktoré sa pridávajú k chybovej funkcií a ich podrobnejšie vysvetlenie je možné nájsť v [17]. V práci sa použil parameter L2 v konvolučných vrstvách. Jeho hodnotu sa odporúča nastavovať na 0,1, 0,01, 0,001,.... Na vylepšenie procesu trénovania sa zvolila hodnota 0,0001.

- Bias
  Posledným parametrom, ktorý sa pridal do siete. Nastavil sa na hodnotu jedna, čo vyjadruje, že daná vrstva má použiť bias.

Okrem pridaných parametrov,by bolo vhodné pristúpiť ešte k niekoľkým úpravám. Keďže veľkosť obrázku, ktorý prichádza na vstup siete je 120×90 pixlov, v prvej konvolučnej vrstve by bolo vhodné použiť väčšiu masku. Mal by sa zvoliť rozmer 5×5. Je to z toho dôvodu, že rozmer snímky je stále dostatočne veľký a v prvej vrstve by sa mali detegovať príznaky, ktoré sú rozsiahlejšie. Taktiež sa doporučuje zníženie hodnoty filtrov v prvej vrstve. V literatúre [33] sa odporúča začať na prvej konvolučnej vrstve s menším počtom filtrov a postupne ich zvyšovať.

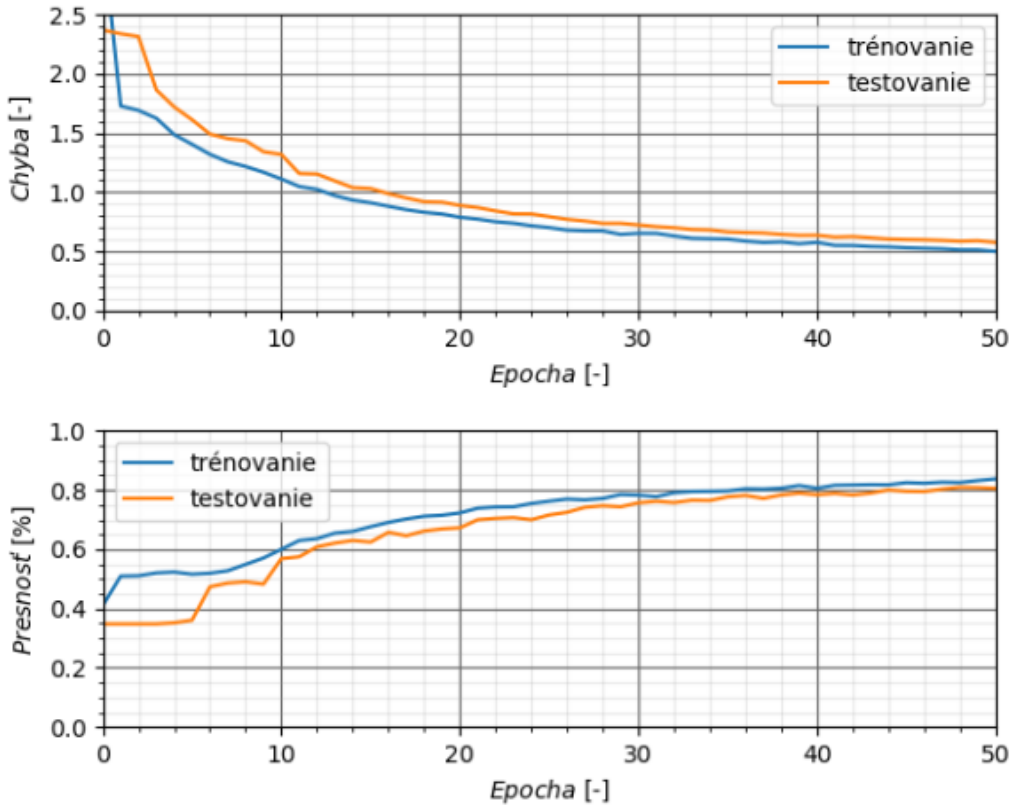
Samozrejme je možné kombinovať toho ešte viac. Môže sa zmeniť aktivačná funkcia, typ chyby tak isto aj optimalizačný algoritmus. Vhodné je preštudovať dokumentáciu ktorú poskytuje samotný Keras [17] a skúšať, pri akej kombinácií parametrov je možné dosiahnuť najlepšie výsledky.

