VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Jaroslav Peška

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

DETEKCE CESTY POMOCÍ DAT Z KAMERY POHYBLIVÉHO ROBOTU
ROAD DETECTION USING DATA FROM MOBILE ROBOT CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jaroslav Peška**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Petyovský, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Jaroslav Peška ***ID:*** 186160
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Detekce cesty pomocí dat z kamery pohyblivého robotu

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je pro existující mobilní robotickou platformu navrhnout vhodné algoritmy realizující detekci cesty pomocí dat ze senzorů robota (kamera, GPS, senzory vzdálenosti).

1. Prostudujte problematiku a aplikace rozpoznávání cesty pomocí dat z kamery pohyblivého robotu. Proveďte rešerši současných přístupů k automatizované detekci cesty z obrazových dat.
2. Definujte limitní parametry úlohy pro detekci cesty (výpočetní výkon, rychlost pohybu, parametry senzorů obrazu).
3. Navrhněte a vytvořte sadu testovacích dat, vytvořte aplikaci pro zpracování testovacích dat.
4. Navrhněte a realizujte algoritmus pro rozpoznávání cesty na základě dat z kamery pro definovaný HW.
5. Vyhodnoťte úspěšnost a časovou efektivitu algoritmu rozpoznání cesty na testovacích datech.
6. Nalezněte paralelizovatelné oblasti algoritmu a zvolte vhodné prostředky pro akceleraci výpočtů. Následně implementujte tyto prostředky do algoritmu pro detekci.
7. V reálných podmínkách vytvořte novou, rozšířenou sadu testovacích dat. Určete výslednou přesnost navržených algoritmů na rozšířených testovacích datech.
8. Zhodnoťte dosažené výsledky a navrhněte další možná řešení vedoucí k lepším výsledkům rozpoznávání cesty.

DOPORUČENÁ LITERATURA:

[1] Sonka,M. , Hlavac, V. Boyle, R.: Image Processing, Analysis, and Machine Vision, 3rd Edition, Thomson 2007, ISBN 049508252X

[2] Everett, H.,R.: Sensors for Mobile Robots theory and application, CRC Press 1995, ISBN 1568810482

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Petr Petyovský, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá řešením problému detekce cesty mobilních robotů využitím dat z kamery. V první části je provedena rešerše současných přístupů a zhodnocení možnosti jejich uplatnění v navrženém výsledku. Následně jsou definovány limitní parametry aplikace. Pro výsledné řešení byl definován proces automatického srovnávání přesnosti výsledků s využitím člověkem definovaného etalonu a naměřeny dvě sady testovacích dat. Byla implementována první verze algoritmu, která byla následně optimalizována a akcelerována za pomoci GPGPU. Navržený algoritmus je nakonec vyhodnocen a jsou naznačeny další možná vylepšení.

KLÍČOVÁ SLOVA

detekce cesty, mobilní robot, zpracování obrazu, sledování cesty, detekce překážek, neuronová síť, strojové učení

ABSTRACT

Semestral thesis deals with the problem of road detection by mobile robots using data obtained from its camera. First, current solutions are researched and considered for use in the proposed algorithm. Afterwards we define limit parameters of the entire solution. An automatic process using human-created reference was then devised to programatically determine the accuracy of individual versions of proposed solutions. First, an initial version of the solution was implemented, which was subsequently optimized and accelerated using GPGPU. Lastly, proposed algorithm is evaluated and possible future changes are outlined.

KEYWORDS

road detection, mobile robot, image processing, path tracking, obstacle detection, neural network, machine learning

PEŠKA, Jaroslav. *Detekce cesty pomocí dat z kamery pohyblivého robotu*. Brno, 2018, 59 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Petr Petyovský, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Detekce cesty pomocí dat z kamery pohyblivého robotu“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petru Petyovskému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné připomínky k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V současné době jsou čím dál tím více kladeny požadavky na bezpečnost, případně automatizaci dopravy. Jedním z velkých doposud ne zcela vyřešených problémů je detekce cesty, po které by se mělo vozidlo pohybovat. Ačkoliv výzkum v této oblasti probíhá již několik desetiletí, teprve nyní se algoritmy dostávají do stavu, kdy je možné je nasadit do provozu v osobních automobilech. V budoucnosti budou pilířem autonomního řízení, kdy člověk nebude muset sledovat cestu.

Vývoj takovýchto algoritmů neprobíhá pouze ve velkých společnostech a na univerzitách, ale je předmětem i mnoha soutěží. Tato práce se proto bude zabývat vývojem takového algoritmu, který bude schopný obstát na soutěži autonomních robotů Robotour, kde je hlavním úkolem dojet na místo určení, aniž by robot sjel z cesty. Nebude však bráno jako chyba pokud vzniklé řešení bude možné adaptovat do osobních či jiných vozidel.

Před samotným návrhem algoritmu detekce cesty jsou projity současné přístupy k řešení problému a vyhodnoceny, zda by nemohly být uplatněny i v této bakalářské práci. Jsou také definovány limitní parametry, ve kterých musí navržený algoritmus korektně a spolehlivě pracovat. Tím je myšlen cílový podvozek, jaká je dostupná kamera, kolik výpočetního výkonu je k dispozici a jaké jsou požadavky na reakční dobu systému.

Pro usnadnění vývoje a srovnávání verzí algoritmu je navrženo měřítko, podle kterého jsou jednotlivé verze vyhodnocovány. K tomu je zajištěn etalon, se kterým budou výstupy srovnávány – člověkem označené snímky cesty.

Prvně je navržena základní verze algoritmu, která je funkční, ale není příliš robustní a nevyužívá dostupné prostředky optimálně. Tyto nedostatky jsou následně odstraněny zkoumáním způsobů, kterými by bylo možné algoritmus zrychlit. Po zrychlení využitím GPGPU akcelerace je nově vzniklá výkonová rezerva využita pro další vylepšení algoritmu.

Nakonec je vyhodnocena celková přesnost algoritmu na všech testovacích datech a zmíněny další kroky, kterými by se vývoj mohl ubírat.

1 Problematika a současná řešení

V této kapitole je prvně definován problém definice cesty, využití detekce a následně provedena rešerše historických i současných přístupů k tomuto problému.

1.1 Definice problému

Pojmem detekce cesty se myslí rozpoznávání cesty, případně předem určené značky označující bezpečný koridor, ve kterém je možné se pohybovat. V ideálním případě by algoritmus měl být sám schopen rozpoznat i nestrukturovanou cestu (sešlapaná cesta), pro účely této práce však postačuje detekce „typické“ cesty - štěrková cesta v parku, vozovka, zámková dlažba.

1.2 Využití detekce cesty

Detekce cesty má v současné době využití zejména v dopravě, ať už zboží, případně i osob. Detekce cesty je kritická pro pohyb autonomních robotů v předem neznámém či měnícím se prostředí. Momentálně je využívána především v průmyslu, ale v budoucnosti se pravděpodobně hlavním uplatněním stane navigace autonomních vozidel.

1.3 Současné přístupy

V současnosti existuje více přístupů k detekci cesty, kdy každý má svoje výhody a nevýhody. V této podkapitole jsou některé minulé i současné způsoby detekce cesty.

1.3.1 Sledování předdefinované trati

Detekce čáry patří mezi základní a nejjednodušší způsoby detekce cesty. Cesta je předem vytyčena, a robotu postačuje pouze sledovat značky, které ji označují.

Sledování drátu

Princip navigace sledováním čáry spočívá v detekci předem známé značky na sledovaném povrchu. Pro řešení založená na jiném způsobu detekce než jsou obrazová data je možné použít vodicí drát zapuštěný do povrchu, na kterém se mobilní robot pohybuje. Pro samotnou detekci drátu zapuštěného v podložce se využívá vysílání

elektromagnetického signálu a jeho detekce navigovaným robotem. Pro základní navigaci postačuje aby drátem protékal elektrický proud, poté je pro navigaci využíváno vznikající magnetické pole okolo vodiče. Případně může drát sloužit jako anténa a vysílat rádiový signál, což umožňuje robotu předávat další informace, například na jaké trati se aktuálně nachází.

Mezi výhody patří odolnost, protože nespoléhá na čistotu detekovaného povrchu. Hlavní nevýhodou je náročnost instalace, kdy vodicí drát musí být do podkladu vestavěn již při stavbě povrchu, případně jej lze zabudovat později, avšak za mnohonásobně vyšších nákladů. Hlavní využití je v průmyslu, zejména ve skladech a výrobních prostorech, kde je využívána dopravními roboty. [2]

Navigace pomocí neviditelných značek nicméně není proveditelná využitím dat z kamery, tudíž není pro účely této práce vhodná.

Sledování čáry

Pokud není možné z jakéhokoliv důvodu do podložky mechanicky zasahovat aby byl položen vodicí drát, je možné využívat optické značky nanesené na povrchu podložky. Hlavní výhodou sledování optických značek je možnost snadné změny trasy, ve srovnání se zabudovaným vodicím drátem. Existuje více řešení, většina využívá vysokého kontrastu (bílá na černé, černá na bílé) značek (většinou spojité čáry). Značky bývají vytvořeny nanesením retroreflexní barvy či pásky. Čára však nemusí mít nutně jinou viditelnou barvu než okolní prostředí, kritickým parametrem je možnost rozlišit ji od okolí. To je možné zajistit například jinou povrchovou úpravou sledované plochy, aby čára měnila určitý parametr, který je možné poté detekovat senzory (drsnost povrchu, kapacita, teplo).

Přítomnost viditelných značek na trase je možné považovat i jako bezpečnostní prvek varující okolní osoby o trase robotů.

Snímání samotné čáry je poté prováděno polem snímačů směřujících na snímaný povrch. Z rozdílů jednotlivých naměřených hodnot je poté vypočtena poloha na čáře. [2] Se zvýšenou dostupností výpočetního výkonu je v současnosti pro snímání viditelné čáry možné využít kamery namontované na robotu. Výhodou navigace po čáře pomocí kamery je zejména možnost vidět dopředu více než by umožnilo konvenční snímání povrchu. Znalost budoucí trasy umožňuje zvýšit rychlost pohybu.

Mezi výhody patří dříve zmíněná možnost změny trasy a odolnost proti přerušení čáry, kdy přerušení značení nevyřadí zbytek trasy z provozu, jak by se stalo v případě vodicího drátu.

Obr. 1.1: Ukázka line follower robotu z průmyslu [21]

1.3.2 Detekce neznámé cesty

Pro aplikaci v obecných podmínkách je nutné využít algoritmu nevyžadujícího přítomnost předem definovaných značek označujících cestu. Základním požadavkem na algoritmus detekce cesty je segmentace dat na alespoň dvě základní množiny - cesta, ostatní. Tímto problémem se již v minulosti zabývalo mnoho odborných prací a v praxi je využito několika různých přístupů. Následující přístupy byly vybrány jako základní, ze kterých bude možné při návrhu vlastního algoritmu vycházet nebo se jimi alespoň inspirovat.

Obr. 1.2: Ukázka cesty bez předdefinovaných značek

Detekce jízdního pruhu

Algoritmy pro detekci jízdního pruhu využívají horizontální značení na povrchu vozovky pro určení dráhy jízdního pruhu. V současnosti jsou využívány zejména v automobilovém průmyslu pro asistenční systémy držení se v jízdním pruhu. Nutná podmínka funkce pro normální případy je přítomnost horizontálního značení, ale v určitých situacích se lze obejít bez něj (jízda v koridoru ohraničeném překážkami). Ve většině systémů nasazených v osobních automobilech je využíváno pouze rozlišování čar, nikoliv překážek (dopravní kužely, betonové zátarasy).

Implementace systému detekce jízdního pruhu sledujícího čáry na vozovce je popsána v [14]. Obraz z kamery je rozdělen na popředí a pozadí. Popředí je následně převedeno do černobílého barevného prostoru pro zjednodušení výpočtů. Pro potlačení šumu je použit Gaussovský filtr. Na vyfiltrovaném obrazu popředí je aplikován algoritmus detekce hran pro další snížení množství dat k výpočtu. Využitím Houghovy transformace jsou nalezeny přímky reprezentující hranice jízdních pruhů. Detekce opouštění jízdního pruhu reaguje na překročení hraniční šířky pruhu v dané vzdálenosti. Pro jízdu rovnoběžně s pruhem je šířka menší než při vybočování. [14]

Podmínka přítomnosti značení jízdních pruhů omezuje algoritmus pro použití na udržovaných komunikacích. Na silnicích s poškozeným, případně nepřítomným, značením není možné tento systém využívat.

