VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Aleš Raszka

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KLASIFIKACE VOZIDEL S POUŽITÍM RADARU

VEHICLE CLASSIFICATION USING RADAR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. ALEŠ RASZKA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. LUKÁŠ MARŠÍK**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19268/2016/xraszk03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Raszka Aleš, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Klasifikace vozidel s použitím radaru**
Vehicle Classification Using Radar
Kategorie: Zpracování signálů

Pokyny:

1. Nastudujte literaturu související se zpracováním radarového signálu a strojovým učení.
2. Vytipujte vhodné příznaky včetně různých tříd, mezi kterými bude možno klasifikaci provádět.
3. Extrahujte příznaky pro klasifikaci různých tříd objektů na základě radarových dat.
4. Implementujte klasifikátor pro vybrané úlohy.
5. Nasbírejte reálná data na zapůjčeném HW. Na těchto datech proveďte sérii vhodných testů pro zjištění výkonnosti a spolehlivosti systému.
6. Diskutujte dosažené výsledky a navrhněte možné pokračování práce.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Maršík Lukáš, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá tématem využití radarového signálu ke klasifikaci silničního provozu. Součástí práce je použití radarových modulů s kontinuální vlnou využívajících Dopplerův jev. Radarový signál je následně podroben sérii metod pro zpracování signálu, jež je zakončena Fourierovou transformací. Ze získaných dat jsou vytvořeny klasifikátory metodami SVM a AdaBoost, pomocí kterých jsou vozidla klasifikována do skupin.

Abstract

This Master thesis deals with usage of radar signal for vehicle classification. The thesis uses radar modules with continuos wave based on Doppler effect. Radar signal is processed by a series of signal processing method finished by Fourier transform. Data produced by FFT is used to create SVM and AdaBoost classifier which can be used to classify vehicles into groups.

Klíčová slova

Dopplerův jev, radar, zpracování signálu, klasifikace, vozidlo, SVM, Adaboost

Keywords

Doppler effect, radar, signal processing, classification, vehicle, SVM, Adaboost

Citace

RASZKA, Aleš. *Klasifikace vozidel s použitím radaru.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Maršík Lukáš.

Klasifikace vozidel s použitím radaru

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Lukáše Maršíka a uvedl jsem všechny zdroje, ze kterých jsem čerpal.

.....
Aleš Raszka
21. května 2017

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Ing. Lukáši Maršíkovi za cenné rady a ochotu, kterou projevoval po celou dobu tvorby této práce. Děkuji také své přítelkyni, rodině a přátelům, kteří mi byli důležitou oporou po celou dobu mého studia.

Obsah

Kapitola 1

Úvod

Dnešní doba, v níž automatizovaná práce nahrazuje tu manuální a počítačové aplikace usnadňují práci nám lidem, vyžaduje mnoho metod k tomu, aby počítač sám bez přispění člověka dokázal alespoň částečně přemýšlet. Slovo přemýšlet musíme brát v tomto kontextu s velkou rezervou, jelikož dnešní umělá inteligence počítačů zatím nedosahuje ani zdaleka takových výsledků jako ta lidská. Obor zabývající se touto problematikou se nazývá strojové učení, celosvětově známe jako *machine learning*.

Toto technologické odvětví obsahuje celou řadu metod pro to, aby stroje dokázaly alespoň částečně samy přemýšlet a učit se. Jedna z jeho podkategorií se zabývá klasifikací objektů do různých tříd na základě jejich vzhledu a vlastností. Této techniky můžeme využít v celé řadě odvětví a například namísto toho, aby u výrobního pásu stál člověk, počítal a kategorizoval vyrobené zboží, může tuto práci dělat stroj a to daleko rychleji a mnohdy i přesněji. Tuto techniku lze uplatnit i v automobilovém průmyslu, a to konkrétně při monitorování dopravy na silnicích. Mnohdy se snažíme optimalizovat dopravu tak, aby nevznikaly ve městech dopravní zácpy a podobné problémy. K tomu abychom mohli úspěšně optimalizovat dopravu, musíme získat celou řadu statistik o provozu na silnicích. Právě k získání těchto statistik nám může pomoci některá z metod strojového učení a klasifikace.

Ve většině případů, kdy se snažíme dopravu jakkoli monitorovat, jsou použity k snímání silnic kamery. Kamery jsou nám blízké, jelikož dokáží zachytit zhruba podobné informace jako naše lidské oko formou rastrového obrazu. Následně pomocí metod rozpoznávání v obraze lze z pořízených snímků kamery zachytit například tvar vozidla, barvu, SPZ a řadu dalších informací. Jsou však určité případy, kdy klasické kamery selhávají a tudíž nejsou použitelné. Těmito případy jsou například špatné světelné nebo povětrnostní podmínky, kdy nastává mlha, déšť či sněžení.

Právě z výše uvedených důvodu je vhodné využít jinou technologii získání přehledu dění ve scéně. Jednou z možných alternativ je využití radaru. *Radar* je elektronický přístroj vysílající do prostoru elektromagnetické vlnění známé jako radarové vlny. Tyto vlny nejsou ovlivněny světelnými podmínkami, jelikož je radar aktivní senzor, a tak může fungovat i v noci. Pod pojmem radar si většina lidí představí měření rychlosti na silnicích, avšak radar může sloužit i k daleko širšímu spektru využití. Na základě odražených radarových vln a správného zpracování přijatého signálu můžeme získat soubor hodnot popisující objekt. Na rozdíl od snímku z kamery sice pomocí tohoto číselného popisu nejsme, my jako lidé, schopni rozeznat objekty, avšak pro počítačové zpracování je tento popis mnohdy i lepší, jelikož neobsahuje spousty přebytečných informací.

Tato práce si klade za cíl vytvořit softwarovou implementaci, která bude schopna klasifikovat radarová data ze silničního provozu. Základem celého systému bude radar s kon-

tinuální vlnou, který bude využívat Dopplerova jevu. Nasbíraná data ze silničního provozu budou dále zpracována metodami pro zpracování signálu, ze kterých bude získán příznakový vektor pro klasifikaci.

Teoretická část práce je členěna do tří logických celků, kdy každý z nich popisuje jednu z pod-částí řetězce, který je zakončen klasifikací. V kapitole 2 je popsán základní princip fungování radaru, dále taky běžně používané typy radaru a na závěr konkrétní radarové moduly. V kapitole 3 je popsána série metod pro zpracování signálu, jimiž musíme přijatý signál z radaru podrobit, abychom získali validní data pro klasifikaci. V další kapitole 4 je popsána problematika strojového učení, a to konkrétně téma klasifikace. V této kapitole jsou rovněž vysvětleny konkrétní metody klasifikace, které budou využity. Druhá část této práce je věnována praktické stránce. Je rozdělena do 3 kapitol, kde kapitola 5 je věnována návrhu výsledné aplikace a jsou zde popsány klasifikační kategorie spolu s příznakovými vektory. V další kapitole 6 je detailně popsána implementační část celé práce a poslední kapitola 7 je věnována dosaženým výsledkům a porovnáním vhodnosti klasifikačních vektorů. Na úplném konci v kapitole 8 shrneme výsledky této práce a nastíníme další možné rozšíření a pokračování práce.

Kapitola 2

Radar

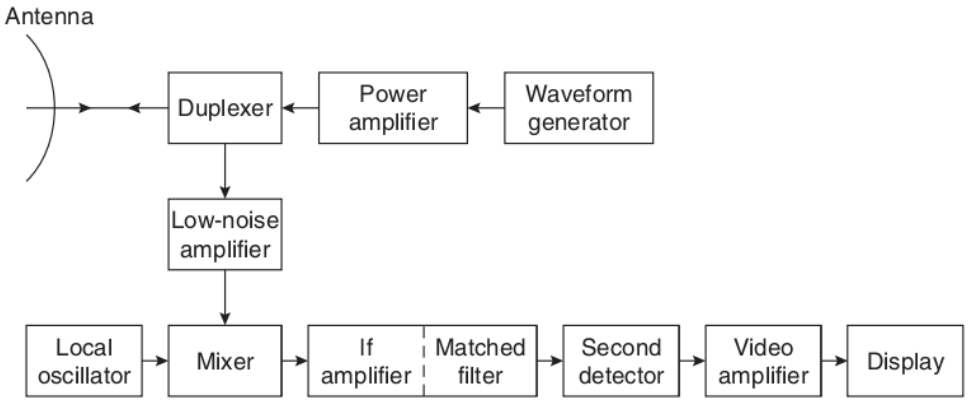
2.1 Principy funkce radaru

Pojem radar je anglický akronym pro Radio Detection and Ranging, což v překladu znamená rádiové rozpoznávání a zaměřování. Jedná se o elektronický přístroj k detekci objektů, určování jejich vzdáleností, rychlostí, směru pohybu a popřípadě jejich tvarů. Radar je založen na principu vysílání elektromagnetického vlnění s určitou frekvencí (tato frekvence se obvykle pohybuje v řádu jednotek MHz až stovek GHz). Konkrátně se jedná o speciální druh elektromagnetického vlnění označovaný jako *radarové vlnění*. Toto vlnění putuje prostorem rychlostí světla, tedy zhruba 299 792 458 metrů za sekundu. Radar tedy vysílá do prostoru vlnění, které putuje prostorem dokud nenarazí na překážku. V momentě styku radarové vlny a objektu je část vlnění pohlcena a další část je odražena zpět do prostoru. Množství odraženého a pohlceného vlnění závisí na typu materiálu a úhlu dopadu vlny. Odražené vlnění opět putuje prostorem zpět, kde jej může radar zachytit. Přijaté vlnění radarem je velice slabé z důvodu ztrát v průběhu putovaní prostorem a pak taky jednotlivými odrazy. Proto je obvyklou součástí radaru i zesilovač, jenž zesiluje přijatý signál pro pozdější zpracování.

Jak již bylo v úvodu řečeno, radar vysílá vlnění s určitou frekvencí. Tato frekvence je u radaru velice důležitá, jelikož frekvence odraženého signálu přijatá radarem může být od té vysílané odlišná. Na základě tohoto rozdílu frekvencí jsme schopni určit, zda se objekt pohybuje a popřípadě jakou má rychlost. Tomuto jevu se říká *Dopplerův jev*, jenž bude detailněji popsán v kapitole 2.2.

2.1.1 Z čeho se radar skládá

Radar je z pohledu elektrotechniky přístroj, který se skládá z určitých částí, kdy každá část má svou vlastní funkci. Pokud bychom se bavili o obecném radaru, tak ve většině případů zde nalezneme soubor společných částí. Samozřejmě jsou zde i radary, jež určeny na speciální aplikaci, které obsahují daleko více částí, ale i ty mají společné jádro. Právě toto jádro si dále detailněji popíšeme.



Obrázek 2.1: Obecné blokové schéma radaru s jednou anténou. V horní polovině se nachází část pro generování a vysílání signálu a v dolní polovině je část pro příjem a zpracování signálu. [14]

- **Vysílač** - tato část radaru se stará o generování vlnění, jenž radar následně vysílá. Součástí vysílače je i oscilátor, jenž generuje vlnění, které je dále zesilovačem zesíleno na požadovanou úroveň. Výkon vysílače se může měnit v závislosti na typu aplikace, ale běžně se pohybuje v řádech miliwattů, až megawattů.
- **Anténa** - tato součást je stěžejní část celého radaru. Jedná se totiž o prvek, která komunikuje s okolním světem. Antény se liší svým tvarem, který ovlivňuje jejich vlastnosti, a to zejména směrovost. Antény se dají rozdělit do dvou kategorií, a to na *vysílací antény* a *přijímací antény*. V mnoha typech radarů však oba tyto druhy antén tvoří anténa jediná, která je zároveň vysílačem i přijímačem.
- **Přepínač** - jak již bylo řečeno u antén je možné, aby jedna anténa plnila funkci vysílače i přijímače zároveň. Právě proto je součástí radaru přepínač, jenž se stará o to, aby v jeden okamžik procházel přes anténu signál z vysílače a po určité době, kdy se přepínač přepne, aby anténa přijímala signál směrem k přijímači. Přepínač tedy plní úlohu výhybky mezi přijímačem, anténou a vysílačem. Tuto součást tedy nalezneme jen u radaru s jednou anténou, což jsou ve většině případu radary pulzní.
- **Přijímač** - jakmile signál dorazí skrz anténu dostane se k přijímači. Přijatý signál bývá velice slabý, a proto je nutné jej zesílit pomocí zesilovače. Zároveň je velice důležité, aby přijímač odfiltroval jakýkoli šum, který by mohl zkreslit přijatá data. Součástí přijímače je i sekce pro zpracování signálu, v níž se například nachází *mixér*, jenž porovnává signál generovaný oscilátorem a signál přijatý. Na základě tohoto porovnání se následně rozhoduje, co se bude signálem dále provádět.

2.2 Dopplerův jev

Z principu radaru víme, že radar je zdrojem elektromagnetického vlnění, které se odráží od předmětu a vrací se zpět k přijímači. Pokud je jak radar, tak zkoumané těleso v klidu a oba mají nulovou rychlost je odražená frekvence rovna té vyslané. Pokud však mezi radarem a objektem nastane během vysílání vlnění pohyb, tak se při odrazu frekvence změní v závislosti na směru a rychlosti obou těles.

Tento princip odrazu vln popisuje *Dopplerův jev* [10]. Dopplerův jev byl objeven rakouským fyzikem Christianem Dopplerem v roce 1942. Tento jev popisuje co se děje při odrazu vln pokud je mezi zdrojem vlnění a jeho příjemcem relativní rychlost. Pojem relativní rychlost je v tomto případě důležitá, jelikož nezáleží zda se pohybuje zdroj, příjemce nebo snad oba zároveň.

Princip Dopplerova jevu se dá vysvětlit velice jednoduše. Pokud si představíme pohybující se objekt vysílající nebo odrážející jakékoli vlnění, tak můžeme říci, že čelo každé vlny jež je vysláno ve směru pohybu je vysláno z jiného místa než vlna předcházející. Z toho plyne, že vlna musí urazit kratší vzdálenost a tím pádem jsou vlny od sebe vzdáleny kratší vzdálenost než kdyby se těleso nepohybovalo. Stejně se dá popsat i opačný případ, kdy vlna putuje opačným směrem než objekt, jenž vlny vysílá. V tomto případě se ale vlny od sebe vzdalují. Praktický příkladem Dopplerova jevu je projíždějící sanitka. V momentě, kdy se sanitka se zapnutou sirénou přibližuje, je zvuk sirény slyšet s vyšší frekvencí. Pokud sanitka kolem nás projede a začne se vzdalovat, její zvuk uslyšíme s nižší frekvencí.

Z tohoto příkladu plyne, že odražené vlnění má větší frekvenci pokud se objekt přibližuje a má tedy kladnou rychlost a naopak odražené vlnění má nižší frekvenci pokud se objekt vzdaluje a má tedy zápornou rychlost. U tohoto jevu se často můžeme setkat s pojmem *Dopplerova frekvence*. Jedná se o rozdíl frekvencí, jenž vznikne mezi odeslanou a přijatou frekvencí právě kvůli odrazu vlnění s výskytem Dopplerova jevu.

Dopplerův jev lze popsat univerzální rovnicí jako

$$f = f_0 \cdot \frac{c + v_r}{c + c_s}, \tag{2.1}$$

kde f je hodnota odražené frekvence, f_0 je hodnota vysílané frekvence, c je rychlost pohybu vlnění v daném prostředí, v_r je rychlost pozorovatele a v_c je rychlost zdroje vlnění.

2.3 Druhy radarů

Radarová technologie se dělí na několik základních typů podle principu jejich fungování. V závislosti na našich požadavcích je pak možno vybrat si typ, který je vhodný pro naši aplikaci. Radary se dají rozdělit do dvou základních typů. Jsou jimi:

- **Primární radar** - jedná se o typ radaru, kde veškerou logiku řízení nese vysílač, který je zároveň přijímačem. Radar tedy vyšle signál s určitou frekvencí a výkonem do prostoru. Tento signál putuje prostorem dokud nenarazí na zkoumaný objekt. Od něj se původní vlnění odrazí a putuje zpět až dorazí opět k radaru. Radar zachytí tento signál svou přijímací anténou a následně jej zpracuje. V tomto případě je tedy zkoumaný objekt zcela pasivní a negeneruje žádné své vlnění.
- **Sekundární radar** - u tohoto typu radaru jsou aktivní, jak vysílací radar, tak i zkoumaný objekt. Můžeme si to představit tak, že vysílací objekt vyšle zakódovanou zprávu. Tento signál dorazí k objektu, který zprávu zachytí a rozkóduje. Na základě přijaté zprávy se zkoumaný objekt rozhodne jak na zprávu odpovědět a vyšle svou zakódovanou zprávy zpět směrem k radaru. Tento princip lze přirovnat k síťové komunikaci typu klient-server. Sekundární typy radaru tedy vyžadují aktivní spolupráci sledovaného cíle. S těmito typy radarů se můžeme setkat například v leteckém průmyslu, kde komunikují letecké základny s letadly ve vzduchu.

Radary se dále dělí podle principu, se kterým manipulují s vysílaným signálem. V tomto druhu dělení se můžeme setkat s následujícími typy radarů:

- **Pulzní radary** - tento typ radaru vysílá do prostoru vysokofrekvenční krátké impulzy s vysokým výkonem. Poté co radar vyšle impulz, ukončí vysílání a přepne se do módu, kdy čeká až k němu dorazí odražený signál, aby jej mohl zpracovat. Tento druh radaru tedy potřebuje ke svému chodu jen jedinou anténu na vysílání a přijímání signálu, jelikož nikdy tyto dvě činnosti nedělá současně. Tento typ radaru je vhodný na měření vzdáleností, jelikož jsme schopni změřit čas, který uplyne od vyslání pulzu až k jeho přijetí. V předchozí kapitole jsme si řekli, že radarové vlny se šíří rychlostí světla a tudíž už víme vše potřebné k vypočítání vzdálenosti. Vzdálenost je možno vypočítat jednoduchým vzorcem

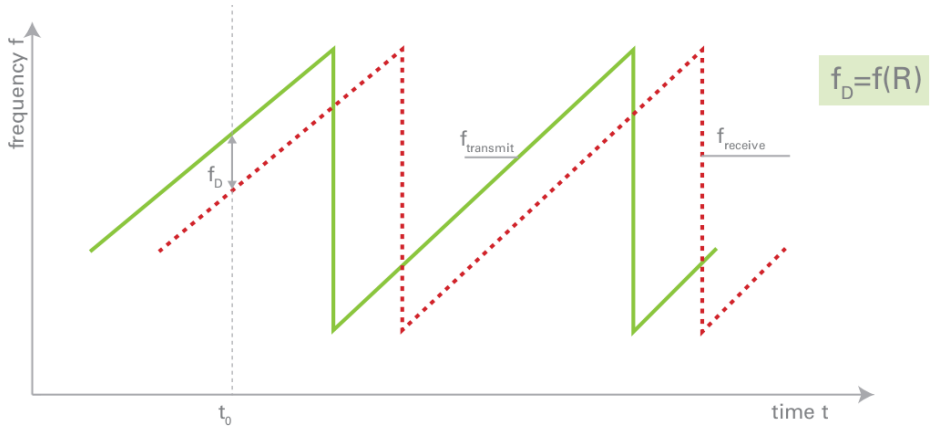
  $$s = \frac{t_\Delta \cdot v_c}{2}, \tag{2.2}$$

  kde t_Δ je právě rozdíl v čase mezi vyslaným a přijatým pulzem a v_c je zmiňovaná rychlost světla.