Ďalšie návrhy siete

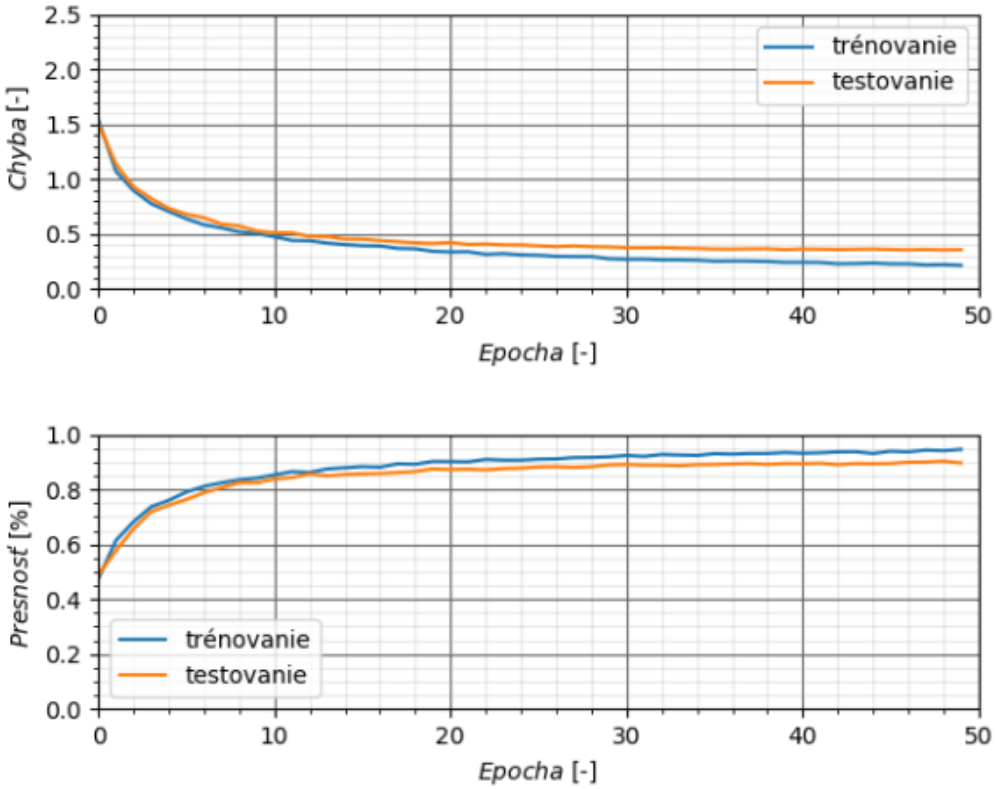
V tejto časti sa uvedie niekoľko návrhov siete, ktoré boli realizované s ohľadom na predošlé výsledky a s použitím optimalizačných parametrov. Celkovo sa navrhli tri výsledné konvolučné neurónové siete, z ktorých sa vybrala tá, ktorá dosahovala najlepšie výsledky. Postupne bude popísaná štruktúra každej z nich a ku každej bude pridaný graf, ktorý bude vyjadrovať priebeh chyby a presnosti počas trénovania siete v závislosti na epoche.

Na trénovanie siete sa vo všetkých prípadoch použili rovnaké dáta. Bolo použitých približne 6000 obrázkov rozmeru 480×360 pixlov rozdelených na trénovaciu a testovaciu množinu. Všetky návrhy siete sú obsiahnuté v prílohe B.

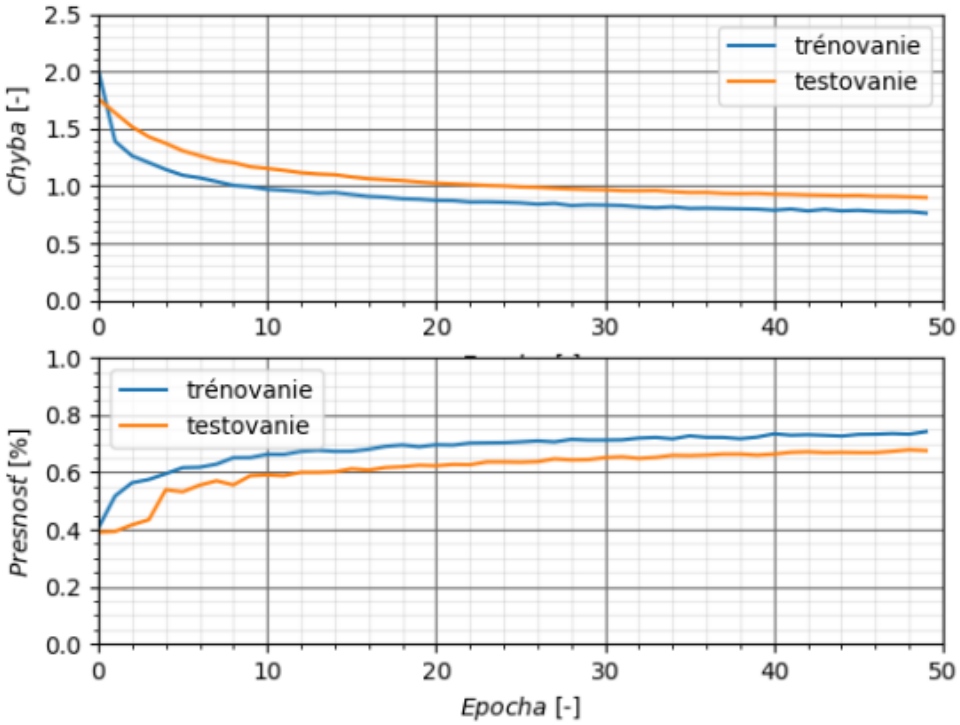
Boli testované siete Net1 až Net3. Priebehy chyby a úspešnosti sietí v zácislosti na epoche po natrénovaní, sú zobrazené na obr. č. 9.7 - 9.9.