Úběžník cesty

Výstupem metody hledání úběžníku cesty je bod kam směřuje cesta. Metoda je založena na úvaze, že dvě paralelní přímky ležící v rovině ne kolmé ke kameře se budou v určitém bodě konvergovat. Tento bod je v anglické literatuře pojmenován jako **vanishing point**. Pro uměle vytvořené cesty je většinou možné využít „kaskádované“ Houghovy transformace. Tento způsob však většinou selhává v případě nestrukturované cesty. [18]

Jednou možnou implementací je například [13]. Zmíněná metoda využívá hlasování o směru textury pro každý pixel. Každému pixelu také přiřadí jistotu, s jakou byl směr určen. Oblasti s nízkou jistotou jsou vyřazeny. Protože metoda nevyužívá barevné složky obrazu, řeší dobře i změny osvětlení. [13]

Neuronové sítě

S rostoucím výpočetním výkonem bylo umožněno detekovat cesty i netradičními způsoby. Neuronové sítě jsou systémy inspirované reálným světem a místo předem zadaných pevných požadavků a pokynů, kterých se musí držet, fungují na principu učení. Síť se předem natrénuje pomocí člověkem vytvořených požadovaných výsledků a následně je schopna sama zpracovávat vstup tak, jak byla natrénována. Vzhledem

k tomu, že dostatečný výpočetní výkon je pro praktické nasazení dostupný pouze několik let, existuje ohledně tohoto přístupu méně článků než k ostatním metodám. Tato metoda má však velký potenciál, jak je naznačeno v [17].

Nevýhodou neuronových sítí pro detekci cesty je jejich vlastnost, kdy programátor nemůže zjistit, co vlastně síť detekuje. Výhodou naopak je v současnosti velký rozvoj celé této oblasti, kdy vzniká nový hardware přímo stavěný pro neuronové sítě, občas dokonce již nasazovaný do provozu.

2 Limitní parametry aplikace

Před započetím návrhu samotného algoritmu je nutné definovat meze a požadavky, ve kterých má finální řešení fungovat. Většina, ne-li všechny, parametry omezující algoritmus plynou z požadavku, že primární cíl celého řešení je nasazení v soutěži Robotour.

2.1 Soutěž Robotour

Jedná se o každoroční soutěž autonomních robotů, jejíž první ročník se konal již v roce 2006. Cílem soutěže je podpořit vývoj robotů schopných autonomně vykonávat funkci dopravy. Soutěž probíhá v obyčejných parcích, tedy na komunikacích bez dopravního značení.

Zásadními pravidly je časový limit jedné hodiny pro vykonání úkolu a plná autonomnost robotu během pokusu. [7] Sjetí robotu z parkové cesty znamená konec pokusu, nejdůležitější tedy bude zajistit přesnost detekce cesty alespoň na takovou úroveň, aby bylo možné případnou chybu detekce v jednom snímku napravit příštími snímky.

Ačkoliv všechna bodovaná kola se zatím konala během dne, v budoucnosti je možné, že se část kol bude konat v noci. Alespoň jedno noční testovací kolo se již konalo po skončení ročníku 2013. [6]

Minimální velikost robotu je omezena povinností snadno umístit a jezdit s plným pětilitrovým soudkem piva. Maximální velikost je omezena možností snadné manipulace, kdy dvě dospělé osoby jej musí být schopné odnést na vzdálenost několika desítek metrů. Robot samozřejmě nesmí být tak velký, aby nebyl schopen se sám pohybovat po cestách v parku.

Detekce cesty může využívat navigační systémy jako jsou GPS, Glonass či Galileo. Definice cest parků pro soutěže jsou dostupné díky projektu OpenStreetMap.[7]

2.2 Popis cílového prostředí

Cílové prostředí je normální park se zpevněnými cestami, které nejsou nijak označeny. Minimální šířka cest je přibližně jeden metr. Na cestách je možné očekávat překážky, cílem tohoto algoritmu však není je spolehlivě rozpoznávat, to je v praxi většinou řešeno dalšími senzory (LIDAR či ultrazvukové snímače vzdálenosti). Cesty mohou být slepé, tudíž není možné předpokládat, že cesta bude vždy někam směřovat.

2.2.1 Povrchy cest

Mezi dominantní povrchy patří jemný štěrk, dlažební kostky, asfalt a zámková dlažba. Povrchy se mohou plynule měnit. Výsledný algoritmus tedy musí být schopný detekovat cestu na všech površích, nemůže spoléhat na monotónnost povrchu cesty.

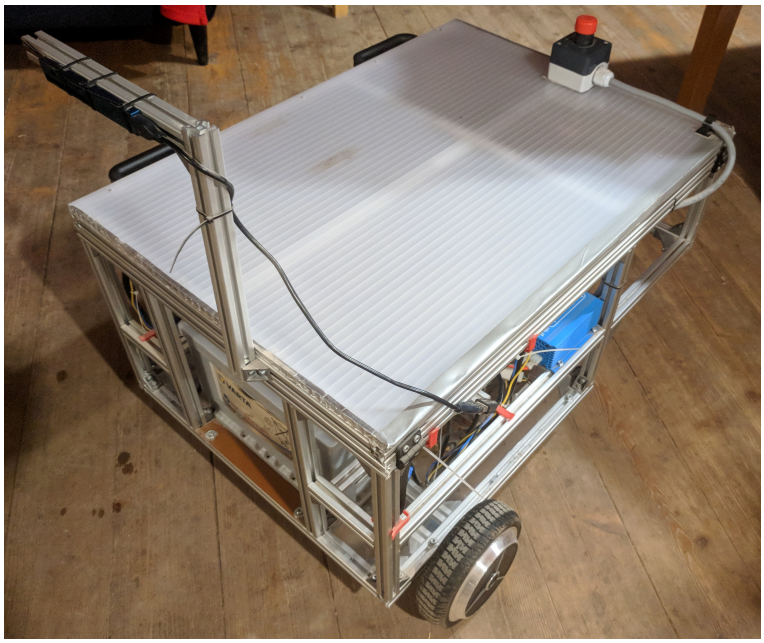
Obr. 2.1: Ukázka přechodu cesty

2.3 Podvozek robotu

Použitý podvozek je již čtvrtou generací vlastních konstrukcí autonomních robotů. Jeho rozměry jsou 50x70x40 cm šířka x délka x hloubka. Konstrukce je sestavena z extrudovaných hliníkových profilů 20x20 mm. Celý podvozek je koncipován jako vývojová platforma pro zkoušení nových přístupů/modulů. Z toho důvodu je jeho obsah modulární – většinu součástí lze snadno vyměnit.

2.3.1 Pohon podvozku

Podvozek je poháněn párem nábojových (hub) motorů o kombinovaném maximálním výkonu 500 W a teoretické maximální rychlosti 12 km/h. Nábojové motory jsou vestavěné přímo do kola a pohání jej přímo – mezi kolem a pohonem tak neexistuje vůle. Pro účely soutěže však bylo v parametrech řídicí jednotky nastaveno omezení rychlosti na 4 km/h. Zatáčení je řešeno diferenciálně za pomoci dvou předních motorů a zadního volně otočného kola. To umožňuje prakticky nulový poloměr otáčení.



Obr. 2.2: Fotografie podvozku robotu

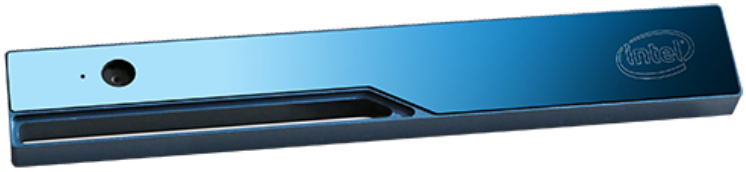
2.3.2 Dostupné senzory

Detekci překážek zajišťuje trojice ultrazvukových senzorů HC-SR04 pokrývající úhel 120° před robotem. Pozice je získávána kombinací magnetického kompasu s kompenzací náklonu a GPS přijímače. Na podvozku se také nachází vícespektrální kamera Intel RealSense R200, která je více rozebrána v podkapitole 2.4. Kamera se nachází ve výšce jednoho metru nad zemí, směřující tak, aby vrchní hrana obrazu snímala cestu ve vzdálenosti přibližně 4 metrů. Tím je zároveň vyřešeno odstraňování horizontu ze zpracovávaného obrazu.

2.4 Kamera

Jako modul kamery byla použita vícespektrální kamera RealSense R200 od společnosti Intel. Jedná se o již nevyráběný vývojový kit, v případě potřeby ale jej lze plně nahradit novějšími modely z řady RealSense. S počítačem komunikuje pomocí rozhraní USB 3. Samotný modul obsahuje tři obrazové snímače, jeden RGB snímač viditelného světla s rozlišením 1920x1080 px a dva snímače infračerveného spektra o rozlišení 640x480 px vybavené pásmovou propustí pro infračervené záření. Infračervené snímače jsou vybaveny globální závěrkou, není tedy pro ně nutné softwarově kompenzovat deformaci způsobenou pohybující se kamerou. Obrazová data jsou dostupná 60x za sekundu, s výjimkou dat z kamery pro viditelné spektrum, která je schopna dodávat rozlišení 1920x1080 px pouze 30x za sekundu. Frekvence

aktualizace dat může být ovlivněna nastavením času expozice.



Obr. 2.3: Vícespektrální kamera Intel Realsense R200 [22]

Výstupy z kamery zahrnují jednotlivé výstupy obrazových snímačů. Kamera také umožňuje získávat syntetizovaný obraz hloubkové mapy, vypočtené pomocí stereo vidění infračervených snímačů. Hloubka je vypočítávána pouze pro body rozlišitelné mezi kamerami, což ve venkovním prostředí nebude pravděpodobně zásadní problém, není ale možné spoléhat na dostupnosti hloubkových dat pro všechny body mapy. Pro blízké plochy je možné využít vestavěného emitoru strukturovaného světla, který promítá mřížku bodů o vlnové délce 859 nm. Efektivní dosah hloubkové kamery je zdola omezen na 0,5 m a shora možností identifikace jednotlivých bodů, ve venkovním prostředí je však za dobrého osvětlení až 10 metrů. [22]

2.5 Rychlost výpočtů a dostupný výpočetní výkon

Kritickým parametrem pro návrh algoritmu je nutnost provádět výpočty téměř v reálném čase. Tím je omezena složitost algoritmu, který nesmí být pro cílový hardware příliš výpočetně náročný.

2.5.1 Rozměry a napájení

Možný výpočetní výkon je limitován množstvím energie, kterou je schopen podvozek dodávat a prostorem, kam je možné počítač uložit. Prostor pro uložení počítačů a souvisejících síťových prvků je v podvozku omezen rozměry 46x23x13,5 cm. Celková spotřeba energie by se měla pohybovat maximálně do hranice 150W, avšak současná řešení (NVIDIA Drive PX2) se pohybují okolo hranice 250W. Už jen díky této skutečnosti je pravděpodobné, že výsledný algoritmus nebude tak přesný/rychlý jako algoritmy implementované v současných autonomních vozidlech. Cílem této

práce však není tyto algoritmy překonat, ale vyvinout vlastní řešení, které jim bude případně schopno konkurovat.

2.5.2 Hardware pro výpočty

Vzhledem k omezení volby hardware pouze spotřebou energie a rozměry je možné zvolit téměř jakýkoliv hardware v současnosti dostupný na trhu. Z toho důvodu byl jako počítač pro provádění veškerých výpočtů zvolen notebook ASUS ZenBook Pro UX501JW, vybavený procesorem Intel i7-4720HQ a grafickou kartou NVIDIA 960M. Přítomnost dedikované grafické karty obsahující 640 CUDA jader může v případě nutnosti umožnit akcelerovat výpočty na GPU. Počítač také obsahuje příslušnou konektivitu pro připojení senzorů, komunikaci s podvozkem (pomocí rozhraní Ethernet) a komunikaci s venkovním světem.

2.5.3 Rychlost výpočtů

Byl stanoven předpoklad, že robot nesmí od zpozorování překážky ujet bez reakce více než 0,5 metru. Vzhledem k maximální rychlosti 4 km/h ($1{,}\bar{1}$ m/s) to znamená, že robot musí zpracovat celý obraz alespoň 2,2x za sekundu. Z důvodu určitého zpoždění při přenosu povelů k zastavení/změně směru řídicí jednotce byl předem stanoven tvrdý limit zpracování alespoň 5 snímků za sekundu – maximálně 200 ms na zpracování.

3 Testovací data

Aby bylo možné programaticky testovat přesnost navrženého algoritmu, případně trénovat neuronové sítě, je nutné vytvořit sadu správných výsledků, proti které bude porovnáván výsledek navrženého algoritmu.

3.1 Sběr dat

První sada testovacích dat byla nasnímána 6.11.2017 v Broumově (Královéhradecký kraj) v klášterní zahradě a ve Schrollově parku v poledních hodinách za jasného počasí. Z toho důvodu je v testovacích datech možno pozorovat stín vrhaný objekty. Ideálně by testovací data byla naměřena v době, kdy bylo slunce skryto oblačností, aby objekty nevrhaly stíny a osvětlení bylo rovnoměrné.

3.1.1 Snímaná data a jejich formát

Data byla snímána z celkem čtyř úseků v dříve zmíněných parcích. Robot byl v konfiguraci jako při soutěži, pouze byl řízen manuálně. Snímány byly čtyři datové toky - RGB viditelné světlo, levá a pravá IR kamera, vypočtená hloubková data. Vše bylo snímáno v rozlišení 640x480 px (628x468 pro hloubková data). Maximální možná naměřená hloubka byla nastavena na 8 m aby byl s větší přesností pokryt „zajímavý“ prostor před robotem. Také bylo nastaveno automatické vyvážení bílé a automatické nastavení expozice.