- **Radary s kontinuální vlnou** - tento typ radaru je mnohdy označován jako CW radar podle anglického *continuous wave*. Oproti předchozímu typu tento radar nevysílá krátké pulzy, ale vysílá signál nepřetržitě. Radar tedy vyžaduje více jak jednu anténu, jelikož jednou signál nepřetržitě vysílá a druhou signál přijímá. S tímto typem radaru lze snadno nepřetržitě měřit rychlost objektů pomoci Dopplerova jevu 2.2, ale bohužel není zcela jednoduché měření vzdálenosti. U pulzních radarů jsme měli schopnost měřit interval od vyslání po přijetí impulzu. Jelikož u radaru s kontinuální vlnou žádný impulz nevzniká, nedokážeme přesně říci, která přijatá část signálu odpovídá části odeslané.

  Existuje však podskupina radarů s kontinuální vlnou, jenž disponují funkcí frekvenční modulace, pomocí které jsme schopni měřit vzdálenosti i u těchto typů radarů. V principu se jedná o to, že radar nevysílá konstantní frekvenci, ale vysílá signál, který mění svůj tvar v průběhu času. Vyslaný a přijatý signál tedy vypadají přibližně stejně (nebudeme-li započítávat ztráty v průběhu přenosu a změnu frekvencí způsobenou Dopplerovým jevem) a jsme schopni z rozdílu frekvenčních hodnot získat zmiňovanou vzdálenost.

  Existuje řada způsobů, jak lze vysílaný signál modulovat. Pro ukázku uvedeme například **pilovou modulaci**.



Obrázek 2.2: Ukázka pilové modulace na vyslaný signál (zeleně značený signál) a časově posunutého přijatého signálu (červeně značený signál). [5]

Jedná se o lineární modulaci, kdy na začátku je vysílána minimální frekvence a s postupem času tato frekvence stoupá. Jakmile frekvence dosáhne maxima, opět spadne na minimální hodnotu a celý proces se opakuje. Přijatý signál má časový posun, který lze vidět na obrázku 2.2 a tím pádem jsme schopni v daném čase změřit rozdíl frekvencí odesílaného a přijatého signálu. Tento rozdíl frekvencí je značen jako f_D a mnohdy je nazýván jako *beat frequency*. Z této rozdílové frekvence jsme následně schopni vypočítat vzdálenost objektu následujícím vzorcem

$$R = \frac{c_0}{2} \cdot T \cdot \frac{f_D}{\Delta f}, \tag{2.3}$$

kde R je vzdálenost objektu od radaru, c_0 je rychlost světla, T je modulační perioda, f_D je zmiňovaný rozdíl ve frekvenci a Δf je hloubka modulace.

2.4 Konkrétní typy radarových modulů

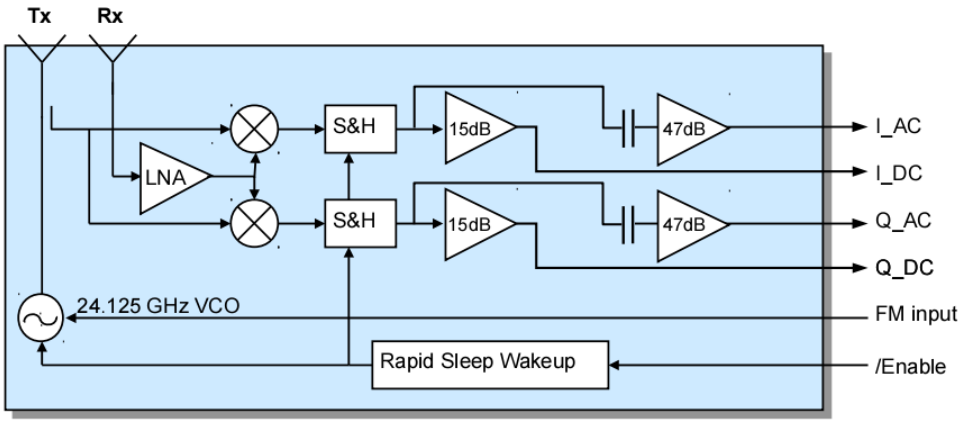
K účelům měření a získávání reálných dat byly k této práci zapůjčeny radarové moduly společnosti RFBeam Microwave GmbH. Konkrétně se jedná o modely K-MC1 a K-MC4. Oba tyto modely jsou radary s kontinuální vlnou, takže pro měření vzdálenosti je potřeba použít frekvenční modulace. Rovněž se vyznačují svými malými rozměry, nízkým příkonem a přijatelnou cenou 150, respektive 250 euro. K samotným radarovým modulům je přidán i komunikační modul, jenž se stará o zpracování signálu a zasílání vyhodnocených dat pomocí síťového rozhraní ve formě paketu do počítače.

Zde si detailněji popíšeme vlastnosti a vnitřní schéma jednotlivých modulů.

2.4.1 K-MC1

Tento modul obsahuje jednu vysílací a jednu přijímací anténu. Obě tyto antény jsou tvořeny 30 ploškami na povrchu modulu. Vysílací frekvence radaru je 24GHz, kterou lze modulovat pomocí vstupního pinu označeného jako FM input. Uplatnění tohoto radaru je zejména v monitorování dopravy a měření rychlosti objektu.

Radar přijatý signál zpracovává pomocí Dopplerova jevu a odečítá vysílaný signál od přijatého. Výstupem je tedy Dopplerova frekvence v podobě signálu I. Jelikož frekvence nemůže být v tomto případě záporná, nemůžeme určit, zda se objekt pohybuje směrem k radaru nebo směrem od něj. Proto je zde i druhý výstup Q, jenž je v ideálním případě posunut o π nebo o $-\pi$, podle směru pohybu objektu. Dohromady tyto signály můžeme považovat za komplexní hodnotu. Výstupy I a Q jsou zde zdvojeny a jedna jejich dvojice je zesílená tak, aby bylo možné zpracovávat frekvence s nízkou intenzitou.

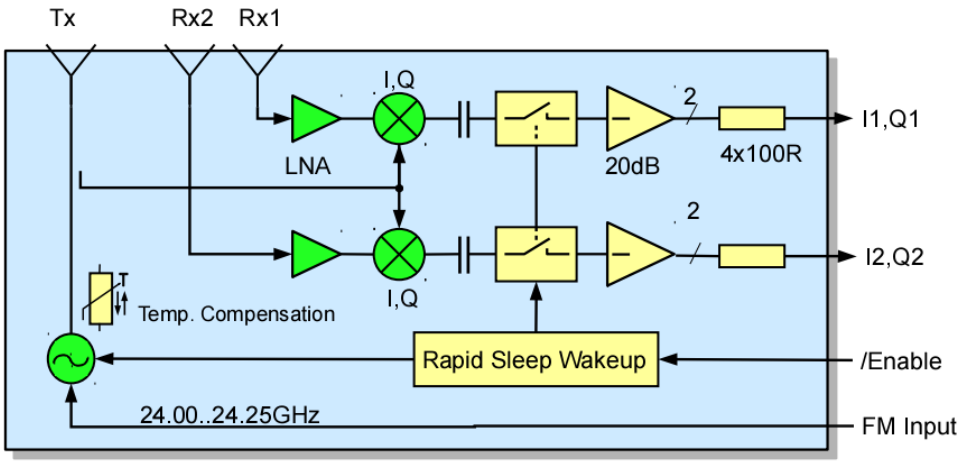


Obrázek 2.3: Blokové schéma radarového modulu K-MC1. [12]

Tento modul je vybaven funkcí *Rapid Sleep Wakepu*, která dokáže šetřit spotřebu radaru až o 90% v momentě, kdy se na snímané scéně nic neděje. Tato funkce se nejvíce hodí v aplikacích, kdy je radar napájen z baterií.

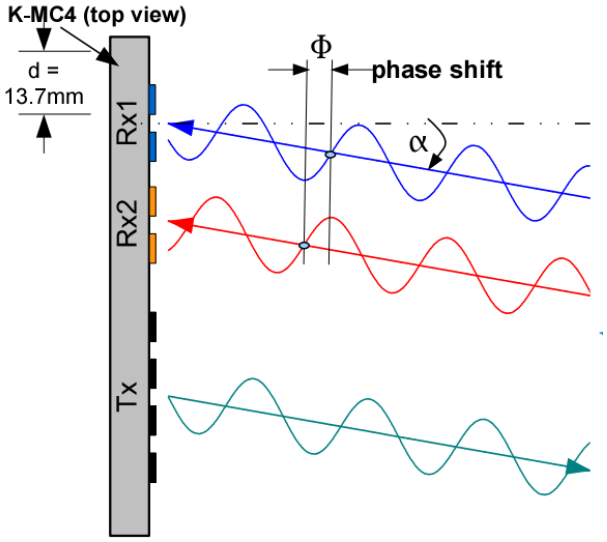
2.4.2 K-MC4

Tento modul víceméně sdílí většinu vlastností s modelem předchozím. Rozdíl je však v počtu přijímacích antén a jejich využití. Radar obsahuje jednu vysílací anténu a dvě přijímací antény. Tyto dvě antény nám umožňují detekovat úhel, pod kterým dopadl přijatý signál na radar. Z obrázku 2.5 lze vidět, že signál dopadající na antény pod úhlem α, dopadá na druhou přijímací anténu s lehkým fázovým posunem oproti první anténě. Na základě tohoto fázového posunu jsme tedy schopni zjistit dopadající úhel.

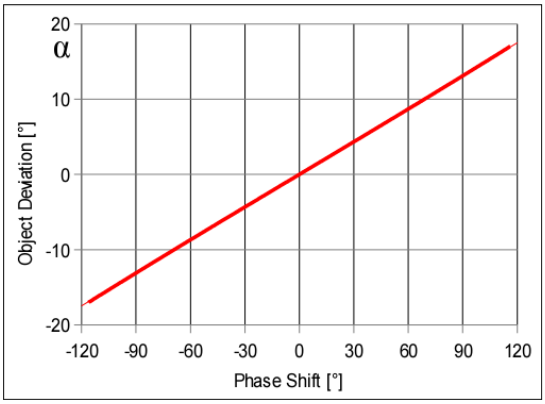


Obrázek 2.4: Blokové schéma radarového modulu K-MC4. [13]

Modul opět obsahuje výstup I, který produkuje Dopplerovu frekvenci a v závislosti na směru pohybu je fázově posunut výstup Q. I zde jsou výstupy zdvojeny, avšak oproti předchozímu modelu jsou zdvojené výstupy použity právě pro možnost měření úhlu. Zdvojené výstupy náleží dvěma přijímacím anténám, jež jsou vůči sobě fázově posunuty a tento fázový posun lze pomocí funkce zobrazené na obrázku 2.6 převést na úhel dopadajícího signálu.



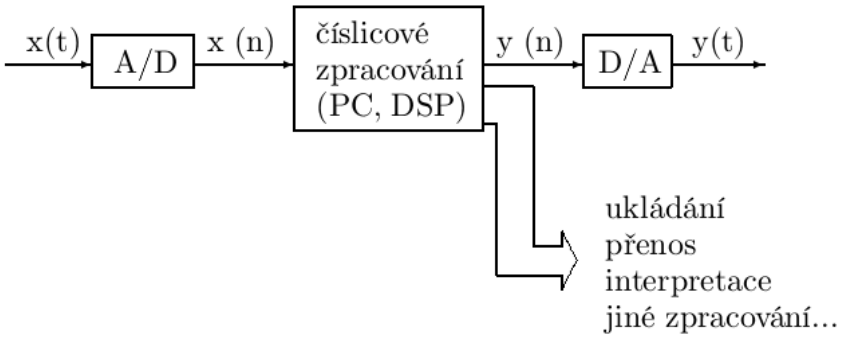
Obrázek 2.5: Ukázka dopadu signálu na obě antény s fázovým posunem závislým na úhlu dopadu. [13]



Obrázek 2.6: Funkce značící převod fázového posunu mezi anténami na úhel dopadajícího signálu. [13]

Kapitola 3

Zpracování signálu



Obrázek 3.1: Obecné schéma systému, který zpracovává signál. [18]

3.1 Převod na digitální signál

Elektromagnetické vlnění je spojitá analogová veličina, jež se dá pomocí přijímacích antén převést na elektrický analogový signál. Avšak dnešní číslicové systémy pracují s digitálním signálem v podobě informací nesených pomocí zakódovaných bitů. Abychom tedy mohli dále pracovat se signálem v podobě bitů, je potřeba jej nejdříve převézt z analogové podoby na digitální. Tento proces se nazývá analogově digitální převod a je prováděn za pomoci elektronických obvodů, takzvaných AD převodníků. Samotný převod je v elektronických zařízeních poměrně běžný, a proto vznikla řada druhů AD převodníků pracujících s různými metodami převodu.

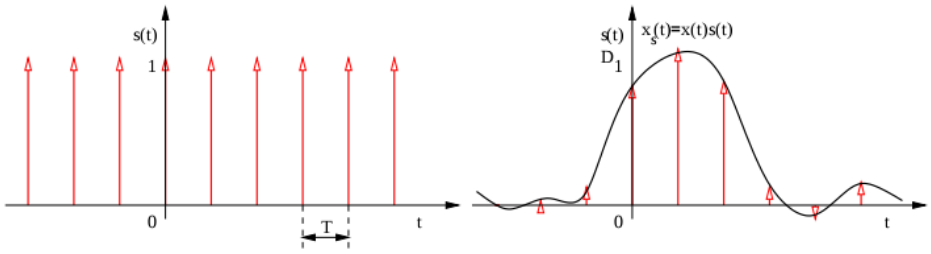
- **Aproximační převodník** - tento převodník se skládá z napěťového komparátoru, digitálně analogového převodníku a aproximačního registru. Při startu převodu je nastaven nejvyšší bit v registru a následně je tato hodnota převedena za pomoci DA převodníku na analogový signál. Tento analogový signál je porovnán se vstupním napětím a na základě tohoto porovnání je bit v registru uchován nebo vynulován. Takto se postupně nastavují všechny bity v registru od MSB až k LSB. Tímto postupem se hodnota v registru čím dál tím více přibližuje skutečné hodnotě vstupního signálu. Doba převodu závisí na počtu bitů registru. Pro N bitový registr je potřeba vykonat N kroků.

- **Převodník s čítačem** - tento typ převodníku je velice podobný tomu předcházejícímu. Rozdílem oproti předchozímu typu je, že aproximační registr je nahrazen čítačem. Tento čítač postupně čítá od 0 až po maximální hodnotu a jakmile komparátor zjistí, že hodnota čítače je větší než hodnota vstupního signálu, ukončí se převod a digitální hodnota je uložena v čítači. Tento převodník je časově náročnější, jelikož jeho maximální počet cyklů může dosáhnout až N^2 kroků, kde N je počet bitů čítače.
- **Integrační převodník** - tento převodník funguje na bázi dvojího převodu. A to sice z vstupního napětí na frekvenci a následně z frekvence na digitální podobu vstupního signálu. V první části se pomocí integračního článku generuje napětí pilového tvaru s frekvencí odpovídající vstupnímu napětí. Integrační článek postupně zvyšuje napětí až dosáhne určité hranice a následně napětí klesne znovu na nulu. Tímto je generováno napětí o určité frekvenci, kterou jsme schopni převést pomocí čítače na digitální hodnotu. Převodníky tohoto typu jsou velice přesné, a to díky dobrým vlastnostem operačních zesilovačů, jenž jsou použity jako integrační články.
- **Paralelní převodník** - anglicky se tento převodník označuje jako Flash, tedy rychlý. Je to z toho důvodu, že zde není třeba čekat až se vykoná X cyklů. Celý princip spočívá v sérii rezistorů, jež tvoří napěťový dělič pro referenční napětí. Mezi-výstupy tohoto děliče jsou přivedeny na napěťový komparátor se vstupním napětím. Čím větší je vstupní napětí, tím více komparátorů je sepnuto. Na základě výstupu z komparátorů je pomocí dekodéru určena výstupní digitální hodnota.

Samotný převod na digitální formu signálu se skládá ze dvou hlavních kroků. Jsou jimi vzorkování a kvantování. Tyto metody jsou podrobněji rozepsány ve vlastních podkapitolách.

3.1.1 Vzorkování

Vzorkování je první z kroků převodu z analogové podoby signálu na digitální. Účelem vzorkování je získat ze spojitého signálu, který tvoří nekonečně mnoho hodnot, signál diskrétní. Toho dosáhneme tím, že původní signál vynásobíme signálem, jenž je tvořen sledem obdélníkových impulzů s konstantní periodou. Pro tento účel se používá takzvaný *Dirakův impulz*. Jedná se o signál obdélníkového tvaru, jenž má nekonečně velkou hodnotu amplitudy a zároveň je nekonečně úzký a jeho délka se blíží k nule. Tímto je zaručeno, že jeho obsah a tedy mocnost je rovna jedné a nedochází k žádným ztrátám po vynásobení původního signálu. Výsledkem násobení je opět sled Dirakových impulzů, ale tentokrát s velikostí původního signálu.



Obrázek 3.2: Násobení signálu sledem Diracových impulzů. [18]

Z konstantní periody Dirakových impulzů, jež se označuje jako T, dokážeme určit takzvanou vzorkovací frekvenci F_s, která je dána následujícím vztahem:

$$F_s = \frac{1}{T} \tag{3.1}$$

Tato hodnota udává kolikrát za sekundu je vstupní signál vzorkován a tedy kolik diskrétních hodnot získáme.

Vzorkovací frekvence je důležitá i z pohledu korektního zachování všech frekvencí v signálu. Vztah vzorkovací a nejvyšší frekvence určuje *Shannonův–Kotelnikovův–Nyquistův vzorkovací teorém*. Tento teorém říká, že pro korektní zachování původního signálu je potřeba, aby vzorkovací frekvence byla minimálně 2 krát vyšší než maximální frekvence vzorkovaného signálu.

$$F_s > 2f_{max} \tag{3.2}$$

Za předpokladu, že víme předchozí rovnici, mohou nastat tyto dvě situace:

- $F_s \leq 2f_{max}$ - Jednotlivé kopie spektra původního signálu se překrývají, a tudíž není možné rekonstruovat původní signál bez toho, aniž by docházelo k **antialiasingu**.
- $F_s > 2f_{max}$ - Jednotlivé kopie spektra původního signálu se nepřekrývají, a tudíž je možno rekonstruovat původní signál tak, že aplikujeme filtr dolní propusti s hranicí $F_s/2$.