Obr. 9.7: Grafy zobrazujúce priebeh chyby a úspešnosť v závislosti na epoche v sieti Net1



Obr. 9.8: Grafy zobrazujúce priebeh chyby a úspešnosť v závislosti na epoche v sieti Net2



Obr. 9.9: Grafy zobrazujúce priebeh chyby a úspešnosť v závislosti na epoche v sieti Net3

	Trénovanie		Testovanie	
	chyba [-]	*presnosť* [%]	*chyba* [-]	*presnosť* [%]
Net1	0,36	88,87	0,46	86,37
Net2	0,26	94,64	0,36	89,71
Net3	0,73	77,93	0,91	70,12

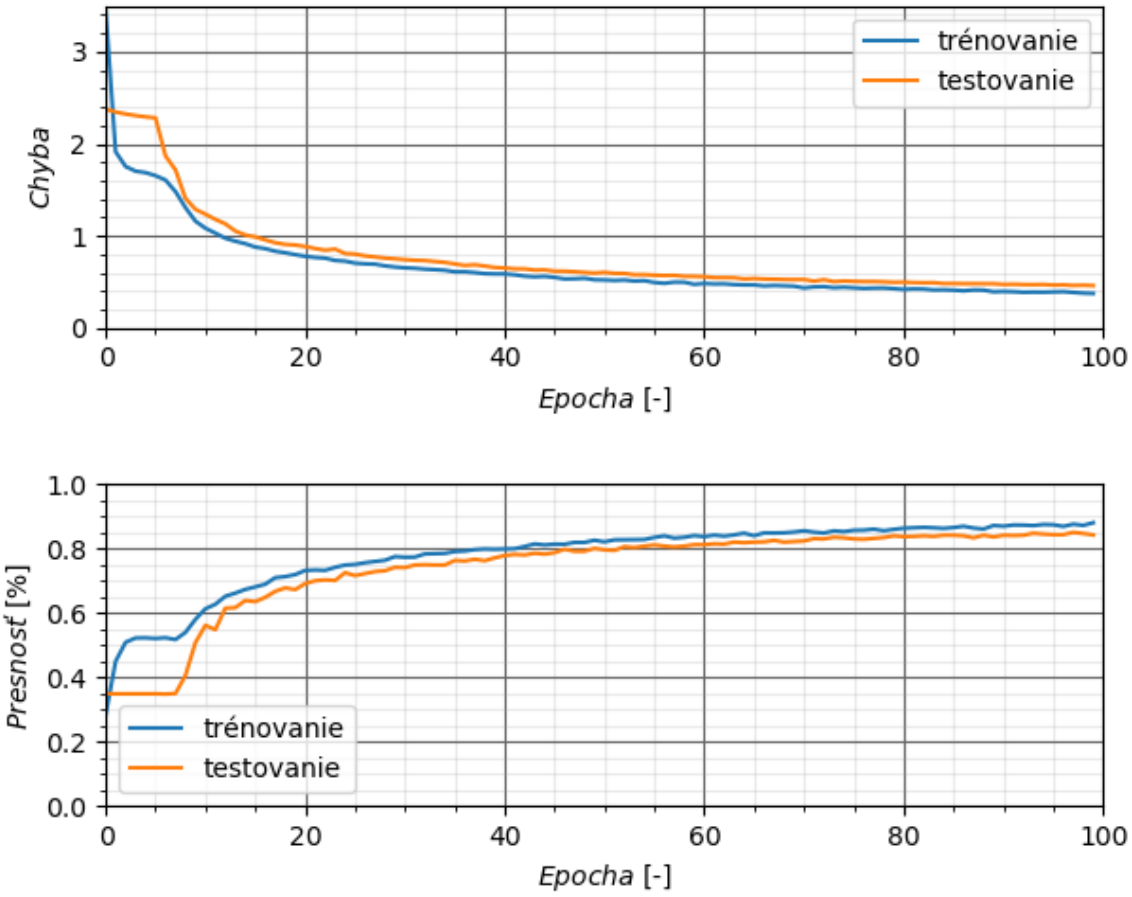
Tabuľka 9.3: Dosiahnuté výsledky sietí po tréningu

Z tabuľky 9.3 vyplýva, že najväčšiu percentuálnu úspešnosť aj najmenšiu chybu dosiahla sieť Net3. Bolo potrebné zvoliť najlepšiu z nich. Predpokladalo by sa, že sa zvolila sieť, ktorá dosahovala najlepšiu percentuálnu úspešnosť. S prihliadnutím na priebehy grafov, bolo zjavné, že sieť už podliehala miernemu overfittingu, preto sa nakoniec zvolila za najlepšiu sieť Net1. Parametre z tejto siete budú použité na overenie percentuálnej úspešnosti siete na novej množine dát.

10 Dosiahnuté výsledky

V prechádzajúcej kapitole bola vybraná sieť, na ktorej sa overí navrhnuté riešenie. Cieľom tejto kapitoly bude overiť úspešnosť natrénovania siete na iných dátach ako boli použité pri tréningu a testovaní. Taktiež sa v tejto časti práce popíšu výhody a nevýhody navrhnutého riešenia, zhodnotí sa časová náročnosť a popíšu sa možné modifikácie prípadne ďalšia možná práca.

V prvom kroku sa sieť trénovala ešte raz. Zvolil sa väčší počet epóch, z dôvodu dosiahnutia nižšej chyby a vyššej presnosti. Výsledné grafy priebehu chyby a presnosti počas trénovania sú zobrazené na obr. č. 10.1.