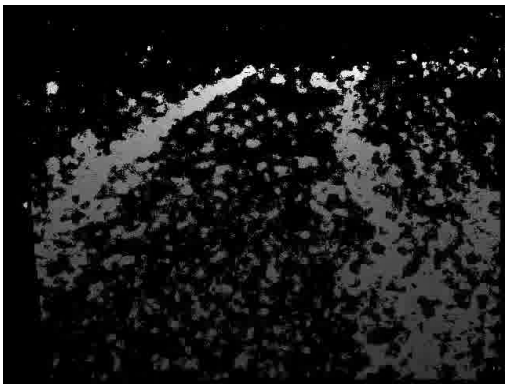
Z důvodu vysokých nároků na skladování (i JPEG komprimovaných) dat o frekvenci 30 snímků za jednotku času byl jako skladovací kontejner zvolen formát H.264. To sice přináší ztrátu detailů, ale domnívám se, že je výhodnější uchovat více snímků za sekundu s nižší kvalitou než naopak. Po vyhotovení prvních verzí algoritmu bude jasné, zda je důležitější rozlišení dat nebo jejich množství. Celkem bylo nasbíráno 1,27 GB dat o celkové délce 605 sekund (10 minut a 5 sekund).

3.2 Zpracování nasbíraných dat

Naměřená data, pro účely skladování uložená v H.264 formátu, nelze snadno následně používat. Pro zpracování je tedy prvně nutné extrahovat jednotlivé snímky. K vyzkoušení funkčnosti algoritmu není nutné mít všechny snímky, na základní ověření postačuje i např. jeden snímek za sekundu. Extrahování všech snímků ze zdrojového videa bylo provedeno programem `ffmpeg`:

```
ffmpeg -i color.avi %05d-color.png
```

(a) RGB



(b) Hloubka

(c) Levá IR složka

(d) Pravá IR složka

Obr. 3.1: Jednotlivé výstupy kamery

Formát PNG byl zvolen, aby se předešlo další ztrátové kompresi dat. Při vývoji také již příliš nevadí, že testovací data zabírají mnoho místa na disku. Během nasazení algoritmu nejsou tato data vyžadována. Vzhledem k tomu, že všechny datové toky byly snímány najednou, je následné spárování snímků mezi kamerami snadné - číselný prefix v názvu souboru reprezentuje číslo snímku ze zdrojového videa. Zdroj je identifikován slovem za pomlčkou.

3.3 Označení dat

Za využití knihovny OpenCV byla vytvořena jednoduchá aplikace `classify`, která načte snímky v dané složce a každý třicátý snímek poté uživateli zobrazí a umožní mnohoúhelníkem označit cestu. Levým tlačítkem se přidávají body mnohoúhelníku, pravé tlačítko ukončí kreslení a spojí první a poslední body. Aplikace prozatím umožňuje označení pouze jednoho mnohoúhelníku pro každou fotografii, toto omezení však není nijak zásadní – všechny dosud označené snímky neobsahovaly dvě

cesty, které by nebylo možné spojit. Výstupem aplikace je PNG soubor s maskou a textový soubor obsahující souřadnice bodů definujících mnohoúhelník. Bílá barva v masce označuje cestu, černá označuje okolí.

(a) Fotografie cesty (b) Maska

Obr. 3.2: Fotografie cesty a odpovídající maska

3.4 Vyhodnocení výsledků

Přesnost detekce cesty bude testována porovnáním vypočteného obrazu cesty proti člověkem vytvořenému etalonu. Z důvodu možnosti budoucího rozšíření algoritmu o určení jistoty, zda se jedná o cestu, bylo nutné navrhnout řešení reflektující tuto skutečnost.

$$přesnost = 1 - \frac{\sum_{x=1}^{šířka} \sum_{y=1}^{výška} |\boldsymbol{etalon}_{y,x} - \boldsymbol{vypočtený}_{y,x}|}{šířka \times výška} \quad [-] \qquad (3.1)$$

kde matice **etalon** je reference a matice **vypočtený** je výstupem algoritmu. Obě matice jsou normalizovány na rozsah hodnot $\langle 0, 1 \rangle$.

V praxi se bude k hodnotě přesnosti 1 možné dostat pouze u snímků obsahujících pouze cestu nebo naopak neobsahujících cestu vůbec. V obecném případě se k obrazu etalonu bude možné pouze přiblížit, protože ani etalon nebude nikdy dokonale přesný. Pro vzájemné relativní srovnávání úspěšnosti jednotlivých algoritmů detekce to však nepředstavuje problém.

3.5 Rozšířená sada testovacích dat

Na základě bodu 7 zadání této bakalářské práce jsem vytvořil rozšířenou sadu testovacích dat. Vzhledem k poznatkům nabraným při sběru původních testovacích dat,

implementaci původní verze algoritmu (kapitola 4) a nabízejícím se řešením při vývoji jsem se rozhodl, že rozsah původních testovacích dat byl dostatečný. Není tedy nutné snímat nějakou další veličinu (pozice, rychlost, ...), ani není nutné pozměňovat způsob sběru sbíraných dat.

Oproti datům sbíraným v roce 2017 je tedy málo rozdílů – roční období a prostředí. Nová sada dat byla sbírána na jaře roku 2018 v prostředích, která spíše odpovídají parkům nacházejícím se na soutěži Robotour. Jedná se zejména o šířky cest a o jejich povrchy (asfalt se v původních testovacích datech vůbec nevyskytoval)

4 Prvotní návrh algoritmu

Cílem této kapitoly je navrhnout základní verzi algoritmu, která bude v navazující kapitole optimalizována a rozšířena. Nejprve popisuji důvod, proč jsem se vydal při řešením směrem neuronových sítí. Poté jsou popsány použité prostředky – zejména neuronové sítě. Nakonec je algoritmus navržen, jsou popsány jeho jednotlivé kroky a na základní sadě testovacích dat vyhodnoceny výsledky.

4.1 Volba přístupu

Vzhledem k velkému množství doposud zveřejněného výzkumu detekce cesty s využitím klasických metod (prahování, detekce úběžníku, detekce jízdního pruhu) jsem se rozhodl pokusit se detekovat cestu pomocí neuronových sítí. Uvědomuji si, že toto řešení pravděpodobně nebude v současnosti energeticky nejúspornější, avšak s čím dál častější přítomností akcelerace pomocí GPGPU, speciálních instrukcí v čipech, případně dedikovaných procesorů (Google TPU [11], Intel Nervana [20]) se tato skutečnost může v budoucnosti změnit. Tyto specializované procesory využívají mnoho optimalizací, které by byly v obecných procesorech nedostupné. TPU tak dosahují až 83x větší účinnosti než CPU a 28,6x větší účinnosti než GPU. [11]

V případě nevyhovující rychlosti/přesnosti může být tato metoda v budoucnosti zkombinována s nějakou další metodou a vzájemně se doplňovat - jedním případem by mohl být přesný výpočet pomocí neuronové sítě a poté malé korekce jiným algoritmem, dokud nebude vypočten další výstup. Nejnovější automobily cílící na schopnost autonomního řízení bez jakéhokoliv zásahu řidiče bývají vybaveny počítačem NVIDIA Drive PX2 [23], který je vybaven výkonými GPU pro akceleraci neuronových sítí. To umožňuje případně přenést vyzkoušený algoritmus do normálního automobilu, kde by mohl fungovat s minimálními úpravami.

4.2 Využité prostředky

Jako základ (proof-of-concept) byla využita knihovna OpenCV, zejména díky tomu, že je již využívána pro zpracovávání obrazu z kamer.

4.2.1 Knihovna OpenCV

Knihovna OpenCV (Open Source Computer Vision Library) je uvolněna s BSD licencí a obsahuje mimo jiné i C++ rozhraní. Účelem knihovny je umožnit snadno získávat a manipulovat s obrazovými daty. V případě potřeby je možné využít akceleraci určitých funkcí pomocí GPGPU. Vzhledem k tomu, že obsahuje i základní

implementaci neuronových sítí - vícevrstvých perceptronů, byla využita i pro prvotní zkoušku, zda je možné pomocí neuronových sítí detekovat cestu. [5]

4.2.2 Volba programovacího jazyka

Jako programovací jazyk, ve kterém bude psána implementace této práce, jsem zvolil C++ . Tento jazyk byl zvolen z několika důvodů. Zejména v něm již byl napsán celý řídicí software podvozku robotu. Pro tento jazyk je také dostupných mnoho knihoven pro práci jak s nízkoúrovňovými zařízeními, tak i pro abstrakci. Existuje také mnoho nástrojů pro případné profilování/optimalizaci kódu, což bude nutné v druhé části této práce.

4.3 Neuronové sítě

Neuronové sítě jsou výpočetní systémy inspirované biologickými neurálními sítěmi. Takové systémy jsou schopny učit se rozpoznávat pomocí předem připravených příkladů. Díky tomu není nutné vymýšlet vlastní charakteristické body v obraze, pomocí nichž by byla cesta identifikována. Uživateli (programátorovi) stačí připravit dostatečně velkou sadu učících dat. Nevýhodou je nutnost výběru správných dat a modelu sítě.

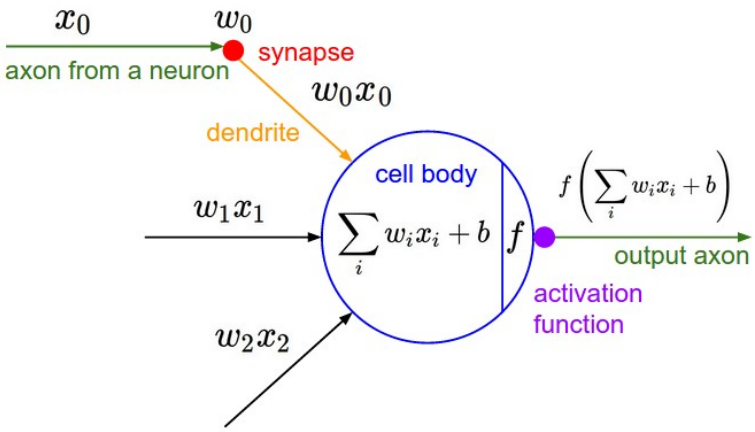
4.3.1 Popis neuronu

Neuron je základní stavební blok neuronových sítí. Díky jejich univerzálnosti se nekladou meze jejich využití – například klasifikace dat, aproximace funkcí či předpovídání řad.

Model neuronu

Každý neuron se skládá ze tří základních částí, viz. obr. 4.1:

- Vstupy - každá vazba s dalšími neurony je reprezentována váhou (důležitostí), s jakou ovlivňuje výsledek výpočtu.
- Sumátor, sčítající všechny vstupy konkrétního neuronu, vynásobené jejich důležitostí.
- Aktivační funkce, upravující amplitudu výstupu neuronu. Typicky je výstup transformován na rozsah <-1; 1>, případně <0; 1>. [8] V případě knihovny OpenCV a volby sigmoidní aktivační funkce je výstupem rozsah <-1,7519; 1,7519>. [24]



Obr. 4.1: Model neuronu [15]

4.3.2 Architektura sítě

Neuronová síť se může skládat z jedné a více vrstev neuronů. V praxi se rozlišují tři principiálně rozdílné architektury.

Jednovrstvé sítě

Ve *vrstvené* neuronové síti jsou neurony organizovány ve formě vrstev. Základní formou takové sítě je síť obsahující pouze jednu vrstvu neuronů. Protože neurony nemohou ovlivňovat předešlé vrstvy (žádné neexistují), jsou tyto sítě čistě dopředné.

Vícevrstvé sítě

Druhá třída dopředných neuronových sítí se rozlišuje tak, že architektura obsahuje alespoň jednu *skrytou* vrstvu. Úkolem skrytých vrstev je zasahovat mezi vstupem sítě a výstupem. Přidáváním skrytých vrstev je umožněno neuronové síti extrahovat z dat údaje vyššího řádu. Ve volném smyslu je možné říci, že síť díky tomu získává globální přehled. Užitečnost skrytých neuronů roste s rostoucí velikostí vstupní vrstvy.