3.1.2 Kvantování

Na základě vzorkování jsme získali hodnoty diskrétní v čase, ale pro digitální použití to stále nestačí, neboť signál může nabývat velkého množství hodnot. V počítačovém zpracování se setkáváme s omezením paměťového prostoru pro uložení konkrétní hodnoty. Nejčastěji tuto velikost určuje počet bitů. Klasicky se tedy setkáváme s N bitovými proměnnými, které mají omezený počet hodnot. Pokud bychom to chtěli ukázat na příkladu, tak si vezmeme v úvahu vstupní napětí 5V, které je přivedené na 8 bitový analogově digitální převodník. Osm bitů dokáže zakódovat 2^8 hodnot, čili 256. Vstupních 5 voltů je tedy rozděleno do 256 hodnot, jež jsme schopni zakódovat v digitální podobě a uložit do paměti. Právě tento proces se jmenuje kvantování. Plynule tedy navazuje na vzorkovací část analogově digitálního převodníku a jednoduše řečeno plní úlohu zaokrouhlování výstupu hodnot ze vzorkovací části na kvantizační hladiny. Kvantizační hladina je pojem, jenž označuje konkrétní hodnoty, na které jsme schopni číslo zakódovat. Rozdíl mezi jednotlivými kvantizačními hodnotami se nazývá rozlišovací schopnost a v předchozím příkladě je to 0.195V. V našem zájmu je, aby rozlišovací schopnost byla co nejnižší, jelikož jen tak zajistíme, aby chyba vzniklá při kvantizaci byla co nejmenší. Tato chyba je označována jako kvantizační šum a je tvořena rozdílem mezi skutečnou hodnotou signálu a její zakódovanou hodnotou.

Další možnost zanesení chyby je ta, že dojde k přesaturování. Toto může nastat v situaci, kdy je nastavena maximální kvantizační hodnota a vstupní signál ji převyšuje. V tomto okamžiku nejsme schopni rozlišit o jak velkou odchylku se jedná a tím se zvětšuje kvantizační šum.

3.2 Odstranění stejnosměrné složky

Stejnosměrná složka je část signálu, která se do něj dostává nejčastěji nedokonalostí AD převodníků. Jedná se o složku signálu, která celý signál posouvá nad osu x do kladných hodnot.

Stejnosměrná složka je sice nedílnou součástí signálů, avšak je v mnohých aplikacích nepotřebná, ba dokonce nežádoucí. V pozdějších fázích zpracování signálu by tato složka zkreslovala hodnoty spektra, ze kterých budeme čerpat největší část informací v této aplikaci.

Stejnosměrná složka by se dala jednoduše popsat jako průměr hodnot v určité oblasti, avšak ne v každé aplikaci je to takto jednoduché. Odstranění stejnosměrné složky se provádí prostým odečtením od každého vzorku v signálu. Tento proces popisuje následující rovnice

$$s'[n] = s[n] - \mu_s, \tag{3.3}$$

kde $s[n]$ je hodnota signálu obsahující stejnosměrnou složku a μ_s je hodnota stejnosměrné složky.

Jak bylo popsáno výše, ne vždy je výpočet hodnoty stejnosměrné složky prováděn pouhým průměrováním hodnot. Střední hodnotu je možno počítat dvěma způsoby:

- **Střední hodnota off-line** - jedná se o metodu, kdy dopředu víme všechny hodnoty signálu a můžeme střední hodnotu odhadnout průměrováním všech hodnot signálu. Střední hodnotu lze tedy vypočítat jako

$$\mu_s = \frac{1}{N} \cdot \sum_{n=1}^{N} s[n] \tag{3.4}$$

- **Střední hodnota on-line** - jedná se o metodu, kdy dopředu buďto nevíme všechny hodnoty signálu a nebo je signál příliš dlouhý. Střední hodnotu v tomto případě musíme odhadovat rekurzivně z předchozích vzorků například pomoci následujících rovnic

$$\mu_s[n] = \gamma\mu_s[n-1] + (1-\gamma)s[n] \tag{3.5}$$

  V této rovnici je na nás jakou hodnotu γ použijeme. Pokud bychom ji směřovali k hodnotě jedna, tak se jedná o filtr s impulzní odezvou

$$h = [(1-\gamma)(1-\gamma)\gamma(1-\gamma)\gamma^2 \dots] \tag{3.6}$$

3.3 Rozdělení na rámce

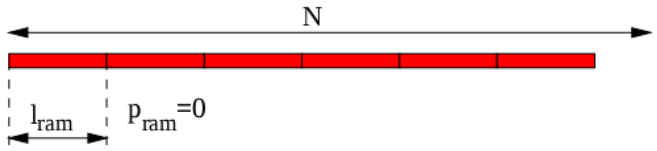
Pro pozdější zpracování signálu není vhodné pracovat se signály dlouhé délky, jelikož mohou obsahovat více probíhajících dějů a nemusí být stacionární. Proto je nutné použít metodu, která rozdělí signál na menší části, takzvané rámce. Délka rámce se volí tak, abychom mohli o signálu říct, že je stacionární a nekolísá například při změně rychlosti automobilu. Na druhou stranu je pro nás žádoucí, aby rámec měl co možná nejvíce vzorků proto, abychom získali co největší přesnost po aplikaci Fourierovy transformace a získaní spektra. Toto téma bude podrobněji popsáno v kapitole 3.4.1.

Existují dva způsoby, jak signál na rámce rozdělit:

- **Rámce bez překrytí** - vstupní signál je rozdělen tak, že konec každého rámce je začátkem rámce následujícího. V tomto případě jediné co je potřeba je samotná délka rámce, kterou označíme jako l_{ram}. Pokud nezpracováváme signál v reálném čase, je možné určit počet rámců, které ze signálu získáme. Počet rámců je dán následující rovnicí:

$$N_{ram} = \left[\frac{N}{l_{ram}} \right], \tag{3.7}$$

kde N označuje počet vzorků signálu.

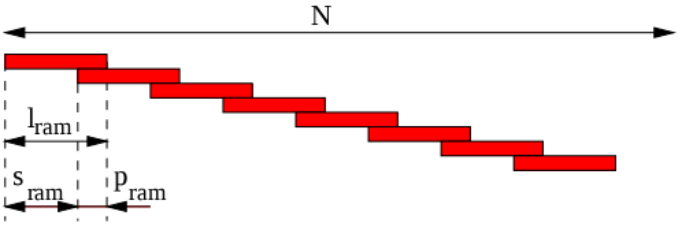


Obrázek 3.3: Rozdělení signálu na rámce bez překrytí. [18]

Tato metoda zajišťuje rychlý časový posun, malé nároky na paměť, ale hodnoty mezi jednotlivými rámci se mohou výrazně lišit a nebo může nastat situace, kdy rozpůlíme rámce v momentě, kdy nás signál zajímá nejvíce.

- **Rámce s překrytím** - tato metoda oproti předchozí variantě nerozděluje rámce jeden za druhým, ale přidává mezi rámce překryv. Zde je tedy nutno přidat další dva parametry, které určují délku překryvu a délku rámce bez překryvu. Budeme je dále značit jako p_{ram} a s_{ram}. Výpočet počtu rámců je zde trochu složitější než u předchozí varianty a je dán následujícím vztahem:

$$N_{ram} = \left[\frac{N - l_{ram}}{s_{ram}} \right] \tag{3.8}$$



Obrázek 3.4: Rozdělení signálu na rámce s překrytím. [18]

Tato metoda zajišťuje pomalý časový posun a větší nároky na paměť a procesor. V závislosti na zvolených parametrech se může stát, že si rámce budou velice podobné, a tudíž bychom zpracovávali vícekrát hodně podobný signál.

Z výše uvedeného je patrné, že každá varianta má své klady a zápory. Velikou roli hraje také nastavení parametrů délek rámců a podobně. Nedá se tedy jednoznačně říct, která metoda bude vhodná a jaké parametry budou nejlepší. Vše je tedy na bázi experimentování, jelikož každá aplikace a její vstupní signály vyžadují různé zpracování.

3.3.1 Okénkové funkce

Pro získávání rámců z původního signálu se používají okénkové funkce. Jsou to signály různého tvaru, kterými se násobí původní signál. Po této operaci nám zbude právě jeden rámec, se kterým můžeme dále pracovat. I zde se můžeme setkat s celou řadou různých okénkových funkcí. Za zmínku stojí dvě nejpoužívanější:

- **Pravoúhlé okno** - jedná se o okno definované jako

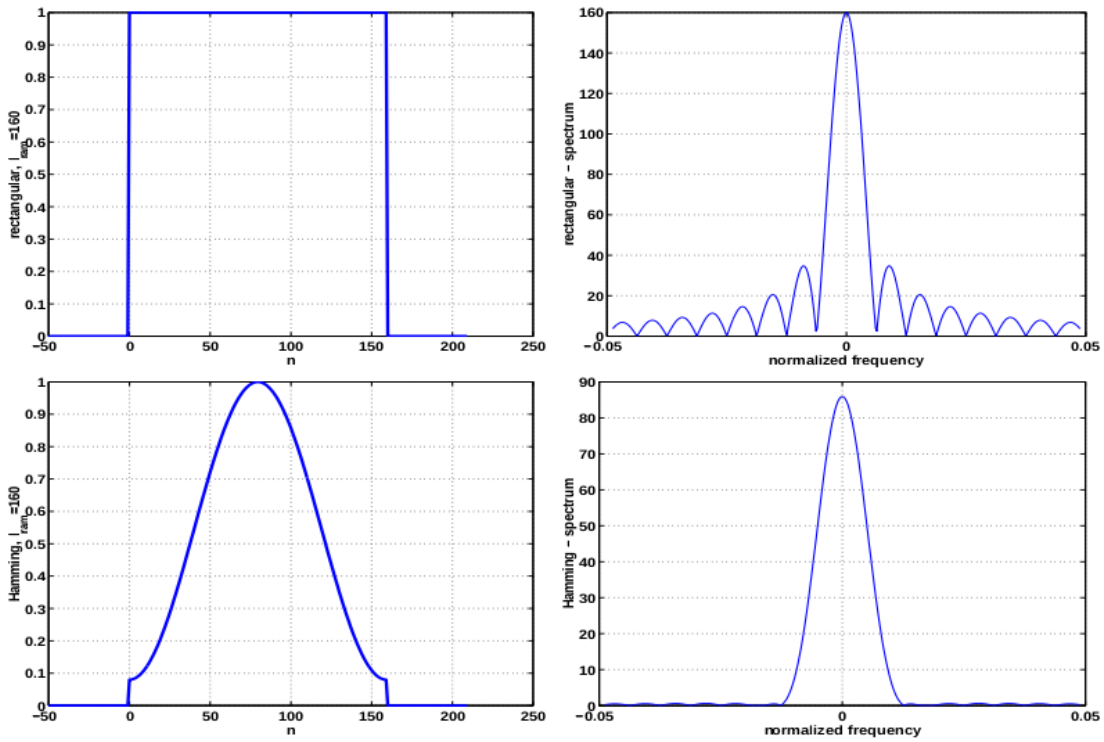
$$w[n] = \begin{cases} 1 & pro \quad 0 \leq n \leq l_{ram} - 1 \\ 0 & jinde \end{cases} \tag{3.9}$$

  Tato funkce vykrojí ze signálu přesně danou část od začátku rámce až po jeho konec bez jakýchkoli útlumů. Nevýhodou tohoto okna je, že do spektra signálu zanese rušivé vysoké frekvence, které nejsou žádoucí. Je to zapříčiněno tím, že okraje signálu jsou prudce useknuty a tyto oblasti jsou právě příčinou vysokých frekvencí.

- **Hammingovo okno** - jedná se o okno definované jako

$$w[n] = \begin{cases} 0.54 - 0.46cos(\frac{2\pi n}{l_{ram}-1}) & pro \quad 0 \leq n \leq l_{ram} - 1 \\ 0 & jinde \end{cases} \tag{3.10}$$

  Tato funkce propustí největší část signálu ve středu okna, zatímco signál na začátku a konce okna utlumí. Tímto se zbavíme velké nevýhody předchozí funkce a v signálu se zbavíme nežádoucích vysokých frekvencí.



Obrázek 3.5: Obrázek znázorňuje tvar pravoúhlého a Hammingova okna a jejich vliv na spektrum vykrojeného okna. [18]

Tím, že násobíme signál jakoukoli variantou okénkové funce, měníme výsledné spektrum daného rámce. Násobení signálu oknem v časové oblasti totiž odpovídá konvoluci spektra signálu a spektra okénkové funkce.

$$X(f) = S(f) \cdot W(f), \tag{3.11}$$

kde $S(f)$ je spektrum signálu a $W(f)$ je spektrum okna.

3.4 Frekvenční analýza

Jak jsme si již v předchozích částech řekli, hlavní části informace, které nás budou zajímat jsou ve frekvenční oblasti. Proto potřebujeme převést signál, který je zatím jen jako funkce fyzikální veličiny (v našem případě intenzita signálu převedená AD převodníkem na napětí) závislá na čase, na funkci ve frekvenční doméně. Toho lze docílit metodou **Fourierovy transformace**[11]. Tato metoda popisuje, že každý signál lze převést na sérii sinových a kosinových funkcí s různou amplitudou a fázovým posunem. Tato série se nazývá Fourierova řada a rovnicí lze vyjádřit jako

$$f(t) \sim \frac{a_0}{2} + \sum_{k=1}^{\infty}[a_k \cdot cos(k \cdot t) + b_k \cdot sin(k \cdot t)], \tag{3.12}$$

kde a_0 je právě hodnota stejnosměrné složky a a_k a b_k jsou koeficienty, které se vyjádří následujícími rovnicemi

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x) \cdot cos(k \cdot x)dx, \quad k \ = 0, 1, 2, \ldots \tag{3.13}$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x) \cdot sin(k \cdot x)dx, \quad k \ = 1, 2, \ldots \tag{3.14}$$

3.4.1 Rychlá Fourierova transformace

Fourierova transformace má řadu variant. Nás však bude zajímat metoda pro diskrétní signál, a tedy metoda DFT neboli **Diskrétní Fourierova transformace**. Tato metoda má kvadratickou složitost $O(n^2)$, a proto pro její urychlení vznikla metoda FFT, což je **Rychlá Fourierova transformace**, neboli Fast Fourier transform. Tato metoda má řadu implementací, avšak mnoho z nich funguje na principu faktorizace N, a proto má složitost linearitmickou $O(n\ log\ n)$. FFT tedy produkuje stejné výsledky, ale za daleko kratší dobu.

Výsledkem Fourierovy transformace je spektrum, jež vyjadřuje hodnoty intenzit pro jednotlivé frekvence zastoupené v signálu. Spektrum u diskrétních signálů velice závisí na vzorkovací frekvenci F_s, jelikož obsahuje celkem N vzorků, kde hodnota N je rovna hodnotě vzorkovací frekvence. Z této informace jsme schopni odvodit, že spektrum diskrétního signálu je periodické a opakuje se každých N vzorků a navíc je symetrické oproti vzorkovací frekvenci. Zde narážíme na vzorkovací teorém a tudíž víme, že frekvence pod $\frac{F_s}{2}$ budou ve spektru zachovány zatímco frekvence vyšší poruší spektrum a signál nebude možno rekonstruovat korektně zpět.

Samotný výpočet spektra lze provést podle následující rovnice

$$X(K) = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi\frac{nK}{N}} \quad pro \quad k \ \in < 0, N-1 >, \tag{3.15}$$

kde se pro každou hodnotu spektra dělá suma všech vzorků diskrétního signálu násobeného komplexní exponenciálou s parametry závislými na aktuálním vzorku.

3.5 Zero padding

V sekci 3.4 jsme si řekli, že rozlišení ve spektrální doméně je přímo závislé na množství vzorků ve vyříznutém rámci a na vzorkovací frekvenci. Někdy je žádoucí toto rozlišení alespoň pomocí interpolace zvětšit tak, abychom dosáhli delšího výstupního vektoru FFT. Proto abychom tedy mohli zvětšit rozlišení ve frekvenční oblasti, je nutno změnit jednu z výše uvedených veličin. Vzorkovací frekvence je většinou pevně daná a délku rámce taky není možné měnit ve všech případech, jelikož chceme zachovat statické hodnoty signálu. Právě proto je zde metoda Zero padding, která zvyšuje rozlišení ve frekvenci a zároveň nepřidává žáden šum do výsledného spektra. Metoda spočívá v tom, že se rámec nadstaví vzorky s hodnotami nula. Touto operací tedy prodloužíme velikost rámce, aniž bychom poškodili signál či zanesli do výsledku jakoukoli chybu. Jediné co tedy získáme je detailnější rozlišení ve frekvenční ose.

Kapitola 4

Klasifikace

4.1 Co je to klasifikace a její typy

Klasifikace je jednou z forem strojového učení, což je v dnešní době velice populární téma. Klasifikace slouží k tomu, aby nám určila do jaké třídy patří konkrétní objekt, veličina či cokoli jiného, co se dá matematicky popsat. Pro popis zmiňovaného objektu slouží takzvaný *příznakový vektor*. Tento vektor je bodem v N-rozměrném prostoru zvaném *feature space*. Velikost příznakového vektoru přímo ovlivňuje v kolika rozměrném prostoru se budeme pohybovat.

Způsob extrakce příznakového vektoru je velice důležitý pro pozdější klasifikaci, jelikož vektory stejné kategorie by měly být v N-rozměrném prostoru blízko u sebe, zatímco vektory z jiné třídy by měly být od sebe vzdálené. Nesprávně zvolená extrakce příznakového vektoru může způsobit to, že jednotlivé třídy od sebe nepůjdou rozeznat, a tudíž budou špatně klasifikovány.

Proces klasifikace je tedy rozdělení N-rozměrného prostoru tak, aby jednotlivé hranice rozdělovaly prostor na základě jednotlivých tříd. Součástí klasifikace jsou dva procesy. Prvním z nich je samotné natrénování klasifikátoru. K tomu slouží množina *trénovacích dat*. Jedná se o zhruba 2/3 dat z celkové dostupné množiny. Na trénovacích datech se metoda klasifikace snaží najít takové dělení prostoru, aby chyba byla co nejmenší. Pak je zde i druhá množina dat nazývaná jako *testovací data*. Obě tyto množiny by měly být disjunktní tak, aby se vzájemně neovlivňovaly a chyba na jedné množině nezpůsobovala chyby na druhé množině. Testovací data slouží k vyhodnocení natrénovaného modelu a to tak, že jsou předložena natrénovanému klasifikátoru a ten je podle svého rozdělení umístí do patřičné třídy.

Učení s učitelem

Metoda učení s učitelem je jednou z hojně využívaných technik strojového učení. U této techniky máme k dispozici množinu trénovacích dat. Tato množina se skládá z příznakového vektoru, který popisuje daný objekt v příznakovém prostoru. Společně s těmito daty je ke každému příznakovému vektoru k dispozici informace o tom do jaké kategorie patří pokud se zaměříme na problematiku klasifikace. Máme tedy k dispozici množinu dvojic $(data_i, label_i)$. Algoritmy založené na klasifikaci metodou učení s učitelem tedy dopředu ví, jak data rozdělit do skupin (zároveň dopředu ví na kolik tříd množinu dat rozdělit) a nemusí si v datech hledat určité seskupení samo.