Obr. 10.1: Grafy zobrazujúce veľkosť chyby a presnosť v závislosti na epoche

Výsledné hodnoty, ktoré dosiahla sieť po natrénovaní sú zaznačené v tabuľke č. 10.1.

	Trénovanie	Testovanie
Dosiahnutá úspešnosť [%]	89,93	87,11
Chyba [-]	0,3752	0,4549

Tabuľka 10.1: Výsledné hodnoty získané po natrénovaní siete

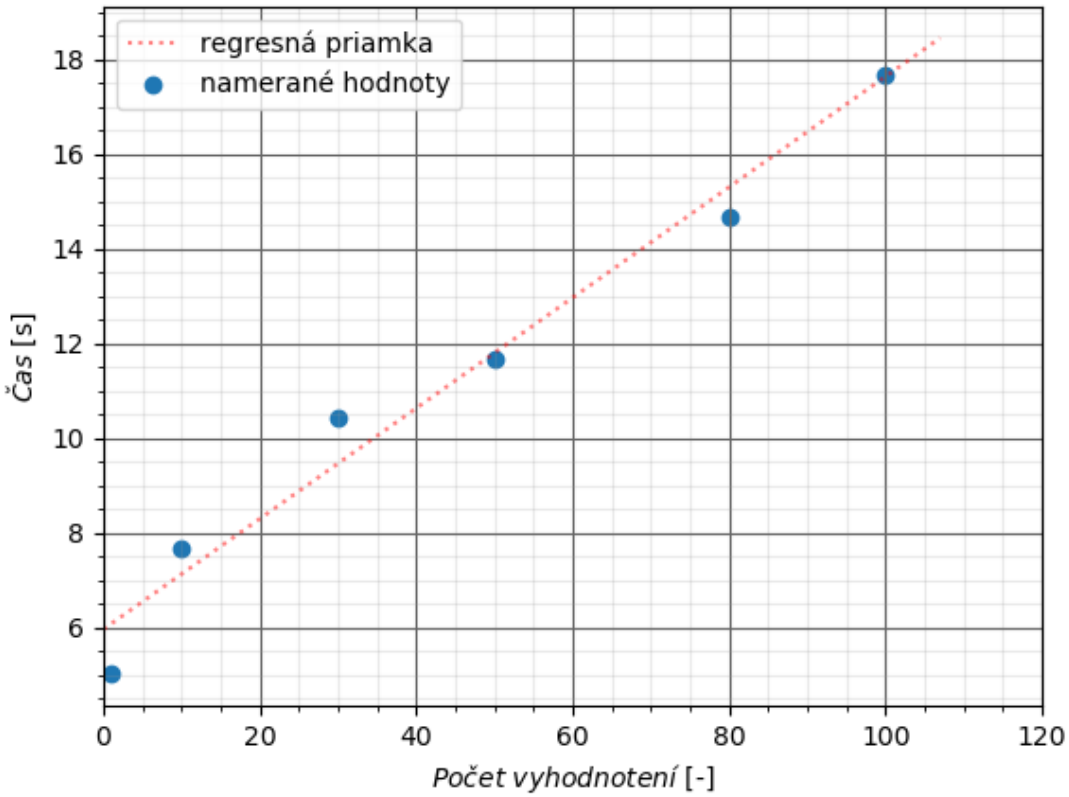
Sieť bola trénovaná s využitím GPU. Približný čas potrebný na natrénovanie siete bol 4500 sekúnd.

Následne sa pristúpilo k overeniu výsledkov na novej množine dát. Pripravilo sa ďalších 1000 snímok, ktoré neboli použité v trénovacej ani testovacej množine. Cieľom je overenie výsledkov získaných z neurónovej siete. Znova je potrebné definovať, aké výsledky sa pokladajú za správne. Sú to výsledky, kde cesta zaniká vo vyznačenom výseku. Úspešnosť na nových testovacích dátach podľa vyššie napísaného kritéria vyjadruje tabuľka č. 10.2.

Celkový počet snímok	Správne detekovaná cesta	Nesprávne detekovaná cesta	Percentuálna úspešnosť
1000	893	107	89,3

Tabuľka 10.2: Výsledky získané na novej testovacej množine

Dosiahnutá percentuálna úspešnosť približne odpovedá výsledkom, ktoré sa získali z výstupu siete, čo sa hodnotí ako pozitívne. Sieť dokáže nájsť cestu nie len na rôznom type povrchu a pri rôznych svetelných podmienkach, ale aj na snímkach na ktorých sa nachádzajú ľudia. Ďalšou výhodou je jej časová náročnosť. Na vyhodnotenie jednej snímky bolo potrebných približne 110 ms, čo vyplýva aj z obr. č. 10.2.



Obr. 10.2: Graf vyjadrujúci časovú závislosť na počte spracovaných snímok

Z obr. č. 10.2 je možné vidieť, že prvé vyhodnotenie je časovo náročné. Trvá približne 4 sekundy. Je to pravdepodobne z dôvodu inicializácie výpočtu.

Veľkou nevýhodu je nedostatok trénovacích dát v okrajových oblastiach, čo má za následok viac nesprávne vyhodnotených snímok. Pre lepšie výsledky by bolo potrebné

dané kategórie o tento typ dát rozšíriť alebo dané kategórie zlúčiť. Ukážka získaných výsledkov z neurónovej siete je na obr. č. 10.4.