Rekurentní sítě

Rekurentní neuronové sítě se od předchozích variant odlišují tím, že obsahují alespoň jednu zpětnou vazbu. [8]

4.3.3 Aktivační funkce

Každá aktivační funkce (nelinearita) bere jedno číslo, na kterém provádí danou matematickou operaci. V praxi se vyskytuje několik nejpoužívanějších: [15]

Sigmoidní aktivační funkce

Sigmoidní aktivační funkce je popsána vzorcem 4.1. Bere reálné číslo, které následně „zmáčkne“ do rozsahu <0; 1>. V minulosti byla tato aktivační funkce velmi používána, neboť se podobá skutečnému neuronu – neaktivní (0) nebo plně saturovaný (1). V praxi se již příliš nevyužívá. Hlavní nevýhodou je saturace výstupu k hodnotám 0 nebo 1. U krajních hodnot sklon aktivační funkce tak velký, důsledkem čehož se i při velké změně vstupu výstup mění pouze velmi málo. [15]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

ReLU

ReLU (Rectified Linear Unit) se v posledních několika letech stala velmi populární volbou. Definovaná funkcí viz vzorec 4.2. Velmi urychluje konvergenci gradientu při trénování neuronových sítí (toto bylo ověřeno v kapitole 5.7.1). Oproti sigmoidní aktivační funkci může být ReLU snadno implementována prahováním matice aktivací na 0. Nevýhodou je možnost, že ReLU jednotky „zemřou“ během trénování. V takovém případě se již nikdy neoživí a daný neuron pak ve výsledné síti nevykonává žádnou činnost. Tomuto problému je možné předejít správným nastavením trénování. [15]

$$f(x) = \max(0, x) \tag{4.2}$$

4.3.4 Trénování sítí

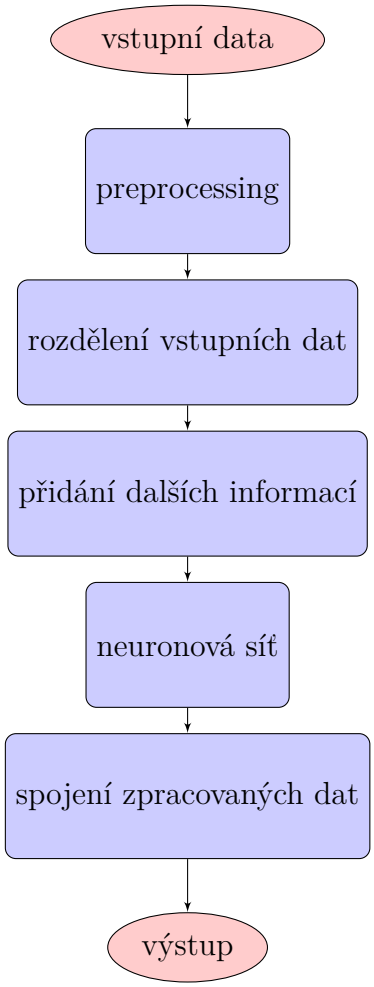
Před použitím neuronové sítě pro její úkol je nutné ji správně natrénovat. Tím se provede správné nastavení vah jejích neuronů.

Validace

Protože vyžadujeme, aby neuronová síť reagovala na vstupy, na které nebyla naučena (jinak by bylo možné využít jiných metod), je množina trénovacích dat rozdělena na dvě podmnožiny - trénovací a testovací. Jak již předpovídá název, pomocí trénovací množiny je neuronová síť natrénována, a pomocí testovací množiny je poté vyzkoušeno, s jakou přesností funguje (detekuje cestu). Tím je předejito volbě takového modelu, který je přeučen na testovacích datech, která umí velmi dobře detekovat, ale nefunguje pro data nedostupná během trénování. V současné konfiguraci je každý pátý snímek zařazen do testovací množiny.

4.4 Architektura algoritmu

Algoritmus byl navržen tak, aby byl snadno rozšiřitelný. Z toho důvodu je rozdělen na několik základních částí, jak je vidět na obrázku 4.2. Nejprve jsou vstupní data předzpracována, zejména kvůli úpravě rozměrů a odstranění šumu. Následně jsou rozdělena do menších bloků za účelem snížení komplexnosti celé neuronové sítě. K jednotlivým blokům jsou přidána další data (v současnosti pouze původní poloha). Takto rozšířené bloky jsou zpracovány a zpracovaná data jsou nakonec sešita zpět do stejného poměru stran, jaký měl vstup. Proces rozdělení je více popsán v podkapitole 4.6 a na obrázku 4.4.



Obr. 4.2: Průběh navrženého algoritmu

4.4.1 Zvolená konfigurace

Konfigurace algoritmu byla zvolena následovně: 32x24 sekcí, každá sekce o rozměrech 10x10 px. Výstup z každé sekce je 1 px, celkem je tedy detekován obraz 32x24

px. Přibližná rychlost zpracování na cílovém hardware je 10 snímků za sekundu, což splňuje podmínky definované v kapitole 2.5.3. Tato rychlost je pro algoritmus který nebyl příliš optimalizován a stále pracuje pouze s jedním jádrem procesoru. Proto lze předpokládat, že po optimalizacích, případně s využitím GPU, bude možné algoritmus buď rozšířit či zásadně zrychlit.

Neuronová síť využitá pro rozpoznání cesty v každé sekci obsahuje jednu skrytou vrstvu, jejíž velikost je definována takto: $velikost\ vrstvy = velikost\ sekce \cdot 4$. Protože neexistuje jedna správná rovnice na volbu velikosti skryté vrstvy, byla tato rovnice zvolena jako vyšší hranice doporučených rozsahů. Vstupní a výstupní vrstvy obsahují stejně neuronů jako je počet pixelů které obsluhují, vynásobený počtem barevných složek. Pro vstupní vrstvu jsou ještě přičteny přídavné vstupy. V současnosti jsou jedinými přídavnými vstupy dva neurony popisující polohu sekce v původním snímku. Tím je síti umožněno zohledňovat pozici – sekce vpravo nahoře bude obsahovat cestu v málo případech, naopak sekce uprostřed snímku ji bude obsahovat relativně často.

4.5 Preprocessing dat

Primárním účelem preprocessingu je upravit vstupní data do takového formátu, aby bylo možné je následně zpracovat neuronovou sítí. Jedná se zejména o změnu rozlišení obrazu, případně konverzi do správného datového formátu. Tyto kroky je možné přeskočit v případě kdy jsou již připravena na další zpracování.

Druhotným účelem předzpracování dat je jejich úprava tak, aby neuronové síti bylo zjednodušeno rozhodování. Takovými úpravami mohou být například změna barevného modelu nebo vyhlazení obrazu konvolučním filtrem, aby byl potlačen vliv šumu.

4.5.1 Změna barevného modelu

Segmentace obrazu často nevyužívá RGB barevného modelu, protože každá složka definuje intenzitu jiné barvy. Pro tuto práci byl barevný model, ve kterém jsou data předkládána neuronové síti, vybrán na základě empirických dat.

Jak je patrné z tabulky 4.1, z testovaných barevných modelů jsou nejpřesnější RGB a YCrCb. Barevný model RGB má proti YCrCb výhodu v celkové přesnosti vyšší o 2,1% a fakt, že vstupní data není nutné konvertovat do jiného barevného modelu, protože většina kamer na trhu již poskytuje data v tomto modelu. Nevýhodou je nejmenší přesnost nižší o 6,7%. Vzhledem k tomuto faktu byl zvolen barevný model YCrCb, protože se domnívám, že je prospěšnější i v nejhorších případech detekovat větší plochu cesty než mírně zvýšit celkovou přesnost.

Barevný model	Celková přesnost	Nejnižší dosažená přesnost
RGB	**89,1%**	29,8%
YUV	43,9%	8,2%
HSV	82,5%	32,4%
HLS	84,0%	28,0%
Lab	85,2%	25,1%
Luv	86,1%	30,2%
YCrCb	87,0%	**36,5%**

Tab. 4.1: Srovnání přesnosti algoritmu pro různé barevné modely

4.5.2 Vyhlazení vstupního obrazu

Šum ve vstupním signálu je potlačen vyhlazením. Obyčejné vyhlazovací algoritmy využívají pouze průměr okolních bodů, což však rozmaže i hrany, které následně nejsou dostatečně zřetelné. Pro zachování hran byl využit **mediánový filtr**. Jedná se o nelineární filtr, který vybere všechny pixely v okolí a z jejich hodnot následně vybere medián, který se stane novou hodnotou konkrétního bodu. Toto je provedeno pro všechny body zdrojového obrazu.

4.6 Rozdělení a spojení sekcí

Protože zpracování celého obrazu najednou (při zachování alespoň nějakých detailů) by bylo nesmírně výpočetně náročné (při zpracování na CPU), je obraz rozdělen do tzv. sekcí. Sekce mají konfigurovatelnou šířku, výšku. Pro celý obraz je poté definován počet sekcí, do kterých bude rozdělen. Před rozdělením je obraz upraven na velikost danou počtem sekcí a jejich rozměry.

Spojení je opakem rozdělení, kdy jsou jednotlivé sekce poskládány zpět vedle sebe a vznikne kompletní obraz výstupu.

4.7 Přidání dalších informací

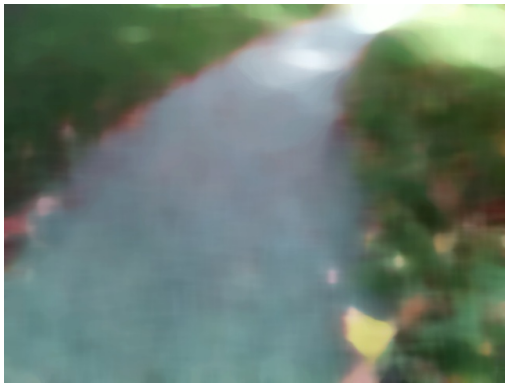
Neuronová síť nezpracovává pouze obrazová data, pro přidání globální perspektivy mohou být do vstupu přidána pomocná data, popsaná v následujících podkapitolách.

4.7.1 Pozice sekce ve snímku

Prozatím jediným pomocným vstupem přidávaným do vstupních dat je pozice konkrétní sekce v zdrojovém obrazu. Přidání takovéto informace není relativně náročné,

(a) Zdrojový snímek

(b) Box filtr



(c) Mediánový filtr

Obr. 4.3: Ukázka filtrů

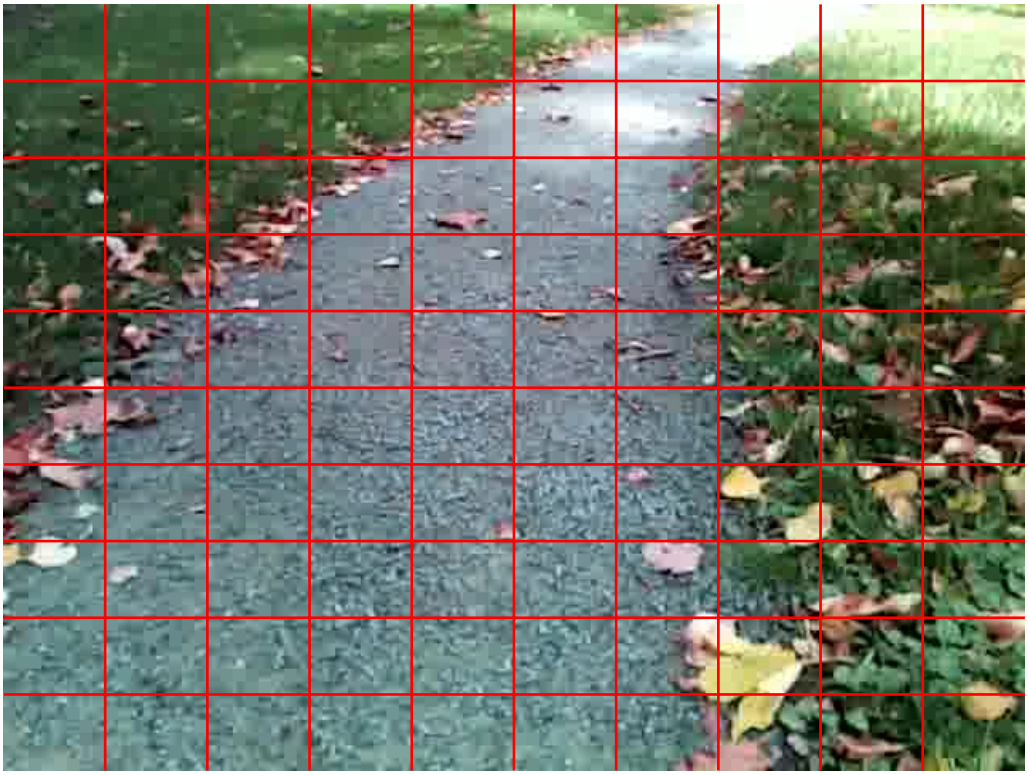
jedná se pouze o dva další neurony, označující pozici v osách X a Y. Ztráta výkonu je prakticky neměřitelná, ale přesnost algoritmu v konfiguraci 32x24 sekcí o rozměrech 10x10 px byla zvýšena o 0,5 % a nejnižší přesnost byla zvýšena o 1,4 %.

Jak je patrné ze obrázku 4.4, bez přídavných vstupů je v konfiguraci s snímkem rozřezaným do více sekcí neuronová síť naučena pouze na rozpoznávání barev, tudíž je velmi žádoucí vnést alespoň informaci o pozici. Taková informace by nemusela být nutná v případě zpracovávání celého obrazu najednou.

4.7.2 Další data

Mezi další možná data, která by mohla být v budoucnosti použita pro obohacení vstupu, patří například IR složky z kamery či hloubková data. Zejména IR složka z kamery by mohla být zajímavá, z obrázku 3.1 je patrné, že cesta narozdíl od travního porostu prakticky neodráží infračervené záření a je tak velmi dobře viditelná.