V druhé části, kdy máme klasifikátor natrénovaný a víme do kolika skupin jsou data rozdělena, můžeme použít klasifikátor na reálných datech. Pokud tedy vložíme libovolný

příznakový vektor do N-dimenzionálního prostoru, snadno získáme sub-prostor a tedy třídu, do které daný vektor patří.

Učení bez učitele

U metod učení s učitelem bylo hlavním cílem naučit se mapovat vstupní hodnoty na výstupní s přispěním někoho, kdo znal správnou výstupní hodnotu. U metod učení bez učitele [1] zde není nikdo takový, kdo by znal správnou hodnotu výstupu a tudíž jediné co víme je množina vstupních hodnot. Cílem je nalézt jakoukoli pravidelnost ve vstupních datech. V takovýchto datech se mohou objevovat struktury s určitým vzorem, které se vyskytují více než jiné. Našim cílem je prozkoumat vstupní data a zjistit, co se v nich děje a co ne. Ve statistice se toto nazývá *density estimation*. Jedna z metod density estimation je metoda zvaná *clustering*, kde je hlavním cílem nalézt ve vstupních datech clustery neboli shluky.

Metodu učení bez učitele využívá řada algoritmů. Ty nejznámější je možno vidět zde:

- **clustering**
  - k-means
  - mixture models
  - hierarchický clustering
- **detekce anomálií**
- **neuronové sítě**

4.2 Support Vector Machine

Support Vector Machine [6], zkratkovitě známý jako SVM, je jedna z nejpopulárnějších metod v moderním oboru strojového učení. Tento algoritmus byl poprvé představen Vladimírem Vapnikem v roce 1992. Od té doby se jeho využívání rapidně zvedlo díky mnohdy lepším výsledkům než mají jiné algoritmy strojového učení na rozumně velkých množinách dat. Support Vector Machine je algoritmus využívající principu učení s učitelem, tudíž je mu nutno předat množinu dat s jejími popisy na natrénování.

Tato metoda je založena na nalezení hyperplochy v N-rozměrném prostoru (ve 2D prostoru se jedná o přímku) tak, aby oddělovala množiny bodů na zcela disjunktní podmnožiny. Kromě této hyperplochy se algoritmus snaží najít co možná největší okraj od přímky, ve kterém neleží žáden datový bod. Čím větší se nám podaří nalézt tento okraj, tím lepších výsledků můžeme dosáhnout. Body, jež leží nejblíže hyperplochy se označují jako *support vector*.

Implementaci tohoho algoritmu si můžeme ukázat v jednoduchém 2D prostoru, kde hranici mezi třídami bude tvořit přímka. Tuto přímku můžeme vyjádřit následující rovnicí.

$$y = w \cdot x + b, \tag{4.1}$$

kde w je váhový vektor, x je vstupní vektor a b je hodnota zvaná jako bias, která posouvá tuto přímku.

Body ležící nad touto přímkou můžeme označit jako + a rovnicí je lze vyjádřit jako

$$w \cdot x + b > 0 \tag{4.2}$$

naopak body ležící pod přímkou lze označit jako − a rovnicí je lze vyjádřit jako

$$w \cdot x + b < 0 \tag{4.3}$$

Jelikož algoritmus SVM nehledá jen body nad a pod přímkou, musíme do obou rovnic započítat i již zmíněný okraj **M**. Obě rovnice tedy musíme upravit do tvaru

$$w \cdot x + b \geq +M \tag{4.4}$$

$$w \cdot x + b \leq -M \tag{4.5}$$

Pokud získáme bod z množiny x^+, který se nachází nejblíže hledané rozdělovací přímky, získáme zmiňovaný support vector. Opačný support vector z množiny x^-, je opět vzdálený od dělící přímky ve vzdálenosti **M**. Z toho vyplývá, že vzdálenost mezi support vectory z množin bodů x^+ a x^- je přesně 2M. Na základě těchto informací můžeme říct, že vektor w je kolmý k separační přímce a zároveň i k linii okrajů hranic + a −. Tento vztah můžeme zapsat následující rovnicí

$$x^- = x^+ + \nu \cdot w \tag{4.6}$$

Víme, že $|x^- - x^+| = 2M$, takže pomocí rovnice 4.6 můžeme odvodit následující rovnici

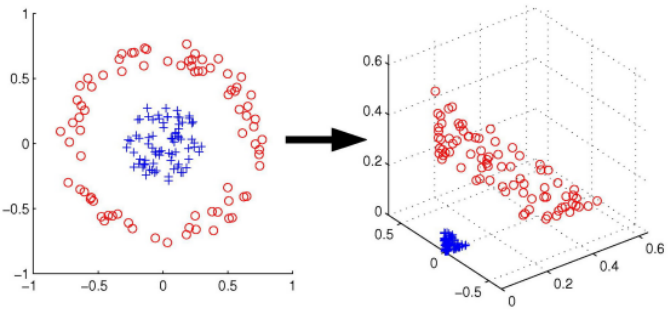
$$M = \frac{1}{2|w|} = \frac{1}{2\sqrt{w \cdot w}} \tag{4.7}$$

Nyní když zníme rovnici klasifikační přímky, můžeme spočítat velikost okraje M. Snažíme se najít takový vektor přímky w a velikost b tak, aby velikost okraje M byla co největší. Z rovnice 4.7 plyne, že největší velikost M získáme tak, že hodnota $w \cdot w$ bude co nejmenší. Pokud by tohle byla jediná podmínka, tak nejjednodušší by bylo položit $w = 0$ a problém by byl vyřešen. Bohužel však tohle není jediná podmínka. Další důležitou podmínkou je nastavit w tak, aby rozdělovala množinu bodu na + a −. Nastává tedy rozhodovací problém nalézt co největší velikost okraje M a zároveň co nejnižší vektor w tak, aby separační linie rozdělovala množinu bodů správně.

4.2.1 Kernel trick

Doposud jsme spoléhali na to, že data pro klasifikaci v příznakovém prostoru budou lineárně rozdělitelná. Na tomto principu je totiž metoda Support Vector Machine založena. Mnohdy však není možné data rozdělit pouhou hyperplochou tak, aby chyba rozdělení na třídy byla malá. Tento problém můžeme řešit například tím, že pro popis objektu použijeme jiné příznakové vektory, které již jsou lineárně separabilní. Ne vždy je však možné takovéto vektory nalézt. Právě pro tento případ je možno využít klasifikaci SVM za použití kernelových metod, mnohdy označovaných jako *kernel trick*.

Metoda s využitím kernelu přemapovává množinu příznaku z jednoho prostoru do jiného. Mnohdy tyto prostory nemusí mít stejnou dimenzi, jelikož například v N-rozměrném prostoru nemusíme nalézt patřičnou lineární hyperplochu, ale v N+1 dimenzionálním prostoru již tuto lineární plochu nalézt můžeme.



Obrázek 4.1: Ukázka přemapování příznaků, které nelze ve 2D prostoru lineárně rozdělit, na 3D prostor, jenž už hyperplochou rozdělit lze. [16]

Snažíme se tedy nalézt funkci ϕ, která vhodně přemapovává body z jednoho prostoru do jiného.

$$\Phi : \mathcal{R}^M \longrightarrow \mathcal{R}^N \tag{4.8}$$

Příkladem může být přemapování na elipsu:

$$\Phi : (x_1, x_2) \longrightarrow (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \tag{4.9}$$

Kernel lze tedy vyjádřit následující rovnicí

$$K(x, y) = \Big\langle \Phi(x), \Phi(y) \Big\rangle \tag{4.10}$$

Takže zbývá jen otázka jaký kernel lze použít? V podstatě můžeme použít jakoukoli funkci, která je pozitivně definitní. Toto je výsledek *Mercersova teorému* [6], který dokonce říká, že můžeme vzít dva kernely, spojit je dohromady a výsledkem nám vznikne kernel další. Každopádně existují tři základní druhy funkcí [6], které jsou běžně používány a mají kernely s nimi spojené.

- Polynomiální funkce stupně s s elementy x_k vstupního vektoru s kernelem:

$$K(x, y) = (1 + x \cdot y)^s \tag{4.11}$$

- Funkce sigmoid vektoru x_k s parametry κ a δ a kernelem:

$$K(x, y) = tanh(\kappa \cdot x \cdot y - \delta) \tag{4.12}$$

- Radiální funkce s prvky x_k vstupního vektoru s parametrem σ a kernelem:

$$K(x, y) = exp\Big(- \frac{(x - y)^2}{2\sigma^2} \Big) \tag{4.13}$$

4.3 AdaBoost

Boosting je mocná technika pro kombinování více bázových klasifikátorů, která dokáže poskládat silný klasifikátor, jehož výkon bude signifikantně vyšší než jakýkoli z bázových klasifikátorů. V této práci si popíšeme jeden z široce využívaných boostovacích algoritmů s názvem AdaBoost[2], což je zkratka pro adaptivní boostování. Tato metoda byla vyvinuta Freundem a Shapirem v roce 1996. Boosting může poskytovat dobré výsledky, i když bázové klasifikátory mají úspěšnost o něco málo vyšší než náhodnou. Proto tyto klasifikátory označujeme jako slabé klasifikátory.

Hlavním rozdílem mezi boostováním a jinými standardními algoritmy je to, že bázové klasifikátory jsou trénovány v sériích a každý bázový klasifikátor je trénován s použitím vah datových bodů, které byly určeny předchozími klasifikátory. Konkrétně body, které byly špatně klasifikovány v některém z předchozích klasifikátorů, získávají vyšší váhu pro následující klasifikátory. Jakmile jsou všechny klasifikátory natrénovány, je jejich predikce zkombinována za použití přidělených vah.

Uvažujeme-li klasifikační problém dvou tříd, kde máme trénovací data uložena jako vektory $x_1, ..., x_N$, které korespondují s binárními cílovými třídami $t_1, ...t_N$, které mohou nabývat hodnot $t_n = \{-1, 1\}$. Každému datovému bodu je přidělena hodnota váhy, která je na počátku inicializovaná na $\frac{1}{N}$. V každé fázi algoritmu, AdaBoost trénuje klasifikátor s použitím dat ze všech předchozích bázových klasifikátoru a přiděluje vyšší váhu špatně klasifikovaným bodům.

Algoritmus metody AdaBoost: [6]

- inicializuj všechny váhy na $\frac{1}{N}$, kde N je počet datových bodů
- dokud $0 \leq \epsilon_t \leq \frac{1}{N}$ (a $t < T$, maximální počet iterací)
  - trénuj klasifikátor na $\{S, w^{(t)}\}$, získaní hypotézy $h_t(x_n)$ pro datový bod x_n
  - vypočítej trénovací chybu $\epsilon_t = \sum_{n=1}^{N} w_n^{(t)} I(y_n \neq h_t(x_n))$
  - nastav $\alpha_t = log(\frac{1-\epsilon_t}{\epsilon_t})$
  - aktualizuj váhy s použitím:

$$w_t^{(t+1)} = w_n^{(t)} \frac{exp(\alpha_t \cdot I(y_n \neq h_t(x_n))}{Z_t} \tag{4.14}$$

    kde Z_t je normalizační konstanta
- **výstup**: $f(x) = sign\Big(\sum_{t=1}^{T} \alpha_t \cdot h_t(x)\Big)$

Kapitola 5

Návrh řešení

V úvodní kapitole jsme si popsali problematiku klasifikace vozidel v běžném provozu a nastínili, jak k této klasifikaci přistupovat. Jedna z možných cest je právě s použitím radarových modulů, které jsou už dnes běžně k mání a zároveň mají kompaktní rozměry tak, aby je bylo možné využít právě k takovýmto účelům.

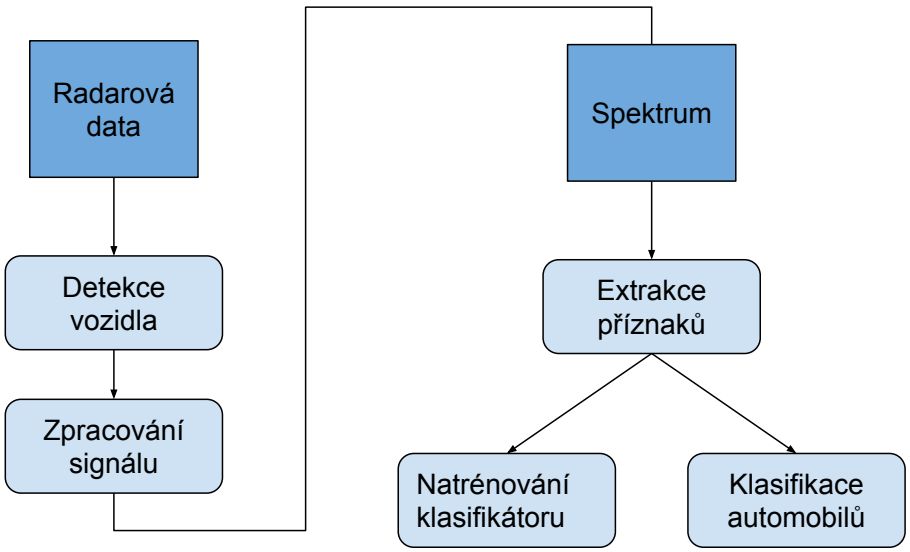
Postupně v dalších kapitolách jsme se detailněji seznámili s danou problematikou a zjistili, jaké operace je potřeba vykonat, abychom ze surového radarového signálu získali cenné informace pro klasifikaci. S využitím těchto znalostí si nastavíme cíle této práce a popíšeme si jednotlivé kroky tohoto procesu.

Cílem této práce je tedy seznámit se s oborem zabývajícím se radarovými daty. Konkrétně se budeme zabývat podkategorií radarů s kontinuální vlnou fungující na principu Dopplerova jevu. Je potřeba seznámit se s radarovou problematikou tak, abychom byli schopni vyextrahovat užitečné informace pro samotnou klasifikaci. Jelikož se velká část této práce zabývá zpracováním signálu, je našim cílem vytvořit knihovnu, jejíž funkčnost bude zaobalovat veškerou manipulaci s obecným signálem a později ji rozšířit o funkčnost spojenou s konkrétním radarovým signálem. Tato knihovna bude schopna načítat radarový signál ve zvoleném formátu, odfiltrovat z něj synchronizační bity a uložit do předem dané struktury. S takovýmito daty uloženými v dané struktuře budeme moci provádět celou řadu operací se signálem. Od počátečního dělení signálu na rámce, detekcí průjezdu vozidel až po výpočet Fourierovy transformace pro získání spektra, ze kterého budeme schopni poskládat spektrogram, což je zobrazení spektra měnícího se v čase. Právě z tohoto spektrogramu budeme těžit největší množství informací pro samotnou klasifikaci. Tato knihovna bude detailněji popsána v kapitole 6.2.

Další významnou částí této práce je samotná klasifikace radarových dat do předem zvolených tříd. Úspěšnost klasifikace bude velice záležet na tom, jak zvládneme předpracovávat signál, zbavit jej veškerého šumu a hlavně oddělit jednotlivá vozidla do samostatných celků. Neméně důležitým prvkem bude také volba patřičných příznakových vektorů k popisu jednotlivých kategorií. Součástí bude také porovnání dvou klasifikačních algoritmů z kategorie učících metod pomocí učitele. Konkrétně se budeme zabývat metodami využívajícími Support Vector Machine a jejich modifikací pomocí Kernel trick a jako druhý zástupce klasifikačních algoritmů byl zvolen AdaBoost. Oba tyto algoritmy byly vybrány z důvodu předchozích zkušeností, kdy se jejich úspěšnost pohybovala mezi nejlepšími, a taky proto, že byly zařazeny do kategorie top 10 machine learning algoritmů [17].

Obrázek 5.1 detailněji popisuje celý proces samotné aplikace. Na úplném počátku máme nijak nezpracovaná data z radaru využívající Dopplerův jev. Následně je potřeba detekovat průjezdy jednotlivých vozidel a zpracovat jejich signál do podoby, ze které jsme schopni

vyextrahovat užitečná data k matematickému popisu jednotlivých vozidel. Jakmile máme tyto příznakové vektory, může nastat fáze trénovaní klasifikátoru a později samotná klasifikace na takto natrénovaném klasifikátoru. V této fázi budeme schopni získat procentuální úspěšnost klasifikace, ze které můžeme vyvodit závěry. Samozřejmě je cílem této práce je dosáhnout co možná nejlepších klasifikačních výsledků tak, aby bylo možné vytvořenou aplikaci použít v reálném provozu.



Obrázek 5.1: Blokové schéma navržené aplikace znázorňující jednotlivé kroky procesu.

Součástí výsledné aplikace bude, společně s knihovnou pro zpracování signálu a klasifikační části i grafické uživatelské rozhraní. Rozhodnutí naimplementovat grafické rozhraní padlo hlavně z toho důvodu, aby uživatel mohl vizuálně vnímat, jak vypadá daný úsek radarového signálu a mohl jej přiřadit k datům, která budou extrahována jako klasifikační vektory. Součástí tohoto rozhraní bude i možnost propojit radarový signál se zachyceným videozáznamem z kamery a tedy jasně přiřadit snímek videa ke konkrétnímu projíždějícímu vozidlu.

5.1 Klasifikační třídy

Cílem této práce je klasifikovat vozidla v běžné provozu na silnicích. Na počátku si tedy musíme zvolit klasifikační třídy. Na základě pozorování silničního provozu při zkoumání videozáznamu, jenž byl pořízen spolu s radarovými daty, bylo rozhodnuto vytvořit následující klasifikační třídy:

- **Osobní automobily** - do této kategorie se řadí běžná osobní vozidla spadající do kategorie do 3.5 tuny, která jsou určena ke každodenní přepravě. Vyznačují se zejména svými malými rozměry a to především svou krátkou délkou a malou výškou. Svým početným zastoupením se řadí mezi nejběžnější typy vozidel a můžeme tedy předpokládat, že se v radarových datech budou vyskytovat s největší frekvencí.

- **SUV** - tato kategorie je velice podobná kategorii předchozí. Jedná se o vozy převážně terénní, jež se liší od předchozí kategorie hlavně svou výškou a celkově větším objemem. Opět i tato kategorie spadá do třídy pod 3.5 tuny a můžeme předpokládat, že její četnost bude daleko nižší než u klasických osobních automobilů.

- **Dodávky** - u této třídy se přesouváme do kategorie převážně užitkových a nákladních aut. Tato kategorie se vyznačuje většími rozměry a jejich povrch není tak členitý jako u předcházejících kategorií a tudíž nedochází k lomeným odrazům radarového vlnění. Vozidla této třídy se nachází na hraně vozidel kategorie do a nad 3.5 tuny.

- **Kamiony** - jedná se o poslední třídu vozidel jejíchž hlavním rysem jsou velké rozměry. Do této kategorie patří veškerá kamionová doprava společně s nákladními vozidly spojenými s přívěsy. Tato vozidla jezdí převážně na hlavních tazích, dálnicích a rychlostních komunikacích. Na základě tohoto poznatku musíme provádět měření na těchto úsecích tak, abychom měli dostatek dat ke klasifikaci. Vozidla této kategorie mají převážně dlouhé tvary s členitým povrchem, který se bude jasně jevit při odrazu radaru. Kamiony se skládají ze dvou částí, kdy první tvoří kabina vozidla a druhou přívěs s nákladovým prostorem. Tyto dvě části lze teoreticky rozeznat z odraženého signálu.