Obr. 10.4: Vyznačená cesta

Ďalšie výsledky sa nachádzajú v prílohe C.

Záver

Celá práca bola zameraná na nájdenie vhodných algoritmov na detekciu cesty v obraze. V prvom kroku sa postup riešenia rozdelil na dve časti, na použitie klasických metód a neurónových sietí. Do celej práce boli zavedené dva predpoklady a to, že cesta sa nachádza pred pozorovateľom, ktorý sa tiež nachádza na pozorovanej ceste a vstupnými dátami sú pre obe metódy snímky získané z mobilného robota B2.

V prvej časti práce sa venovalo návrhu a následnej realizácií riešenia použitím klasických metód. Keďže použité snímky boli vytvorené vo vonkajšom prostredí, v parku Lužánky, bolo potrebné dané riešenie navrhnúť s ohľadom na charakter prostredia. Po počiatočnej úprave snímok a ich hlbšom preštudovaní sa zvolil upravený rozklad na farebné zložky, ktorý zaistil nezávislosť na prudké zmeny svetla. V ďalších častiach sa teda pracovalo len s výslednou modrou zložkou. Naštudovali sa rôzne metódy spracovania obrazu a následne sa skombinovali tak, aby sa dosiahlo požadované riešenie. Modifikované priemerovanie sa použilo na zmenšenie prípadne odstránenie oblastí, ktoré vznikli pri procese vytvorenia modrej zložky. Následne sa zo segmentačných metód použil Cannyho hranový detektor na vyznačenie hraníc v snímke. Zvyšné nežiadúce čiary po segmentácií sa odstránili prostredníctvom nájdenia uzavretých kriviek v obraze. Na záver sa získané hranice cesty vyznačili v pôvodnom obrázku. Celý postup bol navrhnutý v programovacom jazyku Python s použitím knižníc Numpy a OpenCV.

Na overenie funkčnosti navrhnutej metódy sa algoritmus otestoval na 260 snímkach rozmeru 480×360 pixlov. Medzi výhody navrhnutej metódy určite patrí jej presnosť. Veľkou nevýhodou, vďaka ktorej algoritmus nie je možné použiť v reálnom čase je veľká časová náročnosť. Na skrátenie času bolo navrhnuté optimalizované riešenie, ktoré spočíva v zmenšení vstupných snímok a následnom použití menšieho počtu filtrov. Celkový výpočetný čas sa skrátil, ale nie natoľko, aby bolo možné danú metódu použiť v real-time aplikácií. To sa môže pokladať za dosť veľkú nevýhodu, ale navrhnutý algoritmus je možné využiť aj inak. Keďže sa v poslednej dobe do popredia dostali neurónové siete, ktoré na svoj vstup potrebujú privádzať predpripravené dáta, tak práve na ich prípravu by bolo možné použiť navrhnutý algoritmus.

Druhou časťou práce bolo použitie neurónových sietí na detekciu cesty. V prvom kroku riešenia bolo potrebné zvoliť vhodnú reprezentáciu cesty. Vybralo sa rozloženie obrazu na rovnomerné oblasti podľa uhlu, pričom sa za jeho vrchol pokladal bod, ktorý je najbližšie skutočnej polohe robota, teda stred spodnej hrany. Detekcia cesty podľa zvolenej stratégie viedla na klasifikačnú úlohu. Pre ďalší postup bolo potrebné snímky priradiť do jednotlivých kategórií a následne ich rozdeliť na trénovaciu a testovaciu množinu. Pre rozšírenie vstupných dát sa použili viaceré úpravy (Gaussov filter, priblíženie a horizontálne prevrátenie). Následne sa prešlo k návrhu štruktúry siete. Najskôr sa určila tzv. štartovacia sieť. Slúžila na zistenie toho, ktoré parametre bude potrebné pridať, pre dosiahnutie čo najlepších výsledkov chyby a presnosti po ukončení tréningu. Následne sa sieť optimalizovala pridaním konvolučných a poolingových vrstiev. Taktiež pridaním parametrov a aj úpravou už existujúcich. Vybrala sa sieť, pri ktorej priebeh chyby a presnosti pri trénovaní a testovaní mal najmenší rozdiel. Pre dosiahnutie lepších výsledkov sa použila na jej trénovanie väčšia trénovacia množina, ktorá obsahovala približne 10000 snímok. Po natrénovaní sa overili získané výsledky na novej množine o veľkosti 1000 snímok. Dosiahnutá percentuálna úspešnosť bola obdobná v porovnaní s výsledkami získanými zo siete.

Medzi výhody použitej metódy patrí jej rýchlosť vyhodnotenia. Nevýhodou tejto metódy je potreba veľkého množstva vstupných dát, ktoré je nutné klasifikovať. Tento krok je potrebné spraviť „ručne“. Pre predstavu, za hodinu bolo možné s použitím prezentovaného GUI kategorizovať približne 800 snímok. Ďalšou nevýhodou je čas potrebný na natrénovanie siete, ktorý sa s narastajúcim počtom dát bude len zväčšovať. Taktiež nutnosť použitia GPU, z dôvodu časovej náročnosti trénovania siete.