Protože detekujeme cestu ne na náhodných snímcích, ale defacto garantujeme,



Obr. 4.4: Ukázka rozdělení snímku do 100 sekcí

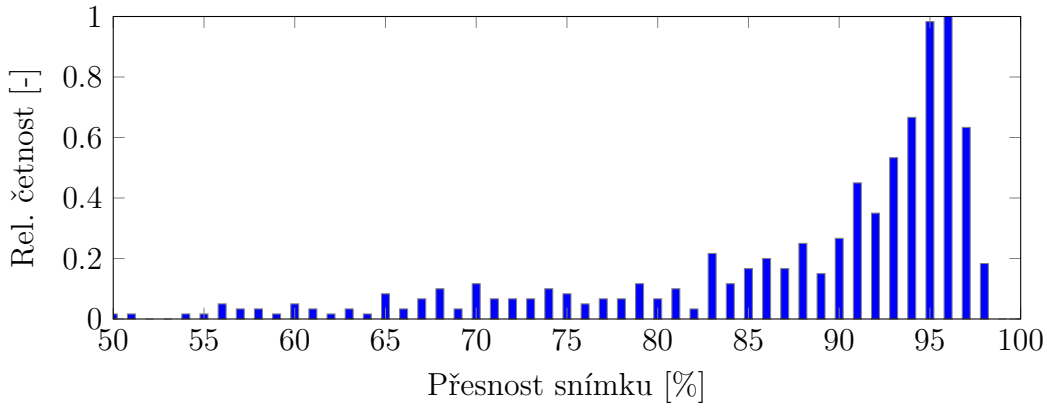
že následné snímky byly zaznamenány po sobě, nabízí se možnost dávat neuronové síti historii jejích výsledků. Tím by teoreticky mohlo být možné potlačit šum, kdy se změní pouze malá část obrazu, případně se celý obraz posune, což je při pohybu normální.

4.8 Postprocessing dat

Realizovaný algoritmus po provedení výpočtů neuronovou sítí a sestavení výsledku data již nijak neupravuje. Je však pravděpodobné, že v budoucnosti bude postprocessing alespoň částečně implementován. Jedním z možných důvodů implementace by bylo potlačení šumu ve výstupu, způsobeného malými „překážkami“, zejména spadlými listy na cestě.

4.9 Vyhodnocení prvotního návrhu

Navržený algoritmus dosahuje na testovacích datech průměrné přesnosti 87,7%, což je postačující pro algoritmus v této fázi návrhu. Rychlost výpočtů také splňuje požadavky definované v kapitole 2.5.3, kdy latence celého algoritmu vstup-výstup je přibližně 81ms (propustnost 12,3 snímků za sekundu).



Obr. 4.5: Histogram přesnosti výsledků prvotního návrhu algoritmu

(a) Vstup (b) Výstup

Obr. 4.6: Ukázka výstupu algoritmu

Jak je patrné z obrázku 4.6, algoritmus je schopen potlačit vliv stínů. Občas je však ve výstupu chyba, zejména v případech kdy je cesta zakrytá napadaným listím. Doufám však, že tyto problémy bude možné vyřešit optimalizacemi, které umožní zvětšit velikost sekce, a tím dodat neuronové síti více informací o kontextu.

5 Zrychlení a vylepšení algoritmu

V této kapitole naváži na předchozí část práce (kapitola 4), kde jsem navrhl prvotní verzi algoritmu a dokázal, že řešení za využití neuronových sítí má možnost fungovat. Tento návrh však nebral velké ohledy jak na optimalizaci, tak i na využívání dostupných prostředků. Zde budou tyto nedostatky odstraněny, díky čemuž bude možné zvětšit velikost sekce, na kterou neuronová síť kouká, čímž by mohla být dále zvýšena přesnost, případně celý algoritmus díky lepšímu využití výkonu zrychlen.

5.1 Profilace algoritmu

Za účelem zjištění, kolik času zabírají jednotlivé kroky celého programu byla provedena profilace vytížení procesoru.

Profilace celého programu `learn_road`, využívajícího předběžnou verzi algoritmu z kapitoly 4 došla k těmto výsledkům:

Funkce	Strávený čas	Komentář
main	98,03%	
LoadImage	9,01%	Načítání snímků z disku
RoadDetection::Load	0,18%	Načtení uložené sítě
RoadDetection::Predict	85,09%	Algoritmus detekce cesty
cv::ml::StatModel::predict	**65,99%**	**Inference**
RoadDetection::Blur	12,84%	Filtrování vstupu
RoadDetection::Prepare	1,31%	Změna formátu dat
RoadDetection::AppendPosition	1,09%	Přidání dat o poloze
cv::Mat::clone	0,46%	Tvorba kopie matice
cv::Mat::Mat	0,26%	Konstruktor matice
RoadDetection::ConstructOutput	0,25%	Spojení výstupu
RoadDetection::Cut	0,08%	Rozdělení vstupu

Tab. 5.1: Výsledky profilace předběžné verze algoritmu

V tabulce 5.1 jsou zobrazeny výsledky profilace předběžného návrhu algoritmu, uvedeny jsou všechny funkce, které představovaly alespoň 0,05% času běhu programu. Tučně je označena funkce, ve které program tráví nejvíce času.

5.2 Optimalizace a paralelizace algoritmu

Již na první pohled jsou v tabulce 5.1 vidět funkce, které je potřeba optimalizovat, případně snížit počet jejich volání. Budou však existovat funkce, které nebude možné citelně optimalizovat. Jednou z takových funkcí bude pravděpodobně `LoadImage`, která je limitována zejména rychlostí čtení z disku a ne výpočetním výkonem.

To, že samotná inference neuronové sítě je předpokládaný výsledek, nečekané však je, že filtrování vstupních dat mediánovým filtrem představuje přes osminu celkového času běhu programu, včetně načtení vstupních dat z disku. Na tuto funkci se bude nutné zaměřit při využití GPGPU akcelerace, případně alespoň knihoven paralelizujících maticovou aritmetiku na CPU. Optimalizace v této fázi není možná, protože funkce slouží prakticky pouze jako wrapper nad knihovnou OpenCV a neprovádí žádné vlastní výpočty.

5.2.1 Paralelizace na CPU

Protože cílový počítač obsahuje vícejádrový procesor, jako očividný směr zrychlení celého algoritmu se nabízí využití více jader zároveň pro samotnou inferenci, kdy každé vlákno by zpracovávalo svoji sekci původního obrazu, a následně by byly všechny vypočtené sekce zpět složeny do výstupního obrazu.

Výhody a nevýhody paralelizace na CPU

Největší výhodou paralelizace přímo v CPU je fakt, že není nutné přesouvat data do jiné paměti, jak by tomu bylo například při akceleraci s využitím GPGPU. Výhodou i nevýhodou pak je fakt, že celkově bude algoritmus možné zrychlit maximálně počtem dostupných vláken (8 v případě cílového počítače 2.5). Tento koeficient však může (a pravděpodobně bude) zásadně nižší, zejména z důvodu vzájemného využívání ALU/FPU více vlákny zároveň.

Dostupná je také možnost optimalizace pomocí vektorových instrukcí podporovaných „novými“ procesorech. Jedná se zejména o AVX, AVX2, případně i AVX-512, rozšiřující instrukční sadu x86. Zatěžování více částí procesoru najednou s sebou nese penalizaci v podobě snížení frekvence procesoru, je tedy nutné se zamyslet, zda získaný výkon vyváží nižší frekvenci (race-to-sleep).

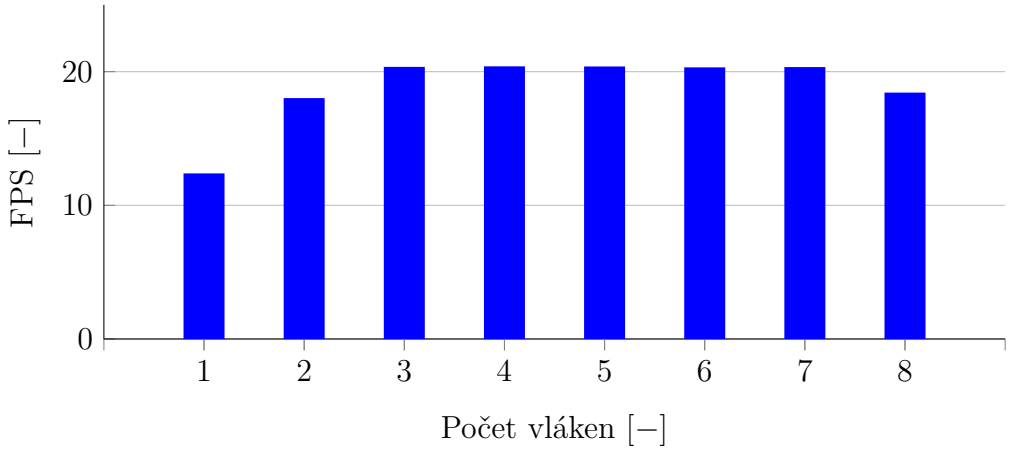
5.2.2 Výsledky optimalizace a paralelizace algoritmu

Algoritmus implementovaný v kapitole 4 byl optimalizován pouze částečně, neboť již z výsledků profilace je patrné, že u knihovny OpenCV nebude možné pro neuronové sítě setrvat. Toto rozhodnutí bylo podpořeno skutečností, že neuronové sítě

v knihovně OpenCV jsou nejen pomalé, ale také nepodporují složitější konfigurace (konvoluce, dropout vrstvy, ...). [1]

I přes výše zmíněné důvody byla již pro algoritmus využívající OpenCV implementaci neuronových sítí přidána podpora využívání více jader při inferenci. Proti jednovláknové verzi je rozdíl pouze v novém zásobníku, ze kterého si jednotlivá vlákna načítají indexy doposud nezpracovaných sekcí a ukládají je do výstupního kontejneru stejně, jako by se tomu dělo u pouze jednoho vlákna. Souběhům (race condition) je předcházeno využíváním lockfree fronty z knihovny Boost. Protože výsledná data se zapisují na různá místa v paměti, není nutné nikde zamykat proměnné, díky čemuž upravený algoritmus funguje i při využití pouze jednoho zpracovávajícího vlákna prakticky stejně rychle jako neupravený algoritmus.

Prvně bylo pro každou sekci vytvořeno samostatné vlákno, které po zpracování své sekce opět zaniklo. Toto řešení fungovalo, ale velký počet vláken způsobil nárůst množství změn kontextu, což ve výsledku drasticky snížilo celkovou rychlost. Tento problém byl opraven v druhé verzi, kdy bylo vytvořeno pouze malé množství vláken (ideálně stejně jako počet fyzických jader procesoru). Každé vlákno poté samostatně zpracovává jednotlivé nezpracované sekce dokud nejsou zpracovány všechny (viz popis výše).



Obr. 5.1: Vliv počtu paralelně běžících vláken na rychlost algoritmu

Jak je patrné z grafu 5.1, zvyšování počtu pracovních vláken po určitém bodě postrádá smysl je pouze plýtváním prostředky. Jako optimální řešení tak bylo zvoleno využívat 3 vlákna, což ponechá jedno fyzické jádro procesoru pro ostatní činnosti.

[1] Od verze 3.1.0 podporuje knihovna OpenCV díky Deep Neural Network modulu i složitější konfigurace, a to dokonce i akcelerované pomocí OpenCL. Neumožňuje však trénování, tudíž je pro účely této práce nepoužitelná.

5.3 Nová knihovna pro neuronové sítě

Jak bylo nastíněno v kapitole 5.2.2, již při začátku optimalizací celého algoritmu bylo jasné, že bude nutné najít lepší knihovnu implementující neuronové sítě.

5.3.1 Požadavky na knihovnu

Volba nové knihovny byla podmíněna několika parametry – zejména podpora programovacího jazyka C++ , možnost využití více aktivačních funkcí, architektur vrstev (plně spojené, dropout, konvoluce, ...) a nakonec možnost využití grafických karet, případně i jiných akcelerátorů, pro výpočty.

5.3.2 Zvažované knihovny

Mezi výběrová kritéria patří zejména podpora jazyka C++ , možnost implementace složitějších architektur, možnost akcelerace pomocí GPGPU, případně i specializovaného hardware, a v neposlední řadě i snadnost použití v projektu. Volba nové knihovny byla rychle zúžena na pouze několik alternativ:

TensorFlow

TensorFlow je open-source knihovna vyvíjená společností Google. Jedná se o matematickou knihovnu, kterou je možné využívat pro strojové učení. [3] Největší výhodou je velká komunita okolo této knihovny a možnost využívat Google TPU, akcelerátory navržené přímo pro tuto aplikaci. [11]. Podporována je také technologie CUDA, OpenCL oficiálně podporováno není, jsou však podnikány kroky k odstranění tohoto nedostatku. [25]

Ačkoliv se tato knihovna zdá jako ideální volba, nakonec nebyla zvolena z důvodu chabé podpory C++ . Jazyk je sice oficiálně podporován, ale v době volby nešlo v C++ API trénovat neuronové sítě. To by sice mohlo být obejito trénováním v jazyce Python a následným transportem dat zpět do jazyka C++ (například pomocí knihovny boost.python), ale pro účely této práce by takové řešení bylo příliš složité.