5.2 Návrh příznakových vektorů

Na počátku každé klasifikace je potřeba si určit takzvané příznakové vektory. Ty slouží k popisu objektu a jsou jakýmsi matematickým vyjádřením vlastností zkoumaného objektu. Volba příznakových vektorů je klíčovým procesem každé klasifikace, jelikož špatně zvolené vektory mohou mít za následek nepřesnou klasifikaci a nízkou úspěšnost rozpoznání typu vozidla. Je tedy důležité volit příznaky tak, aby z nich bylo co nejjednodušší rozeznat kategorie navzájem. Dalším důležitým kritériem, které musíme při volbě zohlednit, je množství šumu v signálu zapříčiněné různými nedokonalostmi, ať už se jedná o nepřímé odrazy či nedokonalosti techniky, se kterou měření provádíme. Musíme tedy volit příznaky tak, aby bylo množství šumu zanedbatelné a negativně neovlivňovalo naměřené a vypočítané výsledky.

Poslední věcí, na kterou je potřeba upozornit je to, že není potřeba, aby získané vektory přesně odpovídaly například fyzickým rozměrům vozidel. Jedná se totiž o klasifikační problém, kde není nutné získat naprosto přesné rozměry, ale daleko důležitější je, aby všechna vozidla stejné klasifikační třídy měla tyto hodnoty podobné. Pokud bychom to uvedli na příkladu, tak není nezbytné, abychom změřili délku vozidla na centimetr přesně, ale je nutné, aby všechna vozidla dané třídy měla tyto odchylku podobnou a aby tato odchylka nezasahovala do jiné třídy.

Na základě uvedených skutečností a znalostí o tom, jak vypadá radarový signál byly zvoleny následující příznakové vektory pro klasifikaci.

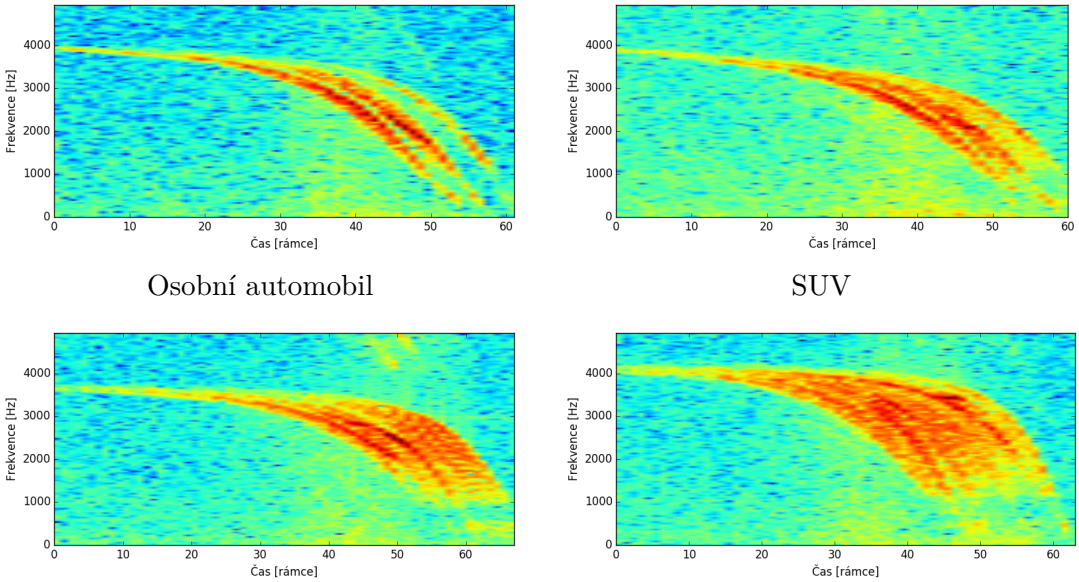
- **Rychlost** - jedna z prvních věcí, která většinu lidí napadne ve spojení s radarem. Rychlost lze z radarového signálu relativně snadno extrahovat, jelikož většina radarů s kontinuální vlnou funguje na principu Dopplerova jevu. Dopplerův jev totiž popisuje vzájemný vztah rychlosti a změny frekvence signálu při odrazu. Rychlost je sice zařazena mezi příznakové vektory, ale nebudeme ji využívat k samotné klasifikaci. Jedná se totiž o veličinu, která nepopisuje vlastnosti vozidel na základě jejich tříd. Ačkoli

tato veličina nebude použita ke klasifikaci, jedná se o důležitou informaci pro výpočet dalších příznakových vektorů, popřípadě k jejich korekci.

- **Délka vozidla** - jedná se o první fyzickou vlastnost vozidel, jež bereme v potaz jako příznakový vektor. Pomocí délky vozidel, lze jednoznačné rozdělit vozidla do kategorií malá, střední a dlouhá. Problémem tohoto vektoru při klasifikaci může být to, že sice rozpoznáme nákladní automobil od osobního, ale jasně rozpoznat osobní automobil od SUV půjde s obtížemi, jelikož tyto třídy vozidel se většinou liší svým objem. Abychom dosáhli úspěšného rozpoznání mezi těmito dvěma kategoriemi, budeme muset příznakový vektor délky spojit s dalšími vektory. Délku vozidla lze měřit jednoduchým způsobem, a to tak, že rychlost vozidla vynásobíme s délkou radarového signálu. Aby byl výpočet délky přesný, je potřeba brát v úvahu i úhel rozsahu vysílaných radarových paprsků.

- **Spektrální hustota výkonu** - u tohoto vektoru se plně přesouváme od fyzických vlastností vozidel k tomu, co dokážeme získat z radarového signálu. Jedná se o jedno číselnou reprezentaci spektra. Z předchozích kapitol víme, že se frekvence radarového signálu mění v závislosti na úhlu, pod kterým dopadá na odrazovou plochu. S tím je tedy spojeno, že ve spektrogramu jsme schopni naměřit více dominantních frekvencí právě v souvislosti s odrazovými plochami a jejich dopadových úhlech. Další důležitou informací je to, že čím větší je odrazová plocha vozidla, tím je amplituda dominantní frekvence větší. Na základě těchto informací můžeme odvodit, že spektrální hustota výkonu reprezentuje celkovou odrazovou plochu vozidla. Ve spojení s předchozím příznakovým vektorem délky můžeme jasně popsat tvar, velikost a délku, jež bude důležitá pro samotnou klasifikaci.

- **Spektrální výkon napříč rámci** - tento příznakový vektor je hodně podobný tomu předcházejícímu. Jedinou výjimku tvoří to, že zachycujeme spektrální výkon pro každý rámec signálu zvlášť a jsme tedy schopni zachytit měnící se spektrální výkon v čase. Při pohledu na spektrogram daného průjezdu vozidla lze vypozorovat, že jednotlivé kategorie mají odlišný průběh v čase, a to především v druhé půlce spektrogramu. Jedná se tedy o první příznakový vektor, který se nachází za hranicí jednorozměrného prostoru. Vícerozměrný vektor sice neznamená automaticky lepší úspěšnost klasifikace, ale budeme moci porovnat, jak si povede tento vektor s předchozími jmenovanými.



Obrázek 5.2: Ukázka spektrogramu jednotlivých tříd vozidel. Již pouhým okem lze ve spektrogramu vypozorovat značné rozdíly, které lze využít při klasifikaci.

Výše uvedené body jsou pouze návrhem toho, co budeme při klasifikaci používat. Detailnější popis, jak lze příznaky ze signálu extrahovat a vypočítat přesné hodnoty je uveden v kapitole implementace 6.3.

Kapitola 6

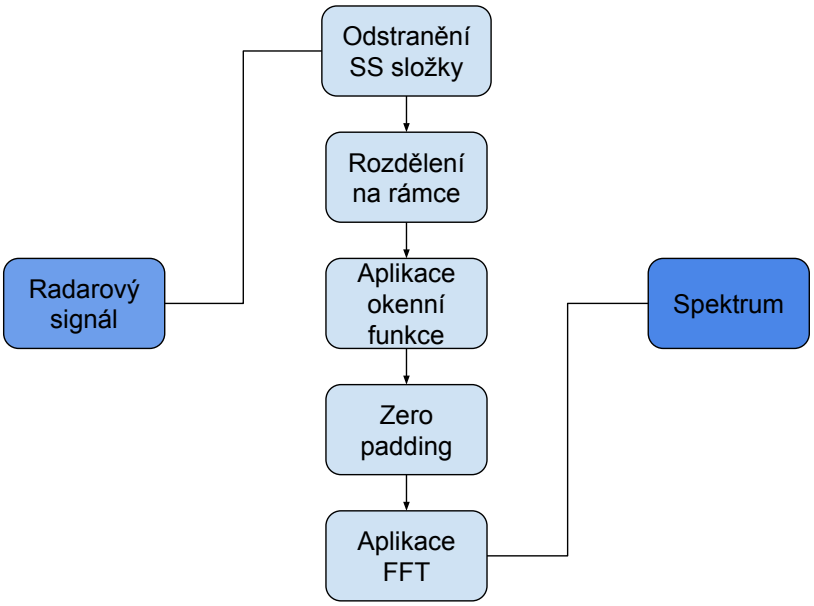
Implementace

Výsledkem této práce bude klasifikátor, jenž bude schopen klasifikovat projíždějící vozidla ze záznamu radarové nahrávky. Aby bylo možné klasifikátoru poskytnout data, ze kterých bude schopen úspěšně rozřadit průjezdy do správných kategorií, je potřeba vytvořit část starající se o zpracování signálu. Bude se jednat o knihovnu, jež bude možné použít i v jiných projektech zabývající se radarovou tématikou. Jako implementační jazyk byl zvolen jazyk Python ve verzi 3.5. Tento jazyk byl zvolen kvůli své popularitě v poslední době a hlavně kvůli dostatečné škále podpůrných knihoven. Právě tyto knihovny hrají v této práci velkou roli, jelikož není nutné implementovat vše od nuly. Daleko důležitější je soustředit se na námi zvolenou problematiku.

Jelikož se jedná o práci, kde se zpracovává signál, což se jinými slovy dá vyjádřit jako velké pole čísel, tak hlavní použitou knihovnou je knihovna *Numpy* [8]. Pomocí této knihovny jsme schopni reprezentovat signál ve formě matice a nativně s ní provádět celou řadu matematických operací. Další důležitou věcí je i to, že ostatní knihovny podporují tyto formáty a jsou vzájemně kompatibilní. Tím se dostáváme k další využité knihovně. Tou je známa knihovna pro práci se signálem s názvem *Scipy* [9]. Pomocí ní jsme schopni generovat okénkové funkce, provádět konvoluce nebo interpolovat příznakové vektory. Další důležitou knihovnou pro samotnou klasifikaci je knihovna *Sklearn* [3], jež obsahuje celou řadu funkcí pro strojové učení, ze kterých nás nejvíce zajímají metody klasifikace. Jelikož v této práci bude pro demonstraci výsledků naimplementováno i grafické uživatelské rozhraní, bude použita i další známá knihovna *Qt* [15], konkrétně se jedná o její portaci pro Python *PyQt5*. Jak byly knihovny zakomponovány do zdrojového kódu je podrobněji popsáno v kapitole 6.2.

6.1 Zpracování signálu

Zpracování radarového signálu je pro tuto práci stěžejní, jelikož jednotlivými úpravami ovlivňujeme, jak budou data vypadat a jak obtížné z nich bude extrahovat informace pro klasifikační vektory. Jak již víme z předchozích kapitol, výstupem radaru je diskrétní signál, který byl zachycen přijímací anténou, poupraven a následně navzorkován. Vzorkovací frekvence použitého radaru K-MC4 je 50kHz. Zpracováním signálu v tomto projektu je myšlena sada kroků, která je zobrazená na obrázku 6.1, na jejímž počátku je radarový signál a výsledkem je spektrum v čase.



Obrázek 6.1: Blokové schéma procesu zpracování radarového signálu.

1. **Rozdělení signálu na rámce** - tímto krokem rozdělíme celý signál, který může být velice dlouhý na menší části. Částečně toto děláme z důvodu snížení paměťové náročnosti, ale hlavním důvodem je to, že se signál může v čase měnit. Pokud tedy signál rozdělíme na dostatečně malé části, ve kterých víme, že nemůže docházet k velkým výkyvům, můžeme signál prohlásit za stacionární. Výběr délky rámce závisí na konkrétní aplikaci, avšak po sérii experimentů byla vybrána délka rámce o 1024 vzorcích. Dalším důležitým elementem u dělení signálu na rámce je i to, zda se rozhodneme pro rámce bez překryvu či s překryvem. Opět i zde experimenty ukázaly, že nejlepší varianta je použít překryv 50% délky rámce. Jedná se tedy o 512 vzorků. Překryv volíme z důvodu abychom nepřišli o důležité informace na okrajích rámců, jelikož vliv použité okenní funkce obvykle utlumuje signál na okrajích rámce.

```
def split_to_frames(self, samples, overlap, frame_count):
    frames = []
    frames_shift = samples - overlap

    frame_start = 0
    frame_end = samples
    for i in range(frame_count):
        frame_data = self.data[frame_start:frame_end]

        frame = Frame(data=np.array(frame_data), fs=self.fs)

        frames.append(frame)
        frame_start += frames_shift
        frame_end += frames_shift
    return frames
```

Listing 1: Ukázka kódu rozdělujícího signál na rámce s překrytím.

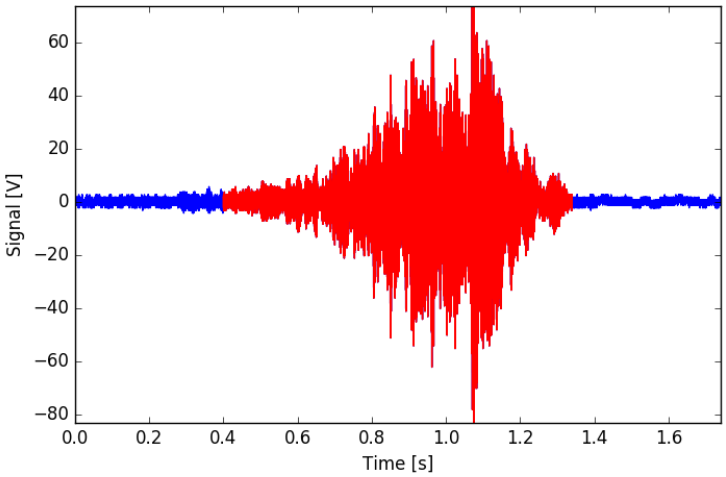
2. **Odstranění stejnosměrné složky** - odstranění stejnosměrné složky provádíme z toho důvodu, že nenese žádnou užitečnou informaci a může negativně ovlivnit výpočet energie rámce. Jelikož předem známe všechny hodnoty rámce, nemusíme střední hodnotu odhadovat a můžeme ji přesně vypočítat. Jedná se o střední hodnotu, kterou získáme funkcí `numpy.mean()`. Následně je potřeba tuto hodnotu odečíst od všech vzorků rámce. Jelikož využíváme knihovnu `numpy`, nemusíme iterovat skrz každý vzorek a postupně odečítat, ale můžeme provézt klasický odečet `data_array - mean`.

3. **Aplikace okenní funkce** - samotný proces dělení signálu na rámce je možno považovat za aplikaci okenní funkce. Jedná se totiž o aplikaci obdélníkového okna, které odřízne okraje rámce. To je v mnohých případech nežádoucí, jelikož se ve spektru mohou vyskytovat vysoké frekvence způsobené právě tímto odseknutím. Proto je dobré využít okenní funkci takovou, která okraje rámce postupně utlumí. V tomto projektu je použita jedna z nejznámějších okenních funkcí. Konkrétně se jedná o Hammingovo okno, které je reprezentováno třídou `HammingWindow`. Tato třída si předem vypočte Hammingovo okno na základě počtu vzorků v rámci. Následným voláním funkce `HammingWindow.apply(frame)` je okno aplikováno na daný rámec. Hammingovo okno utlumí okraje rámce, a proto je důležité použít rámce s překrytím, abychom neztratili cenné informace.

4. **Výpočet spektra** - tento krok je zároveň posledním krokem celého zpracování signálu. Pomocí Rychlé Fourierovy transformace analyzujeme frekvenční složky předzpracovaného signálu. Pro výpočet Rychlé Fourierovy transformace je použita funkce `numpy.fft.fft()`, jejímž výstupem je série kompletních čísel. Výsledné spektrum diskrétního signálu je periodické v závislosti na vzorkovací frekvenci. Jelikož pracujeme s komplexním signálem složeným z kanálu I a Q, můžeme horní polovinu spektra překlopit do záporné poloviny a tím získat informace o případném směru pohybu vozidla. To jsme schopni provézt vygenerováním frekvenční osy pomocí funkce `numpy.fft.fftfreq()`. Výsledek Fourierovy transformace je reprezentován objektem třídy `Spectrum`.

Detekce vozidel

Detekce aktivity v jakémkoli signálu je téma, na které neexistuje zaručený recept jak ji dělat se 100% úspěšností. U radarového signálu se jedná o stejný problém, jelikož měření neprovádíme v ideálním světě, ale v reálném provozu. Ten obsahuje spoustu šumu a rušivých odrazů, které negativně ovlivňují detekci.



Obrázek 6.2: Detekce projíždějícího vozidla vyznačená červeně v původním signálu.

V tomto projektu byly použity dva principy detekce. První z nich využíval informaci střední krátkodobé energie signálu konkrétního rámce. Dá se totiž předpokládat, že rámce s vysokou intenzitou energie zachycují projíždějící vozidlo. Střední krátkodobý výkon můžeme vyjádřit vzorcem jako:

$$E_F = \frac{1}{N} \sum_{n=0}^{N-1} x[n]^2, \tag{6.1}$$

kde N je počet vzorků v rámci a $x[n]$ je diskrétní hodnota vzorku signálu. Hodnotu energie je následně nutné porovnat s předem určenou hodnotou prahu.

$$E_T < E_F \tag{6.2}$$

Pokud je energie vyšší než daný práh, je tento rámec brán jako aktivní a je zařazen k průjezdu vozidla.

Tato metoda bohužel nefungovala spolehlivě, jelikož bylo obtížné nalézt hodnotu prahu tak, aby oddělila průjezdy malých vozidel od vzniklého šumu. Druhá metodou, která byla použita v tomto projektu, byla založena na analýze dominantních frekvencí. Rušivé odrazy a vzniklý šum totiž nemají ve frekvenční složce vysoké hodnoty. Pro tuto metodu je potřeba vypočítat Fourierovu transformaci a následně odfiltrovat nízké a vysoké frekvence, což je oblast, kde se běžně odražené radarové frekvence nepohybují. V takto vyfiltrovaných datech musíme nalézt, zda obsahuje dominantní frekvenci. Tato kontrola je prováděna porovnáním spektra s určitým prahem. Tato metoda vykazovala lepší výsledky než předchozí, jelikož nedetekovala falešné průjezdy.