Literatúra

[1] 7 Types of Activation Functions in Neural Networks: How to Choose? MissingLink [online]. [cit. 2020-05-25]. Dostupné z: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/?fbclid=IwAR2FCf0J9zpSNd4ReL7aOZrfDBUzocZd4MVksoYr5LeuVV1PqsU7J-YUXjs

[2] ALOM, M. Z., T. M. TAHA, C. YAKOPCIC, et al. A State-of-the-Art Survey on Deep Learning Theory and Architectures. Electronics [online]. 2019, 8(3) [cit. 2020-05-25]. ISSN 2079-9292. Dostupné z: https://www.mdpi.com/2079-9292/8/3/292

[3] AMIDI, A. a S. AMIDI. Recurrent Neural Networks Cheatsheet [online]. [cit. 2020-05-25]. Dostupné z: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks?fbclid=IwAR1F6rXKCgSjrUQ1a0QVuz-FhO-nHrnusa8-LdZgMQD1eakxHztTfnDjR18#architecture

[4] Artificial Neural Network - Basic Concepts. Tutorials Point [online]. [cit. 2020-06-25]. Dostupné z: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm?fbclid=IwAR2VmwUg3t8lYsXPC9JUaz0j8J84uynZwDaGjmXA720-0SI-CWl1CDrcmlA

[5] BOYAT, A. K. a B. K. JOSHI. A Review Paper: Noise Models in Digital Image Processing. Signal & Image Processing: An International Journal [online]. 2015, 6(2), 63-75 [cit. 2020-05-25]. ISSN 22293922. Dostupné z: http://www.aircconline.com/sipij/V6N2/6215sipij06.pdf

[6] CHAO, W. Gabor wavelet transform and its application [online]. 2010, 1-16 [cit. 2020-03-26]. Dostupné z: https://www.semanticscholar.org/paper/Gabor-wavelet-transform-and-its-application-Chao/cda4fb9df653b5721ad4fe8b4a88468a410e55ec

[7] CRISTÓFORIS, P., M. A. NITSCHE, T. KRAJNÍK a M. MEJAIL. Real-time monocular image-based path detection. Journal of Real-Time Image Processing [online]. 2016, 11(2), 335-348 [cit. 2020-03-26]. DOI: 10.1007/s11554-013-0356-z. ISSN 1861-8200. Dostupné z: http://link.springer.com/10.1007/s11554-013-0356-z

[8] GOODFELLOW, I., Y. BENGIO a A. COURVILLE. Deep Learning. MIT Press, 2016. ISBN 978-0262035613.

[9] GONZALEZ, R. C. a R. E. WOODS. Digital image processing. 3rd ed. Upper Saddle River: Pearson, c2008. ISBN 978-0131687288.

[10] GRAVES, A. a N. JAITLY. Towards End-to-End Speech Recognition with Recurrent Neural Networks. 31st International Conference on Machine Learning. 2014, 1764-1772.

[11] GREGOR, M. Detektory objektů v obraze a jejich realizace. Brno, 2008. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunika ních technologií. Vedoucí práce Ing. František Kyselý.

[12] HALLIDAY, D., R. RESNICK a J. WALKER. Fyzika 2. Druhé. Brno: VUTIUM, 2013. ISBN 978-80-214-4123-1.

[13] HLAVÁČ, V. Hledání hran a hranových bodů. Praha. Dostupné také z: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/22EdgeDetectionCz.pdf?fbclid=IwAR00JeChzmO1aLt2W68802YXFdTZFf9wFN8j-_c-b6VKZgWZZsu5-6YlJM4

[14] HLAVÁČ, V. a M. SEDLÁČEK. Zpracování signálů a obrazů. Praha: Vydavatelství ČVUT, 2000. ISBN 80-01-02114-9.

[15] HUANG, J., B. KONG, B. LI a F. ZHENG. A New Method of Unstructured Road Detection Based on HSV Color Space and Road Features. In: 2007 International Conference on Information Acquisition [online]. IEEE, 2007, s. 596-601 [cit. 2020-03-26]. DOI: 10.1109/ICIA.2007.4295802. ISBN 1-4244-1219-6. Dostupné z: http://ieeexplore.ieee.org/document/4295802/

[16] JUNG, C. R. a C. R. KELBER. An Improved Linear-Parabolic Model for Lane Following and Curve Detection. In: XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05) [online]. IEEE, 2005, s. 131-138 [cit. 2020-03-26]. DOI: 10.1109/SIBGRAPI.2005.10. ISBN 0-7695-2389-7. Dostupné z: http://ieeexplore.ieee.org/document/1599093/

[17] Keras [online]. [cit. 2020-06-20]. Dostupné z: https://keras.io/

[18] KHALIFA, O. O., I. M. KHAN, A. A. M. ASSIDIQ, A. ABDULLA a S. KHAN. A Hyperbola-Pair Based Lane Detection System for Vehicle Guidance [online]. 2010, 4 [cit. 2020-03-26]. Dostupné z: https://www.researchgate.net/publication/49586742_A_Hyperbola-Pair_Based_Lane_Detection_System_for_Vehicle_Guidance