Caffe

Caffe je knihovna určená pro hluboké strojové učení původně vytvořená na univerzitě v Berkeley. [10] Oficiální verze sice podporuje pouze akceleraci pomocí CUDA, existují však verze, které implementují podporu OpenCL. [4]

Knihovna nakonec nebyla zvolena kvůli velkým obtížím při sestavování programu využívajícího tuto knihovnu, kdy bylo nutné přibalit mnoho závislostí, které není možné snadno automaticky kompilovat/stahovat.

Dlib

Dlib je moderní sada nástrojů, obsahujících mimo jiné i komponenty pro vývoj grafických rozhraní, síťové komunikace, zpracování obrazu a mnohých dalších aplikací. Pro účely této práce je však důležitá podpora strojového učení.

Akcelerace výpočtů probíhá buď pomocí CUDA na grafických kartách NVIDIA, v případě využívání CPU je možné urychlit výpočty knihovnami BLAS a LAPACK. [12] Nevýhodou je, že není možné ve stejný okamžik využívat CPU a GPGPU zároveň, tudíž nelze kupříkladu využívat GPGPU pro dynamické trénování modelu na právě sbíraných datech a CPU pro inferenci již natrénovaného modelu.

Ostatní knihovny

Existuje mnoho dalších knihoven pro strojové učení, každá se svým vlastním důvodem pro existenci. Většina jich byla vyřazena kvůli nesplňování nějakého kritéria. Nevylučuji však, že v případě reálného nasazení navrženého algoritmu bude možné využít jinou knihovnu.

5.3.3 Zvolená knihovna

Jako novou knihovnu jsem nakonec zvolil Dlib, zejména z důvodu velmi snadného přidání knihovny do existujícího projektu. V nástroji CMake stačí pouze přidat odkaz na knihovnu, kterou je dokonce možné držet jako git submodul. Knihovna si s pomocí CMake najde všechny potřebné závislosti a na vnější pohled vystupuje jako další součást programu. Podporu CMake navíc nebylo nutné do programu přidávat, sestavování zajišťuje již od počátku vývoje.

5.4 Efektivita využití vektorových instrukcí při výpočtech na CPU

Protože využívaný procesor obsahuje vektorové instrukce definované v rozšířeních instrukčního setu x86 AVX a AVX2, je nesmyslné toho nevyužít.

Knihovna Dlib umožňuje využití knihoven BLAS (Basic Linear Algebra Subprograms) a LAPACK (Linear Algebra Package). Existuje velké množství jednotlivých implementací těchto knihoven, některé optimalizované více než ostatní. Z důvodu cílového počítače využívajícího procesor Intel byl pro implementaci těchto knihoven využit balíček Intel Math Kernel Library (MKL) [9], který implementuje maticové operace právě s využitím vektorových instrukcí.

Jak je patrné z tabulky 5.2, využívání BLAS a LAPACK je v případě knihovny Dlib nutné již na vyrovnání rychlosti s původní OpenCV verzí. Výhodou využití Intel

	OpenCV verze	Bez knihoven	BLAS	LAPACK	BLAS + LAPACK
FPS	20	4	16	4	16

Tab. 5.2: Srovnání rychlosti algoritmu bez a s využitím Intel MKL pro 3 paralelní vlákna

MKL je však fakt, že trénování neuronové sítě probíhá s využitím všech fyzických jader procesoru. Konkurenční knihovny (OpenBLAS) toto nedokáží. Z vlastních testů na dalším počítači vyplynulo, že i přes obavy, že Intel uměle omezuje svůj kompilátor na cizích platformách, případná změna výrobce procesoru na AMD by neměla mít zásadní vliv na výkon (při srovnatelném výkonu procesoru).

Srovnatelný výkon Dlib verze oproti původní OpenCV implementaci jen prokazuje, že původní algoritmus nebylo možné znatelně optimalizovat, a je nutné se spíše ubírat směrem akcelerace s pomocí GPGPU, případně využít funkcionality nedostupné pro OpenCV verzi.

5.5 Akcelerace pomocí GPGPU

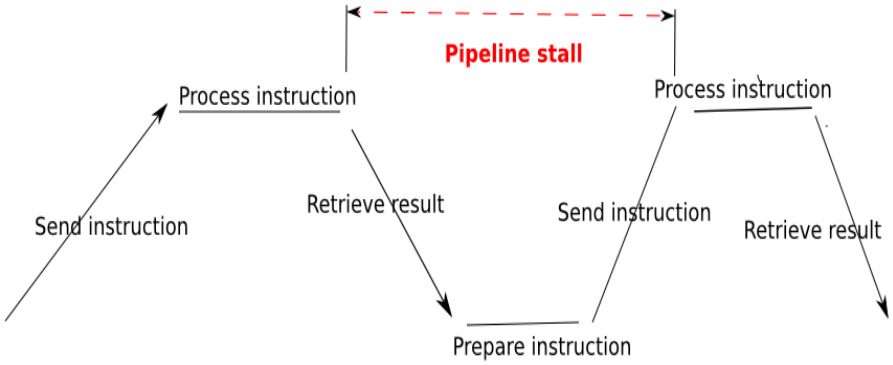
Neuronové sítě využívají zejména maticovou algebru, tedy operace podobné manipulacím s obrazovými daty. Díky tomu lze neuronové sítě snadno akcelerovat na grafických kartách. Existuje více technologií umožňujících snadnou akceleraci pomocí GPGPU prostředků, nejrozšířenější je však v současnosti cuDNN, využívající technologii CUDA společnosti NVIDIA.

5.5.1 Srovnání CPU/GPU verze

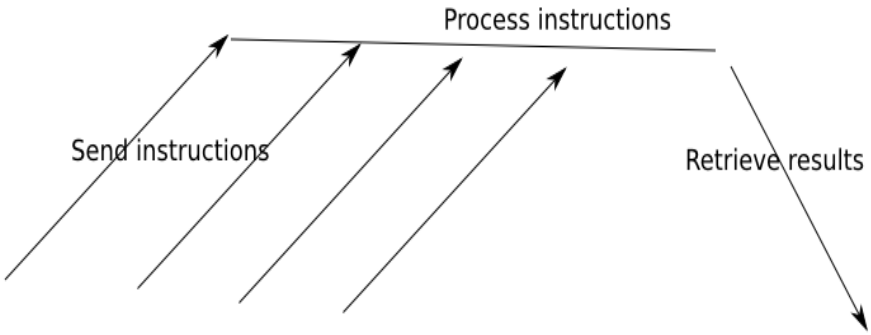
Využití GPGPU akcelerace v knihovně Dlib je velmi snadné, stačí pouze povolit využití CUDA a knihovna se postará o vše potřebné.

Protože pro výpočet s pomocí grafické karty je nutné nejprve přesunout data tam, a následně zpět do CPU, je tento proces velmi náchylný na tzv. „pipeline stall“, viz ukázka na obrázku 5.2. V takovém okamžiku není možné využívat zpracovávání dat okamžitě když je dostupný výpočetní výkon, ale je nutné čekat než se původní data vrátí a odešlou nová. Předejít tomu lze pomocí více nezávislých vláken, čímž bude zajištěno, že i při čekání jednoho vlákna na přesun dat bude jiné vlákno data zpracovávat. Nevýhodou takového řešení je nutnost obsluhovat více vláken, s čímž roste paměťová náročnost a celková složitost algoritmu.

Jak je vidět v grafu 5.3, zvyšování počtu vláken má jistý vliv i pro výpočty na grafické kartě, ale i pro velmi velký počet sekcí (přesunů dat) je rozdíl prakticky zanedbatelný (přibližně 10%). Z toho důvodu jsem se rozhodl, že raději využiji



(a) Čekání na přesun dat



(b) Řešení pauzy způsobené čekáním na data

Obr. 5.2: Znázornění příčiny a řešení pauzy způsobené čekáním na přesun dat [19]

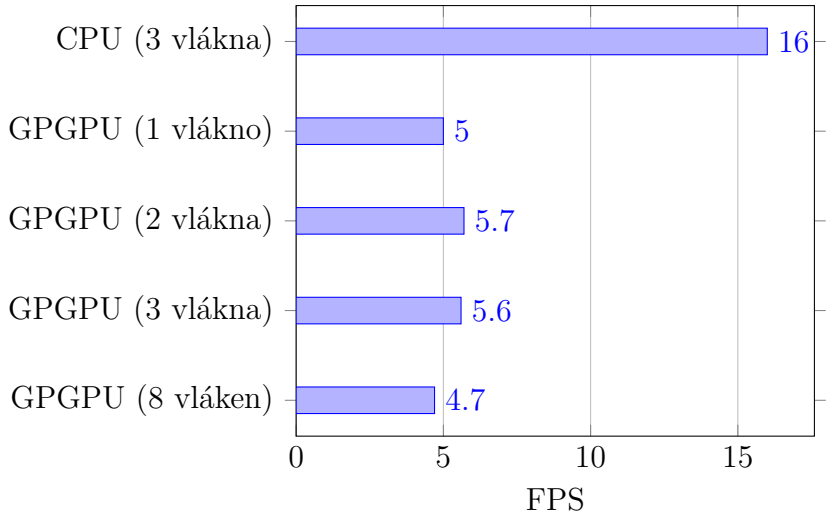
ušetřenou paměť pro větší velikost jednotlivých sekcí, čímž bude počet přesunů dále snížen.

5.6 Zvětšení sekcí

Již v kapitole 4.9 jsem zmínil, že se domnívám, že pro zvýšení přesnosti celého algoritmu bude nutné zvětšit velikost sekcí, čímž bude neuronové síti dodán kontext o okolí. Tehdy však byla velikost sekce omezena výpočetním výkonem, kdy se rychlost algoritmu velmi špatně škálovala s velikostí sekce. Díky jiné knihovně a případné akceleraci výpočtů je však nyní možné s většími sekcemi experimentovat.

5.6.1 Důvody pro a proti zvětšení sekcí

Hlavní myšlenka za zvětšováním sekcí je možnost dodání více kontextu, kdy v případě malých sekcí je nutné jich mít více pro pokrytí stejného vstupního obrazu. To



Obr. 5.3: Vliv počtu vláken na rychlost zpracování při využití GPGPU akcelerace

pak vede na nemožnost rozpoznávání tvarů a dalších prvků, neboť malé sekce tyto informace nemají jak obsahovat.

Nevýhodou větších sekcí je z důvodu zvolené architektury neuronové sítě fakt, že velikost vstupní vrstvy velmi zásadně ovlivňuje požadavky na paměť a výpočetní výkon, jelikož kvůli požadavku na přidávání přídavných dat do sítě je vstupní vrstva pouze jednorozměrná a ne dvojrozměrná a nelze tak na ni přímo provádět konvoluce jako u klasických konvolučních neuronových sítí. Maximální velikosti sekcí je také omezena dostupnou pamětí. V tomto případě se maximální velikost sekce pohybuje okolo 64x48 px na vstupu a 8x6 px na výstupu.

5.6.2 Výsledky zvětšení sekcí

Hlavní důsledek větších sekcí je skutečnost, že není nutné pro dosáhnutí stejného počtu vstupních neuronů používat mnoho malých sekcí. Touto změnou není sníženo rozlišení, neboť algoritmus umožňuje více výstupních hodnot pro jednu sekci. Přesnost také mírně stoupla, neboť algoritmus má nyní možnost rozpoznávat i hrany, což u velmi malých sekcí nebylo možné.

5.7 Změny v architektuře sítě

Se změnou knihovny zajišťující neuronové sítě a s využitím GPGPU akcelerace je možné měnit architekturu celé sítě, a v důsledku tedy i celého navrhovaného algoritmu.

5.7.1 Změna aktivační funkce

Knihovna OpenCV implementovala pouze dvě aktivační funkce – indentitu a symetrickou sigmoidní. Z toho důvodu bylo nutné využívat symetrickou sigmoidní funkci, neboť identita je ve většině případů pro účely neuronových sítí nevhodná.

Jako novou aktivační funkci jsem zvolil ReLU, zaprvé protože je v dnešní době využívána téměř ve všech nových publikacích, a zadruhé protože by podle [15] měla mnohem rychleji konvergovat než sigmoidní aktivační funkce. Toto bylo ověřeno při trénování, kdy nová neuronová síť dosáhla stejné chyby při učení prakticky okamžitě (v prvních několika generacích).

Změna aktivační funkce ze sigmoidní na ReLU přinesla zvýšení průměrné přesnosti algoritmu o přibližně 0,6% při nulovém zpomalení.

5.7.2 Využití konvolučních neuronových sítí (CNN)

Plně propojené neuronové sítě se obtížně škálují, kdy počet neuronů roste kvadraticky. V současnosti se pro detekci objektů a segmentaci používají konvoluční neuronové sítě, které automaticky předpokládají, že vstupem jsou obrazová data. Vstup neuronové sítě prochází alespoň jednou vrstvou konvolucí a následně do plně propojených vrstev jako u normální neuronové sítě.

Vzhledem k zvolenému přístupu k řešení problému – možnost přidávat další parametry k obrazovým datům. Z tohoto důvodu jsou vstupní data převedena do vektoru, nad kterým sice konvoluce provádět lze, ale protože už se nejedná o obrazová data v původní podobě taková činnost postrádá smysl.