V předchozím kroku jsme identifikovali konkrétní rámce, na kterých jsme detekovali průjezd vozidel. Následně musíme pospojovat sousední rámce dohromady a vytvořit tak signál průjezdu vozidla. Je tedy potřeba odfiltrovat úseky, které obsahují malý počet rámců tak, abychom nedetekovali falešné průjezdy.

6.2 Rozvržení do tříd

Součástí této práce je i univerzální knihovna, jež dokáže pracovat s radarovým signálem. Tato sekce podrobně popisuje návrh a funkčnost celé knihovny a detailněji rozebírá její jednotlivé části tedy třídy. Při návrhu knihovny byl kladen důraz na využití vlastností objektově orientovaného jazyka a hlavně využití dědičnosti. Knihovna je rozdělena na několik vrstev tak, že v nejnižší vrstvě třídy reprezentují obecný signál a v dalších vrstvách třídy reprezentují radarový signál či signál spjatý s průjezdem vozidla.

GenericSignal

Tato třída je rodičovská třída, která obsahuje většinu funkcí pro práci se signálem a od níž jsou dědičností odvozeny ostatní specifické třídy. Tato třída nese informaci o obecném signálu a její hlavní komponentou je proměnná v níž jsou uloženy hodnoty signálu v čase. Data jsou uložena v proměnné typu `numpy.array`. Signál může být reprezentován jako série reálných čísel a nebo jako série komplexních čísel. Komplexní čísla je možno ukládat z toho důvodu, že radarový signál je reprezentován kanály I a Q, jež dohromady tvoří komplexní číslo. Dalšími důležitými proměnnými z konstruktoru objektu jsou vzorkovací frekvence a počáteční čas signálu. Signál je možno rozdělit na menší rámce metodou `split_to_frames()`, které poskytneme jako parametr délku rámce buďto v čase nebo ve vzorcích a taky délku překryvu. Výsledkem je pole rámců reprezentované třídou `Frames`.

Další možností, jak vytvořit objekt této třídy je poskládat původní signál ze série rámců. Pro tyto účely je zde metoda `join_signal_from_frames()`. Tato funkce je užitečná v případě, že načítáme data z RRC formátu pomocí načítacího skriptu, který neprodukuje celý signál, ale právě jen jednotlivé rámce původního signálu.

Pak zde máme celou řadu funkcí upravující původní signál. První z nich je metoda `remove_dc_component()`, která ze signálu odstraní stejnosměrnou složku. Metodou `zero_padding()` můžeme nadstavit signál nulami tak, abychom získali detailnější výstup Fourierovy transformace. Další metodou, která je použita při detekci vozidel, je metoda `energy()`, jež spočítá výkon daného signálu.

Poslední sadou metod jsou metody pro získávání spektra a spektrogramu. Metoda `get_fft()` spočte Rychlou Fourierovu transformaci pomocí `numpy.fft.fft()`. Jelikož je výsledek této metody vrácen v podobě komplexního čísla, musíme provézt převod na reálné číslo pomocí absolutní hodnoty. Tato metoda vrací objekt typu `Spectrum`, jenž je popsán níže. Poslední metoda `get_spectrogram()` využívá předešlé metody k poskládání spektrogramu z rámců signálu.

RadarSignal

Tato třída reprezentuje konkrétní radarovou nahrávku. Je odvozena od třídy `GenericSignal`, takže obsahuje stejné proměnné i celkovou funkčnost. Ke své základní funkčnosti přidává metody pro načítání radarového signálu ze souboru. Existují celkem dva formáty souboru, do nichž může radarový modul exportovat nahraná data. První z nich je soubor typu CSV,

který obsahuje několik sloupců v závislosti na počtu kanálů. Pro tyto soubory je zde připravena metoda `load_csv_data()`. Tato metoda postupně načte všechna data ze souboru a složí z nich pole komplexních čísel. Problémem tohoto formátu je to, že radar generuje synchronizační hodnoty, které se vyskytují co 256 hodnot a jsou nežádoucí. Musíme je tedy ze signálu odstranit pomocí metody `_clear_sync_values()`.

Dalším typem souboru je soubor ve formátu RRC. Jedná se o binární formát, který uchovává nejen hodnoty signálu, ale i metadata o nahrávce. Pro načtení tohoto typu souboru je využita metoda `load_rrc_data()`, která ke svému chodu využívá dodanou načítací třídu `RadarReader`, která odfiltrovává veškeré synchronizační hodnoty a načítá signál po rámcích tak, abychom mohli zpracovávat signál například v reálném čase.

Poslední funkčností této třídy je možnost detekce projíždějících vozidel. K tomuto účelu je vytvořena metoda `detect_vehicles()`, která na základě aktivity ve spektru poskládá z rámců signál průjezdu vozidla a vrátí seznam průjezdu reprezentovaný třídou `VehicleSignal`.

VehicleSignal

Touto třídou se přesouváme ke specifickým třídám reprezentující průjezdy vodila v radarovém signálu. Opět i tato třída je odvozená od základní třídy `GenericSignal`. Jelikož její instance reprezentují konkrétní průjezdy vozidel, tak je většina její funkčnosti vztažena právě na získávání informací o vozidle. Máme zde řadu metod pro získávání klasifikačních vektorů. Konkrétně se jedná o metody `get_speed()` pro získání rychlosti ze spektrogramu, `get_length()` pro získání délky vozidla, `get_reflected_surface()` pro získání odrazové plochy a nakonec `get_frame_power()` k získání spektrálního výkonu napříč rámci. V této třídě nalezneme i metodu `get_category()`, které musíme předložit natrénovaný klasifikátor a pomocí něj jsme schopni zařadit vozidlo do dané kategorie.

Spectrum

Tato třída je výsledkem Rychlé Fourierovy transformace a nese informace o tom, jaké frekvence jsou v signálu zastoupeny. Základem je proměnná `spectrum`, která je složena ze dvou částí. První část reprezentuje „kladné“ frekvence, jež značí vozidla, která se pohybují směrem k radaru a druhá část reprezentuje „záporné“ frekvence, jež značí vozidla pohybující se směrem od radaru. V předchozí větě jsou použity uvozovky, jelikož kladné a záporné frekvence neexistují, ale v tomto kontextu to takto můžeme použít, jelikož pracujeme s oběma kanály I a Q.

Tato třída napomáhá detekci vozidel, jelikož obsahuje funkci `threshold()`, která určí, zda jsou ve spektru nějaké dominantní frekvence a zda jejich hodnota překračuje daný práh. Ne vždy je potřeba pracovat s celým rozsahem frekvencí, jelikož nejzajímavější hodnoty se pohybují do 7000Hz. Právě pro tyto účely je zde metoda `limit_frequency()`, která omezí maximální frekvenci spektra danou horní hranicí.

Spectrogram

Poslední třídou, kterou si zde představíme je třída reprezentující spektrogram. Jedná se o vizuální zobrazení spektra jednotlivých rámců v čase. Základem je tedy dvourozměrné pole typu `numpy.array`, kdy jeho první dimenzí je čas v rámcích a druhou je zastoupení frekvencí v rámci. Tato třída má spojitost s předchozí, jelikož využíváme pole spekter k poskládání spektrogramu. Tak jako třída `Spectrum` má i tato možnost omezit maximální

frekvenci metodou `limit_max_freq()`. Další užitečnou metodou, kterou můžeme zvýraznit dominantní frekvence, je metoda `threshold()`, která propustí pouze část původního spektra. Samozřejmostí je i zobrazení spektrogramu do grafu. To je provedeno převodem spektra na decibely pomocí vzorce 6.3 a následnou interpolací funkcí `plt.imwhow()`.

$$X(k)_n = 20 \cdot log_{10}(X(k)) \tag{6.3}$$

6.3 Extrakce příznaků

Rychlost

V této práci, jak již bylo zmíněno, využíváme kategorii radaru fungující na Dopplerově jevu. Tento jev popisuje změnu vysílané a přijímané frekvence na základě relativního pohybu pozorovatele a zdroje vysílání. V našem případě se jedná o situaci, kdy radar stojí na jednom místě a má tedy nulovou rychlost a vysílá vlnění do prostoru. V tomto zkoumaném úseku se pohybují vozidla různou rychlostí a mění tak frekvenci odražených vln. Rozdíl frekvencí vyslaných a přijatých nazýváme Dopplerovou frekvencí. Tato frekvence je pro získání rychlosti pohybu vozidla stěžejní.

V publikacích o Dopplerově jevu velmi často nalezneme základní vzorec, jenž popisuje vztah mezi rychlostí pozorovatele, vysílače a vysílanými frekvencemi. Tento vzorec má následující tvar:

$$f = \left(\frac{c + v_r}{c + v_s}\right) f_0, \tag{6.4}$$

kde f je Dopplerova frekvence, f_0 je frekvence vysílače, c je rychlost šíření vln v daném prostředí, v_r je rychlost pozorovatele a v_s je rychlost vysílače. Tuto rovnici můžeme dále poupravit, jelikož víme, že radar je stacionární a jeho rychlost je nulová. Upravený vzorec vypadá následovně.

$$f_D = \frac{2v}{c_0} f_0, \tag{6.5}$$

který finální úpravou transformuje na námi požadovaný vzorec pro výpočet rychlosti.

$$v = \frac{f_D \cdot c}{2 f_0} \tag{6.6}$$

V tomto vzorci se vyskytují nám všechny známé hodnoty, a to konkrétně c_0 jako rychlost světla ($1079252848,8 \frac{km}{h}$), f_0 je vysílací frekvence radaru. V případě použití radaru K-MC4 je tato frekvence 24Ghz. Pak zde máme změnu frekvence f_D a rychlost pohybu vozidla v.

Z výše uvedeného vzorce je patrné, že musíme získat Dopplerovu frekvenci, kterou získáme z radarových dat. Tuto frekvenci získáme výpočtem rychlé Fourierovy transformace nad každým rámcem signálu, kde zrovna vozidlo projíždí. V ideálním případě bychom ve spektrogramu viděli pouze jednu dominantní frekvenci po celou dobu průjezdu vozidla. Toto ovšem v reálném světě neplatí, jelikož signál obsahuje spoustu šumu a signál se od různých ploch vozidla odráží různým způsobem. Z experimentů prováděných nad radarovými daty vyplynulo, že nejpřesnější měření rychlosti je hned na počátku průjezdu vozidla, kdy je úhel mezi směrem vysílání vln a směrem pohybu vozidla nejmenší. Dominantní frekvence je tedy vypočtena z prvních 10% rámců, a to tak, že je v každém rámci získána hodnota frekvence, která je nejvíce zastoupena a následně je na tyto maximální frekvence použit medián, kterým zajistíme, že hodnoty šumu nabývajících vysokých nebo naopak nízkých hodnot budou odfiltrovány.

Další důležitou věcí, na kterou při výpočtu rychlosti nesmíme zapomenout, je i korekce vypočtené rychlosti takzvaným *faktorem kosinového úhlu*. Ten se projevuje vždy pokud vysíláme vlny pod úhlem větším než 0°. Korekci lze provézt následující rovnicí:

$$v_r = \frac{v_z}{cos(\alpha)}, \tag{6.7}$$

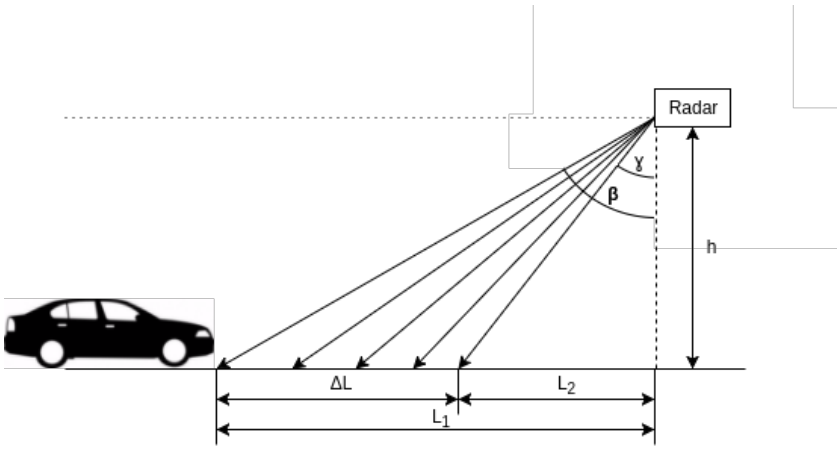
kde v_z je získaná rychlost pomocí rovnice 6.6, α je úhel pod kterým jsou vysílány vlny a v_r je reálná rychlost projíždějícího vozidla.

Délka vozidla

Délka vozidla je jednou z fyzických vlastností automobilu, pomocí které dokážeme rozlišit vozidla na malá, velká a střední. První myšlenou, jak vypočítat délku vozidla pomocí radarového signálu byla ta, že využijeme dobu signálu, po kterou je vozidlo snímáno radarovým signálem a vynásobíme jí rychlostí, kterou získáme na základě Dopplerova jevu ze spektrogramu. Jednoduchá rovnice by tedy vypadala následovně:

$$L_{voz} = t_s \cdot v_d \tag{6.8}$$

Tento postup je sice na první pohled logický, ale bohužel není správný. V tomto případě je důležité si uvědomit, jak přesně funguje vysílání radarového signálu do prostoru. Stejně tak jako většina snímacích zařízení, má i radar svůj úhel rozsahu, ve kterém vlny vysílá. To má za následek fakt, že vysílaný kužel vln, který putuje od radaru je ve skutečnosti rozprostřen na silnici v řádech několika metrů. Vzhledem k tomu, že nejvhodnější metodou měření je vysílaní vln z výšky směrem k vozovce, dochází k tomu, že rozsah radaru negativně ovlivní toto měření. Pokud si představíme co konkrétně v této situaci nastává, zjistíme, že při vstupu vozidla do radarového rozsahu jsou první paprsky odráženy od přední části kapoty. Dále musí vozidlo projet skrz celý rozsah radaru a jako poslední část, kterou radar zachytí je zadní část vozidla. Pokud bychom tedy chtěli použít vzorec 6.8 k výpočtu délky, bylo by potřeba, aby radar měl co nejmenší úhel rozsahu. To však u radarových modulů není běžné a obvykle se tyto rozsahy pohybují okolo 12 - 30 stupňů v závislosti na orientaci antény.



Obrázek 6.3: Ukázka reálné situace měření radarovým modulem z výšky. Na schématu lze pozorovat odchylku ΔL, kterou je potřeba od délky vozidla odečíst.

Z výše uvedeného vyplývá, že potřebuje vzorec poupravit tak, abychom eliminovali přebytečnou délku, která vzniká jako nežádoucí efekt způsobený vysílacím úhlem radaru. V našem případě, kdy měříme provoz z výšky, musíme znát 2 věci k tomu, abychom dopočítali délku záběru vysílaného signálu na vozovce. Jedná se o výšku, ve které je radar umístěn nad silnicí a druhou informací je úhel, pod nimž je radar nakloněn oproti rovině silnice. Tyto dvě hodnoty společně s úhlem záběru radaru lze společně s běžnými goniometrickými funkcemi použít k výpočtu kýžené odchylky. Jedná se totiž o běžnou trojúhelníkovou nerovnost. Vzorec pro výpočet odchylky vypadá následovně:

$$\Delta L = h \cdot (\tan(\beta) - \tan(\gamma)) \tag{6.9}$$

Jakmile máme vypočítanou odchylku, můžeme přejít k samotnému výpočtu správné délky vozidla. Budeme vycházet z původního vzorce a odečteme od něj odchylku.

$$L_{real} = ((v_d \cdot t_s) - \Delta L)) \cdot C_k, \tag{6.10}$$

kde L_{real} vyjadřuje skutečnou délku vozidla, v_d je rychlost vypočítaná na základě Dopplerova jevu, t_s je doba signálu, ΔL je vypočtená odchylka pomocí vzorce 6.9 a C_k je kalibrační konstanta.

Ze vzorce 6.10 je patrné, že výpočet délky závisí na celé řadě jiných veličin, a proto je našim cílem změřit či vypočítat tyto hodnoty co možná nejpřesněji tak, aby chyba délky vozidla byla co nejmenší. Velký důraz je kladen hlavně na dobu signálu, která je spojená s korektní detekcí vozidla v radarovém signálu. Neméně důležitou hodnotou je i extrahovaná rychlost, avšak zde je její výpočet snazší, jelikož je mnohdy ve spektru jen úzká část dominantních frekvencí. Ačkoli se to možná nezdá, důležitou součástí je i tvar vozidla. Konkrétně jeho přední a zadní část. První a poslední kontakt radarového signálu a vozidla závisí právě na těchto částech a nákladní vozidla mohou signál odrážet déle právě kvůli své vyšší výšce zadní části.

Spektrální hustota výkonu

Další z možných příznakových vektorů, které můžeme z radarového signálu vytáhnout, je spektrální výkon. Již při pohledu na spektrogram je jasné, že jednotlivé kategorie vozidel mají určitá specifika a lze je rozlišit pouhým okem. Vysílaný radarový signál se odráží od různých částí vozidel v závislosti na odrazové ploše a taky na úhlu, pod kterým signál dopadá. Tento poznatek můžeme využít, jelikož třídy aut, které jsme si zvolili mají většinou různé tvary a různě velkou plochu. Rozeznat od sebe kamion a osobní automobil lze v tomto ohledu s velkou přesností.

Pokud vezmeme v potaz výše uvedené, lze ze spektrogramu pomocí běžných operací získat jednorozměrný příznakový vektor. K tomu, abychom snadno oddělili vozidla s nízkou a vysokou odrazivostí, použijeme hodnotu *spektrální hustoty výkonu* [18]. Ta se běžně vypočte jako jako:

$$G_{DFT}(k\Delta f) = \frac{1}{N}|X[k]^2|. \tag{6.11}$$

Spektrální hustotu vypočteme pomocí vestavěných funkcí knihovny `numpy`. Konkrétně použijeme funkci `sum(x ** 2)`, která spočte sumu druhých mocnin výstupu Fourierovy transformace. Tento výpočet provedeme pro každý rámec signálu a na závěr opět funkcí `sum` spočteme celkový výkon daného spektra v čase. Hodnoty Fourierovy transformace v daném rámci nemůžeme jakkoli normalizovat, jelikož bychom přišli o důležité informace intenzity

odraženého signálu. Jediná normalizace, kterou musíme udělat je v časové složce. Jelikož projíždějící vozidla mají různou rychlost, projíždějí v radarovém zorném poli různou dobu. Abychom předešli chybám způsobeným tím, že pomalá vozidla budou mít vyšší výkon než vozidla rychlá, je nutné vydělit získaný výkon počtem rámců daného signálu.

Spektrální výkon napříč rámci

Doposud jsme ze signálu získávali veličiny, které se nacházejí pouze v jednorozměrném prostoru. V takovémto případě jakákoli nepřesnost výpočtu či zanesení chyby vlivem šumu, může zpusobit chybnou klasifikaci. Je to dáno tím, že vozidlo je při jednorozměrném příznakovém prostoru pouze bod na ose v jedné dimenzi. Hranice mezi zvolenými kategoriemi tvoří opět pouze body.