[19] KONG, H., J. AUDIBERT a J. PONCE. Vanishing point detection for road detection. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition [online]. IEEE, 2009, s. 96-103 [cit. 2020-03-26]. DOI: 10.1109/CVPR.2009.5206787. ISBN 978-1-4244-3992-8. Dostupné z: https://ieeexplore.ieee.org/document/5206787/

[20] LAI, A.H.S. a N.H.C. YUNG. Lane detection by orientation and length discrimination. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) [online]. 2000, 30(4), 539-548 [cit. 2020-03-29]. DOI: 10.1109/3477.865171. ISSN 10834419. Dostupné z: http://ieeexplore.ieee.org/document/865171/

[21] Layer activation functions. Keras: the Python deep learning API [online]. [cit. 2020-06-20]. Dostupné z: https://keras.io/api/layers/activations/?fbclid=IwAR3AofPoe_cvvvk33JhlC7Vx-SUstvF_wrSfZRXIcAqmoZFC9ONLvqmOV_8

[22] MALIK, U. A., S. U. AHMED a F. KUNWAR. A self-organizing neural scheme for road detection in varied environments. In: The 2011 International Joint Conference on Neural Networks [online]. IEEE, 2011, s. 3049-3054 [cit. 2020-05-26]. ISBN 978-1-4244-9635-8. Dostupné z: http://ieeexplore.ieee.org/document/6033623/

[23] matplotlib [online]. [cit. 2020-06-20]. Dostupné z: https://matplotlib.org/

[24] MURÁŇ, J. Úvod do konvolučných neurónových sietí. Umelá Inteligencia [online]. 28. novembra 2019 [cit. 2020-06-20]. Dostupné z: https://umelainteligencia.sk/uvod-do-konvolucnych-neuronovych-sieti/?fbclid=IwAR10JAuVJL2lpXYpmjfZ0UgJmMNBS2IQd6SlktEs5JyLXdhRIJxAFaVZ_K0

[25] NIELSEN, M. Neural Networks and Deep Learning [online]. Determination Press, 2015 [cit. 2020-05-25]. Dostupné z: http://neuralnetworksanddeeplearning.com/

[26] NumPy [online]. c2019–2020 [cit. 2020-04-03]. Dostupné z: https://numpy.org/

[27] OpenCV [online]. c2020 [cit. 2020-04-03]. Dostupné z: https://opencv.org/

[28] PETKOV, N. a M. B. WIELING. Gabor filter for image processing and computer vision [online]. 2008-07-03 [cit. 2020-03-26]. Dostupné z: http://matlabserver.cs.rug.nl/edgedetectionweb/web/edgedetection_params.html

[29] python [online]. c2020 [cit. 2020-04-03]. Dostupné z: https://www.python.org/

[30] REBROVÁ, K. Vnímanie a pomenovávanie farieb a farebných kategórií. Bratislava, 2007. Bakalárska práca. Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky. Vedúcí práce RNDr. Martin Takáč.

[31] ROJAS, R. Neural Networks: A Systematic Introduction. Berlin: Springer, 1996. ISBN 978-3540605058.

[32] ROSEBROCK, A. Deep Learning for Computer Vision with Python: Starter Bundle. PyImageSearch, 2017. ISBN 1032261119.

[33] ROSEBROCK, A. Keras Conv2D and Convolutional Layers. PyImageSearch [online]. December 31, 2018 [cit. 2020-06-20]. Dostupné z: https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/?fbclid=IwAR2fyn4mVQtK5zUwKtIVns1XxO7sizS7u3gjtRWI-dFvxCl5QVlm56kvP78

[34] SAHA, S. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Towardsdatascience.com [online]. 15 December 2018 [cit. 2020-06-20]. Dostupnéz:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[35] SHAO, J., W. FOERSTNER, H. EBNER, Ch. HEIPKE a K. EDER. Gabor wavelets for texture edge extraction [online]. 1994-8-17, s. 745-752 [cit. 2020-03-26]. DOI: 10.1117/12.182839. Dostupné z: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=982631

[36] SHI, X., B. KONG a F. ZHENG. A New Lane Detection Method Based on Feature Pattern. In: 2009 2nd International Congress on Image and Signal Processing [online]. IEEE, 2009, s. 1-5 [cit. 2020-03-25]. DOI: 10.1109/CISP.2009.5304294. ISBN 978-1-4244-4129-7. Dostupné z: http://ieeexplore.ieee.org/document/5304294/

[37] SLÁVKA, M. Playing Gomoku with Neural Networks. Brno, 2019. Bachelor's Thesis. Brno University of Technology, Faculty of Information Technology. 2019-06-11. Supervised by Hradiš Michal. Dostupné z: https://www.fit.vut.cz/study/thesis/21764/

[38] SONKA, M., V. HLAVAC a R. BOYLE. Image processing, analysis, and machine vision. 4th. United States of America: Cengage Learning, 2015. ISBN ISBN : 978-1-133-59369-0.