5.7.3 Přidávání dalších plně propojených vrstev

Jak jsem odůvodnil v kapitole 5.7.2, využití konvolučních neuronových sítí není v současném návrhu možné. Jediným možným krokem je tedy přidávat další skryté plně propojené vrstvy, čímž vznikne hluboká neuronová síť (DNN - deep neural network).

Tento krok probíhal souběžně s dalšími vylepšeními (překrývání a zvětšování sekcí). Optimálním řešením bylo přidat před výstup neuronové sítě jednu další skrytou plně propojenou vrstvu, o počtu neuronů stejném jako jsou rozměry vstupní sekce.

5.8 Překrývání jednotlivých sekcí

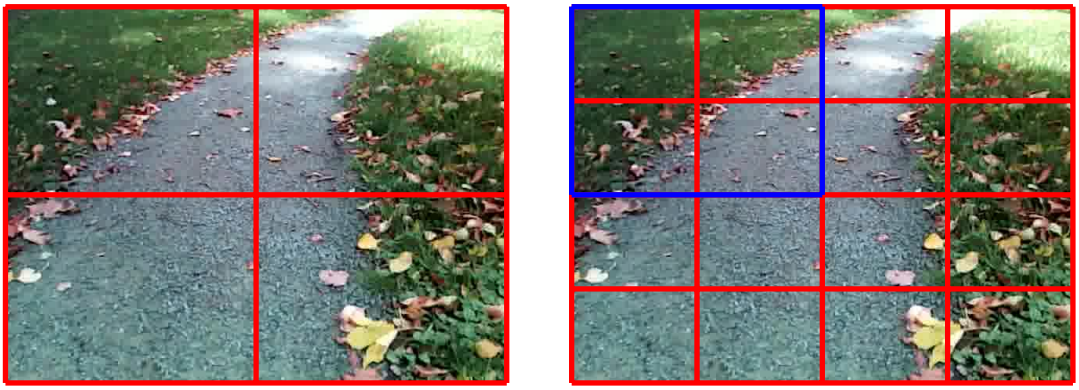
V původním návrhu se jednotlivé sekce nepřekrývají, o každém výstupním bodu tedy rozhoduje pouze jedna sekce. To není žádoucí, neboť pokud se konkrétní bod

nachází na okraji sekce, tak algoritmus vlastně neví co je za hranou, a nemůže tak správně rozhodnout.

Vzhledem k dostupnému výpočetnímu výkonu získanému prováděním výpočtů na grafické kartě je však možné implementovat jisté „pohyblivé okno“ (sliding window), kdy z původního obrazu budou použity nejen sousední sekce, ale i sekce, které leží na rozhraní jiných sekcí. Touto změnou je zajištěno nejen že všechny body budou alespoň jednou přibližně uprostřed zpracovávané sekce, ale také vzroste množství trénovacích vzorů pro neuronovou síť.

Pro účely této práce bylo implementováno pouze překrývání o polovinu šířky, respektive výšky jednotlivé sekce, ale nebyl by případně problém tuto skutečnost změnit. Z důvodu jednodušší implementace (aby nebylo nutné řešit „poloviční“ pixely) byla do algoritmu zavedena podmínka, že šířka i výška každé sekce při rozdělování i spojování dělitelná dvěma.

Protože se sekce mohou překrývat, roste i celkový počet jednotlivých sekcí, které z každého snímku vzniknou a následně budou vyhodnocovány, asymptoticky roste až k čtyřnásobku počtu sekcí při původním způsobu rozřezávání.



(a) Ukázka původního rozdělování obrazu (b) Demonstrace plovoucího okna

Obr. 5.4: Ukázka změny rozřezávání obrazu do sekcí

Jak je vidět na obrázku 5.4b, díky využití posouvajícího se okna (modrý obdélník) se body původně (obrázek 5.4a) ležící na hraně sekce dostanou v jiné sekci prakticky doprostřed, a může u nich tak být usuzováno více o kontextu. Plovoucí okno o stejných rozměrech jako je původní sekce se posouvá v červené mřížce.

5.8.1 Hlasování o výstupu

Protože kromě rohových podsekcí (čtvrtinová plocha původní sekce) se ostatní sekce alespoň jednou překrývají, je vlastně dosaženo toho, že o jednotlivých bodech neuronová síť rozhoduje vícekrát. Sestavování výstupu probíhá tak, že algoritmus začne

s maticí výstupů a maticí vah, oběma inicializovanými na hodnoty 0. Výstup každé sekce poté přičte na odpovídající místo v matici výstupu a zároveň hodnoty 1 na odpovídající místa v matici vah. Po provedení této operace pro všechny sekce je výstupní matice vydělena hodnotami odpovídajících prvků v matici vah. Tím je výstupní matice zpět normalizována na rozsah hodnot <0; 1>, kde výsledná hodnota značí, s jakou jistotou se algoritmus domnívá, že se v daném bodě jedná o cestu.

1	2	2	1
2	4	4	2
2	4	4	2
1	2	2	1

Tab. 5.3: Ukázka tabulky vah pro rozřezání obrazu jako na obrázku 5.4b

5.8.2 Druhá profilace

Po implementaci možnosti překrývání sekcí byla provedena znovu profilace celého programu, aby bylo zjištěno, zda spojování překrývajících se sekcí nemá příliš velký vliv na výkon. Z výsledků vyplynulo, že zpětné spojení je přibližně stejně rychlé (pravděpodobně díky optimalizaci maticí vah) jako v minulosti, ale rozdělování matice je náročnější, a v současnosti vyžaduje přibližně stejně času jako spojování. Při první profilaci bylo rozdělování matice přibližně třikrát rychlejší.

5.8.3 Výsledek

Z testování je patrné, že algoritmus díky tomuto vylepšení mnohem lépe detekuje velké plochy cesty, kdy v minulosti bez překrývání sekcí býval občas střed cesty vyhodnocován jako „ne cesta“, po této změně jsou již tato problematická místa detekována správně. Nevýhodou je snížení rychlosti algoritmu dvou až čtyřnásobně, což je však řešitelné v případě rozřezávání obrazu do méně sekcí.

6 Další vylepšení

Řešení navržené v této práci není jediné, které může vést k úspěšné metodě detekce cesty. V této kapitole se pokusím nastínit různé způsoby, jaké další vylepšení by mohly vylepšit současný algoritmus.

6.1 Pipelining při zpracování algoritmu

V současnosti algoritmus zpracovává vstupní data sériově, v jeden okamžik pouze jeden snímek. Je však možné řadit snímky za sebe, podobně jako se řadí instrukce v moderních procesorech („pipelining“). Naneštěstí takto nelze snížit latenci, je však možné tímto způsobem zvýšit celkovou propustnost.

6.2 Dynamický rozsah kamery

Jak je patrné z obrázku 6.1, použitá kamera nemá dostatečný dynamický rozsah, kdy u určitých snímků neobsáhne zároveň povrchy ve stínu i povrchy na které dopadá přímé sluneční záření. V takových případech pak nemá algoritmus jak určit, zda se jedná o cestu nebo ne, i když se nejedná přímo o chybu algoritmu, jelikož ani člověk si s takovými případy nedokáže poradit.

6.3 Generování více trénovacích dat

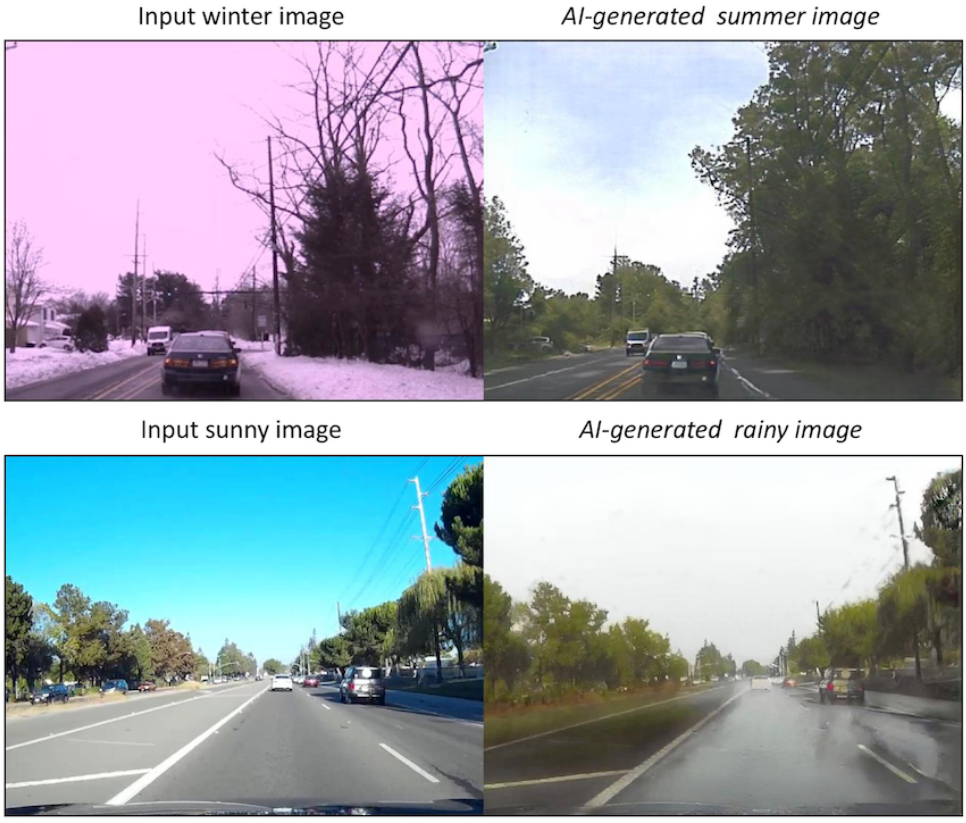
Základním problémem trénování neuronových sítí je skutečnost, že je nutné vytvořit velké množství dat, ze kterých se síť následně učí. Tyto data však musí připravovat člověk, a tak je tato činnost velmi časově náročná.

V kapitole 5.8 představuji jeden způsob jak získat více dat ze stejného množství člověkem označených snímků. Dalším způsobem by bylo například [16], kdy další neuronová síť mění vstupní data v uživatelem zadaném způsobu – například slunečný den transformuje na zasněžené prostředí.

6.4 Změna hodnocení přesnosti

V současnosti je přesnost algoritmu hodnocena vzorcem 3.1. Tento vzorec však nezohledňuje pozici ve snímku, což nemusí být nutně ideální při porovnávání různých verzí algoritmu.

Obr. 6.1: Ukázka nedostatečného dynamického rozsahu kamery



Obr. 6.2: Ukázka změny obrazu bez uživatelského zásahu [16]

7 Vyhodnocení navrženého řešení

Vyhodnocování konečného algoritmu probíhalo ve stejném duchu jako vyhodnocení předběžné verze v kapitole 4.9.

Algoritmus byl vyhodnocován jak na původních testovacích datech, tak i na rozšířených testovacích datech z kapitoly 3.5.

7.1 Výsledná konfigurace

Základní princip algoritmu se neodchýlil od prvního konceptu. Obraz je stále rozdělován do sekcí, které jsou jednotlivě vyhodnocovány a následně jsou sekce zpět spojeny aby vznikl výsledný obraz. Z konečné verze algoritmu bylo vypuštěno filtrování vstupního obrazu mediánovým filtrem, neboť vliv filtrace na přesnost byl minimální, ale zpomalení celého algoritmu vlivem filtrace již zanedbatelné nebylo (viz kapitola 5.1).

Vstupní obraz je dělen do 4x3 sekcí o rozměrech 36x36px. Každá sekce představuje výstup 8x8 bodů. Je využito překrývání sekcí popsané v kapitole 5.8, takže je celkem zpracováváno 35 sekcí. Celkové rozlišení výstupu je 32x24 bodů, algoritmus tedy rozhoduje o cestě v 768 bodech.

Konfigurace samotné neuronové sítě je popsána v tabulce 7.1. Jedná se o neuronovou síť se dvěma skrytými vrstvami.

	Rozměr	Počet neuronů [-]
vstup	36x36x3 + 2	3890
1. skrytá vrstva	36x36x4	5184
2. skrytá vrstva	36x36	1296
výstup	8x8	64

Tab. 7.1: Architektura výsledné sítě

7.2 Rychlost výpočtů

Požadavky na rychlost/latenci algoritmu byly splněny, na cílovém počítači algoritmus zpracuje obraz od přijetí průměrně za 84 ms, což je přibližně 12 FPS. To je více než dvojnásobná rychlost než jaká byla na úvodu celého řešení práce stanovena jako cílová. Prakticky stejné hodnoty dosahoval také prvotní návrh algoritmu.

Stejný algoritmus spuštěný na vývojovém počítači vybaveném grafickou kartou NVIDIA 960 s 2 GB paměti zpracovává přibližně 15 snímků za sekundu i přes to, že

má více než dvojnásobný počet CUDA jader. Z toho vyvozuji, že grafická karta není plně využívána a v budoucnu by mohlo být možné využít tento dostupný výpočetní výkon.

7.3 Vyhodnocení na testovacích datech

Výsledný testovací algoritmus byl testován na datech nepoužitých při trénování neuronové sítě – ověřován byl tedy pouze každý pátý označený snímek.