Abychom předešli výše uvedeným chybám, je potřeba ze signálu vyextrahovat příznakové vektory z vícerozměrného prostoru. Příkladem takovéhoto vektoru může opět být spektrální hustota výkonu. Od předešlého příznakového vektoru se tento liší v tom, že využijeme informace z celého průběhu signálu. Jednotlivé kategorie vozidel totiž mají průběh spektrálního výkonu výrazně odlišný. Zatímco u osobních automobilů pozorujeme téměř konstantní průběh s mírným vzestupem na konci signálu, u větších vozidel je tento vzestup daleko razantnější a déle trvající.

Spektrogram, ze kterého budeme brát užitečné informace, si můžeme představit jako dvourozměrné pole hodnot, kde jedna dimenze určuje čas v rámcích a druhá dimenze určuje míru zastoupení jednotlivých frekvencí. Spektrální výkon vypočteme stejně jako v předchozím případě, s tím rozdílem, že nadefinujeme, v jaké ose chceme sumu počítat. Opět nám pomůže knihovna `numpy` a funkce `sum(x ** 2, axis=1)`. Výsledkem této funkce je jednorozměrné pole spektrálního výkonu pro každý rámec.

I zde se opět setkáváme s problémem různého počtu rámců, který závisí na rychlosti vozidla. U předchozího příkladu bylo možné normalizovat hodnotu prostým vydělením počtem rámců. V tomto případě se tato metoda nedá využít, jelikož chceme zachovat informace vícerozměrného vektoru. Z radarových dat, které máme k dispozici lze vypozorovat, že množství rámců na jeden průjezd vozidla je zhruba 40 až 70. Tyto hodnoty byly odvozeny při použití rámců s pevnou délkou 1024 hodnot a překryvem 512 vzorků. Z výše uvedeného lze určit, že je potřeba normalizovat příznakový vektor přibližně na 50 hodnot. Nemůžeme však délku vektoru zmenšit, popřípadě zvětšit na tuto hodnotu prostým prodloužení či naopak odříznutím zbytku pole. Pro korektní zachování průběhu je potřeba provést změnu velikosti interpolací. To provedeme tak, že si vypočítáme poměr mezi aktuální a finální délkou vektoru. Dalším krokem pomocí interpolace vytvořit funkci `F`, jež následně pomocí zmíněného poměru transformuje vstupní pole na požadovaný rozsah. Interpolaci lze provést funkcí `interp1d(x, y)` z knihovny `scipy`. Výsledkem takto provedené interpolace je vektor hodnot s konstantní požadovanou délkou a se zachovaným průběhem.

6.4 Grafické uživatelské rozhraní

V této diplomové práci bylo vytvořeno grafické uživatelské rozhraní tak, aby bylo možné výsledky prezentovat v takové podobě, aby byl uživatel schopen zjistit, na jakém principu funguje zpracování radarového signálu spolu s následnou klasifikací. Grafické rozhraní bylo navrženo tak, aby bylo co možná nejjednodušší na ovládání. S tím je spojena jeho minimalistická podoba tak, aby uživatel byl schopen intuitivně ovládat aplikaci bez jakékoli předchozí zkušenosti.

Jelikož je celá aplikace vytvořená v jazyce Python, tak bylo několik možností jak vytvářet grafické uživatelské rozhraní. Existuje totiž celá řada frameworků pro tyto účely. Nejznámější takovýto framework je Qt. V tomto projektu je konkrétně využita jeho portace pro jazyk Python, která se nazývá PyQt ve verzi 5.

Existují dva postupy, jak přistupovat k návrhu grafického uživatelského rozhraní. Prvním z nich je kompletní vytváření vzhledu aplikace přímo pomocí API, jenž poskytuje framework. Tento přístup je zdlouhavý, jelikož musíme manuálně vytvářet objekty grafických prvků a ty použít v celkovém rozhraní. Druhým přístupem, jak vytvářet rozhraní, je pomocí aplikace Qt Designer, jež umožní definovat design pomocí předem daných grafických prvků, které pouze skládáme dohromady a přiřazujeme jim určité vlastnosti. Právě tento přístup byl využit v této práci, jelikož primárním cílem grafické aplikace je pouze poskytnout demo k hlavní části práce, jež se zabývá problematikou radarového signálu a jeho zpracováním. Výstupem aplikace Qt Designer je XML soubor popisující hierarchii grafických prvků a jejich vlastností. Pro správné fungovaní tohoto souboru v Pythonu je potřeba použít utilitu z balíku Qt nazývající se pyuic5, která převede rozhraní z formátu XML na kód pracující s objekty přímo v Pythonu.

Po spuštění aplikace se uživateli objeví okno, které je současně jediným oknem celé aplikace. Prvotní start může nějakou dobu trvat, jelikož se v této době provádí načítaní defaultního signálu s čímž je spojený veškerý preprocesing. Aby uživatel věděl, co se v této době děje, je součástí rozhraní také informační stavový řádek naznačující celkový postup preprocessingu. Jakmile je preporcessing dokončen naplní se okno všemi informacemi. Hlavní okno aplikace je rozděleno do 3 logických celků. Prvním z nich je seznam nacházející se na pravé části okna. Tento seznam obsahuje položky značící průjezdy jednotlivých vozidel. Vozidlo v seznamu je reprezentováno časem průjezdu, který je získán z epochy signálu. Druhou dominantní částí aplikace je část obsahující grafy. Jedná se o 3 grafy, které nesou informace o signálu. První z nich ukazuje podobu signálu, který získáváme přímo z radaru. Druhý z nich ukazuje spektrum signálu měnícího se v čase. Jedná se tedy o spektrogram zobrazující dominantní frekvence. Poslední z nich je graf zobrazující výstup Fourierovy transformace pro konkrétní rámec. Abychom si mohli zobrazit spektrum napříč rámci, je pod grafy připraven posuvník, kterým si můžeme zvolit, který rámec si přejeme zobrazit. Tímto posuvníkem zároveň zvýrazníme rámec uvnitř celého signálu v prvním grafu. Další vertikální posuvník se nachází na pravo od grafu. Tímto posuvníkem můžeme omezovat maximální frekvenci a tím měnit podobu jak spektra rámce, tak i celkového spektrogramu. Tato funkce je velice užitečná chceme-lo zkoumat frekvence v dolní části spektra.

Poslední částí grafické aplikace je část zobrazující meta informace a klasifikační data k jednotlivým průjezdům. Tato část se nachází vespod okna a je rozdělena na dvě pod-části. V první pod-části se nachází informace o počtu rámců, délce signálu, extrahované rychlosti, vypočítané délce a klasifikační třídě. V druhé pod-části se nachází snímek z videa, které bylo pořizováno spolu s radarovou nahrávkou. Tento snímek slouží pro vizuální zhodnocení, zda predikovaná třída opravdu odpovídá typu vozidla, které bylo zaznamenáno.

Zakomponování grafů

Součástí výše uvedeného návrhu je i zobrazení 3 grafů. Nejlepším nástrojem v Pythonu pro práci s vizualizací dat ve formě grafu je knihovna *Matplotlib* [4]. Tato knihovna však není nativním objektem frameworku QT, kterým je tvořen zbytek grafického rozhraní. Právě z tohoto důvodu bylo potřeba nalézt určitý kompromis tak, aby bylo možné grafy v QT rozhraní zobrazit.

Základním stavebním prvkem v QT frameworku je takzvaný *Widget*. Abychom mohli zobrazit grafy v tomto prvku je potřeba nahradit výchozí canvas pro kreslení grafu právě tímto widgetem. To lze provést doinstalováním alternativního backendového vykreslovacího jádra knihovny matplotlib. Konkrétně se jedná o třídu `FigureCanvas`, která se pro QT tváří jako klasický widget a naopak pro knihovnu matplotlib jako prvek známý jako *Figure*. Tento prvek lze dále dělit na podgrafy tak, abychom mohli zobrazit všechny 3 zmíněné grafy.

Videozáznam

Kromě grafu je dalším grafickým prvkem snímek z videozáznamu. Snímek je extrahován z videa, které bylo pořizeno společně s radarovým záznamem. Volba snímku závisela na časovém údaji začátku průjezdu vozidla. Důležitým prvkem je to, že videozáznam musí být synchronizován společně s radarem, aby bylo vůbec možné snímek pořídit v ten pravý čas. Experimentálně bylo zjištěno, že snímek je potřeba získat těsně předtím než je vozidlo detekováno radarem. Negativním prvkem při získávání snímku je to, že videozáznam je obvykle komprimován a je složen z klíčových snímků a ostatních, které se určují na základě předcházejících nebo následujících snímků. Proto nelze přímo extrahovat snímek v přesném čase, ale je potřeba nalézt nejbližší klíčový snímek. V našem případě tato chyba nebyla příliš zásadní, jelikož zpozdění mezi reálným a klíčovým snímkem není natolik velké, aby mezitím vozidlo zmizlo ze záběru.

Snímek byl extrahován pomocí knihovny *Imageio*, která je založena na sadě nástrojů *FFmpeg*[7]. Následně byl snímek zmenšen na požadovanou velikost pomocí knihovny *Scipy* a poté převeden na objekt typu `QImage`, který lze zobrazit na grafickém prvku `label`.

6.5 Klasifikace

Závěrečnou etapu implementované aplikace tvoří samotná klasifikace vozidel. Jak již bylo řečeno v kapitole návrh, pro klasifikaci je využito dvou různých metod. Jedná se o klasifikační algoritmy Support Vector Machine a AdaBoost. Obě tyto metody jsou založeny na principu učení s učitelem a tudíž jim musíme předat příznakové vektory společně s označením o jakou kategorii vozidla se jedná. Tato část práce byla vcelku pracná, jelikož bylo nutné označit všechny průjezdy v signálu ručně podle videozáznamu, který byl k dispozici.

Pro klasifikaci byla vybrána knihovna *Sklearn*, která kromě klasifikačních algoritmů obsahuje další metody pro strojové učení. Metody této knihovny byly zabaleny do vlastních tříd, jenž přidávaly další funkčnost. Metody byly reprezentovány třídami `SvmClassifier` a `AdaBoostClassifier`. Obě tyto třídy měly možnost uložit natrénovaný klasifikátor do souboru metodou `save_to_file()`. Tuto možnost lze využít v případě, kdy chceme klasifikátor použít pro klasifikaci, abychom jej nemuseli po každém spuštění přetrénovávat. Načíst takto natrénovaný klasifikátor lze pomocí metody `load_from_file()`.

Jakmile máme správně oanotována data, můžeme přejít k samotnému trénování klasifikátoru. Jednotlivé průjezdy si musíme rozdělit na trénovací a testovací množinu. Trénovací proces zahájíme metodou `train()`, již předložíme pole příznakových vektorů a pole anotací. V druhém kroku využijeme testovací množinu dat, abychom získali celkovou úspěšnost klasifikace. To provedeme metodou `test_classifier()`, již opět předložíme pole příznakových vektorů a jejich anotací. V tomto kroku je pro každý vektor volána metoda `predict()`, která vrátí identifikátor třídy, do které bylo vozidlo na základě jeho příznakového vektoru zařazeno. V klasifikačních metodách knihovny Sklearn je možno zapnout takzvanou měkkou klasifikaci (soft classification), která nevrací při predikci jen jednu třídu, ale vrátí pole

pravděpodobností pro každou třídu. Tímto způsobem jsme schopní zjistit, jak moc si jsou třídy blízké v příznakovém prostoru.

Kapitola 7

Vyhodnocení a testování

7.1 Měření radarových dat

V prvotní fázi vývoje aplikaci byla využita radarová data, které byla pořízena na přípravných workshopech. Při tomto měření byl použit radarový modul K-MC1, který snímal pohyb ruky před radarem. Takto naměřená data byla poté uložena do sady souborů typu CSV. Tato data sice neobsahovala reálná data ze silničního provozu, avšak jejich obsah byl plně dostačující pro vývoj první části knihovny pro práci se signálem.

V pozdějších fázích kdy bylo potřeba extrahovat z radarového signálu konkrétní data pro průjezd vozidel, byla tato datová sada nahrazena reálnými radarovými daty ze silničního provozu. Tato radarová nahrávka byla pořízena jako součástí jiného projektu, avšak obsahovala vše potřebné pro účely této práce. Tato datová sada byla uložena v binárním formátu RRC, a tudíž pro její načítání musel byt použit načítací skript `RadarReader`, jenž byl poskytnut vedoucím této práce. Měření probíhalo na silnici Hradecká v Brně Řečkovicích způsobem kdy byl v záběru vždy pouze jeden pruh silnice. Záznam byl pořízen tak, že radarový modul byl umístěn na mostě nad vozovkou a tedy v ideálním místě pro získaní přesných dat. Tím, že radarový modul snímal vozidla z výšky bylo nutné provézt korekci dat pomocí faktoru kosinova úhlu.

Radarový modu	K-MC4
Délka nahrávky	1:08 hod
Výška nad vozovkou	7.8m
Úhel sklonu radaru	$-45°$
Vzorkovací frekvence	50 kHz
Frekvence radaru	24.125 GHz

Společně s radarovým modulem byly při snímání použity lidary na okrajích vozovky, které snímaly projíždějící vozidla z boční strany. Pomocí nich byly získaná přesná data o rychlosti a délce vozidla. Tyto metadata byly uloženy do formátu JSON, a byly využity jako pomocný nástroj při klasifikaci. K těmto datům bylo přidáno pole `category`, které obsahovalo ručně vyplněnou klasifikační kategorii.

```
[
    {
        "a": -0.4,
        "category": 1,
        "lane": 1.0,
        "epoch11": 1458735673575782.0,
        "v21": 25.209,
        "L": 4.526,
        "epoch21": 1458735674677540.0,
        "id": 9.0,
        "vAvg": 25.429
    }
]
```


Listing 2: Ukázka metadat projíždějícího vozidla, jenž byla použita k anotacím jednotlivých průjezdů.

Další důležitou částí, jež byla zaznamenávána při pořizování radarových dat byla videonahrávka. Video zachycující provoz sloužilo jako jediná možnost jak vizuálně zařadit vozidlo do správné kategorie a přiřadit jí k radarovým úsekům.

7.2 Výsledky klasifikace

Získána radarová data, která byla popsána v předchozí podkapitole byla použita pro vytvoření klasifikátorů. V kapitole návrhu jsme si dali za cíl porovnat dva typy klasifikačních algoritmů. Jednalo se o algoritmy Support Vector Machine a AdaBoost. U metody Support Vector Machine byly použity i různá jádra, tak abychom zjistili která jsou vhodná právě pro příznaky extrahované z radarového signálu. Konkrétně se jednalo o obecné lineární SVM, dále pak jádro rbg a sigmoid.

Z průjezdů jednotlivých vozidel byly získány všechny výše uvedené klasifikační příznakové vektory. Tyto vektory byly použity jak samostatně, tak i v kombinaci s jinými. Získali jsme tedy jednorozměrné vektory i vektory ve 2D prostoru. Výjimku tvořil příznakový vektor spektrálního výkonu rámců, který se nacházel v prostoru s 50 dimenzemi.

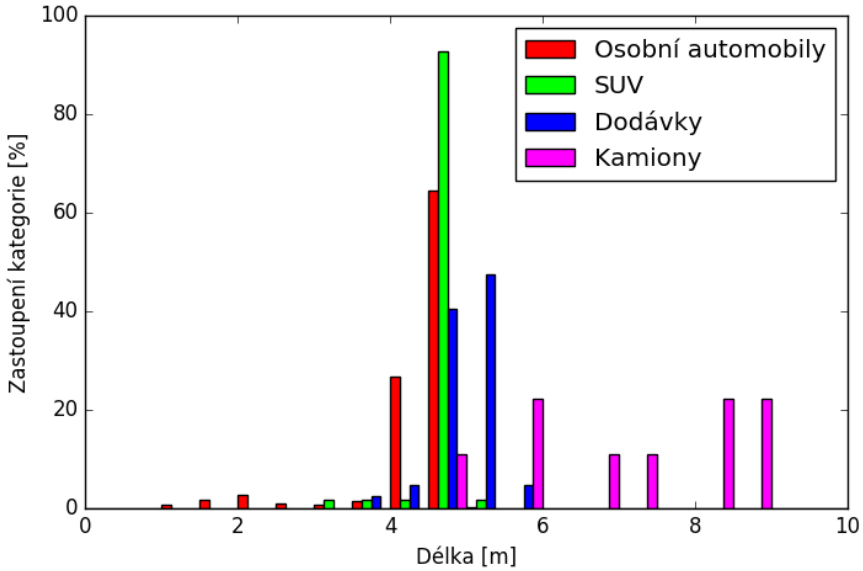
V získané datové sadě nejsou jednotlivé kategorie rovnoměrně zastoupeny. Je to dáno faktem, že v reálném provozu na silnicích převládají určité kategorie na jinými. Tato datová sada byla rozdělena v poměru 50:50 na trénovací data a na testovací data. Datová sada nebyla rozdělena náhodně, nýbrž tak, aby jak v trénovacích tak testovacích datech byly v každé kategorii přibližně stejný počet vozidel. Pomocí trénovací sady a jednotlivých příznakových vektorů byly natrénovány výše uvedené klasifikátory. Následně pomocí testovací sady byly postupně klasifikátorům předloženy jednotlivé průjezdy a jejich vektory, čímž jsme získali odhadnutou klasifikační třídu. Tato třída byla následně porovnána s reálnou třídou a na základě porovnání byla vypočtena úspěšnost klasifikace.

Výsledky klasifikace lze vidět v tabulce 7.1. Jak lze vidět, nejúspěšnějším klasifikátorem je AdaBoost s využitím příznakového vektoru, kdy je spojena délka vozidla s odrazovou plochou. Klasifikátor SVM s lineárním jádrem má u třech vektorů pomlčku. Je to z důvodu, že klasifikaci nedokončil v konečném čase a tudíž pro něj nešla získat úspěšnost.