[39] SUDHAKARAN, S. What is Color Space and the Tristimulus Values XYZ? Wolfcrow [online]. [cit. 2020-05-25]. Dostupné z: https://wolfcrow.com/what-is-color-space-and-the-tristimulus-values-xyz/?fbclid=IwAR2zmuMOmJffhO-GtInHkLGeZinzqkxQpgVW_mqLBcdb5aX1zcpuTBe4QJY

[40] SUN, T., S. TANG, J. WANG a W. ZHANG. A robust lane detection method for autonomous car-like robot. In: 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP) [online]. IEEE, 2013, s. 373-378 [cit. 2020-03-26]. DOI: 10.1109/ICICIP.2013.6568100. ISBN 978-1-4673-6249-8. Dostupné z: http://ieeexplore.ieee.org/document/6568100/

[41] SURMA, G. Street Lanes Finder-Detecting Street Lanes for Self-Driving Cars. Medium [online]. 2019 [cit. 2020-03-26]. Dostupné z: https://towardsdatascience.com/street-lanes-finder-detecting-street-lanes-for-self-driving-cars-fe069ec5a22d

[42] SZELISKI, R. Computer Vision: Algorithms and Applications. London: Springer, 2010. Texts in computer science. ISBN 9781848829343.

[43] TensorFlow [online]. [cit. 2020-06-20]. Dostupné z: https://www.tensorflow.org/

[44] THRUN, S. Learning To Play the Game of Chess. Advances in Neural Information Processing Systems 7. The MIT Press, 1995, s. 1069-1076. ISBN 9780262201049.

[45] WANG, Y., E. K. TEOH a D. SHEN. Lane detection and tracking using B-Snake. Image and Vision Computing [online]. 2004, 22(4), 269-280 [cit. 2020-05-30]. DOI: 10.1016/j.imavis.2003.10.003. ISSN 02628856. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0262885603002105

[46] WANG, Y., D. SHEN a E. K. TEOH. Lane detection using spline model. Pattern Recognition Letters [online]. 2000, 21(8), 677-689 [cit. 2020-03-26]. DOI: 10.1016/S0167-8655(00)00021-0. ISSN 01678655. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0167865500000210

[47] What is the difference between CCD and CMOS image sensors in a digital camera? HowStuffWorks.com. [online]. 1 April 2000 [cit. 2020-04-02]. Dostupné z: https://electronics.howstuffworks.com/cameras-photography/digital/question362.htm

[48] WALLACE, S. a K. WUERCH. Labeling Paths with Convolutional Neural Networks. 2019. [online]. Dostupné z: https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1328&context=cpesp

[49] ZU, Z., Y. HOU, D. CUI a J. XUE. Real-time road detection with image texture analysis-based vanishing point estimation. In: 2015 IEEE International Conference on Progress in Informatics and Computing (PIC) [online]. IEEE, 2015, s. 454-457 [cit. 2020-03-26]. DOI: 10.1109/PIC.2015.7489888. ISBN 978-1-4673-8086-7. Dostupné z: http://ieeexplore.ieee.org/document/7489888/

A Ďalšie výsledky riešenia pri použití klasických metód






B Navrhnuté štruktúry sietí

Typ vrstvy	Počet filtrov	Veľkosť jadra	Výstupný tvar	Počet parametrov
Konvolučná	16	5x5	90x120x32	1472
Poolingová	-	2x2	45x60x32	0
Konvolučná	32	3x3	45x60x32	9248
Poolingová	-	2x2	22x30x32	0
Vyrovnávacia	-	-	21120x1	0
Plne prepojená	-	-	700x1	14784700
Dropout (0,4)	-	-	700x1	0
Plne prepojená	-	-	60x1	42060
Dropout (0,4)	-	-	60x1	0
Výstupná	-	-	11x1	671
$\sum$				14,838,151

Tabuľka B.1: Navrhnutá štruktúra siete Net1

Typ vrstvy	Počet filtrov	Veľkosť jadra	Výstupný tvar	Počet parametrov
Konvolučná	16	5x5	90x120x16	1216
Poolingová	-	2x2	45x60x32	0
Konvolučná	32	3x3	45x60x32	4640
Poolingová	-	2x2	22x30x32	0
Konvolučná	32	3x3	22x30x32	9248
Poolingová	-	2x2	11x15x32	0
Konvolučná	64	3x3	11x15x64	18496
Poolingová	-	2x2	5x7x64	0
Vyrovnávacia	-	-	2240x1	0
Plne prepojená	-	-	700x1	1568700
Dropout (0,4)	-	-	700x1	0
Plne prepojená	-	-	60x1	42060
Dropout (0,4)	-	-	60x1	0
Výstupná	-	-	11x1	671
$\sum$				1,645,031

Tabuľka B.2: Navrhnutá štruktúra siete Net2

Typ vrstvy	Počet filtrov	Veľkosť jadra	Výstupný tvar	Počet parametrov
Konvolučná	16	5x5	90x120x16	1216
Poolingová	-	2x2	45x60x16	0
Konvolučná	32	3x3	45x60x32	4640
Poolingová	-	2x2	22x30x32	0
Konvolučná	64	3x3	22x30x64	18496
Poolingová	-	2x2	11x15x64	0
Vyrovnávacia	-	-	10560x1	0
Plne prepojená	-	-	480x1	5069280
Dropout (0,4)	-	-	480x1	0
Plne prepojená	-	-	60x1	28860
Dropout (0,4)	-	-	60x1	0
Výstupná	-	-	11x1	671
$\sum$				5,123,163

Tabuľka B.3: Navrhnutá štruktúra siete Net3

C Ďalšie výsledky riešenia pri použití neurónových sietí