Na původních testovacích datech dosáhl průměrné přesnosti 84,9% a mediánové přesnosti 89,1%. Toto je sice mírný pokles úspešnosti oproti prvotnímu návrhu algoritmu, nová verze je však robustnější.

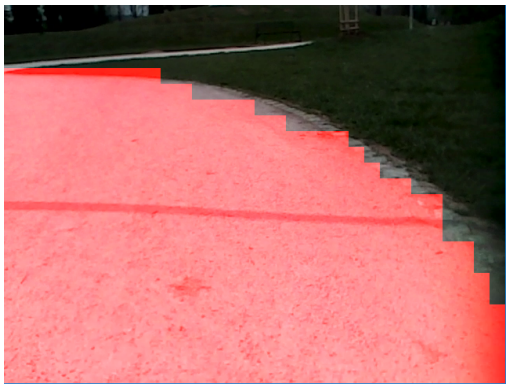
Algoritmus na nových testovacích datech (kapitola 3.5) dosahoval průměrné přesnosti 94,7% a mediánové přesnosti 95,1%.

Na grafu 7.2 jsou zobrazeny histogramy relativních četností výskytu přesností, s jakou algoritmus určil cestu. Graf 7.2b zobrazuje osu x pouze v rozsahu od 80 do 100%, neboť žádné snímky nedosáhly přesnosti nižší než 84%. Velký rozdíl v přesnosti algoritmu na starých a nových testovacích datech je způsoben zejména složitostí původních testovacích dat, kdy na cestě leží popadané listí, v částech testovacích dat je místo upravené cesty pouze pěšina vyložená velkými kameny, případně i velké změny v osvětlení. Takové případy však na soutěži Robotour nenastávají často, tudíž by to mělo být bráno pouze jako nejhorší situace, která teoreticky může nastat.

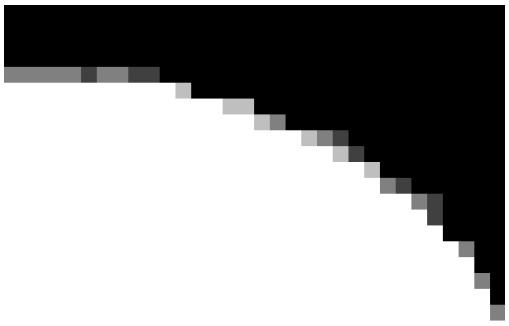
7.4 Osobní vyhodnocení

Navržený algoritmus splňuje všechna mnou zadaná kritéria. Na testovacích datech z prostředí podobnému tomu, do kterého je určen byl algoritmus schopen dosáhnout vysoké přesnosti a nízkého zpoždění při zpracování dat. Díky tomu by neměl být problém jej nasadit při soutěži Robotour.

(a) Vstupní data

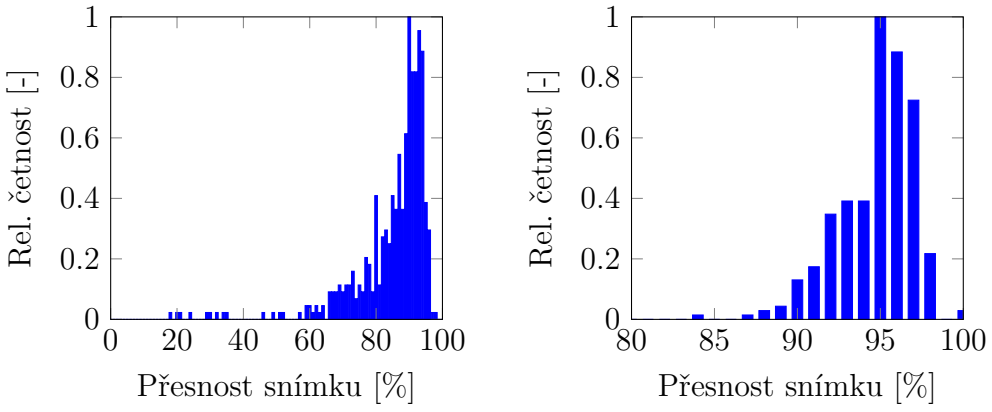


(b) Označený výstup algoritmu



(c) Neupravený výstup algoritmu

Obr. 7.1: Ukázka výstupu navrženého algoritmu



(a) Stará data

(b) Nová data

Obr. 7.2: Histogramy přesností algoritmu na testovacích datech

8 Závěr

V úvodu práce je definován cíl této práce a popsán problém detekce cesty a využití výsledků. Cílem této práce bylo navrhnout postačující řešení problému detekce cesty, které bude schopné obstát na soutěžích Robotour a jí podobných – udržení pohybujícího se robotu na cestě.

V první kapitole byla provedena rešerše současných a historických přístupů k detekci cest. Byly zhodnoceny výhody a nevýhody jednotlivých přístupů, případně uvedeny důvody, proč je nelze aplikovat při návrhu vlastního algoritmu.

Referenční podvozek, na kterém byla snímána testovací data z kapitoly 3 je popsán v kapitole 2. Popsány byly i dostupné snímače a poloha kamery na samotném podvozku. Zvláštní pozornost byla věnována vícespektrální kameře Intel RealSense R200, která umožňuje kromě viditelného světla snímat také infračervené záření a pomocí stereo vidění dopočítat hloubku. Ve stejné kapitole byl také vyhodnocen dostupný výpočetní výkon, respektive byly definovány parametry, které musí počítač splňovat. Tyto parametry byly definovány s ohledem na současná řešení v automobilovém průmyslu.

Před samotnou implementací byla nasbírána pomocí referenčního podvozku testovací data, ve kterých byla manuálně označena cesta. Tyto data jsou využita pro automatizované vypočítávání přesnosti algoritmu, kdy je porovnávána referenční maska vytvořená uživatelem a výstup navrženého řešení. Toto vypočítávání přesnosti je popsáno v kapitole 3. Data byla klasifikována pomocí mnou vytvořeného programu pro rychlé kreslení masky cesty.

Následně v kapitole 4 navrhuji základní verzi algoritmu detekce cesty pomocí neuronových sítí. Neuronová síť byla natrénována pomocí klasifikovaných dat z třetí kapitoly. Navržené řešení je velmi limitováno tím, že je vypočítáváno na CPU a není využito jakékoliv hardwarové akcelerace (zejména GPU). Přesnost navrženého řešení na testovacích datech je 87,7%.

Navržený algoritmus je následně v kapitole 5 nejprve analyzován, aby byla zjištěna jeho časová efektivita a nalezeny paralelizovatelné části. Na základě těchto informací jsou navrženy optimalizace celého algoritmu, zejména využití více jader pro inferenci neuronové sítě. Využití více jader sice rychlost algoritmu zvýšilo, ale ne dostatečně. Pro další zrychlení je vyměněna knihovna zajišťující neuronové sítě a je využito GPGPU akcelerace. Navrhuji také jiné vylepšení původního algoritmu, například zvětšení velikosti sekcí proti původnímu návrhu a změna algoritmu aby umožňoval překrývání sekcí. Tím je získáno více testovacích dat, díky čemuž je algoritmus robustnější.

V kapitole 7 je výsledné řešení problému detekce cesty vyhodnoceno na obou sadách testovacích dat. Shledal jsem, že v podmínkách podobných soutěži Robotour

je algoritmus velmi úspěšný, a i v obtížných situacích je schopný určovat cestu (i když ne tak přesně).

V neposlední řadě se také v kapitole 6 zamýšlím nad dalšími možnými řešeními, které by mohly vést k lepším výsledkům. Jedná se zejména o vylepšení kamery a získání více trénovacích, respektive testovacích dat.

S rozpracovanou verzí této práce jsem se zůčastnil soutěže Student EEICT 2018, kde byla oceněna třetím místem v bakalářské kategorii Kybernetika a automatizace (bakalářská) a speciální cenou společnosti Thermo Fisher Scientific.

Literatura

[1] Sonka, M.; Hlavac, V.; Boyle, R.: *Image Processing, Analysis and Machine Vision*. PWS Publishing, 1998.

[2] Everett, H. R.: *Sensors for Mobile Robots: Theory and Application*. Peters, 2001.

[3] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015, software dostupný z tensorflow.org. URL `https://www.tensorflow.org/`

[4] Advanced Micro Devices: OpenCL-caffe. `https://github.com/amd/OpenCL-caffe`, získáno: 2018-05-13.

[5] Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Dlouhý, M.: Robotour 2013. `https://robotika.cz/competitions/robotour/2013/results/cs`, získáno: 2017-12-27.

[7] Dlouhý, M.; Crha, J.: Robotour - Pravidla. `https://github.com/robotika/robotour/blob/ROBOTOUR2017/rules/pravidla.md`, získáno: 2017-12-27.

[8] Haykin, S.: *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994, ISBN 0023527617.

[9] Intel Corporation: *Intel Math Kernel Library. Reference Manual*. Intel Corporation, 2009, ISBN 630813-054US.

[10] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[11] Kaz Saro, D. P., Cliff Young: An in-depth look at Google's first Tensor Processing Unit (TPU). `https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu`, získáno: 2017-12-19.

[12] King, D. E.: Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, 2009: s. 1755–1758.

[13] Kong, H.; Audibert, J. Y.; Ponce, J.: Vanishing point detection for road detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, ISSN 1063-6919, s. 96–103, doi:10.1109/CVPR.2009.5206787.

[14] Kortli, Y.; Marzougui, M.; Atri, M.: Efficient implementation of a real-time lane departure warning system. In *2016 International Image Processing, Applications and Systems (IPAS)*, Nov 2016, s. 1–6, doi:10.1109/IPAS.2016.7880072.

[15] Li, F.-F.; Karpathy, A.; Johnson, J.: CS231n: Convolutional Neural Networks for Visual Recognition. 2018.
URL `http://cs231n.stanford.edu/`

[16] Liu, M.; Breuel, T.; Kautz, J.: Unsupervised Image-to-Image Translation Networks. *CoRR*, 2017, získáno: 2018-04-18.
URL `http://arxiv.org/abs/1703.00848`

[17] Mendes, C.; Fremont, V.; Wolf, D.: Exploiting Fully Convolutional Neural Networks for Fast Road Detection. In *Exploiting Fully Convolutional Neural Networks for Fast Road Detection*, 05 2016, s. 1–6.

[18] Mikšík, O.: *Dynamic Scene Understanding for Mobile Robot Navigation*. Diplomová práce, Vysoké učení technické v Brně, 2012.

[19] Perkins, H.: Pipeline Stall. `https://github.com/hughperkins/cltorch`, získáno: 2018-05-14.

[20] Carey Kloss: Intel Nervana™ Neural Network Processor: Architecture Update. `https://ai.intel.com/intel-nervana-neural-network-processor-architecture-update/`, získáno: 2017-12-19.

[21] Hi-Tech Robotic Systemz Ltd.: Line follower. `https://commons.wikimedia.org/wiki/File:IntelliCart1.jpg`, získáno: 2018-01-05.

[22] Intel Corporation: Intel® RealSense™ Camera R200. `https://www.mouser.com/pdfdocs/intel_realsense_camera_r200.pdf`, získáno: 2018-01-05.

[23] NVIDIA Corporation: Tesla Motors Partnership. `https://www.nvidia.com/en-us/self-driving-cars/partners/tesla/`, získáno: 2018-01-05.

[24] OpenCV Developer Team: Neural Networks. `https://docs.opencv.org/2.4/modules/ml/doc/neural_networks.html`, získáno: 2017-12-15.

[25] TensorFlow Team: OpenCL support. https://github.com/tensorflow/tensorflow/issues/22, získáno: 2018-05-13.

Seznam symbolů, veličin a zkratek

CPU Central Processing Unit – centrální procesorová jednotka

ALU Arithmetic Logic Unit – aritmeticko-logická jednotka, komponenta procesoru, zpracovávající všechny celočíselné aritmetické a logické operace

FPU Floating-point Unit – matematický koprocesor vykonávající operace s čísly s plovoucí řádovou čárkou

AVX Advanced Vector Extensions – rozšíření instrukční sady x86 umožňující paralelizaci aritmetických operací

GPU Graphics Processing Unit – grafická karta

GPGPU General Purpose GPU – akcelerace výpočtů pomocí grafických karet

LIDAR Light Detection and Ranging – metoda měření vzdálenosti pomocí měření času letu laserového pulzu

px pixel (obrazový bod)

RGB barevný model vyjádřený třemi složkami - červená, zelená, modrá

TPU Tensor Processing Unit

Seznam příloh

A Obsah přiloženého DVD

- `/` ..... kořenový adresář přiloženého DVD
  - `demo` ..... ukázky funkčnosti
    - `CUDA.zip` ..... CUDA verze ukázek
    - `CPU.zip` ..... CPU verze ukázek (pomalé)
    - `demo_Broumov.avi`
    - `demo_HK.avi`
    - `README.txt`
  - `data` ..... nasnímaná data
    - `nasnímaná, případně označená data`
  - `kod` ..... zdrojový kód
    - `BP_xpeska04_kod.zip`
    - `README.txt`
  - `BP_xpeska04.pdf` ..... elektronická verze této práce