	SVM			AdaBoost
Kernel	lineární	rbf	sigmoid	
Délka	82.47	82.47	78.3	83.2
Odrazová plocha	-	77.8	77.8	82.98
Délka + odrazová plocha	-	78.4	78.9	86.08
Výkon rámců	-	76.9	77.4	83.2

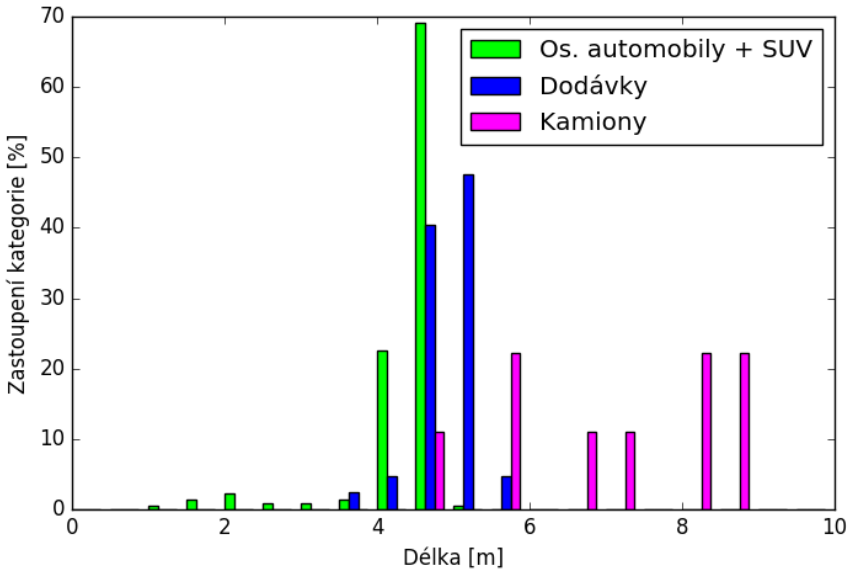
Tabulka 7.1: Úspěšnost klasifikace vyjádřena v procentech s použitím metody SVM s různými kernely a metodou AdaBoost.

Nejlepší výsledky v podobě 86 % úspěšnosti sice nejsou nízké, avšak představy byly trochu vyšší. Při detailnějším pohledu na klasifikační vektory lze pozorovat, že je zde určitý překryv klasifikačních vektorů mezi osobními automobily a SUV. Je to dáno tím, že rozdíl délky a odražené plochy ne vždy býva rozdílovým prvkem a některé osobní automobily mohou svou délkou překonávat SUV. Detailnější rozbor délky napříč kategoriemi lze pozorovat na grafu 7.1



Obrázek 7.1: Graf rozložení délky vozidel napříč zvolenými kategoriemi.

Pokud bychom se rozhodli tyto dvě kategorie sloučit a klasifikovat vozidla na malá, střední a velká překryv vektorů by už nebyl tak velký a celková klasifikační úspěšnost by mohla stoupnout. Rozdíl po sloučení výše uvedených kategorií lze pozorovat na grafu délky 7.2



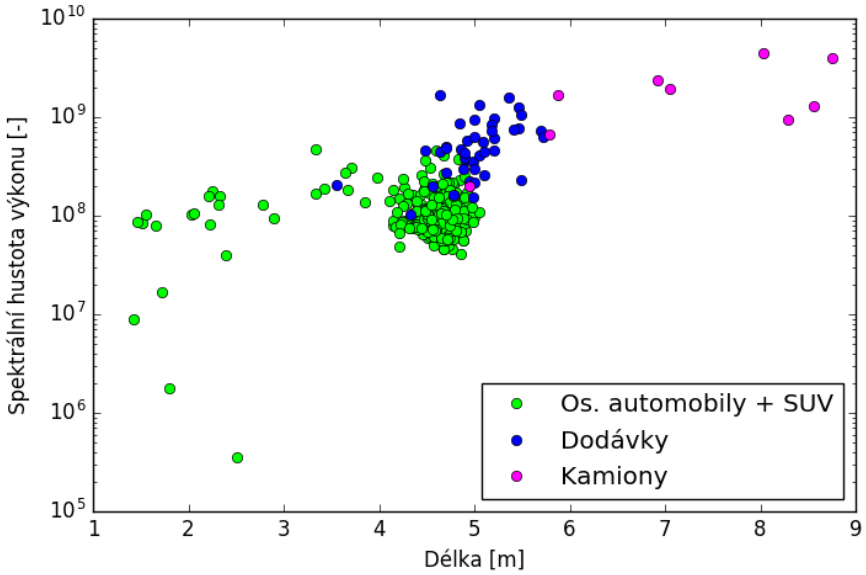
Obrázek 7.2: Graf rozložení délky vozidel napříč zvolenými kategoriemi po sloučení osobních automobilů s SUV.

Po sloučení těchto kategorií byly znovu přetrénovány klasifikátory a opět provedeno testování. Výsledky úspěšnosti klasifikace se viditelně zvýšily na přijatelnou mez. V tabulce 7.2 lze vidět, že je opět nejlepším klasifikátorem AdaBoost při použití příznakového vektoru délky a odrazové plochy. Klasifikační úspěšnost překonala hranici 97%, což je pro reálný provoz na silnici přijatelná hodnota. Zbývající příznakové vektory se pohybují zhruba v podobné úspěšnosti s tím že nejlepší z nich můžeme označit výkon rámců.

	SVM			AdaBoost
Kernel	lineární	rbf	sigmoid	
Délka	82.47	91.3	87.11	94.5
Odrazová plocha	-	88.2	88.6	93.4
Délka + odrazová plocha	-	88.2	88.5	97.1
Výkon rámců	-	87.1	89.3	94.5

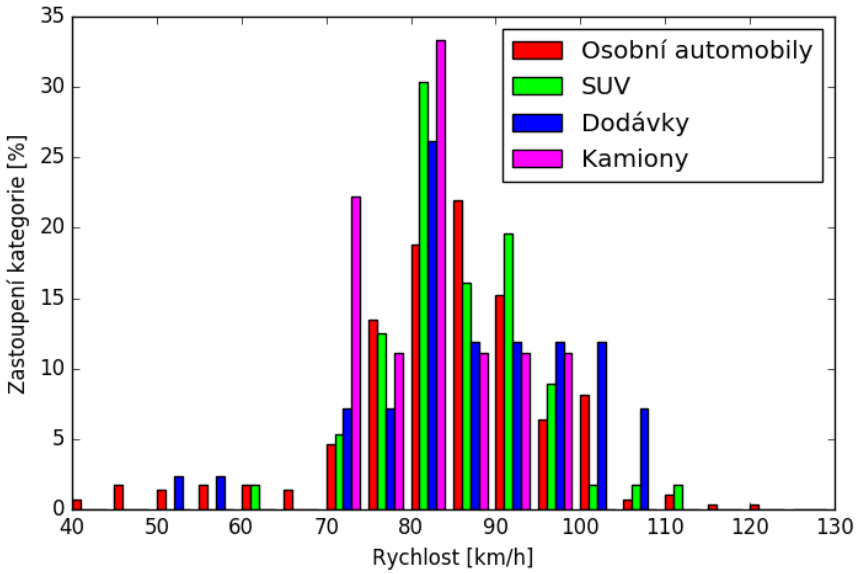
Tabulka 7.2: Vyjádření procentuální úspěšnosti klasifikace po sloučení tříd osobních automobilů a SUV.

Na grafu 7.3 lze jasně pozorovat proč je kombinace délky a odrazové plochy nejlepší volbou pro klasifikační vektor. Jednotlivé třídy jsou jasně odděleny v prostoru a tudíž pro ně lze vytvořit úspěšný klasifikátor.



Obrázek 7.3: Graf zobrazující jednotlivé kategorie vozidel v příznakovém prostoru při použití kombinace délky a odrazové plochy jako příznakových vektorů.

Na závěr tohoto porovnání si můžeme zobrazit statisticky užitečné zobrazení dat. Graf 7.4 zobrazuje jak rychle jela jednotlivá vozidla, popřípadě jednotlivé kategorie vozidel. Graf je založen na vypočtené rychlosti z radarových dat. Jelikož jsme měli k dispozici skutečnou rychlost změřenou pomocí lidaru, mohli jsme zjistit o kolik se liší námi extrahovaná hodnota rychlosti od skutečné. Extahovaná rychlost byla v průměru o 1% nižší než skutečná. Tato hodnota lze brát jako přijatelnou jelikož při povolené rychlosti 90 $\frac{km}{h}$, je tato odchylka méně než jeden kilometr za hodinu.



Obrázek 7.4: Histogram zobrazující rychlost jednotlivých vozidel zachycených radarem.

7.3 Výkonnostní testy

Tato podkapitola se bude věnovat tomu, jak rychle dokáže výsledná aplikace data zpracovávat a klasifikovat. Pro níže uvedené výsledky byla použita datová sada z nahrávky, která trvala hodinu a obsahovala přes 3GB dat. Aplikaci lze rozdělit na celkem tři části podle její zaměření. V první části je načítán radarový signál ze souboru typu RRC a ten je rozdělen na úseky projíždějících vozidel. Druhá část se zabývá metodami pro zpracování signálu, tak abychom mohli extrahovat příznakové vektory pro klasifikaci. A poslední část slouží k samotnému trénování a testování klasifikátorů. Úseky kódu z těchto částí byly obaleny do funkcí, které získávají přesný čas od epochy. Konkrétně se jednalo o funkci `time.time()`. Hodnota získaná na počátku bloku byla odečtena od hodnoty získané na konci a tím byla změřená doba jakou strávil proces vykonáváním této části. Postupně byly změřeny všechny sekce a výsledky je možno vidět v tabulce 7.3

Část aplikace	čas [s]
Načátání dat	35.5
Zpracování signálu	4.3
Trénování klasifikátoru	0.09
Testování klasifikátoru	1.26

Tabulka 7.3: Časová náročnost jednotlivých částí výsledné aplikace.

Z grafu lze pozorovat, že převážnou většinu času v podobě 86.6% stráví proces načítáním dat ze souboru. Je to dáno hlavně tím, že se jedná o hodinovou nahrávku a mezi průjezdy vozidel jsou určité rozestupu, které musí aplikace přeskakovat. Druhou dominantní částí je samotné zpracování signálu. Tato část je pochopitelná, jelikož obsahuje řadu netriviálních metod pro zpracování signálu. Pokud bychom chtěli tuto část zrychlit, mohli bychom pod-

vzorkovat získaná data, čímž bychom ale mohli přijít o důležité informace. Poslední části je část klasifikace a jak lze vidět jedná se o nejméně náročnou část celé aplikace. Samozřejmě by se tato doba prodloužila s použitím větších datových sad. Podrobnější přehled o tom, jak dlouho trvaly klasifikační metody lze vidět v tabulce 7.4.

	SVM			AdaBoost
Kernel	lineární	rbf	sigmoid	
Délka	2.65 / 9.89	1.58 / 15.57	0.82 / 13.09	62.68 / 579.4
Odrazová plocha	-	3.34 / 17.05	0.77 / 9.74	58.17 / 616
Délka + odrazová plocha	-	3.76 / 18.9	0.71 / 9.55	70.22 / 771
Výkon rámců	-	8.08 / 17.91	2.33 / 11.23	95.0 / 578.8

Tabulka 7.4: Časová náročnost jednotlivých klasifikačních metod. První číslo označuje dobu trénovaní a druhé dobu testování. Čas je zobrazen v milisekundách.

Je patrné, že doba trénování je u všech algoritmu kratší než doba testování a lze pozorovat, že nejúspěšnější klasifikátor AdaBoost je zároveň nejpomalejší. Ačkoli je tento algoritmus nejpomalejší, nejedná se o přehnaně dlouhou dobu na to aby algoritmus nemohl byt použit pro reálné využit.

Kapitola 8

Závěr

Tato práce se zabývá problematikou rozpoznávání a klasifikací automobilové dopravy a to tak, že jako zdroj informací ze silnic využívá radar. Radarových modulů je celá řada v závislosti na jejich principu fungování a manipulaci se signálem. Hlavní typy těchto radarů jsou v úvodní části této práce rozebrány a detailně popsány. Konkrétně se jedná o radary založené na Dopplerově jevu, který mění frekvenci v závislosti na relativním pohybu mezi snímačem a zkoumaným objektem. Využity jsou zde radary s kontinuální vlnou, jelikož právě ty byly k dispozici k zapůjčení.

Další neodmyslitelnou částí, která je v této práci podrobně popsána, je část zabývající se zpracováním signálu. Proto, abychom mohli extrahovat užitečná data, pomocí kterých budeme schopni automobily klasifikovat, musíme signál podrobit řadě procesů a metod, které přispívají k tomu, aby na konci tohoto zpracovávajícího řetězce byla užitečná data odfiltrovaná od všech zbytečných rušení.

Na závěr teoretické části této práce je popsána samotná problematika strojového učení, a to konkrétně její podkategorie věnovaná klasifikačním problémům. Je zde popsáno základní dělení klasifikace společně s konkrétními algoritmy, které jsou v dalších částech této práce použity.

Na počátku jsme si vytyčili cíle, které jsme chtěli touto prací dosáhnout. Jednalo se o návrh klasifikačních kategorií vozidel, do kterých budeme řadit jednotlivá vozidla na základě jejich nasnímaného radarového signálu. Proto, abychom splnili tyto cíle byla vytvořena aplikace skládající se ze tří části. První z nich je univerzální knihovna pro práci s radarovým signálem, pomocí které jsme byli schopni extrahovat užitečná data pro klasifikaci. Tato knihovna obsahuje celou řadu metod pro zpracování signálu, počínaje dělení signálu na rámce, až po získání spektra. Druhá velice důležitá část je samotná klasifikace. Byly zde využity algoritmy strojového učení zaměřené na klasifikaci pomoci učitele. Konkretně se jednalo o algoritmy SVM a AdaBoost. Poslední části aplikace je část prezentující výsledky. Tato část je implementována jako grafické uživatelské rozhraní, zobrazující obsah signálu, spektra a metadat průjezdu vozidel.

Jak se s postupem času ukázalo, radarový signál obsahuje celou řadu užitečných informací pro klasifikaci. Pomocí těchto dat byly natrénovány klasifikátory a určena jejich úspěšnost. Ukázalo se, že pro původně navržené klasifikační třídy nejlepší z klasifikátorů dosahoval úspěšnosti 86%. Vyšší úspěšnosti bylo dosaženo při sloučení kategorie osobních vozidel s SUV. Radarový signál totiž při snímání čelní strany vozidla neobsahuje dostatečné množství informací pro úspěšné rozdělení těchto kategorií. S tímto sloučením jsme dosáhli úspěšnosti 97%, což můžeme označit jako kvalitní výsledek. Pokud bychom chtěli vylepšit klasifikační úspěšnost pro původní kategorie, museli bychom například přidat radarový mo-

dul, který by snímal boční profil vozidel. Ukázkovým příkladem využití této práce by byla kontrola rychlostních limitů v závislosti na kategorii vozidla. Vetší vozidla mají totiž běžně jiné rychlostní omezení než ostatní a právě z radarového modulu jsme schopni extrahovat kombinaci klasifikační třídy a rychlosti daného vozidla.

Možné pokračování této práce by bylo především převedení aplikace do reálného provozu. Pro toto provedení by bylo potřeba urychlit algoritmy pro zpracování signálu tak, aby byl signál zpracováván v reálném čase. To by se dalo docílit převodem na specializovaný hardware například v podobě FPGA čipu. Dalším možným rozšířením by mohlo být v podobě použití více radarových modulů, které by snímaly silnici z různých úhlů a tím bychom mohli dosáhnout lepší úspěšnosti rozpoznání osobních automobilů od SUV.

Ověřili jsme si tedy, že radarový modul je schopen s velkou úspěšností klasifikovat provoz na silnicích, a tudíž je schopen konkurovat běžně využívaným metodám založeným na rozpoznávání vozidel z videozáznamu. Oproti těmto metodám má radar řadu výhod i nevýhod. Radar totiž nedokáže získat vizuální vlastnosti vozidla a například extrahovat SPZ vozidla. Na druhou stranu radar může pracovat i za nepříznivých světelných podmínek, kdy ostatní metody selhávají. V ideálním případě by bylo možno zkombinovat obě technologie pro získání většího množství informací.

Literatura

[1] Alpaydin, E.: *Introduction to Machine Learning*. London, England: The MIT Press, Cambridge, druhé vydání, 2010, 537 s., iSBN 978-0-262-01243-0.

[2] Bishop, C. M.: *Pattern Recognition and Machine Learning*. New York: Springer, první vydání, 2006, 738 s., iSBN 0-387-31073-8;ISBN 978-0387-31073-2.

[3] Cournapeau, D.: Scikit-learn, Machine Learning in Python. online, [cit. 8. 5. 2017]. URL http://scikit-learn.org

[4] Hunter, J. D.: Matplotlib. online, [cit. 8. 5. 2017]. URL https://matplotlib.org

[5] InnoSenT: Radar Sensing and Detection of Moving and Stationary Objects. online, [cit. 3. 1. 2017]. URL http://www.innosent.de/fileadmin/media/dokumente/Downloads/Application_Note_I_-_web.pdf

[6] Marsland, S.: *Machine Learning, An Algorithmic Perspective*. New York: CPC Press, Taylor & Francis Group, první vydání, 2009, 383 s., iSBN 978-1-4200-6718-7.

[7] Niedermayer, M.: FFmpeg. online, [cit. 1. 5. 2017]. URL https://ffmpeg.org/

[8] Oliphant, T.: NumPy. online, [cit. 8. 5. 2017]. URL http://www.numpy.org/

[9] Oliphant, T.; Peterson, P.; Jones, E.: SciPy. online, [cit. 8. 5. 2017]. URL https://www.scipy.org/

[10] Petrescu, F. I.: *A New Doppler Effect*. Norderstedt: Books on Demand, první vydání, 2012, 80 s., iSBN 978-3-8482-2990-1.

[11] Proakis, J. G.; Monalakis, D. G.: *Digital Signal Processing: Principles, Algorithms, and Applications*. New Jersey: Prenticel Hall, třetí vydání, 1996, 968 s., iSBN 0-13-373762-4.

[12] RFbeam Microwave GmbH: K-MC1 Radar Transceiver. online, [cit. 3. 1. 2017]. URL https://www.rfbeam.ch/files/products/15/downloads/Datasheet_K-MC1.pdf

[13] RFbeam Microwave GmbH: K-MC4 Monopulse Radar Transceiver. online, [cit. 3. 1. 2017].

URL
https://www.rfbeam.ch/files/products/18/downloads/Datasheet_K-MC4.pdf

[14] Skolnik, M.: *Radar Handbook*. New York: McGraw-Hill Professional, třetí vydání, 2008, 1328 s., iSBN 978-0071485470.

[15] The Qt Company: Qt. online, [cit. 8. 5. 2017].
URL http://doc.qt.io/qt-5/index.html

[16] Thibaux, R.: Kernel trick, Advanced Topics in Learning & Decision Making. online, [cit. 8. 1. 2017].
URL https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf

[17] Wu, X.; Kumar, V.; aj.: Top 10 algorithms in data mining. online, 2007, [cit. 18. 4. 2017].
URL http://www.cs.uvm.edu/~icdm/algorithms/10Algorithms-08.pdf

[18] Černocký, J.: Zpracování řečových signálů — studijní opora. online, [cit. 3. 1. 2017].
URL http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf

Příloha A

Obsah DVD

Součástí této práce je i přiložené DVD, které obsahuje následující části:

- `VehicleClassification` - zdrojové kódy klasifikátoru a grafické aplikace
- `Radar` - složka obsahující radarová data
  - `RRC` - radarová data ve formátu rrc
  - `CSV` - radarová data ve formátu csv
  - `Laser` - pomocná radarová metadata
  - `Images` - snímky vozidel
- `Latex src` - zdrojové soubory této práce
- `Text` - výsledné pdf soubory této práce ve dvou verzích (verze určená pro tisk a verze s hypertextovými odkazy)
- `Libs` - knihovny pro spuštění aplikace